

Advanced Methods and Extensions for Kernel-based Object Tracking



Quang Anh Nguyen

A thesis submitted for the degree of
Doctor of Philosophy
of The Australian National University
February 2010

Declaration

This thesis is entirely my own work except where due reference is made in the text. It contains no material which has been submitted for a previous degree.

A handwritten signature in black ink, appearing to read 'Quang Anh Nguyen', with a long horizontal flourish extending to the right.

Quang Anh Nguyen

To my mother Lan, my father Ha and my late grandfather Huy

To Unity

For your love, support and belief in me.

Abstract

In today's world, the rapid developments in computing technology have generated a great deal of interest in automated video analysis systems such as smart cars, video indexing services and advanced surveillance networks. Amongst those, object tracking research plays a pivotal role in providing an unobtrusive means to detect, track and alert the presence of the objects of interest in a given scene with little to no supervisor interaction. The application fields for object tracking can vary from smart vehicles, advanced surveillance networks to sport analysis and perceptual user interfaces. The diversity in its applications also gives rise to a number of different tracking algorithms tailored to suit the corresponding scenarios and constraints. Along this line, the kernel-based tracker has emerged as one of the benchmark tracking algorithms due to its real-time performance, robustness to noise and tracking accuracy.

In this thesis, we explore the possibility of further enhancing the original kernel-based tracker. We do this by firstly developing a probabilistic formulation for the mean-shift algorithm which, in turn, provides a means to estimate the target's severe transformations in size, shape and orientation. For changes in colour appearance due to poor lighting condition of the scene, we opt to combine multiple low complexity image features such as texture, contrast, brightness and colour to improve the tracking performance. To achieve this, we advocate the use of a graphical model to abstract the image features under study into a relational structure and, subsequently, make use of graph-spectral methods to combine them linearly and in a straight-forward manner. Furthermore, we also present an on-line updating method to adjust the tracking model to incorporate new changes in the image features during the course of tracking. To cope with the problem of object occlusion in a high density traffic area, we propose a geometric method to extend the mean-shift algorithm to a 3D setting by the use of multiple cameras with overlapped views of the scene. The methods presented in this thesis not only show significant performance improvements on real-world sequences over a number of benchmark algorithms, but also encompass high generalisation in the spatial and feature domains for future development purposes.

Keywords: Kernel-based Object Tracking, Mean-shift Optimisation Scheme, Feature Combination for Object Tracking, 3D Object Tracking.

Acknowledgments

I would like to thank my principal supervisor, Antonio Robles-Kelly, for his excellent support, guidance and patience during the course of my PhD.

Furthermore, I wish to express my profound gratitude to David Everitt for his generous support, and more importantly, his trust in me throughout the last six months of this thesis. I am extremely thankful to Richard Hartley for his invaluable inputs to the 3D work in Chapter 5. I am also indebted to Peter Christen for his administrative support during the submission and examination process of this thesis.

I would like to extend my gratitude to my colleagues at the Australian National University (ANU) and the National ICT Australia (NICTA) whom I have greatly enjoyed discussion and collaboration with, especially Cong Phuoc Huynh, Pat Khuwuthyakorn, Peng Wang (Norman), Surya Prakash, Tim Raupach and Zhouyu Fu. Special thanks must go to Cong for his thorough comments on the final draft of this thesis.

Many thanks to the ANU and NICTA for providing me with generous financial support and a state-of-the-art research environment.

This thesis would have been impossible without the care and support of my family. I would like to thank my parents, Lan Tran and Ha Nguyen, for their endless dedication, sacrifices and encouragement over the last 26 years. I also wish I could say thanks to my late grandfather Huy Nguyen for his silent support and care that I did not fully understand until many years after he passed away.

Finally, and most importantly, my deepest gratitude goes to my gorgeous partner, Unity Batenburg Nguyen, for her unconditional love, dedication and support during the best and worst times of my life. She is the most amazing person I know and I am a better person because of her. It is to Unity, my parents and my late grandfather that I dedicate this thesis.

Contents

Abstract	vii
Acknowledgments	ix
1 Introduction	1
1.1 Background	1
1.2 Motivation and Contributions of the Thesis	3
1.3 Outline of the Thesis	5
2 Preliminaries and Literature Review	7
2.1 Background	7
2.2 Object Representations for Tracking	8
2.2.1 Shape Representations	9
2.2.2 Appearance Representations	11
2.3 Feature Selection for Tracking	12
2.3.1 Colour	12
2.3.2 Optical Flow	14
2.3.3 Edges	14
2.3.4 Texture	16
2.3.5 Feature Combination in Object Tracking	19
2.4 Object Tracking Algorithms	20
2.4.1 Probabilistic Approaches	21
2.4.1.1 Kalman Filter	22
2.4.1.2 Particle Filter	23
2.4.2 Deterministic Approaches	25
2.4.2.1 Kernel-based Tracking	26
2.4.2.2 Literature on Kernel-based Tracking	29
2.4.2.3 Kernel-based Tracking versus Particle Filter	31
2.5 Multi-camera Tracking	32

2.5.1	Multi-view Preliminaries	33
2.5.1.1	2D Homography	33
2.5.1.2	Multi-planar Homography	35
2.5.1.3	Epipolar Geometry	35
2.5.1.4	The Fundamental Matrix	37
2.5.1.5	Preliminary Remarks	38
2.5.2	Literature on Multi-camera Tracking	38
2.5.2.1	Feature Matching Approaches	38
2.5.2.2	Geometry-based Approaches	39
2.6	Concluding Remarks	41
3	Kernel-based Tracking From A Probabilistic Viewpoint	43
3.1	Maximum Likelihood Formulation	43
3.1.1	Kernels and Probabilities	45
3.1.2	Joint-likelihood Function	46
3.1.3	Expectation-Maximisation	47
3.1.3.1	E-Step	47
3.1.3.2	M-Step	48
3.2	Minimising Mutual Information	48
3.3	Implementation	51
3.3.1	Automated Orientation Estimation	53
3.3.2	Automated Scale Estimation	55
3.3.3	Tracking with Arbitrary Kernel Profile	57
3.3.3.1	Epanechnikov Kernel	58
3.3.3.2	Diffusion Kernel	59
3.4	Experiments	60
3.5	Conclusions	66
4	Feature Combination in Object Tracking	69
4.1	Kernel-based Tracking in Arbitrary Feature Spaces	69
4.1.1	Texture versus Colour	70
4.1.1.1	Normal RGB Inputs	71
4.1.1.2	Monochromatic and Thermographic Inputs	72
4.1.2	Remarks	74
4.2	Multiple Features Formulation	74
4.3	Feature Combination via Graph Embedding	75
4.3.1	Graph Embedding and Computer Vision	75

4.3.2	Feature Mapping	77
4.3.3	Double Centering	79
4.3.4	Minimising Feature Correlation	80
4.4	On-line Feature Weight Updating	81
4.5	Algorithm Description	83
4.6	Experiments	85
4.7	Conclusions	93
5	3D Kernel-Based Object Tracking	97
5.1	Mean-shift in 3D	98
5.2	Consistent Point Sampling Across Multiple Views	102
5.2.1	Vanishing Points and Vanishing Lines Consistency	102
5.2.2	Point Mapping Between Cameras Using Reference Object	104
5.2.3	Multiple Planar Homography	107
5.3	Camera Weights Estimation	110
5.4	Algorithm Description	111
5.5	Experiments	114
5.6	Conclusions	124
6	Conclusions and Future Directions	125
6.1	Contributions	125
6.2	Future Directions	127
A	List of Publications	129
	Bibliography	130

Chapter 1

Introduction

1.1 Background

Object tracking and recognition is a classical problem in pattern recognition, computer vision and robotics. In today's world, with the rapid advancement in computing power and the availability of affordable high-resolution recording devices, the increasing need for automated video analysis has generated a great deal of interest in object tracking research. The application areas for robust object tracking are vast, with some of the notable ones being

- Advanced surveillance: real-time monitoring systems that can detect and track suspicious events such as unattended luggage, breach of security or aggressive behaviour. Human tracking, as a result, plays a prominent role in this research area.
- Smart cars: real-time embedded systems in vehicles that can detect obstacles such as pedestrians or other surrounding vehicles on the road to minimise the chance of collision while travelling at high speed. Such a system often consists of a combination of both normal and infra-red cameras along with other sensors. Here, the tracking task is usually performed at a more subtle level, for example, lane tracking and path planning.
- Traffic monitoring: smart camera network that can analyse traffic statistics to direct traffic flow. The traffic information can be obtained by either processing the camera images in a centralised computer system or, more recently, in each camera itself thanks to the improvements of modern smart cameras. In this area, object tracking algorithms are often targeted at rigid objects such as trucks, cars

and, in most cases, are required to operate in conjunction with an object detection algorithm.

- Sport analysis: this application area typically includes ball and athlete tracking so that the 2D and 3D trajectory data can be used for better visualisation for the audience and quantitative analysis for training purposes.
- Perceptual user interface: facial tracking and human gait recognition have been used to facilitate users to interact with computer without the need for a physical input devices such as keyboard or pointer. Applications in this field can vary from driving assistant programs in high-end automobiles, facial animation modelling in the movie industry or virtual 3D interactive gaming platform such as project NATAL of Microsoft (Microsoft Corporation, 2010).

The significance of this research area, on the one hand, has led to a momentous increase in the volume of research in the community over the years, amongst which there are some milestone algorithms such as the particle filter (Pérez et al., 2002) and the kernel-based tracker (Comaniciu et al., 2003), also known as the mean-shift tracker. On the other hand, a number of historical problems in object tracking still remain with substantial headroom for improvements, they are

- Loss of information caused by the projection of the 3D world on a 2D image. This is a permanent issue with single-view object tracking due to the limited field of view of the camera itself. The effect can be mitigated by increasing the number of cameras observing the scene. However, this is not a viable option in many practical scenarios due to the cost and location constraints.
- Image noise is generated during the formation of the image on the camera sensor. These artificial noises are often found in images of scenes which are poorly lit, resulting in black spots with no information and are filled with random colour responses of the sensor. Increasing the size and the sensitivity, i.e. the resolution, of the the sensor will help reduce the noise significantly, yet also raise the cost by a similar proportion.
- Complex object motion is a well known problem in object tracking, especially with objects whose movement cannot be accurately predicted ahead of time such as those found in sports and animals tracking problems.
- Partial and full object occlusions appear in most tracking scenarios with non-rigid objects. As the object moves around the scene, its appearance, size and

shape can drastically change after being occluded for a period of time. As a result, trackers which rely upon the appearance features of the object often need to employ additional information such as the last known positions; a database of the recorded appearance of the object; or information from one or more additional viewpoints to detect the object in the scene and continue the tracking task.

- Significant changes in object illumination are caused when the lighting condition of the scene varies during the tracking sequence or when the object moves from a bright area to a shaded one and vice versa. This problem mostly affects colour-based trackers which nowadays represent a minority in the tracking research volume.
- Significant object deformation in size, shape and orientation is a common problem in most tracking scenarios. This problem occurs when: the tracking object moves closer or further away from the camera; the camera is not mounted upright; or the camera lens is non-conventional such as those found in an omnidirectional camera.
- Limitation in the processing power to meet real-time performance. This problem arises from the fact that many state-of-the-art tracking algorithms require a considerable amount of pre-processing steps to produce various image features to enhance the tracking performance. These steps can sometimes delay the processing speed to a certain extent and, hence, should be taken into consideration seriously.

These above elements often coexist in most practical scenarios, amongst which high complexity factors such as occlusions and transformations are more frequent than others. Their existence is one of the main reasons why object tracking still remains as a challenging task despite the maturity of this research field.

1.2 Motivation and Contributions of the Thesis

Realising the need for a robust tracking framework, this thesis aims to explore, improve and extend the present approaches related to object tracking to cope with the confounding factors addressed in the previous section. In particular, we will focus on the kernel-based method proposed by Comaniciu et al. (2003). To this end, we aim to develop an extension framework based upon the original algorithm and make use of

advanced statistical methods, multiple image features and multiple cameras to solve the problems of severe transformations, colour variation and target occlusions during the course of tracking.

The motivation to use a kernel-based method as the base algorithm is due to the fact that the kernel-based tracker, along with the particle filter tracker (Pérez et al., 2002), is one of the most popular approaches towards robust tracking. The tracker relies on the mean-shift optimisation scheme (Cheng, 1995), which has been proven to be very robust and capable of being extended to incorporate additional advanced features. Furthermore, we realise the trend to use multiple image features to enhance the tracking performance in complex lighting conditions. To this end, we take advantage of a robust collection of image features such as Scale Invariant Feature Transform (SIFT), Local Binary Pattern (LBP) and Harr-like, which will be discussed in the later chapters. Finally, we explore the possibility of extending the kernel-based tracker to 3D by using multiple cameras. This treatment is twofold. First, the use of multiple cameras for better coverage has become a common practice in many surveillance systems including banks, airports, plazas, parking lots and even convenience stores. Second, robust multi-camera tracking not only provides a significant leap over single-camera tracking, but also gives rise to other potential applications such as 3D scene reconstruction, image stitching and object position tracing.

With that in mind, the roadmap for the work in this thesis and its main contributions can be summarised as follows

- Develop a robust probabilistic tracking framework for a single camera based upon the kernel-based method proposed by Comaniciu et al. (2003). The algorithm aims to improve the tracking performance against abrupt object deformation in size, shape and orientation. To do this, we cast the tracking task into an Expectation-Maximisation setting where the target deformation can be estimated on the polar coordinate system. In addition, our approach is also quite general and can employ a variety of kernel profiles as well as image features to suit the tracking constraints. Through real-world video sequences with quantitative results, we show that our algorithm achieves significant gain in tracking performance over a number of benchmark algorithms elsewhere in the literature.
- Develop a graph-based method to combine multiple image features such as texture, contrast and colour information to enhance the tracking performance in the presence of object occlusion and illumination variation. Furthermore, an on-line feature weight updating mechanism is presented so as to adjust the tracking model to incorporate new changes in the image features during the course of

tracking. We illustrate the robustness of our algorithm by providing quantitative analysis on real-world sequences, with comparison to two other competing algorithms. Throughout the experiments, our algorithm shows remarkable gap over the alternatives in tracking accuracy thanks to the combination process prioritising the most discriminant image features effectively. The novelty of this approach is the use of graph-based methods commonly found the literature of image segmentation as a means to combine multiple image features for object tracking algorithms, or more specifically, the mean-shift tracker.

- Extend the kernel-based tracker to a 3D setting by using multiple cameras and multi-view geometric techniques. To this end, we construct a 3D coordinate system making use of the homography matrices between the cameras observing the scene. The tracking task can then be cast into a unified framework and the results can be back-projected to each camera view with high consistency. Furthermore, we illustrate the utility of our algorithm on real-world multi-camera sequences, with quantitative comparison to a benchmark multi-camera tracker. Here, our algorithm shows performance leads throughout the experiments, especially with two and three cameras. The main contribution here, however, is to show how the mean-shift procedure can be deployed in a 3D setting to cope with full occlusion. This treatment is also devoid from a heavy-computational 3D voxelisation process, while maintaining the integrity of the mean-shift tracker to open-up the possibility of a 3D implementation for other existing 2D methods elsewhere in the literature.

Here, it is worth noting that the kernel-based tracker is often referred as the mean-shift tracker, hence these two names will be used interchangeably in this thesis.

1.3 Outline of the Thesis

Having addressed the motivations and objectives of work of this thesis, the thesis structure can be summarised as follows.

In the present chapter, an overview of object tracking is presented. The motivations for the work of the thesis and its contributions are posed in Section 1.2. The layout of the thesis is presented in Section 1.3 and the list of publications for the work in the thesis can be found in Appendix A.

Chapter 2 presents the preliminaries and the literature review relevant to the work of this thesis. This chapter commences with an overview of object tracking and its

historical issues. It is followed by a review on some well known single-view tracking algorithms the particle filter and the kernel-based tracker. The chapter subsequently focuses on the kernel-based methods with comparison to other existing techniques. A brief introduction of multi-view geometry is also presented together with a literature review on multi-camera tracking methods.

In Chapter 3, we present a statistical characterisation of the mean-shift tracker presented in (Comaniciu et al., 2003). We attempt to view the mean-shift optimisation process as an Expectation-Maximisation process and hence, provide a novel technique to recover the object scale and orientation making use of the polar coordinates. The chapter also studies the use of different kernel profiles suitable for the kernel-based tracker such as the diffusion kernel and Epanechnikov kernel. Finally, the results on real-world sequences are provided with comparison to a number of alternative methods.

In Chapter 4, we investigate the use of feature combination to enhance the performance of the mean-shift tracker, especially in the presence of abrupt changes in the object appearance due to illumination variation and temporary occlusion. In our approach, we abstract the low complexity features used for purposes of tracking to a relational structure and employ graph-spectral methods to combine them. Furthermore, we propose an on-line updating method to incorporate new features to the target model during the course of tracking. Finally, we provide quantitative analysis on the low complexity features and their suitability towards various input video types such as thermographic, infra-red and low-resolution cameras.

In Chapter 5, we extend the kernel-based tracking algorithm to a 3D setting making use of multiple cameras with some overlapped view of the tracking scene. In particular, a 3D scene is constructed from the planar homography matrices between the observing cameras. Here, a robust point sampling method is presented to consistently map the set of sample points of the tracking target from the 3D space to each observing view. To this end, the object is subsequently tracked in the 3D scene by a 3D mean-shift formulation whose sample points, using our sampling method, are drawn from all participating views. This treatment naturally lends itself to the recovery of the camera weights, which measure how well the target is seen in each view. Here, we illustrate the robustness of our algorithm on a number of real-world multi-camera sequences and provide quantitative comparison with a benchmark multi-camera tracker.

Finally, concluding remarks are provided in Chapter 6 summarising the work done in this thesis and identifying possible future directions.

Chapter 2

Preliminaries and Literature Review

Due to the maturity of this research area, the literature body of object tracking is vast and, hence, writing a comprehensive survey on this topic, by itself, is a daunting task. In this chapter, we only present the preliminaries and literature review related to the work of this thesis. In particular, the chapter starts with a brief overview of the tracking literature, followed by some of the state-of-the-art tracking algorithms such as the particle filter and the mean-shift tracker. We then provide a brief introduction to multi-view geometry and the some of the most recent achievements in this research field. For a complete survey of object detection, segmentation and tracking algorithms, we direct the reader to the work by Yilmaz et al. (2006).

2.1 Background

Object tracking, in its simplest form, can be defined as the problem of estimating the trajectory of one or multiple object(s) on an image plane as it moves around the scene (Yilmaz et al., 2006). In this process, the tracker makes use of different information about the target at the beginning of the tracking sequence or from a predefined database, so as to construct a tracking model. The tracking model can consist of various information about the object such as: appearance, position known in the last frames, travel direction. amongst others. To this end, simple trackers often rely on low complexity features such as brightness, contrast and colour appearance whereas advanced trackers exploit additional information such as size, shape, orientation and motion likelihood to further enhance the tracking performance. Regardless of which information is used, the new object location is estimated as the region on the image plane that yields the highest similarity to the tracking model.

With the existence of uncertain factors such as image noise and occlusion, object

tracking algorithms often impose a number of constraints to suit a certain tracking environment. For example, in vehicle and pedestrian tracking, many trackers assume that the object motion is reasonably smooth with no abrupt changes. In sport tracking, one can safely constrain the shape, size and appearance of the object, say a tennis ball, to be relatively consistent throughout the tracking sequence. Prior knowledge about the object appearance is also widely used for highly specific tasks such as face and hand tracking. In short, most existing tracking methods practice the use of constraints, with different extent, in order to maximise the tracking performance with respect to its working environment.

Due to the nature of the application field, the performance of an algorithm is usually measured on a frame-to-frame basis in which the tracker successfully locates the target. The computational overhead is also one of the major aspects upon which the tracker is assessed. This is due to the fact that real-time performance is now required in most, if not all, surveillance tracking scenarios. As a result, real-time robust algorithms such as the mean-shift tracker (Comaniciu et al., 2003), the particle-filter tracker (Pérez et al., 2002) and the Adaboost face detector (Viola and Jones, 2001) have become increasingly popular as they can be implemented on small-scale systems like those found on robots and automated vehicles.

Having identified the context and the requirements of the problem, a tracking algorithm can be developed by solving the following questions

- Which object representation is suitable for a given tracking environment.
- Which image feature(s) can best differentiate the object from its surrounding.
- What strategy can be used to track the object making use of the selected features and representation.

With that in mind, the following sections in this chapter will discuss about object representation, image features and the current state-of-the-art tracking algorithms.

2.2 Object Representations for Tracking

Object representation, as the name suggests, is how an object is defined in a tracking environment. In general, an object can be represented using shape representations, appearance representations or a mixture of both. The selection process for a suitable object representation given a tracking task is based upon the object characteristics such as colour, shape and rigidity, as well as the surrounding environment. In this section,

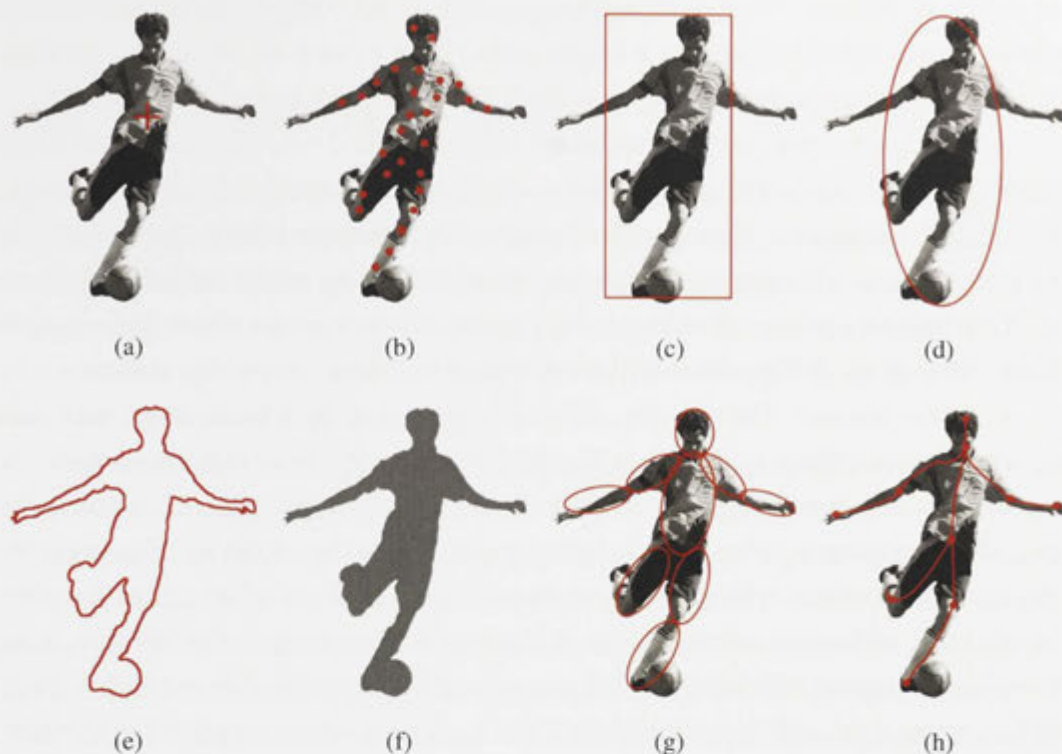


Figure 2.1: Object representations: (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) object contour, (f) object silhouette, (g) multiple patches with joints, (h) object skeleton.

we follow the work in (Yilmaz et al., 2006) by firstly describing the common shape representations and, subsequently, addressing the joint representations of shape and appearance used in object tracking.

2.2.1 Shape Representations

There are a number of ways to represent an object using the characteristic of their shapes and sizes. Figure 2.1 shows some of the most popular shape representations including points, primitive shapes, articulated shapes and silhouette.

Points representation: The object is represented by a point as the centroid of the object or by a set of points as shown in Figure 2.1(a) and (b), respectively. The former representation can be suitable for objects whose size are significantly small compared to the size of the frame such as individual birds of a bird flock, a tennis or a ping-pong ball or human objects marching in a formation. The application domain for single-point representation, as a result, is often associated with motion analysis of a cloud

of points or objects. For instance, Shafique and Shah (2005) track distant birds by viewing each individual bird as a single point. In a related development, Veenman et al. (2001) use point representation to track the seeds in a moving dish sequence.

As the object size increases, multiple points are deemed necessary to maintain high tracking performance, which also induce significant computational cost compared to the single-point process. However, the density of these sample points can be controlled by performing a sub-sampling process on the object region with a sampling rate, say S_r . One representative work using multiple points drawn from the object is the particle filter (Pérez et al., 2002), which will be discussed in details later in this chapter.

Primitive shapes: The tracking object is represented by a basic shape such as a rectangle or an ellipse as shown in Figure 2.1(c) and (d), respectively. Despite the assumption that basic shapes are suitable for simple rigid objects such as trucks and cars, they are quite popular in tracking non-rigid objects due to the fact that primitive shapes can be defined with a few parameters and non-rigid transformations can often be enclosed within the coverage area of these primitive shapes. For instance, most kernel-based trackers (Collins, 2003; Zivkovic and Krose, 2004; Han and Davis, 2004) following the work of Comaniciu et al. (2003) use a rectangle or an ellipse to represent the tracking object.

Object silhouette and contour: Contour representation defines the boundary between an object and its surrounding background as shown in Figure 2.1(e). The region inside this contour is called the silhouette of the object as shown in Figure 2.1(f). Silhouette and contour representations are suitable for tracking complex nonrigid objects such as humans or animals. Haritaoglu et al. (2000) use silhouette for tracking humans in a surveillance application. It is, however, worth noting that, the contour extraction process can be computationally expensive and, in many cases, complicated non-uniform backgrounds can penalise the tracking performance.

Articulated shape models: An articulated object is composed of different body parts that are held together by a number of joints. Each of these parts can be represented by a cylinder or an ellipse. For instance, in Figure 2.1(g), the human body is an articulated object that consists of the head, the arms, the torso and the legs connected by joints. The joint of the neck connects the head, two arms and the body limb whereas the joint of the knee only connects the upper and lower parts of the leg. As a result, specific constraints such as kinetic motion, displacement bandwidth and joint angles can be applied to improve the tracking performance. Furthermore, the object skeleton can also be extracted by applying medial axis transform to the object silhouette. This model is commonly used as a shape representation for recognising objects such as the work in (Ali and Aggarwal, 2001). Skeleton representation can also be used to model

both articulated and rigid objects as shown in Figure 2.1(h).

2.2.2 Appearance Representations

Appearance representation, in contrast to shape representation, does not necessarily rely on the geometric properties of an object. In fact, a tracking model is constructed making use of the object appearance such as colours, gradient and textures. The most common appearance representations include probability density function (PDF), templates, active appearance models (AAM) and multi-view appearance models, which will be discussed in this section.

Probability density function (PDF): This representation aims to abstract the object appearance such as colours and textures in a PDF that later can be used for purposes of tracking. The PDF can be of parametric form, such as Gaussian (Zhu and Yuille, 1996) and mixture of Gaussians (Paragios and Deriche, 2005), or non-parametric such as the kernel density estimation method, also known as Parzen windows (Elgammal et al., 2002), and M -bin histograms (Comaniciu et al., 2003). In most cases, these PDFs can be computed from the image region specified in the shape models discussed in the previous section. This representation, however, does not depend on the shape and size of the object and hence can be used for non-rigid objects such as humans.

Templates: The template representation is constructed making use of both geometric shapes and appearance information of the object (Fieguth and Terzopoulos, 1997). This information is often extracted from a single viewpoint and, hence, makes the template representation suitable for rigid objects with small variation in pose and appearance. For example, the shape and colour appearance of a car can be safely presumed to be relatively consistent during a tracking sequence and, thus, can be a suitable candidate for the template representation.

Active appearance model (AAM): The active appearance model draws information from both the region covered by the object and the object contour edges, so as to construct a statistical model of the shape and the grey-level appearance of the tracking object. The tracking task is then cast to a matching framework in which the target is found by involving a set of parameters that minimises the difference between the search region and the AAM. As a result, this method requires a training phase in which the model structure is learned from a set of samples using machine learning techniques such as Principle Component Analysis (PCA) or Support Vector Machine (SVM). A representative work in this field is by Edwards et al. (1998), in which the object face is found in each image by fitting the image to a generic AAM of human face with feature nodes such as eyes, eyebrows, nose and lips.

Multi-view appearance models: The multi-view appearance model method, as the name suggests, constructs the tracking model using information extracted from different views of the object. This information is not necessarily obtained at one instance in time from a multi-camera setting. On the contrary, they can be collected from a single viewpoint during the training phase. The multi-view information is then subsequently encoded in the tracking model by training a set of classifiers such as the SVM (Avidan, 2004) or the Bayesian network (Park, 2004).

Overall, object representation plays a pivotal role in a robust tracking framework as it defines how an object is seen, modelled and structured in the scene. The choice of which object representation to use, as expected, is highly dependant on the tracking scenarios, in which there are a variety of impact factors such as the nature of the object, the scene background, the number of cameras and, most importantly, the tracking algorithm itself. Furthermore, shape representation and appearance representation can be used separately or jointly to improve the tracking performance.

2.3 Feature Selection for Tracking

Following the previous section, the second crucial part of building a robust tracking framework is to select the image features to best distinguish the tracking object from its surrounding background. In practice, the study of image features and descriptors, on its own, gives rise to an active research field within the computer vision community, with applications spread across tracking, pattern recognition and segmentation. The contrast information derived from the integral image, for instance, has been used for face detection (Viola and Jones, 2001) and object tracking (Yin and Collins, 2008). Colour features are often used for histogram-based appearance representation, whereas object edges are used for contour-based representation. It is also common these days to see an algorithm which employs a combination of these features to improve robustness, hence this section will describe a number of popular image features in the tracking literature.

2.3.1 Colour

The colour of an object is influenced by two physical factors: the spectral power distribution of the light source(s), and the surface reflectance properties of the object. The internal specifications of the camera such as calibration parameters and wavelet filters also affect the final colour output of the object perceived by the human eye. Colour information is often formulated in different colour spaces. A colour space is a method

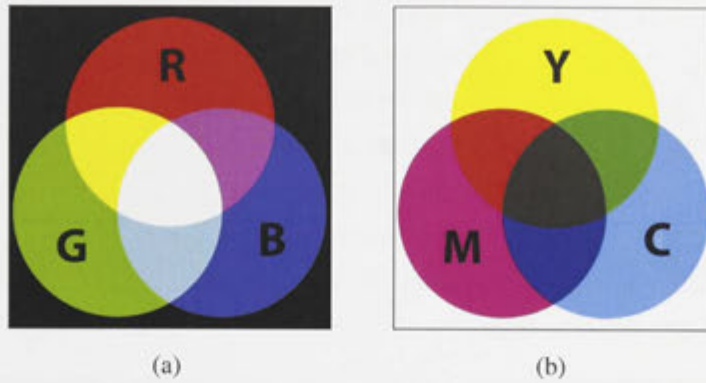


Figure 2.2: Generic colour spaces: (a) The RGB colour space using additive colour mixing; (b) The CMKY colour space using subtractive colour mixing.

by which we can specify, create and visualise colours. As humans, we may define a colour by its attributes of brightness, hue and colourfulness. A computer may describe a colour using the amounts of red, green and blue phosphor emission required to match a colour. A printing press may produce a specific colour in terms of the reflectance and absorbance of cyan, magenta, yellow and black inks on the printing paper. In general, colour can be classified in two broad categories: tri-chromatic colours and monochromatic colours.

Tri-chromatic Colour Spaces: Tri-chromatic colour spaces are ones where each colour is specified using three coordinates, or parameters. These parameters describe the position of the colour within the colour space being used. They do not tell us what the colour is, as that depends on what colour space is being used. There are a number of tri-chromatic colour spaces, amongst which the RGB (Red, Green, Blue) colour space is the most common 3D colour space (see Figure 2.2(a)). It is an additive colour system based on tri-chromatic theory found in systems that use a CRT to display images, hence it has become the most popular colour space in computer systems. The RGB colour space, however, comes with a number of drawbacks. For instance, the differences between colours in the RGB space do not correspond to the differences perceived by humans (Paschos, 2001), i.e. the space is not perceptually uniform. The RGB dimensions are also highly correlated. In contrast, $L^*u^*v^*$ and $L^*a^*b^*$ are perceptually uniform colour spaces, whereas HSV (Hue, Saturation, Value) is an approximately uniform colour space. Other notable colour spaces are CMYK (see Figure 2.2(b)), YCbCr, YCC and normalised RGB.

Monochromatic Colour Spaces: A monochromatic colour space, as suggested by the name, only offers one single channel for the colour intensity. In computer vision,

such space is often normalised between 0 and 1 and the image is shown in grayscale level for illustration purposes. As opposed to the tri-chromatic colour spaces, there are numerous ways to construct a custom monochromatic colour space to serve a specific task. For example, an RGB image can be converted a grayscale image as monochromatic version. There are more advanced algorithms such as the work by Collins et al. (2005), in which a feature pool of 49 monochromatic feature spaces is constructed by unique combinations of R, G, B values. In mathematics term, let $\Phi = \{\phi_i\}_{i=1\dots 49}$ be the set of 49 image features where

$$\phi_i \equiv \{k_1 R + k_2 G + k_3 B \mid k_* \in [-2, -1, 0, 1, 2]\} \quad (2.1)$$

In a related development, Han and Davis (2004) combine two different tri-chromatic colour spaces, RGB and the normalised *rgb*, to construct a set of 14 monochromatic colour spaces for purposes of tracking.

To sum up, it is worth noting that these colour spaces are susceptible to image noise and change in illumination. As each has its own pros and cons, there is no ‘best’ colour space and, hence, a variety of colour spaces have been used in tracking, both singly and in combination.

2.3.2 Optical Flow

Optical flow is a gradient-based image feature which consists of a dense field of displacement vectors defining the translation of each pixel in an image region. This vector field is computed over two or more consecutive frames, in which point correspondence is established using two primary constraints: the flow field smoothness constraint and the brightness constancy constraint. The former constraint assumes the motion of a moving object to be reasonably smooth with no abrupt changes, while the latter constraint assumes the apparent brightness of the object remains relatively constant during the course of tracking. As a result, most methods using this image feature make ample use of these assumptions. Optical flow has been used extensively in motion-based segmentation and tracking applications, with representative work including (Horn and Schunk, 1981; Lucas and Kanade, 1981; Szeliski et al., 1994; Black and Anandan, 1996). To this end, Barron et al. (1994) provide a detailed evaluation on the performance of the optical flow methods.

2.3.3 Edges

Object edges are the object boundaries that separate the object from the background. In computer vision terms, these boundaries generate strong changes in the image in-

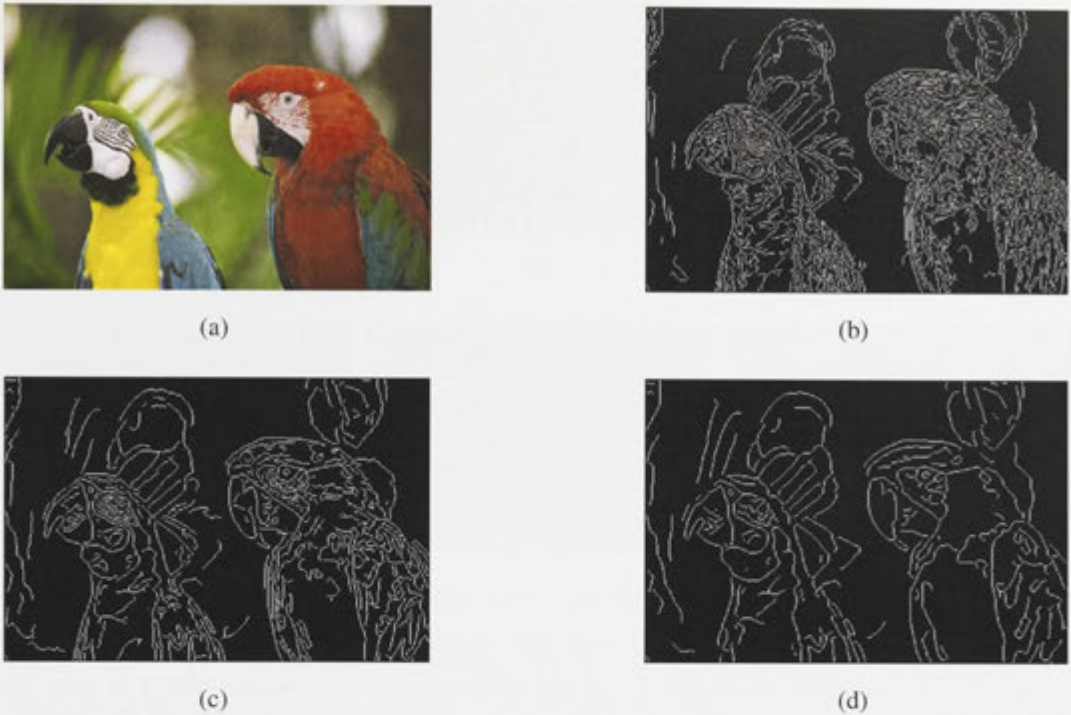


Figure 2.3: The performance of the Canny edge detector (Canny, 1986) with different parameters for the Gaussian filter: (a) The original image; (b) $\sigma = 0.5$; (c) $\sigma = 1.0$ and (d) $\sigma = 1.5$.

tensity, or in other words, discontinuity in the image gradient. As a result, algorithms that track the boundary of the objects usually employ edges as the object representation. An evaluation of the edge detection algorithms is provided by K. Bowyer and Dougherty (2001), amongst those, the Canny edge detector (Canny, 1986), the Sobel method (Sobel and Feldman, 1968) and the zero-crossing method are the most popular ones.

These edge detectors, however, often need to be ‘tuned’ with specific parameters depending on the context of the problem. The Canny edge detector, for instance, relies on two threshold parameters to distinguish between strong and weak edges, as well as the standard deviation of the Gaussian filter used to filter the image. Similarly, the Sobel method requires two thresholds and an optional parameter to specify which direction the edge detector iterates, i.e. horizontal or vertical. Figure 2.3 shows the results of the Canny method on an image of two parrots with different Gaussian filters governed by the parameter σ . It is obvious that the number of detected edges varies significantly with different σ . Consequently, the overall performance of the edge detector can suffer from an incorrect selection of these parameters.

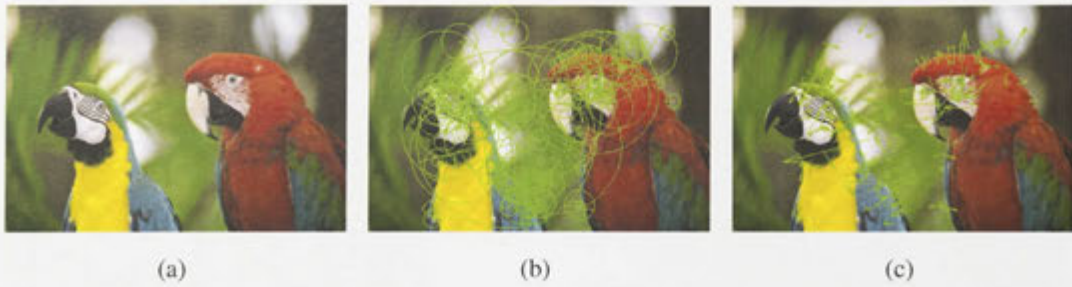


Figure 2.4: SIFT features (Lowe, 1999): (a) the original image; (b) and (c) SIFT features with supporting base as the area of the circles and the main orientation as pointed by the arrows, respectively.

2.3.4 Texture

Texture is a measure of the image intensity on a surface of interest, which quantifies the degree of smoothness and regularity on that surface. Texture descriptors are usually much less susceptible to image noise and change in illumination, which is also the main advantage over colour features. However, it also requires additional processing steps to generate the descriptors from the colour information of the original image. To date, there have been an increasing number of texture descriptors used in object tracking. Amongst those, the following texture features are the most popular.

Scale Invariant Feature Transform (SIFT): The SIFT features (Lowe, 1999) are based on the the appearance of the object at particular interested points that are invariant to image scale and rotation. The method makes use of the Difference of Gaussian (DoG) effect on an image to extract maxima and minima points, also known as extrema points, which are invariant to scale variation. After the outliers are removed, these extrema points are assigned one or more orientations based on local image gradient directions, so as to achieve invariance to rotation. Due to its robustness against changes in illumination, noise and minor changes in viewpoint, SIFT features have been widely adopted in various applications such as object recognition (Beis and Lowe, 1997), panorama stitching (Brown and Lowe, 2003) and 3D-object tracking (Gordon and Lowe, 2006). Figure 2.4 illustrates the SIFT feature points detected on the image of the two parrots. Here, each feature point has a supporting base governed by the area of the circle as shown in Figure 2.4(b), as well as an orientation vector denoted by the arrow as shown in Figure 2.4(c).

Harr-like Features: The Harr-like feature introduced by Viola and Jones (2001) is a multi-scale contrast operator that relies on a fast computation of the integral image. In its simplest form, a simple rectangular Haar-like feature can be defined as the dif-

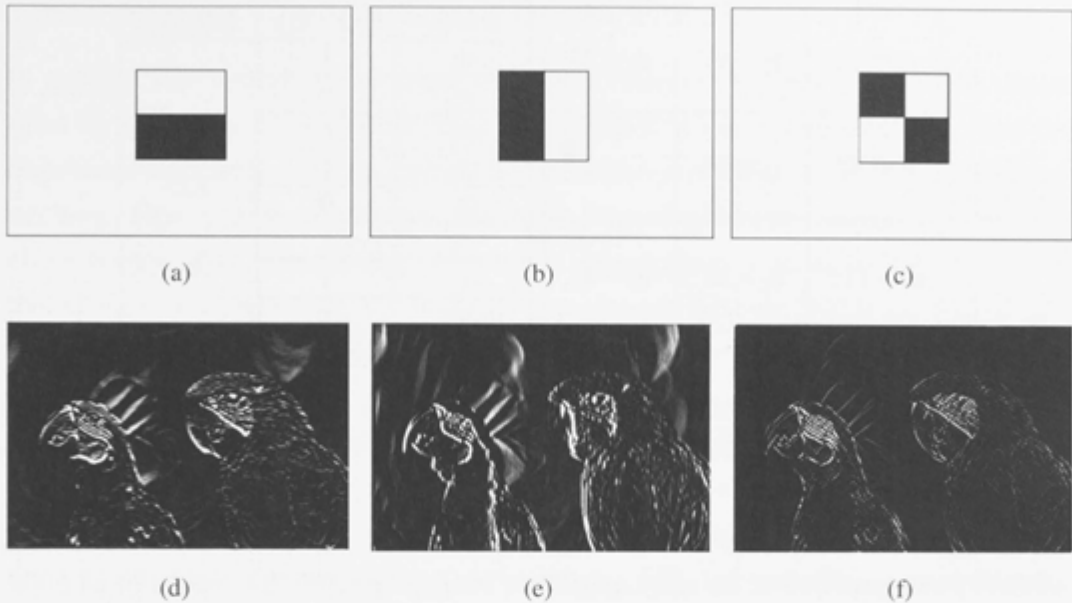


Figure 2.5: Harr-like rectangle features (Viola and Jones, 2001). Top row: (a) two horizontal rectangles; (b) two vertical rectangles; (c) four cross type rectangles. The feature is calculated as the sum of the pixels in the white box minus that of the black box. Bottom row: (d), (e) and (f) are the corresponding results for the features in (a), (b) and (c), respectively.

ference of the sum of pixels of areas inside the rectangle, which can be at any position or scale within the original image. This quantity can be found by first computing the Summed-area Tables (Crow, 1984) of the image in one iteration, also known as the integral image. The difference of the sum of pixels can then computed by a direct lookup from the integral image, hence makes the feature extraction process extremely fast regardless of the size and shape of the component rectangles. For instance, Figure 2.5 illustrates the effects of the Harr-like filters on an the image using two horizontal rectangles, two vertical rectangles and four cross-type rectangles. In this case, the input image is filtered to highlight the vertical, horizontal and diagonal edge, respectively.

To date, Harr-like features are extremely popular in the object detection field inspired by the work in (Viola and Jones, 2001). Moreover, there has been an increasing trend to use Harr-like features in object tracking. For instance, a number of particle filter trackers such as the work in (Yang et al., 2005; Li and Wang, 2005) employ this robust feature to enhance the tracking performance.

Local Binary Pattern (LBP): The LBP (Ojala et al., 1996) is a a primitive texture descriptor which captures the contrast of the an image by comparing the intensity value

$a_1 = 62$	$a_2 = 41$	$a_3 = 23$	Index	$g(a_i - a_j)$	$LBP(x_j)$
			a_1	1	135
			a_2	0	
			a_3	0	
			a_4	0	
			a_5	0	
			a_6	1	
			a_7	1	
			a_8	1	
$a_8 = 85$	$a_0 = 50$	$a_4 = 17$			
$a_7 = 115$	$a_6 = 92$	$a_5 = 45$			

Figure 2.6: Local Binary Pattern (LBP). Left: The image patch surrounding the centre pixel a_0 . Right: The LBP of the centre pixel a_0 is 10000111 in binary unit, or 135 in decimal unit.

of each pixel with that of its eight neighbour pixels. This process results in an eight-digit binary sequence for each pixel, which can then be converted back to the decimal unit and hence can be used as a means to measure the local contrast of each image pixel. For example, Figure 2.6 shows an example of how to compute the LBP feature of an image patch. In particular, the centre pixel x_i has an intensity value given by a_0 . The eight neighbourhood pixels, which are indexed from 1 to 8, have intensities given by the variables $\{a_j\}_{j=1...8}$. The LBP is then given by

$$LBP(x_i) = \sum_{j=1}^8 g(a_i - a_j) 2^{(j-1)} \quad (2.2)$$

where a_j is the intensity of the pixel indexed j in the set of eight-neighbours of x_i and $g(a_i - a_j)$ is a function defined as follows

$$g(a_i - a_j) = \begin{cases} 1 & \text{if } a_i - a_j \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

As result, the LBP of the centre pixel a_0 , in this case, is 10000111 in binary unit, or 135 in decimal unit. The idea underpinning the computation of the LBP is to create a binary code that describes the local texture pattern by thresholding the eight neighborhood pixels by the intensity of their centre. As a result, we can compute the values for the function $g(\cdot)$ and, subsequently, the binary code is converted to a decimal quantity that can be used for fixed point operations. The LBP descriptors have been used in face tracking (Choi and Lam, 2008; Wang and Li, 2008) as well as non-rigid object tracking (Chen et al., 2006).

2.3.5 Feature Combination in Object Tracking

In general, one would aim to select the image features that best separate the object from its surrounding, as well as being discriminant to other image features used for object representation. To date, colour has been the most widely used feature in object tracking. However, most colour bands suffer from illumination variation. When this effect is inevitable, other image features are incorporated to enhance the appearance model of the tracking target. For instance, Cremers and Schnorr (2003) use optical flow as a feature for contour tracking. Jepson et al. (2003) use steerable filter responses for tracking.

Alternatively, there has been a momentous increase in the use of a combination of image features in modern tracking algorithms. Stern and Efros (2002) improve the tracking performance by adaptively swapping the tracking features across five predefined colour-space combinations, namely RG, rg, HS, YQ and CbCr. As the target, in this case a human face, evolves through multi-positioned light sources with varying types of illumination, Stern and Efros (2002) show that switching between the colour spaces results in increased tracking performance. Nguyen and Smeulders (2004) use a set of Gabor filters to transform the image intensities into texture information. The tracking algorithm then maintains a set of discriminant functions, of which each function recognises a pattern in the object region against the background patterns that are currently relevant. Object matching is then performed efficiently by maximisation of the sum of the discriminant functions over all object patterns. As a result, the tracker searches for the region that matches the target and it also avoids background patterns seen before.

More recently, Collins et al. (2005) deploy the mean-shift tracker (Comaniciu et al., 2003) on a feature pool of 49 log-likelihood images comprised by unique combinations of R, G and B values as in Equation 2.1. These features form a pool from which the discriminant score for each feature is computed from the Variance Ratio between the histogram of the target object and that of the surrounding background. The Variance Ratio is defined as

$$VR(L, \mathbf{Q}_{\phi_i}, \mathbf{P}_{\phi_i}) \equiv \frac{\text{var}(L; (\mathbf{Q}_{\phi_i} + \mathbf{P}_{\phi_i})/2)}{[\text{var}(L, \mathbf{Q}_{\phi_i}) + \text{var}(L, \mathbf{P}_{\phi_i})]} \quad (2.4)$$

where $\mathbf{Q}_{\phi_i}$ and $\mathbf{P}_{\phi_i}$ are the M -bin histograms of feature ϕ_i corresponding to the target and the candidate regions respectively; $\text{var}(L, \mathbf{Q}_{\phi_i})$ is the variance of L with respect to $\mathbf{Q}_{\phi_i}$ and L is the log likelihood of the feature ϕ_i given by

$$L_{\phi_i} = \log \frac{\max\{\mathbf{Q}_{\phi_i}, \delta\}}{\max\{\mathbf{P}_{\phi_i}, \delta\}} \quad (2.5)$$

where δ is a small value to prevent dividing by zero or taking the log of zero. With that in hand, the tracker subsequently selects the five most discriminant combinations for the tracking task. In a similar approach, Han and Davis (2004) combined two different colour spaces, RGB and normalised RGB (rgb), so as to construct 14 log likelihood images in R, G, B, RG, GB, BR, RGB as well as r , g , b , rg , gb , br , rgb . Feature extraction is then achieved by performing Principle Component Analysis (PCA) on the foreground and the local image background.

In addition to the algorithms mentioned above, machine learning techniques such as Adaboost have also been employed to enhance multiple-feature trackers (Avidan, 2005; Grabner and Bischof, 2006). The Adaboost technique (Freund and Schapire, 1999), also known as Adaptive Boosting, is a machine learning algorithm that combines a number of ‘weak’ classifiers to construct a ‘strong’ global one. In (Avidan, 2005), the tracking task is cast as a binary classification problem, where an ensemble of weak classifiers is trained to distinguish between the object and the background in an on-line manner. These classifiers are combined into a strong classifier using Adaboost. The strong classifier is then used to label pixels in the next frame as either belonging to the object or the background, giving a confidence map. The peak of the map, and hence the new position of the object, is found using the mean-shift optimisation scheme (Cheng, 1995).

2.4 Object Tracking Algorithms

Having addressed the most popular target representations and image features used in the literature of object tracking, we now turn our attention to the existing methods that utilise these resources to carry out the tracking task. In general, the main objective of tracking is to estimate the target position X^t at frame t , given all the measurements $Z^{1 \dots t}$ up to that moment, or more specifically, to construct a probability density function (PDF) $P(X^t | Z^{1 \dots t})$. To this end, the trajectory of the target, X^t , can be estimated by a probabilistic approach or a deterministic approach. Trackers belong to the former category use the state space model to incorporate the target properties such as position and velocity into a sequence of states indexed by the frame number. The target correspondence at the current frame is then established via a recursive Bayesian Estimation framework. Trackers belong to the latter category, on the other hand, define a cost function associating each candidate object in frame t to the last known position of the tracking target in frame $t - 1$. The position of the target at the current frame t is then computed via an optimisation scheme that aims to minimise this cost function

accordingly.

As there is a rich literature on this topic, this section only aims to discuss the representative methods in each category that are related to the work of this thesis. Here, we commence by reviewing the probabilistic approaches with notable work such as the Kalman filter tracker and the particle filter tracker. We continue this section by presenting the deterministic approaches including the mean-shift tracker and its representative work. To take our analysis further, we provide a direct comparison between the mean-shift tracker and the particle filter tracker and, subsequently, draw the conclusions on the rationale for using the mean-shift tracker as the base algorithm for the work of this thesis.

2.4.1 Probabilistic Approaches

As mentioned earlier, probabilistic methods for object tracking take the object measurements to model other motion uncertainties when establishing target correspondences between frames. In this stream, the object spatial properties such as position, velocity and acceleration are modelled using the state space approach, that is the object location, for instance, is defined by sequence of states $X^t : t = 0, 1, 2, \dots$. Note that $t = 0$ is the initialisation stage. The change in state is governed by the dynamic equation:

$$X^t = f^t(X^{t-1}) + W^t \quad (2.6)$$

where W^t is the white noise. The relationship between the measurement and the state is specified by the measurement equation $Z^t = h^t(X^t, N^t)$, where N^t is the white noise and is independent of W^t . The object position is then determined by constructing a conditional probability function $P(X^t|Z_0^{1:t})$, i.e. computing the current state of the object position at time t given all the measurements up to that moment. To do this, a theoretical solution is provided by a recursive Bayesian Estimation which solves the problem in two steps. The *prediction step* computes the the estimated state X^t using the previous state X^{t-1} as in Equation 2.6. The *correction step* computes the likelihood function $P(Z^t|X^t)$ of the current estimation, and subsequently computes the posterior density function $P(X^t|Z_0^t)$ as follows.

$$P(X^t|Z_0^t) = \frac{P(X^t|Z_0^{t-1})P(Z^t|X^t)}{P(Z^t|Z_0^{t-1})} \quad (2.7)$$

where the normalisation constant $P(Z^t|Z_0^{t-1}) = \int P(X^t|Z_0^{t-1})P(Z^t|X^t)dX^t$. This formalism gives rise to two representative approaches: the Kalman filter and the particle filter.

2.4.1.1 Kalman Filter

A Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noise measurement. The term linear means that the Kalman filter only works with the assumptions that the function $f^t(\cdot)$ and $h^t(\cdot)$ are linear, and the noises have Gaussian distribution. Viewed in this manner, the prediction step uses the state model to compute the predicted state of the variables and the predicted covariance, that is

$$\bar{X}^t = \mathbf{D}X^{t-1} + W \quad (2.8)$$

$$\bar{\Sigma}^t = \mathbf{D}\Sigma^{t-1}\mathbf{D}^T + Q^t \quad (2.9)$$

where $\bar{X}^t$ and $\bar{\Sigma}^t$ are the state and covariance prediction at time t . $\mathbf{D}$ is the state transition matrix which defines the relation between the state variables at time t and $t - 1$. Q is the covariance of the noise W . Subsequently, the correction step uses the current observation $Z^t = h^t(\bar{X}^t, N^t)$ to compute the optimal Kalman as follows

$$K^t = \bar{\Sigma}^t \mathbf{M}^T [\mathbf{M} \bar{\Sigma}^t \mathbf{M}^T + R^t]^{-1} \quad (2.10)$$

where $\mathbf{M}$ is the measurement matrix, and K is the Kalman gain, which then is used to update the state the variables and the covariance of the noise.

$$X^t = \bar{X}^t + K^t [Z^t - \mathbf{M} \bar{X}^t] \quad (2.11)$$

$$\Sigma^t = \bar{\Sigma}^t - K^t \mathbf{M} \bar{\Sigma}^t \quad (2.12)$$

It is also worth noting that, the updated state X^t is still distributed by a Gaussian. In case the function $f^t(\cdot)$ and $h^t(\cdot)$ are non-linear, they can be linearised by using the Taylor series expansion to obtain the extended Kalman filter (EKF) (Bar-Shalom, 1987), which also assumes that the state is distributed by a Gaussian.

The literature on tracking algorithms using Kalman filter is vast. In (Broida and Chellappa, 1986), Kalman filter is used to track points in noisy images. In (Rosales and Sclaroff, 1999), the EKF is used to estimate 3D trajectory of an object from 2D motion. Beymer and Konolige (1999) employ the Kalman filter to track multiple people in a stereo-camera setup.

Despite being effective, the Kalman filtering approach has a major drawback, that is the assumption that the state of the tracking variables is normally distributed. As a result, the trackers relies on this assumption will not perform effectively in case the state variables do not follow Gaussian distribution. Fast moving objects such as sport players with unpredictable motion, for instance, can produce large errors in the true posterior mean and covariance of the state variables, which then lead to sub-optimal performance and sometimes divergence of the tracker.



Figure 2.7: Results of the particle filter tracker proposed by Pérez et al. (2002). The yellow rectangles represent the final position of the tracking particles and the cyan rectangle is the mean position, i.e. the object position, computed as the weighted average of these particles.

2.4.1.2 Particle Filter

Particle filtering is often referred as the Monte Carlo importance sampling techniques, the bootstrap methods (Gordon et al., 1993) or the CONDENSATION algorithm (Isard and Blake, 1996, 1998) which is also the first application on video processing. The abbreviation CONDENSATION arises from the name CONDitional DENSity propagation.

As discussed in the previous section, the main limitation of the Kalman filter approach lies on the assumption that the state variables come from a Gaussian distribution. The particle filter addresses this problem by modelling a set of carefully chosen sample points, so as to estimate the distribution of the state variables. These sample points induce the true mean and covariance of the state variables. Hence, while being propagated through a non-linear system, they capture the mean and covariance of the posterior accurately to the second order of the Taylor series expansion. The particle filtering formalism can be summarised as follows.

Given the object positions defined as a sequence of states $X^t : t = 0, 1, 2, \dots$ as in Section 2.4.1, the conditional state density $P(X^t|Z^t)$ at time t is represented by a set of samples, or particles, $S^t = \{s_k^t : k = 1, 2, \dots, N\}$ with weights $\{\pi_k^t\}_{k \in N}$, i.e. sampling probability. In other words, the weight π_k^t defines the importance of the sample s_k^t at time t , that is, its observation frequency (Isard and Blake, 1998). For each tuple (s_k, π_k) , a cumulative weight c_k is also stored with $\sum_{k=1}^N c_k = 1$. Subsequently, the new samples at time t are drawn from $S^{t-1} = \{(s_{(k)}^{t-1}, \pi_{(k)}^{t-1}, c_{(k)}^{t-1} : k = 1, \dots, N\}$ at the previous time $t - 1$ step based on the important sampling method as shown in Algorithm 1. It is also worth noting that the particle filter can be initialised with the first measurements of $s_{(k)}^0 = X^0$ and the initial weights $\pi_{(k)}^0 = 1/N$, or by training the system using predefined sample sequences.

Algorithm 1: Particle Filtering Algorithm

From the set of particles at time $t - 1$, $S^{t-1} = \{s_k^{t-1}, \pi_k^{t-1} | k = 1, 2, \dots, N\}$

1. For each particle, the cumulative probability c_k^{t-1} is computed as

$$c_k^{t-1} = c_{k-1}^{t-1} + \pi_k^{t-1}$$

where $k = 1, 2, \dots, N$ and $c_0^{t-1} = 0$. In this way, we have

$$S^{t-1} = \{(s_{(k)}^{t-1}, \pi_{(k)}^{t-1}, c_{(k)}^{t-1} : k = 1, \dots, N\}$$

2. Select M random samples from S^{t-1} (re-sampling) by carrying out the following procedure

- Generate a random number $r \in [0, 1]$ using uniform distribution.
- Find the smallest j such that $c_{(j)}^{t-1} \geq r$.
- The elected state is $\hat{s}_{(k)}^t = s_{(j)}^{t-1}$.

3. For each elected sample $\hat{s}_{(k)}^t$, generate a new sample by using the model $s_{(k)}^t = f(\hat{s}_{(k)}^t, W^t)$ as in Equation 2.6, where $W_{(k)}^t$ is a zero mean Gaussian error and f is a non-negative function, i.e. $f(s) = s$.

4. The weights $\pi_{(k)}^t$ corresponding to the new samples $s_{(k)}^t$ are computed from the measurements z^t by

$$\pi_{(k)}^t = P(z^t | x^t = s_{(k)}^t)$$

where $P(\cdot)$ can be modelled as a Gaussian density.

5. The weight $\pi_{(k)}^t$ is normalised so that $\sum_{k=1}^M \pi_k^t = 1$ and the new particle set S^t can be derived as

$$S^t = \{x_k^t, \pi_k^t | k = 1, 2, \dots, M\}$$

6. Using the new sample set S^t , the new object position at time t can be computed as

$$\epsilon^t = \sum_{k=1}^M \pi_k^t x_k^t$$

Particle filter approaches have recently become popular in the tracking research field, along with its kernel-based counterparts (Comaniciu et al., 2003). The particle filter tracker, in general, can offer great flexibility and can be extended in a number of useful ways. For example, Pérez et al. (2002) employ the particle filter track an

object in the HSV colour space to decouple chromatic information from the shading effects. The colour information is populated into an HS histogram to construct the colour model. The colour model q^t associated with each hypothesis state x^t is then compared with the reference colour model q^* of the object to compute the weight $\pi_{(k)}^{t+1}$ as follows

$$\pi_{(k)}^{t+1} = K \exp \sum_{n=1}^N \lambda \sqrt{q^*(n)q^{t+1}(n; \tilde{x}_k^{t+1})} \quad (2.13)$$

where K is a normalisation constant and $\lambda = 20$ is an arbitrary number. Figure 2.7 shows the results of the proposed method on a video sequence of a ping-pong match, in which the target experiences fast movements and scale variation throughout the course of tracking.

Extending from this method, Yang et al. (2005) use a hierarchical particle filter to perform multiple objects tracking. To do this, the Harr-like features are used together with the colour cues to construct colour and edge orientation histograms. Quasi-random sampling of the particles is used to allow the filter to achieve higher convergence rate. The resulting tracking algorithm maintains multiple hypotheses and offers robustness against clutter or short period occlusions. More recently, Lu et al. (2009) use a boosted version of the particle filter to track multiple hockey players. The method is powered with a new observation model and a template updater to improve robustness. Lu et al. (2009) also use the Histograms of Oriented Gradient (HOG) to represent the players, and introduces a probabilistic framework to model the appearance of the players by a mixture of local subspaces. In addition, multi-view object tracking using particle filter also emerged in recent years with the work in (Nickel et al., 2005; Wang et al., 2007) and (Lien and Huang, 2008).

A potential weakness of the particle filter, however, is that the particle set can become too diffuse, spreading across the image plane rather than clustering around the target. When this happens particles tend to migrate towards local maxima in their evaluation function, becoming caught on clutter and losing track of the true target.

2.4.2 Deterministic Approaches

Deterministic methods, as opposed to the probabilistic ones, define a cost function associating each candidate object in frame t to the last known position of the tracking target in frame $t - 1$. The target correspondence at the current frame t is then computed via an optimisation scheme that aims to minimise this cost function accordingly. In this category, the cost function often arises from a number of a motion constraints and the minimisation process make use of optimal assignments methods such as the

greedy search methods (Sethi and Jain, 1987; Veenman et al., 2001) or the mean-shift procedure (Cheng, 1995; Comaniciu et al., 2003).

Deterministic approaches have been used extensively in single point tracking scenarios, in which the target is represented by a single point. For example, in (Sethi and Jain, 1987), the point correspondence is solved by a greedy approach based on the proximity and rigidity constraints between two consecutive frames. The matching process is initialised by the nearest neighbour criterion. The correspondences are exchanged iteratively to minimise the cost. In a related development, Veenman et al. (2001) extend the work in (Sethi and Jain, 1987) by introducing the common motion constraint for coherent tracking of points to track 80 black dots on a rotating dish. Other notable methods include the work by Shafique and Shah (2005) that tracks the movements of a flock of birds. Here, the birds are detected by a simple background subtraction algorithm and the point correspondence problem is cast into a graph-theoretic setting, in which the temporal coherency of the speed and motion is preserved by the use of multiple frames.

In this section, we will focus on the kernel-based tracking algorithm (Comaniciu et al., 2003) as it has emerged as one of the most robust tracking frameworks in the deterministic branch in recent years. In kernel-based object tracking, the target is typically represented by a template and/or a density-based appearance model derived from an ellipsoidal or a rectangle region on the image. The new position of the object in the current frame is determined by computing the motion of the object from the previous frame. To do this, a deterministic optimisation scheme is employed to compute a spatial vector of motion which indicates the displacement of the object region between the two frames.

2.4.2.1 Kernel-based Tracking

In a nutshell, the kernel-based tracker (Comaniciu et al., 2003), also known as the mean-shift tracker, uses one or more image feature spaces to characterise the target model by constructing a non-parametric *probability density function* (PDF) such as an M -bin histogram. This PDF is further regularised by an isotropic kernel which prioritises pixels that are closer to the target centre than the ones further away. The new target centre is then estimated by applying the mean-shift optimisation scheme (Cheng, 1995) recursively in a search window to ‘shift’ the target centre to a new position at the current frame that yields the highest similarity to the target model. Here, it is worth noting that the specific choice of the kernel profile gives rise to the name *kernel-based tracking*, whereas the mean-shift optimisation procedure is responsible for the name

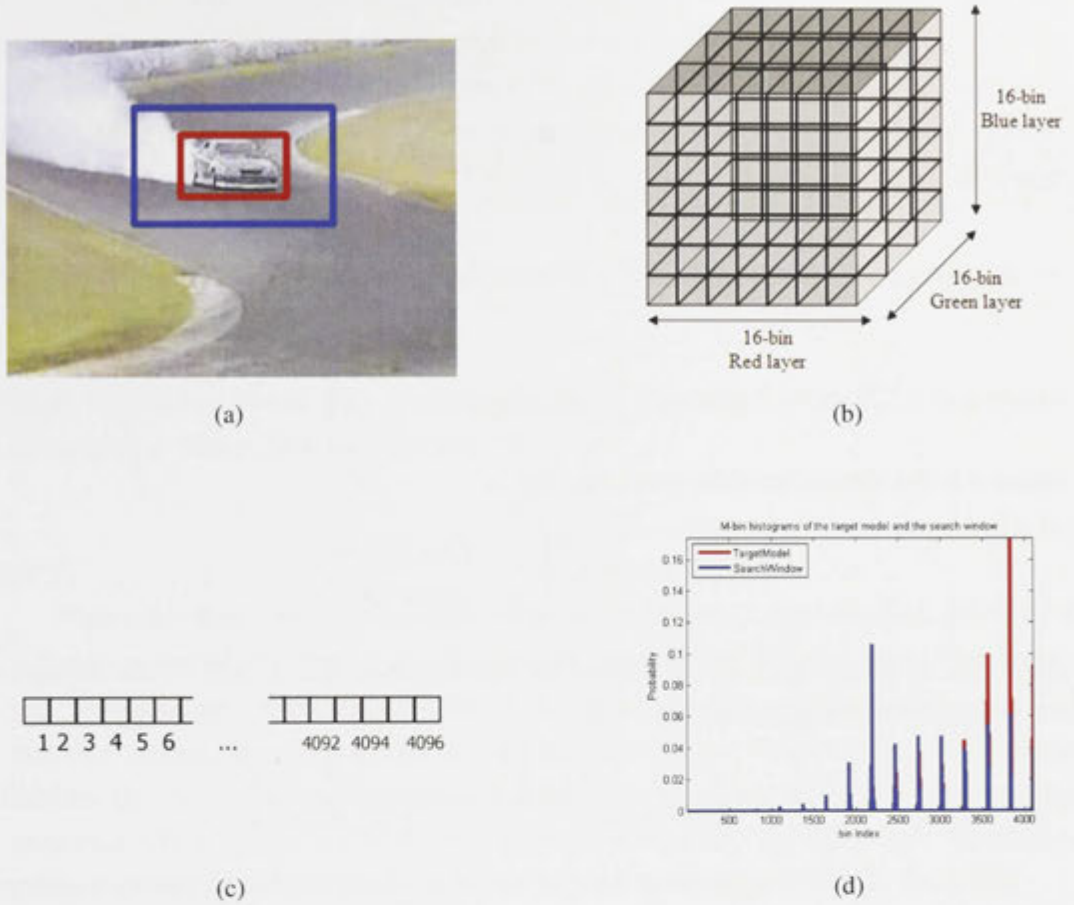


Figure 2.8: Kernel-based object tracking. (a) The target region (red rectangle) and the search window (blue rectangle); (b) A $16 \times 16 \times 16$ -bin histogram viewed as a 3D cube and (c) as a 1×4096 -bin 1D array; (d) The two 1×4096 -bin histograms corresponding to the target region and the search window in (a).

mean-shift tracking. As a result, it is commonly found that both of these names are used interchangeably in the literature of object tracking as well as in the work of this thesis.

In the original kernel-based tracker (Comaniciu et al., 2003), the first instance of the target is selected as a region by the user at the initial frame of the sequence. Depending on the user preference, this region can be encapsulated within a rectangular or ellipsoidal region as shown in Figure 2.8(a). Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1 \dots N}$ and $\mathcal{X}_s = \{\mathbf{x}_i\}_{i=1 \dots N_s}$ be the coordinates of the N -pixels and N_s -pixels in the target region and the search window centered at y , respectively. The function $b : R^n \mapsto \{1, 2, \dots, M\}$ allocates the M bin indexes for each pixel $\mathbf{x}_i$ in the quantised n -dimensional feature space. For

instance, Figure 2.8(b) shows the quantised n -dimensional space as a $16 \times 16 \times 16$ 3D cube corresponding to the RGB cues. In practice, this 3D cube is often converted to its 1D equivalent form, i.e. a 1×4096 -bin array (see Figure 2.8(c)), for better computing efficiency. The probability of the feature indexed $\psi \in M$ in the target model and the candidates are given by

$$\hat{q}_\psi = C \sum_{i=1}^N \mathcal{K}(\mathbf{x}_i, y) \delta[b(\mathbf{x}_i) - \psi] \quad (2.14)$$

$$\hat{p}_\psi = C_s \sum_{i=1}^{N_s} \mathcal{K}(\mathbf{x}_i, y) \delta[b(\mathbf{x}_i) - \psi] \quad (2.15)$$

where δ is the Kronecker delta function, that is

$$\delta[b(\mathbf{x}_i) - \psi] = \begin{cases} 1 & \text{if } b(\mathbf{x}_i) = \psi \\ 0 & \text{otherwise} \end{cases} \quad (2.16)$$

and $\mathcal{K}(\mathbf{x}_i, y)$ is an isotropic kernel function with bandwidth h ; C and C_s are normalisation constants defined as follows

$$C = \frac{1}{\sum_{i=1}^N \mathcal{K}(\mathbf{x}_i, y)} \quad \text{and} \quad C_s = \frac{1}{\sum_{i=1}^{N_s} \mathcal{K}(\mathbf{x}_i, y)} \quad (2.17)$$

With both M -bin histograms at hand, a weight w_i depicting the similarity between the model and the candidate is derived by linearising the Bhattacharyya distance between the two M -bin histograms $\mathbf{Q} = \{\hat{q}_\psi\}_{\psi \in M}$ and $\mathbf{P} = \{\hat{p}_\psi\}_{\psi \in M}$ as follows

$$w_i = \sum_{\psi \in M} \sqrt{\frac{\hat{q}_\psi}{\hat{p}_\psi}} \delta[b(\mathbf{x}_i) - \psi] \quad (2.18)$$

The new target centre position η is then computed using a mean shift iterative optimisation scheme governed by the weighted average

$$\eta = \frac{\sum_{i=1}^{N_s} \mathbf{x}_i \mathcal{G}(\mathbf{x}_i) w_i}{\sum_{i=1}^{N_s} \mathcal{G}(\mathbf{x}_i) w_i} \quad (2.19)$$

where $\mathcal{G}(\cdot) = -\mathcal{K}'(\cdot)$ is also known as the shadow kernel of $\mathcal{K}(\cdot)$. In the original mean-shift tracker (Comaniciu et al., 2003), the Epanechnikov kernel is used as the profile kernel $\mathcal{K}(\cdot)$, that is

$$\mathcal{K}_E(\mathbf{x}_i, y) = \begin{cases} \frac{2}{\pi h^2} (1 - \frac{\|\mathbf{x}_i - y\|^2}{h^2}) & \text{if } \|\frac{\mathbf{x}_i - y}{h}\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.20)$$



Figure 2.9: Results of the mean-shift tracker (Comaniciu et al., 2003) on a sequence of a football match.

Here, the shadow kernel $\mathcal{G}(\cdot)$ can be omitted from Equation 2.19 as $\mathcal{K}'_E(\cdot) = \alpha$, where α is a scalar. Hence, Equation 2.19 is reduced to

$$\eta = \frac{\sum_{i=1}^{N_s} \mathbf{x}_i w_i}{\sum_{i=1}^{N_s} w_i} \quad (2.21)$$

Figure 2.9 shows the results of the mean-shift tracker (Comaniciu et al., 2003) on a video sequence of a football match in which the tracking target is indicated by a rectangle. In this sequence, the tracking target undergoes through severe conditions such as fast blur motion, large displacement and partial occlusion. Despite these confounding factors, the mean-shift tracker manages to track the target successfully throughout the sequence. Here, a $16 \times 16 \times 16$ -bin histogram is used and the mean-shift iteration is computed over a search window of twice the size of the target.

It is also worth stressing that the principle of kernel-based tracking is, in general, not restricted to a set of image feature spaces or a fixed size of the M -bin histogram. Furthermore, the tracker does not require a prior knowledge about the tracking object or the tracking environment, and since it employs only low complexity features, the computational cost is much lower than other probabilistic methods elsewhere in the literature.

2.4.2.2 Literature on Kernel-based Tracking

As introduced in the previous section, the mean-shift tracker is introduced in (Comaniciu et al., 2003) as a robust optimisation method aiming at estimating the translation shift of a sub-image area. Despite being effective, the original method also comes with several drawbacks. Firstly, the algorithm does not specify an automatic mechanism to compute a sufficient bandwidth for the search region, i.e. the size of the search window. Secondly, the method is not robust against non-rigid transformations in scale and orientation. Finally, the tracker relies on the highly correlated RGB colour space to construct the appearance model for the tracking target. This treatment can potentially

lead to mislocalisation when the target approaches other objects sharing similar colour appearance, or in the case where the scene illumination varies significantly.

Along these lines, there have been a number of improved algorithms aiming to tackle the aforementioned problems. Elgammal et al. (2003) cast the tracking framework in a more general form to model joint feature-spatial distributions. Viewed in this manner, the spatial structural information of the target can be utilised to improve the performance of the tracker. Yang et al. (2005) use a fast Gaussian transform to improve on the computation cost of the mean-shift optimisation procedure.

In a related development, Collins (2003) has employed a scale-governed kernel in addition to the spatial one so as to recover the scale of the target. In his paper, Collins showed that the theory of Lindeberg for feature scale selection can be used to solve the problem of selecting the kernel scale for mean-shift blob tracking. In particular, a difference of Gaussian (DoG) kernel is used to deploy the mean-shift tracker in the scale space. This treatment, however, requires modifying the Bhattacharyya weight in Equation 2.18 to a log-likelihood, so as to introduce negative sample weights into the scale-shift formula.

In (Zhang et al., 2005), a quadratic density distance is adopted for tracking objects under affine transformations. The basis of this method is a kernel-based similarity measure called *affine matching* that describes the relationship between image regions with respect to affine transformation parameters. Based on the similarity measure, a mathematical solution is derived for estimating the transformation parameters for moving objects in videos. Fan et al. (2005) build upon (Zhang et al., 2005) and use multiple collaborative kernels to enhance the kernel observability by imposing additional constraints upon the tracking process. They also present a strategy so as to avoid singularities in the optimisation involved in the kernel tracking task.

Furthermore, Hager et al. (2004) also employ multiple kernels so as to address invariance to rotation and scaling. Here, in order to simplify the optimisation procedure, the tracking equation is linearised and solved by a Newton-style iteration. An akin approach, which employs colour-based distributions to estimate not only the position but also the target shape using its histogram covariance matrix, is proposed in (Zivkovic and Krose, 2004). Here, the new method gives rise to a 5-degrees of freedom (DOF) colour histogram, which subsequently is used to perform the tracking task on non-rigid objects. The algorithm, however, shows limited performance on objects that have non-uniform texture appearance or dynamic background.

The methods presented above, amongst many others, are the representative work following the original algorithm. The vast literature of kernel-based object tracking also consolidates the algorithm as a promising approach in the pursuit of robust track-

ing. To date, the kernel-based approach has been a popular alternative to the particle filter-based techniques thanks to its efficiency and robustness.

2.4.2.3 Kernel-based Tracking versus Particle Filter

As mentioned earlier, the particle filter and the kernel-based trackers are the two most popular approaches towards robust tracking in modern computer vision. Each has its own pros and cons. The particle filter is a top-down process which requires a prior knowledge about the target model, and subsequently makes use of a recursive Bayesian filter on a sample set to predict the object location. The particle filter, in general, can offer great flexibility and can be extended in a number of useful ways. However, the computational complexity of this approach is usually quite demanding, especially when additional hypotheses are implemented to improve robustness. Another shortcoming of this approach arises from the fact that the set of particles can sometimes become too diffused, spreading across the image plane rather than clustering around the target, resulting in continuous mislocalisation.

The kernel-based approach, on the contrary, is a bottom-up process which hill climbs towards the new target position on the image lattice via the mean-shift optimisation scheme. The kernel-based method employs low complexity features to construct the target model and, hence, is beneficial to the computational effort. The spatial kernel governing more weights towards pixels at proximity also provides some robustness to noise and partial occlusion. The algorithm can converge relatively fast as long as the target object does not move beyond the search window between two successive frames. The tracking accuracy of the kernel-based approach, however, can be greatly affected by the choice of the bandwidth for the search window. In particular, an oversized search window might cause incorrect convergence due to the excessive amount of noise polluting the candidate histogram. An undersized search window, on the other hand, can substantially trap the mean-shift iteration into a local maxima and hence loses the tracking target, a common problem in the kernel-based approach.

Having reviewed both approaches, we subsequently opt for the mean-shift algorithm as our tracking framework for the following reasons

- Real-time and robust tracking performance can be obtained by using a set of highly discriminant image features, while maintaining a sufficient bandwidth for the search window.
- Flexibility in choice of image features. The mean-shift algorithm can work with a variety of image features including colour, contrast, texture, amongst others.

Additionally, multiple low complexity features can be used simultaneously to enhance robustness thanks to the linearity of the mean-shift operation.

- Flexibility in the choice of kernel profiles. A suitable kernel profile can be targeted at a certain tracking scenario, in which the size of the object, the uniformity of the background and the type of motion are known ahead of time through training or camera configuration.
- Flexibility in the operation domains. In particular, the mean-shift optimisation procedure can be performed on spatial coordinates, space coordinates and polar coordinates.
- The mean-shift algorithm can be deployed in a multi-view setting, in which the target is tracked in a uniformed 3D space and the tracking results can be back-projected to each camera observing the scene.

These aforementioned points suggest the potential headroom for improvements in the mean-shift tracker. Some of which have already been tackled elsewhere in the literature, while others still remain unsolved. In this thesis, we will explore these potential extensions of the mean-shift tracker, so as to develop a robust tracking framework that can cope with appearance deformations, illumination variations and occlusions.

2.5 Multi-camera Tracking

The motivation to use multiple cameras for tracking purposes arises from a number of reasons. Firstly, it is not possible for one camera to provide an adequate coverage of the environment due to its limited field of view (FoV). Multiple cameras provide additional viewpoints to the scene and, as these viewpoints are placed in a related structure, a significantly larger area of coverage can be constructed. This advantage is critical in handling the occlusion problem in comparison to that provided by the single view counterparts. Secondly, the use of multiple cameras also gives rise to an extra dimension, i.e. the depth dimension, and hence the object is tracked in a 3D space. Viewed in this manner, additional trajectory information can be drawn to facilitate the tracker as well as makes possible for other applications such as 3D reconstruction, image stitching and object position tracing.

Multi-camera tracking is still a relatively new research area in computer vision, but one that has gained increasing interest in recent years. Before we present the literature review on this topic, it is necessary to provide some preliminaries for multi-view geometry that are commonly employed by the trackers in this research field.

2.5.1 Multi-view Preliminaries

In this section, we consider the following topics: 2D Homography, Epipolar Geometry and Fundamental Matrix. Note that in multi-view geometry, the points and lines are represented by homogeneous vectors. For example, a line in a plane is represented by an equation such as $ax + by + c = 0$, with different a , b and c giving rise to different lines. Thus, a line can be represented by the vector $(a, b, c)^T$. Similarly, a point $\mathbf{x} = (x, y)^T$ lies on the line $l = (a, b, c)^T$ by satisfying the equation $ax + by + c = 0$, and can be represented by the vector $(x, y, 1)^T$. For more details on the homogeneous coordinate system, we direct the reader to (Hartley and Zisserman, 2004).

2.5.1.1 2D Homography

2D homography is relevant to multi-camera tracking as it provides a mapping between the image points of two cameras that share some common field of view. In mathematics term, a homography is an invertible transformation from one projective plane to another projective plan that maps straight lines to straight lines. In particular, a 2D homography is one that maps the set of points $\mathbf{x}_i$ on the image produced by Camera 1 to another set of points $\mathbf{x}'_i$ on the image produced by Camera 2. That is, if we denote $\mathbf{H}$ to be the homography matrix, the transformation is given by the equation

$$\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i \quad (2.22)$$

Equation 2.22 may be written in terms of vector cross product as $\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = 0$. This form will enable a simple linear solution for $\mathbf{H}$ to be derived.

If the j -th row of the matrix $\mathbf{H}$ is denoted by $\mathbf{h}^{jT}$, then we may write

$$\mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{pmatrix} \quad (2.23)$$

Writing $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^T$, the cross product may then be given explicitly as

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i \mathbf{h}^{3T}\mathbf{x}_i - w'_i \mathbf{h}^{2T}\mathbf{x}_i \\ w'_i \mathbf{h}^{1T}\mathbf{x}_i - x'_i \mathbf{h}^{3T}\mathbf{x}_i \\ x'_i \mathbf{h}^{2T}\mathbf{x}_i - y'_i \mathbf{h}^{1T}\mathbf{x}_i \end{pmatrix} \quad (2.24)$$

Since $\mathbf{h}^{jT}\mathbf{x}_i = \mathbf{x}_i^T \mathbf{h}^j$ for $j = 1, \dots, 3$, this gives a set of three equations in the entries of $\mathbf{H}$, which may be written in the form

$$\begin{bmatrix} 0^T & -w'_i \mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ w'_i \mathbf{x}_i^T & 0^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0 \quad (2.25)$$

Algorithm 2: The Direct Linear Transformation (DLT) algorithm

Data: Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathbf{H}$ such that $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$

1. For each correspondence $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$ compute the matrix $\mathbf{A}_i$ as in Equation 2.25. Only the first two rows will be used in general.
2. Assemble the $n \times 9$ matrices $\mathbf{A}_i$ into a single $2n \times 9$ matrix $\mathbf{A}$.
3. Obtain the SVD of $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ with $\mathbf{D}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathbf{V}$.
4. The matrix $\mathbf{H}$ is determined from $\mathbf{h}$ as in Equation 2.26.

These equations have the form $\mathbf{A}_i\mathbf{h} = 0$, where $\mathbf{A}_i$ is a 3×9 matrix, and $\mathbf{h}$ is a 9-vector made up of the entries of the matrix $\mathbf{H}$,

$$\mathbf{h} = \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix}, \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad (2.26)$$

Although there are three equations in (2.25), only two of them are linearly independent as the third row can be obtained from the sum of x'_i times the first row and y'_i times the second row. It is usual to omit the third row in solving for $\mathbf{H}$, thus Equation 2.25 becomes

$$\begin{bmatrix} 0^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & 0^T & -x'_i\mathbf{x}_i^T \end{bmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0 \quad (2.27)$$

or $\mathbf{A}_i\mathbf{h} = 0$ where $\mathbf{A}_i$ is now a the 2×9 matrix of (2.27).

With this formalism at hand, it is obvious that $\mathbf{H}$ contains nine entries, and hence eight degrees of freedom. As each of the correspondence $\mathbf{x}'_i \leftrightarrow \mathbf{x}_i$ gives rise to two constraints, a minimum of four point-to-point correspondences is required to yield an exact solution for $\mathbf{H}$. In practice, however, it is common that more than four correspondences are given, and as a result, an over-determined solution is required. Such solution can be derived by estimating a vector $\mathbf{h}$ that minimises a suitable cost function, also known as the Direct Linear Transformation (DLT) algorithm (Algorithm 2). The DLT algorithm is a basic way to estimate the homography matrix $\mathbf{H}$ given a set of correspondences between two images. Another variation of the DLT algorithms is the normalised DLT algorithm (Hartley and Zisserman, 2004).

As of lately, modern camera calibration methods often employ auto-homography estimation techniques over the manual insertion of point correspondences. Such a method often involves an automatic feature points detection algorithm that computes

a set of feature points such as SIFT (Lowe, 2004) in each image. A set of point correspondences is then computed based on proximity and similarity of their intensity neighbourhood. This set of correspondences is further refined by the RANSAC algorithm (Schaffalitzky and Zisserman, 2000; Hartley and Zisserman, 2004) to eliminate the outliers and, as a result, the remaining set of inliers is used to estimate the homography $\mathbf{H}$. Automatic homography computation is widely used in modern multi-camera tracking algorithms such as (Khan and Shah, 2003, 2006).

2.5.1.2 Multi-planar Homography

In most cases, the 2D homography matrix $\mathbf{H}$ usually refers to the ground plane between the two viewpoints. However, multiple homography matrices corresponding to successive planes parallel to the ground plane can also be computed using the vanishing point of the reference direction (Khan et al., 2007). In particular, given the homography $\mathbf{H}_{i\pi j}$ induced by a reference plane π between view i and view j , the homography $\mathbf{H}_{i\phi j}$ induced by a plane ϕ parallel to π in the reference direction can be computed as:

$$\mathbf{H}_{i\phi j} = (\mathbf{H}_{i\pi j} + [0|\gamma\mathbf{v}_{\text{ref}}])(I_{3\times 3} - \frac{1}{1+\gamma}[0|\gamma\mathbf{v}_{\text{ref}}]) \quad (2.28)$$

whereas γ is a scalar, $[0]$ is a 3×2 matrix of zeros and $\mathbf{v}_{\text{ref}}$ is the vanishing point of the reference direction. A 3D scene has three vanishing points corresponding to the x , y and z direction. These points can also be computed automatically by detecting straight lines in the scene and finding their intersection in a RANSAC framework as in (Hartley and Zisserman, 2004).

To sum up, the homography matrix plays a critical role in a multi-camera environment. A robust estimation of the homography matrix will guarantee an accurate mapping of the object position between different camera viewpoints. However, it does not provide any 3D information such as the depth of the scene or the relative position of these cameras in the scene. To this end, planar homography methods such as (Khan and Shah, 2006; Khan et al., 2007) have been used to model 3D object from multiple 2D images of the object taken at different viewpoints.

2.5.1.3 Epipolar Geometry

The epipolar geometry between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baselines as axis (the baseline is the line joining the camera centres) as shown in Figure 2.10(a).

To elaborate it further, in Figure 2.10(b), a point $\mathbf{X}$ in 3-space is imaged in two views, at $\mathbf{x}$ in the first and $\mathbf{x}'$ in the second. As a result, the points $\mathbf{x}$, $\mathbf{x}'$, space-point $\mathbf{X}$



Figure 2.10: Epipolar geometry: (a) Family of planes π rotate around the baseline as the 3-space point X varies. (b) An image point x back-projected to a ray in 3-space defined by the first camera centre C , and x . This ray is imaged as a line l' in the second view. The 3-space point X which projects to x must lie on this ray, so the image of X in the second view must lie on l' (Hartley and Zisserman, 2004).

and the two camera centres are coplanar, i.e. they lie on the same plane. Denote this plane as π . It is obvious that the rays back-projected x and x' intersect at the 3-space point X , and they also lie on the same plane π . This property is extremely important as it leads to how to search for a correspondence. That is, in this example, given the image point x from the first camera, the point x' can be constrained on the intersection line of the plane π and the image plane of the second camera, denoted as the line l' in Figure 2.10(b). This is extremely beneficial for stereo-vision as it restrict the searching region for point x' to the line l' . Here, we introduce some terminology used in epipolar geometry:

- The **epipole** is the point of intersection of the line joining the camera centres, i.e. the baseline, with the image plane.
- An **epipolar plane** is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes.
- An **epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

These above properties of epipolar geometry is extremely important to solve point

correspondence in multi-view as it reduces the searching area to a single line. This intrinsic geometry is encapsulated by the fundamental matrix $\mathbf{F}$, which will be discussed in the next section.

2.5.1.4 The Fundamental Matrix

The fundamental matrix $\mathbf{F}$ is an algebraic representation of the epipolar geometry. If a point in 3-space $\mathbf{X}$ is imaged as point $\mathbf{x}$ in the first view, and $\mathbf{x}'$ in the second, then the image points satisfy the relation

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \quad (2.29)$$

The geometric derivation of the fundamental matrix $\mathbf{F}$ can be found in (Hartley and Zisserman, 2004). The properties of the fundamental matrix can be summarised as follows.

- $\mathbf{F}$ is a 3×3 homogeneous matrix of rank 2 with 7 degrees of freedom.
- **Point correspondence:** If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image point, then $\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$
- **Epipolar lines:**
 - ◊ $\mathbf{l}' = \mathbf{F} \mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$
 - ◊ $\mathbf{l} = \mathbf{F}^T \mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$
- **Epipoles:** $\mathbf{F} \mathbf{e} = 0$ and $\mathbf{F}^T \mathbf{e}' = 0$
- **Relation to Homography:** $\mathbf{F} = [\mathbf{e}']_{\times} \mathbf{H}_{\pi}$ where $[\cdot]_{\times}$ is the cross product notation. In this case, $\mathbf{F}$ is the cross product of the epipole $\mathbf{e}'$ and the homography map $\mathbf{H}_{\pi}$.

There are various ways to compute the fundamental matrix $\mathbf{F}$ given a number of point correspondences between two images. Hartley and Zisserman (2004) provide an evaluation on the existing methods such as the MLE “Gold Standard” algorithm and the Sampson distance. In (Hartley and Zisserman, 2004), an automatic computation of the fundamental matrix $\mathbf{F}$ is also presented with the use of the SIFT features (Lowe, 1999) as feature points and the RANSAC sampling process (Fischler and Bolles, 1987) to remove outliers.

2.5.1.5 Preliminary Remarks

In multi-camera tracking, the additional views give rise to a number of advantages such as 3D information and better coverage. In most practical scenarios, these cameras are calibrated to construct a relation in their field of views. A survey on camera calibration techniques can be found in (Hemayed, 2003). These calibration techniques can compute the fundamental matrix F or/and the homography H between cameras and, subsequently, improve the tracking performance by restricting the search region to a line or around a point, respectively.

2.5.2 Literature on Multi-camera Tracking

As mentioned earlier, multi-camera tracking is still a relatively new research area, yet there has been an momentous increase in the amount of work done in this field. Papers on this topic can be organised in two loose categories, namely, feature matching approaches and geometry-based approaches. This section will provide a survey on the representative work done in each category to date.

2.5.2.1 Feature Matching Approaches

The first and probably the most simple approach is to use the colour information of the object to establish consistent labelling of the target across different field of views. Orwell et al. (1999) present a tracking algorithm to track multiple objects in multiple views using colour tracking. They model the connected blobs obtained from background subtraction using colour histogram techniques and use them to match and track objects. Utsumi et al. (2000) propose to use a Kalman filter framework to perform multi-camera tracking from non-synchronized observations. The observations are constructed by segmenting the object out of the environment background. Objects are then matched by minimising the Mahalanobis distance between them. This framework is further extended in (Utsumi and Tetsutani, 2004), where the human head appearance model is used to track multiple human objects. In a related development, Cai and Aggarwal (1999) propose a framework for tracking coarse human models from sequences of synchronised monocular grayscale cameras. In this approach, only the relative calibration between cameras is used and the correspondence is established using a set of feature points in a Bayesian probability framework. The intensity features are constrained to that from the torso of object upward in each projection view, so as to reduce the difference between perspectives. In addition, geometric features such as the physical height of the person are also used to increase the matching accuracy. Kang et al.

(2003) introduced a Kalman Filter framework that uses probabilistic information fusion to track within and across camera streams. The labelling and fusion process can also be done on the Bayesian Belief Networks (BBN) (Dockstader and Tekalp, 2001).

Colour information, however, is highly unreliable due to a number of confounding factors such as: the variation of the scene illumination at different viewpoints; the difference in the internal camera parameters such as focal length and sensor type; and the appearance angle at which the object is exposed to the camera, for instance, a human object might wear a t-shirt with different colour of the front and the back. As a result, different methods have attempted to combine the visual appearance with other geometry features to improve the matching accuracy. For instance, Chang and Gong (2001) use a Bayesian Network to track multiple objects by employing number of image features, together with some calibration constraint to improve robustness. In particular, the features are categorised in two different groups. The first group is geometry-based modalities, which includes epipolar geometry, homography and landmark modalities. The second group is recognition-based modalities containing visual information such as object height and colour modalities.

2.5.2.2 Geometry-based Approaches

This category can be further divided into two subcategories. The first one has the luxury of a calibrated environment in which the camera calibration and the 3D environment are known. In this case, correspondence can be established by projecting the location of the target in the world coordinate system, and subsequently, establishing equivalence of objects that occupy the same position. This is the approach taken in (Kelly et al., 1995), where each camera is calibrated, the world is a known ground plane and the 3D environment model is constructed using voxel feature. Humans were modelled as a collection of voxels in the 3D model to resolve the correspondence problem. In (Sato et al., 1994), a CAD-based environment is constructed to extract the 3D locations of unknown moving objects. Jain and Wakimoto (1995) also use the calibrated camera network to obtain 3D locations of each object in an environment model for the Multiple Perspective Interactive Video. In this subcategory, Tyagi et al. (2007) propose a 3D kernel-based tracking algorithm in a fully calibrated system, i.e. the camera positions and their projection matrices are available prior to the tracking course. To this end, the tracking task is cast into a 3D space, in which the target representation is a visual hull constructed by a voxelisation process and the use of epipolar lines. It is worth noting that a 3D calibrated environment is not common in many practical surveillance systems. These aforementioned approaches, therefore, may be of bene-

fit in controlled environments such as those found in football or tennis stadiums, for which they are developed.

In the second subcategory, the objective is to perform tracking tasks with multiple uncalibrated cameras. Khan and Shah (2003) propose a multiple humans tracking framework based on the computation of the field of view (FoV) lines, i.e. the edges of the FoV of a camera as seen in other cameras. These FoV lines are generally computed by putting the system through a training phase, in which a single object is scripted to move across the cameras. Once the FoV lines are determined, the object can be consistently tracked and labelled in the correspondent trajectories provided by the FoV lines. Calderara et al. (2005) extend the method by Khan and Shah (2003) to solve the problem of false correspondence in the case of a person entering from the bottom of the image in one camera. In such a situation, the head of the object will be recognised as the feet in other view. To resolve this issue, Calderara et al. (2005) propose the use of the Entry Edges of Field of View (E^2oFoV) to solve the consistent labelling problem between overlapped views. The method not only computes the E^2oFoV lines as in (Khan and Shah, 2003) but also exploits the homography to establish the consistent labelling whenever a new object is detected in one of the cameras. A Camera Transition Graph (CTG) is also constructed to speed up the establishment process by reducing the search area. The idea is that whenever a new object is detected in one camera, the system must check to see if the object is already present in other cameras.

More recently, Khan and Shah (2006) present a multi-camera algorithm that can reliably track multiple people in a crowded environment, in which people may be partially or completely occluding each other. This is achieved by a novel use of multiple cameras in synergy so that information from all views is combined to detect objects. In particular, the foreground images of all cameras are firstly achieved by a background subtraction algorithm, in this case the mixture of gaussians method (Stauffer and Grimson, 1999), as shown in the top two row of Figure 2.11. A synergy map is subsequently constructed by warping the all other views into a chosen reference view, making use of the homography matrices between cameras, and multiplying them together. The bottom row of Figure 2.11 shows the synergy map in view 4. The foot positions are then detected by thresholding this synergy map and, subsequently, the tracking task is carried out by performing a normalised-cut segmentation (Shi and Malik, 2000) on a stack of synergy maps corresponding to a window of n frames. The labels associated with each target are then passed to the next stack of n frames to maintain the target identity throughout the course of tracking.

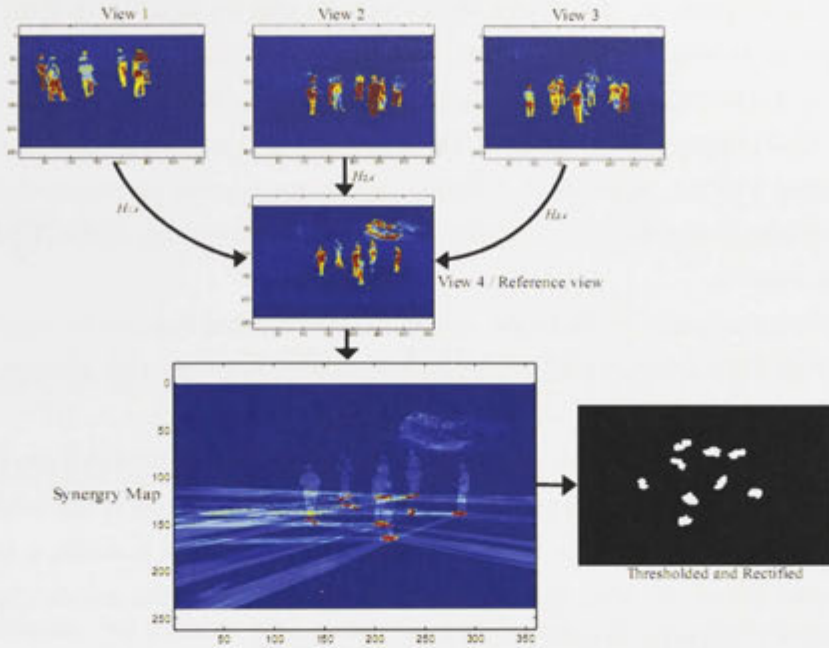


Figure 2.11: Tracking Multiple Occluding People (Khan and Shah, 2006). The first two rows show the foreground likelihood maps obtained from the background model on the available views. View 4 is chosen to be the reference view. The third row shows the synergy map obtained by warping views 1, 2 and 3 into view 4. The pixels representing the ground locations of the people are segmented out by applying an appropriate threshold. The binary image shown is the result of applying the threshold and rectifying with the ground plane (copyright IEEE 2006).

2.6 Concluding Remarks

In this chapter, we have provided the reader with a comprehensive review on the topic of object tracking related to the work done in this thesis. We do this by dividing the tracking task into three main components, namely Object Representations in Section 2.2, Feature Selection in Section 2.3 and Object Tracking Algorithms in Section 2.4 and Section 2.5, addressing the three following questions

- Which object representation is suitable for a given tracking environment.
- Which image feature(s) can best differentiate the object from its surrounding background.
- What strategy is the most effective given the selected features and representations.

For the first question, the object representation can be based either on primitive shapes such as rectangle, ellipse, points, or by appearance models such as probability density functions, templates or active appearance models. Regardless of which representation model being used, in general, there is a strong connection between the object representation and the application domain, for instance, point representation is often used for bird tracking whereas primitive shapes such as rectangle and ellipse are used for human tracking.

For the second question, there are various image features that can be used in object tracking, with the most popular one being colour features, followed by texture-based features such as the SIFT (Lowe, 1999), Harr-like (Viola and Jones, 2001) and LBP (Ojala et al., 1996). While colour features are easy to compute and relatively faster than other features, they often produce sub-optimal performance if the target moves close to another one that shares similar colour appearance. As a result, a handful of trackers in the literature either use texture-based features over the colour cues, and/or a combination of features is used to improve robustness.

For the last question, due to vast literature on object tracking algorithms, we have only presented the representative methods related to the work of this thesis. To this end, we attempt to classify object tracking methods into two categories, namely the probabilistic approach and the deterministic approach. In this manner, we review the representative algorithms of each field, and focus on the particle filter tracker (Pérez et al., 2002) and the kernel-based tracker (Comaniciu et al., 2003) as the two benchmark algorithms for the probabilistic and deterministic categories, respectively. A comparison of the two approaches is presented in Section 2.4.2.3, along with our elaborations on the rationale for choosing the kernel-based approach as the base tracking vehicle.

To extend the framework further, we have also provided a brief survey on multi-camera tracking. Even though this research field is relatively new, it is one that has gained momentous attention from the tracking community in recent years. Here, we present the reader with some preliminaries of multi-view geometry in Section 2.5.1, followed by a review on the most recent achievements in this research area in Section 2.5.2.

In this thesis, we endeavour to explore the use of the kernel-based tracker to solve several challenging problems in tracking, namely object transformations in scale and orientation; colour variations due to the changes in illumination of the scene; and object occlusions including both partial and total occlusions. We do this by presenting a number of advanced methods and extensions for the kernel-based tracker such as the use of multiple coordinate systems, multiple image features and multiple cameras.

Chapter 3

Kernel-based Tracking From A Probabilistic Viewpoint

In this chapter, we present a probabilistic formulation of the kernel-based tracking algorithm to solve the challenging problems of target transformations in scale and orientation. To this end, we view the coordinates for the pixels in both, the target model and its candidate as random variables and make use of a generative model so as to cast the tracking task into a maximum likelihood framework. Viewed in this manner, we make use of the EM-algorithm to estimate a set of latent variables that can be used to update the new target centre position. Furthermore, we show that the algorithm is very general in nature and can be deployed with a variety of kernel profiles and coordinate systems, so as the scale and orientation of the tracking target can be updated simultaneously during the tracking course. We also illustrate the robustness of our approach on real-world video sequences by providing quantitative comparison with a number of competing algorithms, to which our algorithm has shown consistent lead in tracking performance throughout the experiments.

3.1 Maximum Likelihood Formulation

As mentioned earlier, we aim at developing a probabilistic formulation for the mean-shift tracker proposed by Comaniciu, Ramesh, and Meer (2003). Here, we commence by viewing the pixel-coordinates for the model and the candidate regions as random variables in a quantised feature space. Viewed in this way, these random variables can be used to obtain probability distributions for purposes of tracking. These two distributions, corresponding to the model and the target, are, for the sake of consistency, constructed in the same manner. Thus both regions are treated equally for purposes of

developing the likelihood function for the tracking process.

Recall that, in Section 2.4.2.1, the coordinates $\mathcal{X} = \{\mathbf{x}_i\}_{i=1\dots N}$ of the N -pixels define either the target model or candidate regions. Here, we consider the coordinates $\mathcal{X}$ as random variables. For these random variables, the function $b: R^n \mapsto \{1, 2, \dots, M\}$ is a mapping which assigns the random variables $\mathbf{x}_i$ to a set of M mutually exclusive intervals $\{\mathcal{W}_j\}_{j=1\dots M}$ in the quantised n -dimensional feature space $\mathcal{W}$.

For tracking purposes, we employ a distribution of $\mathcal{X}$ which is defined with respect to the coordinates y of the centre pixel for the tracking region in the current frame. It is important to note that the independent observations in $\mathcal{W}$ are not recorded, but rather the number N_j of them that “fall” in $\mathcal{W}_j$ together with their coordinates in the image lattice. That is, individual observations are made in the n -dimensional feature space $\mathcal{W}$, but only their class intervals $\mathcal{W}_j$ and coordinates are available. This is an important observation, since it suggests a quantisation process in which the probability density functions of the target model and candidate are governed by “frequency” variables which capture the probability of a given observation being in each of the class intervals.

Since our N independent observed random variables are given by the coordinates of those pixels in $\mathcal{W}$, the first moment, or average value, of $\mathcal{X}$ is y by construction. That is,

$$y = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \quad (3.1)$$

Similarly, the second moments are given by the squared distances

$$\mu_2[\mathbf{x}_i] = \langle \mathbf{x}_i - y, \mathbf{x}_i - y \rangle \quad (3.2)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Note that the second moments of $\mathcal{X}$ are also given by the quantity $E[(\mathcal{X} - y)^2]$, where $E[\cdot]$ is the expectation operator. These relations are important since they permit, in the following sections, to cast the tracking task as that of estimating the updated coordinates of the centre pixel for the tracking region making use of the second moments of $\mathcal{X}$.

From the theoretical point of view, the use of the second moments as defined above guarantees that the kernel-based tracking process is effected in a Hilbert space. The consequences of this are twofold. Firstly, this opens-up the possibility of tracking in arbitrary coordinate spaces where metrics can be defined in terms of inner products. Secondly, it introduces, by definition, the requirement that the kernel function be a product of two probability density functions. We elaborate further on this in the following section.

3.1.1 Kernels and Probabilities

Following the previous section, we can now formulate the tracking problem as that of obtaining an improved estimate of the target coordinates y by making use of distribution of second moments for the target model and candidate. We can do this, by viewing the expectation of the distribution of second moments $\mu_2[\mathcal{X}]$ given a class interval as a predictor of the first moment bias, which governs the probability distribution that gives rise to the structure of the random variables in $\mathcal{X}$.

Furthermore, the fact that the second moment $\mu_2[\mathbf{x}_i]$ can be formulated as an inner product, permits the construction of a kernel function $\mathcal{K}$. For purposes of analysis, consider the class of kernels given by $\mathcal{K}(\mathbf{x}_i, y) = P(\mathbf{x}_i)P(y)$ where $P(\cdot)$ is a probability density function. We can extend these kernels by taking sums over products of weighted probability distributions (Bishop, 2006). We get

$$\mathcal{K}(\mathcal{X}, y) = \sum_{j=1}^M P(\mathcal{X} | h_j) P(y | h_j) P(h_j) \quad (3.3)$$

where we have used conditional probabilities as an alternative to the density functions and introduced the set of parameters $\{h_j\}_{j=1, \dots, M}$ that correspond to the bandwidths for the class intervals $W_j \in \mathcal{W}$. It is worth noting in passing that the formalism above applies to a number of generative models such as Gaussians, multinomials and exponential families (Jebara, 2004).

In this manner, the kernel $\mathcal{K}(\mathcal{X}, y)$ can be viewed as a function governed by the bandwidth and the second moments $\mu_2[\mathcal{X}]$ of $\mathcal{X}$. Thus, the kernel $\mathcal{K}(\mathcal{X}, y)$ can be viewed as a family of mixture distributions of the form

$$\begin{aligned} P(\mu_2[\mathcal{X}]) &= \sum_{j=1}^M \pi_j P(E[(\mathcal{X} - y)^2] | h_j) \\ &= \sum_{j=1}^M \pi_j P(\mu_2[\mathcal{X}] | h_j) \end{aligned} \quad (3.4)$$

where $\pi_j = P(h_j)$ is the mixture weight. Thus the marginal for the distribution of second moments with respect to the class interval is

$$P(\mu_2[\mathcal{X}])_{W_j} = \frac{P(E[(\mathcal{X} - y)^2] | h_j)}{\sum_{k=1}^M \pi_k P(E[(\mathcal{X} - y)^2] | h_k)} = \frac{P(\mu_2[\mathcal{X}] | h_j)}{P(\mu_2[\mathcal{X}])} \quad (3.5)$$

which can be rewritten, given Equations 3.3 and 3.4 as follows

$$P(\mu_2[\mathcal{X}])_{W_j} = \frac{\sum_{\mathbf{x}_i \in W_j} \mathcal{K}(\mathbf{x}_i, y)}{\sum_{k=1}^M \sum_{\mathbf{x}_i \in W_k} \mathcal{K}(\mathbf{x}_i, y)} \quad (3.6)$$

where, as expected, $\mathcal{K}(\mathcal{X}, y)$ is given by the sum over all the random variables $\mathbf{x}_i$ in $\mathcal{X}$. The relevance of the relations above will become evident when we consider a number of arbitrary kernel profiles later in this chapter. For now, we are concerned with a generative model for the probability distribution of $\mu_2[\mathcal{X}]$ which captures whether or not the moments for the observed random variables belong to a class interval W_j .

3.1.2 Joint-likelihood Function

In this section, we adopt a model in which the observed second moment probability distribution arises through a set of Bernoulli trials. This is a reasonable assumption since the class intervals W_j are mutually exclusive. As a result, each random variable $\mathbf{x}_i$ in $\mathcal{X}$ corresponding to the coordinates for the pixels in either the model or the candidate have a joint-likelihood function in the class interval W_j which is binomial in nature and governed by the probability $P(\mu_2[\mathcal{X}])_{W_j}$ and a parameter q_j . The parameter q_j can then be viewed as the probability of “success”, i.e. the probability the random variable $\mathbf{x}_i$ is indeed in $\mathcal{X}$, whereas its complement $1 - q_j$ is the probability of “failure” for the Bernoulli trail. With these ingredients, the joint-likelihood is given by

$$P(q_j|\phi_j) = \phi_j^{q_j} (1 - \phi_j)^{(1-q_j)} \quad (3.7)$$

where $\phi_j = P(\mu_2[\mathcal{X}])_{W_j}$ is, as mentioned above, given by the distribution of the second moments $\mu_2[\mathcal{X}]$ in the class interval W_j .

The treatment above is important since it opens-up the possibility of including unobservable variables for the intervals W_j making use of the likelihood $P(q_j|\phi_j)$. Moreover, we can view the parameters q_j as frequency variables for the class intervals W_j . This is, the parameter q_j can be related to the normalised histogram bins in the quantised space $\mathcal{W}$. Recall that, in Section 3.1, we employed the function $b : R^n \mapsto \{1, 2, \dots, M\}$ so as to allocate the bin index for each pixel in the quantised n -dimensional feature space. These imply that, for each class interval W_j , there corresponds a normalised frequency q_j for the pixels in the tracking region. By viewing this frequency variable as the parameter q_j in Equation 3.7, we can give an intuitive interpretation to the likelihood $P(q_j|\phi_j)$. Consider the case in which all the pixels in the tracking region correspond to the class interval W_i . This yields $q_i = \phi_i = 1$ and $q_j = \phi_j = 0 \forall j \neq i$. Then we have $P(q_i|\phi_i) = 1$ and zero otherwise. In this manner, the likelihood of the normalised frequency in the class interval indexed j is a binomial distribution whose probability of success is the conditional probability of the second moments given W_j . Here we view the frequency variables q_j , i.e. the parameters of

the binomial distribution above, as hidden variables that should be estimated using the EM-algorithm.

3.1.3 Expectation-Maximisation

In this section, we focus on finding the expected values ϕ_j and the unobservable frequencies q_j which maximize the likelihood function appearing in Equation 3.7. This could be effected using a number of optimisation methods including simulated annealing and Markov Chain Monte Carlo. However, here we use the apparatus of the EM algorithm originally developed by Dempster, Laird, and Rubin (1977). Our reason for doing this is that the unobservable frequencies q_j must be regarded as quantities whose distribution is governed by the moment-set structure. Since at the outset we know neither the unobservable frequencies nor the optimal value for the probability of success for the Bernoulli trial, this information must be treated as hidden data. In other words, we must use the EM algorithm to estimate them.

The idea underpinning the EM algorithm is to recover maximum likelihood solutions to problems involving missing or hidden data given a set of sample data, by iterating between two computational steps. In the E (or expectation) step we estimate the posteriori probabilities of the hidden data. The M-step in-turn aims to recover the parameters which maximise the expected value of the log-likelihood function. It is the available a posteriori probabilities from the E-step which allows the weighting of log-likelihood required in the maximisation-step. In this case, the set of sample data is drawn from the variables $\mathbf{x}_i \in \mathcal{X}$ corresponding to the coordinates for the pixels in either the model or the candidate.

To commence, we consider the log-likelihood function for the generative model in the previous section. For the likelihood function appearing in Equation 3.7, the expected log-likelihood function is given by

$$\mathcal{L}(\phi_j, q_j) = q_j \ln(\phi_j) + (1 - q_j) \ln(1 - \phi_j) \quad (3.8)$$

Performing algebra and collecting terms, we have

$$\mathcal{L}(\phi_j, q_j) = q_j \ln \left(\frac{\phi_j}{1 - \phi_j} \right) + \ln(1 - \phi_j) \quad (3.9)$$

3.1.3.1 E-Step

With the log-likelihood at hand, we compute the expectation of the hidden data by making use of a gradient-based analysis of the log-likelihood function. This constitutes

the E-step of the algorithm, in which, we commence by computing the derivatives of the expected log-likelihood function with respect to the frequency variables. That is

$$\frac{\partial \mathcal{L}(\phi_j, q_j)}{\partial q_j} = \ln \left(\frac{\phi_j}{1 - \phi_j} \right) \quad (3.10)$$

Note that since the associated saddle-point equations are not tractable in closed form, we use the soft-assign ansatz of Bridle (1990) to update the cluster membership assignments variables. This is a form of naive mean field theory (Ghahramani and Jordan, 1997). According to mean field theory the latent variables should be updated by replacing them with their expected values (Hofmann and Buhmann, 1997). Rather than performing the detailed expectation analysis, soft-assign allows the cluster memberships to be approximated by exponentiating the partial derivatives of the expected log-likelihood function. The updated frequency variables are given by

$$\hat{q}_j = \frac{\exp \left[\frac{\partial \mathcal{L}(\phi_j, q_j)}{\partial q_j} \right]}{\sum_{j=1}^M \exp \left[\frac{\partial \mathcal{L}(\phi_j, q_j)}{\partial q_j} \right]} = \frac{\frac{\phi_j}{1 - \phi_j}}{\sum_{j=1}^M \frac{\phi_j}{1 - \phi_j}} \quad (3.11)$$

3.1.3.2 M-Step

With the updated frequency variables at hand, the update of the random variable ϕ_j is a straightforward task. To maximise the log-likelihood, we calculate the derivatives of $\mathcal{L}(\phi_j, q_j)$ with respect to ϕ_j and equate the result to zero, i.e. we solve

$$\frac{\partial \mathcal{L}(\phi_j, q_j)}{\partial \phi_j} = 0 \quad (3.12)$$

It is somewhat surprising that, after some algebra and collection of terms, we find that the log-likelihood is maximised when $\hat{\phi}_j = \hat{q}_j$. Recall that $\phi_j = P(\mu_2[\mathcal{X}])_{W_j}$. Hence, the maximum likelihood estimate for the probability of the moments in the class interval W_j is given by its frequency variable.

3.2 Minimising Mutual Information

Thus far, we have focus on the probability distributions and likelihoods for the target model and candidate rather than their mutual relationship. For tracking, we are interested in updating the coordinates $\hat{y}$ of the model-centre pixel given the random variables corresponding to the candidate. In this section, we show how this can be effected by making use of an information theoretic criterion.

Following the notation above, let the two sets of random variables for the model and the candidate be given by $\mathcal{X}_M$ and $\mathcal{X}_D$, respectively. Similarly, the second moments for the two sets of random variables are denoted $\mu_2[\mathcal{X}_M]$ and $\mu_2[\mathcal{X}_D]$. It is straightforward to show that, if the two sets of random variables $\mathcal{X}_M$ and $\mathcal{X}_D$ are equivalent, then the Kullback-Leibler divergence between the distributions of their second moments over the feature space $\mathcal{W}$ is equal to zero. Further, since the intervals $W_j \in \mathcal{W}$ are mutually exclusive, we can write

$$KL(P(\mu_2[\mathcal{X}_M])||P(\mu_2[\mathcal{X}_D])) = \sum_{W_j \in \mathcal{W}} P(\mu_2[\mathcal{X}_M]_{W_j}) \ln \left(\frac{P(\mu_2[\mathcal{X}_M]_{W_j})}{P(\mu_2[\mathcal{X}_D]_{W_j})} \right) \quad (3.13)$$

The advantages of the formulation above are twofold. Firstly, we can exploit the fact that, for our generative model in the previous section, the maximum likelihood estimate for the expectation of the second moments of $\mathcal{X}_M$ and $\mathcal{X}_D$ is given by their corresponding frequency variables. Secondly, it permits the use of the physical interpretation of the moments to update the coordinates y by recovering the shift in y that minimises the divergence in Equation 3.13. Note that, so far, we have worked with the second moments of the observed random variables instead of the variables themselves. As a result, we have not, so far, included the coordinates y implicitly, but rather used them to recover “distances” that we can employ for purposes of inference. As a result, we can update the target centre coordinates by using the rule

$$\hat{y} = y + y^* \quad (3.14)$$

where y^* is the deviation from y that minimises the Kullback-Leibler divergence between the model and the candidate moment distributions.

To minimise the divergence in Equation 3.13, we note that the updated variables $\hat{\phi}_j$ are the maximum likelihood estimates of the expectation values in Equation 3.7. Moreover, from Equation 3.6, we can write

$$\hat{\phi}_j = \frac{\sum_{\mathbf{x}_i \in W_j} \mathcal{K}(\mathbf{x}_i, \hat{y})}{\sum_{\mathbf{x}_k \in \mathcal{W}} \mathcal{K}(\mathbf{x}_k, \hat{y})} = \frac{\sum_{\mathbf{x}_i \in W_j} f(\hat{\mu}_2[\mathbf{x}_i], h)}{\sum_{\mathbf{x}_k \in \mathcal{W}} f(\hat{\mu}_2[\mathbf{x}_k], h)} \quad (3.15)$$

where we have written $\hat{\mu}_2[\mathbf{x}_i]$ to imply that the second moments above correspond to updated variables of the form

$$\begin{aligned} \hat{\mu}_2[\mathbf{x}_i] &= \langle \mathbf{x}_i - \hat{y}, \mathbf{x}_i - \hat{y} \rangle \\ &= \langle (\mathbf{x}_i - y) - y^*, (\mathbf{x}_i - y) - y^* \rangle \end{aligned} \quad (3.16)$$

and used the shorthand $f(\hat{\mu}_2[\mathbf{x}_i], h) = \mathcal{K}(\mathbf{x}_i, \hat{y})$ to stress that the kernel is here a function of the second moments and the bandwidth h . Note that this notation is consistent

with that used in section 3.1.1. By substituting Equations 3.15 and 3.16 into Equation 3.13, we get

$$KL(P(\mu_2[\mathcal{X}_M])||P(\mu_2[\mathcal{X}_D])) = \frac{1}{\sum_{x_k \in \mathcal{W}} f(\hat{\mu}_2[x_k], h)} \sum_{W_j \in \mathcal{W}} \left\{ \sum_{\mathbf{x}_i \in W_j} f(\hat{\mu}_2[\mathbf{x}_i], h) \ln \left(\frac{\varphi_i}{\psi_i} \right) \right\} \quad (3.17)$$

where φ_i is the estimated frequency variable $\hat{q}_j$ for the second moments in the target model random variables $\mathcal{X}_M$ corresponding to the interval W_j (see Equation 3.11). The value of ψ_i is, in turn, the estimated frequency variable $\hat{p}_j$ for the second moments of the candidate random variables $\mathcal{X}_D$ given the class interval W_j . As the mean-shift operation is performed in the candidate region $\mathcal{X}_D$, we can recover the minimum of the equation above with respect to y^* by differentiating and equating to zero. By manipulating terms, this yields

$$\gamma \sum_{\mathbf{x}_i \in \mathcal{X}_D} \beta_i \frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} \frac{d\hat{\mu}_2[\mathbf{x}_i]}{dy^*} = 0 \quad (3.18)$$

where γ is a proportionality constant given by $\frac{1}{\sum_{x_k \in \mathcal{W}} f(\hat{\mu}_2[x_k], h)}$ and $\beta_i = \ln \left(\frac{\varphi_i}{\psi_i} \right)$. By using Equation 3.16 to compute the differential of the second moment with respect to y^* and some algebra, we get

$$y^* = \frac{\sum_{\mathbf{x}_i \in \mathcal{X}_D} (\mathbf{x}_i - y) \frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} \beta_i}{\sum_{\mathbf{x}_i \in \mathcal{X}_D} \frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} \beta_i} \quad (3.19)$$

and, hence, the updated target centre coordinates become

$$\hat{y} = y + \frac{\sum_{\mathbf{x}_i \in \mathcal{X}_D} (\mathbf{x}_i - y) \frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} \beta_i}{\sum_{\mathbf{x}_i \in \mathcal{X}_D} \frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} \beta_i} \quad (3.20)$$

Note that the negative derivative of $f(\cdot)$ in the above equations is defined as the shadow of the kernel $f(\cdot)$ in kernel methods (Cheng, 1995). Here, the new target centre position is governed by the quantity $\beta_i = \ln \left(\frac{\varphi_i}{\psi_i} \right)$. This quantity can be seen as a weight defined in terms of the marginal probability of the feature indexed $u \in M$ in the target model. Considering the function $b : R^n \mapsto \{1, 2, \dots, M\}$, which allocates the M bin indexes for each pixel in the target model and candidate regions. We can view the marginal probabilities as two sets of variables indexed to the feature space bins. Thus, can express the marginal probability for the model as follows

$$r_j = C_r \sum_{\mathbf{x}_i \in \mathcal{X}_M} \mathcal{K}(\mathbf{x}_i, y) \delta[b(\mathbf{x}_i) - j] \quad (3.21)$$

where δ is the Kronecker delta function, $\mathcal{K}(\cdot)$ is the kernel function and C_r is a normalisation constant defined as follows

$$C_r = \frac{1}{\sum_{\mathbf{x}_i \in \mathcal{X}_M} \mathcal{K}(\mathbf{x}_i, y)} \quad (3.22)$$

After successfully calculating the variables r_j for the target model, the marginal probabilities for the target candidate are computed as follows

$$s_j = C_s \sum_{\mathbf{x}_i \in \mathcal{X}_D} \mathcal{K}(\mathbf{x}_i, y) \delta[b(\mathbf{x}_i) - j] \quad (3.23)$$

where, again, y is the centre pixel of the current tracking region and C_s is a normalisation constant given by

$$C_s = \frac{1}{\sum_{\mathbf{x}_i \in \mathcal{X}_D} \mathcal{K}(\mathbf{x}_i, y)} \quad (3.24)$$

Given the two sets of variables r_j and s_j , it is a straightforward task to compute the weight β_i as follows

$$\beta_i = \sum_{u=1}^M \ln \left(\frac{\frac{r_u}{1-r_u} \sum_{j=1}^M \frac{s_j}{1-s_j}}{\frac{s_u}{1-s_u} \sum_{j=1}^M \frac{r_j}{1-r_j}} \right) \delta[b(\mathbf{x}_i \in \mathcal{X}_D) - u] \quad (3.25)$$

With the equation above at hand, we can proceed to obtain the update rules for the target position using Equation 3.20.

Furthermore, it is worth noting that the weight $\beta_i = \sum_u \ln(\cdot)$ is computed making use of the Kullback-Leibler divergence between the target model and the candidate regions. This treatment differs from the original method (Comaniciu et al., 2003) where the pixel weights $\beta_i = \sum_u \sqrt{\frac{q_\psi}{p_\psi}} \delta[\cdot]$ arise from the Bhattacharyya distance. The Kullback-Leibler divergence weight hence introduces negative weights into the mean-shift operation. This is important since it opens-up the possibility of “shifting” the target in different domains where “negative” weights can be practically useful such as “shifting” in the scale and orientation domains, which will be discussed in following sections.

3.3 Implementation

Having presented the theoretical foundations for our method in the previous sections, we now focus on the implementation of our algorithm for the scale, orientation and spatial domains. To this end, we use the image coordinate system for the spatial mean-shift to compute the target centre position in the current frame and, simultaneously,



Figure 3.1: An elliptical region covering the tracking target. The orientation and size of the ellipse can be computed directly from the covariance matrix of the pixel set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1\dots N}$.

make use of the polar coordinate system to estimate the target scale and orientation. To do this, the target is represented by an ellipse parameterised by a centre pixel x , a tilting angle θ , a major axis a and a minor axis b as shown in Figure 3.1.

Recall that, in the previous section, the spatial coordinates $\mathcal{X} = \{\mathbf{x}_i\}_{i=1\dots N}$ of the N pixels in either the target model or candidate region is viewed as random variables. Here, given the initial target pixels $\mathcal{X}_0$, an elliptical region covering the target can be computed directly from $\mathcal{X}_0$, which in-turn gives rise to the initial orientation and size of the target. In particular, the centre of the ellipse is also the target centre, i.e. $\bar{x} = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{X}_0} \mathbf{x}_i$. The covariance matrix of the spatial coordinates $\mathcal{X}_0$ is defined as follows

$$C_{\mathcal{X}_0} = \sum_{\mathbf{x}_i \in \mathcal{X}_0} E[(\mathbf{x}_i - \bar{x})(\mathbf{x}_i - \bar{x})^T] \quad (3.26)$$

Note that $C_{\mathcal{X}_0}$ is a 2×2 matrix, of which we can compute the eigenvalues matrix λ and eigenvectors matrix Π , i.e. $C_{\mathcal{X}_0} \Pi = \lambda \Pi$, which, subsequently, can be used to compute the initial orientation angle $\bar{\theta}$ and the lengths of the two axes $\bar{a}$ and $\bar{b}$ as follows

$$\begin{aligned} \bar{\theta} &= \arctan \frac{\Pi_{1,1}}{\Pi_{1,2}} \\ |\bar{a}| &= \sqrt{\text{diag}(\lambda(1))} \\ |\bar{b}| &= \sqrt{\text{diag}(\lambda(2))} \end{aligned} \quad (3.27)$$

With the initial set of parameters defining the target ellipse at hand, we can proceed to formulate the mean-shift procedure in the polar coordinates. For every iteration, the

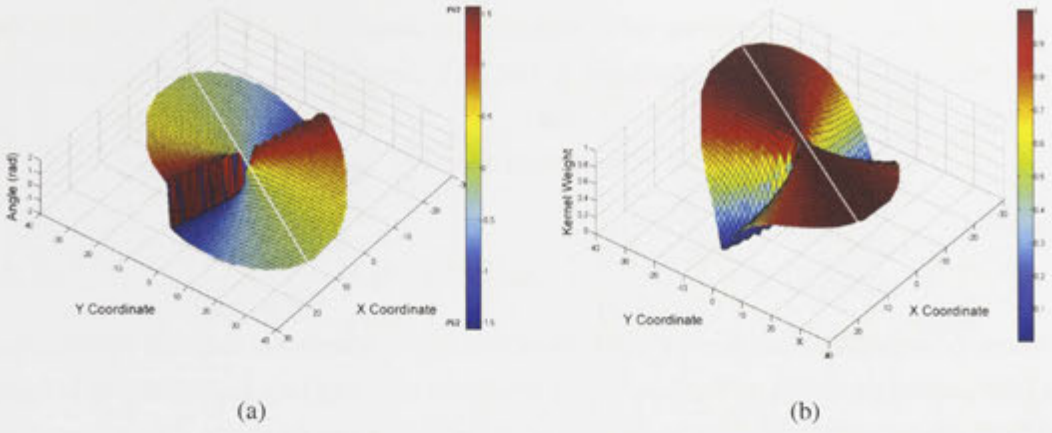


Figure 3.2: Left: The target ellipse is translated to angle dimension. Right: The kernel weights corresponding to the angles (pixels closer to the major axis have more weight).

changes in these parameters will serve twofold. In particular, the change in the angle of the target ellipse θ gives rise to the new orientation of the target, while the change in the axes a and b governs the new target scale.

3.3.1 Automated Orientation Estimation

The automated orientation selection scheme presented here can be viewed as a 1D mean-shift in the angle feature space, with respect to the mean angle $\bar{\theta}$ derived from the previous section. To this end, the binning process for all pixels in the target ellipse is similar to that of the original mean-shift tracker (Comaniciu et al., 2003). In particular, the mapping function $b : R^n \mapsto \{1, 2, \dots, M\}$ assigns each pixel in the target ellipse a bin-number based on its RGB values.

Here, the kernelisation process governed by the kernel profile $\mathcal{K}(\cdot)$, which associates a spatial weight with each pixel based on the Euclidean distance to the target centre, requires some adjustments since we are performing the mean-shift operation on the angle coordinates θ , as opposed to the xy coordinates. The key point of this formulation lies in the partition of the target ellipse as shown in Figure 3.2(a). Here, we have divided the target ellipse into two halves corresponding to positive angle gain and negative angle loss, respectively. This hinges in the notion that the first half of the ellipse “pushes” the ellipse along the clock-wise direction, whereas the second half “pulls” the ellipse in the anti-clockwise direction and, in this formulation, the centre of the ellipse serves as the equilibrium point of rotation.

In mathematics term, we commence by defining that, pixels belong to the first and

third quadrants will have negative coordinates, i.e. $\theta < 0$, whereas pixels belong to the second and forth quadrants have positive coordinates, i.e. $\theta \geq 0$. To do this, the mapping function assigning an angle θ_i for each pixel $\{\mathbf{x}_i\}_{i=1\dots N}$ can be defined as follows

$$\theta_i = \arctan \frac{\mathcal{F}(\mathbf{x}_i, \bar{a})}{\mathcal{F}(\mathbf{x}_i, \bar{b})} \quad (3.28)$$

where

$$\mathcal{F}(\mathbf{x}_i, \bar{a}) = \frac{\langle \mathbf{x}_i - \bar{x}, \bar{a} \rangle}{|\mathbf{x}_i - \bar{x}| |\bar{a}|} \quad \text{and} \quad \mathcal{F}(\mathbf{x}_i, \bar{b}) = \frac{\langle \mathbf{x}_i - \bar{x}, \bar{b} \rangle}{|\mathbf{x}_i - \bar{x}| |\bar{b}|} \quad (3.29)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Note that this is consistent with the formulation of the second moments in Section 3.1, in which the tracking task is guaranteed in coordinate spaces where metrics can be defined in terms of inner products. Once the spatial set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1\dots N}$ is translated to the polar set $\{\theta_i\}_{i=1\dots N}$, the M -bin histogram, for the sake of simplicity, can be constructed using the 1D Epanechnikov kernel:

$$\mathcal{K}(\theta_i, \bar{\theta}) = \begin{cases} 1 - \frac{(\theta_i - \bar{\theta})^2}{h^2} & \text{if } \left\| \frac{(\theta_i - \bar{\theta})}{h} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.30)$$

where the bandwidth $h = \pi/2$ as shown in Figure 3.2(a) and (b). Furthermore, Figure 3.2(b) shows how this kernel profile applies more weights towards pixels that are closer to the mean angle $\bar{\theta}$ and less weights for the pixels further away, in the polar coordinate system. The M -bin histograms of the target model and the candidate regions, subsequently, can be computed as follows

$$\hat{q}_\psi = C \sum_{\theta_i \in \mathcal{X}_M} \mathcal{K}(\theta_i, \bar{\theta}) \delta[b(\theta_i) - \psi] \quad (3.31)$$

$$\hat{p}_\psi = C_s \sum_{\theta_i \in \mathcal{X}_D} \mathcal{K}(\theta_i, \bar{\theta}) \delta[b(\theta_i) - \psi] \quad (3.32)$$

where C and C_s are the normalisation constants defined as

$$C = \frac{1}{\sum_{\theta_i \in \mathcal{X}_M} \mathcal{K}(\theta_i, \bar{\theta})} \quad \text{and} \quad C_s = \frac{1}{\sum_{\theta_i \in \mathcal{X}_D} \mathcal{K}(\theta_i, \bar{\theta})} \quad (3.33)$$

With the two M -bin histograms of the target model and the candidate regions at hand, the new angle of the target ellipse is can be computed making use of the 1D mean-shift procedure in the angle space. That is

$$\hat{\theta} = \bar{\theta} + \frac{\sum_{\theta_i \in \mathcal{X}_D} (\theta_i - \bar{\theta}) \beta_i}{\sum_{\theta_i \in \mathcal{X}_D} \beta_i} \quad (3.34)$$

where $\beta_i = \ln \left(\frac{\psi_i}{\psi_i} \right)$. It is worth noting that the shadow kernel $\mathcal{G}(\cdot) = -\mathcal{K}'(\cdot)$ is omitted from Equation 3.34 due to the fact that the derivative of the 1D Epanechnikov kernel is a scalar, i.e. $\mathcal{K}_E(\cdot) = \alpha$, where α is a scalar.

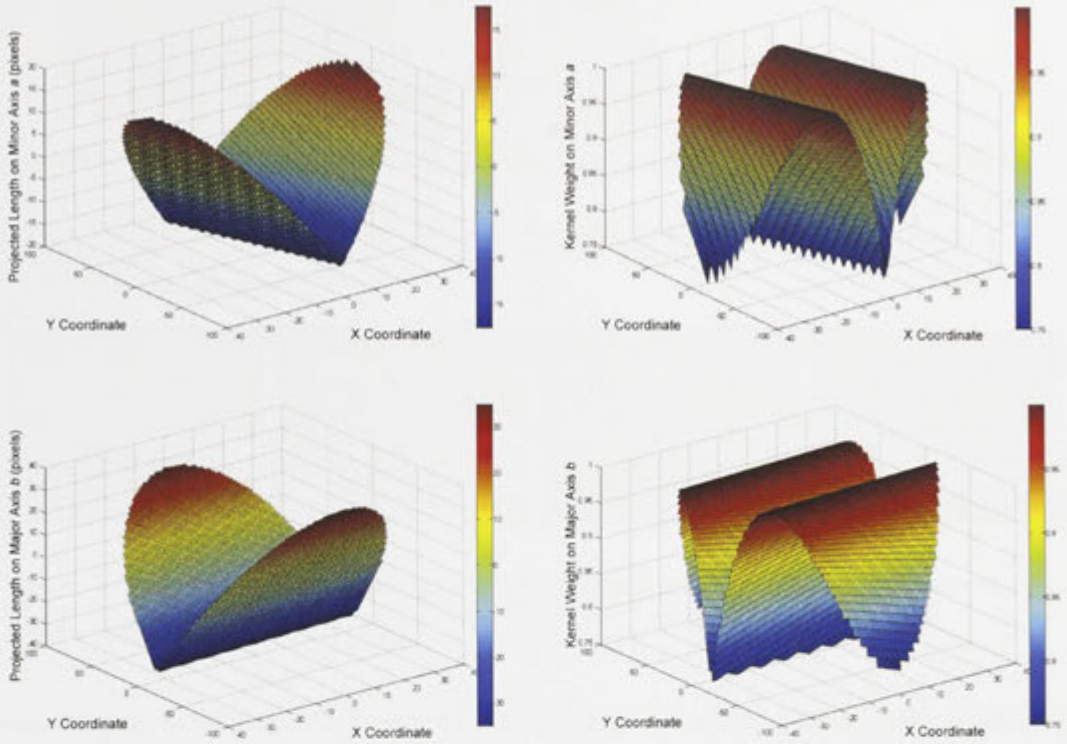


Figure 3.3: Left column: The target ellipse is translated to $\bar{a}$ and $\bar{b}$ dimension. Right column: The kernel weights corresponding to the projected lengths (pixels closer to the midpoints on each axis have more weight).

3.3.2 Automated Scale Estimation

In the previous section, we have utilised the angle information of the ellipse to perform the mean-shift operation in the angle space to cope with orientation variation. Here, we proceed to formulate an automated scale selection scheme making use of the major and minor axis of the target ellipse. In particular, each axis can be cast into an 1D mean-shift procedure which aims to find the ellipse that best fits the target in the current frame. This treatment not only leads to the best scale of the object in the current frame, but also is able to cope with possible sheer transformations the object might experience, a common problem in human tracking.

With that in mind, the scale mean-shift can be formulated as a combination of two 1D mean-shifts on the major and minor axis of the target ellipse whose initial values are $\bar{a}$ and $\bar{b}$, respectively. Similarly to the angle mean-shift in the previous section, it is required that the spatial coordinates $\mathcal{X} = \{\mathbf{x}_i\}_{i=1\dots N}$ are translated into the axis-coordinates corresponding to the major and minor axes of the target ellipse. We do this

by making use of the equilibrium points that divide each axis in half, i.e.

$$a' = \frac{\bar{a}}{2} \quad \text{and} \quad b' = \frac{\bar{b}}{2}$$

As a result, each coordinate $\{\mathbf{x}_i\}_{i=1\dots N}$ can be projected on the major and minor axes as follows

$$\Delta_{ai} = \frac{\langle x_i - \bar{x}, \bar{a} \rangle}{|\bar{a}|} \quad (3.35)$$

$$\Delta_{bi} = \frac{\langle x_i - \bar{x}, \bar{b} \rangle}{|\bar{b}|} \quad (3.36)$$

where Δ_a and Δ_b correspond to the direction of the major axis $\bar{a}$ and minor axis $\bar{b}$ respectively. The M -bin histograms in $\bar{a}$ and $\bar{b}$ dimensions can be constructed using the 1D Epanechnikov kernel

$$\mathcal{K}(\Delta_{ai}, a') = \begin{cases} 1 - \frac{(\Delta_{ai} - a')^2}{h^2} & \text{if } \|\frac{\Delta_{ai} - a'}{h}\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.37)$$

where the bandwidth $h = \bar{a}/2$. Figure 3.3 shows how this kernel $\mathcal{K}(\cdot)$ assigns more weight for the pixels that are closer to the equilibrium point on each axis. Subsequently, the M -bin histograms of the model and candidate regions can be computed as follows

$$\hat{q}_\psi = C \sum_{\Delta_i \in \mathcal{X}_M} \mathcal{K}(\Delta_{ai}, a') \delta[b(x_i) - \psi] \quad (3.38)$$

$$\hat{p}_\psi = C_s \sum_{\Delta_i \in \mathcal{X}_D} \mathcal{K}(\Delta_{ai}, a') \delta[b(x_i) - \psi] \quad (3.39)$$

where C and C_s are the normalisation constants defined as follows:

$$C = \frac{1}{\sum_{\Delta_i \in \mathcal{X}_M} \mathcal{K}(\Delta_{ai}, a')} \quad \text{and} \quad C_s = \frac{1}{\sum_{\Delta_i \in \mathcal{X}_D} \mathcal{K}(\Delta_{ai}, a')} \quad (3.40)$$

With these ingredients, the new axis of the ellipse is then estimated making uses of the 1D mean-shift, that is

$$\hat{\Delta}_{ai} = a' + \frac{\sum_{\Delta_{ai} \in \mathcal{X}_D} (\Delta_{ai} - a') \beta_i}{\sum_{\Delta_{ai} \in \mathcal{X}_D} \beta_i} \quad (3.41)$$

$$\hat{\Delta}_{bi} = b' + \frac{\sum_{\Delta_{bi} \in \mathcal{X}_D} (\Delta_{bi} - b') \beta_i}{\sum_{\Delta_{bi} \in \mathcal{X}_D} \beta_i} \quad (3.42)$$

where $\beta_i = \ln\left(\frac{\varphi_i}{\psi_i}\right)$. Finally, Figures 3.4 and 3.5 illustrate the effectiveness of the mean-shift presented above on two real-world sequences. In particular, Figure 3.4

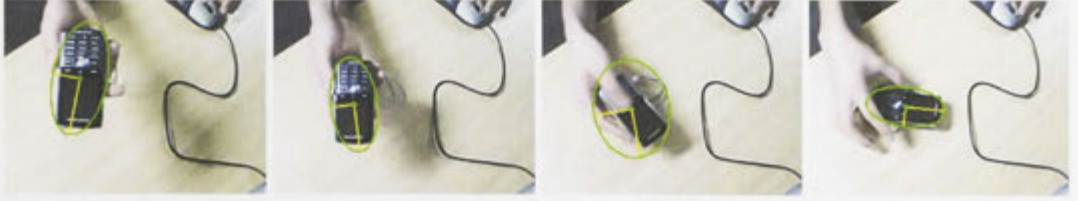


Figure 3.4: Spatial, orientation and scale mean-shift with the “Phone” sequence. Extracted frames: 6, 71, 91, and 106.

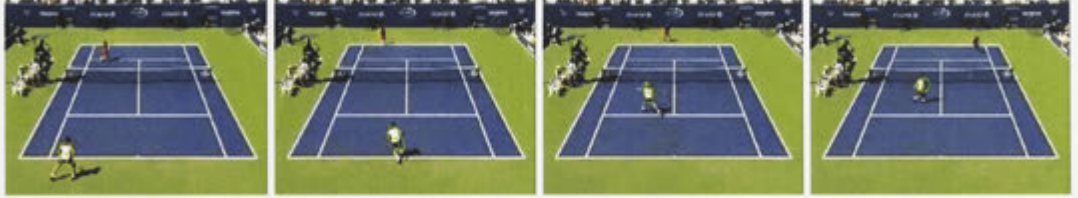


Figure 3.5: Spatial, orientation and scale mean-shift with the “Tennis” sequence. Extracted frames: 1, 29, 67, and 105.

depicts a scene of a mobile phone undergone some severe transformations in scale and orientation, while Figure 3.5 shows a tennis game in which the tracking player moves quickly from the back to the front of the court. Despite these challenging factors, the mean-shift algorithm presented here accurately tracks both targets, in both spatial, orientation and scale domains. It is worth noting that, in both sequences, the target ellipse is expanded and contracted in both axes simultaneously and independently, as shown in frame 91 of the “Phone” sequence and frame 105 of the “Tennis” sequence.

3.3.3 Tracking with Arbitrary Kernel Profile

Recall that, in Section 2.4.2.1, the specific choice of the kernel profile gives rise to the name *kernel-based tracking*, whereas the mean-shift optimisation scheme is responsible for the name *mean-shift tracking*. In fact, the mean-shift tracker is not restricted to a specific kernel profile $\mathcal{K}(\cdot)$, but, on the contrary, can be used with a variety of kernel profiles such as the Epanechnikov, the diffusion, the Tukey and the Hampel profiles (Li, 2001). Figure 3.6 illustrates the effects of different kernel profiles on a target ellipse presented in the previous sections. Here, we have plotted the pixel weights derived by applying the Tukey kernel profile, the diffusion kernel profile and the Hampel kernel profile on the target ellipse. Note that, despite the difference in the shape of these kernel profiles, they share a similarity in governing more weight for pixels that are closer to the centre of equilibrium, or in particular, the centre pixel of the ellipse for

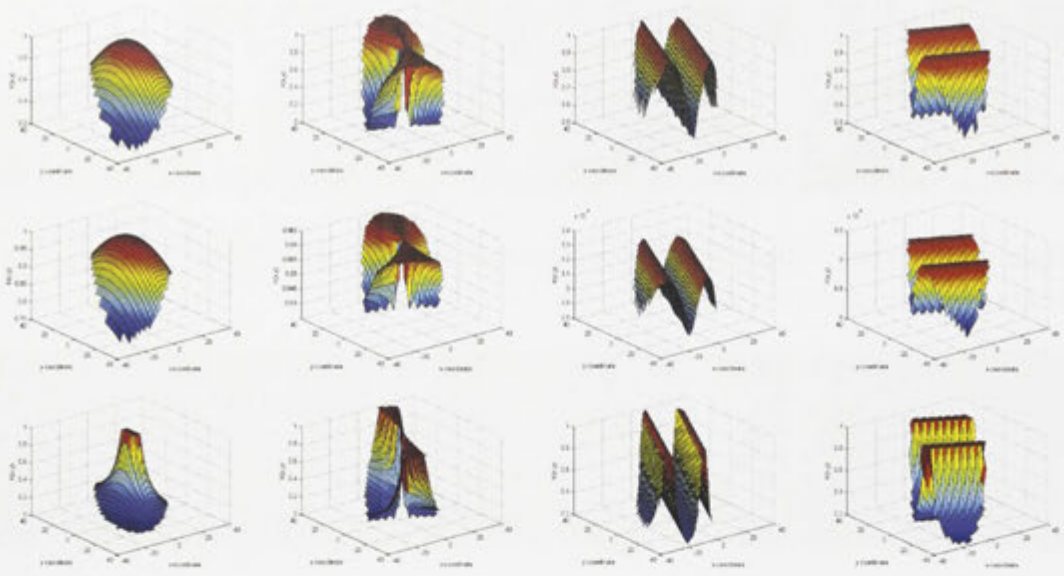


Figure 3.6: Different kernel profile $\mathcal{K}(\mathbf{x}, y)$ of the target ellipse. From left to right: $\mathcal{K}(\mathbf{x}, y)$ in the image coordinates; $\mathcal{K}(\theta, \bar{\theta})$ in the polar coordinates; $\mathcal{K}(\Delta_a, a')$ and $\mathcal{K}(\Delta_b, b')$ as the projection on the major and minor axes, respectively. From top to bottom: Tukey kernel; diffusion kernel and Hampel kernel, respectively.

the spatial mean-shift, the major axis of the target ellipse for the orientation mean-shift and the midpoints of the major and minor axes for the scale mean-shift. In this section, we focus on the two most popular kernel profiles used in kernel-based methods, namely the Epanechnikov and the diffusion kernels.

3.3.3.1 Epanechnikov Kernel

In (Comaniciu et al., 2003), the profile kernel $\mathcal{K}(\mathcal{X}, y)$ for each pixel $\{\mathbf{x}_i\}_{i=1\dots n}$ is defined as a function of the form

$$\mathcal{K}(\mathbf{x}_i, y) = \begin{cases} 1 - g(\mathbf{x}_i, y)^2 & \text{if } \|g(\mathbf{x}_i, y)\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.43)$$

where $g(\mathbf{x}_i, y)$ is a monotonic function which, for tracking purposes is defined with respect to the centre pixel y of the tracking region in the current frame. Thus, for rectangular tracking regions centered at y of width w and height z the square of $g(\cdot)$ is typically given by

$$g(x_i, y)^2 = \frac{4\|y - \mathbf{x}_i\|^2}{w^2 + z^2} \quad (3.44)$$

where $h^2 = w^2 + z^2$ is the bandwidth parameter as before. As a result, the function $f(\mu_2[\mathcal{X}], h)$ is given by

$$f(\mu_2[\mathbf{x}_i], h) = \begin{cases} 1 - \frac{4\mu_2[\mathbf{x}_i]}{h^2} & \text{if } \left\| \frac{4\mu_2[\mathbf{x}_i]}{h^2} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.45)$$

For the function above, the derivatives with respect to the second moments $\hat{\mu}_2[\mathbf{x}_i]$ are

$$\frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} = \begin{cases} -\frac{4}{h^2} & \text{if } \left\| \frac{4\hat{\mu}_2[\mathbf{x}_i]}{h^2} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.46)$$

Hence, making use of Equations 3.20 and after some algebra, the update rule for the target centre becomes

$$\hat{y} = y + \frac{\sum_{\mathbf{x}_i \in \mathcal{X}_D} (\mathbf{x}_i - y) \beta_i}{\sum_{\mathbf{x}_i \in \mathcal{X}_D} \beta_i} \quad (3.47)$$

3.3.3.2 Diffusion Kernel

Having presented the update rules for the Epanechnikov kernel, we now focus in the analogue equations for the diffusion kernel in Cheng (1995). This profile kernel is defined as a function of the form

$$\mathcal{K}(\mathbf{x}_i, y) = \begin{cases} \lambda \exp(-g(\mathbf{x}_i, y)^2) & \text{if } \|g(\mathbf{x}_i, y)\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.48)$$

where $\lambda = \frac{1}{2\pi h^2}$ and $g(\mathbf{x}_i, y)$ is a function of the target centre pixel y and the pixel coordinates $\mathbf{x}_i$. The function $g(\cdot)$ is typically given by

$$g(x_i, y)^2 = \frac{\|y - \mathbf{x}_i\|^2}{2h^2} \quad (3.49)$$

where h is the bandwidth parameter. Thus, the function $f(\mu_2[\mathcal{X}], h)$ is

$$f(\mu_2[\mathbf{x}_i], h) = \begin{cases} \frac{1}{2\pi h^2} \exp\left(-\frac{\mu_2[\mathbf{x}_i]}{2h^2}\right) & \text{if } \left\| \frac{\mu_2[\mathbf{x}_i]}{2h^2} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.50)$$

Similarly, the partial derivative of $f(\hat{\mu}_2[\mathbf{x}_i], h)$ with respect to $\hat{\mu}_2[\mathbf{x}_i]$ becomes

$$\frac{\partial f(\hat{\mu}_2[\mathbf{x}_i], h)}{\partial \hat{\mu}_2[\mathbf{x}_i]} = \begin{cases} -\frac{1}{4\pi h^4} \exp\left(-\frac{\hat{\mu}_2[\mathbf{x}_i]}{2h^2}\right) & \text{if } \left\| \frac{\hat{\mu}_2[\mathbf{x}_i]}{2h^2} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.51)$$

In contrast with Equation 3.46, the expression above depends on the moments $\hat{\mu}_2[xi] = ||\mathbf{x}_i - \hat{y}||^2$, which are governed by our aim of computation, i.e. the variable $\hat{y}$. Furthermore, due to the exponential term involved, the evaluation of the associated update rule becomes non-tractable in closed form. Rather than making a detailed analysis, we note that, if the shift in the target centre coordinates is small, we can consider the moments $\hat{\mu}_2[xi] = ||\mathbf{x}_i - \hat{y}||^2$ to be approximately equal to $\mu_2[xi] = ||\mathbf{x}_i - y||^2$. By substituting Equation 3.51 into Equation 3.20 and using $\mu_2[\mathbf{x}_i]$ as an alternative to $\hat{\mu}_2[\mathbf{x}_i]$, we get

$$\hat{y} = y + \frac{\sum_{\mathbf{x}_i \in \mathcal{X}_D} (\mathbf{x}_i - y) \exp\left(-\frac{\mu_2[\mathbf{x}_i]}{2h^2}\right) \beta_i}{\sum_{\mathbf{x}_i \in \mathcal{X}_D} \exp\left(-\frac{\mu_2[\mathbf{x}_i]}{2h^2}\right) \beta_i} \quad (3.52)$$

With the above formulations at hand, we have effectively generalised the kernel-based tracking algorithm in (Comaniciu et al., 2003), so as the optimisation procedure can be performed in the scale and orientation spaces, along with the spatial space, as well as with various kernel profile functions. We do this by viewing the coordinates for the pixels in both the target model and candidate regions as random variables and, subsequently, cast the tracking task into a maximum likelihood framework where the updated spatial, scale and orientation positions of the target can be updated simultaneously in each iteration using the EM-algorithm. With that in mind, we can now proceed to compare the tracking performance of our algorithm to other competing alternatives in the following section.

3.4 Experiments

In this section, we illustrate the robustness of our tracking algorithm by presenting the results on two surveillance footages from the PETS-ECCV 2004 database¹. The first sequence depicts a scene, in which a person, the tracking target, walks in, falls down and remains immobile for a period of time. In this sequence, the outfit of the target is quite distinctive to that of the background and the orientation of the target changes negligibly across the scenes. The second sequence poses a much more challenging scenario, in which a target approaches to an ATM while wearing an outfit that has the lower part colour similar to that of the background. During the footage, the target experiences both scale and orientation variation with significant discrepancies. Both sequences are recorded at a standard resolution of 384×288 . More importantly, they are supplied with ground truth data, including the coordinates of the target centroid

¹PETS dataset can be accessed from <http://www.cvg.rdg.ac.uk/slides/pets.html>

and the target bounding box in each frame, which subsequently makes the performance evaluation extremely precise and intuitive.

We have compared our results to those yielded by three competing algorithms elsewhere in the literature. The first of the alternatives is the EM-like mean-shift tracker proposed by Zivkovic and Krose (2004)², which shares with our approach the capability of estimating both target position and scale. The other two alternatives are the particle filter-based tracker described in (Pérez et al., 2002) and the original mean-shift tracker in (Comaniciu et al., 2003). The mean-shift tracker by (Comaniciu et al., 2003) is tuned to vary the target scale in steps of $\pm 10\%$ and select the one with the largest Bhattachayya coefficient. For the particle filter, the state space is given by the target position and scale. Following (Pérez et al., 2002), we have adopted the HSV colour histogram with 110 bins for our implementation of the particle filter. In all the experiments, we have set the number of particles to 600 and chose the standard deviation of the Gaussian noise to the value which provides the best results for each sequence. For our tracker, we have opted for the RGB colour intensities as the feature space, along with the $16 \times 16 \times 16$ -bin colour histogram as the step size for the quantisation process. We have also used two different kernel profile functions, the Epanechnikov and the Diffusion kernels, to directly evaluate the generalisation of the algorithm.

In all experiments, the initial position and scale of the tracking target are set to that of the ground truth. All of the trackers are set to run through the two sequences frame by frame, with tracking data recorded into corresponding text files. It is worth noting that the ground truth scale of the target is provided as a rectangular bounding box covering the target. The EM-like tracker (Zivkovic and Krose, 2004), the mean-shift tracker (Comaniciu et al., 2003) and our tracker, on the other hand, employ an elliptical contour as the tracking envelop. The target scale for these trackers, therefore, is calculated as the area of the smallest rectangle that covers the ellipse. We then assess the performance of these trackers against the provided ground truth basing on two basics: the differences in target centre position and the target area. The detailed numerical results are plotted in Figure 3.8 and Figure 3.10, as well as tabulated in Table 3.1 and Table 3.2.

In Figure 3.7, we presents the results for the frames 600, 800, 900 and 1000 of the PETS-ECCV 2004 “Falling” sequence. Here, we track person that appears to lose his balance when approaching to the middle of the scene. The person subsequently falls down and remains immobile for 10 seconds, then recovers and walks away. The supplied ground truth is placed at the top-most row of the figure, followed by the results of

²For our experiments, we have used the code available at the author’s website. The code can be downloaded from <http://staff.science.uva.nl/~zivkovic/>

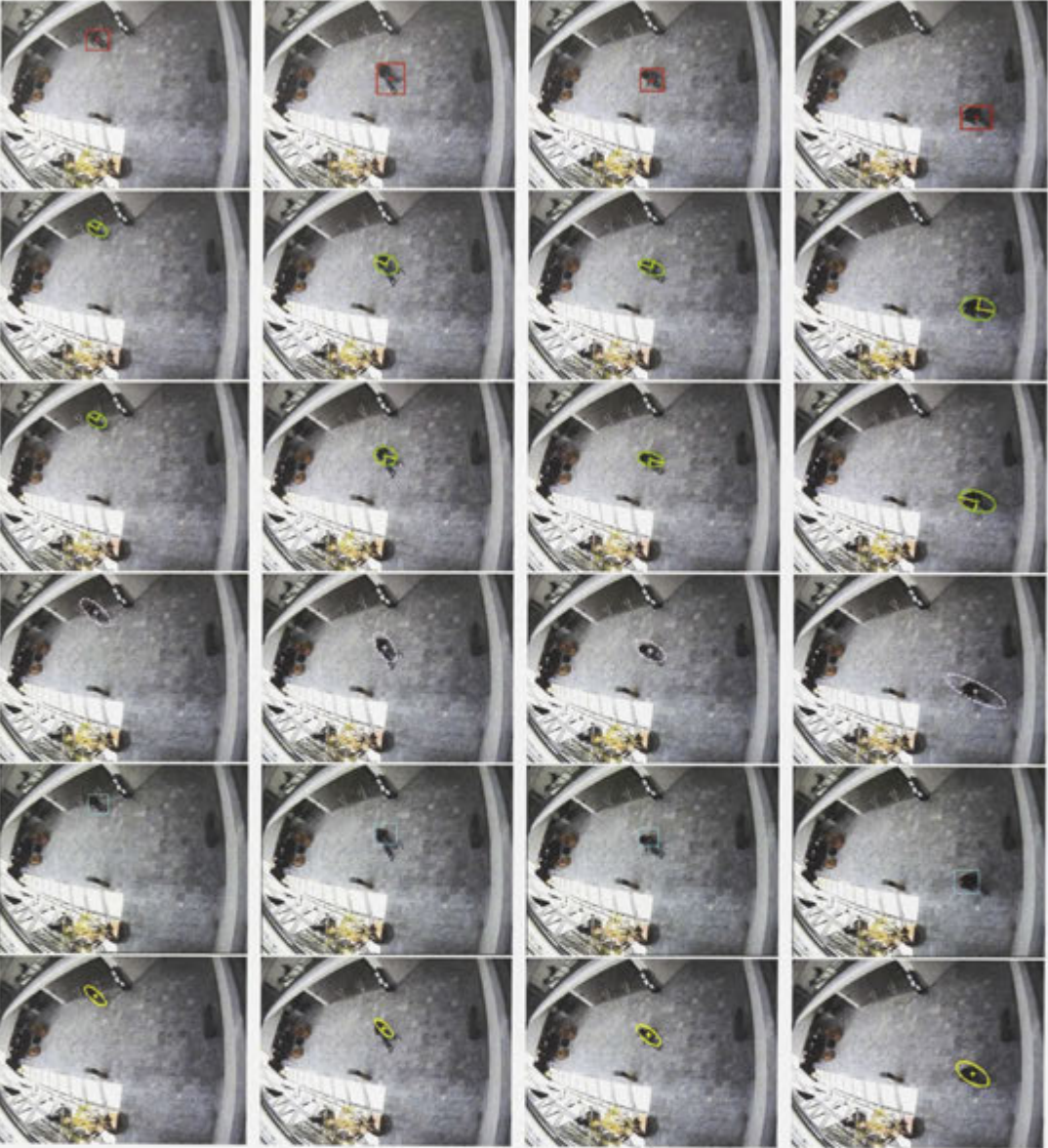


Figure 3.7: Results for the “Falling” sequence at frames 600, 800, 900 and 1000. From top to bottom: Ground truth, results yield using the Epanechnikov and the Diffusion Kernels, EM-like Shift (Zivkovic and Krose, 2004), Particle filter (Pérez et al., 2002) and Mean-shift (Comaniciu et al., 2003).

the Epanechnikov and Diffusion Kernels in the second and third rows. The forth, fifth and sixth rows show the results of the EM-like shift (Zivkovic and Krose, 2004), Particle filter (Pérez et al., 2002), and the mean-shift (Comaniciu et al., 2003), respectively. In this sequence, the outfit of the target is quite distinctive to that of the background. As a result, all the trackers under study manage to track the target throughout the scene

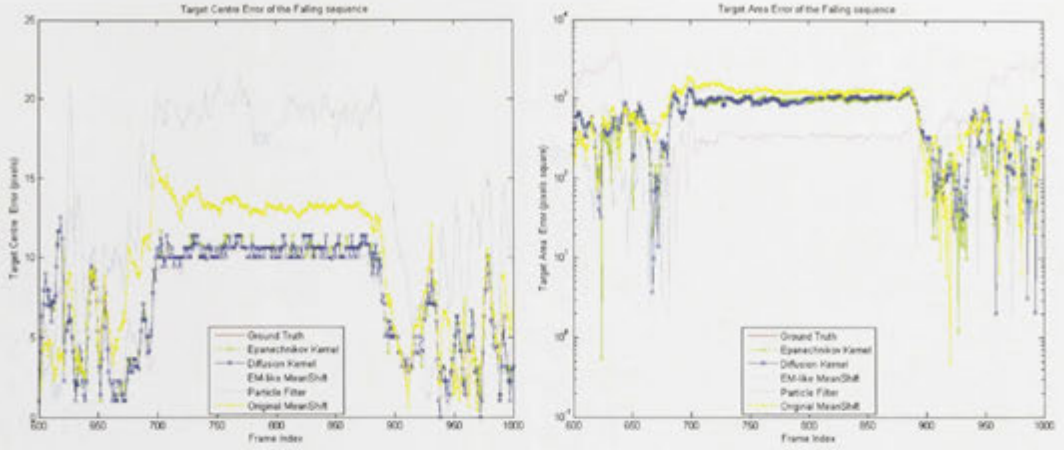


Figure 3.8: Left: Target centre square error, Right: Target area square error of the “Falling” Sequence.

with ease, except the particle filter tracker shows some mislocalisation between frames 700 and 900 in which the target falls down and hence, undergoes some moderate appearance deformations.

The above elaborations become more evident in Figure 3.8. In particular, we have evaluated the performance of the trackers under study making use of two numerical measurements. The left graph plots the target centre error in pixel distance of each tracker against the ground truth, while the right graph shows the target area error in pixels square. In the target centre error graph, the implementations of our tracker with the Epanechnikov kernel and Diffusion kernel score the lowest target centre of 7.50 ± 3.41 and 7.41 ± 3.42 pixels, respectively. This is followed by the EM-like mean-shift, the original mean-shift and the particle filter with the target centre errors of 7.51 ± 3.23 , 9.47 ± 4.25 and 13.97 ± 5.98 pixels, respectively. Furthermore, Figure 3.8 shows that mislocalisation for all four trackers is worst from frame 700 to 900, when the tracking object falls down and experiences some moderate degree of deformation.

In the scale error graph, we have employed the logarithmic scale to better illustrate the pixel square error for the target area, thus the closer the tracker scores toward the line 10^0 , the better the tracker performs. In this sequence, the EM-like mean-shift (Zivkovic and Krose, 2004) yields the best scale approximation between frame 700 and 900. The EM-like tracker, however, also gives the most inaccurate scale estimation during the first and last 100 frames. This inconsistency in performance of the EM-like mean-shift (Zivkovic and Krose, 2004) results in a large target area error of 821.74 ± 910.47 pixel square. The original mean-shift (Comaniciu et al., 2003), on the other hand, shows a poor performance in estimating the scale of the target, resulting the

highest target area error of 839.11 ± 501.32 pixel square. The particle filter (Pérez et al., 2002) shows a better performance with a target area error of 729.44 ± 507.50 pixel square. Our trackers, in this case, scores the lowest target area errors of 647.53 ± 382.33 and 684.87 ± 384.16 pixel square, for the Epanechnikov and the Diffusion kernel profiles, respectively. We have also tabulated the above results in Table 3.1. It is worth noting that, in the scale error measurements, we have squared up the target area error and its standard deviation, hence, resulting in these high values.

Moving on to the more challenging scenario, in Figure 3.9, we present the example results for the frames 400, 500, 700, and 800 of the PETS-ECCV 2004 “Browse” sequence. Here, the target lower outfit has similar colour to that of the background. Additionally, the target experiences both scale and orientation deformation as compared to only scale deformation in the “Falling” sequence. From the top three rows, it is clear that both of our trackers, using the Epanechnikov kernel and diffusion kernel, are able to track the person in both, spatial and scale domain. More importantly, both trackers illustrate more accurate estimation of the target’s orientation and scale. On the other hand, the other three competing trackers were unable to follow the target as it enters the area in front of the ATM machine. Being unable to recover from the prior mislocalisation, the EM-like tracker (Zivkovic and Krose, 2004) and the original mean-shift tracker (Comaniciu et al., 2003) grow their target sizes rapidly until the end of the sequence. The particle-filter tracker (Pérez et al., 2002), although does not exhibit a large scale estimation as the other two, yet fails to localise the target as well.

As we turn our attention to the numerical results plotted in Figure 3.10, the visual evaluation above becomes more evident. In fact, the EM-like mean-shift (Zivkovic and Krose, 2004), the particle filter (Pérez et al., 2002) and the original mean-shift (Comaniciu et al., 2003) lose the target at approximately 100 frames into the sequence, by more than 100 pixels error in average, which is well above one forth of the video window horizontal size. This continuous mislocalisation results in a large target centre errors for the three trackers, or in particular, 102.32 ± 47.24 , 103.87 ± 37.71 and 122.05 ± 90.03 pixels for the particle filter, the EM-like mean-shift and the original mean-shift, respectively. The target area errors, as a results, are also recorded significantly high, i.e. 1771.19 ± 1031.07 , 53361.11 ± 14122.49 and 12374.06 ± 10806.13 for the the particle filter, the EM-like mean-shift and the original mean-shift, respectively.

In this sequence, both of our trackers, thanks to the capability of handling target transformations in scale and orientation effectively, stay on the target with an impressive target centre errors of 8.18 ± 2.68 and 7.49 ± 2.80 pixels, and the target area errors of 295.71 ± 197.10 and 278.17 ± 226.79 pixel square, for the Epanechnikov and the Diffusion kernel profiles, respectively. We have also tabulated the above results in

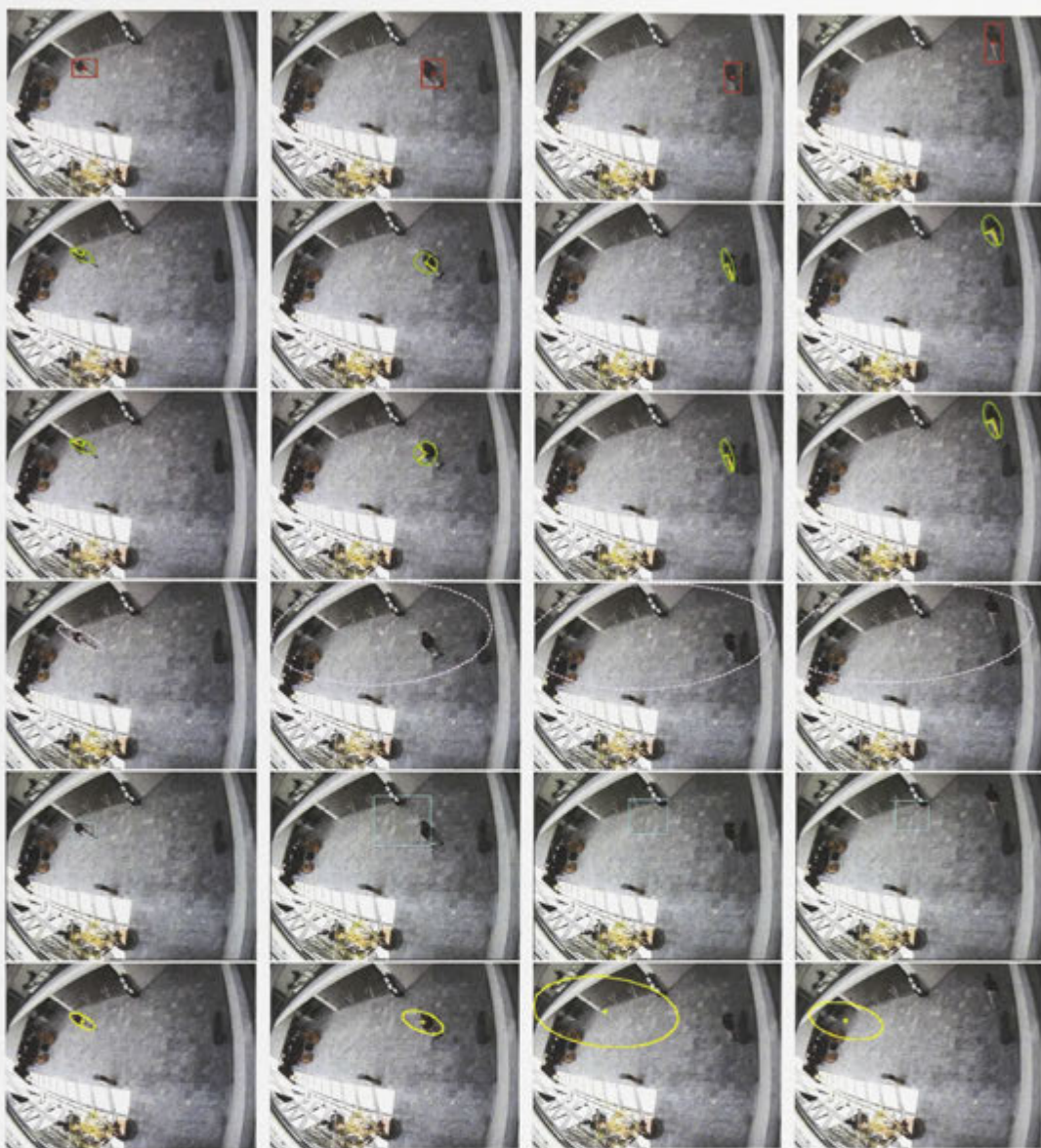


Figure 3.9: Results for the “Browse” sequence at frames 400, 500, 700 and 800. From top to bottom: Ground truth, results yield using the Epanechnikov and the Diffusion Kernels, EM-like Shift (Zivkovic and Krose, 2004), Particle filter (Pérez et al., 2002) and Mean-shift (Comaniciu et al., 2003).

Table 3.2.

Throughout the experiments, we can conclude that our approach performs much better than the EM-like mean-shift (Zivkovic and Krose, 2004), the particle-filter based tracker (Pérez et al., 2002) and the original mean-shift tracker (Comaniciu et al., 2003) with the tracking object that exhibits moderate to significant scale and orientation de-

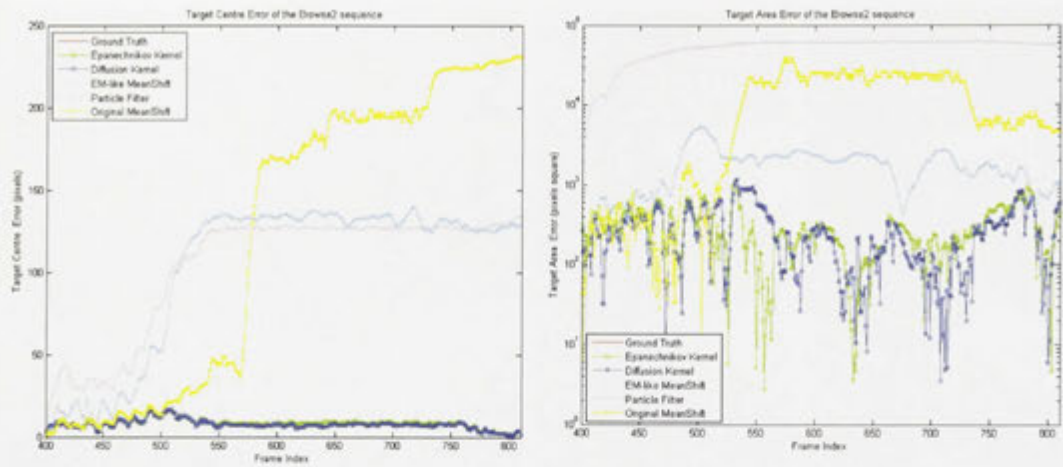


Figure 3.10: Left: Target centre square error, Right: Target area square error of the “Browse” Sequence.

Falling sequence				
	Mean Centre Error	STD Centre Error	Mean Area Error Square	STD Area Error Square
GroundTruth	0	0	0	0
Epanechnikov kernel	7.50	3.41	647.53	382.33
Diffusion kernel	7.41	3.42	684.87	384.16
EM-like mean-shift	7.51	3.23	821.74	910.47
Particle Filter	13.98	5.98	729.44	507.50
Original mean-shift	9.47	4.25	839.11	501.32

Table 3.1: Tracking performance evaluation for the “Falling” sequence.

formation, which is highly likely in a practical setup. It is also worth to stress that we have showed and consolidated the generalisation of the algorithm by achieving similar results with different monotomic kernel profiles, i.e. the Epanechnikov and the diffusion kernels.

3.5 Conclusions

In this chapter, we have presented a new probabilistic interpretation for kernel-based object tracking. By viewing the coordinates for the pixels as random variables, the tracking task can be cast into a maximum likelihood framework. This treatment natu-

Browse sequence				
	Mean Centre Error	STD Centre Error	Mean Area Error Square	STD Area Error Square
GroundTruth	0	0	0	0
Epanechnikov kernel	8.18	2.68	295.71	197.10
Diffusion kernel	7.49	2.80	278.17	226.79
EM-like mean-shift	103.87	37.71	53361.11	14122.49
Particle Filter	102.32	47.24	1771.19	1031.07
Original mean-shift	122.05	90.03	12374.06	10806.13

Table 3.2: Tracking performance evaluation for the “Browse” sequence.

rally lends itself to the recovery of the scale and orientation of the target in the scene. In particular, we have used the Kullback-Leibler divergence to measure the distance between the probability density functions of the target model and the candidate regions. This treatment, in turn, introduces negative weights into the mean-shift operation and hence, allows the mean-shift procedure to be performed in feature spaces where negative weights can be practically useful, such as in the scale and orientation spaces. To this end, we employ the polar coordinate system to estimate the scale and orientation of the target. We also demonstrate the generality of our method by using two different kernel profiles, i.e. the diffusion and the Epanechnikov kernels, to robustly track the target in several challenging real-world sequences. Throughout the experiments, our tracker has shown significant lead in tracking performance compared to a number of alternatives including the original method (Comaniciu et al., 2003), the EM-like mean-shift (Zivkovic and Krose, 2004) and the particle filter (Pérez et al., 2002).

Finally, it is worth noting that the methods presented in this chapter only employ generic image features such as RGB and HSV colour spaces and hence, their performances are susceptible to noise in colour appearance due to variation in the scene illumination, or when the target approaches others that share similar outfit. To this end, one would aim to exploit other image features such as texture and contrast to improve robustness. To do this, in the next chapter we will explore the possibility of employing other image features beyond the colour spaces as well as a combination of image features for our kernel-based tracking framework.

Chapter 4

Feature Combination in Object Tracking

In this chapter, we continue the work in Chapter 3 by exploring a new approach to combine multiple image features in the mean-shift tracker based upon graph embedding techniques. The method presented here abstracts the low complexity image features used for purposes of tracking to a relational structure and employs graph-spectral methods to combine them. This gives rise to a feature combination scheme which minimises the mutual cross-correlation between features and is devoid of free parameters. It also allows an analytical solution making use of matrix factorisation techniques. The new target location is recovered making use of a weighted combination of target-centre shifts corresponding to each of the features under study, where the feature weights arise from a cost function governed by the embedding process. This treatment permits the update of the feature weights in an on-line fashion in a straightforward manner. We illustrate the performance of our method in real-world image sequences and compare our results to a number of alternatives.

4.1 Kernel-based Tracking in Arbitrary Feature Spaces

As mentioned earlier, the kernel-based tracker (Comaniciu et al., 2003) makes use of the spatially-weighted histogram of the target region as input to a similarity function which the tracker aims at maximising via the mean-shift optimisation scheme (Cheng, 1995). In order to characterise a target, one or more feature spaces must be determined so that a non-parametric probability density function (PDF) such as M -bin histogram can be estimated. It is desirable to select a feature space that is distinctive to the target with respect to the surrounding background while being robust to noise and image

corruption.

Recall that, the mean-shift tracker presented in the previous chapter, along with the original method and the EM-like mean-shift tracker employ the RGB colour intensity as the feature space for the tracking task and, as a result, the M -bin histogram is a three-dimensional cube in which every dimension corresponds to a colour channel, i.e. Red, Green and Blue. The RGB feature space, however, is prone to image noise due to the fact that the three channels R,G and B are highly correlated. This shortcoming can potentially cause target mislocalisation when there are other objects with similar appearance present in the scene, or when the scene illumination varies during the tracking course. As a result, modern trackers often seek for additional image features to improve the tracking robustness. To this end, one notable advantage of the kernel-based tracker is that the method is not restricted to any particular feature space, and on the contrary, multiple image features can be used singly or simultaneously to enhance robustness. Here, we commence by introducing a particular case, in which texture image features outperform the colour ones. A literature review on the use of various image features in object tracking has been provided in Section 2.3.

4.1.1 Texture versus Colour

To commence, we consider deploying the mean-shift tracker on monochromatic and thermographic video inputs. While these inputs are not popular in the consumer market, they can be found in various security devices such as infra-red cameras, night-vision goggles and hyper-spectral cameras. In these cases, colour-based tracking is not applicable due to the nature of the image formation process for monochromatic and thermographic imagery, where, instead of colour samples, yields brightness and temperature measurements, respectively. This two intensity values on the image plane are one of the input features we use for our kernel-based tracking approach.

The other feature at input to our tracker is the Local Binary Pattern (LBP) (Ojala et al., 1996; Rodriguez and Marcel, 2006) for each of the pixels in both, the model and the target candidates. The LBP is a texture primitive based upon the intensity values, which has been employed for purposes of background modelling and is robust to variations in contrast and changes in illumination (Heikkila and Pietikainen, 2006). In our technique, the brightness value accounts for the shading information extracted from the target. The LBP, on the other hand, can be viewed as the texture descriptor for the object being tracked. For a detailed review on this image descriptor, we refer the reader to Section 2.3.4.

As mentioned above, from now on, we view either, temperature measurements or

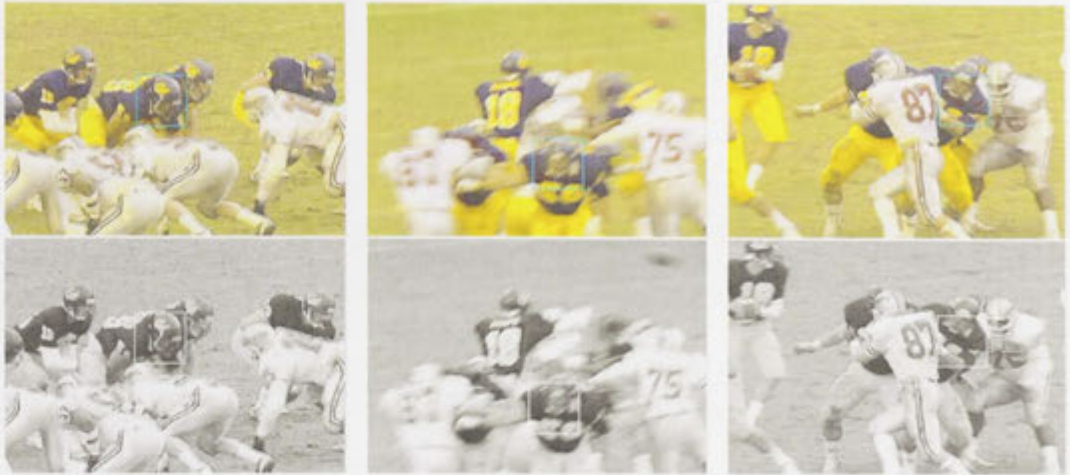


Figure 4.1: Results for the “Football Players” sequence. Top row: results yielded by the original mean-shift tracker (Comaniciu et al., 2003); Bottom row: results yielded by our method.

brightness values as a scalar on the image plane akin to gray-scaled pixel intensity. As a result of this treatment, the LBP can then be considered to capture the contrast of the image as an encoded string. We use these two features, i.e. pixel intensity and contrast, as those that are least affected by both, illumination changes on monochromatic videos and colour map transformations in thermographic videos. Following (Comaniciu et al., 2003), we use the intensity and LBP values as indices to allocate each of the pixels in the target image region to a bin in a 16×16 histogram. In this manner, our choice of feature space implies a mean-shift iteration on a two-dimensional feature vector set. This compares favourably with those kernel-based tracking approaches based upon colour spaces, where the feature vectors are three-dimensional in nature.

4.1.1.1 Normal RGB Inputs

To show the robustness of the LBP in coping with blurred, fast moving targets in monochromatic videos, we have used the “Football Players” sequence as shown in Figure 4.1. This video sequence consists of 83 frames depicting a close-up of an American Football down. The target is given by the helmet of the player number 59. The reasons for using this sequence are twofold. Firstly, it is a challenging one that has been used elsewhere in the literature. Secondly, this is a colour-native sequence that permits direct comparison with the approach of Comaniciu et al. (2003). In the bottom row of Figure 4.1, we show example results for our algorithm. The results for the kernel-based approach in (Comaniciu et al., 2003) are provided in the top row of

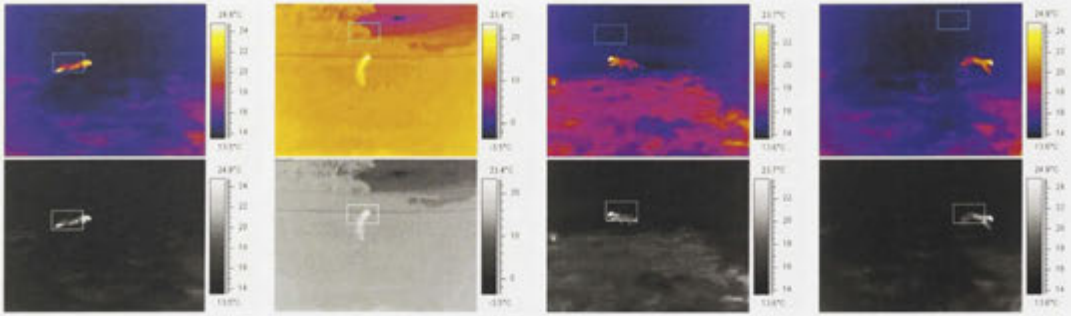


Figure 4.2: Results for the “Cat” sequence. Top row: results yielded by the original mean-shift tracker (Comaniciu et al., 2003); Bottom row: results yielded by our method.

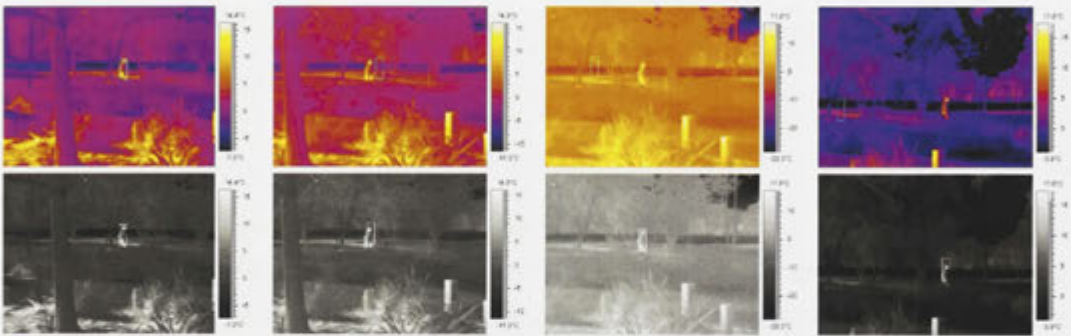


Figure 4.3: Results for the “Person in the Woods” sequence. Top row: results yielded by the original mean-shift tracker (Comaniciu et al., 2003); Bottom row: results yielded by our method.

Figure 4.1. To provide a fair comparison, the input to our method has been gray-scaled as a preprocessing step, whereas the results for the alternative have been computed using the full-colour information. From the figure, we can conclude that, despite being deprived of the colour information, our algorithm does not lose the target. Furthermore, the performance is comparable to that provided by the original mean-shift (Comaniciu et al., 2003).

4.1.1.2 Monochromatic and Thermographic Inputs

With similar performance achieved in full-colour sequences, we now turn our attention to the case of monochromatic and thermographic inputs. To do this, we make use of two real-world image sequences. The first of these is that of a cat jumping on and off a fence which we have called the “Cat” image sequence. The second of our thermal footages is the “Person in the Woods” sequence, which depicts a passer-by in

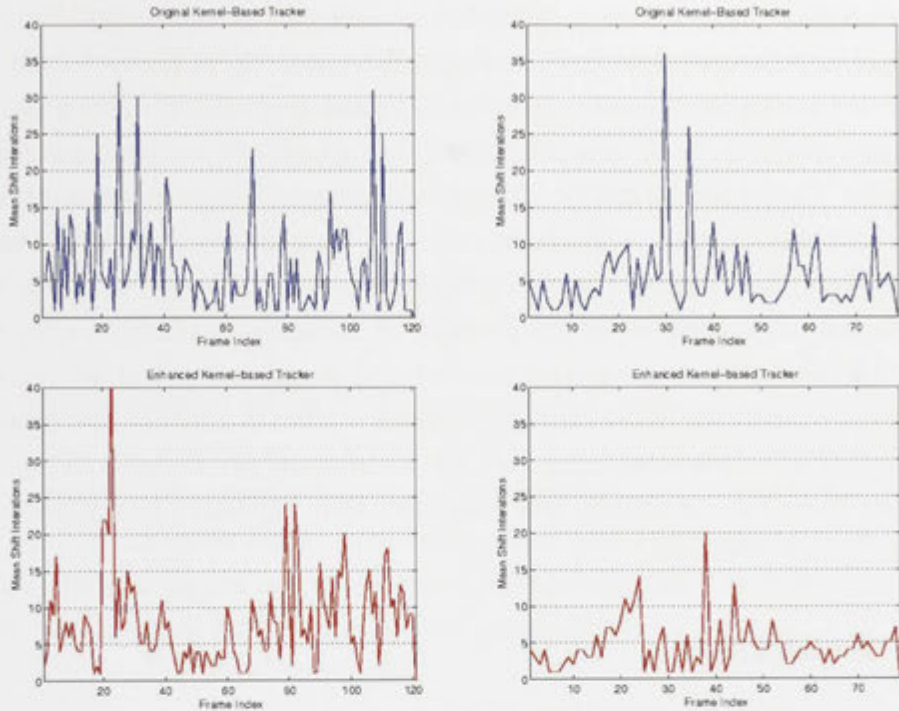


Figure 4.4: Number of shifts per frame for the “Cat” and “Person in the Woods” sequences. Top row: results yield by the original mean-shift tracker (Comaniciu et al., 2003); Bottom row: results yielded by our method.

a wooded area. In contrast with the “Football Players” sequence, in these cases, the colour information is not available at capture. Hence, to provide comparison with the alternative, we have resorted to the pseudo-colour image frames. In our experiments, the pseudo-colour palette is given by the sequence of the black body radiator colours in the thermal scale. Thus, our colourmap is such that warm objects appear bright orange, while cold surfaces in the scene are black.

In Figure 4.2 we provide example results for the “Cat” sequence. Figure 4.3 shows the results for the “Person in the Woods” footage. From the figures, it is clear that our method is capable of tracking the subjects in situations where the original method fails to follow the target. Furthermore, as a result of the use of the LBP, our algorithm can cope with changes in background texture and target temperature. It is worth noting that normalisation can always be applied as a preprocessing step. However, the use of LBP also accounts for thermal phenomena, such as convection, which may change relative brightness between objects in the scene, but preserves inter-object contrast.

Now we turn our attention to a more quantitative analysis on the performance of both, our method and the original mean-shift tracker, by focusing in the number of

shifts per frame. We do this so as to provide a means to the assessment of the computational cost for the two algorithms. In Figure 4.4, we show the plots of shifts versus frame index corresponding to the “Cat” and “Person in the Woods” sequences. Our tracker also averages lower mean number of shifts per frame, measuring at 8.01 and 4.50 for the “Cat” sequence and the “Person in the Woods” sequence, respectively. The original mean-shift tracker, while being unable to track the object, results in an average of 6.89 and 5.46 shifts per frame for the two sequences. From the figures, we can conclude that our algorithm requires a minimum margin of increase, as compared to the original mean-shift, in the average number of shifts per frame. It is worth stressing that, for our method, the optimisation process takes place in a two-dimensional space rather than a three dimensional one, i.e. LBP and intensity in contrast with RGB. This makes the shifts for our method less computationally expensive and, therefore, our algorithm is more efficient.

4.1.2 Remarks

The above examples have shown that, in practice, there are many cases in which texture and contrast image features can substantially replace the colour cues to perform the tracking task. In fact, the number of trackers relying solely on colour cues has become obsolete in recent years. On the contrary, multiple image features have been widely used in tracking community to enhance robustness. To this end, we have provided a review on the literature of multiple features trackers in Section 2.3.5 and subsequently in this chapter, we explore a novel approach in combining multiple image features for the mean-shift tracking algorithm making use of graph-embedding techniques.

4.2 Multiple Features Formulation

To extend the kernel-based tracking algorithm to a multi-feature framework, we require some formalism. Let $\Phi = \{\phi_1, \phi_2, \dots, \phi_{|\Phi|}\}$ be the set of image feature spaces used for purposes of tracking. This feature set can consists of a range of image features from the colour-based ones such as RGB, HSV, grayscale and YCbCr to textures such as Local Binary Pattern (LBP), Edges, Scale Invariant Feature Transform (SIFT) and Harr-like. For the feature space ϕ_i , the new target position η_{ϕ_i} can be recovered making use of the two M -bin histograms $\mathbf{Q}_{\phi_i}$ and $\mathbf{P}_{\phi_i}$ corresponding to the target model and the search window, respectively. That is

$$\eta_{\phi_i} = \frac{\sum_{n=1}^N \mathbf{x}_n \mathcal{G}(\mathbf{x}_n) w_n}{\sum_{i=1}^N \mathcal{G}(\mathbf{x}_n) w_n} \quad (4.1)$$

where w_n is the similarity weight for the n^{th} pixel $\mathbf{x}_n$ in the search window and $\mathcal{G}(\cdot)$ is the shadow kernel of the kernel profile $\mathcal{K}$. For further details on the equation above, we direct the reader to the preliminaries of the mean-shift in Section 2.4.2.1.

With the $|\Phi|$ “weak” shifts $\{\eta_{\phi_i}\}_{\phi_i \in \Phi}$ at hand, we can compute the “global” shift η as the weighted average of these “weak” shifts as

$$\eta = \sum_{i=1}^{|\Phi|} \gamma_{\phi_i} \eta_{\phi_i} \quad (4.2)$$

where γ_{ϕ_i} is the feature weight for the updated target-centre η_{ϕ_i} corresponding to the feature space ϕ_i . A good set of feature weights $\{\gamma_{\phi_i}\}_{i \in \Phi}$, as a result, allocates more weight towards discriminative features and suppresses weight of the less discriminative ones, with respect to the surrounding background. This treatment, therefore, not only improves the tracking performance, but also provides a means to evaluate the suitability of each image feature for different types of video inputs.

4.3 Feature Combination via Graph Embedding

In this section, we provide a principled link between the graph embedding techniques and the recovery of the feature weights γ_{ϕ_i} . This section commences with a brief introduction to the idea behind the graph-based approach and its analytical solutions from the eigenpairs (the eigenvalue and the eigenvector) of the Laplacian matrix. We then cast the problem of finding the optimal feature weights γ_{ϕ_i} into a graph-theoretic setting, whereas the solution for the embedding vector can be recovered from the eigenvectors of a purposely constructed double-centered matrix.

4.3.1 Graph Embedding and Computer Vision

In computer vision, the graph-based approach is often used as an effective tool for image clustering (Sarkar and Boyer, 1998; Perona and Freeman, 1998; Shi and Malik, 2000), in which a pixel or a group of pixels are modelled as vertices on a graph whose edges defined by the (dis)similarity amongst the neighbourhood pixels. This graph-based clustering process is often solved by minimising or maximising the (dis)similarity between every pair of nodes of the graph, i.e. pairwise clustering.

One of the most elegant solutions for the pairwise clustering problem comes from spectral graph theory, i.e. the study of properties of a graph in relationship to the eigenvalues and the eigenvectors of the Laplacian matrix and the adjacency matrix. In

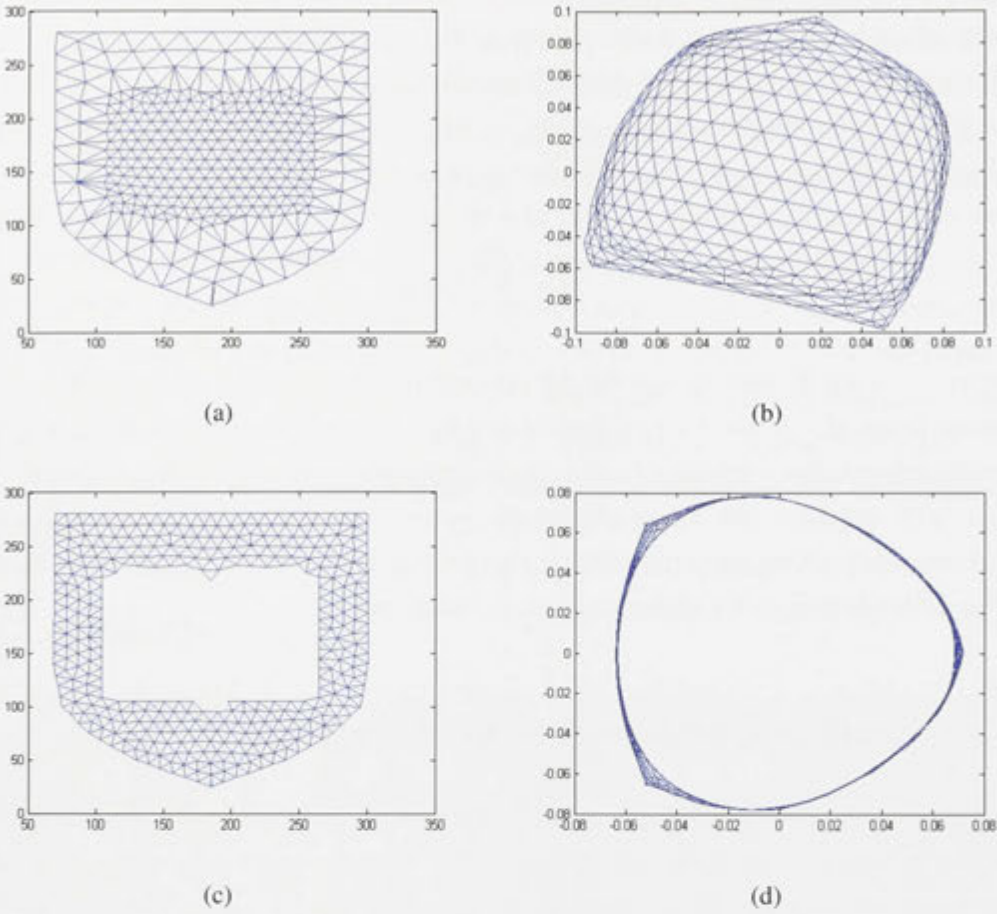


Figure 4.5: Graph embedding technique: (a) A planar graph of the emblem of the Yale University and (c) with the middle part removed; (b) and (d) are the corresponding embedded graphs on a 2D coordinates using the eigenvectors of the two smallest non-zero eigenvalues of the Laplacian matrix of the graph.

particular, the eigenvector of the Laplacian matrix of a graph can be viewed as a function on the vertices, that is, it assigns a real number to each vertex. This treatment, in turn, allows the use of eigenvectors to embed a graph G , for instance, onto a metric plane Σ . For example, Figure 4.5(a) shows the emblem of the Yale University¹ as a connected planar graph. This planar graph is embedded to an Euclidean 2D space as shown in Figure 4.5(b). In particular, the embedding process makes use the eigenvectors corresponding to the two smallest non-zero eigenvalues of the Laplacian matrix of the graph as the x and y coordinates for each vertex. This embedded graph, however,

¹The MATLAB code for the planar graphs in Figure 4.5 can be accessed from <http://www-math.mit.edu/spielman/eigs/>

is significantly different with the middle part of the original graph removed, as shown in Figure 4.5(c) and (d). This simple illustration has shown the close link between a planar graph and its embedded images via the Laplacian matrix. The spectral embedding process, as a result, not only provides a good understanding of the structure of a planar graph, but also allows a means to embed the graph to another plane, in which the vertices can be conveniently clustered or grouped accordingly. When the projected plane is at a lower dimension, the process is often referred as multi dimensional scaling (MDS).

In this light, some of the earliest work was done by Scott and Longuet-Higgins (1990), in which a method for refining the block-structure of the affinity matrix is obtained by relocating its eigenvectors. Sarkar and Boyer (1998) presented a spectral method which locates clusters that maximise the average association to detect changes in image features of aerial images. In particular, four measures for image organisational change were proposed for site monitoring. The measures are based on the change in the relationship among image features. The relationships are captured by the eigenpairs of the relation graph embodying the organisation among the image features. In (Perona and Freeman, 1998), a similar approach is proposed but employs the second largest eigenvector of the affinity matrix. An image segmentation method proposed by Shi and Malik (2000), on the other hand, is based upon the normalised cut, an unbiased measure of disassociation between subgroups of a graph with a nice property that minimising normalised cut leads directly to maximising the normalised association. More recently, Robles-Kelly (2005) shows how the incidence mapping and the graph Laplacian can be used to pose the problem of embedding as that of recovering a Gram matrix of scalar products, and subsequently, uses this embedding vector to group a set of shape silhouettes of different objects.

These aforementioned graph-based methods, however, have not been used in object tracking, where the problem of clustering different objects on an image can be posed as that of clustering image features that are most discriminant for the object with respect to its background, as well as to other tracking features. Here, we explore the possibility of using graph spectral methods to combine multiple features in a weighted manner for the tracking task.

4.3.2 Feature Mapping

We now turn our attention to the recovery of the feature weight γ_{ϕ_i} . To this end, we cast the problem of feature combination into a graph-theoretic setting. In particular, we aim at embedding the set of pairwise correlations between features in a metric space.

To do this, we abstract the pairwise relationships between low complexity features into a relational structure and make use of graph-spectral methods, i.e. the eigenvalues and eigenvectors of the Laplacian matrix (Chavel, 1995), so as to cast the feature weight γ_{ϕ_i} in an optimisation setting that leads to a Rayleigh Quotient. This can be viewed as the recovering of a graph embedding such that the correlation between features is minimum.

This embedding process commences by viewing the probability density functions (PDFs) for the target foreground and its surrounding background as nodes on a weighted graph, whose edge-weights are given by their correlation in its geometric sense, i.e. the inner product of the pairwise PDFs. Viewed in this way, the Laplacian of the graph can be related to a Gram matrix of scalar products. This treatment, in turn, allows the use of matrix factorisation techniques to recover the coordinates for the embedding of the graph. Thus, the problem of finding the feature weight γ_{ϕ_i} turns into that of recovering the set of variables that maximises the pairwise distances between the features under consideration and, therefore, minimises their cross-correlation via the use of the eigenvalues and eigenvectors of a purposely-constructed matrix.

To commence, we require some formalism. Let $G = (V, E, W)$ denote a weighted graph with index-set V , edge-set $E = \{(u, v) | (u, v) \in V \times V\}$ and edge-weights $W : E \mapsto [0, 1]$. Recall that, as mentioned earlier, the nodes of the graph are the PDFs for the target model and the scene background, i.e. $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ and $\{P_{\phi_i}\}_{\phi_i \in \Phi}$ respectively. As a result, we let the weight $W(u, v)$ associated with the edge connecting the pair of nodes u and v corresponding to the i^{th} and j^{th} features in Φ be given by the normalised cross-correlation

$$W(u, v) = \begin{cases} \left\langle \frac{Q_{\phi_i}}{\|Q_{\phi_i}\|}, \frac{P_{\phi_i}}{\|P_{\phi_i}\|} \right\rangle & \text{if } i = j \\ \left\langle \frac{Q_{\phi_i}}{\|Q_{\phi_i}\|}, \frac{Q_{\phi_j}}{\|Q_{\phi_j}\|} \right\rangle & \text{otherwise} \end{cases} \quad (4.3)$$

Note that W is a symmetric matrix of scalar products, in which the diagonal elements are given by the cross-correlation between the PDFs for the foreground and the background of the same feature, i.e. Q_{ϕ_i} and P_{ϕ_i} , while the off-diagonal elements are the cross-correlation between the PDFs for the foreground for different features, i.e. Q_{ϕ_i} and Q_{ϕ_j} .

To take our analysis further, we proceed to define the squared distance between features on the graph. Here, we set the pairwise squared distance between a pair of nodes as their correlation value. This is akin to the approaches in pairwise grouping such that in (Robles-Kelly, 2005). We define

$$W(u, v) = \|\varphi(u) - \varphi(v)\|^2 \quad (4.4)$$

where $\varphi(u)$ is the embedding vector, i.e. the vector of coordinates for the feature ϕ_i corresponding to the node u in V . Since W is a symmetric matrix of scalar products, the squared distance can also be expressed in terms of a set of inner products as follows

$$W(u, v) = \langle \varphi(u), \varphi(u) \rangle + \langle \varphi(v), \varphi(v) \rangle - 2 \langle \varphi(u), \varphi(v) \rangle \quad (4.5)$$

This permits viewing the correlation between tracking features as pairwise distances in a metric space making use of the inner products.

4.3.3 Double Centering

To provide a link between the edge-weights $W(u, v)$ and the coordinate vectors $\varphi(u)$, we make use of double-centering (Borg and Groenen, 1997). In particular, this can be achieved by firstly relating the edge-weight matrix W to the Laplacian matrix $\mathcal{L}$ (Chung, 1997). With the Laplacian matrix at hand, a double-centered matrix of scalar products $\mathbf{H}$ can be computed. This operation introduces a linear dependency over the columns of the matrix $\mathbf{H}$ while preserving the symmetry of W .

This treatment is important because it allows us to also view the double centered matrix $\mathbf{H}$ as a matrix of scalar products, i.e. $\mathbf{H} = \mathbf{J}\mathbf{J}^T$, which can then be interpreted as the sums of squared, pairwise distances $\|\varphi(u) - \varphi(v)\|^2$ as in Equation 4.4. Furthermore, it can be shown that the matrix $\mathbf{H}$ is, in fact, the double-centered graph Laplacian (Borg and Groenen, 1997). As a result, the element of the matrix $\mathbf{H}$ corresponding to the nodes $u, v \in V$ is given by

$$\mathbf{H}(u, v) = -\frac{1}{2} \left[\mathcal{L}(u, v)^2 - \frac{1}{|V|} \sum_{u \in V} \mathcal{L}(u, v)^2 - \frac{1}{|V|} \sum_{v \in V} \mathcal{L}(u, v)^2 + \frac{1}{|V|^2} \sum_{u, v \in V} \mathcal{L}(u, v)^2 \right] \quad (4.6)$$

That is, the scalar products of the centered coordinates are $-1/2$ times the squared distances with the row and column means removed and the grand mean added back in. The graph Laplacian $\mathcal{L}$ is defined as

$$\mathcal{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2} \quad (4.7)$$

where $\mathbf{D}$ is a diagonal matrix such that $\mathbf{D} = \text{diag}(\text{deg}(1), \text{deg}(2), \dots, \text{deg}(|V|))$ and $\text{deg}(u) = \sum_{v \in V} W(u, v)$ is the degree of the node $u \in V$.

Let ξ_l be the l^{th} eigenvector of $\mathbf{H}$ scaled so its sum of squares is equal to the corresponding eigenvalue λ_l . Since $\mathbf{H}\xi_l = \lambda_l\xi_l$ and $(\mathbf{J}\mathbf{J}^T)\xi_l = \mathbf{H}\xi_l$, it follows that the

squared distance between a pair of nodes in Equation 4.4 can be now written as

$$\begin{aligned} \|\varphi(u) - \varphi(v)\|^2 &= \sum_{l=1}^{|V|} \lambda_l (\xi_l(u) - \xi_l(v))^2 \\ &= \mathbf{H}(u, u) + \mathbf{H}(v, v) - 2\mathbf{H}(u, v) \end{aligned} \quad (4.8)$$

4.3.4 Minimising Feature Correlation

With these ingredients, we can introduce the variables $\pi(u)$ such that the weighted correlations between low complexity features are minimum. We do this by making use of the quantity

$$\epsilon = \sum_{u,v \in V} \|\pi(u)\varphi(u) - \pi(v)\varphi(v)\|^2 \quad (4.9)$$

which we aim at minimising. The cost function above can also be interpreted as the sum of squared weighted cross-correlations between the PDFs used for purposes of tracking. Thus, we can use Equation 4.8 and, after some algebra, we write

$$\begin{aligned} \epsilon &= \sum_{u,v \in V} (\pi(u)^2 \mathbf{H}(u, u) + \pi(v)^2 \mathbf{H}(v, v) - 2\pi(u)\pi(v)\mathbf{H}(u, v)) \\ &= \sum_{u,v \in V} \pi(u)^2 \mathbf{H}(u, u) + \sum_{u,v \in V} \pi(v)^2 \mathbf{H}(v, v) - 2 \sum_{u,v \in V} \pi(u)\pi(v)\mathbf{H}(u, v) \\ &= 2|V| \sum_{u \in V} \pi(u)^2 \mathbf{H}(u, u) - 2 \sum_{u,v \in V} \pi(u)\pi(v)\mathbf{H}(u, v) \end{aligned} \quad (4.10)$$

where we use the following facts

$$\sum_{u,v \in V} \pi(u)^2 \mathbf{H}(u, u) = |V| \sum_{u \in V} \pi(u)^2 \mathbf{H}(u, u) \text{ and } \sum_{u,v \in V} \pi(v)^2 \mathbf{H}(v, v) = \sum_{u,v \in V} \pi(v)^2 \mathbf{H}(v, v)$$

Note that, the second term on the right hand side of Equation 4.10 can be divided into two sets of terms. The first of these corresponds to the diagonal matrix of $\mathbf{H}$. The other set accounts for the off-diagonal elements of $\mathbf{H}$. Rearranging terms, we get

$$\begin{aligned} \epsilon &= 2|V| \sum_{u \in V} \pi(u)^2 \mathbf{H}(u, u) - \sum_{\substack{u,v \in V \\ u=v}} 2\pi(u)^2 \mathbf{H}(u, u) - \sum_{\substack{u,v \in V \\ u \neq v}} 2\pi(u)\pi(v)\mathbf{H}(u, v) \\ &= 2|V| \sum_{u \in V} \pi(u)^2 \mathbf{H}(u, u) - 2|V| \sum_{u \in V} \pi(u)^2 \mathbf{H}(u, u) - \sum_{\substack{u,v \in V \\ u \neq v}} 2\pi(u)\pi(v)\mathbf{H}(u, v) \end{aligned} \quad (4.11)$$

Hence, Equation 4.11 can be reduced to

$$\epsilon = - \sum_{\substack{u,v \in V \\ u \neq v}} 2\pi(u)\pi(v)\mathbf{H}(u,v) \quad (4.12)$$

which can be written in compact form by defining a matrix $\hat{\mathbf{H}}$ which comprises the off-diagonal elements of $\mathbf{H}$ as follows

$$\hat{\mathbf{H}}(u,v) = \begin{cases} \mathbf{H}(u,v) & \text{if } u \neq v \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

This yields

$$\epsilon = -2\mathbf{\Pi}^T \hat{\mathbf{H}} \mathbf{\Pi} \quad (4.14)$$

where $\mathbf{\Pi} = [\pi(1), \pi(2), \dots, \pi(|V|)]^T$ is a column vector of order $|V|$. Note that the expression above is the numerator of a Rayleigh Quotient $\frac{\mathbf{\Pi}^T \hat{\mathbf{H}} \mathbf{\Pi}}{\mathbf{\Pi}^T \mathbf{\Pi}}$ where the omitted denominator, $\mathbf{\Pi}^T \mathbf{\Pi}$, is a normalisation constant. Thus, minimising ϵ is equivalent to maximising $\mathbf{\Pi}^T \hat{\mathbf{H}} \mathbf{\Pi}$ and, therefore, $\mathbf{\Pi}^* = \underset{\mathbf{\Pi}}{\operatorname{argmin}} \{ \epsilon \}$ is given by the leading eigenvector of $\hat{\mathbf{H}}$ which corresponds to the eigenvalue whose rank is the largest.

The vector $\mathbf{\Pi}^*$, hence, is the minimiser of the squared distances between the nodes in the graph, i.e. the correlation between features. As a result, the set of feature weights $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ corresponding to the “weak” shifts is given by

$$\gamma_{\phi_i} = \frac{\pi(u)}{\sum_{u \in V} \pi(u)} \quad (4.15)$$

where the i^{th} feature ϕ_i corresponds to the node u in V .

4.4 On-line Feature Weight Updating

As kernel-based trackers (Comaniciu et al., 2003; Collins et al., 2005; Han and Davis, 2004) rely on the M -bin histograms of the model to determine the target location via the mean-shift optimisation scheme, the validity of these histograms is extremely important for robust tracking. In (Collins et al., 2005), these M -bin histograms are modified after every frame by randomly selecting pixels from the target foreground so as to modify the tracking models across the feature spaces. Despite effective, this “mixing” method does not discriminate between pixels and, hence, is susceptible to mislocalisation due to histogram bias.

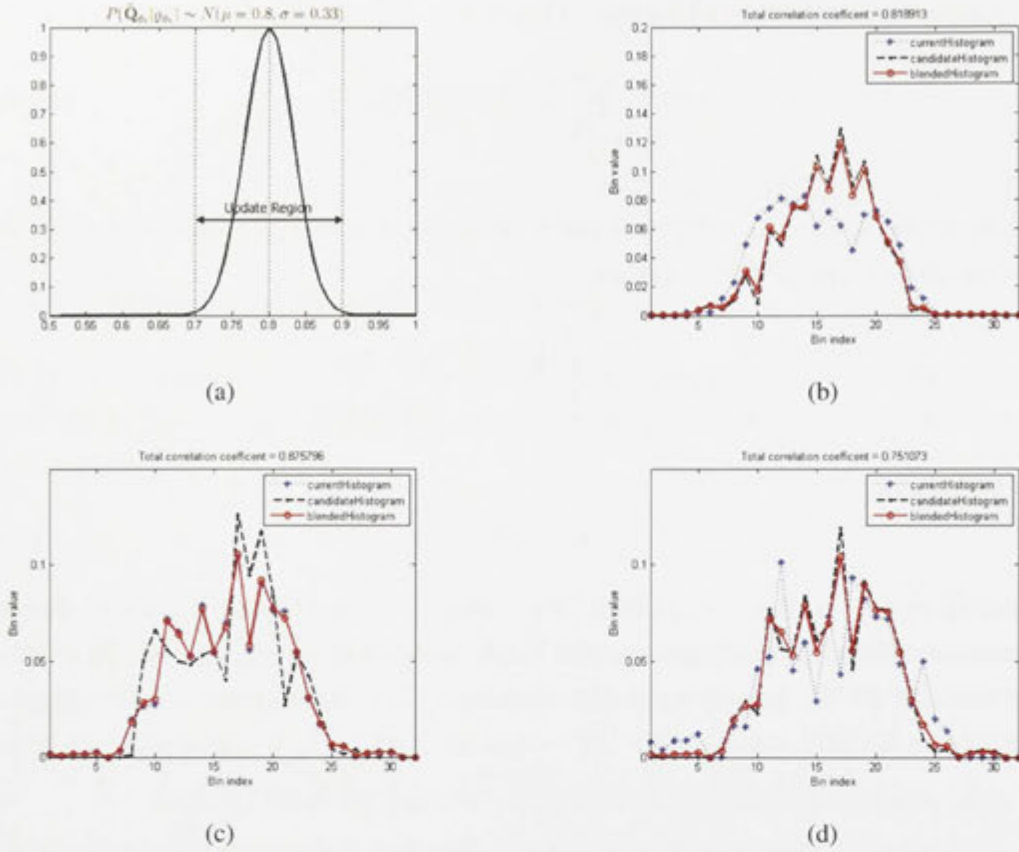


Figure 4.6: Histogram blending operation: (a) The PDF of $P(\hat{Q}_{\phi_i} | q_{\phi_i}) \sim N(0.8, 0.33)$; (b) The blended histogram is heavily weighted in favour of the candidate histogram, $q_{\phi_i} = 0.818$; (c) The blended histogram is kept the same with the current histogram, $q_{\phi_i} = 0.875$; and (d) The blended histogram is an even combination of the current histogram and the candidate histogram, $q_{\phi_i} = 0.751$.

Here, we present an on-line feature weight updating method based upon the cross-correlation between the histograms of the current target model and that corresponding to the recovered target-centre after each mean-shift application. This technique is based upon the weighted cross correlation between histograms and, thus, is devoid of pixel-sample selection and injection. Moreover, we calculate the total cross-correlation as a weighted average similarly to one presented in Section 4.3.

To commence, let $\{\hat{Q}_{\phi_i}\}_{\phi_i \in \Phi}$ be the set of the M -bin histograms obtained from the new target position and q_{ϕ_i} be the cross-correlation between Q_{ϕ_i} and the current target histogram $\hat{Q}_{\phi_i}$. That is

$$q_{\phi_i} = \left\langle \frac{Q_{\phi_i}}{\|Q_{\phi_i}\|}, \frac{\hat{Q}_{\phi_i}}{\|\hat{Q}_{\phi_i}\|} \right\rangle \quad (4.16)$$

The total cross-correlation between the two sets of histograms, $\{\hat{Q}_{\phi_i}\}_{\phi_i \in \Phi}$ for the new target position and $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ for the current model, can be computed as a linear combination of the weighted feature cross-correlation ϱ_{ϕ_i} as

$$\Gamma = \sum_{\phi_i \in \Phi} \gamma_{\phi_i} \varrho_{\phi_i} \quad (4.17)$$

where $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ is the set of feature weights derived from Section 4.3. This treatment, in turn, allows us to set decision bounds for the updating operation. We do this by updating the model M -bin histograms only when the condition $0 \leq \kappa_0 < \Gamma < \kappa_1 \leq 1$ is satisfied, where κ_0 and κ_1 are constants. This hinges in the confidence of the tracking operation by following the notion that, if the total correlation between the new target-centre histograms and that of the target model is close to unity, there is no need to update since the two sets are sufficiently “close”. On the contrary, if the total correlation is too low, then updating would “corrupt” the model. Updating is hence, appropriate when the correlation is not so low so as to introduce noise corruption but not as high as to be a computational burden without improving tracking accuracy.

When update operations are deemed necessary, the histogram set $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ for the target model is updated making use of a mixture model of the form

$$Q'_{\phi_i} = P(\hat{Q}_{\phi_i} | \varrho_{\phi_i}) \hat{Q}_{\phi_i} + \left(1 - P(\hat{Q}_{\phi_i} | \varrho_{\phi_i})\right) Q_{\phi_i} \quad (4.18)$$

This can be viewed as a “blending” operation between the two histograms. It is, indeed, a two-class expectation for the two PDFs $\hat{Q}_{\phi_i}$ and Q_{ϕ_i} , whose prior is given by the probability of the new target position given the feature cross-correlations ϱ_{ϕ_i} .

Figure 4.6 shows an example of the histogram blending algorithm in action. Here, the conditional probability for the update operations is set to be normally distributed with $\mu = 0.8$ and $\sigma = 0.33$, i.e. $P(\hat{Q}_{\phi_i} | \varrho_{\phi_i}) \sim N(0.8, 0.33)$, as illustrated in Figure 4.6(a). In Figure 4.6(b), the cross-correlation $\varrho_{\phi_i} = 0.818$. As a result, the updated histogram Q' is nearly identical to the candidate histogram $\hat{Q}$, due to the fact that the update operation gives negligible weight for the current histogram Q . This proportion, however, is reversed with a higher or lower ϱ_{ϕ_i} as shown in Figure 4.6(c) and (d), where the cross-correlation $\varrho_{\phi_i} = 0.875$ and $\varrho_{\phi_i} = 0.751$, respectively. Beyond this range, there is no update since the candidate histogram is either corrupted due to mislocalisation ($\varrho_{\phi_i} < 0.7$) or nearly identical to the current histogram ($\varrho_{\phi_i} > 0.9$).

4.5 Algorithm Description

With the developments presented in the previous sections, the tracking algorithm can be divided into two stages. The first stage is the training phase, in which the user is

Algorithm 3: Training**Data:** Selected region of the target model.**begin** **Sample** N_1 pixels from the foreground, N_2 pixels from the background. **Compute** the set of M -bin histograms $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ and $\{P_{\phi_i}\}_{\phi_i \in \Phi}$ across the feature spaces Φ . **Compute** W as in Equation 4.3 **Compute** H as in Equation 4.6 and $\hat{H}$ as in Equation 4.13 **Compute** Π^* as the leading eigenvector ξ_1 of $\hat{H}$ **Compute** the feature weights $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ using Π^* as in Equation 4.15 **Save** $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ and $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ **end**

required to select the target to track at the initial frame of the sequence. The samples inside the selected region are then used to compute a set of M -bin histograms corresponding to the feature spaces under study. In a similar manner, the area around the target is also sampled to create a set of background M -bin histograms. In our implementation, for the sake of efficiency, we perform background sampling in the area of twice the size of the target. With the two sets of foreground and background PDFs at hand, we compute the corresponding cross-correlation weight matrix W . Subsequently, the double-centering matrix H is determined, followed by its off-diagonal matrix $\hat{H}$. The set of feature weights $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ is then recovered from the leading eigenvector of $\hat{H}$.

In the second stage, the tracking vehicle is the mean-shift tracker presented in (Comaniciu et al., 2003). After each new target position, the total cross-correlation Γ is calculated to determine if the set of model M -bin histograms needs to be updated, i.e. $\kappa_0 < \Gamma < \kappa_1$. This implies that the feature weights and the target-model feature histograms will only be updated if the tracking operation is reliable, i.e. with a $\Gamma > \kappa_0$, while keeping computational cost low by avoiding updating operations when the candidate and the model are virtually the same, i.e. with a $\Gamma < \kappa_1$. The pseudo codes for the two stages are presented in Algorithm 3 and 4, respectively.

Algorithm 4: Tracking

Data: $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$, $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ and target centre y

begin

for $idx = \text{StartFrame}$ **to** EndFrame **do**

while *true* **do**

Compute the set of M -bin histograms $\{P_{\phi_i}\}_{\phi_i \in \Phi}$ for the searching window

Compute the target centre $\{\eta_{\phi_i}\}_{\phi_i \in \Phi}$ of each mean-shift as in Equation 4.1

Compute the new target centre $\eta = \sum_{\phi_i \in \Phi} \gamma_{\phi_i} \eta_{\phi_i}$

if $\|\eta - y\| \leq \varepsilon$ **then**

$idx = idx + 1$

break

else

Update the target centre $y = \eta$

end

end

Compute $\{\hat{Q}_{\phi_i}\}_{\phi_i \in \Phi}$ at the new target centre

Compute $\varrho_{\phi_i} = \left\langle \frac{Q_{\phi_i}}{\|Q_{\phi_i}\|}, \frac{\hat{Q}_{\phi_i}}{\|\hat{Q}_{\phi_i}\|} \right\rangle$

Compute $\Gamma = \sum_{\phi_i \in \Phi} \gamma_{\phi_i} \varrho_{\phi_i}$

if $\kappa_0 < \Gamma < \kappa_1$ **then**

Compute $P(\hat{Q}_{\phi_i} | \varrho_{\phi_i})$

Update $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ to $\{Q'_{\phi_i}\}_{\phi_i \in \Phi}$ using Equation 4.18

Compute the new feature weights $\{\gamma_{\phi_i}\}_{\phi_i \in \Phi}$ using the updated $\{Q_{\phi_i}\}_{\phi_i \in \Phi}$ as in Algorithm 1

end

end

end

4.6 Experiments

In this section, we illustrate the robustness of our algorithm by presenting results on two image sequences from the PETS-ECCV 2004 dataset². In the first sequence, the target moves from a bright area in the scene to a shady region, meets another person and then walks away. The second sequence shows a group of four people moving across the scene with some body-overlapping as they approach the camera. For each

²PETS dataset can be accessed from <http://www.cvg.rdg.ac.uk/slides/pets.html>

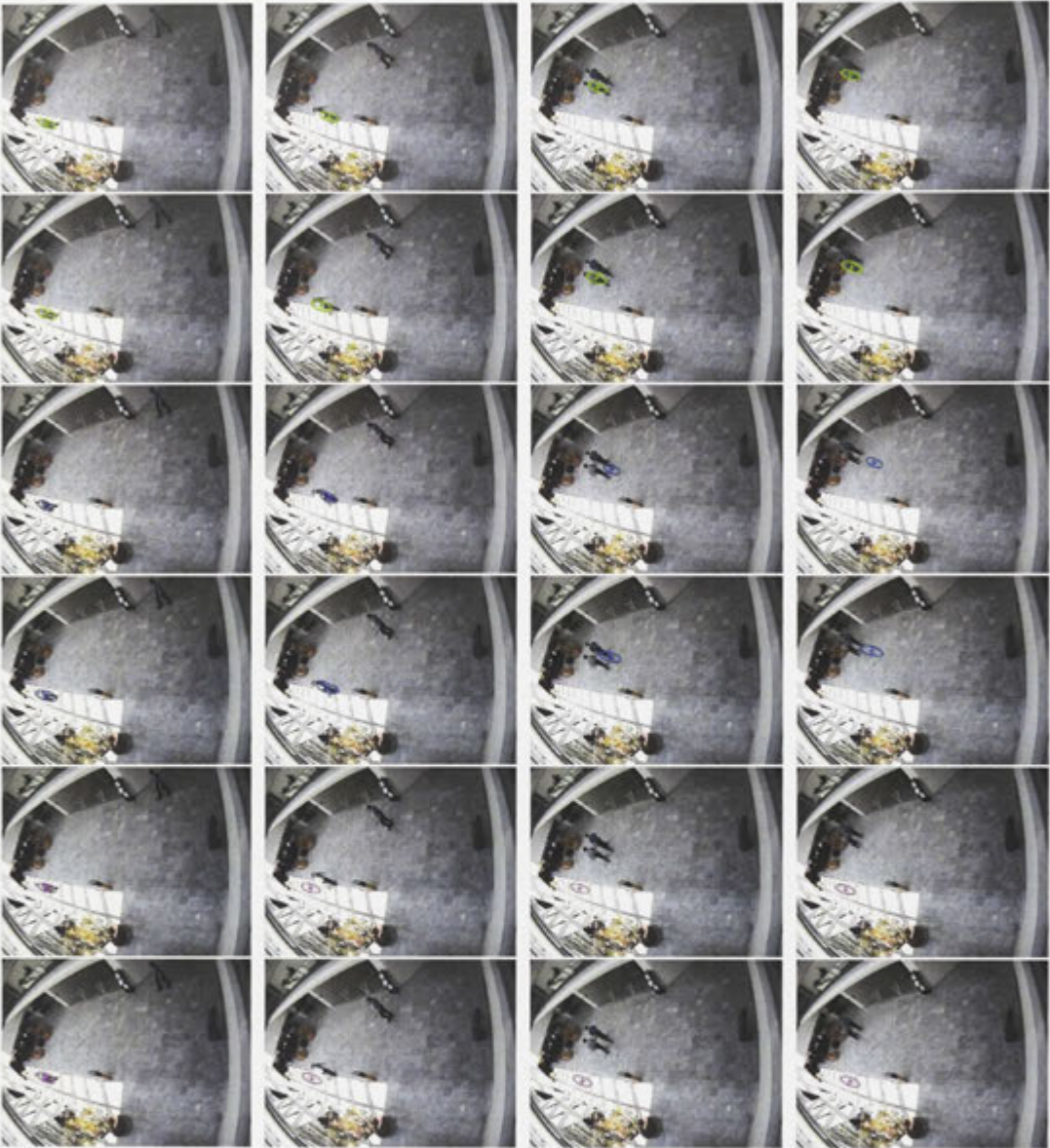


Figure 4.7: Results for the “Meet and Walk Together 1” sequence at frames 250, 290, 380 and 420. From top-to-bottom: results yielded by our algorithm using the 49-feature set and the 11-feature set (first and second rows), the on-line VR-based tracker (Collins et al., 2005) using the 49-feature set and the 11-feature set (third and forth rows), and the on-line PCA-based tracker (Han and Davis, 2004) using the 49-feature set and the 11-feature set (fifth and sixth rows).

of these sequences, the tracking target is manually selected by the user at the initial frame.

We have compared our results to those yielded by two competing algorithms. These

are the on-line Variance Ratio-based (VR-based) method proposed by Collins et al. (2005) and the on-line PCA-based method by Han and Davis (2004). These methods both make use of the mean-shift algorithm as the tracking vehicle and employ multiple image features to enhance robustness. To this end, the method proposed in (Collins et al., 2005) ranks image features based on the Variance Ratio between the foreground and background likelihood images, and hence, we call it the VR-based method. The tracker in (Han and Davis, 2004) combines image features via the Principle Component Analysis procedure, thus we call the on-line PCA-based method. Note that these methods have been reviewed in Section 2.3.5 and have significant improvement in performance over the random weights.

We have also implemented two sets of features. The first set consists of 49 linear combinations of R,G,B pixel values (Collins et al., 2005), namely the 49-feature set. The second set is a mix of gradient, contrast and texture features including brightness, normalised RGB, Local Binary Patterns (LBPs) and six Haar-like features (Viola and Jones, 2002), which we call the 11-feature set. The Harr-like features include vertical and horizontal 2,3 and 4-rectangle features. This combination of features encompasses both texture and colour information yet they can be computed rapidly due to the low complexity level.

In our implementation of the VR-based tracker, we select the set of five log-likelihood images that yield the highest Variance Ratio as the tracking features for the current frame. For the PCA-based tracker, the eigenvectors associated with the eigenvalues whose normalised sum is greater than 0.7 are used so as to reduce the dimensionality of the log-likelihood images. As suggested in (Han and Davis, 2004), a Gaussian filter is also implemented in order to reduce the amount of unwanted noise in the likelihood image corresponding to the leading eigenvalue.

For our tracker, we consider the conditional probability for the update operations to be normally distributed, i.e. $P(\hat{\mathbf{Q}}_{\phi_i} | \mathcal{Q}_{\phi_i}) \sim N(\mu, \sigma)$. Moreover, we set $\mu = (\kappa_0 + \kappa_1)/2$ and $\sigma = (\kappa_1 - \kappa_0)/4$, i.e. two standard deviations from the mean so as to account for 95% of the interval between κ_0 and κ_1 . This treatment allows the set of M -bin histograms for the target model to be updated based upon their individual correlations given the upper and lower bounds set for the update operation as a whole. We set the constants which govern the model update operations to $\kappa_0 = 0.7$ and $\kappa_1 = 0.9$.

The tracking vehicle used in all three algorithms under studies is the original mean-shift tracker (Comaniciu et al., 2003), in which the the Epanenikov kernel is used as the kernel profile $\mathcal{K}(\cdot)$. The size of the search window is defined as twice the size of the target at the initial frame and an 512-bin histogram is used to construct the PDF for each image feature space. The choice of 512-bin histogram is consistent with the

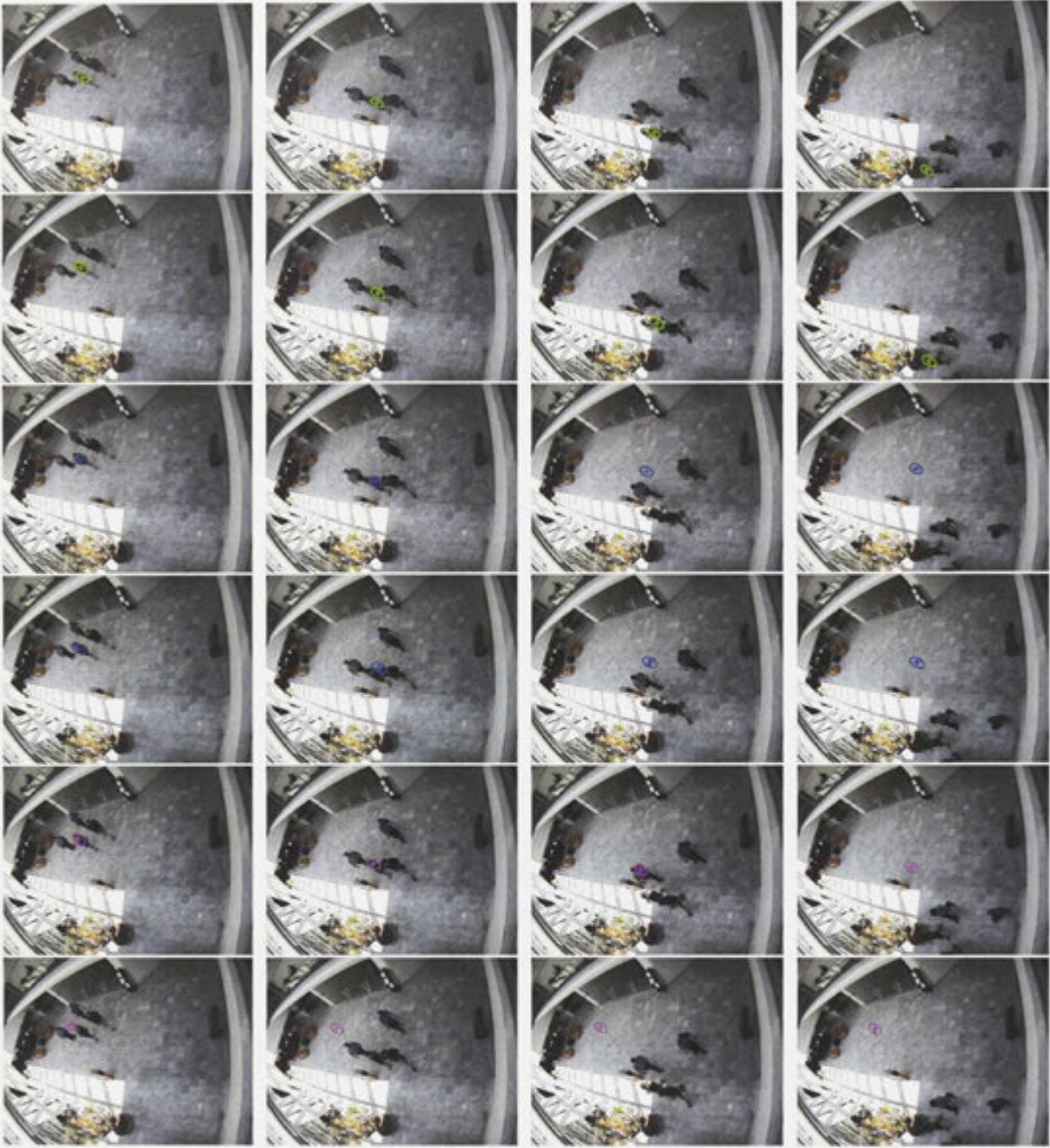


Figure 4.8: Results for the “Group Walk 1” sequence at frames 190, 250, 280 and 320. From top-to-bottom: results yielded by our algorithm using the 49-feature set and the 11-feature set (first and second rows), the on-line VR-based tracker (Collins et al., 2005) using the 49-feature set and the 11-feature set (third and forth rows), and the on-line PCA-based tracker (Han and Davis, 2004) using the 49-feature set and the 11-feature set (fifth and sixth rows).

description of the mean-shift tracker used in (Collins et al., 2005) and (Han and Davis, 2004).

In Figure 4.7, we present the sample results for frames 250, 290, 380 and 420 of

the PETS-ECCV 2004 "Meet and Walk Together 1" sequence. In this sequence, the target appearance varies remarkably as it moves from the bright area into the shade between frames 290 and 310. As a result, the target model is subjected to significant change. Moreover, the target remains close to the other person from frame 330 onwards, which serves as a confounding factor that further complicates the tracking task. Despite these difficulties, the feature combination approach presented here allows the tracker to follow the target throughout the scene. This applies to both of the feature sets under consideration. The VR-based tracker (Collins et al., 2005), on the other hand, loses the target as the subject approaches the other person between frames 380 and 420. The PCA-based approach (Han and Davis, 2004), however, cannot adapt to the significant change in illumination and subsequently fails in tracking the target from frame 290 until the end of the footage.

We present a more quantitative analysis of the tracker performance in Figure 4.9(a) and (b). In these figures, we have plotted the target centre error as a function of frame index with respect to the ground truth provided with the PETS-ECCV 2004 dataset. For the sake of clarity, the error in the figure is shown in a logarithmic scale. Note that our tracker has the lowest mean target centre errors of 5.63 ± 2.77 pixels and 4.40 ± 3.44 pixels for the 49-feature set and the 11-feature set, respectively. The VR-based tracker (Collins et al., 2005), has a mislocalisation mean of 15.40 ± 15.44 pixels and 16.22 ± 13.69 pixels, respectively. The PCA-based tracker (Han and Davis, 2004), being unable to track the target after frame 290, has a mean centre error of 49.86 ± 22.34 pixels and 48.27 ± 24.84 pixels. This is consistent with the behaviour described above.

We now turn our attention to the contribution of each feature to the global "strong" shift across the sequence. Here, we consider the 11-feature case as shown in Figure 4.10 (a). During the sequence, there are 22 updates during the footage. The first 20 updates occur between frames 250 and 310, in which the target moves across the bright area to the shady region in the scene. As a result, the appearance of the target varies significantly. During this frame range, the features such as the vertical and horizontal 3-rectangle Harr-like are assigned the highest weights, with an overall contribution of approximately 60%. In contrast, colour-based features such as the brightness and the normalised RGB channels are given much lower weights, with a contribution of less than 10%. The last few updates occur after frame 320, corresponding to the frames where the target moves completely inside the shady area. These adjustments reduce the weight of the Harr-like features and increase the contribution of the image brightness.

Moving on to the second of our experimental vehicles, Figure 4.8 shows the results for frames 190, 250, 280 and 320 of the PETS-ECCV 2004 "Group Walk 1" video sequence. The sequence records a group of four people moving across the scene, of

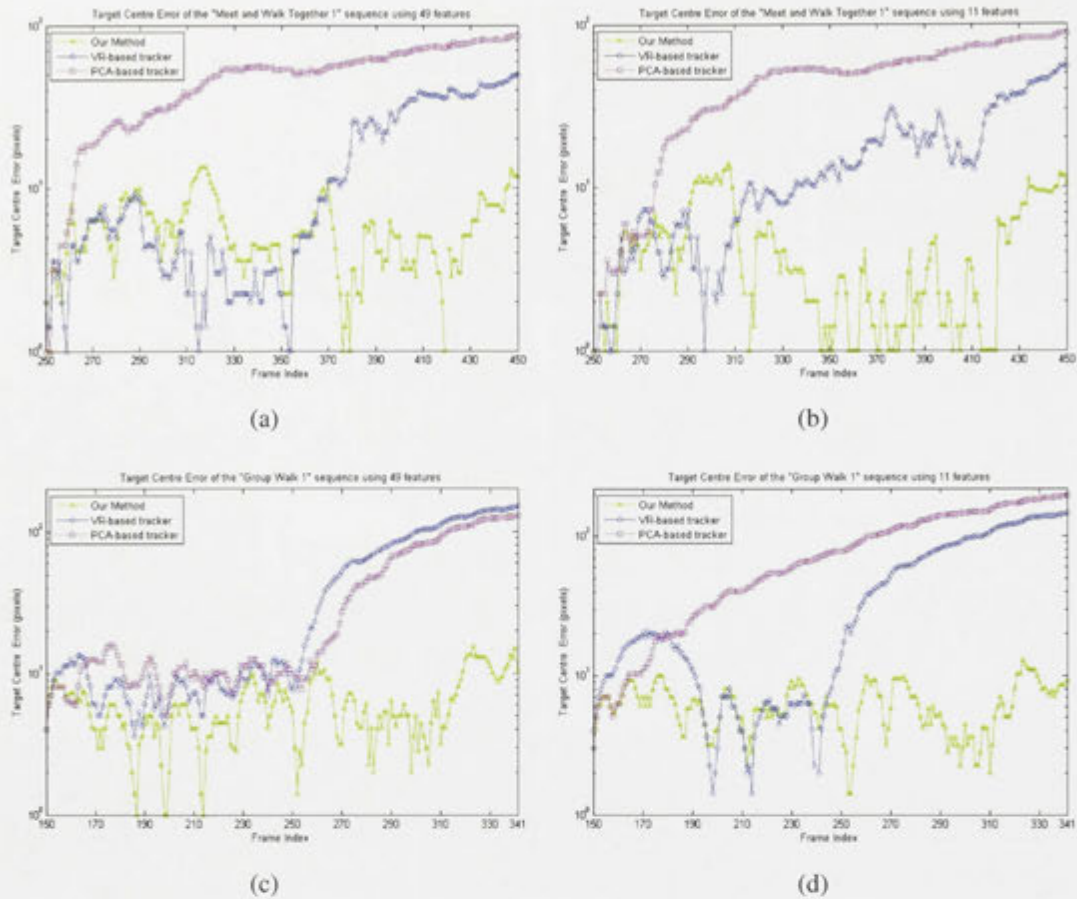
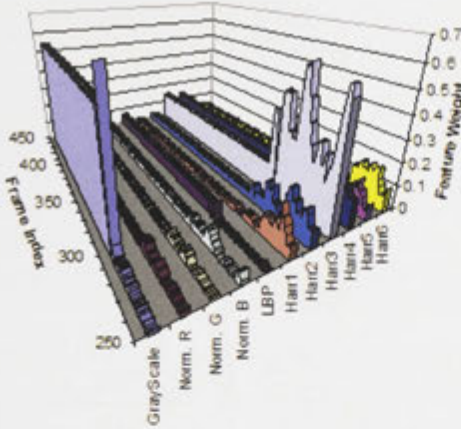


Figure 4.9: Target Centre Error for our method, the on-line VR-based tracker (Collins et al., 2005), and the on-line PCA-based tracker (Han and Davis, 2004). (a)(b): “Meet and Walk Together 1” sequence using the 49-feature set and the 11-feature set, respectively; (c)(d): “Group Walk 1” sequence using the 49-feature set and the 11-feature set, respectively.

which we track the female target. In this footage, there is no significant illumination change as in the previous sequence. However, as the group approaches the camera, their bodies overlap one another before exiting the scene. The similarity in their outfit colour further complicates the tracking task.

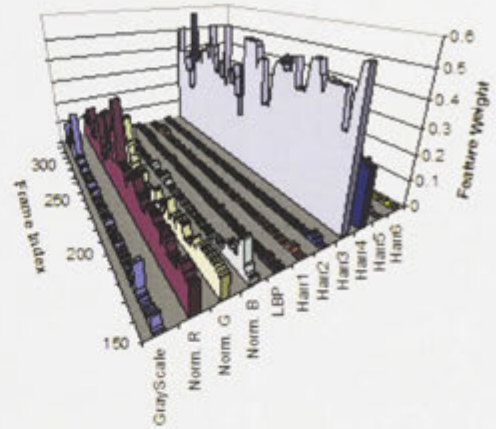
In this sequence, the VR-based tracker (Collins et al., 2005) performs well in the first 100 frames with both feature sets. However, it quickly loses the target once the target is partially occluded by another member of the group. This results in target centre errors of up to 46.47 ± 49.50 pixels and 46.75 ± 47.52 pixels, as shown in Figure 4.9 (c) and (d). The performance of the PCA-based tracker (Han and Davis, 2004) has high variation across the sequence. In particular, the PCA-tracker shows a

Feature Weight vs Frame for the "Meet and Walk 1" sequence



(a)

Feature Weight vs Frame for the "Group Walk 1" sequence



(b)

Figure 4.10: Individual weights for each of the feature in the 11-feature set used in our tracker for the "Meet and Walk Together 1" sequence (left-hand panel) and the "Group Walk 1" sequence (right-hand panel).

similar performance to that of the VR-based tracker when the 49-feature set is used. It also loses the target at the frames where the subject bodies overlap, being unable to recovery afterwards. However, in the 11-feature set case, the PCA-tracker only manages to track the target in the first 20 frames. As a result, the error measurements are significant, 38.42 ± 41.00 pixels and 86.65 ± 60.16 pixels for the 49-feature set and the 11-feature set, respectively. For our tracker, the model integrity is preserved as a consequence of the use of the total correlation as a measure of tracking confidence. Our tracker successfully follows the target throughout the scene with low target centre-errors, i.e. 6.09 ± 3.03 pixels and 6.25 ± 2.31 pixels for the 49-feature set and the 11-feature set, respectively. The results for the two sequences are tabulated in Table 4.1.

Figure 4.10(b) shows the contribution of each feature to the global "strong" shift, whereas there are 37 updates throughout the footage. These mainly occur when the subject bodies occlude one another. Nonetheless, the vertical 3-rectangle Harr-like feature is dominant across the sequence. From our experiments we also notice that the normalised RGB colour channels are not as discriminant as the other features in the set. This can be attributed to the fact that the clothing colour of the subjects in the scene does not separate the target from the rest of the crowd.

As can be seen from Figure 4.9, the performance of each tracker under study does not vary significantly with the two different sets of features, with a slight edge towards

Sequence name	Meet and Walk Together 1		Group Walk 1	
Feature set	49-feature	11-feature	49-feature	11-feature
Our algorithm	5.63 ± 2.77	4.40 ± 3.44	6.09 ± 3.03	6.25 ± 2.31
VR-based	15.40 ± 15.44	16.22 ± 13.69	46.47 ± 49.50	46.75 ± 47.52
PCA-based	49.86 ± 22.34	48.27 ± 24.84	38.42 ± 41.00	86.65 ± 60.16

Table 4.1: Target Centre Error as pixel distance for the two sequences “Meet and Walk Together 1” and “Group Walk 1”.

the 11-feature set as it embodies both colour and texture information. To take our analysis further, we show the robustness of our algorithms on three additional challenging sequences. The first two sequences are from the the CAVIAR project depicting a shopping centre scene in which the target appearance is highly correlated to that of the background and the target, at some point, is temporarily occluded by other pedestrians. The last sequence is the thermographic sequence used in Section 4.1.1.2, in which there is an abrupt change in the illumination due to camera self-calibration.

Furthermore, in Figure 4.11, we present the sample results for frames 2170, 220, 290 and 360 of the PETS-ECCV 2004 “EnterExitCrossingPath1fornt” sequence. In this sequence, the target exits a shop and proceeds to walk across the scene. Here, the outfit colour of the target is similar to the light of the background and thus makes it challenging to distinguish the target from the clothes racks of the shop. As a result, the VR-based tracker and the PCA-based tracker are unable to follow the target after the first 20 frames and are trapped in the region with similar appearance to that of the target inside the shop. Our tracker, on the other hand, manages to track the target throughout the scene. Due to the lack of the ground truth, the performance of each tracker is evaluated here based on observing the result directly.

In a similar scenario, Figure 4.12 shows yet another challenging sequence from the PETS-ECCV 2004 database, namely, “WalkByShop1” between frames 1600 and 1900. In this sequence, the target proceeds to enter the scene from the left corner, browse the clothes rack for a period of time before exiting the scene to the right. Here, the target size is significantly bigger than that of other sequences and, apart from the colour similarity of the the target outfit and the clothes in the shop, there is a temporary total occlusion occurring between frames 1865 and 1878, in which the target is completely overlapped by the two pedestrians. Our algorithm, being to cope and update the target model, follows the target smoothly throughout the sequence. The VR-based tracker and the PCA-based tracker, however, are trapped at the shop window at the beginning of the sequence and are unable to recover due to the fact that the model histograms



Figure 4.11: Results for the “EnterExitCrossingPath1front” sequence at frames 170, 220, 290 and 360. From top-to-bottom: results yielded by our algorithm using the 11-feature set, the on-line VR-based tracker (Collins et al., 2005) using the 11-feature set, and the on-line PCA-based tracker (Han and Davis, 2004) using the 11-feature set.

have been corrupted with “poor” samples after the initial mislocalisation.

To take our analysis further, we revisit the example of a thermographic video depicting a cat jumping on a fence in Section 4.1.1.2. In this sequence, there is an abrupt change in the scene illumination due to camera self-calibration. Despite this problem, the tracker presented here was able to maintain consistent tracking on the target, with a few mislocalisations at the peak of the illumination variation. The VR-based tracker and the PCA-tracker, on the other, are unable to perform the tracking task in this extreme scenario.

4.7 Conclusions

In this chapter, we have presented a feature combination approach for object tracking. We have shown how the target-centre may be recovered from a weighted linear combination of “weak” mean-shifts. This feature combination method is based upon graph embedding techniques. Thus, it provides a principled link between feature combination, graph-spectral methods and graphical models. The method performs on-line updating based upon the correlation between the target current model and that of the new target position at the current frame. The updating scheme presented here is gov-



Figure 4.12: Results for the “WalkByShop1” sequence at frames 1610, 1700, 1865 and 1878. From top-to-bottom: results yielded by our algorithm using the 11-feature set, the on-line VR-based tracker (Collins et al., 2005) using the 11-feature set, and the on-line PCA-based tracker (Han and Davis, 2004) using the 11-feature set.

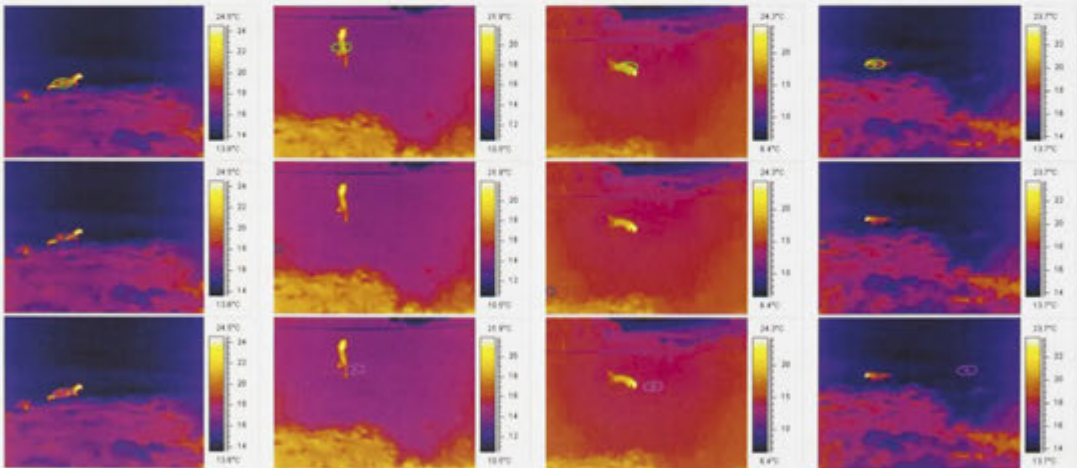


Figure 4.13: Results for the “Cat” sequence at frames 60, 130, 170 and 350. From top-to-bottom: results yielded by our algorithm using the 11-feature set, the on-line VR-based tracker (Collins et al., 2005) using the 11-feature set, and the on-line PCA-based tracker (Han and Davis, 2004) using the 11-feature set.

erned by the reliability of the tracking process. As a result, our method can cope with confusing backgrounds, unexpected fast movements and temporary occlusions by taking advantage of the information drawn from multiple feature spaces corresponding to a number of visual cues. The approach is quite general in nature and can employ other features elsewhere in the literature. We have also compared our results to those delivered by alternative methods. To this end, our algorithm shows remarkable gain in tracking performance throughout the experiments.

Here, it is worth noting that, through Chapter 3 and Chapter 4, we have presented several advanced methods for the kernel-based tracking algorithm to enhance robustness against target deformations in scale, shape and orientation, as well as illumination variations and confusing backgrounds. Even though these methods can provide a certain degree of resistance to temporary and partial occlusions, their performance can suffer from full occlusions for an extended period of time. This limitation also affects most other tracking algorithms in the single-view domain, but can be negated by the use of multiple cameras to provide additional viewpoints, to which we will explore in the next chapter.



Chapter 5

3D Kernel-Based Object Tracking

In this chapter, we explore the possibility of deploying the kernel-based tracking algorithm in a multi-camera environment where a number of cameras share some overlapped field of views of the scene. We do this by firstly providing a 3D formulation for the mean-shift procedure, in which the object is viewed as a 3D cube evolving inside a 3D space resembling the tracking environment. To this end, we present a point mapping strategy, which makes use of a reference object and exploits the consistency of vanishing points, and hence vanishing lines, between cameras, so as to estimate the head of the target in each participating view. This treatment not only provides a means to consistently map an arbitrary elevated point from the 3D scene to each camera view, but also allows us to set the upper bounds for the target region in each view at the initial frame. Furthermore, we discretise the target as a number of parallel slides in the 3D target volume and, subsequently, make use these upper bounds to compute a number of homography matrices mapping these slides from the 3D space to each camera view. This treatment naturally lends itself to the recovery of the camera weights, which measure how well the target is seen in each view. With these ingredients, the tracking task can be carried out in the 3D space, with sample points drawn from each view, while the results can be back-projected to any view of interest even if the object is fully occluded in that view.

We illustrate the utility of our tracker in real-world sequences and compare our results to that of a benchmark algorithm, i.e. the synergy map method proposed by Khan and Shah (2006). Throughout the experiments, our method shows better performance than the alternative, especially in the case of two and three cameras. The objective of the work in this chapter, however, is twofold. Firstly, we aim to show how the kernel-based tracker can be deployed in a 3D setting to cope with the problem of full occlusions. Secondly, this extension opens-up the possibility of a 3D implemen-

tation for the kernel-based methods presented in the previous chapters as well as other existing 2D methods elsewhere in the literature. With that in mind, we commence this chapter by introducing the 3D mean-shift formulation in the following section.

5.1 Mean-shift in 3D

The use of multiple cameras for tracking purposes, as discussed in Section 2.5, has gained increasing attention in recent years thanks to the advancements in computing power, camera technology and the sophistication of modern tracking algorithms. The main advantage of a multi-view setting over a single-view one comes from the better coverage of the scene provided by the redundant information from additional view-points. Naturally, multi-view tracking algorithms attempt to reduce the occluded regions in the scene by combining different Fields of View (FoVs) from the associated cameras into a unified framework. Once established, the tracking task can be carried out in this unified environment making use of various methods such as (Calderara et al., 2005; Khan and Shah, 2006). To this end, we have provided the preliminaries and the related work in this field in Section 2.5.1 and Section 2.5.2, respectively.

Recall that, in the previous chapters, we have shown that the kernel-based tracker can be deployed in different coordinate systems such as the image coordinates and the polar coordinates (Section 3.3), in conjunction with a number of arbitrary image features such as colour, contrast and texture (Section 4.1). Here, we extend the kernel-based tracking algorithm to a 3D setting by introducing the z -dimension into the mean-shift optimisation scheme. This treatment hinges in our motivation to cast the tracking task into a unified 3D framework, where the probability density functions of the target model and the candidate regions can be constructed from a set of 3D sample points, and subsequently, the new target centre can be computed via the mean-shift procedure in 3D.

To commence, it is worth revisiting the mean-shift algorithm on a 2D image. Here, the new object position η_i in view i can be determined via a 2D mean-shift as follows

$$\eta_i = \frac{\sum_{\mathbf{x}_n \in \mathcal{X}} \mathbf{x}_n \mathcal{G}(\mathbf{x}_n) w_n}{\sum_{\mathbf{x}_n \in \mathcal{X}} \mathcal{G}(\mathbf{x}_n) w_n} \quad (5.1)$$

where w_n is the similarity weight for the pixel $\mathbf{x}_n$ in the search window $\mathcal{X} = \{\mathbf{x}_j\}_{j=1 \dots K}$ and $\mathcal{G}(\cdot)$ is the shadow kernel of the kernel profile $\mathcal{K}(\cdot)$. For further details on the equation above, we refer the reader to the preliminaries of the kernel-based tracker presented in Section 2.4.2.1.

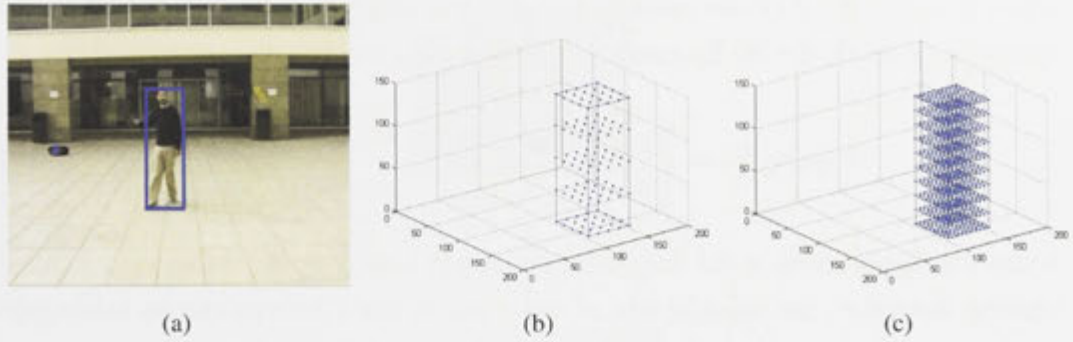


Figure 5.1: Difference between the sampling strategies of the 2D and 3D cases: (a) Sample points are discretised into pixels in a 2D image. (b,c) Sample points are discretised into a set of 3D points in a volume by different sampling rate, i.e. sparser in (b) and denser in (c).

To construct the mean-shift in 3D, we consider a scene observed by N cameras whose camera centres are $\{C_i\}_{i \in N}$. These cameras share some overlapped view of the ground plane, on which the tracking target is represented as a 3D cube as shown in Figure 5.1(b) and (c). As we move from 2D to 3D, by introducing the z -component, the input parameters for the mean-shift summation in Equation 5.1 need to be adjusted accordingly.

Firstly, the sampling window $\mathcal{X}$, for either the target model or the candidate regions, becomes a sampling volume consisting of K sample points, i.e. $\mathcal{V} = \{\mathbf{X}_j\}_{j=1 \dots K}$, which can be a cube or a sphere in the 3D scene based on the tracking preferences. As a result, a sampling rate, say s_r , must be defined to discretise this 3D volume $\mathcal{V}$ into a collection of 3D points. To this end, a high sampling rate will produce a finer and denser set of sample points, yet comes with higher computation cost. A low sampling rate, on the contrary, is faster to compute but can potentially miss out important feature points, and subsequently results in poor tracking performance. These differences are depicted in Figure 5.1(a) and (b) as an example of a tracking target viewed in 2D and 3D, respectively.

Secondly, the kernel profile $\mathcal{K}(\cdot)$, which regulates the weight of a sample point based on how far it is from the object centre, now consists of a 3D Euclidean distance between the sample point and the object centre in the 3D space. For instance, we consider the Epanechnikov kernel with its generic profile

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = \begin{cases} \frac{1}{2} \frac{(d+2)}{h^2 c_d} \left(1 - \frac{\|\mathbf{x}-\mathbf{y}\|^2}{h^2}\right) & \text{if } \left\| \frac{\mathbf{x}-\mathbf{y}}{h} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

where $\mathbf{y}$ is the object centre position and c_d is the volume of the unit d -dimensional sphere. As a result, the 3D Epanechnikov kernel, $\mathcal{K}_{3D}$, can be constructed as follows

$$\mathcal{K}_{3D}(\mathbf{X}, \mathbf{Y}) = \begin{cases} \frac{15}{8\pi h^2} \left(1 - \frac{\|\mathbf{X}-\mathbf{Y}\|^2}{h^2}\right) & \text{if } \left\| \frac{(\mathbf{X}-\mathbf{Y})}{h} \right\| < 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

where the bandwidth h is the volume of the object being tracked. Note that, in most tracking scenarios, the physical size of the object in the 3D scene can be safely presumed to be relatively constant in time, hence its 3D bandwidth h remains unchanged throughout the tracking sequence. This is advantageous over the 2D case since an on-line bandwidth updating scheme such as those presented in Chapter 3 is not necessary.

Thirdly, the power density functions (PDFs) of the 3D case, i.e. the M -bin histograms $\mathbf{Q}$ and $\mathbf{P}$, now consist of a set of M -bin histograms $\{\mathbf{Q}_i\}_{i \in N}$ and $\{\mathbf{P}_i\}_{i \in N}$ corresponding to the target model and the search volume in each view i of the N cameras observing the scene. These M -bin histograms are constructed from the projected sample points of the sampling volume $\mathcal{V}$ from the 3D space into each view i . We will discuss this projection at a later section in this chapter, for now, let the function $m_i : \mathcal{V} \mapsto \{\mathbf{x}_{ij}\}_{i \in N, j \in K}$ define the mapping of the set of 3D sample points $\mathcal{V}$ to a set of 2D sample points $\{\mathbf{x}_{ij}\}_{j \in K}$ in viewpoint i . Subsequently, the M -bin histograms $\mathbf{Q}_i$ and $\mathbf{P}_i$ can be computed as follows

$$\hat{q}_{i\psi} = C \sum_{j=1}^K \mathcal{K}(\mathbf{X}_j, \mathbf{Y}) \delta[b(\mathbf{x}_{ij}) - \psi] \quad (5.4)$$

$$\hat{p}_{i\psi} = C' \sum_{j=1}^{K'} \mathcal{K}(\mathbf{X}_j, \mathbf{Y}) \delta[b(\mathbf{x}_{ij}) - \psi] \quad (5.5)$$

where δ is the Kronecker delta function; C and C' are the normalisation constants of the target volume and the search volume made of K and K' sample points, respectively. Note that, here we have used the 2D projected point $\mathbf{x}_{ij}$ to sample the image feature values in view i , while the kernel weight $\mathcal{K}(\cdot)$ is computed using the original 3D point $\mathbf{X}_j$ in the 3D space.

Finally, with these ingredients, we can proceed to estimate the new target centre in the 3D space making use of the 3D mean-shift procedure as follows

$$\eta = \frac{\sum_{\mathbf{X}_j \in \mathcal{V}} \mathbf{X}_j \mathcal{G}(\mathbf{X}_j) w(\mathbf{X}_j)}{\sum_{\mathbf{X}_j \in \mathcal{V}} \mathcal{G}(\mathbf{X}_j) w(\mathbf{X}_j)} \quad (5.6)$$

in which the weight $w(\mathbf{X}_j)$ associated with sample point $\mathbf{X}_j$ is the 3D version of the

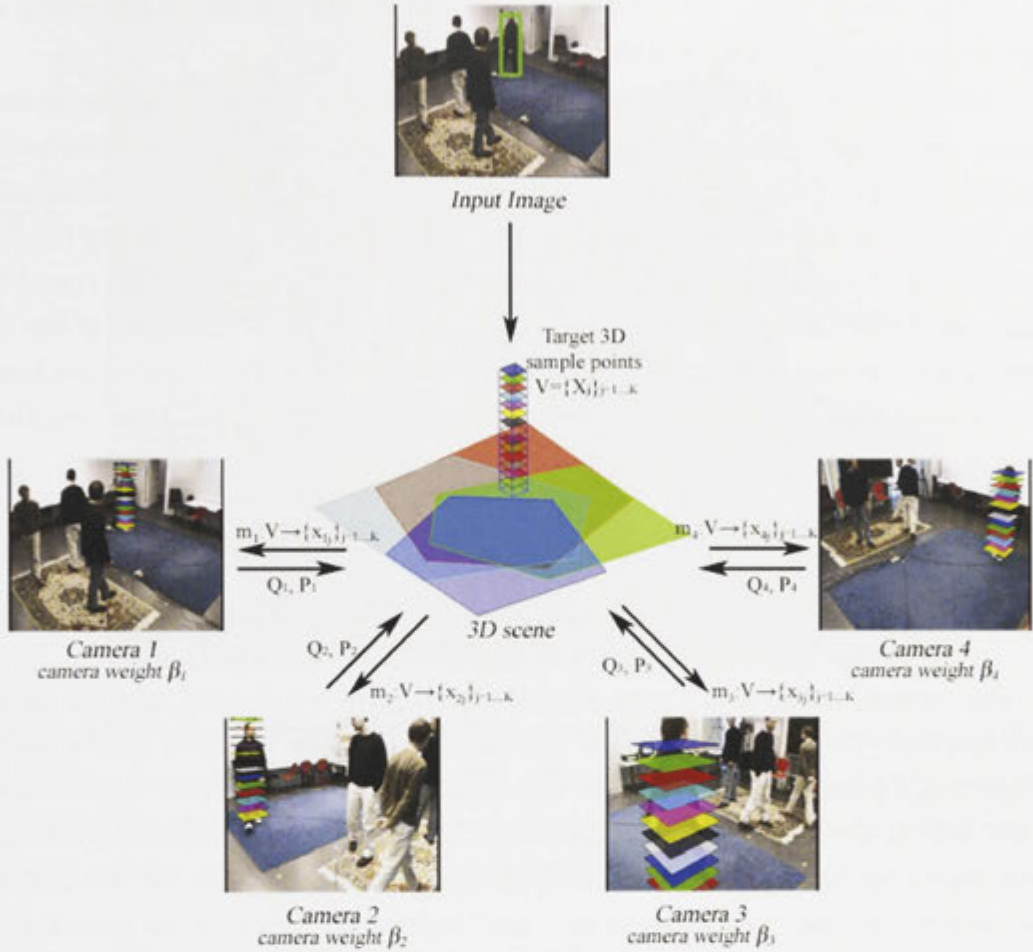


Figure 5.2: 3D Kernel-based Object Tracking. The sample points of the target in the 3D scene are mapped to each camera view i through the mapping m_i . The mapped points are used to construct the corresponding M -bin histograms Q_i and P_i , which are then used in the 3D mean-shift as in Equation 5.6.

Bhattacharyya weight proposed in (Comaniciu et al., 2003)

$$w(\mathbf{X}_j) = \sum_{i=1}^N \sum_{\psi=1}^M \beta_i \sqrt{\frac{q_{i\psi}}{p_{i\psi}}} \delta[b(\mathbf{x}_{ij}) - \psi] \quad (5.7)$$

where β_i is the weight associated with view i and, for the sake of simplicity, we call it *camera weight* in this chapter. This camera weight β_i defines how well the target is sampled in view i . To this end, one can simply nominate this camera weight to be the average of the number of viewpoints, i.e. $\beta_i = 1/N$. This approach, however, can substantially lead to mislocalisation if the target is fully occluded, or being too close

to several camera views. To this end, we will address the problem of estimating an optimal set of camera weights later in this chapter.

With the formulation of the 3D mean-shift procedure at hand, in the following sections, we will propose a sampling strategy that provides consistent mapping for the set of 3D points $\mathcal{V}$ to each participating view i , i.e. $m_i : \mathcal{V} \mapsto \{\mathbf{x}_{ij}\}_{i \in N, j \in K}$. Furthermore, we make use of this sampling strategy to develop an on-line weight updating mechanism, so as to automatically adjust the camera weights $\{\beta_i\}_{i \in N}$ during the course of tracking. These implementations ensure high consistency in the operation of the 3D kernel-based tracker in the multi-camera environment under studies. Last but not least, we have summarised the process described above into Figure 5.2 for better visualisation purposes.

5.2 Consistent Point Sampling Across Multiple Views

In this section, we present a point sampling strategy to consistently map the set of 3D coordinates $\mathcal{V}$ from the 3D space into each camera view i . We do this by firstly presenting a point mapping procedure that maps an elevated 3D point to each camera view making use of a reference object as well as the consistency in vanishing points and vanishing lines between the participating cameras. We subsequently use this method to estimate the head of the target in each view and use the foot positions provided by the ground plane homography, so as to define the upper bound and lower bound of the target in each view. This treatment naturally lends itself to the recovery of multiple homography matrices of multiple parallel planes spanning the target volume, which effectively guarantees the consistency in mapping the sample points $\mathcal{V} = \{\mathbf{X}_j\}_{j \in K}$ from the 3D space into a set of 2D points $\{\mathbf{x}_{ij}\}_{j \in K}$ in each camera view i .

5.2.1 Vanishing Points and Vanishing Lines Consistency

To commence, we require some formalism. Here, we consider a scene containing a reference plane π being viewed by a set of N stationary cameras. As the reference plane π is fixed in the 3D space, its vanishing points are consistent across the images of the participating cameras. The corresponding vanishing lines, therefore, are also consistent across these images as well. In mathematic terms, let $\{\mathbf{v}_i^x\}_{i \in N}$, $\{\mathbf{v}_i^y\}_{i \in N}$ and $\{\mathbf{v}_i^z\}_{i \in N}$ be the sets of vanishing points corresponding to the x , y and z directions in each camera, respectively. As these vanishing points in each camera refer to the

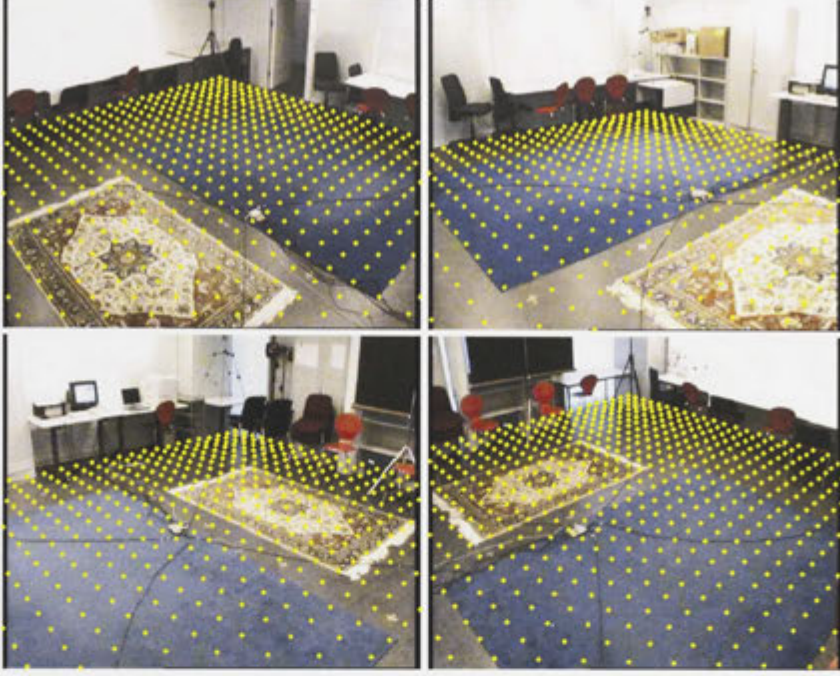


Figure 5.3: The ground plane is commonly selected as the reference plane for a multi-camera tracking scenario. In this example, the even sampling points (yellow dots) are computed by mapping a grid of points on the ground plane from the 3D scene to each camera making use of the corresponding homography matrices.

same vanishing points in the 3D scene, we have

$$\begin{aligned}
 \mathbf{v}_i^x &\equiv \mathbf{v}_{i'}^x \quad \forall i, i' \in N \\
 \mathbf{v}_i^y &\equiv \mathbf{v}_{i'}^y \quad \forall i, i' \in N \\
 \mathbf{v}_i^z &\equiv \mathbf{v}_{i'}^z \quad \forall i, i' \in N
 \end{aligned} \tag{5.8}$$

Let $\mathbf{l}_i^{xy} = \mathbf{v}_i^x \times \mathbf{v}_i^y$ be the vanishing line yielded by the two vanishing points $\mathbf{v}_i^x$ and $\mathbf{v}_i^y$ in view i where $\times$ denotes the cross product (Hartley and Zisserman, 2004). Similar to the 3D consistency of the vanishing points, we also have these vanishing lines in each camera refer to the same vanishing lines in the 3D scene. That is

$$\begin{aligned}
 \mathbf{l}_i^{xy} &\equiv \mathbf{l}_{i'}^{xy} \quad \forall i, i' \in N \\
 \mathbf{l}_i^{yz} &\equiv \mathbf{l}_{i'}^{yz} \quad \forall i, i' \in N \\
 \mathbf{l}_i^{xz} &\equiv \mathbf{l}_{i'}^{xz} \quad \forall i, i' \in N
 \end{aligned} \tag{5.9}$$

In practice, however, most applications employ only the horizon line, i.e. $\mathbf{l}^{xy}$, and the vertical vanishing point, i.e. $\mathbf{v}^z$ as shown in Figure 5.4. This is due to the fact that

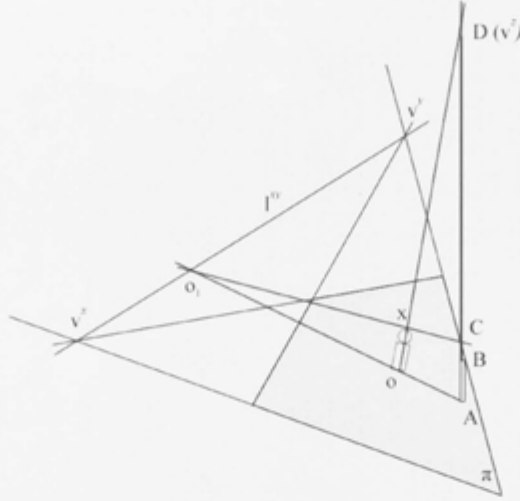


Figure 5.4: Diagram of the single-view metrology. Here, the ground plane is π whose vanishing line is l^{xy} and vertical vanishing point is D (or v^z). The object foot and head positions are o and x , respectively. The reference object is AB whose height is c_{ref} .

the ground plane is often selected as the reference plane and the interested objects often stand atop of the ground plane. It is also worth noting that these vanishing points can be automatically computed in a RANSAC framework such as those presented in (McLean and Kotturi, 1995; Schaffalitzky and Zisserman, 2000; Hartley and Zisserman, 2004).

5.2.2 Point Mapping Between Cameras Using Reference Object

In this section, we present a robust method to map an arbitrary sample point of the target from one view to another, using a reference object and the consistency of vanishing lines presented in the previous section. This projection procedure is similar to the single-view metrology proposed by Criminisi et al. (2000) to compute the height of an object given a reference object from a single image. However, it is worth noting that the objective of the method presented here is to project an elevated point x from view i to its corresponding point x' in view i' , rather than computing the object relative height with respect to the reference object.

Here, the reference object refers to an object that appears in both of the views of interest. This reference object is manually selected to be one that positions on top of the ground plane and in the same direction with the z -axis. For example, the reference objects can include the entrance door of a room, a table leg or even a human object at the initial frame of the sequence. Note that, if the projection matrices $\{P_i\}_{i \in N}$ of the participating cameras are known, a virtual reference object can be erected anywhere

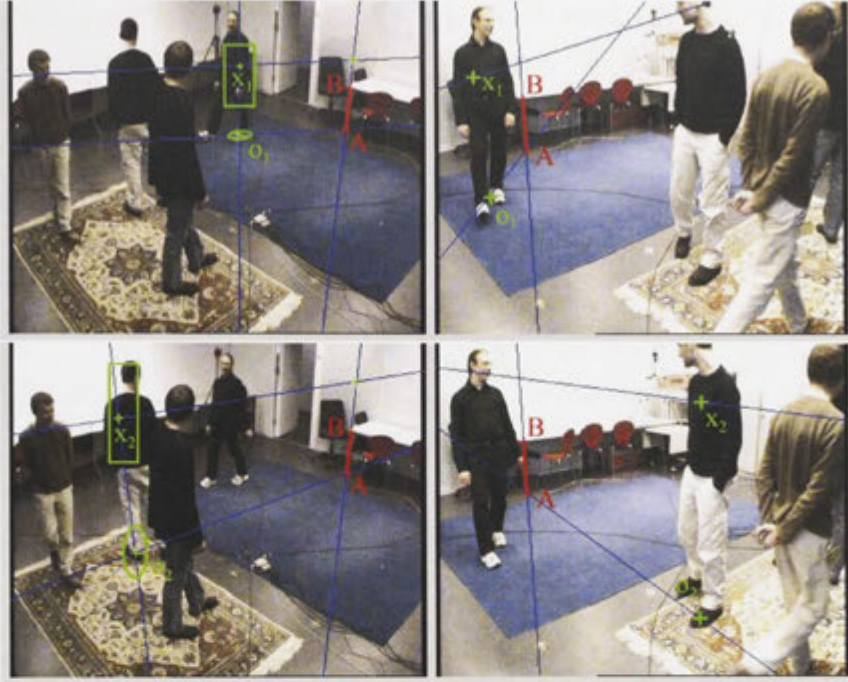


Figure 5.5: Point mapping between two camera views using the consistency of the vanishing lines and a reference object. The reference object (AB) is marked as a red line in both camera views. Top row: two points of the first target (o_1 and x_1). Bottom row: Two points of the second target (o_2 and x_2).

in the 3D scene, without the need for manual selection or visibility in each pairs of camera viewpoints. Although it is encouraged to have the real-life measurements such as width and height of the reference object, it is not strictly necessary. For instance, Figure 5.5 depicts a scene in which the reference object is selected as the leg of the table visible in both camera views. The height of the leg can be either its real-life measurement of 70cm or, without loss of accuracy, be denoted as a constant c_{ref} .

We now turn our attention to the mapping of an arbitrary sample point from one view to another. Let x and o be the sample point of the object and its corresponding ground position in view i , respectively. Also, let A and B be the bottom and the top of the reference object in view i whose vanishing point in the z -direction is D (see Figure 5.4). Similarly, let x' , o' , A' , B' and D' be the corresponding projected points in view i' . Furthermore, we define $H_{i \rightarrow i'}$ as the homography of the ground plane between views i and i' , i.e. $o' = H_{i \rightarrow i'} o$ and $A' = H_{i \rightarrow i'} A$. With the above information at hand, the goal is to compute the projected point x' in view i' .

To commence, we aim to compute the projected distance xo on the reference line

$l_{AB} = A \times B$. This can be done by firstly computing the intersection between the line l_{Ao} and the vanishing line l_i^{xy} , i.e. $o_1 = l_{Ao} \times l_i^{xy}$. The sample point x can then be projected to the reference line l_{AB} as the intersection between the line l_{o_1x} and l_{AB} as shown in Figure 5.4, i.e. $C = l_{o_1x} \times l_{AB}$.

Subsequently, the cross ratio of the four points A, B, C and D in view i and their corresponding projected points A', B', C' and D' in view i' are consistent as follows

$$\frac{AC}{AD} \frac{BD}{BC} = \frac{A'C'}{A'D'} \frac{B'D'}{B'C'} \quad (5.10)$$

Equation 5.10 can then be used to compute the projected point C' since A', B' and D' are predefined in the calibration process. In particular,

$$\frac{A'C'}{B'C'} = \frac{AC}{AD} \frac{BD}{BC} \frac{A'D'}{B'D'} = k \quad (5.11)$$

and since C' also lies on $l_{A'B'}$, that is

$$C' \cdot l_{A'B'} = 0 \quad (5.12)$$

where $\cdot$ is the dot product. To expand Equations 5.11 and 5.12 further, let $A' = [a'_1, a'_2, 1]^T$, $B' = [b'_1, b'_2, 1]^T$, $C' = [c'_1, c'_2, 1]^T$ and $l_{A'B'} = [l_1, l_2, l_3]^T$. Equations 5.11 and 5.12 then become

$$\sqrt{\frac{(c'_1 - a'_1)^2 + (c'_2 - a'_2)^2}{(c'_1 - b'_1)^2 + (c'_2 - b'_2)^2}} = k \quad (5.13)$$

$$l_1 c'_1 + l_2 c'_2 + l_3 = 0 \quad (5.14)$$

Subsequently, Equation 5.14 can be substituted into Equation 5.13 and solve for $C' = [c'_1, c'_2, 1]^T$. The projected sample point x' , as a result, can now be computed by reversing the projection process of finding C in view i . In particular,

$$x' = l_{o'D'} \times l_{o'_1 C'} \quad (5.15)$$

where $o'_1 = C' \times (l_{o'A'} \times l_i^{xy})$. Figure 5.5 shows an example of this point mapping method in which the sample points are projected between two different camera views. Essentially, the projection method presented here is twofold. First, it allows an arbitrary point in one view to be mapped to another view using a reference object and the homography of a reference plane. Second, this method is devoid of the computation for the homography of the plane that the arbitrary point lies upon, which can be challenging due to the large number of such planes lying between the target foot and head in a human tracking scenario.

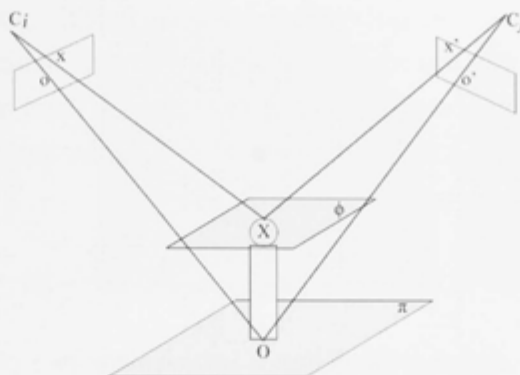


Figure 5.6: Multi-planar homography between parallel planes. The object is in the 3D scene whose reference (ground) plane is π . ϕ is an arbitrary plane parallel with the reference plane.

5.2.3 Multiple Planar Homography

Recall that, in Section 5.1, we aim to provide a consistent mapping method for the set of 3D points $\mathcal{V}$ to each participating view i . This set of 3D points consists of multiple parallel planes that span the 3D target volume defined by the upper bound and the lower bound of the target, i.e. the target head and the target foot positions, respectively. Here, we attempt to recover the set of homography matrices corresponding to these parallel planes, given the homography matrix of the ground plane and the upper bound of the target in each participating view. Note that the target upper bound in each view can be recovered making use of the mapping procedure presented in the previous section.

We commence this section by computing the homography of an arbitrary plane ϕ parallel with the ground plane π between two camera views i and i' whose homography matrices are $\mathbf{H}_{i_\phi i'}$ and $\mathbf{H}_{i_\pi i'}$ respectively. The computation of such homography matrix is critical for the sampling process due to the following reasons. Firstly, a multi-camera scene is often calibrated with respect to the ground plane, and hence, the only available homography matrix is $\mathbf{H}_{i_\pi i'}$. Secondly, each subject with different physical height will induce different set of planar homography matrices $\mathbf{H}_{i_\phi i'}$, therefore, a predefined set of planar homography matrices might not produce optimal performance for all tracking targets. It is also worth noting that the computation of these matrices is required only once at the beginning of the sequence. This is consistent with the assumption that the object physical height in the 3D space remains relatively constant throughout the course of tracking.

Here, we focus on a scenario in which the object is tracked in a 3D scene where the sample points used for the mean-shift operation are drawn from each camera image.

The object foot position of the last frame is known in the 3D scene and can be projected to each camera view i by the corresponding homography matrix $\mathbf{H}_{3D\pi i}$. Here, $\mathbf{H}_{3D\pi i}$ denotes the mapping of a point on the reference plane π from the 3D scene to camera i , as shown in Figure 5.6. Khan et al. (2007) show that the homography of the parallel plane ϕ , $\mathbf{H}_{3D\phi i}$, can be computed from $\mathbf{H}_{3D\pi i}$ as follows

$$\mathbf{H}_{3D\phi i} = (\mathbf{H}_{3D\pi i} + [0|\gamma\mathbf{v}_i^z])(I_{3\times 3} - \frac{1}{1+\gamma}[0|\gamma\mathbf{v}_{3D}^z]) \quad (5.16)$$

whereas γ is a scalar defining the distance between the two planes π and ϕ , $[0]$ is a 3×2 matrix of zeros and $\mathbf{v}_{ref}$ is the vanishing point of the reference direction.

To elaborate further, the second part of the right hand side in Equation 5.16, i.e. $(I_{3\times 3} - \frac{1}{1+\gamma}[0|\gamma\mathbf{v}_{3D}^z])$, maps the point $\mathbf{X}$ on plane ϕ to point $\mathbf{O}$ on the reference plane π in the 3D space. The first part of the right hand side, i.e. $(\mathbf{H}_{3D\pi i} + [0|\gamma\mathbf{v}_i^z])$, then maps point $\mathbf{O}$ to view i and subsequently to plane ϕ . In the case of the 3D scene, however, the homography matrix corresponding to the mapping $\mathbf{X}_{3D} \mapsto \mathbf{O}_{3D}$ will retain the x and y components. In particular, Figure 5.6 shows the 3D scene in which the mapping of point $\mathbf{X}$ on plane ϕ to point $\mathbf{O}$ on plane π results in the same x and y coordinates. As a result

$$I_{3\times 3} - \frac{1}{1+\gamma}[0|\gamma\mathbf{v}_{3D}^z] = \frac{1}{X_3}I_{3\times 3} \quad (5.17)$$

where we have used $\mathbf{X}_{3D} = [X_1, X_2, X_3]^T$ due to the fact that we are considering the 3D space as a collection of multiple planes parallel with the ground plane, where the mapping is provided by the 3×3 homography matrix $\mathbf{H}_{3D\phi i}$. Equation 5.16 is then reduced to

$$\mathbf{H}_{3D\phi i} = \frac{1}{X_3} (\mathbf{H}_{3D\pi i} + [0|\gamma\mathbf{v}_i^z]) \quad (5.18)$$

where the only unknown is the scalar γ , which can be computed making use of a pair of point correspondences between the two views. For instance, let $\mathbf{x}_i$ correspond to the top of the object in view i in which the height of the reference object is c_{ref} . Using single view geometry (Criminisi et al., 2000), the physical height of the object, i.e. the z -coordinate X_3 , is αc_{ref} , where α is a scalar. Substituting into 5.18 yields

$$\begin{aligned} \mathbf{x}_i &= \mathbf{H}_{3D\phi i} \mathbf{X}_{3D} \\ &= \frac{1}{\alpha c_{ref}} (\mathbf{H}_{3D\pi i} + [0|\gamma\mathbf{v}_i^z]) \mathbf{X}_{3D} \end{aligned} \quad (5.19)$$

where γ can be computed directly as follows

$$\gamma = \frac{x_1 \left(h_{31} \frac{O_1}{\alpha c_{ref}} + h_{32} \frac{O_2}{\alpha c_{ref}} + h_{33} \right) - \left(h_{11} \frac{O_1}{\alpha c_{ref}} + h_{12} \frac{O_2}{\alpha c_{ref}} + h_{13} \right)}{x_1 - v_1} \quad (5.20)$$

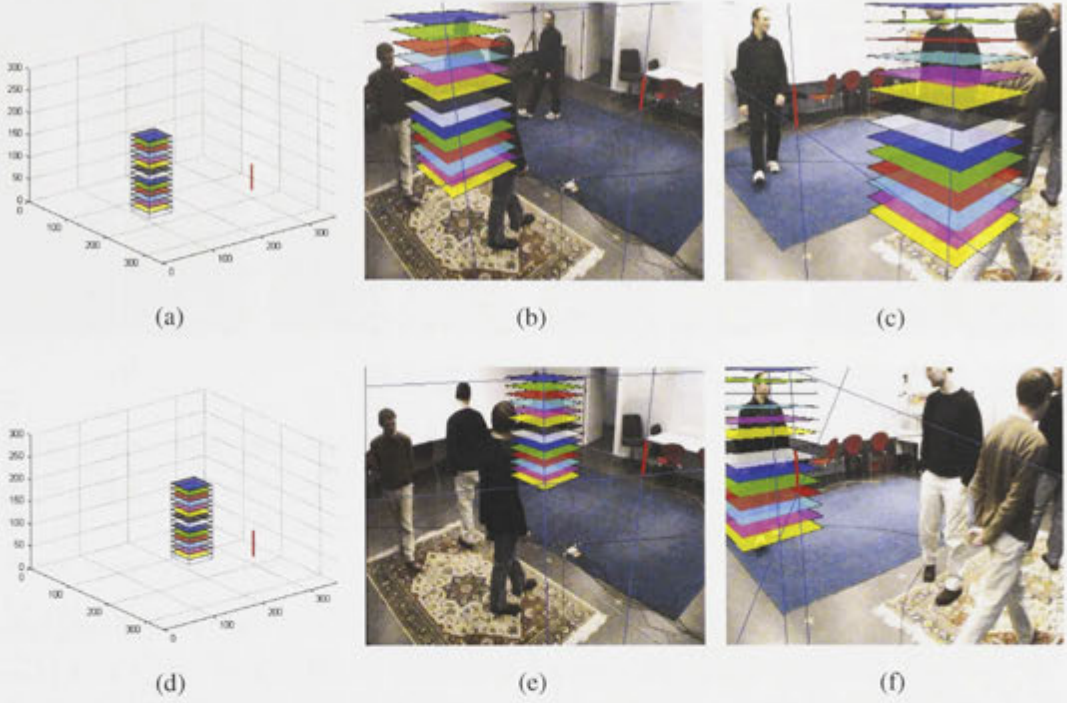


Figure 5.7: Multiple planar homography of 15 planes parallel with the ground plane that map a set of 3D points in space ((a) and (d)) to each camera view. Top row: Target number 1. Bottom row: Target number 2. The reference object is marked as a red line in both cases.

where $\mathbf{x}_i = [x_1, x_2, 1]^T$, $\mathbf{v}_i^z = [v_1, v_2, 1]^T$, $\mathbf{O} = [O_1, O_2, 1]^T$ and $h_{k_1 k_2}$ is the entry indexed k_1, k_2 of the homography matrix $\mathbf{H}_{3D \rightarrow i}$.

Here, we have shown how to compute the homography $\mathbf{H}_{3D \rightarrow i}$ given the correspondence pair $\mathbf{x}_i$ and $\mathbf{X}_{3D}$. This treatment allows the computation of a range of planar homography matrices bounded by γ_1 and γ_2 corresponding to the two points $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$. To illustrate this further, Figure 5.7 shows how this method can be used to compute a range of homography matrices between the head and the foot positions of a human object. In particular, the object is modelled as a cube in a 3D scene as shown in Figure 5.7(a) and (d), in which an arbitrary number of slides is chosen to sub-sample the occupied volume. Here, the reference object marked as a solid red line in each sub-figure is used to project the object head from one view to another. A set of homography matrices is then computed for each camera view as shown in Figure 5.7(b),(c),(e) and (f).

To sum up, the point sampling strategy presented here allows a set of 3D points in the 3D space, $\mathcal{V} = \{\mathbf{X}_j\}_{j \in K}$, to be projected into different viewpoint i , from which

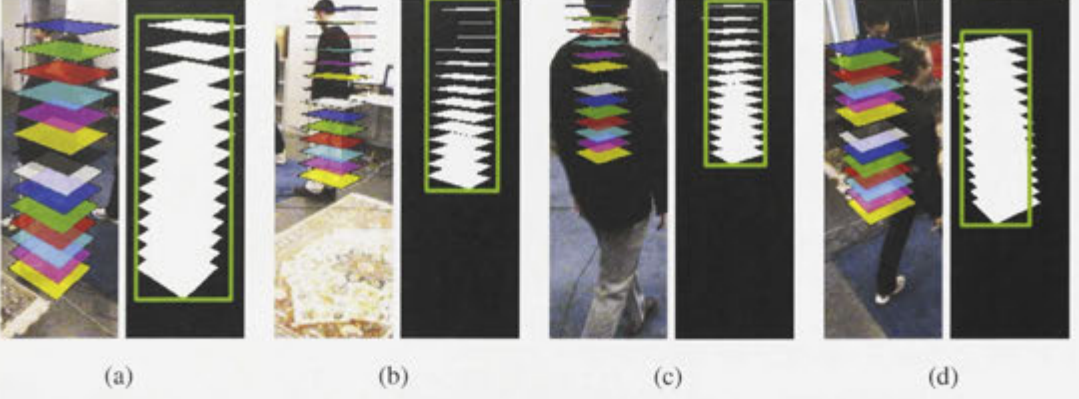


Figure 5.8: Sampling quality of various viewpoints. The tracking target and its corresponding set of sample points being viewed from different cameras: (a) Camera 1; (b) Camera 2; (c) Camera 3; (d) Camera 4.

the sample correspondences $\{\mathbf{x}_{ij}\}_{j \in K}$ are drawn to construct the M -bin histograms for the mean-shift procedure. This sampling method makes use of the ground plane homography between different viewpoints and a reference object, so as to derive an arbitrary number of planar homography matrices between two given points in each view, for example the head and the foot of a human object. This treatment, in turn, permits the flexibility in the number of planes or “slides” in the target volume by subdividing the range γ_1 and γ_2 into $|\Phi|$ intervals, which can be directly beneficial for the overall tracking performance.

5.3 Camera Weights Estimation

Recall that, in Section 5.1, the weight $w(\mathbf{X}_j)$ associated with each 3D sample point $\mathbf{X}_j$ is defined as a linear combination of the Bhattacharyya distances between the M -bin histograms in each participating view as shown in Equation 5.7. This combination is regulated by the set of *camera weights* $\{\beta_i\}_{i \in N}$, in which the individual weight β_i defines how “well” the target is sampled in view i . In this section, we quantify the “goodness” of each view based on the coverage of the sampling area and the cross-correlation between the corresponding M -bin histograms. In particular, we do this by computing the following quantity

$$\epsilon_i = \tau_i \varrho_i \quad (5.21)$$

where τ_i represents the fraction of the image area covered by the projected sample points $\{\mathbf{x}_{ij}\}_{j \in K}$, we call it $\hat{h}_i$, over the image area of the projected target in view i ,

we call it h_i . Figure 5.8 shows $\hat{h}_i$ as the cascaded white regions and h_i as the green rectangle in each camera view. That is

$$\tau_i = \frac{\hat{h}_i}{h_i} \quad (5.22)$$

On the other hand, ϱ_i is defined as the cross-correlation between the M -bin histograms of the target model $\mathbf{Q}_i$ and the candidate region $\mathbf{P}_i$ governed by the set of sample points $\{\mathbf{x}_{ij}\}_{j \in K}$. The definition of ϱ_i is similar to those presented in Section 4.6. That is

$$\varrho_i = \left\langle \frac{\mathbf{Q}_i}{\|\mathbf{Q}_i\|}, \frac{\mathbf{P}_i}{\|\mathbf{P}_i\|} \right\rangle \quad (5.23)$$

Here, the quantity ϵ_i ensures that the set of sample points drawn from view i is “good” for purposes of tracking only if it spans a reasonable area covering the target, and, at the same time, does not contain an excessive amount of noise due to occlusions. In particular, Figure 5.8 depicts a scene in which the tracking target is viewed in 4 cameras as (a), (b), (c), and (d) respectively. Here, the coverage areas of Camera 1 and 4 are much better than that of Camera 2 and 3. That is, τ is computed as ratio of the white-shaded area over the area enclosed within the green bounding box. Furthermore, Camera 3, while having a better coverage percentage than Camera 2, suffers a low cross-correlation due to the fact the target is full occluded in this view, and hence, yields the lowest quantity ϵ amongst others. On the contrary, Camera 1 and 4 induce the highest ϵ thanks to the target having better coverage and higher cross-correlation coefficients in these views.

With these ingredients, the set of camera weights $\{\beta_i\}_{i \in N}$ can be computed by normalising the set of quantity $\{\epsilon_i\}_{i \in N}$. That is

$$\beta_i = \frac{\epsilon_i}{\sum_{k \in N} \epsilon_k} \quad (5.24)$$

Here, it is worth noting that the target position and its perspective viewed in each camera, in most cases, do not vary significantly between two consecutive frames. The weight updating process presented here, therefore, should be carried out once every t frames, in which t can be 10, 20 or 30 frame intervals.

5.4 Algorithm Description

With the developments presented in the previous sections, the 3D kernel-based tracking algorithm can be divided into two stages, whose pseudo codes are presented in Algorithm 5 and 6, respectively.

Algorithm 5: 3D Training**Data:**

- A set of homography matrices of the reference plane $\{\mathbf{H}_{i_\pi i'}\}_{i, i' \in N}$.
- A reference object shared between each pair of viewpoints i and i' .
- The selected region of the target at the initial frame in view i .

begin

Compute the top and bottom pixels $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$ of the target bounding box in view i .

Compute the foot position of the target in the 3D scene as $\mathbf{X}_\pi = \mathbf{H}_{3D_\pi i} \mathbf{x}_{i2}$

Compute the nominated height of the target using single-view metrology (Criminisi et al., 2000).

Compute the target centre $\mathbf{X}_0$ in the 3D scene using $\mathbf{X}_\pi$ and the object height.

Project $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$ into the remaining views $i' \in N$ using the consistency of vanishing points and reference object as presented in Section 5.2.2.

Compute the set of homography matrices $\{\mathbf{H}_{3D_\phi i}\}_{\phi \in \Phi, i \in N}$ using the multiple planar homography method presented in Section 5.2.3.

Save $\mathbf{X}_0$, $\{\mathbf{H}_{3D_\pi i}\}_{i \in N}$ and $\{\mathbf{H}_{3D_\phi i}\}_{\phi \in \Phi, i \in N}$.

end

The first stage is the training phase, in which the user is required to manually select a tracking target in view i at the initial frame of the sequence. The tracking target is denoted by a bounding box whose top and bottom points, namely $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$, are then projected into the remaining views $i' \in N$ making use of the point mapping method described in Section 5.2.2. Subsequently, a set of $|\Phi|$ homography matrices $\{\mathbf{H}_{3D_\phi i}\}_{\phi \in \Phi, i \in N}$ spanning the volume $\mathcal{V}$ in the 3D scene can be computed via the multiple planar homography method presented in Section 5.2.3. Note that the size of the object the 3D scene can be approximated from the two points $\mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$ and the nominal height of the object can be found with respect to the reference object, c_{ref} , by employing the single-view metrology method (Criminisi et al., 2000).

In the second stage, the tracking vehicle is the 3D mean-shift presented in Section 5.1. We commence this stage by projecting the discretised volume $\mathcal{V} = \{\mathbf{X}_j\}_{j \in K}$ into each viewpoint i of the scene using the set of homography matrices $\{\mathbf{H}_{3D_\phi i}\}_{\phi \in \Phi, i \in N}$ derived from the training phase. This mapping procedure results in a set of 2D coor-

Algorithm 6: 3D Tracking

Data: $\mathbf{X}_0$, $\{\mathbf{H}_{3D_{\pi i}}\}_{i \in N}$ and $\{\mathbf{H}_{3D_{\phi i}}\}_{\phi \in \Phi, i \in N}$

begin

Project the set of 3D sample points $\mathcal{V} = \{\mathbf{X}_j\}_{j \in K}$ into each view i as $\{\mathbf{x}_{ij}\}_{j \in K}$ using $\{\mathbf{H}_{3D_{\phi i}}\}_{\phi \in \Phi, i \in N}$.

Compute the set of M -bin histograms $\{\mathbf{Q}_i\}_{i \in N}$ for the target model as in Equation 5.4.

Compute the initial camera weights $\{\beta_i\}_{i \in N}$ as in Equation 5.24.

Set the initial target centre $\mathbf{X}_c = \mathbf{X}_0$.

for $idx = \text{StartFrame}$ **to** EndFrame **do**

while *true* **do**

Project the set of 3D sample points for the search volume $\mathcal{V}' = \{\mathbf{X}_j\}_{j \in K'}$ into each view i as $\{\mathbf{x}_{ij}\}_{j \in K'}$ using $\{\mathbf{H}_{3D_{\phi i}}\}_{\phi \in \Phi, i \in N}$.

Compute the set of M -bin histograms $\{\mathbf{P}_i\}_{i \in N}$ for the search volume as in Equation 5.5.

Compute the new target centre η as in Equation 5.6.

if $\|\eta - \mathbf{X}_c\| \leq \varepsilon$ **then**

$idx = idx + 1$

break

else

Update the target centre in the 3D scene $\mathbf{X}_c = \eta$.

Update the foot position of the target centre in the 3D scene $\mathbf{X}_{\pi}$.

end

end

Project the target position back to each camera view i

$\mathbf{x}_i = \mathbf{H}_{3D_{\pi i}} \mathbf{X}_{\pi}$

for *Every t frames* **do**

Update the camera weights $\{\beta_i\}_{i \in N}$ as in Equation 5.24.

end

end

end

dinates $\{\mathbf{x}_{ij}\}_{j \in K}$, which can be used to construct the M -bin histograms for either the target model $\mathbf{Q}_i$ or the candidate region $\mathbf{P}_i$. These histograms are then fed directly into the mean-shift operation as in Equation 5.6, where the set of camera weights $\{\beta_i\}_{i \in N}$ is computed via the method described in Section 5.3. After each successive frame, the

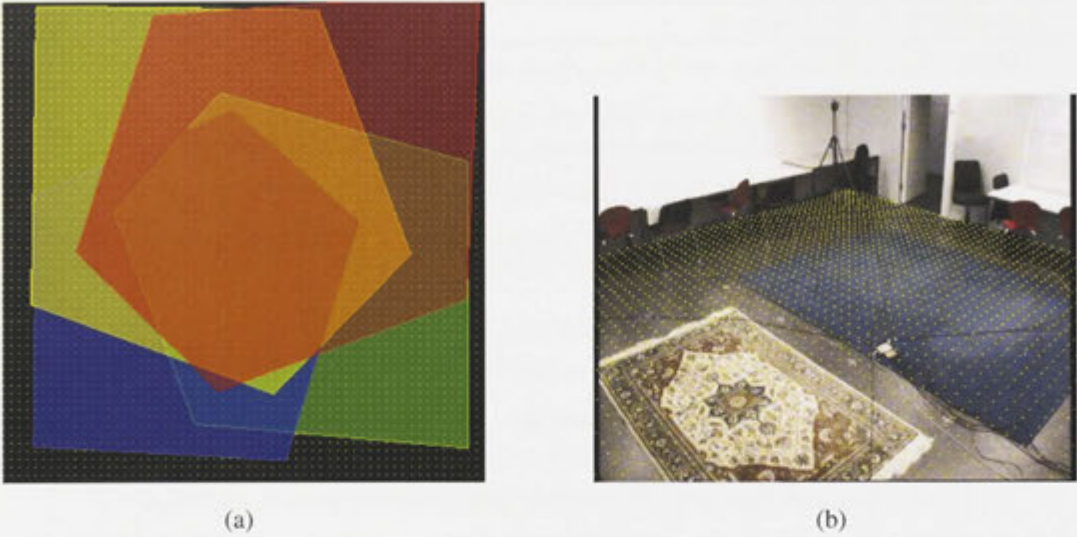


Figure 5.9: The ground plane of the “Laboratory” sequence is discretised into a 59×59 grid coordinates viewed in the topview image (a); and in Camera 1 image (b). The coloured shades show each camera coverage of the scene, or in particular, red, green, blue and yellow correspond to view of Camera 1, 2, 3 and 4, respectively.

new object position is computed by applying the 3D mean-shift on a search volume $\mathcal{V}'$ with greater or equal size to that of the original object volume $\mathcal{V}$.

5.5 Experiments

In this section, we illustrate the effectiveness of our tracking algorithm by presenting the results on two sequences from the POM-EPFL multi-camera pedestrian dataset¹. The first sequence, called the “Laboratory” sequence, depicts a laboratory room observed by four cameras, in which there are four people moving around the scene. The second sequence, called the “Terrace” sequence, were shot outside of a building on a terrace. In this sequence, there are up to seven people evolving in front of four cameras. For both sequences, the cameras are mounted in the corners of the scene and the ground plane homography matrices of the cameras with respect to the bird eye view of the scene is provided. Furthermore, these bird eye views, called *topview* for the sake of convenience, are discretised into a grid system such as one shown in Figure 5.9, in which the ground plane of the “Laboratory” sequence is a 59×59 grid.

We have compared our results to those yielded by a benchmark algorithm by Khan

¹POM-EPFL dataset can be accessed from <http://cvlab.epfl.ch/data/pom/index.html>

and Shah (2006) for the following reasons. Firstly, both algorithms do not require the scene to be fully calibrated, or in other words, the ground plane homography is sufficient for tracking purposes. Secondly, our algorithm relies on a reference object during the initialisation process to derive the set of planar homography matrices, while the alternative (Khan and Shah, 2006) requires the number of targets in the scene to be fixed to achieve robust segmentation. Finally, the alternative tracker is a well-known in the literature for its tracking performance that increases proportionally with the number of cameras observing the scene. For more details about this algorithm, we direct the reader to Section 2.5.2. Here, it is important to note that both of the method presented in this chapter and the benchmark algorithm (Khan and Shah, 2006) only work for tracking scenarios in which the target stays on top of a reference plane. This is due to the fact the both methods fundamentally rely on the homography mapping of the reference plane between a number of camera views to initiate and maintain the tracking task.

For our implementation of the algorithm in (Khan and Shah, 2006), we have used the Gaussian-Mixture-Model (GMM) background subtraction method in (Zivkovic and Heijden, 2006), an improved algorithm over the original method proposed by Stauffer and Grimson (1999), so as to obtain the synergy map from the foreground images of the camera under study. This synergy map, prior to the segmentation process, is undergone a thresholding procedure using empirically chosen values that yield the best possible results for each sequence. For the segmentation process, we have used the normalised graph-cut method described in (Shi and Malik, 2000) to segment the synergy map into different clusters corresponding to different targets. The label of each target is then passed on the next successive topview image to maintain the identity of each target during the tracking course.

For the implementation of our tracker, we deploy the 3D mean-shift in the 3D scene constructed by the homography matrices of the topview image, of which the ground plane is discretised into a grid coordinates and the height of the scene is set to 3 metres. To construct the M -bin histograms for the target model and the search volume, we employ the 3D Epanechnikov kernel presented in Equation 5.3 with the size of the search volume set to 1.5 times of the target volume. We also set the update interval to be 20 frames for both sequences, due to the fact the tracking targets in both sequences evolve around the scene in a relatively slow speed. Furthermore, these 3D volumes are discretised to a number of parallel planes, or more specifically, between 5 and 40 planes. We choose the default number of parallel planes to be 20 and will also provide quantitative analysis on the tracking performance versus the number of planes used in each sequence. Finally, for both sequences, the target is selected by the user at

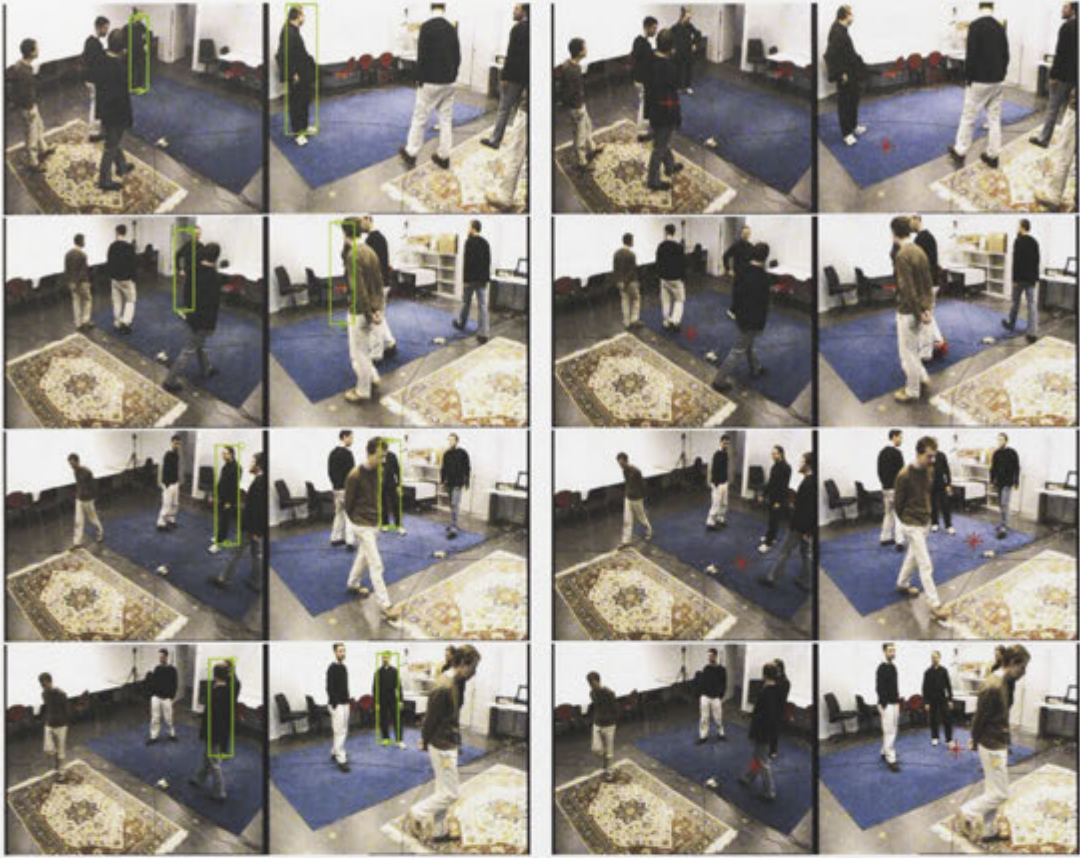


Figure 5.10: Results for the “Laboratory” sequence using 2 cameras at frames 1730, 1790, 1880 and 1905. Left: results yielded by our algorithm. Right: results yielded by the synergy map method (Khan and Shah, 2006).

the initial frame and the tracking course is performed on two, three and four cameras.

In Figure 5.10, we present the sample results for frames 1730, 1790, 1885 and 1905 of the “Laboratory” sequence using two cameras. In this sequence, there are four people constantly moving around the scene, amongst which the target experiences several full occlusions in both camera views. In particular, the target is completely occluded in Camera 1 at frames 1780 and 1905, while full occlusion occurs between frames 1800 to 1880 in Camera 2. Here, we make use of the leg of the table next to the entrance door in Camera 1 view as the reference object and, using the method presented in Section 5.2.2, estimate the target head in Camera 2. The set of the planar homography matrices is then computed to facilitate the 3D sampling scheme. Note that, this procedure is performed only at the initial frame. The 3D mean-shift presented here, subsequently, takes over the tracking task and successfully follows the target throughout the sequence. Moreover, using the head estimation at the initial frame, the

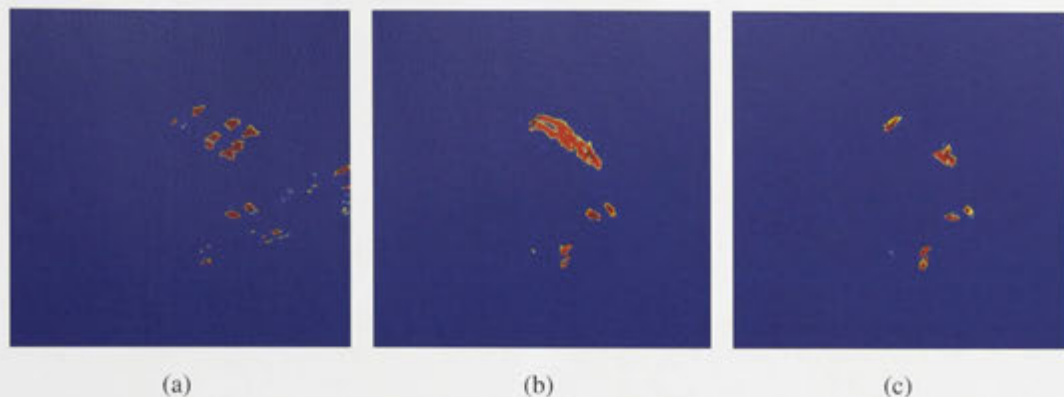


Figure 5.11: The drawback of the synergy map method (Khan and Shah, 2006): the quality of the synergy map is highly dependant on the number of cameras observing the scene. From left to right: synergy map generated at frame 1870 of the “Laboratory” sequence by (a) two cameras, (b) three cameras and (c) four cameras.

bounding box of the target is correctly projected across the cameras, at every frame. This hinges in the assumption that the object volume in the 3D scene does not vary significantly during the course of tracking.

The alternative method (Khan and Shah, 2006), on the other hand, shows a less impressive performance due to the lack of foreground pixels on the synergy map. In fact, this problem occurs if the target is partially occluded in both camera views and can be negated by increasing the number of cameras observing the scene. This becomes evident through Figure 5.11(a), (b) and (c) showing the synergy map recovered from two, three and four cameras, respectively. It is also worth stressing that, the tracking procedure of both of our algorithm and the alternative is not performed on each individual image, but, on the contrary, in a unified framework and the results are projected to each camera view. Here, the unified framework for the alternative method (Khan and Shah, 2006) is the synergy map as shown in Figure 5.11, whereas a 3D scene such as those shown in Figure 5.7 is used as the unified framework for our tracker.

We now turn our attention to the quantitative analysis on the tracking performance of each tracker under study. As per specification of the sequence, the ground plane is discretised into a 59×59 grid and the ground truth is given as the coordinates of this grid in the topview image. In Figure 5.13(a), we have plotted the target centre error as a function of frame index with respect to the ground truth provided with the “Laboratory” sequence. Note that our tracker has a significant lower target centre error of 3.18 ± 2.09 grid points, whereas the alternative (Khan and Shah, 2006), due to the poor quality of the synergy map obtained from 2 cameras, has a higher target centre



Figure 5.12: Results for the “Laboratory” sequence using 3 cameras at frames 1730, 1770, 1845 and 1975. Left: results yielded by our algorithm. Right: results yielded by the synergy map method (Khan and Shah, 2006).

error of 8.31 ± 3.69 grid points. This is consistent with the behaviours described above.

As more cameras are added to the tracking scenario, the performance of both trackers improve significantly. In particular, Figure 5.12 shows the tracking results for frames 1730, 1770, 1845 and 1975 using 3 cameras. The synergy map tracker (Khan and Shah, 2006), with better foreground map, follows the target successfully with a target centre error of 5.28 ± 2.23 grid points. The performance of our tracker is also improved, resulting in a target centre error of 2.02 ± 1.11 grid points. These errors are further reduced as we add the forth camera to the tracking framework. In particular, the target centre error for our tracker and the alternative methods are 1.88 ± 0.98 and 3.08 ± 1.77 grid points, respectively.

Furthermore, in Figure 5.13(d), we have plotted the target centre error of our method as a function of the number of parallel planes used for purposes of tracking. In particular, the tracker only fails to track the target using 5 parallel planes, results in a target centre error of 10.86 ± 6.33 grid points. Reasonable performance is obtained as the number of planes increases to 10 with a target centre error of 4.95 ± 2.48 grid points. Good and excellent performance is achieved when this number is raised to and beyond 20 parallel planes, with the target centre error records at 3.18 ± 2.09 , 3.11 ± 1.92 and 2.86 ± 1.71 grid points for 20, 30 and 40 planes, respectively. Here, the performance of the tracker does not improve linearly with the number of planes used for purposes of

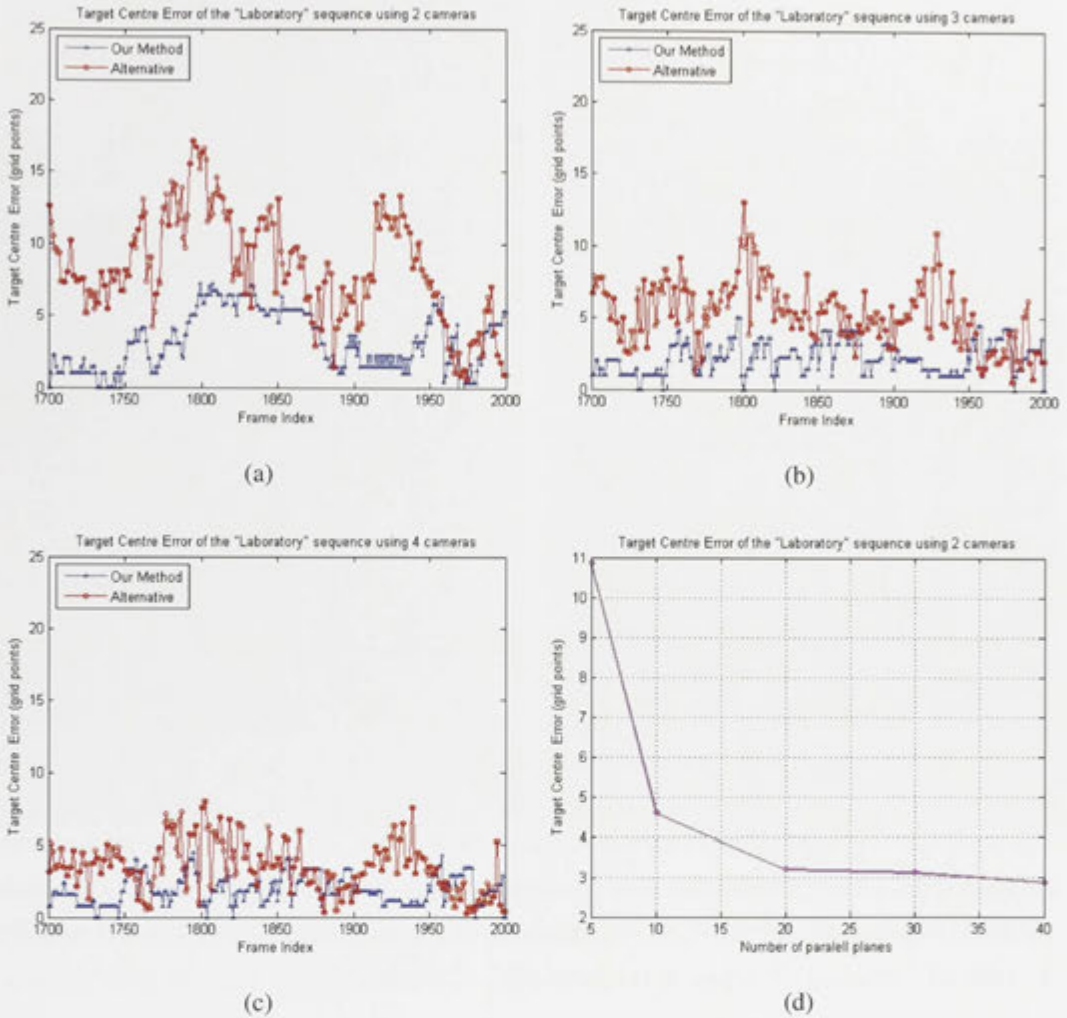


Figure 5.13: Target Centre Error of the “Laboratory” sequence: (a) using 2 cameras; (b) using 3 cameras and (c) using 4 cameras. (d) The Target Centre Error versus the number of parallel planes used in the point sampling process.

tracking, and in particular, minor improvement is obtained beyond 20 parallel planes. This result is important as there is significant overhead on the computational cost as the number of parallel planes increases. The “sweetpot” for the tracking performance, in this case, lies between 15 and 20 planes, as shown in Figure 5.13(b).

Moving on to the second vehicle of our experiment, in Figure 5.15, we present the example results for frames 820, 875, 980 and 1143 of the “Terrace sequence” using two cameras. Here, the target experiences consistent occlusion throughout the footage, or in particular, partial occlusion occurs between frames 850 to 950 and full occlusion takes place between frames 950 to 1050 and between 1100 to 1200. Here,

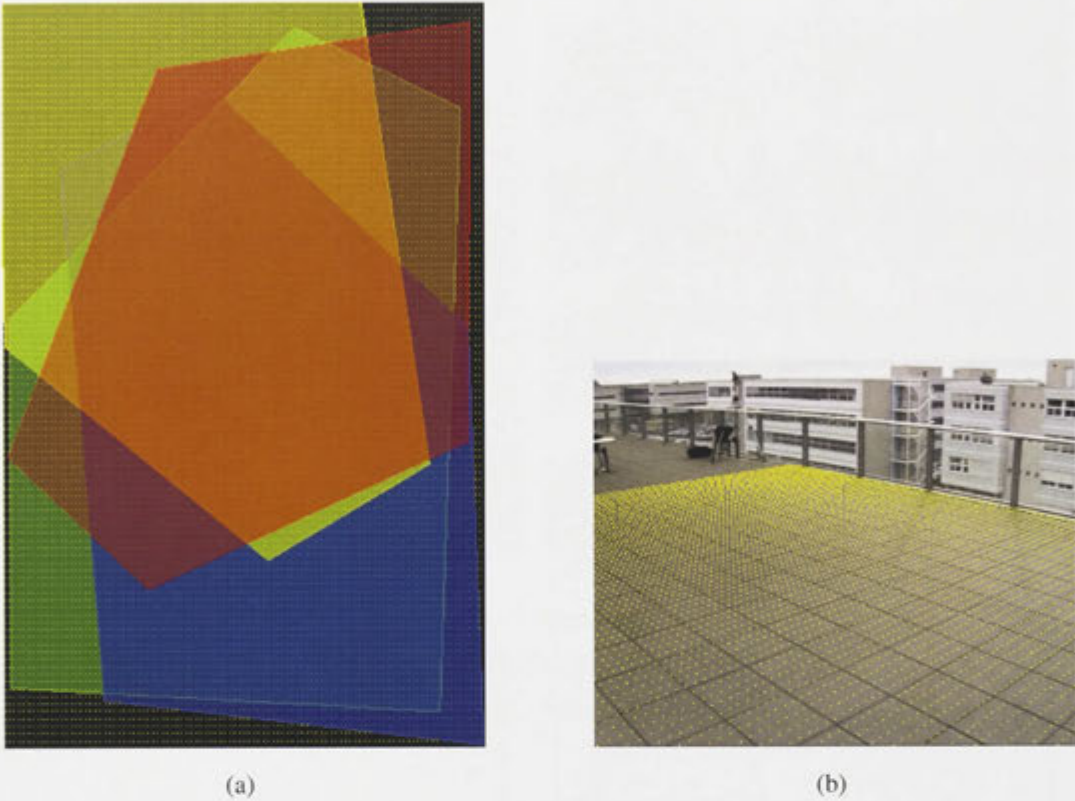


Figure 5.14: The ground plane is discretised into a 100×100 grid coordinates viewed in the topview image (a) and in Camera 1 image (b). The coloured shades show each camera coverage of the scene, or in particular, red, green, blue and yellow correspond to view of Camera 1, 2, 3 and 4, respectively.

we have used another person at the initial frame of Camera 1 as the reference object, so as to map the head of the tracking target to other cameras, prior to the multiple planar homography mapping process. For this sequence, we have used 15 planar homography planes between the foot position and the head position of the target. Additionally, as the ground plane of this scene is larger than that of the “Laboratory” sequence, we opt to quantise the reference plane into a 100×100 grid as shown in Figure 5.14. From Figure 5.15, it is clear that both trackers still manage to track the target during the first half of the sequence, in which the target is only partially occluded from Camera 1 view. However, in the second half of the sequence where full occlusion occurs, both of the trackers fail to follow the target temporarily, but are able to recover once the target is visible in both views afterward.

We now turn our attention to the numerical results plotted in Figure 5.16(a), the visual elaboration above becomes more evident. In particular, high target centre errors



Figure 5.15: Results for the “Terrace” sequence using 2 cameras at frames 820, 875, 980 and 1143. Left: results yielded by our algorithm. Right: results yielded by the synergy map method (Khan and Shah, 2006).

are recorded for both tracker in between 950 to 1050 and between 1100 to 1200 due to the fact that the target is completely occluded from both camera views. Moreover, the mean target centre errors for our tracker and the synergy map tracker (Khan and Shah, 2006) are 7.49 ± 2.79 and 6.02 ± 3.26 grid points, respectively. Here, our method records a higher target centre error due to the fact that the outfit colour of the target is similar to other moving targets in the scene, hence, the tracking window is drift every time a full occlusion occurs.

As we increase the number of camera observing the scene, the performance of both trackers is improved significantly. Figure 5.17 shows the example results from frames 830, 965, 1080 and 1145 of the “Terrace” sequence using three cameras. Here, the additional view helps mitigate the missing information of the target during the frames where full occlusion occurs. The alternative method (Khan and Shah, 2006), thanks to the better synergy map constructed using three camera views, manages to follow the

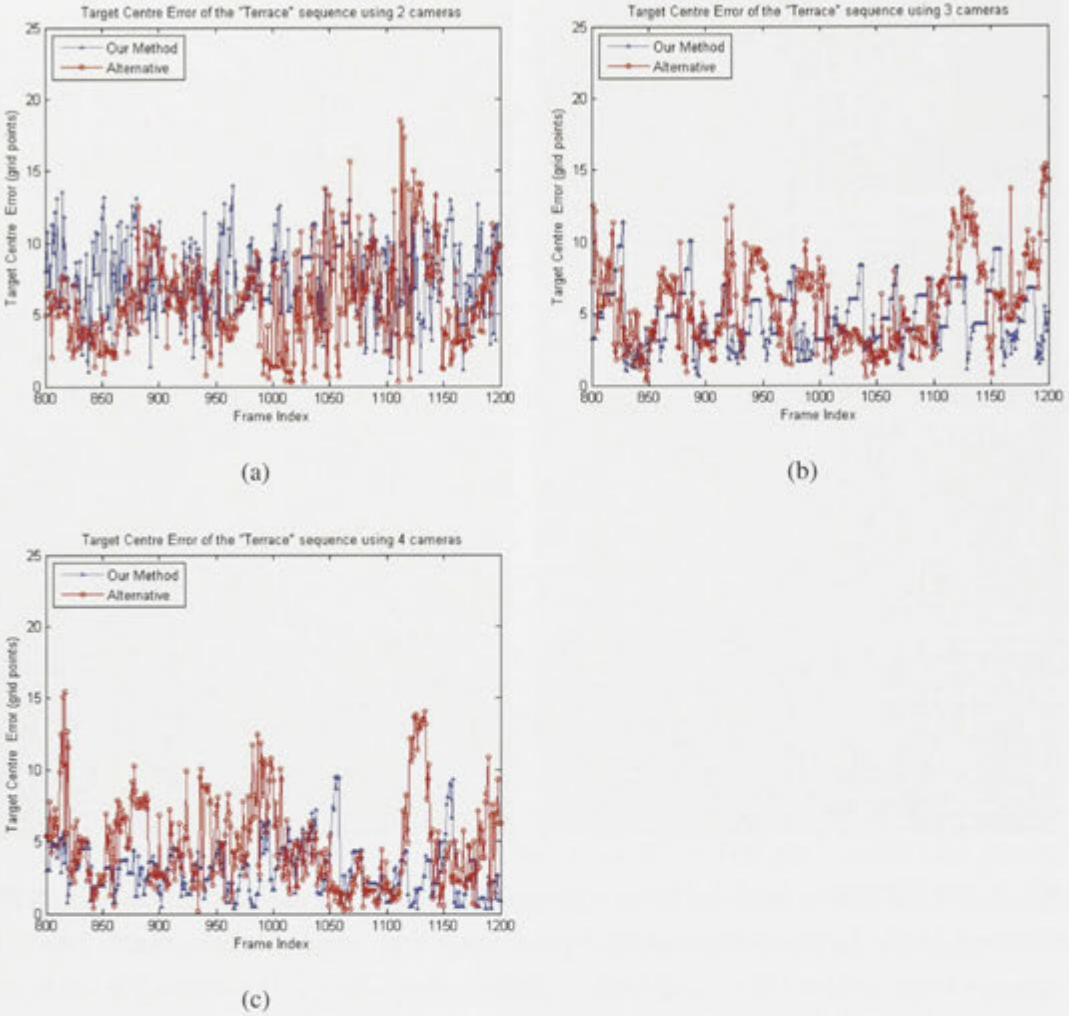


Figure 5.16: Target Centre Error of the “Terrace” sequence: (a) using 2 cameras; (b) using 3 cameras and (c) using 4 cameras.

target from the beginning of the sequence up to frame 1150, and subsequently drift further from the target centre as target approaches to other people in the scene. This effect results in a rather high target centre error of 5.57 ± 3.11 grid points even though the tracking performance is significantly improved. Our tracker, on the other hand, successfully tracks the target throughout the sequence, with a lower target centre error of 4.52 ± 2.11 grid points, as shown in Figure 5.16(b). In the case of four cameras, these figures are further improved thanks to the redundant information provided by the forth camera. In particular, Figure 5.16(c) shows the target centre error of the synergy map tracker (Khan and Shah, 2006) and the method presented in this chapter as 4.78 ± 3.20 and 3.58 ± 2.07 grid points, respectively.



Figure 5.17: Results for the “Terrace” sequence using 3 cameras at frames 830, 965, 1080 and 1145. Left: results yielded by our algorithm. Right: results yielded by the synergy map method (Khan and Shah, 2006).

Sequence name	“Laboratory”		“Terrace”	
	Our tracker	Alternative	Our tracker	Alternative
2 cameras	3.18 ± 2.09	8.31 ± 3.69	7.49 ± 2.79	6.02 ± 3.26
3 cameras	2.02 ± 1.11	5.28 ± 2.23	4.52 ± 2.11	5.57 ± 3.11
4 cameras	1.88 ± 0.98	3.08 ± 1.77	3.58 ± 2.07	4.78 ± 3.20

Table 5.1: Target Centre Error as grid-point distance for the two sequences “Laboratory” and “Terrace”.

Throughout the experiment, we have evaluated the performance of the trackers under study on two challenging sequences, in which there are a high number of moving objects in the scene and, hence, revokes constant occlusion upon the tracking target. Here, we can conclude that our approach performs reasonably better than the synergy map tracker (Khan and Shah, 2006), especially in the case of two and three cameras. The difference in performance, however, is narrowed down significantly as the number of cameras increases.

5.6 Conclusions

In this chapter, we have extended the kernel-based tracking algorithm to a 3D setting by the use of multiple cameras. We do this by firstly providing a 3D formulation for the mean-shift procedure, in which the target is represented by a set of 3D sample points in a 3D space. We subsequently develop a mapping method that allows this 3D set to be projected consistently to each participating view. To this end, we have used a reference object and the homography matrix of the ground plane in each view, to compute a set of planar homography matrices corresponding to a set of planes parallel with the plane. This set of planar homography is only required to be computed once at the beginning of the sequence, and subsequently can be used to draw sample points from each view throughout the tracking course. This treatment naturally lends itself to the recovery of the camera weights, which measure how well the target is seen in each camera. We illustrate the performance of our tracker in real-world image sequences and provide quantitative results in comparison with a benchmark algorithm. Here, our algorithm shows performance leads throughout the experiments, especially with two and three cameras. As the number of cameras increases, this performance gap is significantly narrowed down, as expected.

To conclude, it is worth noting that, the 3D extension of the kernel-based tracker presented here is devoid from a heavy-computational voxelisation approach such as those found in (Tyagi et al., 2007), while maintaining the integrity of the mean-shift procedure to open-up the possibility of a 3D implementation for the 2D methods presented in the previous chapters as well as other existing methods elsewhere in the literature.

Chapter 6

Conclusions and Future Directions

Object tracking is a classical problem in pattern recognition, computer vision and robotics. The research in this thesis has focused on the problem of object tracking in challenging environments using the mean-shift tracker (Comaniciu et al., 2003). The work presented here has extended the original method to incorporate information from multiple coordinate systems, multiple image features and multiple cameras, so as to enhance the tracking performance against poor lighting condition, severe appearance deformation and prominent occlusion. This provides the potential for using the mean-shift tracker for various applications such as advanced surveillance, traffic monitoring and sport analysis, amongst others. Here, we summarise the contributions of the work in this thesis and outline the possible improvements that can be considered for future developments.

6.1 Contributions

Statistical characterisation of the mean-shift tracker

In Chapter 3, we present a probabilistic formulation of the mean-shift tracker (Comaniciu et al., 2003). To this end, we view the coordinates for the pixels in both, the target model and its candidate as random variables and make use of a generative model so as to cast the tracking task into a Expectation Maximisation framework. Furthermore, we show that the algorithm is very general in nature and can be deployed with a variety of kernel profiles, arbitrary feature spaces or in different coordinate systems. In particular, we employ the polar coordinate system to estimate the scale and orientation deformation of the tracking target. We illustrate the robustness of our algorithm by providing quantitative results on real-world video sequences. Throughout the experiments, our tracker has shown significant improvements over a number of benchmark algorithms

including the original method (Comaniciu et al., 2003), the EM-like mean-shift tracker (Zivkovic and Krose, 2004) and the particle filter tracker (Pérez et al., 2002). The work in this chapter, hence, provides a novel approach to track the target variation in size, shape and orientation, which is a common problem in most practical scenarios.

Feature combination for object tracking

In Chapter 4, we continue the work in Chapter 3 by firstly exploiting low level image features such as the Local Binary Pattern (LBP) to cope with highly disguised objects in monochromatic and thermographic videos, so as to emphasize the importance of image features beyond colour spaces in object tracking. We then proceed to explore a new approach to combine multiple image features in the mean-shift tracker to cope with target colour variation due to abrupt changes in the scene illumination. The method presented in this chapter abstracts the low complexity image features to a relational structure and employs graph-spectral methods to combine them. This gives rise to a feature combination scheme, in which the new target location is recovered making use of a weighted combination of target-centre shifts corresponding to each of the image features under study. This treatment permits the update of the feature weights in an on-line fashion in a straightforward manner. Furthermore, we have compared the performance of our algorithm on real-world video sequences with a number of alternatives including the Variance-Ratio based tracker (Collins et al., 2005) and the PCA-based tracker (Han and Davis, 2004). To this end, our algorithm shows remarkable gain in tracking performance throughout the experiments. Here, we have illustrated the possibility of using graph-based methods commonly found in the literature of image segmentation as a means to combine multiple image features for object tracking algorithms, or more specifically, the mean-shift tracker.

3D mean-shift tracking

In Chapter 5, we explore the possibility of employing the mean-shift algorithm for tracking purposes in a multi-camera environment. To this end, we construct a 3D coordinate system making use of the homography matrices between the cameras observing the scene. The tracking task, in this case the 3D mean-shift procedure, can then be cast into a unified framework and the results can be projected back to each camera view with high consistency. We subsequently use the vanishing points of the 3D scene to establish a consistent set of sample points across the cameras which, in turn, are used to construct the M -bin histograms for the 3D mean-shift procedure. This treatment naturally lends itself to the recovery of the camera weights, which measure how well the target is seen in each camera. Furthermore, we illustrate the robustness of our

algorithm on real-world multi-camera sequences, with quantitative comparison to the synergy map tracker (Khan and Shah, 2006). To this end, our algorithm shows performance lead throughout the experiments, especially with two and three cameras. As the number of cameras increase, the performance gap is narrowed down proportionally, as expected. The objective of this chapter, however, is to show how the kernel-based tracker can be deployed in a 3D setting, in which the set of sample points can be obtained via multiple planar homography matrices. This treatment is devoid from a heavy-computational 3D voxelisation process, while maintaining the integrity of the mean-shift procedure to open-up the possibility of a 3D implementation for the 2D methods presented in the previous chapters as well as other existing methods elsewhere in the literature.

6.2 Future Directions

This section concludes the thesis with a brief discussion on the possible departure points for the future research following the work presented here.

Firstly, the mean-shift tracker proposed in Chapter 3 employs primitive geometry shapes such as a rectangle or an elliptical region to define a tracking target. While this approach proves to be sufficient for most rigid objects, it can potentially include a substantial amount of background noises for non-rigid objects such as human and animal objects and subsequently, affects the tracking performance. To this end, it is desirable to develop a more flexible polygon shape to enclose the target effectively. The vertices of the polygon can be optimised to fit the target using the Expectation-Maximisation framework presented in the chapter. Furthermore, potential future work can include the implementation for multiple objects tracking, in which the targets are initialised by an object detection algorithm and the tracking task is carried out simultaneously for all targets using parallel computing techniques.

Secondly, the graph-based feature combination method presented in Chapter 4 can be further enhanced by integrating the scale and orientation estimation algorithm from Chapter 3. This treatment, in turn, can provide a robust algorithm for single-view object tracking, which can cope with the existing problems of severe deformation and colour variation in many practical scenarios. Additionally, the on-line feature weight updating scheme presented in Section 4.4 currently relies on an empirically selected probability density function (PDF), i.e. $P(\hat{\mathbf{Q}}_{\phi_i} | \mathcal{Q}_{\phi_i}) \sim N(\mu_0, \sigma_0)$. Possible future work, therefore, can include the development of an automatic estimation method for the values μ_0 and σ_0 , or more generally, the PDF used for the histogram blending

operation.

Finally, the methods and their improvement suggestions presented above can be extended to a 3D implementation using the 3D framework proposed in Chapter 5. Amongst those, the scale and orientation estimation step can be bypassed with the assumption that the target size remains relatively constant in the 3D scene during the course of tracking. Furthermore, the task of selecting the reference object at the initial frame can be further improved by the use of a virtual reference object. This can be obtained by firstly estimating the projection matrix of each observing camera and, subsequently, erecting a virtual object on the reference of the 3D scene. Last but not least, future work on this chapter could be directed to a methodology to select the optimum number of planar planes to characterise a target based on its relative positions to the cameras observing the scene.

Appendix A

List of Publications

A number of publications have directly resulted from this work. These are

1. Q. A. Nguyen, A. Robles-Kelly and C. Shen. Enhanced Kernel-Based Tracking for Monochromatic and Thermographic Video. In *IEEE International Conference on Video and Signal Based Surveillance*, page 28, 2006.
2. Q. A. Nguyen, A. Robles-Kelly and C. Shen. Kernel-based Tracking from a Probabilistic Viewpoint. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
3. Q. A. Nguyen, A. Robles-Kelly and J. Zhou. A Graph-based Feature Combination Approach to Object Tracking. In *Asian Conference on Computer Vision*, pages 224–235, 2009.

Bibliography

- A. Ali and J. K. Aggarwal. Segmentation and recognition of continuous human activity. *IEEE Workshop on Detection and Recognition of Events in Video*, pages 28–35, 2001.
- S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.
- S. Avidan. Ensemble tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 494–501, 2005.
- Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc., 1987.
- J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.
- J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.
- D. Beymer and K. Konolige. Real-time tracking of multiple people using continuous detection. In *IEEE International Conference on Computer Vision Frame-Rate Workshop*, 1999.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
- M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.
- I. Borg and P. Groenen. *Modern Multidimensional Scaling, Theory and Applications*. Springer Series in Statistics. Springer-Verlag, 1997.

- J. S. Bridle. Training stochastic model recognition algorithms can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems 2*, pages 211–217, 1990.
- T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1): 90–99, 1986.
- M. Brown and D. G. Lowe. Recognising panoramas. In *IEEE International Conference on Computer Vision*, volume 2, page 1218, 2003.
- Q. Cai and J. K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241–1247, 1999.
- S. Calderara, R. Vezzani, A. Prati, and R. Cucchiara. Entry edge of field of view for multi-camera tracking in distributed video surveillance. *Advanced Video and Signal Based Surveillance*, pages 93–98, 2005.
- J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- T. Chang and S. Gong. Tracking multiple people with a multi-camera system. In *Proceedings of the IEEE Workshop on Multi-Object Tracking*, pages 19–26, 2001.
- I. Chavel. *Riemannian Geometry: A Modern Introduction*. Cambridge University Press, 1995.
- Y.-T. Chen, C.-S. Chen, and Y.-P. Hung. Integration of background modeling and object tracking. In *IEEE International Conference on Multimedia and Expo*, pages 757–760, 2006.
- Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.
- W.-P. Choi and K.-M. Lam. An effective shape-texture weighted algorithm for multi-view face tracking in videos. In *Congress on Image and Signal Processing*, volume 4, pages 156–160, 2008.
- F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

- R. Collins, Y. Liu, and M. Leordeanu. On-line selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10): 1631 – 1643, 2005.
- R. T. Collins. Mean-shift blob tracking through scale space. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 234–240, 2003.
- D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.
- D. Cremers and C. Schnorr. Statistical shape knowledge in variational motion segmentation. *Image and Vision Computing*, 21:77–86, 2003.
- A. Criminisi, I. D. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.
- F. C. Crow. Summed-area tables for texture mapping. In *Annual Conference on Computer Graphics and Interactive Techniques*, pages 207–212, 1984.
- A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of The Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- S. Dockstader and A. Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE*, 89(10):1441–1455, 2001.
- G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *International Conference on Face and Gesture Recognition*, pages 300–305, 1998.
- A. Elgammal, R. Duraiswami, D. Harwood, L. S. Davis, R. Duraiswami, and D. Harwood. Background and foreground modeling using nonparametric kernel density for visual surveillance. In *Proceedings of the IEEE*, pages 1151–1163, 2002.
- A. Elgammal, R. Duraiswami, and L. S. Davis. Probabilistic tracking in joint feature-spatial spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 781–788, 2003.
- Z. Fan, Y. Wu, and M. Yang. Multiple collaborative kernel tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 502–509, 2005.

- P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–27, 1997.
- M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740, 1987.
- Y. Freund and R. E. Schapire. A short introduction to boosting. In *International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999.
- Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997.
- I. Gordon and D. G. Lowe. What and where: 3d object recognition with accurate pose. In *Toward Category-Level Object Recognition*, 2006.
- N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.
- H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.
- G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 790–797, 2004.
- B. Han and L. Davis. Object tracking by adaptive feature extraction. In *International Conference on Image Processing*, volume 3, pages 1501–1504, 2004.
- I. Haritaoglu, D. Harwood, and L. S. Davis. Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22: 809–830, 2000.
- R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.
- M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(28):657–662, 2006.

- E. E. Hemayed. A survey of camera self-calibration. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, page 351, 2003.
- T. Hofmann and M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.
- B. K. P. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 17: 185–204, 1981.
- M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, volume 1, pages 343–356, 1996.
- M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal on Computer Vision*, 29(1):5–28, 1998.
- R. Jain and K. Wakimoto. Multiple perspective interactive video. In *IEEE International Conference Multimedia Computing and Systems*, 1995.
- T. Jebara. Multi-task feature and kernel selection for SVMs. In *International Conference on Machine Learning*, page 55, 2004.
- A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1296–1311, 2003.
- G. K. K. Bowyer and S. Dougherty. Edge detector evaluation using empirical roc curve. In *Computer Vision Image Understand*, volume 10, pages 77–103, 2001.
- J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 267–272, 2003.
- P. H. Kelly, A. Katkere, D. Y. Kuramura, S. Moezzi, and S. Chatterjee. An architecture for multiple perspective interactive video. In *ACM International Conference on Multimedia*, pages 201–212, 1995.
- S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1355–1360, 2003.

- S. M. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision*, volume 4, pages 133–146, 2006.
- S. M. Khan, P. Yan, and M. Shah. A homographic framework for the fusion of multi-view silhouettes. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.
- P. Li and H. Wang. Probabilistic object tracking based on machine learning and importance sampling. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 161–167, 2005.
- S. Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag, 2001.
- K.-C. Lien and C.-L. Huang. Multiview-based cooperative tracking of multiple human objects. *Journal on Image and Video Processing*, 2008:1–13, 2008.
- D. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.
- D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- W.-L. Lu, K. Okuma, and J. J. Little. Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1-2):189–205, 2009.
- B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
- G. F. McLean and D. Kotturi. Vanishing point detection by line clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1090–1095, 1995.
- Microsoft Corporation. Project NATAL. www.xbox.com/en-US/live/projectnatal/, 2010.
- H. T. Nguyen and A. W. M. Smeulders. Tracking aspects of the foreground against the background. In *European Conference on Computer Vision*, volume 2, pages 446–456, 2004.

- K. Nickel, T. Gehrig, R. Stiefelhagen, and J. McDonough. A joint particle filter for audio-visual speaker tracking. In *International Conference on Multimodal interfaces*, pages 61–68, 2005.
- T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on featured distribution. *Pattern Recognition*, 1(29):51–59, 1996.
- J. Orwell, S. Massey, P. Remagnino, D. Greenhill, and G. A. Jones. A multi-agent framework for visual surveillance. In *International Conference on Image Analysis and Processing*, page 1104, 1999.
- N. Paragios and R. Deriche. Geodesic active regions and level set methods for motion estimation and tracking. *Computer Vision and Image Understanding*, 97(3):259–282, 2005.
- S. Park. A hierarchical bayesian network for event recognition of human actions and interactions. *Association For Computing Machinery Multimedia Systems Journal*, pages 164–179, 2004.
- G. Paschos. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. In *IEEE Transactions on Image Processing*, volume 10, pages 932–937, 2001.
- P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, volume 2350, pages 661–675, 2002.
- P. Perona and W. T. Freeman. Factorization approach to grouping. In *European Conference on Computer Vision*, volume 1, pages 655–670, 1998.
- A. Robles-Kelly. Segmentation via graph-spectral methods and riemannian geometry. In *International Conference on Computer Analysis of Images and Patterns*, pages 661–668, 2005.
- Y. Rodriguez and S. Marcel. Face authentication using adapted local binary pattern histograms. In *European Conference on Computer Vision*, pages 321–332, 2006.
- R. Rosales and S. Sclaroff. 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–123, 1999.

- S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71(1):110–136, 1998.
- K. Sato, T. Maeda, H. Kato, and S. Inokuchi. Cad-based object tracking with distributed monocular camera for security monitoring. In *IEEE CAD-based Vision Workshop*, pages 291–297, 1994.
- F. Schaffalitzky and A. Zisserman. Planar grouping for automatic detection of vanishing lines and points. *Image and Vision Computing*, 18:647–658, 2000.
- G. L. Scott and H. C. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proximity matrix. In *British Machine Vision Conference*, pages 103–108, 1990.
- I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, 1987.
- K. Shafique and M. Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, 2005.
- J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *Talk presented at the Stanford Artificial Project*, 1968.
- C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2246–2252, 1999.
- H. Stern and B. Efron. Adaptive color space switching for face tracking in multi-colored lighting environments. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 249, 2002.
- R. Szeliski, R. Szeliski, J. Coughlan, and J. Coughlan. Spline-based image registration. In *International Journal of Computer Vision*, pages 194–201, 1994.

- A. Tyagi, G. Potamianos, J. W. Davis, and S. M. Chu. Fusion of multiple camera views for kernel-based 3d tracking. In *IEEE Workshop on Motion and Video Computing*, page 1, 2007.
- A. Utsumi and N. Tetsutani. Human tracking using multiple-camera-based head appearance modeling. *Asian Conference on Computer Vision*, pages 657 – 662, 2004.
- A. Utsumi, H. Yang, and J. Ohya. Adaptive human motion tracking using non-synchronous multiple viewpoint observations. *Asian Conference on Computer Vision*, pages 1034–1039, 2000.
- C. J. Veenman, M. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:54–72, 2001.
- P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.
- P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.
- C.-X. Wang and Z.-Y. Li. A new face tracking algorithm based on local binary pattern and skin color information. In *International Symposium on Computer Science and Computational Technology*, volume 2, pages 657–660, 2008.
- Y.-D. Wang, J.-K. Wu, and A. A. Kassim. Adaptive particle filter for data fusion of multiple cameras. *The Journal of VLSI Signal Processing*, 49(3):363–376, 2007.
- C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *IEEE International Conference on Computer Vision*, pages 212–219, 2005.
- A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38(4):13, 2006.
- Z. Yin and R. T. Collins. Object tracking and detection after occlusion via numerical hybrid local and global mode-seeking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

- H. Zhang, W. Huang, Z. Huang, and L. Li. Affine object tracking with kernel-based spatial-color representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 293–300, 2005.
- S. C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:884–900, 1996.
- Z. Zivkovic and F. V. D. Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.
- Z. Zivkovic and B. Krose. An EM-like algorithm for color-histogram-based object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 798–803, 2004.