Trajectory based video analysis in multi-camera setups

Anjum, Nadeem

Trajectory based video analysis in multi-camera setups

A thesis presented to the University of London
by

Nadeem Anjum

for the degree of
Doctor of Philosophy
in

Electronic Engineering

Queen Mary University of London
Mile End Road
E1 4NS, London, UK
September 2010

I confirm that the work presented in this thesis is my own and the work of other persons is appropriately acknowledged.

Sincerely yours,

Nadeem Anjum

Thesis supervisor Author

Andrea Cavallaro **Nadeem Anjum**

Trajectory based video analysis in multi-camera setups

Abstract

This thesis presents an automated framework for activity analysis in multi-camera setups. We start with the calibration of cameras particularly without overlapping views. An algorithm is presented that exploits trajectory observations in each view and works iteratively on camera pairs. First outliers are identified and removed from observations of each camera. Next, spatio-temporal information derived from the available trajectory is used to estimate unobserved trajectory segments in areas uncovered by the cameras. The unobserved trajectory estimates are used to estimate the relative position of each camera pair, whereas the exit-entrance direction of each object is used to estimate their relative orientation. The process continues and iteratively approximates the configuration of all cameras with respect to each other. Finally, we refine the initial configuration estimates with bundle adjustment, based on the observed and estimated trajectory segments. For cameras with overlapping views, state-of-the-art homography based approaches are used for calibration.

Next we establish object correspondence across multiple views. Our algorithm consists of three steps, namely association, fusion and linkage. For association, local trajectory pairs corresponding to the same physical object are estimated using multiple spatio-temporal features on a common ground plane. To disambiguate spurious associations, we employ a hybrid approach that utilises the matching results on the image plane and ground plane. The trajectory segments after association are fused by adaptive averaging. Trajectory linkage then integrates segments and generates a single trajectory of an object across the entire observed area.

Finally, for activities analysis clustering is applied on complete trajectories. Our clustering algorithm is based on four main steps, namely the extraction of a set of representative trajectory features, non-parametric clustering, cluster merging and information fusion for the identification of normal and rare object motion patterns. First we transform the trajectories into a set of feature spaces on which Mean-shift identifies the modes and the corresponding clusters. Furthermore, a merging procedure is devised to refine these results by combining similar adjacent clusters. The final common patterns are estimated by fusing the clustering results across all feature spaces. Clusters corresponding to reoccurring trajectories are considered as normal, whereas sparse trajectories are associated to abnormal and rare events.

The performance of the proposed framework is evaluated on standard data-sets and compared with state-of-the-art techniques. Experimental results show that the proposed framework outperforms state-of-the-art algorithms both in terms of accuracy and robustness.

Table of Contents

Previously published work

Journal papers

[J1] N. Anjum and A. Cavallaro. Automated localization of a camera network. *IEEE Intelligent Systems*, 2010, to appear.

[J2] N. Anjum and A. Cavallaro. Multi-feature object trajectory clustering for video analysis. *IEEE Trans. on Circuits and Systems for Video Technology*, 18(11):1555-1564, Nov. 2008.

Book chapters

[B1] N. Anjum and A. Cavallaro. Trajectory clustering for scene context learning and outlier detection. *Springer-Verlag series on Studies in Computational Intelligence*, 2010.

[B2] N. Anjum and A. Cavallaro. Multi-camera calibration and global trajectory fusion. *Intelligent Video Surveillance Systems and Technology*, 2010.

Conference papers

[C1] N. Anjum and A. Cavallaro. Localization of distributed wireless cameras. *in Proc. of IEEE Intl. Conf. on Distributed Smart Cameras* , Como (Italy), Aug., 2009.

[C2] N. Anjum and A. Cavallaro. Trajectory association and fusion across partially overlapping cameras. *in Proc. of IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance*, Genoa (Italy), Sep., 2009.

[C3] G. Kayumbi, N. Anjum and A. Cavallaro. Global trajectory reconstruction from distributed visual sensors. *in Proc of IEEE Intl. Conf. on Distributed Smart Cameras*, Stanford, California (USA), Sep., 2008.

[C4] N. Anjum and A. Cavallaro. Unsupervised fuzzy clustering for trajectory analysis. *in Proc of IEEE Intl. Conf. on Image Processing*, San Antonio, Texas (USA), Sep., 2007.

[C5] N. Anjum and A. Cavallaro. Single camera calibration for trajectory-based behavior analysis. *in Proc of IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance*, London (UK), Sep., 2007.

[C6] N. Anjum and A. Cavallaro. Relative position estimation of non-overlapping cameras. *in Proc of IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Honolulu (USA), Apr., 2007.

Electronic preprints are available on the Internet at the following URL:

`http://www.elec.qmul.ac.uk/staffinfo/andrea/publications.html`

Glossary

C	camera network
c_k	k^{th} camera in a camera network
$\mathcal{H}$	homography matrix
$\mathcal{S}$	similarity transformation
ς	affine transformation
ρ	pure projective transformation
l_∞	vanishing line of a plane
Z	complete set of measurements
$z^m{=}(x^m, y^m)$	measurement at instance m
$\tilde{Z}$	estimated set of measurements before filtering
$(\tilde{x}^m_{v,f/b}, \tilde{y}^m_{v,f/b})$	velocity of an object in forward/backward direction
k^m	Kalman gain
$\hat{Z}$	estimated set of measurements after filtering
$p_i = (x_i, y_i)$	position of a camera c_i
r_i	rotation of a camera c_i
$D_{i,j}$	displacement vector from c_i to c_j

$\mathcal{L}$	set of localisation parameters
$\mathcal{L}^*$	refined set of localisation parameters
$T_{i,n}$	trajectory segment of i^{th} object in c_n
$\overline{v}$	average velocity
$\hbar$	trajectory directional histogram
$\mathbf{m}$	trajectory mean
$\mathbf{p}$	PCA components
$\mathbf{d}$	directional distance
$T_{i,G}$	trajectory segment of i^{th} object on ground plane (G)
$\hat{T}_{i,j,G}$	fused trajectory of i^{th} and j^{th} objects on ground plane
$\Omega_{n,m}$	overlapping region between cameras c_m and c_n
A_Ω	association matrix in overlapping region (Ω)
$F_m(.)$	feature extraction function
$\Psi_m^{d_m}$	d_m-dimensional feature space representation
Ξ_j	covariance of j^{th} trajectory
$\hat{f}(.)$	multivariate density estimator
$\nabla \hat{f}(x)$	multivariate density gradient estimator
$h_m^{d_m}$	bandwidth of Mean-shift kernel
$\mathcal{K}$	Mean-shift kernel
ϵ_t	translation error
ϵ_r	rotation error
P_a	precision measure for trajectory association
R_a	recall measure for trajectory association
P_c	precision measure for trajectory clustering
R_c	recall measure for trajectory clustering

Chapter 1

Introduction

1.1 Motivation

Camera networks help in monitoring large scale areas for applications such as traffic-flow analysis on a highway, monitoring of sensitive buildings or patrolling national borders and content production for sporting events. These camera networks commonly provide live video streams to a common location (or control room) where all of the video data is normally analysed manually. The nature of this work (monitoring multiple simultaneous videos), however, can be monotonous for humans, since it is mostly uneventful. This boredom can make the observer prone to error. It is therefore desirable to automate this process using an intelligent system. The system should monitor the video feeds and use the information from multiple streams to summarise results in a compact form. For example, this summary could contain information about common motion patterns and instances of abnormal activities. To develop such a system, typical operational steps include: *camera calibration*, *object correspondence* across multiple views to reconstruct complete trajectories and *scene understanding* by identifying normal and abnormal activities [1]. A block diagram

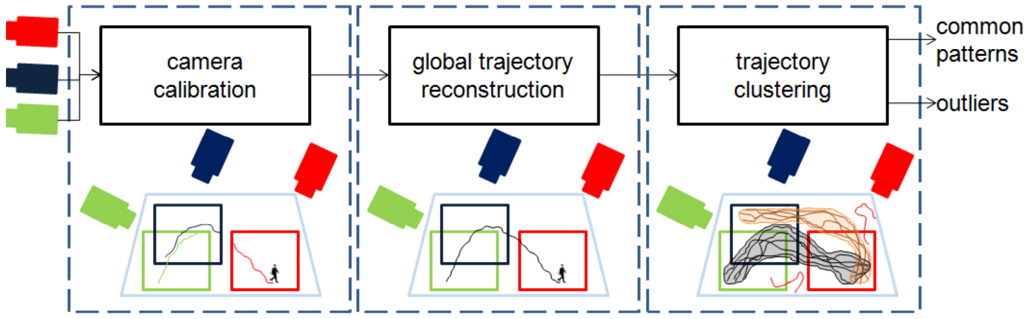


Figure 1.1: Block diagram of my video analysis system.

of a video analysis system is shown in Fig. 1.1.

The development of video based activity analysis system is a challenging task and it may be encountered with several issues related to calibration and tracking. For example, it is common that cameras monitoring scenes are installed before calibration is undertaken. If the field of view of a camera is too limited; it can affect the reliability of calibration based on parallel lines. Also, with the increase of cameras, manual and GPS based solutions become impractical especially when cameras are installed in underground e.g., subways and tube stations [2]. Furthermore, tracking an object in a network of cameras is also a challenging task due to change in appearance and view, non-rigid structures and occlusions [3, 4]. This can result in trajectory breaks and identity switches; thus makes the activity analysis more complex.

This thesis addresses the following important research questions for activity analysis in a multi-camera setup:

- Can we learn the configuration of a camera network using object trajectories?
- Can we establish object's correspondence across multiple views using its trajectory?
- Can we distinguish normal and abnormal activities in a scene using accumu-

lated raw trajectories?

1.2 Main contributions

The main contributions of this thesis are as follows:

1. Our first contribution is to propose an automated localisation or extrinsic calibration algorithm that does not require any prior calibration information. The algorithm uses trajectories to estimate the configuration of a camera network and is applicable also when cameras have non-overlapping fields of view [J1]. The algorithm initially removes outliers using RANSAC from the observations of each camera view. Next, unlike other approaches [5, 6, 7], it iteratively uses spatio-temporal information derived from the available trajectory information to estimate the unobserved trajectory segment that is then used to position the cameras on a common plane. The exit-entrance direction of the object is used to estimate the relative orientation of the cameras. After this initial estimation, the configuration results are refined using bundle adjustment. Major novelties of this algorithm include: (a) the relaxation on the linearity constraint on the object motion in unobserved regions (the object can take a sharp turn); (b) no prior information about the environment or the motion model is required; and (c) the initial configuration can be refined when batch processing is viable.

2. Our second contribution is to establish target's correspondence across multiple cameras using its trajectory [B2,C2]. Existing works perform association either on image plane [8] or on ground plane [9]. As image plane trajectories are heavily affected by the perspective deformations, which cause inaccurate

associations especially if the trajectories are far from cameras. On the other hand, accurate associations on ground plane are hampered by the image to ground plane projections, which do not ensure unique association of an object trajectories observed in multiple cameras. The major novelty of this algorithm is that we use hybrid approach that combines the strength of both image and ground plane associations. Initial correspondence among trajectories is established on ground plane using multiple spatio-temporal features and then image plane reprojections of the matched trajectories are employed to resolve conflicting situations. This makes sure that only one trajectory of an object from each camera is associated to other cameras. Trajectory segments belonging to same object are then fused and linked to reconstruct a complete trajectory.

3. Our final contribution is to propose an algorithm that can analyse object trajectories and can identify normal and abnormal activities [J2,B1]. In general trajectory analysis algorithms use only one feature space for clustering [10, 11, 12, 13, 14, 15]. Even when more features are used, they are not processed simultaneously [16, 17]. This can result in a coarse cost function defined by the proximity measure due to reduced utilisation of complementary information, thus leading to a local minima problem. One way to overcome this problem is to use a stochastic optimisation algorithm. However, the convergence properties (such as the radius of convergence) of such algorithms are limited [18]. The major novelty of this algorithm is that we use multiple feature spaces simultaneously to obtain a higher degree of descriptiveness of the trajectories as opposed to using one feature space only. We propose a partitional trajectory clustering framework that combines internally a fuzzy clustering approach

based on multiple features before generating a final crisp partition. Each feature space is then regarded as the empirical probability density function (*pdf*) of the represented parameter and modes in each space correspond to the maxima of the *pdf*. Once the modes are determined, the members can be associated to each mode to form the clusters. We use Mean-shift in each feature space for mode-seeking and clustering. Mean-shift is a non-parametric clustering algorithm, which always converges to local mode of the data. The clustering results of Mean-shift in each space are then refined by applying a cluster merging procedure. The final clustering is obtained by analysing the clustering results from each feature space. The process results in the definition of the clusters' structures along with the fuzzy membership of a trajectory to the final clusters. The clusters with small number of associated elements and the trajectories that are far from the clusters' centre are considered as outliers.

1.3 Outline of the thesis

Chapter 2 discusses existing video analysis algorithms. The chapter discusses techniques for view rectification and calibration of multi-camera networks with overlapping and non-overlapping views. Also, it explains various trajectory based object association methods. Furthermore, significant works for trajectory clustering and outliers detection are also covered in the chapter. In **Chapter 3**, we motivate and describe our approach for view rectification and camera localisation. **Chapter 4**, explains our method to reconstruct complete objects' trajectories across multiple views. **Chapter 5** provides details of our trajectory clustering algorithm. It also explains the process of combining clustering results from various feature spaces. Furthermore, our anomaly detection method is also discussed in the chapter. **Chapter 6** presents the evaluation of the proposed approaches on synthetic and real video sequences. Finally, in **Chapter 7**, we summarise the achievements of this thesis and we discuss possible extensions of this work.

Chapter 2

Prior works

This chapter provides details of prior works for (i) calibration of single as well as multiple cameras, (ii) object association across multiple views and (iii) trajectory clustering for scene analysis. The details of these approaches are provided in the following sections.

2.1 Camera calibration

In this section, we primarily focus on approaches that learn the intrinsic calibration parameters (also known as self-calibration) of individual cameras. We further extend this discussion to techniques that estimates extrinsic calibration parameters (also known as localisation) of multiple cameras; especially, when the cameras have non-overlapping views.

2.1.1 Self-calibration

It is well known that due to perspective projection the measurements made from the images do not represent metric data. Thus, the obtained object trajectories are projectively distorted, unless we have a calibrated camera. For example, a person moving slowly but close to a camera induces large image motion compared to person walking at a distance with a quicker pace. Therefore, it is desirable to have self-calibrated cameras, where information within cameras' field of views can only be used for calibration.

We classify self-calibration techniques into two classes: (a) *object*-based, and (b) *scene-structure*-based. The initial object-based self-calibration method was presented by Faugeras *et al.* [19]. The method showed that self-calibration was theoretically and practically feasible, if a camera, with *fixed* intrinsic parameters, can take images of an object from multiple views. The approach employed the Kruppa equations [20]. Based on these equations, Mendonca [21] introduced a built-in method for the detection of "critical motions" for each pair of images in the sequence. The approach has a linear step, where only the focal length is computed followed by a non-linear optimisation that refines the estimates obtained in the linear step and allows for the estimation of more intrinsic parameters such as principal point. A related work was presented by Triggs [22], which estimates the absolute dual quadric instead of conic over many views taken by a single camera that is moving with constant velocity. Pollefeys *et al.* [23] developed a practical method, which recovers metric reconstruction from image sequence where the intrinsic parameters of a camera may vary. Furthermore, Agapito *et al.* [24], and Seo and Hong [25] solved the self-calibration of a rotating and zooming camera using the infinite homography constraint.

Scene-structure-based self-calibration approaches primarily assume that metric properties (such as a length ratio and an angle) derived from available scene structures are enough to determine the projective transformation up to a scale [26, 27, 28].

Table 2.1: State-of-the-art methods for localisation of a non-overlapping camera network.

Est. parameters	Constraints	Ref.
Position only	Linear object motion	[5]
	Known environment map	[6, 31]
Rotation only	Linear object motion+vanishing line	[32]
	Far objects with known trajectory	[33]
Position and rotation	Only one smooth turn (in unobserved region)	[7]
	Only one sharp turn (in unobserved region)	[34]

Once the vanishing line of the plane is identified the transformation from world to image plane can be reduced to affinity [29]. Furthermore, with the help of metric properties, the affinity can be reduced to similarity [30].

The information of intrinsic calibration parameters of individual cameras is helpful in extracting the extrinsic calibration parameters of a camera network, especially with non-overlapping views, where there are no common control points between different views. The details of earlier works are provided in the next section.

2.1.2 Extrinsic calibration of non-overlapping cameras

Existing extrinsic calibration (also known as localisation) approaches for non-overlapping cameras primarily assume that objects (e.g., pedestrians or vehicles) move linearly in unobserved regions [5]. Under this assumption, the relative position of two cameras can be estimated by extrapolating the trajectory in unobserved regions using the velocity information learnt from the last few observed instances. Javed *et al.* [5] combine linear velocity with a change in brightness function to learn the relative positions of the cameras.

To estimate the relative orientation of two cameras, either vanishing points [32] or far objects with known trajectories (e.g., stars) can be used [33]. Junejo *et al.* demonstrate that the vertical vanishing point and the knowledge of a line in a plane orthogonal to the vertical direction are sufficient. There exist various approaches that learn the topology of non-overlapping camera networks using graph representation [6]. In these approaches, camera nodes that are constituted by the exit and entrance locations are considered as vertices and their spatial neighbourhood as edges of a graph. Makris *et al.* [6] estimate camera topology from observations by assuming a Gaussian transition distribution. Departures and arrivals within a chosen time window are assumed to be corresponding. Tieu *et al.* [31] generalised the work in [6] to multimodal transition distributions, and handled correspondences explicitly. In their work, camera connectivity is formulated in terms of statistical dependence, and uncertain correspondences are removed in a Bayesian manner. The performances of these approaches degrade substantially when the dynamics of the camera network is complex and also when the object motion deviates considerably from the pre-modelled path in unobserved regions e.g., monitoring a difficult terrain such as hilly area [35].

Other approaches track an object while simultaneously estimating the network localisation parameters [7]. A Bayesian framework can be used to find unknown parameters (localisation and trajectory), given the observations from each camera. Maximum a posteriori (MAP) is estimated using the Newton-Raphson method that makes the method computationally expensive with the increasing of the number of cameras and the length of the trajectory. Furthermore, the performance degrades substantially with noisy observations. In order to address these problems, it is desirable to first estimate trajectory segments in unobserved regions and then to learn the localisation parameters based on both the observed and the estimated trajectory segments [34]. A summary of approaches is provided in Table 2.1.

An important application where camera calibration can play a vital role is the

Table 2.2: State-of-the-art methods for object correspondence across multiple views.

Category	Constraints	Application	Ref.
Supervised	Fixed motion paths	Indoor people tracking	[36] [37]
		Vehicle tracking on a highway	[38]
	Threshold on transition time	Vehicle tracking	[39]
Unsupervised	Airborne cameras	Vehicle tracking in a parking lot	[8]
	-	Sport's analysis	[9]

generation of a global (or complete) trajectory from a network of cameras. In the next section, we discuss some of the important methods regarding reconstruction of complete trajectories.

2.2 Global trajectory reconstruction

An important step for reconstructing complete trajectories is to establish object correspondence across multiple views. A summary of existing approaches is provided in Table 2.2. We categorise object correspondence approaches into *supervised* and *unsupervised* algorithms. Supervised techniques depend upon the information either contained in training samples or supplied manually by users. Several authors have proposed supervised association approaches such as Kettnaker *et al.* [36], Huang *et al.* [38], Dick *et al.* [37] and Wang *et al.* [39]. Unlike supervised techniques, unsupervised techniques do not require training samples or manual selection of the parameters. Recent unsupervised target association algorithms are presented by Kayumbi *et al.* [9] and Sheikh *et al.* [8]. The rest of this section provides the details of these techniques.

Kettnaker *et al.* [36] presented a Bayesian solution to the tracking of people across multiple cameras. The system requires prior information about the environment and the way people move across it. Huang *et al.* [38] presented a probabilistic approach for tracking cars across two cameras on a highway, where transition times were modelled as Gaussian distributions. Like Kettnaker *et al.*, it was assumed that the initial transition probabilities were known. This approach is application-specific, using only two calibrated cameras with vehicles moving in one direction in a single lane. Dick *et al.* [37] use a stochastic transition matrix to describe patterns of motion for both intra- and inter-camera correspondence. The correspondence between cameras has to be supplied as training data. Wang *et al.* [39] connect trajectories observed in multiple cameras based on their temporal information. The trajectories are considered to be corresponding, if they overlap in time for an empirically pre-selected interval.

Kayumbi *et al.* [9] establish correspondence between cameras and a virtual ground plane. Trajectory association is performed on the ground plane using shape and length along with temporal information. The maximum likelihood for association is calculated by cross correlation of spatio-temporal feature vectors. However, this approach cannot differentiate two objects moving with varying speed in the environment. Another approach in this category is presented by Sheikh *et al.* [8]. In their approach, airborne cameras are used with the assumption of the simultaneous visibility of at least one object by two cameras. Taking as input time-stamped trajectories from each view, the algorithm estimates the inter-camera transformations. The maximum likelihood is estimated as a function of the reprojection error. A pair of trajectories is considered as generated from the same object if the reprojection error is minimum.

Once the trajectories are reconstructed, the next step is to apply clustering to learn the underlying hidden motion structures, clusters, and to learn the dynamics of the scene i.e., identification of normal and abnormal trajectories. In the next

Table 2.3: State-of-the-art methods for trajectory representation (Key. *HMMs*: Hidden Markov Models; *PRMs*: Probabilistic and Regression Models; *STFA*: Spatio-temporal Function Approximations; *PCA*: Principal Components Analysis; *ICA*: Independent Components Analysis; *TDH*: Trajectory Directional Histogram).

Category	**Rep.**	**Application**	**Ref.**
Supervised	HMMs	Vehicle tracking	[40]
		Nose tracking	[41]
		Gesture recognition	[42]
		Behaviour analysis	[43]
	PRMs	Hand tracking	[12]
		ECG and cyclone trajectories	[44]
Unsupervised	STFA	Pedestrian tracking	[13]
		Behaviour analysis	[45]
		Pedestrians scene	[46]
		Speech signal analysis	[47]
		Vehicle tracking	[48]
	PCA	Hand tracking,	[49]
		Sport's analysis	[50]
		Traffic analysis	[17]
	ICA	Pedestrians counting	[11]
	TDH	Vehicle tracking	[16]

section, we provide details of existing approaches for trajectory clustering.

2.3 Trajectory clustering

Trajectory clustering plays a vital role in understanding the scene dynamics. The process normally consists of two steps: (a) trajectory representation and (b) trajectory grouping. The details of these are provided in subsequent sections.

2.3.1 Trajectory representation

The choice of a suitable pattern representation provides the core for a clustering algorithm. This section discusses and compares existing approaches for trajectory representation, which are summarised in Table 2.3. Pattern representations can also be divided into *supervised* ([40, 41, 43, 42, 44, 12]) and *unsupervised* [45, 13, 48, 47, 46] classes. A Hidden Markov Model (HMM) is a supervised trajectory representation approach, in which the state transition matrix represents the dynamics of a trajectory. Porikli [40] performs trajectory clustering using eigenvector analysis on the HMM parameter space. Alon *et al.* [41] allow each sequence to belong to more than a single HMM with some probability and the hard decision about the sequence class membership is deferred until a later stage for final clustering. Parameterized-HMMs [42] and coupled-HMMs [43] are also used to recognise more complex events such as moving object interactions. Although HMMs are robust to dynamic time warping, the structures and probability distributions are highly domain-dependent e.g., models to represent people motion would be completely different from vehicle motion models in terms of number of hidden states and probabilities. Moreover, the parameter space increases considerably in size with the complexity of the events, as more hidden states are required for modelling.

Self-organizing maps (SOMs) provide one of the examples of unsupervised representation to project high-dimensional data patterns in a low dimensional space, while keeping the topological properties of the input space [45, 13, 48]. Once SOM nodes are organised, all the data associated with a given node may be made available via that node. For real motion sequences, the convergence of these techniques is slow and the learning phase is usually carried out off-line. The Trajectory Directional Histogram (TDH) is another representation to encode the statistical directional distribution of the trajectories [16]. However, this feature alone does not suffice because it does not encode spatial information. Therefore, two trajectories that are far on the image plane shall be clustered together if they have similar directional histograms. In the clustering literature, Principal Components Analysis (PCA) has been used extensively to reduce the dimensionality of the data set prior to clustering while extracting the most important data variations. Bashir *et al.* [17, 49, 50] represent trajectories as a temporal ordering of sub-trajectories. These sub-trajectories are then represented by their PCA coefficients for optimally compact representation. PCA works well for data with a single Gaussian distribution. For a mixture of Gaussian distributions, Independent Components Analysis (ICA) is used to obtain a compact representation. Antonini *et al.* [11] transform the input trajectories using ICA and then use the Euclidean distance to find the similarities among trajectories. Both PCA and ICA require an accurate estimation of the noise covariance matrix from the data, which is generally a difficult task. Furthermore, in their standard form, they do not contain high-order statistical information and therefore the analysis is limited to second-order statistics.

2.3.2 Trajectory grouping

We classify the trajectory grouping techniques into hierarchical [51] and partitional [52]. The details of existing works for each class are discussed in rest of this section.

Hierarchical clustering

Hierarchical clustering provides a nested sequence of partitions [15]. These methods can further be divided into two classes, namely agglomerative and divisive. Agglomerative clustering methods start by first considering each trajectory as a separate cluster and then merge the clusters in a nested sequence [53]. On the other hand, divisive clustering starts with all trajectories in one single cluster, and then successively splits the cluster to obtain the final partition [54, 55]. The level of the tree structure depends upon the choice of threshold and is application specific. The computation of the tree structures (dendograms) is expensive and impractical with more than a few hundred patterns [56].

Partitional clustering

Partitional clustering is more suitable for the analysis of large data collections as it generates clusters iteratively by minimizing an objective function. Partitional clustering methods can be further divided into two classes, namely hard (crisp) and soft (fuzzy). In hard clustering, each trajectory is assigned to one cluster only; in soft clustering each trajectory is assigned a degree of membership to each cluster. Furthermore, each class can be further divided into parametric and non-parametric sub-classes. The following abbreviations are used in the remainder of this thesis: PHC for parametric-hard-clustering methods, NPHC for non-parametric-hard-clustering methods, PSC for parametric-soft-clustering methods and NPSC for non-parametric-soft-clustering methods.

In PHC, the trajectories are clustered into pre-specified number of partitions with a cluster representative for each cluster. A widely used algorithm is the iterative K-means [57, 58], which works in two steps, namely assignment and update. In the assignment step, each trajectory represented as in a particular feature space is assigned to the cluster, which has the smallest distance from the trajectory. In the update step, the mean of the cluster is re-calculated. The process terminates when either the change in clusters' mean is less than a threshold or the number of iterations reaches a pre-defined value.

Since K-means tends to associate each trajectory to one cluster, outlier trajectories can affect the overall shape of the clusters. To overcome this limitation, a Self-Organizing Maps (SOMs) based approach can be used [14]. Initially, each trajectory can be represented with number of coefficients of the Discrete Fourier Transform (DFT). In the training phase, using these coefficients the SOM randomly initialises a weight vector associated to the neuron outputs, which are initially set to more than require for coarse clustering. The input trajectory is assigned to an output neuron for which it has minimum Euclidean distance. At the end of the training phase, the most similar cluster pairs are merged for fine clustering.

Dual Hierarchical Dirichlet Processes (Dual-HDP) is an example of NPHC algorithm, which is inspired from document mining [39]. Trajectories are treated as documents and the observations of an object on a trajectory are treated as words in a document. Trajectories are clustered in an iterative way. In each iteration the process performs clustering at two levels, namely at observation-level and at trajectory-level. The first level helps in finding the regions and the second level associates each trajectory to one region. An abnormal trajectory is defined as one that does not fall in dense regions. Another NPHC approach is presented in [59], where the first trajectory forms the first cluster. Each next trajectory is compared with the existing cluster(s). A trajectory is assigned to a cluster if it is sufficiently close to that cluster. If a trajectory is not assigned to any of the existing clusters; then a new cluster is initiated with that trajectory. A similar approach is presented in [60], where clusters are constructed online using partial trajectory segments. Clusters are organised by a tree-like structure, where each node corresponds to a cluster of similar partial paths. Events (primarily unusual) are detected based on statistical analysis of the path the object follows through the tree.

Hard partitional clustering methods (PHC and NPHC) work well when the physical boundaries of the clusters are well-defined. An advantage of soft clustering over hard clustering is that it yields more detailed information on the structure of the data as it may assign each element to multiple clusters with an associated membership value [61]. Furthermore, fuzzy clustering is less sensitive to outliers that will have a smaller membership value with a particular cluster and thus affects less on the overall cluster structure.

A popular PSC algorithm is Fuzzy K-means, a variant of the K-means algorithm, which assigns each trajectory a certain degree of belongingness to the clusters. Thus, trajectories on the edge of a cluster would have a smaller degree of membership than the trajectories in the centre of the cluster. Although this algorithm minimizes the intra-cluster variance, it has the same local minimum problem as K-means, and the results depend on the initial choice of the number of clusters.

2.3.3 Outliers

An outlier trajectory is the one that deviates so much from other trajectories as to raise suspicions that it is a result of an abnormal event. In the anomaly detection literature, distance-based techniques are frequently used. Zhou *et al.* [62] use an Edit distance to find the common pattern. A trajectory that is far from the common pattern is considered as an anomaly. Similarly, Naftel *et al.* [63] use the Hotelling T^2 test to determine if the Mahalanobis distance of a trajectory to its nearest class centre makes it an outlier. The choice of the threshold value is selected according to the given dataset. Furthermore, Fu *et al.* [15] use a Gaussian distribution to represent the test and template trajectories. If the difference between the test trajectory and the template trajectory is larger than one standard deviation from the mean of a template trajectory, then the test trajectory is considered as abnormal. These distance based outlier detection techniques work well if the trajectory has completely different direction, starting (or ending) points, or velocities from those of other trajectories. However, it is not clear whether these techniques can detect outlying sub-trajectories from a set of very complicated trajectories. Another interesting classification based approach is presented by Li *et al.* [64]. In this approach, common patterns called motifs are extracted from trajectories, and the set of motifs forms a feature space in which the trajectories are placed. Through the transformation into a feature vector, the trajectories are fed into a classifier. This algorithm depends on training. More specifically, a classification model is built using the training set, and a new trajectory is classified into either normal or abnormal. The performance of the algorithm is highly dependent upon the reliability of the training trajectory dataset, which is not always easy to obtain in real scenarios.

2.4 Summary

This chapter has initially provided an overview of existing *camera calibration* techniques. Techniques for estimating intrinsic as well as extrinsic calibration parameters have been discussed. Particularly, for extrinsic calibration of non-overlapping cameras, it has been observed that existing works either know or assume motion models of objects or environment map. Therefore, there is still a need to develop an approach that could localise a network of cameras even without any prior information. Next, this chapter has discussed object correspondence techniques to reconstruct *global* trajectory of an object across multiple cameras. From the literature review, it has been observed that existing approaches perform association of objects either on the image plane or on the ground plane. The association is mainly performed based on a temporal threshold. These approaches work well when there are few objects moving in a scene. The performance degrades substantially when high density of similar objects is moving, which results in either no or multiple associations of an object among multiple views. To overcome these issues, there is a need to develop a hybrid approach that can combine the strengths of both image plane and ground plane associations and should be efficient in performance in both low and high density of simultaneously moving objects. Finally, techniques for *trajectory clustering* have been discussed. We have learnt that these techniques use either single or concatenated multiple feature spaces, which result in a coarse cost function defined by the proximity measure, thus leading to a local minima problem. In order to obtain a higher degree of descriptiveness of the trajectories, it is desirable to use multiple independent features simultaneously.

Chapter 3

Camera calibration

3.1 Introduction

The deployment of camera networks is essential for the observation of large environments in applications such as traffic and behaviour monitoring, remote surveillance and sporting events [65, 66]. These networks may be characterised by the presence of cameras with non-overlapping fields of view. Prior to applying video analytics in a camera network, internal as well as external calibration of each camera of the network has to be performed [67]. This calibration helps in solving camera handovers of targets and reconstructing complete trajectories from partial views [68]. External calibration (also referred to as localisation) provides information about the relative position and orientation of a camera with respect to the rest of the network. When the number of cameras is large, it is impractical to employ manual localisation techniques. Moreover, the use of GPS-based methods might not be viable all the time, as they are more expensive and there are scenarios when they are not suitable at all (e.g., when monitoring an underground train station [2]). There is therefore the need for an automatic camera localisation approach that can exploit the information observed in each camera view. In this chapter, we focus on self-calibration of a single camera and extrinsic calibration of non-overlapping camera network.

3.2 Self-calibration via view rectification

Self-calibration plays a fundamental role in case of using single camera [69, 70, 71]. In this section, we discuss the approach of self-calibration of a camera using view rectification.

Instead of assuming a monitored scene as a 3D Euclidean space containing a complete metric structure, we can consider it as being embedded in an affine or even projective space [19]. Under this assumption, Liebowitz describes the geometry, constraints and algorithmic implementation for metric rectification of planes [29]. Let $\aleph_r$ and $\aleph_i$ represent the real and the image plane, respectively. The mapping between the two planes is a general planar homography on the form $\aleph_i = \mathcal{H}\ \aleph_r$, with $\mathcal{H}$ a 3×3 matrix of rank 3. This projective transformation can be visualised as a chain of transformations on the form

$$\mathcal{H} = \mathcal{S}\varsigma\rho, \tag{3.1}$$

where $\mathcal{S}$ represents the similarity transformation, ς represents the affine transformation, and ρ represents the pure projective transformation. Since the similarity transformation changes the coordinates linearly and does not play any role in the perspective view, in this work we have not removed this transformation from the image. The details of view rectification are provided in subsequent sections.

3.2.1 Inverse perspective transformation

The first step for view rectification is to determine the transformation ρ defined as

$$\rho = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}, \tag{3.2}$$

where $l_\infty = (l_1, l_2, l_3)^T$ is the vanishing line of the plane. Parallel lines on the real-plane intersect at vanishing points in the image plane on the vanishing line. A set of real plane parallel lines are identified by manually selected four points $\{P1, \ldots, P4\}$ on the image plane. The lines are projected to find the intersection or vanishing point for the lines. An illustration of vanishing line construction is given in Fig. 3.1. Once ρ is determined, the image can be affine-rectified and affine properties can be measured.

3.2.2 Inverse affine transformation

To remove affine projection, the transformation ς can be represented with two degree of freedoms:

$$\varsigma = \begin{pmatrix} \frac{1}{\kappa_2} & -\frac{\kappa_1}{\kappa_2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.3}$$

where κ_1 and κ_2 represent the image as of the circular points in the complex domain. The assumption is useful for an invariant representation of the image to Euclidean transformation. Liebowitz [29] presented various procedures to solve for the values of the two parameters. To generate constraint circles from known angles and length ratios in the image, we manually select four points $\{S1, \ldots, S4\}$ (square structure on

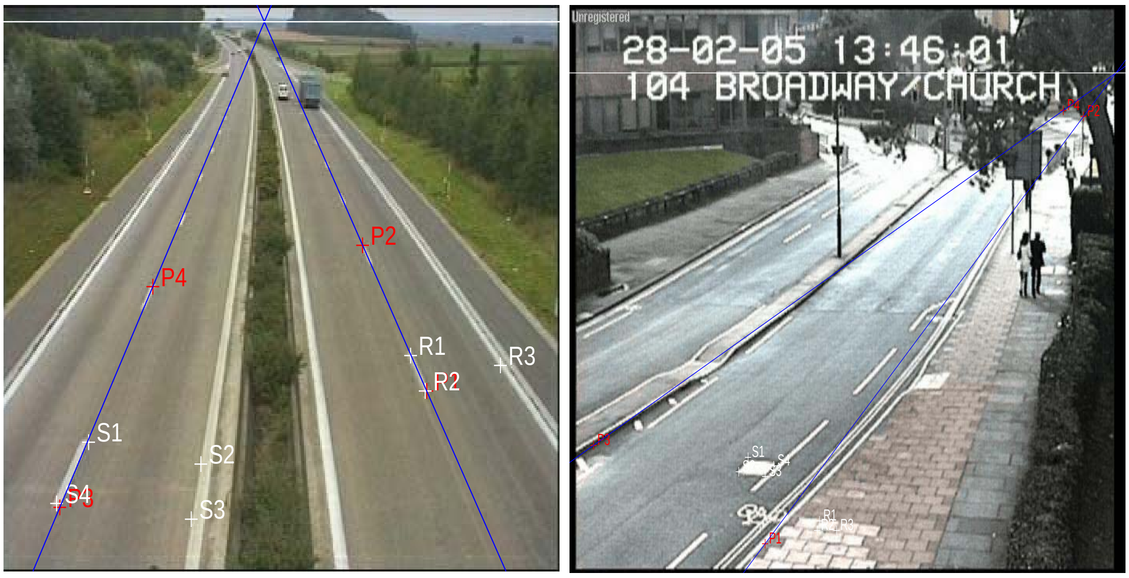


Figure 3.1: Example of vanishing line construction. The blue lines represent pairs of parallel lines intersecting on the vanishing points. The white line represents the vanishing line. $\{S1, \ldots, S4\}$ and $\{R1, \ldots, R3\}$ are used for known angles and length ratio selection.

the real plane) to represent line segments with known length ratio and three points $\{R1, \ldots, R3\}$ to represent right angle formed in real scene. Fig. 3.1 also shows the selected points for square and right-angled structures.

The circle parameters with known angles are calculated as

$$\left\{ \begin{array}{l} \left(\frac{d_1+d_2}{2}, \quad \frac{d_1-d_2}{2} cot\theta \right) \\ \left| \frac{d_1-d_2}{2sin(\theta)} \right| \end{array} \right. , \tag{3.4}$$

where $(\frac{d_1+d_2}{2}, \frac{d_1-d_2}{2}cot\theta)$ is the 2D coordinate of the centre of the constraint circle, $|\frac{d_1-d_2}{2sin(\theta)}|$ is the radius of the constraint circle with $\theta = \frac{\pi}{2}$.

To calculate circle parameters with known length ratio, let d_{lx} and d_{ly} represent the horizontal and vertical directions of a line l and let s be the known length ratio.

Then,

$$\begin{cases} \left(\frac{d_{1x}d_{1y}-s^2d_{2x}d_{2y}}{d_{1y}^2-s^2d_{2y}^2}, \ 0 \right) \\ \left| \frac{s(d_{2x}d_{1y}-d_{1x}d_{2y})}{d_{1y}^2-s^2d_{2y}^2} \right| \end{cases}, \tag{3.5}$$

again, $(\frac{d_{1x}d_{1y}-s^2d_{2x}d_{2y}}{d_{1y}^2-s^2d_{2y}^2}, 0)$ and $|\frac{s(d_{2x}d_{1y}-d_{1x}d_{2y})}{d_{1y}^2-s^2d_{2y}^2}|$ are the centre and radius of the circle. The final values of κ_1 and κ_2 are calculated by finding a point of intersection of both constraint circles. The affine removal makes it possible to get the metric properties of the plane. The steps for the perspective rectification procedure can be summarised as follows:

- Select $\{P1, \ldots, P4\}$ to define the vanishing line; $\{S1, \ldots, S4\}$ and $\{R1, \ldots, R3\}$ that define known angle and length ratios within a real-world structure. The structures may exist at different heights in the scene.

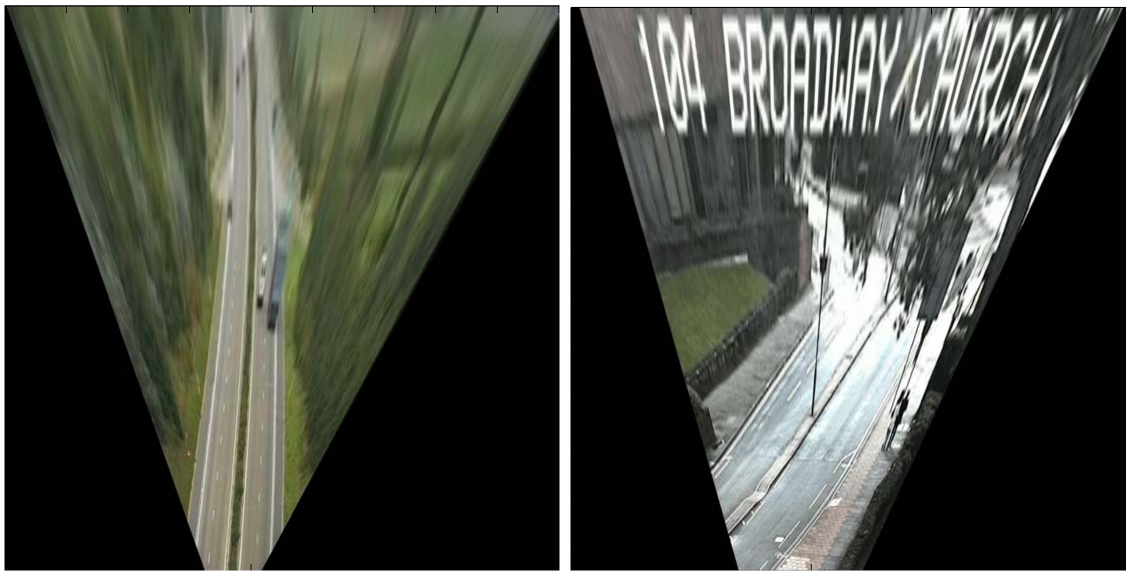
- Calculate the inverse projection transformation (Eq. 3.2).

- Rectify the pure projection from the image by applying ρ.

- Calculate the inverse affine transformation (Eq. 3.4 and Eq. 3.5).

- Rectify the affine transformation from the image by applying ς.

Sample rectified images transformed using this procedure are shown in Fig. 3.2. In next section, we extend the concept of the calibration from single camera to multiple cameras.



(a) (b)

Figure 3.2: Sample rectified images using single camera calibration.

3.3 Extrinsic calibration of non-overlapping cameras

Let $C = \{c_1, c_2, ..., c_N\}$ be a network of N cameras and $z^m = (x^m, y^m)$ the position of a moving object within the field of view[1] of camera c_i at time instant m. Let each camera provide a vertical top-down view of a portion of the scene, either because its optical axis is perpendicular[2] to the ground plane or because its view is normalised [72]. Under this assumption, the parameters for localising camera c_i are its *position*, (x_i, y_i), and its *rotation angle*, r_i, about its optical axis, measured with respect to the horizontal axis of the ground plane. The set $\mathcal{L}$ of unknowns (Fig. 3.3) is therefore

$$\mathcal{L} = \{(x_1, y_1, r_1), (x_2, y_2, r_2), ..., (x_N, y_N, r_N)\}. \tag{3.6}$$

[1]For clarity of notation we do not use the subscript i for the position of an object within camera c_i.

[2]Due to top-down assumption, the optical centre of a camera is perpendicular to the image centre

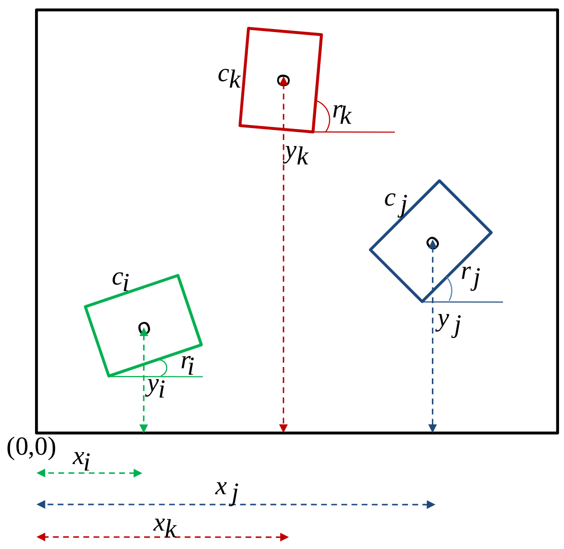


Figure 3.3: Illustration of the unknown localisation parameters for the definition of the configuration of the top-down camera network. The centre of each camera c_i is shown by a circle; (x_i, y_i) is the position of c_i and r_i is the orientation or rotation of c_i with reference to the horizontal axis of a common ground plane.

Moreover, let us define camera c_i and camera c_j as a *camera-pair* i.e., when an object exits the field of view of c_i and enters the field of view of c_j, where $j \neq i$, without being observed by another camera. Note that, according to this definition, c_i and c_j may become a camera-pair, even if c_i and c_j are not physically close to each other (Fig. 3.4). The flow diagram of the proposed approach for an iteration is shown in Fig. 3.5. The details of algorithmic steps for the entire network are provided in Algo.1 and described in the following sections.

3.3.1 Pre-processing

The observations in the field of view of each camera (trajectory segments) are first pre-processed to remove outliers i.e., noisy observations. As the trajectory segment can be corrupted by various types of estimation noises, we use RANSAC [20] to smooth it before further processing. Instead of using all the available data,

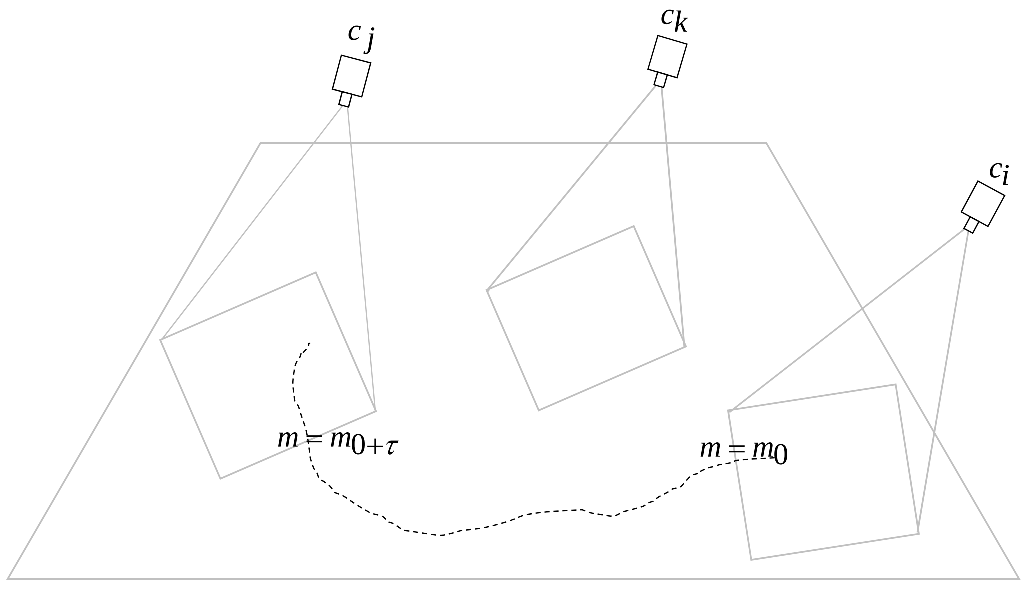


Figure 3.4: Example of formation of a camera-pair (c_i and c_j): an object exits from c_i at m=m_0 and enters into the field of view of c_j at m=$m_{0+\tau}$ without being observed by another camera.

RANSAC starts with a small subset of the complete data and then adds data which are consistent with the assumed model. This makes RANSAC able also to filter out noisy observations that are unrelated to the instantaneous function of the object state (beyond the measurement error).

Let the motion parameters be approximated with a polynomial of degree l. We start with n_b observations chosen randomly from a trajectory segment. Next, within this subset, we find the observations that are within a given tolerance of the polynomial model. We repeat the process until the number of observations within this tolerance is more than 50% of the total number of observations of that trajectory segment in the camera view. The observations that fall outside the region of tolerance are considered outliers and therefore discarded from further processing.

Once the trajectories are filtered within the field of view of a camera, they are used to estimate the movement of the object in the unobserved regions. We assume

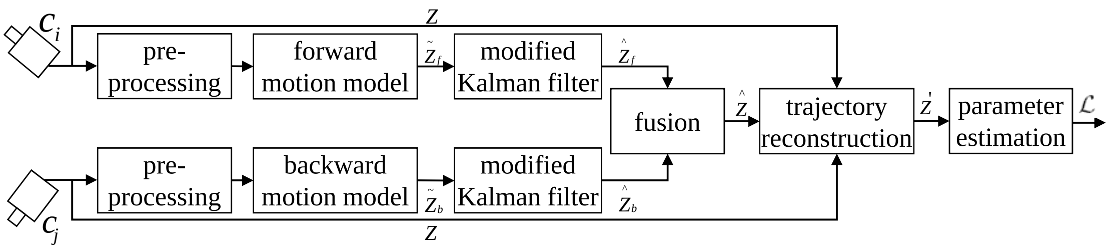


Figure 3.5: Iteration of the parameter estimation for a camera-pair.

that the object can take one sharp turn in unobserved region. The details of the process is described in the next section.

3.3.2 Trajectory estimation

In order to generate the complete trajectory

$$Z' = \{z^0, ..., z^{m_0}, \hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}, z^{m_0+\tau}, ..., z^{m_0+\tau+m_1}\} \tag{3.7}$$

of the moving object, we use the spatio-temporal information derived from available trajectory segments

$$Z = \{z^0, ..., z^{m_0}, z^{m_0+\tau}, ..., z^{m_0+\tau+m_1}\} \tag{3.8}$$

in the fields of view of the cameras in order to estimate the trajectory points in the unobserved regions

$$\hat{Z} = \{\hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}\}. \tag{3.9}$$

The estimation of the trajectories in the unobserved region between c_i and c_j is performed in three steps: (i) forward and backward estimations, (ii) Kalman filtering and (iii) fusion. First, we estimate the sequence of positions from time instant m_0, corresponding to the last observation in c_i, to $m_0 + \tau$, corresponding to the first observation in c_j. This estimation is performed in the forward and backward directions. The forward estimation generates the series of candidate positions

$$\tilde{Z}_f = \{\tilde{z}_f^{m_0+1}, ..., \tilde{z}_f^{m_0+\tau-1}\}; \tag{3.10}$$

whereas the backward estimation generates the series of candidate positions

$$\tilde{Z}_b = \{\tilde{z}_b^{m_0+1}, ..., \tilde{z}_b^{m_0+\tau-1}\}. \tag{3.11}$$

Let each observation $\tilde{z}_f^{m+1}$=$(\tilde{x}_f^{m+1}, \tilde{y}_f^{m+1})$ be generated by a forward motion model as

$$\begin{bmatrix} \tilde{x}_f^{m+1} \\ \tilde{x}_{\nu,f}^{m+1} \\ \tilde{y}_f^{m+1} \\ \tilde{y}_{\nu,f}^{m+1} \end{bmatrix} = \begin{bmatrix} 1 & a_{x,f} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_{y,f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_f^{m} \\ \tilde{x}_{\nu,f}^{m} \\ \tilde{y}_f^{m} \\ \tilde{y}_{\nu,f}^{m} \end{bmatrix} + \begin{bmatrix} v_x^m \\ v_{\nu,x}^m \\ v_y^m \\ v_{\nu,y}^m \end{bmatrix}, \tag{3.12}$$

where $(\tilde{x}_{\nu,f}^m, \tilde{y}_{\nu,f}^m)$ is the velocity of the object. Furthermore, $a_{x,f}$ and $a_{y,f}$ are computed independently for each available trajectory segment using the l^{th} order polynomial fitting on observations from c_i [73]. Moreover, $\mathbf{v}^m(\mathcal{N}(0, \Sigma_{v^m}))$ is modelling additive noise with covariance Σ_{v^m}=$diag([1e^{-3}, 1e^{-3}, 1e^{-3}, 1e^{-3}])$.

Similarly, let each observation $\tilde{z}_b^m$=$(\tilde{x}_b^m, \tilde{y}_b^m)$ be generated by a backward motion model as

$$\begin{bmatrix} \tilde{x}_b^m \\ \tilde{x}_{\nu,b}^m \\ \tilde{y}_b^m \\ \tilde{y}_{\nu,b}^m \end{bmatrix} = \begin{bmatrix} 1 & a_{x,b} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_{y,b} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_b^{m+1} \\ \tilde{x}_{\nu,b}^{m+1} \\ \tilde{y}_b^{m+1} \\ \tilde{y}_{\nu,b}^{m+1} \end{bmatrix} + \begin{bmatrix} v_x^{m+1} \\ v_{\nu,x}^{m+1} \\ v_y^{m+1} \\ v_{\nu,y}^{m+1} \end{bmatrix}, \tag{3.13}$$

where $a_{x,b}$ and $a_{y,b}$ are calculated like $a_{x,f}$ and $a_{y,f}$, but using the observations from c_j.

Before fusing the forward and backward estimated trajectories, we filter them using a modified Kalman filtering to obtain smoothed estimated positions, i.e., $\hat{Z} = \{\hat{z}^{m_0+1}, ..., \hat{z}^{m_0+\tau-1}\}$. The Kalman gain, k^m, is calculated as

$$k^m = \Sigma_{v^{m-1}} A[A^T \Sigma_{v^{m-1}} A + \Sigma_{w^m}]^{-1}, \tag{3.14}$$

where Σ_{w^m} is the covariance of the observation noise at instant m and A maps the state vector with the measurements. Both Σ_{v^m} and Σ_{w^m} are updated as described in [74]. The object position is updated using a forward motion model as

$$\hat{z}_f^{m+1} = \hat{z}_f^m + k^m(\tilde{z}_b^m - A(\hat{z}_f^m)). \tag{3.15}$$

Note that $\tilde{z}_b^m$ is used to calculate the residual $\tilde{z}_b^m - A(\hat{z}_f^m)$, which is the discrepancy between the forward estimation and the backward estimation. Zero residual means that the forward and the backward measurements are in agreement. This modification ensures that the object will be in the correct state at $m_0 + \tau$, in accordance with the observation in c_j.

Algorithm 1 Automated localisation of a camera network.

```
Variables:
$ID^t$: ID of the camera observing the target at $t$
$I$, $J$: IDs of the first and second cameras in a camera-pair
$\zeta$: overall change in camera positions
$\delta$: minimal change threshold
$\kappa$: iteration number
Initialisations
t=1; $\zeta$ = Inf; $\kappa = 0$
 1: while $\zeta \geq \delta$ do
 2:     increment $\kappa$          % start iterations
 3:     $I = ID^{t-1}$
 4:     while $I == ID^t$ do
 5:         $z_I^{t-1} = (x_I^{t-1}, y_I^{t-1})$
 6:         $m_0$ = t-1
 7:         increment t
 8:     end while
 9:     if target is unobserved then
10:         increment t
11:     else
12:         $J = ID^t$
13:         $m_{0+\tau}$ = t
14:         increment t
15:         while $J == ID_t$ do
16:             $z_J^{t-1} = (x_J^{t-1}, y_J^{t-1})$
17:             $m_{0+\tau+m1}$ = t-1
18:             increment t
19:         end while
20:         apply RANSAC on Z (Sec. II.B)
21:         compute $\tilde{Z}_f$ and $\tilde{Z}_b$ using Eq.(7) and Eq.(8), respectively
22:         compute $\hat{Z}_f$ and $\hat{Z}_b$ using Eq.(10) and Eq.(11), respectively
23:         compute $\hat{Z}$ using Eq.(12)
24:         compute $D_{I,J}^{\kappa}$ using Eq.(13)
25:         compute $r_{I,J}^{\kappa}$ using Eq.(18)
26:         $p_I^{\kappa} = (x_I, y_I)$
27:         $p_J^{\kappa} = (x_J, y_J) + D_{I,J}^{\kappa}$
28:         $\mathcal{L}_{I,J}^{\kappa} \leftarrow (p_I^{\kappa}, p_J^{\kappa}, r_{I,J}^{\kappa})$
29:         t = $m_{0+\tau}$
30:         $\zeta = \sum_{n=1}^{N} Norm_2(p_n^{\kappa}, p_n^{\kappa-1})$
31:     end if
32: end while                  % end iterations
33: compute $\mathcal{L}^*$ using Eq.(19)   % non-iterative refinements
```

Similarly, the object position is updated using the backward motion model as

$$\hat{z}_b^{m+1} = \hat{z}_b^{m} + k^m(\tilde{z}_f^m - A(\hat{z}_b^m)). \tag{3.16}$$

Again, in the residual term $\tilde{z}_f^m - A(\tilde{z}_b^m)$, $\tilde{z}_f^m$ is used to ensure that the estimated object state at instant m_0 is in accordance with the observation in c_i.

The filtered forward and backward estimated segments are finally fused as:

$$\hat{z}^m = (1 - \beta^m)\hat{z}_f^m + \beta^m \hat{z}_b^m, \tag{3.17}$$

where $\beta^m = m/m_0 + \tau$ for $m = 1, ..., m_0 + \tau$. The forward estimations get higher weights when the object is closer to c_i and these weights are progressively reduced when the object moves away from c_i.

Figure 3.6 shows a trajectory estimation comparison with and without Kalman filtering corrections. Without the corrections, both the forward and the backward estimates end far from the real object position in the fields of view of the cameras. Kalman filtering followed by fusion improves the estimation of the object position (green).

3.3.3 Position and orientation estimation

To localise the cameras we need to calculate their relative position (the distance between their centres) and their relative orientation. To estimate the relative position of c_j with respect to c_i, we use the displacement vector, D_{ij}, calculated as

$$D_{ij} = z^{m_0} - \hat{z}^{m_0+\tau}, \tag{3.18}$$

where z^{m_0} is the last observation in c_i and $\hat{z}^{m_0+\tau}$ is the first estimated object position in c_j. Furthermore, to estimate the relative orientation $\hat{r}_{ij}$ of the camera-pair, we could calculate the angle between the observed object position $z^{m_0+\tau}=(x^{m_0+\tau}, y^{m_0+\tau})$

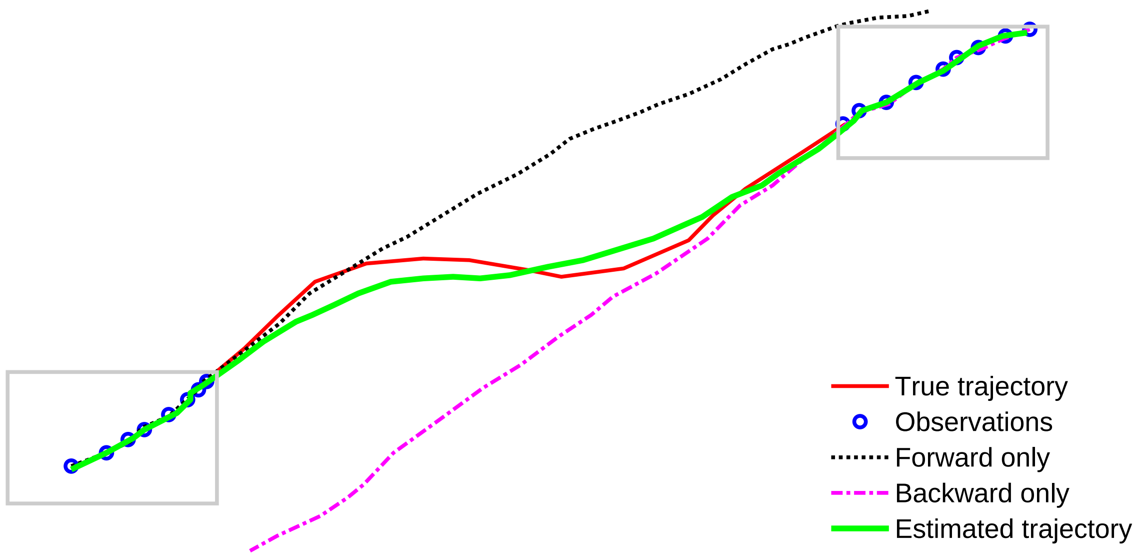


Figure 3.6: An illustration of trajectory estimation with and without Kalman corrections on synthetic data (key; *gray*: two non-overlapping cameras; *blue* circles: observations in each camera (Z); *red* line: the true trajectory; *black* dots: forward estimation ($\tilde{Z}_f$); *magenta* line-dots: backward estimation ($\tilde{Z}_b$); *green* line: estimated trajectory ($\hat{Z}$)).

and the corresponding estimated object position $\hat{z}^{m_0+\tau}$=$(\hat{x}^{m_0+\tau}, \hat{y}^{m_0+\tau})$ in camera c_j as

$$\hat{r}_{ij} = cos^{-1}\frac{(x^{m_0+\tau}, y^{m_0+\tau}).(\hat{x}^{m_0+\tau}, \hat{y}^{m_0+\tau})}{|(x^{m_0+\tau}, y^{m_0+\tau})||(\hat{x}^{m_0+\tau}, \hat{y}^{m_0+\tau})|}. \tag{3.19}$$

However, to increase the robustness of the orientation estimation process, instead of relying on single instances of observation and estimation, we use a sub-segment of the available trajectory segment in c_j. This sub-segment is extracted using the directional angle $\alpha(.)$, calculated as

$$\alpha(q) = tan^{-1}\frac{y^{m_0+\tau+q} - y^{m_0+\tau+q-1}}{x^{m_0+\tau+q} - x^{m_0+\tau+q-1}}, \tag{3.20}$$

where $q = 1, ..., m_1$ and m_1 is the number of consecutive observations in c_j. To find the time m_s at which the object changes its direction considerably, we use

$$m_s = \begin{cases} m_0 + \tau + q & \text{if} \quad |\alpha(q) - \alpha(q+1)| > \xi \\ m_0 + \tau + m_1 & \textit{otherwise} \end{cases}, \tag{3.21}$$

where m_s is the instant of change. Equation 3.21 explains that if the change in object direction is significant ($\xi = \pi/12$) at $m_0 + \tau + q$ within the field of view of c_j, then we consider the trajectory segment only to that point for estimating the relative orientation; otherwise, the complete trajectory segment is used. To generate an estimated sequence

$$\hat{Z}^s = \{\hat{z}^{m_0+\tau}, ..., \hat{z}^{m_s}\}, \tag{3.22}$$

we extrapolate further the trajectory estimate from instant $m + \tau$ to m_s. We rotate the observed segments with $\theta = -\pi + n.\frac{\pi}{180}$ where $n = 0, 1, ..., 360$. The final relative orientation, r_{ij}, between c_i and c_j is calculated as

$$r_{ij} = \arg\min_{\theta} \ \mathcal{D}(\hat{Z}^s, \mathcal{V}_\theta(Z^s)), \tag{3.23}$$

where $\mathcal{V}_\theta(.)$ rotates Z^s at an angle θ and $\mathcal{D}(.,.)$ calculates the L_2 distance between the estimated and (rotated) observed trajectory segments.

The process of estimating the relative position and orientation continues and iteratively approximates the configuration of all cameras with respect to each other. The process terminates when the total change in position estimation is smaller than a pre-defined threshold, δ, for example $\delta = 1\%$ of the area representing the field of view of a camera.

Finally, we can refine the localisation results, $\mathcal{L} = \{(p_1, r_1), (p_2, r_2), ..., (p_N, r_N)\}$

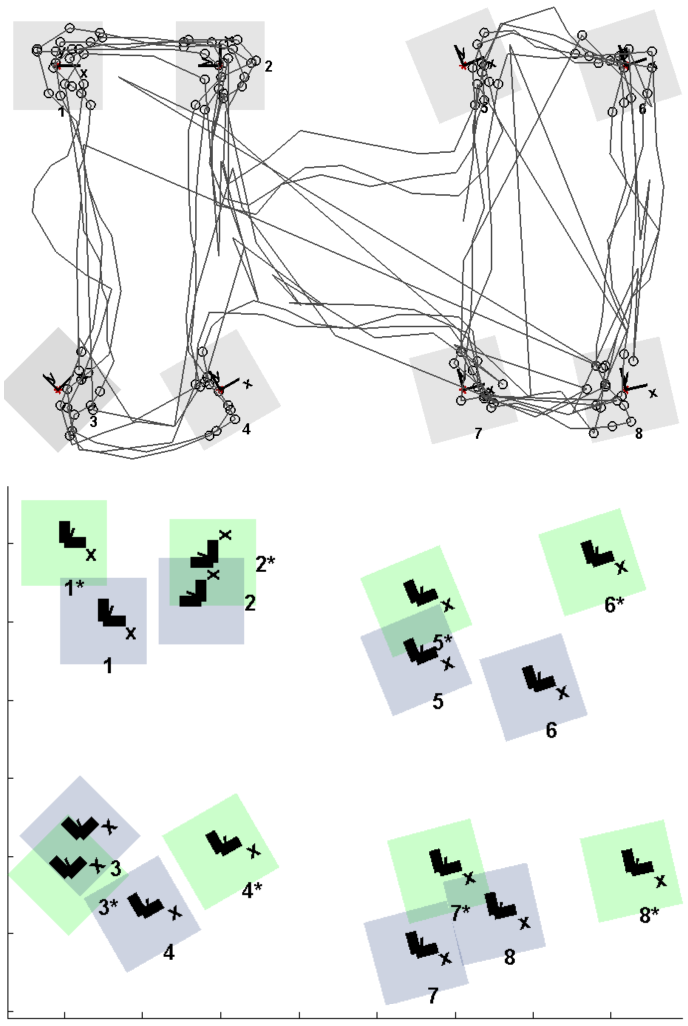


Figure 3.7: An example of improvements of the estimation results using bundle adjustment: (top) input dataset (key; gray square: camera's field of view; gray line: ground truth trajectory; gray circles: observations) and (bottom) localisation of each camera, where the estimation before (blue) and after (green and * with each camera ID) bundle adjustments are superimposed.

with $p_i{=}(x_i, y_i)$, by employing bundle adjustment [75]. The cost function for bundle adjustment is defined to ensure that the localisation parameters allow for the minimal difference between observations and estimates:

$$\mathcal{L}^* = \underset{\mathcal{L}=(r_i,p_i)}{argmin} \sum_{i=1}^{N} ||Z_i - r_i(p_i + \hat{Z}_i)||^2, \tag{3.24}$$

where Z_i and $\hat{Z}_i$ are sets of observed and estimated positions in c_i. Figure 3.7 shows an example of refinements with bundle adjustment. Figure 3.7(top) shows the true camera configuration of the network along with the object trajectory. Figure 3.7(bottom) shows the initial configuration results (blue) that are refined after applying bundle adjustment (green). Significant improvements can be observed in the relative positioning of c_3, c_4, c_7 and c_8.

3.4 Summary

In this chapter, we have presented an approach for self-calibration of a single camera using view rectification and then we focused on the extrinsic calibration of a non-overlapping camera network. To this end, we proposed an iterative algorithm for the localisation of a network of non-overlapping cameras that allows us to relax the traditional linearity or smooth turn constraint on the motion of objects. The algorithm recovers the position and the orientation of each camera based on two main steps: the estimation of the unobserved trajectory in regions not covered by the cameras' fields of view and the estimation of the relative orientations of the cameras. Kalman filtering initially employed on the available trajectory segments to extrapolate their values in the unobserved regions, with the modification that the forward motion model provides measurements for the backward trajectory estimation and vice versa. This helped in improving the accuracy of the trajectory estimation in the unobserved regions. The relative orientation of the cameras is then obtained by using the estimated and the observed exit and entry point information in each camera field of view. The initial localisation of the network is refined by bundle adjustment. The evaluation of the proposed approach and its comparison with an existing work on simulated as well as real datasets is presented in Ch. 6. Furthermore, the calibrated cameras can be used to fuse various streams and to generate a global view [76]. To this end, a challenging task is to establish object correspondence across multiple views [77]. In Ch. 4, we present an algorithm that estimates object correspondence and reconstruct a global trajectory for each object on a common plane.

Chapter 4

Global trajectory reconstruction

4.1 Introduction

The reconstruction of objects' trajectories across cameras facilitates the recognition of global behaviours for large scale events in applications such as sports analysis, remote sensing and video surveillance. This requires a mechanism for associating and integrating partially observed data in each camera view. Local trajectory information from individual cameras may be corrupted by inaccuracies due to noise, objects re-entrances, occlusions and by errors due to crowded scenes. Therefore, trajectory association becomes a difficult task under such complex scenarios. In this chapter, we consider the problem of object association across partially overlapping cameras using local trajectories. Initial correspondence among trajectories is established on the ground plane using multiple spatio-temporal features and then image plane reprojections of the matched trajectories are employed to resolve conflicting situations. This makes sure that only one trajectory of an object from each camera is associated to other cameras. The fusion is then applied to combine matched trajectories. A spatio-temporal linkage procedure connects the fused segments in order

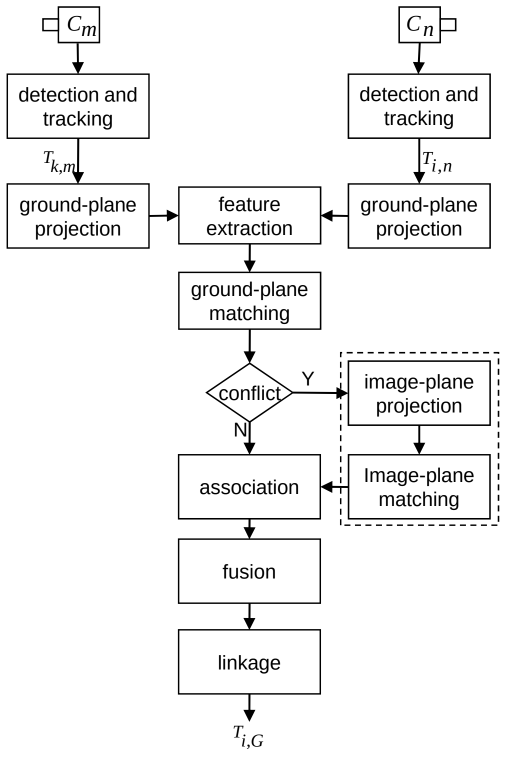


Figure 4.1: Flow diagram for reconstruction of a global trajectory.

to obtain the complete global trajectories across the distributed set-up. Figure 4.1 shows the flow diagram of the proposed approach for the reconstruction of global trajectories.

Let $C = \{c_1, c_2, ..., c_N\}$ be a set of N partially overlapping synchronised cameras (Fig. 4.2). Let $O_{i,n}$ represent the i^{th} object observed in c_n. We perform video object extraction (foreground segmentation) using a statistical colour change detector and then we associate them across consecutive frames using graph-matching [78]. Let $(x^t, y^t)_{i,n}$ be the resulting observation (track-point) of $O_{i,n}$ at instant t in camera c_n. The trajectory of $O_{i,n}$ is a set of all observations i.e.,

$$\mathbf{T}_{i,n} = \{(x^0, y^0), (x^1, y^1), ..., (x^J, y^J)\}, \tag{4.1}$$

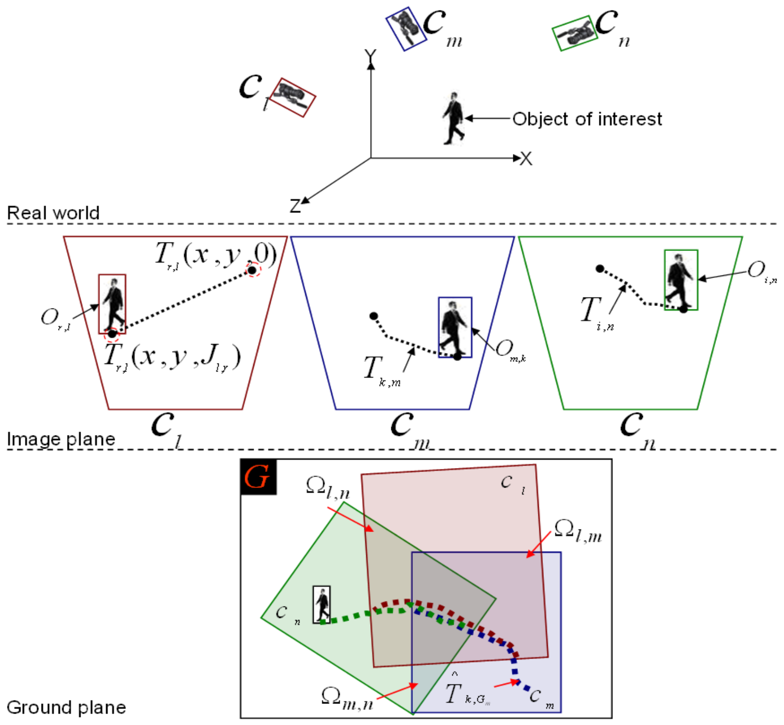


Figure 4.2: Illustration of notations. (Top) an object is observed by a network of cameras; (middle) trajectory segments of the object in image planes of each camera; and (bottom) projections of trajectory segments on a common ground plane.

where J represent the length of the trajectory.

We construct a virtual ground plane (G) from the available image plane views from multiple cameras and the image plane to ground plane projection is estimated by applying the homography matrix $\mathcal{H}_{G_n}$ [79] i.e.,

$$\hat{T}_{i,G_n}(\hat{x}^t, \hat{y}^t) = \mathcal{H}_{G_n}(x^t, y^t)_{i,n}, \tag{4.2}$$

where, $\hat{T}_{i,G_n}(\hat{x}^t, \hat{y}^t)$ is the local ground plane projection of $(x^t, y^t)_{i,n}$ and $\mathcal{H}_{G_n}$ is the homography matrix. $\mathcal{H}_{G_n}$ is constructed by manually selecting control points to establish the image and ground plane correspondence. However, these local projections result in differences in the overlapping region ($\Omega_{m,n}$) on the ground plane. Figure 4.3(top) shows a network of two partially overlapping cameras and accu-

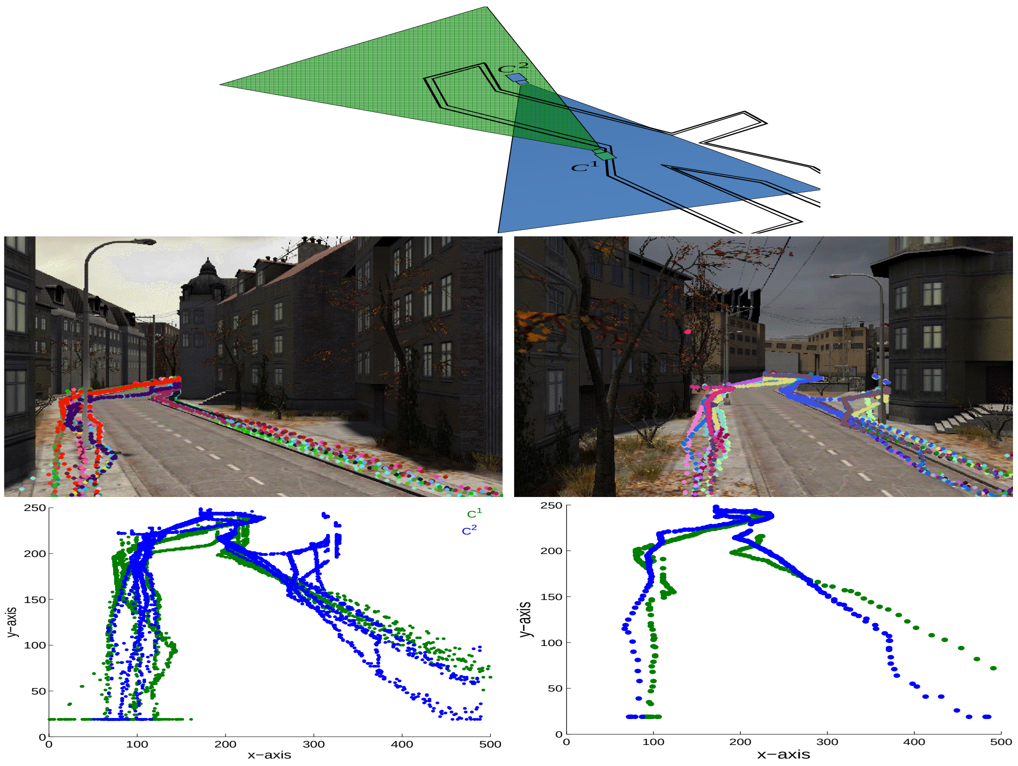


Figure 4.3: A network of partially overlapping cameras. (Top) configuration of the cameras; (middle-row) accumulated trajectories in each view; (bottom-left) ground plane projection from the two views and (bottom-right) an example of the differences on the ground plane.

mulated trajectories in each view (Fig. 4.3(middle-row)). Figure 4.3(bottom-left) shows the local projections of the trajectories on a common ground plane, where there are considerable differences of an object's trajectory viewed in two cameras (Fig. 4.3(bottom-right)). This leads to the requirement of a process which can establish a proximity matrix to associate every p^{th} trajectory to all q^{th} trajectories in $\Omega_{m,n}$. The final goal is to reconstruct a complete global trajectory of an object by fusing the trajectory segments across the entire environment.

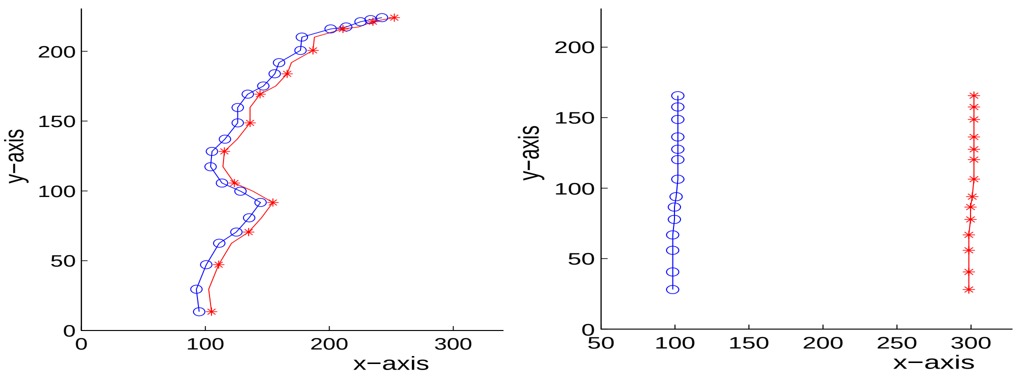


Figure 4.4: Illustration of feature's limitations. (Left) shape and length features unable to distinguish the trajectories belonging to different objects and (right) shape, length and average velocity features unable to differentiate two trajectories that are spatially far from each other.

4.2 Trajectory association

In the case of partially overlapping cameras, we need to establish the correspondence between transformed trajectory segments ($\hat{\mathbf{T}}_{i,G_n}$) in the overlapping regions on the ground plane. To find the relative pair-wise similarities for association, we use both spatial and temporal features extracted from the trajectory segments. We assume that a pair of trajectories from different cameras has to be similar both in time and space for the association and fusion. In [9], the shape (β_{i,G_n}) of the trajectories , which is approximated by polynomial coefficients of second order, and length ($\mathbf{d}_{i,G_n}$) are used to find the similarity. However, these features are not generalised enough to handle the variety of trajectories. Figure 4.4(left) shows an example, where two trajectories are considered as similar in shape using these features. In fact, the second object is moving twice the speed of the first one. We expand the feature set by including the average target velocity, $\overline{\mathbf{v}}_{i,G_n}$, which helps in describing the rate of change of the i^{th} object position and is calculated as

$$\overline{\mathbf{v}}_{i,G_n} = \frac{1}{J}\sum_{j=1}^{J-1}\left(\hat{x}^{j+1} - \hat{x}^{j}, \hat{y}^{j+1} - \hat{y}^{j}\right). \tag{4.3}$$

However, $\overline{\mathbf{v}}_{i,G_n}$ defines the average rate of change of an entire trajectory segment. For localising (time and position) the abrupt changes in a trajectory, we employ the sharpness of turns ($\hbar_{i,G_n}$), which defines the statistical directional characteristics of a trajectory and is calculated as

$$\hbar_{i,G_n} = H(\theta_{i,G_n}), \tag{4.4}$$

where $H(.)$ is a histogram function calculated over the directional angles ($\theta_{i,G_n} = tan^{-1}(\hat{y}^{j+1} - \hat{y}^{j}/\hat{x}^{j+1} - \hat{x}^{j})$. We take the indices to the top three peaks of $\hbar_{i,G_n}$ as they are sufficient to describe the dominant angles in the trajectory. Furthermore, we consider a situation where two trajectories with similar shape, length and velocity are present in completely different regions of the environment (see Fig. 4.4(right)). In order to distinguish them, trajectory mean ($\mathbf{m}_{i,G_n}$) is used and is defined as

$$\mathbf{m}_{i,G_n} = \frac{1}{J}\sum_{j=1}^{J}\left(\hat{x}^{j}, \hat{y}^{j}\right). \tag{4.5}$$

The features discussed so far define the overall pattern of a trajectory. In order to get the variation in observation information at the sample level, we include PCA components analysis ($\mathbf{p}_{i,G_n}$). We apply PCA on sample points of each trajectory by considering the covariance matrix as

$$\mathbf{\Xi}_{i,G_n} = \frac{1}{J}\widetilde{T}_{i,G_n}\widetilde{T}^{T}_{i,G_n}, \tag{4.6}$$

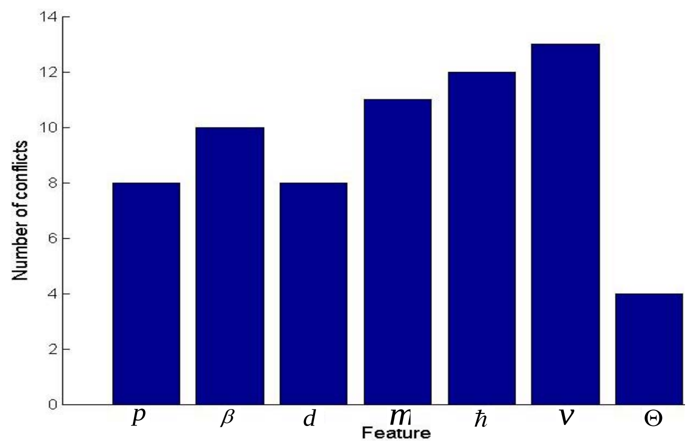


Figure 4.5: Number of conflicts in individual feature spaces.

where $\widetilde{T}_{i,G_n}$ is the mean-shifted version of T_{i,G_n}. The eigenvalue decomposition of Ξ_{i,G_n} results in eigenvalues, $\alpha = \{\alpha_j\}_{j=1}^{J}$, and corresponding eigenvectors, $\varphi = \{\varphi_\mathbf{j}\}_{j=1}^{J}$. After sorting α in descending order, we consider first two $\varphi_\mathbf{k}, \varphi_\mathbf{l} \in \varphi$, corresponding to the top two eigenvalues, $\alpha_k, \alpha_l \in \alpha$, as most of the variability in the track points lies in these two components. The final (normalised) feature vector for each trajectory is:

$$\mathbf{\Theta}_{i,G_n} = (\beta_{i,G_n}, \mathbf{d}_{i,G_n}, \overline{\mathbf{v}}_{i,G_n}, \hbar_{i,G_n}, \mathbf{m}_{i,G_n}, \mathbf{p}_{i,G_n})^T, \tag{4.7}$$

where T denotes the transpose operator. Because of its robustness to the scale variation, we use cross correlation as a *proximity measure*. For $\mathbf{T}'_{i,G_n}$ and $\mathbf{T}'_{k,G_m}$ in $\Omega_{n,m}$ the association matrix is calculated as:

$$A_\Omega(\hat{T}_{i,G_n}, \hat{T}_{k,G_m}) = \mathcal{F}(\mathbf{\Theta}_{i,G_n}, \mathbf{\Theta}_{k,G_m}), \tag{4.8}$$

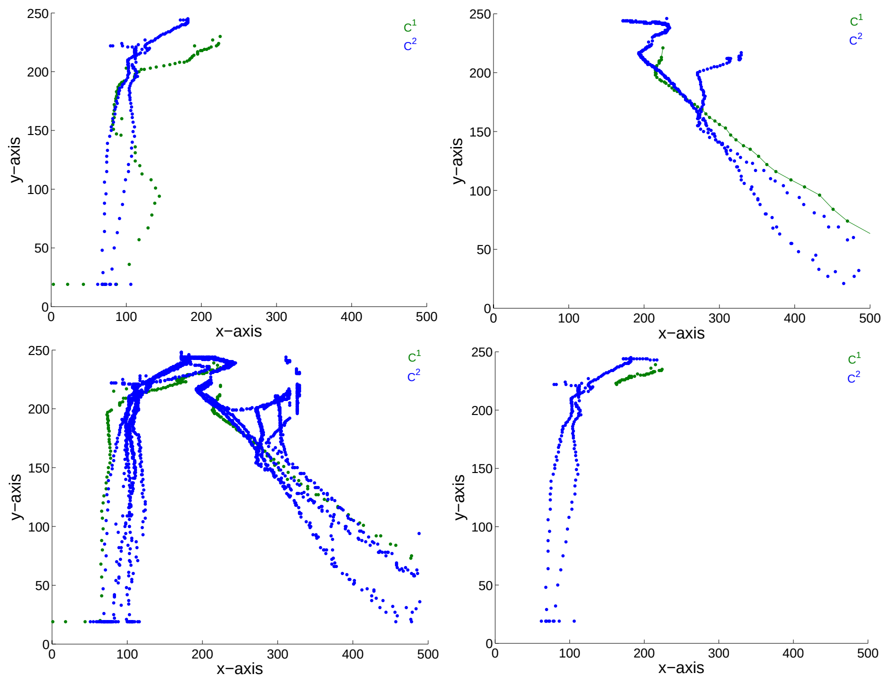


Figure 4.6: Examples of conflicting situations in (top-left) p, (top-right) m, (bottom-left) combined v and $\hbar$ and (bottom-right) d.

where, $\mathcal{F}$ is the correlation function. A trajectory $\hat{\mathbf{T}}_{i,G_n}$ will be associated to any trajectory $\hat{\mathbf{T}}_{k,G_m}$ for which it has maximum correlation i.e.,

$$\mathcal{D}_\Omega = \arg\max_r (A_\Omega(\hat{T}_{k,G_l}, \hat{T}_{r,G_m})) \; \forall \, O_{r,m} \in c_m. \tag{4.9}$$

It is noticed that individual features result in spurious associations as shown in Figure 4.5. There are number of conflict situations (examples are shown in Fig. 4.6), however, the use of combined feature set reduces this number considerably. However, still there are cases where multiple trajectories can correspond to a trajectory. Figure 4.7 demonstrates such an example, where two out of three trajectories match the input trajectory. In order to have single trajectory segment belong to a physical object in the overlapping region, we need to resolve this conflict situation. For this, we perform matching in image plane by reprojecting the matched trajectories from

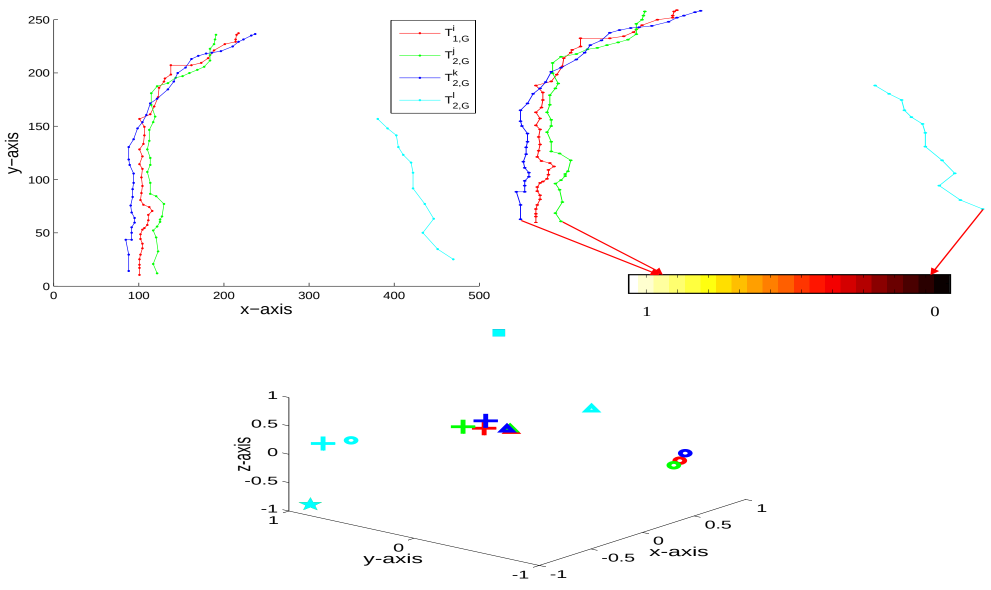


Figure 4.7: An example of association conflict. (Top-left) four sample trajectories (see Fig. 4.3); (top-right) association results with two trajectories (blue and green) have equal scores; (bottom) representation of trajectories in each feature space with a marker colour maps to trajectory colour (key; star:$\overline{\mathbf{v}}$; cross:$\mathbf{m}$; triangle: β; circle: $\mathbf{p}$; square: $\hbar$); particular to $\overline{\mathbf{v}}$ and $\hbar$ all trajectories coincide.

the ground plane. Suppose, trajectory segment $\hat{\mathbf{T}}_{i,G_n}$ can be associated to $\hat{\mathbf{T}}_{k,G_m}$ and $\hat{\mathbf{T}}_{s,G_m}$ (or even more), we reproject the trajectories onto the image plane using $\mathcal{H}^{-1}_{G_n}$. Resampling is done in order to have equal length trajectories and then standard Euclidean distance (d) is employed as proximity measure. The trajectory is selected for which the distance is minimum i.e.,

$$K = \arg\min_{l}(d(\hat{T}_{i,G_n}, \hat{T}_{l,G_m}))\,\forall\, l = 1, ..., L, \tag{4.10}$$

where L is the total number of matched trajectories on the ground plane.

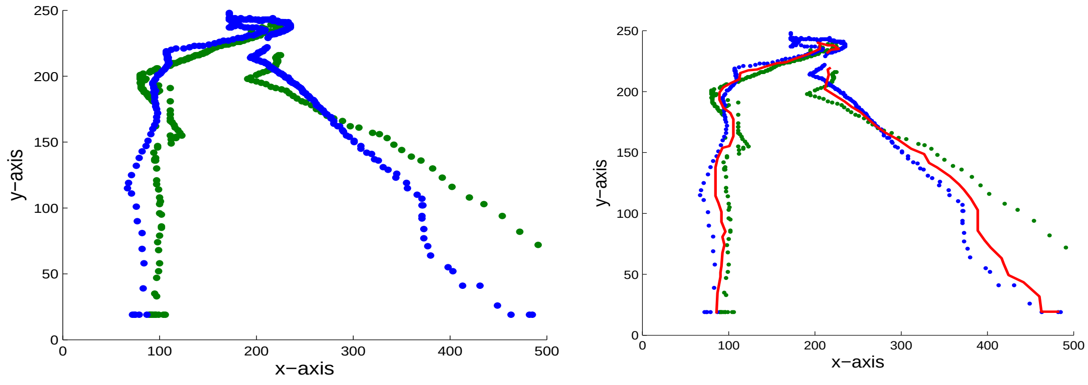


Figure 4.8: An example of trajectory fusion. (Left) matched trajectories from a pair of overlapping cameras (Fig. 4.3) and (right) Fusion result.

4.3 Trajectory fusion

Once association is done, the next step is to fuse a pair of corresponding trajectories in overlapping regions. The fusion is required to combine the trajectory segments observed in multiple cameras . To fuse $\hat{\mathbf{T}}_{i,G_n}$ and $\hat{\mathbf{T}}_{k,G_m}$, where both trajectories are generated from the same object in real world, we use an adaptive weighting method i.e.,

$$\hat{T}^t_{i,k,G_{n,m}} = \begin{cases} w_1\hat{T}^t_{i,G_n} + w_2\hat{T}^t_{k,G_m} & in\, R_{n,m} \\ \hat{T}^t_{i,G_n} & in\, R_n \\ \hat{T}^t_{k,G_m} & in\, R_m, \end{cases} \tag{4.11}$$

where $R_{n,m}$ is the region where observations from both $\hat{T}_{i,G_n}$ and $\hat{T}_{k,G_m}$ are available at t. R_n and R_m are the regions where the observation is available from either $\hat{T}_{i,G_n}$ or $\hat{T}_{k,G_m}$, respectively. At each time t when the observation from both trajectories are available, the trajectory segment which has more track points is given higher weight than the other; otherwise, we utilise the available observation from one of the trajectories. The weights $(w_i : i = 1, 2)$ are calculated as function of number of

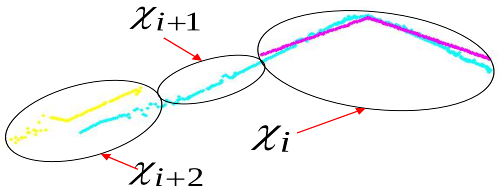


Figure 4.9: An example of trajectory linkage.

observations for each trajectory:

$$w_1 = \frac{|\hat{T}_{i,G_n}|}{|\hat{T}_{i,G_n}| + |\hat{T}_{k,G_m}|}, w_2 = \frac{|\hat{T}_{k,G_m}|}{|\hat{T}_{i,G_n}| + |\hat{T}_{k,G_m}|}, \tag{4.12}$$

where $|.|$ is the number of observations in a trajectory and w_1+w_2=1. In order to have smoother overall trajectory to avoid small fluctuations due to the observation's gaps, we apply a moving average approach where window size is set to 5 observations.

Finally, to construct a complete trajectory across the entire environment, we connect all segments that belong to same object (see Fig. 4.9) i.e.,

$$T_{i,G} = \bigcup_{i=1}^{\kappa} \chi_i, \tag{4.13}$$

where κ is total number of connected regions. Also, χ_i is the segment observed in overlapping region between c_n and c_m and χ_{i+1} is the segment observed in non-overlapping region (i.e. only in C^m) and χ_{i+2} is the segment observed in overlapping region between c_m and c_l. In this way, a complete trajectory is constructed for cameras c_l, c_m and c_n, whereby c_l and c_n are non-overlapping by configuration.

4.4 Summary

In this chapter, we have addressed the problem of trajectory association across camera network, without imposing constraints on the camera placement. Local trajectory segments from each camera are projected on a common ground plane. Multiple spatio-temporal features are then analysed to find the degree of proximity between the trajectories. The matching is verified via the ground plane to image plane reprojections. The proposed approach generates a complete trajectory belonging to a physical object in an unsupervised way. The performance of the proposed approach is evaluated in Ch. 6 and compared with existing works. Furthermore, these trajectories are important elements for the analysis and the representation of behaviours ([80], [81]). In the next chapter, we present an algorithm for video analysis using trajectory clustering.

Chapter 5

Trajectory clustering

5.1 Introduction

Clustering is a key component of trajectory analysis when instead of modelling and analysing the motion of an individual object, multiple trajectories are processed together to discover the inherent structures of activities in a video. This process aims to classify trajectories into two major classes, namely normal trajectories, which belong to common patterns, and outliers, which exhibit a deviant behaviour. Clustering groups unlabeled data in such a way that elements in a single cluster have similar characteristics, and elements in different clusters have the most dissimilar characteristics ([82], [83], [84]).

In this chapter, we propose a partitional trajectory clustering framework that combines internally a fuzzy clustering approach based on multiple features before generating a final crisp partition. We use multiple feature spaces simultaneously to obtain a higher degree of descriptiveness of the trajectories as opposed to using one feature space only. Each feature space is then regarded as the empirical probability density function (*pdf*) of the represented parameter and modes in each space correspond to the maxima of the *pdf.* Once the modes are determined, the members can be associated to each mode to form the clusters. We use Mean-shift in each feature space for mode-seeking and clustering. The clustering results of Mean-shift in each space are then refined by applying a cluster merging procedure. The final clustering is obtained by analysing the clustering results from each feature space. The process results in the definition of the clusters' structures along with the fuzzy membership of a trajectory to the final clusters. The clusters with small number of associated elements and the trajectories that are far from the clusters' centre are considered as outliers. As a consequence the algorithm may consider a normal trajectory, within a cluster, as an outlier; but it makes sure to reduce the chance of considering an outlier as a normal trajectory, which is desirable especially in surveillance applications. Figure 5.1 shows the flow diagram of the proposed approach. The details of the proposed approach are provided in following sections.

5.2 Feature extraction

We propose a multi-feature trajectory clustering algorithm that improves the overall clustering performance by exploiting the descriptiveness of multiple feature spaces. Let a trajectory T_j be represented as $T_j = \{(x_j^i, y_j^i); i = 1, \ldots, N_j\}$, where (x_j^i, y_j^i) is the estimated position (mid-point of bottom of the bounding box) of the j^{th} target on the image plane, N_j is the number of trajectory points and $j = 1, ..., J$. J is number of trajectories. Note that the trajectories are likely to have different length; therefore, we extract features from each input trajectory to have equal lengths (dimensions). Let $F_m(.)$ be a feature extraction function defined as $F_m(.) : T_j \rightarrow \Psi_m^{d_m}$, with $m = 1, ..., M$, where M is total number of feature spaces. F_m

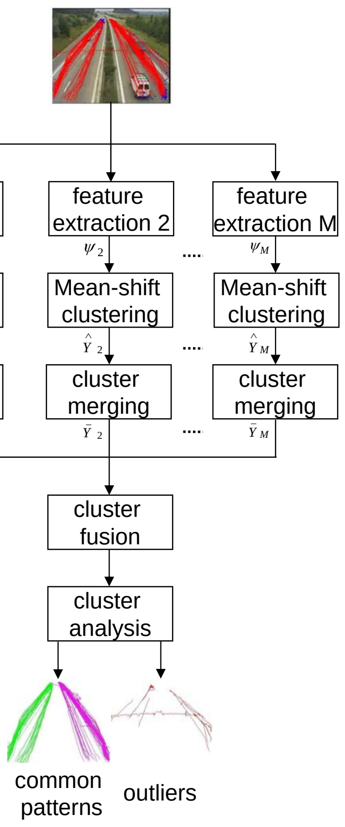


Figure 5.1: Flow diagram of the trajectory clustering algorithm.

map every trajectory T_j to a d_m-dimensional feature space $\{\Psi_m^{d_m}\}_{m=1}^{M}$. The feature spaces are treated independently in order to avoid the need for normalisation, which is required if features are processed together, and to help in analysing multi-domain (spatial and angular) non-orthogonal feature spaces together. Moreover, it also provides a framework for parallel clustering using different features and integrates them together to avoid problems comparing non-similar features.

Feature selection depends upon the application and each feature space helps in finding the coarse structures from the input data, which are then integrated for fine-grained clustering. For this purpose, we investigate spatial and angular trajectory representations, namely (a) the average target velocity, (b) the directional distance, (c) the target trajectory mean, (d) the combination of the initial target position, its

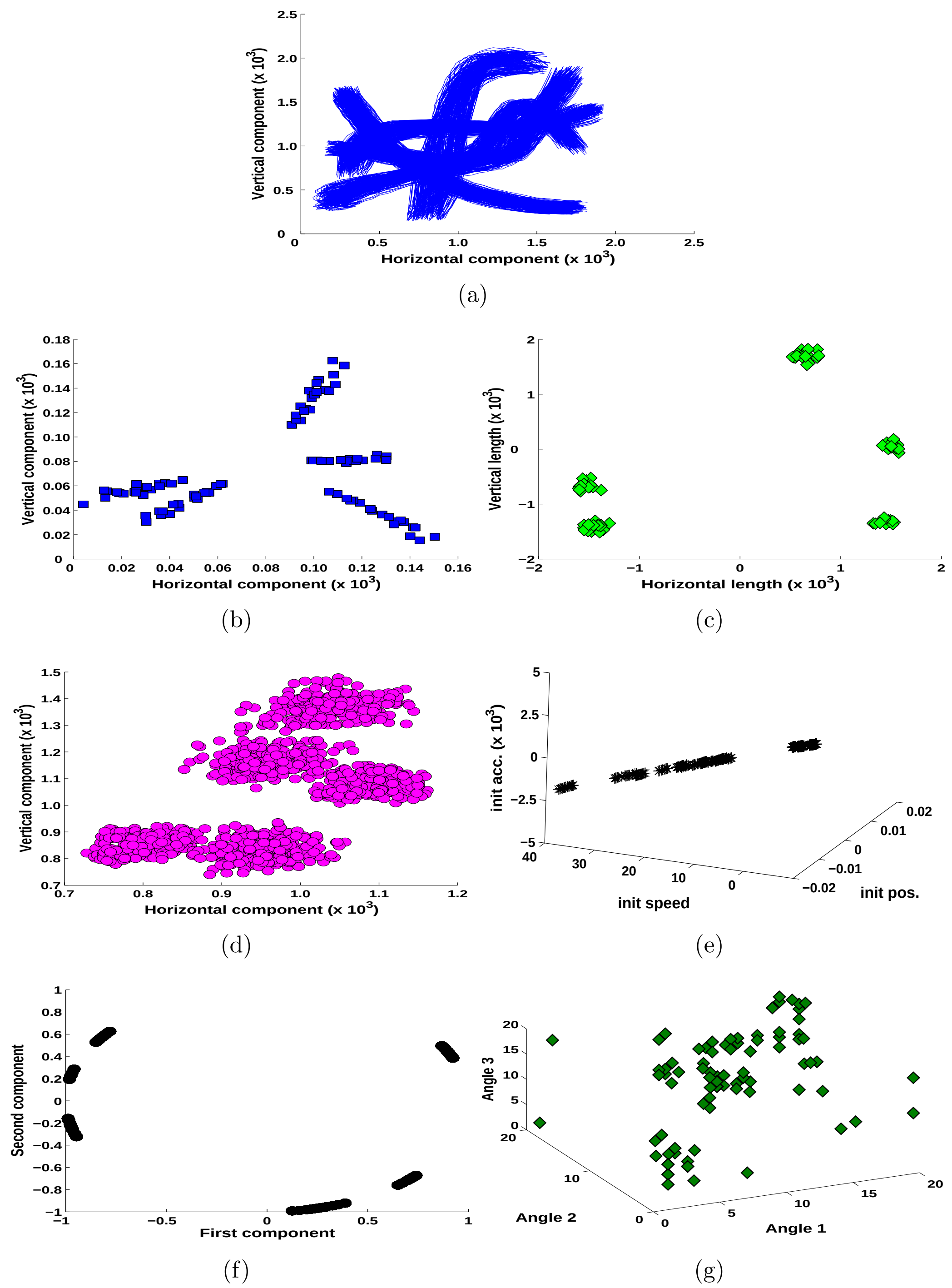


Figure 5.2: (a) Sample set of 1100 synthetic trajectories and their projections on the following feature spaces: (b) average velocity, (c) directional distance, (d) trajectory mean, (e) combination of initial position, speed and acceleration, (f) principal components and (g) trajectory turns (three dominant angles).

speed and its acceleration, (e) the PCA of the trajectory points, and (f) trajectory turns[1]. Next, we provide details of each feature space.

The average target velocity, $\overline{\mathbf{v}}_\mathbf{j}$, describes the rate of change of the j^{th} object position. This feature helps separating the trajectories of objects moving at varying pace. $\overline{\mathbf{v}}_\mathbf{j}$ is defined as

$$\overline{\mathbf{v}}_j = \frac{1}{N_j - 1} \sum_{i=1}^{N_j - 1} \left(x_j^{i+1} - x_j^i, y_j^{i+1} - y_j^i\right). \tag{5.1}$$

The directional distance, $\mathbf{d_j}$, of the j^{th} object is considered as the second feature to extract the horizontal and vertical length of a trajectory. Moreover $\mathbf{d_j}$ also encodes the direction of motion (moving toward or away from the camera). This feature helps distinguishing longer trajectories from shorter ones and also trajectories in opposite directions. $\mathbf{d_j}$ is calculated as

$$\mathbf{d_j} = \left(x_j^{N_j} - x_j^0, y_j^{N_j} - y_j^0\right). \tag{5.2}$$

The third spatial feature encodes the horizontal and vertical components of j^{th} trajectory mean ($\mathbf{m_j}$). This feature works well to distinguish the trajectories belonging to different regions on the image plane and is calculated as

$$\mathbf{m_j} = \frac{1}{N_j} \sum_{i=1}^{N_j} \left(x_j^i, y_j^i\right). \tag{5.3}$$

In order to model the shape of the j^{th} trajectory irrespective of its length and sample points we use polynomial regression as fourth feature space. The matrix notation

[1]Note that the *trajectory turns feature is used to analyse the degree of turns in a particular trajectory. This is different from the* acceleration*, which is the change of velocity over time*

for the model estimation of T_j is written as

$$\mathbf{y_j}^{'} = \begin{pmatrix} 1 & \mathbf{x_j} & (\mathbf{x_j})^2 & ... & (\mathbf{x_j})^{\rho} \end{pmatrix} (\beta_0 \beta_1 ... \beta_{\rho})^T \; + \epsilon, \tag{5.4}$$

where the first term of the R.H.S is a $N_j \times \rho$ matrix with $\mathbf{x_j} = \{x_j^i\}_{i=1}^{N_j}$, the second term is $\rho \times 1$ vector and the last term is a N_jx1 vector. The output vector is also N_jx1 vector. The goal here is to find the optimal values of β_i for which $\epsilon = |\mathbf{y}^{'} - \mathbf{y}|$ becomes minimum. The process requires an inherent trade-off between accuracy and efficiency. As the degree of the polynomial increases, the fit grows in accuracy but only up to a point. We find the appropriate degree by starting with a first degree polynomial and continually monitoring the fit to see if the degree needs to be increased. If so, the regression is restarted with the degree incremented by one. Here we have fixed $\rho = 2$ as an increase of the value of ρ does not affect the overall accuracy. The three coefficients $(\beta_0, \beta_1, \beta_2)$ maps to the initial position, speed and acceleration of the object.

In order to consider the variation information of each trajectory, we apply PCA on sample points of each trajectory. For simplicity of notation, let $\mathbf{x_j} = (x_j, y_j)$, then T_j can be rewritten as $T_j = \{\mathbf{x_j^i}\}_{\mathbf{i=1}}^{\mathbf{N_j}}$. After subtracting the trajectory mean ($\mathbf{m_j}$) from each trajectory point,

$$\widetilde{T}_j = \{\mathbf{x_j^i} - \mathbf{m_j}; i = 1, ..., N_j\}, \tag{5.5}$$

we consider the covariance matrix as

$$\mathbf{\Xi_j} = \frac{1}{N_j} \widetilde{T}_j \widetilde{T}_j^T. \tag{5.6}$$

The eigenvalue decomposition of Ξ_j results in eigenvalues, $\alpha = \{\alpha_i\}_{i=1}^{N_j}$, and corresponding eigenvectors, $\varphi = \{\varphi_{\mathbf{i}}\}_{i=1}^{N_j}$. After sorting α in descending order, we consider the first two $\varphi_{\mathbf{k}}, \varphi_{\mathbf{l}} \in \varphi$, corresponding to the top two eigenvalues, $\alpha_k, \alpha_l \in \alpha$, as most of the variability of the data lies in these two components.

Lastly, to consider the sharpness of turns in T_j, the directional histograms ($\hbar_j$) are calculated using the method presented in [16] as

$$\hbar_{\mathbf{j}} = \mathcal{H}(\theta_{\mathbf{j}}^{\mathbf{i}}), \tag{5.7}$$

where $\mathcal{H}(.)$ is a histogram function calculated over the directional angles ($\theta_{\mathbf{j}}^{\mathbf{i}} = tan^{-1}(y_j^{i+1} - y_j^i / x_j^{i+1} - x_j^i)$). We take the indices of the top three peaks of $\hbar_{\mathbf{j}}$ as they describe the dominant angles in the trajectory. To show relative placements of trajectories in a particular feature space, we have constructed a synthetic dataset consisting of 1100 trajectories as shown in Fig. 5.2(a). The dataset contains five normal patterns with regions where trajectories from different patterns intersect each other to make the dataset challenging. Figure 5.2(b-g) shows the relative placement of each trajectory of a given dataset in each feature space.

Once the trajectories are transformed into multiple feature spaces, the next step is to analyse each feature space to form clusters. Without prior knowledge on the type of target we are observing, we consider all the features equally important and give them equal weights for the final clustering. The detailed discussion about the clustering process is given in the next section.

5.3 Clustering

Each feature space is regarded as the empirical probability density function (*pdf*) of the distribution of the trajectories in a particular feature space [85]. To develop a more generic solution we use Mean-shift, which is an unsupervised and non-parametric clustering algorithm. We apply Mean-shift on the normalised feature spaces to find the modes of the *pdf* and then we associate each trajectory with the nearest mode to form the clusters.

Mean-shift is a clustering technique that climbs the gradient of a probability distribution to find the nearest dominant mode or peak ([86]). Let $\chi_l \in \Psi_j^{d_j}$; $l = 1, ..., L$ be a set of L data points. The multivariate density estimator $\hat{f}(x)$ is defined as

$$\hat{\mathbf{f}}(\mathbf{x}) = \frac{1}{L h_m^{d_m}} \sum_{l=1}^{L} \mathcal{K}\left(\frac{x - \chi_l}{h_m^{d_m}}\right), \tag{5.8}$$

where $h_m^{d_m}$ is the bandwidth of the kernel, $\mathcal{K}(.)$. The choice of $h_m^{d_m}$ plays an important role in Mean-shift clustering. We employ an incremental procedure to select this value. Initially, $h_m^{d_m}$ is set to 10% of each dimension of m^{th} feature space and it iteratively increases to 80%. The lower bound prevents clusters containing a single trajectory, while the upper bound avoids a cluster with all trajectories grouped together. Although a smaller $h_m^{d_m}$ produces less biased density estimator, it increases the variance. In order to find the compromise between the two quantities we used Mean Integrated Squared Error (MISE) $\mathcal{E}$ [86], defined as

$$\mathcal{E}(x) = \int E((\chi_l - \hat{\mathbf{f}}(\mathbf{x}))^2) dx, \tag{5.9}$$

The value of $h_m^{d_m}$ for which $\mathcal{E}(x)$ is minimum is considered to be the optimal one.

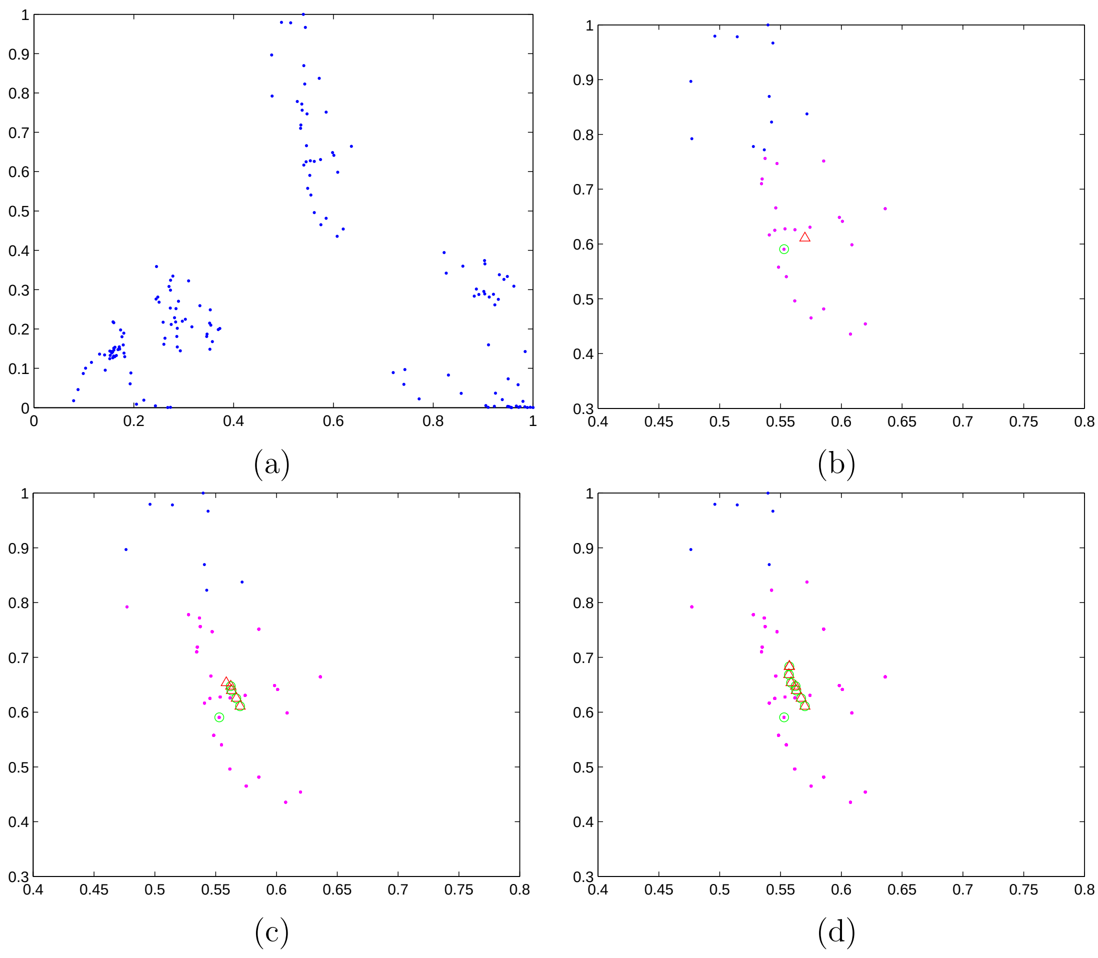


Figure 5.3: Sample Mean-shift clustering results on the first two principal components of the highway traffic sequence $S3$. Each point represents a trajectory. (a) Initial trajectory representation. Results (zoom) of the (b) 1st, (c) 5th, and (d) 8th iteration of the mode seeking procedure. (Key. Blue: unprocessed points. Magenta: points within the kernel bandwidth. Green: mode-seeking path. Triangles: mode at each iteration).

Moreover, $\mathcal{K}(.)$ in Eq. (5.8) is defined as

$$\mathcal{K}(x) = \begin{cases} \frac{1}{2V_m^{d_m}}(d_m+2)(1-x^T x) & \text{if } x^T x < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

where $V_m^{d_m}$ represents the volume of a d_m-*dimensional* unitary sphere. The density gradient estimate of the kernel can be written as

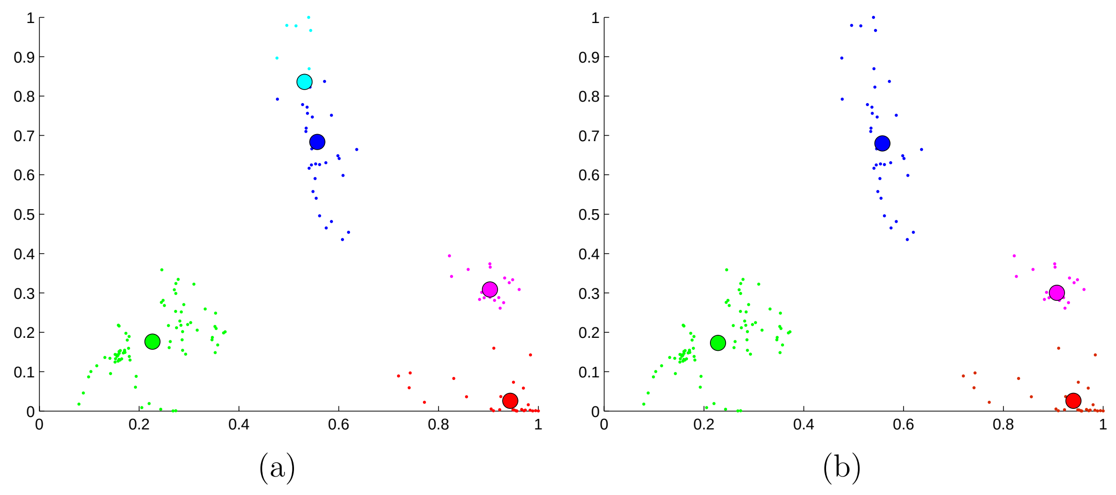


Figure 5.4: Sample cluster merging results. (a) Initial trajectory clustering result (5 clusters) before cluster merging; (b) final clustering result after cluster merging (4 clusters).

$$\hat{\nabla}\mathbf{f}(\mathbf{x}) = \nabla\hat{\mathbf{f}}(\mathbf{x}) = \frac{1}{Lh_m^{d_m}} \sum_{l=1}^{L} \nabla\mathcal{K}\left(\frac{x-\chi_l}{h_m^{d_m}}\right). \tag{5.11}$$

Equation (5.11) can be re-written as

$$\hat{\nabla}\mathbf{f}(\mathbf{x}) = \frac{d+2}{h_m^{d_m} V_m^{d_m}} \left(\frac{1}{L_c} \sum_{\chi_l \in S(x)} (\chi_l - x) \right), \tag{5.12}$$

where *S(x)* is a hypersphere of radius $h_m^{d_m}$, with volume $h_m^{d_m} V_m^{d_m}$, centred in x and containing L_c data points. The Mean-shift vector, $\zeta_{\mathbf{h}}(\mathbf{x})$, is defined as

$$\zeta_{\mathbf{h}}(\mathbf{x}) = \frac{1}{L_c} \sum_{\chi_l \in S(x)} (\chi_l - x), \tag{5.13}$$

and, using Eq. (5.12), we can express $\zeta_{\mathbf{h}}(\mathbf{x})$ as

$$\zeta_{\mathbf{h}}(\mathbf{x}) = \frac{h_m^{d_m} V_m^{d_m}}{d_m + 2} \frac{\hat{\nabla}\mathbf{f}(\mathbf{x})}{\hat{\mathbf{f}}(\mathbf{x})}. \tag{5.14}$$

The output of the Mean-shift procedure is the set of data points associated to each

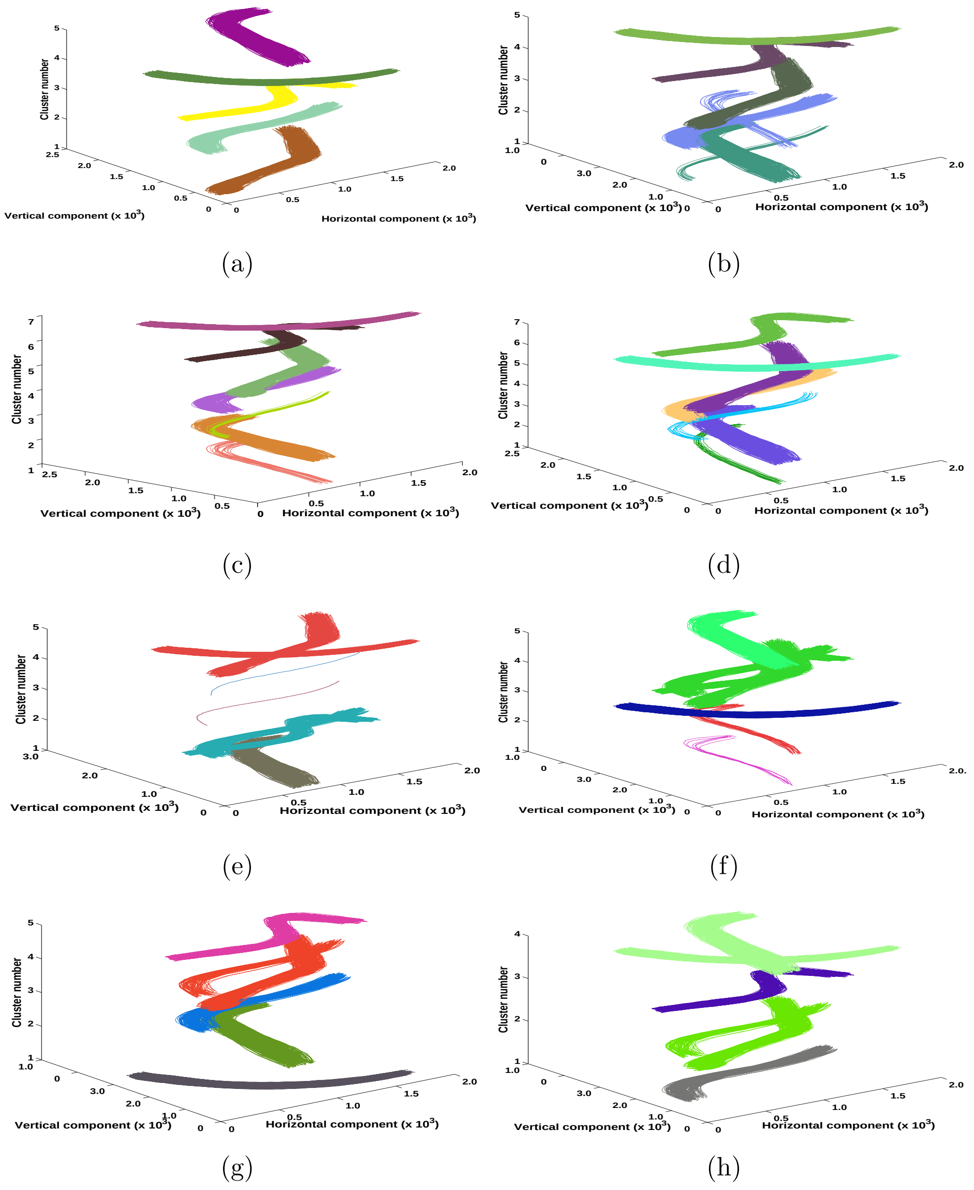


Figure 5.5: Comparison of single and multiple feature clustering results corresponding to Fig. 5.2: (a) independent multiple features (proposed), (b) concatenated multiple feature, (c) average velocity only, (d) directional distance only, (e) trajectory mean only, (f) combination of initial position, speed, acceleration only, (g) principal components only and (h) trajectory turns (the three dominant angles) only.

mode. This process is illustrated in Fig. 5.3. Initially, the mode seeking process starts by fixing a trajectory as a seed point; then after the Mean-shift process converges to the local mode, all the trajectories within the bandwidth, $h_m^{d_m}$, of the kernel, $\mathcal{K}(.)$, are assigned to that mode [87]. These trajectories are then not considered for future iterations. The next seed point is selected randomly from the unprocessed trajectories. The process terminates when all trajectories are assigned to a corresponding local mode.

A small bandwidth causes an increase in the number of modes and a larger variance, which results in unstable variations of local density. This artefact can be eliminated by merging the closely located modes [88]. In this work, we merge adjacent clusters if their modes are apart by less than $h_m^{d_m} + 0.1(h_m^{d_m})$. A sample result of cluster merging is shown in Fig. 5.4, where two modes in Fig. 5.4(a) (upper region: dark and light blue) are located within the threshold value and are merged as shown in Fig. 5.4(b). The cluster merging threshold is selected empirically; a higher threshold may result in inappropriately large clusters.

5.4 Post processing

The final partitioning of the trajectories is obtained after analysing the refined clustering results from each feature space. The integration of the clusters consists of three steps, namely the estimation of the final number of clusters, the establishment of the correspondence between clusters in different feature spaces, and the association of each trajectory to a final cluster.

Let $\xi = \{\aleph_k\}_{k=1}^{M}$ be the set containing the number of clusters for each feature space $\Psi_k^{d_k}$. The final number of clusters $\aleph$ is selected as the median value of the set ξ. After selecting the final number of clusters, we construct the final clusters by taking the initial clustering results from each feature space. We estimate the structure of clusters as characterised by a single mode; we model each cluster with a univariate Gaussian with bandwidth of the kernel defining the variance of the cluster itself.

Algorithm 2 Generalised cluster fusion process.

```
$\xi = \{\aleph_1,\aleph_2,...,\aleph_M\}$: number of clusters for each feature space
$C^i_j$: $j^{th}$ cluster in the $i^{th}$ space;
$\aleph$: number of final clusters;
 1: Compute: $\aleph$
 2:     $\aleph = \text{median}(\xi)$
 3: Compute: $C^f$
 4:     l $\leftarrow$ find $\xi = \aleph$
 5: $C^f_i = C^l_i : i = 1,...,\aleph$
 6: for $n = 1$ to $M$ do
 7:     if $n \neq l$
 8:     for $i = 1$ to $\aleph$ do
 9:         find $\hat{\nu} = \arg\max_j |C^l_i \cap \{C^n_j\}^{\xi_n}_{j=1}|$
10:         $C^f_i = (C^f_i \cap C^n_{\hat{\nu}})$
11:     end for
12:     end if
13: end for
```

In order to find the structure of each cluster, we start the process with a feature space $\Psi^{d_l}_l \in \{\Psi^{d_k}_k\}^M_{k=1}$ such that $\xi_l = \aleph$. The initial parameters of the final clusters ($C^f_i; i = 1, ..., \aleph$) are those defined by $\Psi^{d_l}_l$. To refine the parameters according to the results of the other feature spaces, we find the correspondence of $\Psi^{d_l}_l$ with all the other feature spaces $\Psi^{d_n}_n$ with $\Psi^{d_n}_n \in \{\Psi^{d_k}_k\}^M_{k=1}$ and $n \neq l$.

Let $\hat{\nu}$ be the index of the cluster in $\Psi^{d_n}_n$ that has the maximum correspondence (maximum number of overlapping elements) with the i^{th} cluster of $\Psi^{d_l}_l$:

$$\hat{\nu} = \arg\max_j \left|C^l_i \cap C^n_j\right|, \tag{5.15}$$

where $i = 1, ..., \aleph$ and $j = 1, ..., \aleph_n$. Also, C^l_i and C^n_j represent the i^{th} and j^{th} clusters in $\Psi^{d_l}_l$ and $\Psi^{d_n}_n$, respectively. Then C^f_i is updated by taking the overlapping elements, $C^f_i = (C^f_i \cap C^n_{\hat{\nu}})$. This process continues for all features spaces (see Algo. 2). This results in $\aleph$ clusters consisting of all the trajectories that are consistent i.e., grouped together, across all feature spaces, and are therefore considered to represent a reliable structure for each cluster. At this point we associate to the final clusters the trajectories ($T^{'} \subseteq \{T_j\}_{j=1}^{J}$) which are not consistent across all the feature spaces. To this end, we calculate a conditional probability of $T_o \in T^{'}$ generated from the given cluster model as

$$p(T_o|C_k^f) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\sqrt{2\pi}\sigma_{f,k}}e^{-(\frac{\mu_{i,j}-\mu_{f,k}}{\sigma_{f,k}})^2}, \tag{5.16}$$

where $\mu_{i,j} = \frac{1}{|C_j^i|}\sum_{j=1}^{|C_j^i|}\mathbf{m_j}$ and $\mu_{f,k} = \frac{1}{|C_k^f|}\sum_{k=1}^{|C_k^f|}\mathbf{m_k}$ (see Eq. (5.3)) are the mean values of C_j^i and C_k^f respectively; $\sigma_{f,k} = \frac{1}{|C_k^f|}\sum_{k=1}^{|C_k^f|}\mathbf{\Xi_k}$ (see Eq. (5.6)) is the standard deviation of C_k^f. The motivation behind these parameters is that each cluster is characterised by a single mode with the spread of the cluster can be represented by σ. Note that the decision is made at the cluster level and not at the feature space level, thus removing the dependence on dimensionality or normalisation. T_o will be assigned to C_k^f if

$$p(T_o|C_k^f) > p(T_o|C_l^f), \tag{5.17}$$

where l=1, ..., $\aleph$ and $l \neq k$. Figure 5.5 shows a comparison of trajectory clustering results using single and multiple (concatenated and independent) features. Single features (Fig. 5.5(c-h)) are not always capable of providing an effective data representation as, for example, the average velocity encodes directional information only, without providing any spatial relationship information among the trajectories. This results in grouping trajectories even if they are located far from each other. On the other hand, features such as the directional distance and trajectory mean contain spatial information only, without encoding any variation information. For this reason, these features do not generate clusters of trajectories with similar motion patterns that are not spatially close. The integration of the different features and their properties improves the overall trajectory clustering results by generating a more meaningful grouping, as shown in Fig. 5.5(b). However, in Fig. 5.5(b) there are still 33 wrongly clustered trajectories. The proposed approach further improves the performance by first post-processing the results (cluster merging) at feature space level and then by fusing the post-processed results at later stage (Fig. 5.5(a)).

5.5 Outlier detection

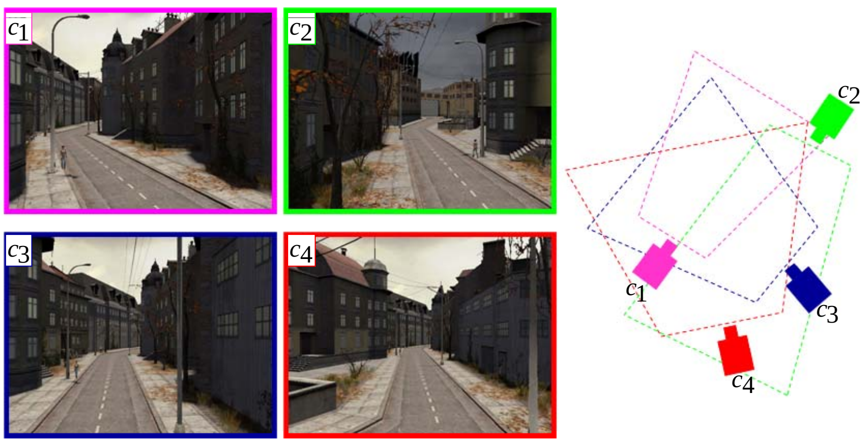
An outlier trajectory is the one that deviates from other trajectories and can correspond to an abnormal behaviour e.g., driver being on the wrong side of the road. In this work we focus on identifying two types of outlier trajectories: those existing in dense regions but exhibiting a different behaviour from the common pattern; and those located in sparse regions.

To detect the first type of outliers, we use a distance-based approach. If a trajectory $T_j' \in C_k^f$, with trajectory mean $\mu_{T_j'}$, lies far from the centre ($\mu_{f,k}$) of the cluster it belongs to, then it is considered as outlier, i.e.,

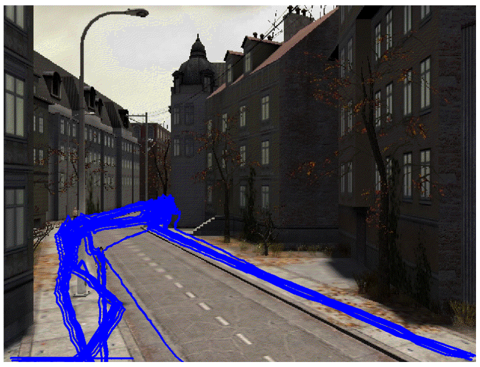
$$\frac{|\mu_{T_j'} - \mu_{f,k}|}{\sigma_{f,k}} > \tau, \tag{5.18}$$

where $\tau = 0.95$ (empirically selected).

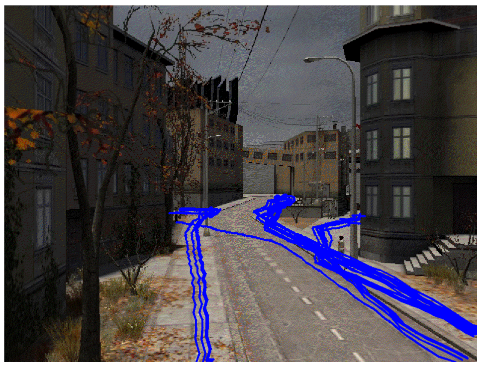
To detect the second type of outliers, we identify trajectories belonging to sparse regions by considering the size of their cluster. If a cluster has few associated trajectories and cannot be merged with a nearby cluster, then it is considered to be



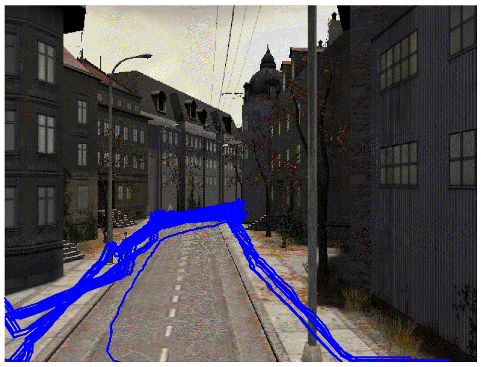
(a)



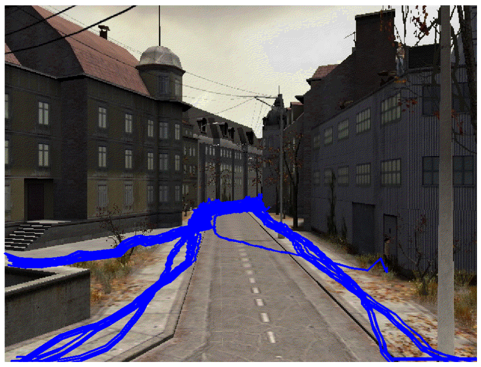
(b)



(c)



(d)



(e)

Figure 5.6: An example of multi-camera setup for trajectory clustering. (a) Camera configuration, (b-e) input trajectories on top of key frame of each view.

a set of outliers. Here the threshold value is set to the 10% of the cardinality of the cluster containing the median number of associated elements.

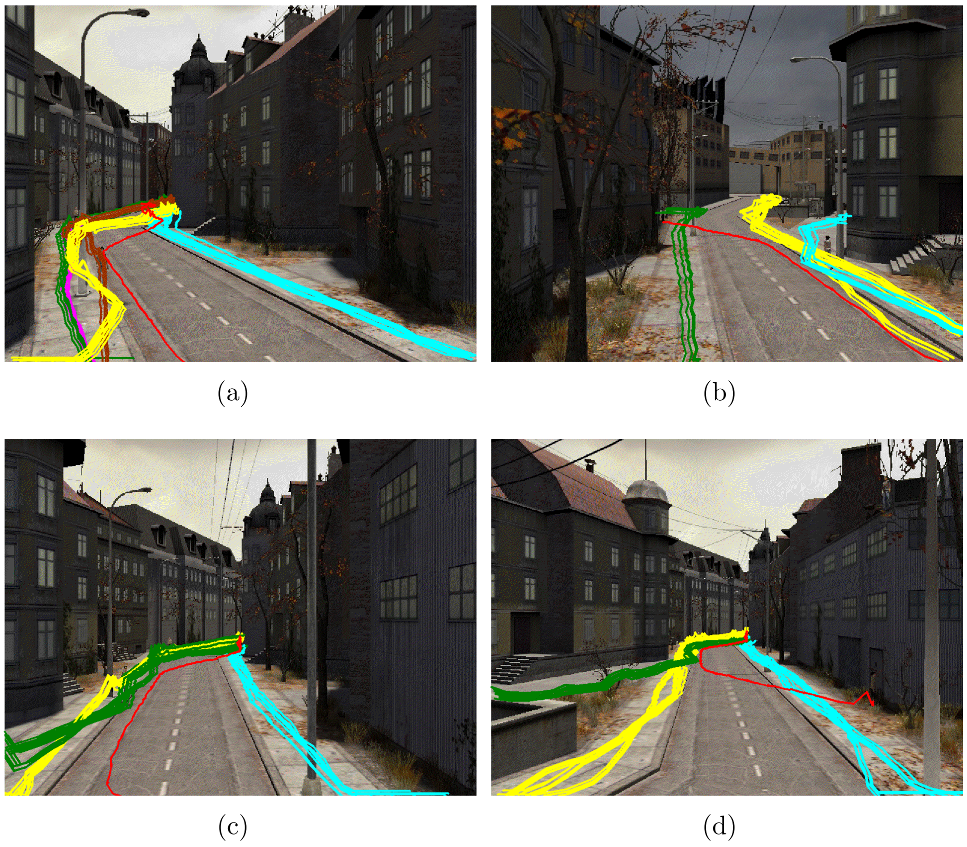


Figure 5.7: Clustering results on each image plane.

5.6 Example of trajectory clustering

To analyse the performance of the clustering algorithm in a multi-camera setup, we consider a simulated camera network that is consisting of four cameras as shown in Fig. 5.6(a). The trajectories from each camera are shown on a key-frame of each view in Fig. 5.6(b-e). We prepared a ground truth for this sequence manually. The ground truth of the sequence is as follows: there are four clusters on left-path (c_1, c_3 and c_4) and one cluster on right-path (c_1, c_3 and c_4) and one trajectory (a person crossing the road) is an outlier.

Initially, we applied clustering on trajectories from each camera independently. The results are shown in Figure 5.7. It can be observed that perspective deformations

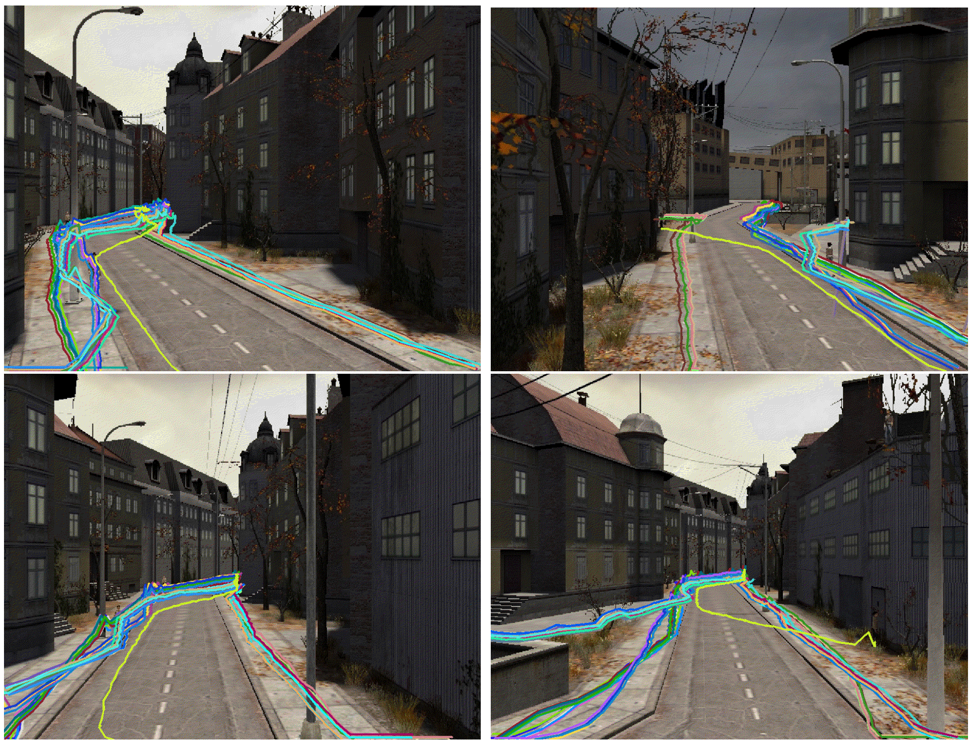


Figure 5.8: Trajectory association results on input trajectories (note: corresponding trajectories in four views are shown with same colour).

have affected the clustering results especially in c_2, c_3 and c_4. Only two clusters are found on the left-path in c_3 and c_4 and the corresponding right-path in c_2. On the other hand results in c_1, which is observing the paths more closely, are accurate. However, c_1 is only observing a small region of the complete area. Therefore, to accurately detect clusters at larger area, it is must to apply trajectory association to reconstruct complete trajectories before applying clustering.

The colour-coded trajectory association results are shown in Fig. 5.8. The same colour in multiple views shows the corresponding trajectories. The final clustering results on complete trajectories are shown on a synthetic common-view in Fig. 5.9. The final results exactly match the ground truth.

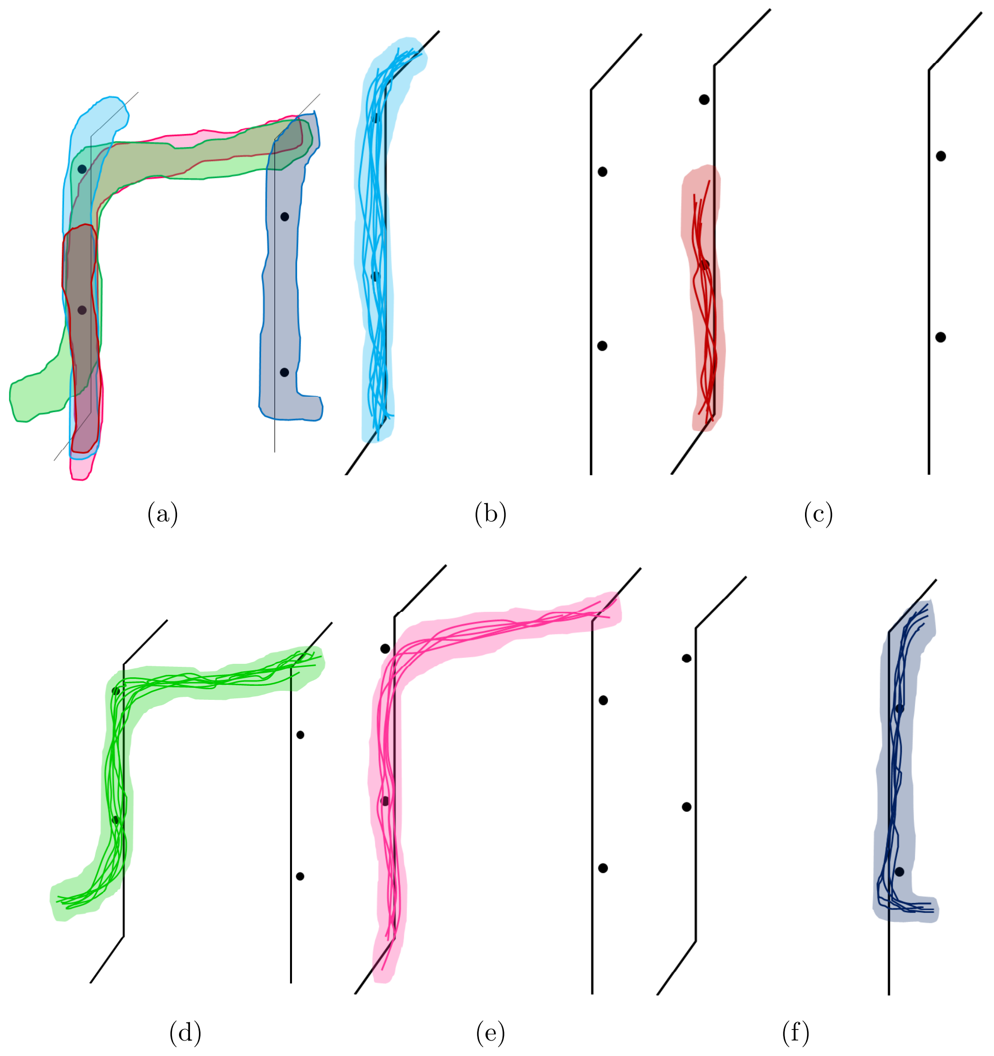


Figure 5.9: Clustering results on a synthetic view. (a) Estimated common patterns and (b-f) complete trajectories within each cluster.

5.7 Summary

In this chapter, we have presented a novel multi-feature video object trajectory clustering algorithm that estimates common patterns of behaviours and isolates outliers. The proposed algorithm is based on four main steps, namely the extraction of a set of representative trajectory features, non-parametric clustering, cluster merging and information fusion for the identification of normal and rare object motion patterns. First we transformed the trajectories into a set of feature spaces on which Mean-shift identifies the modes and the corresponding clusters. Furthermore, a merging procedure is devised to refine these results by combining similar adjacent clusters. The final common patterns are estimated by fusing the clustering results across all feature spaces. Clusters corresponding to reoccurring trajectories are considered as normal, whereas sparse trajectories are associated to abnormal and rare events. An example of trajectory clustering in a multi-camera setup is also presented. The performance of the proposed algorithm is evaluated in Ch. 6 and compared with existing approaches.

Chapter 6

Experimental results

The organisation of this chapter is as follows: Sec. 6.1 provides details of experimental results of the camera calibration algorithm and its comparison with a state-of-the-art approach; Sec. 6.2 discusses the results of object association across multiple cameras and the comparison of the algorithm with existing works and finally, Sec. 6.3 discusses the results of the trajectory clustering algorithm and its comparison with existing works.

6.1 Extrinsic calibration of non-overlapping cameras

6.1.1 Datasets

To evaluate the performance of the proposed extrinsic camera calibration approach, we use two simulated and one real camera setups. The simulated setups consist of six and eight cameras. The total coverage area is 200x200 unit-size. Figure 6.1 shows the simulated setups, where the layout of the cameras is superimposed on Google maps to highlight the difficulty of the problem. The figure also shows that there exist unobserved regions which are at least twice in area as compare to the field of view of each camera. We manually constructed the trajectory of an object, where we intentionally introduce turn in unobserved regions to increase the complexity. Furthermore, the real dataset is generated from five cameras (Fig. 6.2). A toy car moves across the network on a toy car track. Since the car track contains fixed paths, to increase the variability of the trajectory we keep rotating the track with our hands during video acquisition. The coverage area is approximately 4.40 sq. meters. The dataset consists of 19137 frames. We use change detection to extract and to generate the trajectory [89].

6.1.2 Performance evaluation

To compare the results of the proposed approach before bundle adjustment (*PBBA*) and after bundle adjustment (*PABA*), we use a state-of-the-art *MAP*-based approach [7]. Additionally, we initialise the MAP-based approach with the initial configuration estimated by the proposed approach (PBBA) and refer to it as Modified MAP (MMAP).

We constructed the ground truth, where position and rotation of cameras are marked manually. The position estimation error, ϵ_t, is calculated with the L_2 norm between the ground truth and estimated position of cameras; whereas the rotation error, ϵ_r, is calculated with the L_1 norm between the ground truth and estimated rotation of cameras.

The algorithms are developed in Matlab (ver 7.5) and run on a Pentium (R) dual core CPU @ 2.00 GHz with 3.0 GB memory.

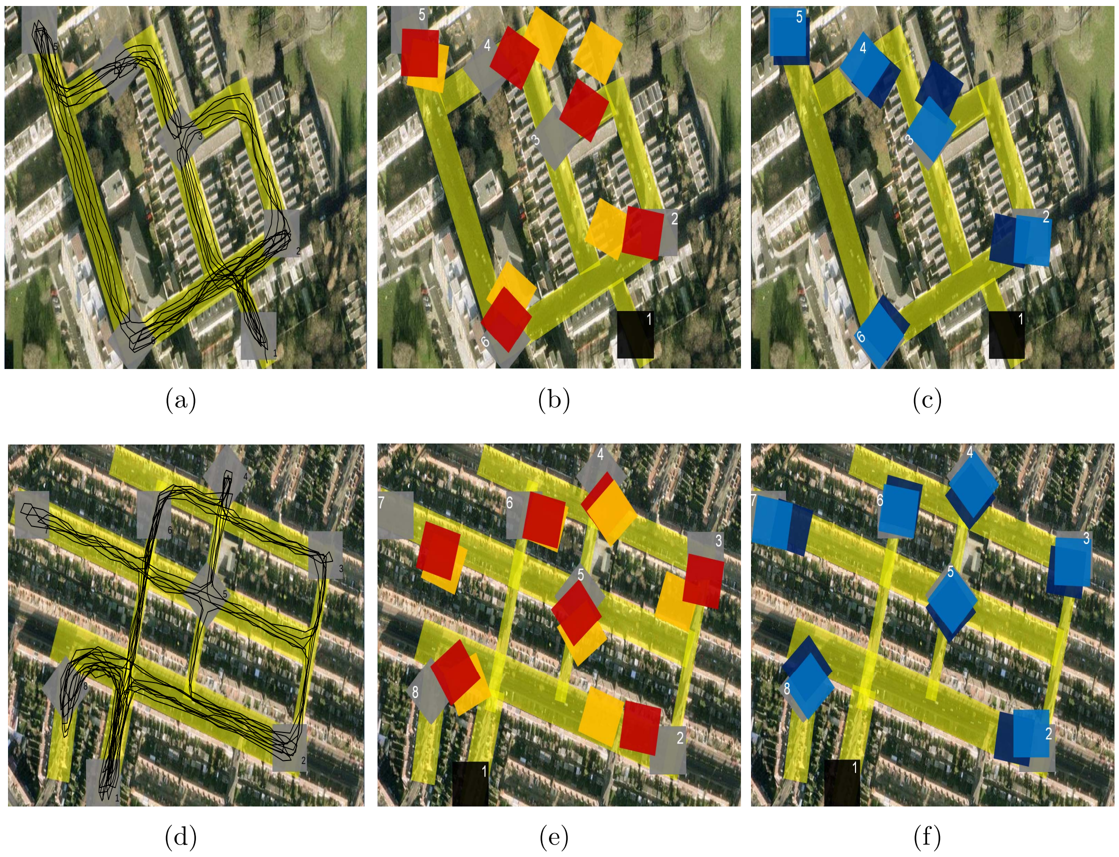


Figure 6.1: Examples of camera network localisation on simulated datasets: (a,d) trajectory of an object (black line) and field-of-view of each camera (gray) (Copyright 2010 Google - Map data); (b,e) localisation results of MAP (orange) and MMAP (red) superimposed on the truth configuration; (c,f) localisation results of PBBA (dark-blue) and PABA (light-blue) superimposed on the truth configuration. c_1 (black) is the reference camera.

6.1.3 Discussion

Figure 6.1(a,d) shows the ground truth configurations of two simulated camera networks. Figure 6.1(b,e) shows the localisation using MAP (orange) and is superimposed on the true configuration. For both sequences, c_1 is considered as the reference (black). The visual analysis shows that there are considerable errors in approximating the cameras' positions. Specifically, in the first sequence, c_2, c_3 and c_4 are noticeably misplaced. Similarly, for the second sequence, c_2, c_3 and c_7 are far from the true configuration. This is primarily because MAP can handle

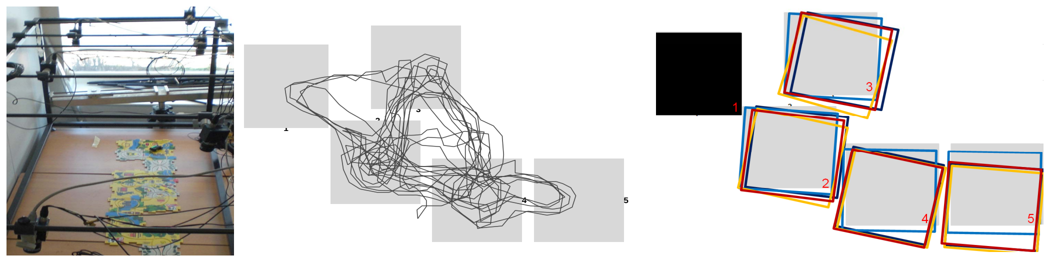


Figure 6.2: Localisation results on a real dataset: (a) the experimental setup; (b) trajectory (line) superimposed on the original configuration (gray); (c) localisation results using MAP (orange), MMAP (red), PBBA (dark-blue) and PABA (light-blue) superimposed on the original configuration (gray).

Table 6.1: Evaluation of the accuracy of the algorithms under analysis on the simulated datasets.

S	Algo.		c_2	c_3	c_4	c_5	c_6	c_7	c_8	**Avg.**
1	MAP	ϵ_t	20.86	19.83	14.58	11.52	3.46	-	-	**14.05**
		ϵ_r	25.00	16.00	16.00	10.00	10.00	-	-	**15.40**
	MMAP	ϵ_t	24.14	18.50	11.79	11.00	2.70	-	-	**13.63**
		ϵ_r	10.00	15.00	15.00	5.00	2.00	-	-	**9.40**
	PBBA	ϵ_t	2.01	1.12	1.24	0.54	0.51	-	-	**1.08**
		ϵ_r	10.00	15.00	10.00	5.00	0.00	-	-	**8.00**
	PABA	ϵ_t	0.25	0.34	0.12	0.06	0.16	-	-	**0.18**
		ϵ_r	5.00	5.00	4.00	0.00	0.00	-	-	**2.80**
2	MAP	ϵ_t	20.02	27.45	21.11	18.16	14.62	4.74	6.99	**16.16**
		ϵ_r	20.00	20.00	4.00	10.00	18.00	21.00	20.00	**16.14**
	MMAP	ϵ_t	23.83	29.97	20.52	17.61	14.44	4.16	5.84	**16.62**
		ϵ_r	0.00	10.00	2.00	6.00	13.00	16.00	10.00	**8.14**
	PBBA	ϵ_t	2.26	0.82	0.14	0.03	0.61	2.29	0.38	**0.93**
		ϵ_r	10.00	8.00	2.00	4.00	12.00	15.00	4.00	**7.86**
	PABA	ϵ_t	0.61	0.17	0.21	0.21	0.34	0.40	0.61	**0.36**
		ϵ_r	0.00	2.00	1.00	0.00	9.00	10.00	3.00	**3.57**

only a subtle turn in the unobserved regions. In these sequences, the object takes a sharp turn in the unobserved regions that affects the performance of the algorithm. The localisation results for MAP are improved when it is initialised with the output of PBBA. On the other hand, Figure 6.1(c,f) shows the results of the proposed approach, where initial results (dark-blue) are improved by employing bundle adjustment (light-blue).

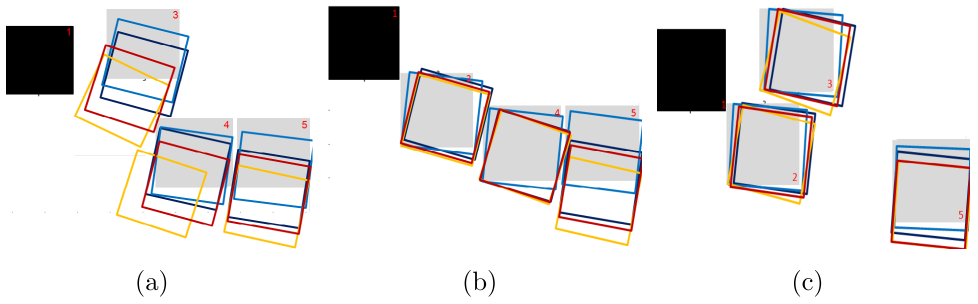


Figure 6.3: Localisation results in case of camera failure: MAP (orange), MMAP (red), PBBA (dark-blue) and PABA (light-blue) when (a) c_2 fails; (b) c_3 fails; and (c) c_4 fails.

Table 6.1 summarises the localisation accuracy for MAP, MMAP, PBBA and PABA on the simulated datasets. The discussion on Table 6.1 is organised as: we start with first simulated dataset by analysing values of ϵ_t and ϵ_r for individual cameras using MAP and MMAP. We compare these results with PBBA and PABA. This is followed by the discussion on average results. Next, we detail the comparison of ϵ_t and ϵ_r on the second simulated dataset. Again, we start with discussion on individual cameras followed by the average results.

For the first sequence, the translation errors using MAP for c_2, c_3, c_4, c_5 and c_6 are 20.86, 19.83, 14.58, 11.52 and 3.46 unit-sizes, respectively. Except for the path between c_5 and c_6, the object always takes sharp turn, as it moves between a pair of cameras. The nature of these turns varies with the camera pairs; hence, it is impossible to construct a single motion model apriori, which can handle these variations of motion paths. Although MMAP reduced the translation error by 1.33, 2.79, 0.52 and 0.76 unit-sizes for c_3, c_4, c_5 and c_6, respectively, it is interesting to note that specifically for c_2, the error is increased by 3.28 unit-size. This hints that better initialisation alone does not help a lot in improving the results for all cameras, but the choice of motion models plays a more vital role in better localisation. Similar trends in the results can be seen for rotation errors, where the maximum error with MAP is 25 deg. compared to 15 deg. with MMAP. On the other hand, the translation errors with PBBA are 2.01, 1.12, 1.24, 0.54 and 0.51 for c_2, c_3, c_4, c_5 and c_6, respectively. Further improvements are achieved by PABA, where errors are reduced to 0.25, 0.34, 0.12, 0.06 and 0.16 for c_2, c_3, c_4, c_5 and c_6, respectively. In terms of error percentage, PABA has 7.29%, 6.88%, 5.11%, 4.05% and 1.17% less error than MAP for c_2, c_3, c_4, c_5 and c_6, respectively. Similarly, the maximum rotation error with PABA is 5 deg., which is 20 deg. less than the maximum rotation error by MAP. On average, MAP has 13.87 size-units larger position estimation error compared to PABA. This error difference is reduced by 0.42 size-units with MMAP. In terms of rotation, MAP results in 12.60 deg. worse error than PABA. With MMAP this error difference is reduced to 6.6 deg.

For the second sequence, the translation errors using MAP for c_2, c_3, c_4, c_5, c_6, c_7 and c_8 are 20.02, 27.45, 21.11, 18.16, 14.62, 4.74 and 6.99 unit-sizes, respectively. Like the first sequence, the object mostly takes a sharp turn, as it moves between a pair of cameras. Note that the translation errors using MMAP are increased by 3.81 and 2.52 unit-sizes for c_2 and c_3, respectively. This confirms our initial observation that better initialisation plays little role in improving the results, but the choice (or learning) of motion models play a more vital role in better localisation. Similarly for rotation errors, the maximum error with MAP is 21 deg. compared to 16 deg. with MMAP. On the other hand, the translation errors with PBBA are 2.26, 0.82, 0.14, 0.03, 0.61, 2.29 and 0.38 for c_2, c_3, c_4, c_5, c_6, c_7 and c_8, respectively. Further improvements are achieved by PABA, where errors are reduced to 0.61, 0.17, 0.21, 0.21, 0.34, 0.40 and 0.61 for c_2, c_3, c_4, c_5, c_6, c_7 and c_8, respectively. In terms of error percentage, PABA has 6.86%, 9.64%, 7.39%, 6.35%, 5.04%, 1.53%

Table 6.2: Comparison of processing time for MAP, MMAP, PBBA and PABA when varying the number of unknown parameters (camera localisation parameters and missing observations).

S	No. of param	Algo.			
		MAP	MMAP	PBBA	PABA
1	18+1187	310.11	280.39	120.12	173.06
2	24+2118	563.05	401.73	155.71	221.10

and 2.41% less error than MAP for c_2, c_3, c_4, c_5, c_6, c_7 and c_8, respectively. Also, the maximum rotation error with PABA is 2 deg., which is 19 deg. less than the maximum rotation error by MAP. Similar to the first sequence, the average position estimation error for PABA is 0.36 size-units, which is 15.8 size-units smaller than that of MAP and 16.26 size-units smaller than MMAP. In summary, analysis of results on two simulated sequences shows that instead of using single global motion model; learning the motion models locally (at camera pair level) helps in better localisation of cameras.

Table 6.2 compares MAP, MMAP, PBBA and PABA on both simulated datasets in terms of processing time (in seconds). For the first dataset the number of unknown parameters is 1205 (12 for the camera positions, 6 for the camera rotations and 1187 for the missing observations) and for the second dataset, the number of unknown parameters is 2118 (16 for the camera positions, 8 for the camera rotations and 2094 for missing observations). By increasing the number of unknown parameters from 1205 to 2118, the processing time for PABA is increased by 27.74% and the processing time for MAP is increased by 81.56%. With a better initialisation (i.e., MMAP) the processing time for MAP is increased by 43.02%, which is however still higher than PABA. This confirms that the proposed approach is less time consuming than MAP. This is an important aspect especially for the scalability of a network.

Figure 6.2(c) illustrates the localisation results of MAP, MMAP, PBBA and

PABA on the real dataset. The corresponding localisation errors are summarised in Table 6.3. The table shows that the maximum translation error with MAP is for c_5. This is primarily due to the reason that c_5 contains fewer observations compared to c_2, c_3 and c_4. The approach tends to find the overall solution that is in agreement with most of the available observations; therefore, the approach localises c_2, c_3 and c_4 better than c_5, although, the trajectory segments between c_4 and c_5 are linear. However, PBBA (or PABA) does not have any such constraint; but approximates the position of cameras locally at camera pair level, thus the error in localising c_5 is approximately the same as the error in approximating the position of c_4. The average position estimation error for the proposed approach is 0.36 mm, while the average position estimation errors for MAP and MMAP are respectively 16.72 mm and 16.63 mm. Similarly, the average rotation error for PABA is 2.25 deg. which is 9.97 deg. smaller than MAP and 6.75 deg. smaller than MMAP.

Furthermore, we also evaluate the robustness of the proposed algorithm to camera failure situations and compare it with MAP. To this end, we dropped all the observations from one camera at a time and applied the algorithms. The results are shown in Fig. 6.3. Table 6.3 summarises the localisation accuracy: c_3 is the most important camera in the network as it is observing the region where the object is taking most of the sharp turns; therefore, in case of its failure the localisation errors for the rest of network increases substantially. The least important camera is c_4, the trajectory segments in c_4 can be approximated with the observations of c_2, c_3 and c_5 and its failure have not affected considerably the performance of network localisation.

In summary, PABA outperforms MAP in three aspects: (a) in terms of the motion model, contrary to MAP, PABA does not consider a global motion model,

Table 6.3: Evaluation of the accuracy and robustness to camera failure of the algorithms under analysis on the real dataset.

Desc.	Algo.		c_2	c_3	c_4	c_5	**Avg.**
All	MAP	ϵ_t	8.91	12.76	17.88	27.34	**16.72**
		ϵ_r	13.00	17.00	13.00	5.00	**12.00**
	MMAP	ϵ_t	8.90	13.04	17.59	26.98	**16.63**
		ϵ_r	7.00	11.00	13.00	5.00	**9.00**
	PBBA	ϵ_t	0.66	1.82	0.28	0.30	**0.76**
		ϵ_r	7.00	11.00	13.00	4.00	**8.75**
	PABA	ϵ_t	0.08	0.79	0.28	0.28	**0.36**
		ϵ_r	4.00	3.00	1.00	1.00	**2.25**
c_2 fails	MAP	ϵ_t	-	9.36	13.94	26.15	**16.48**
		ϵ_r	-	25.00	18.00	12.00	**18.33**
	MMAP	ϵ_t	-	10.24	16.74	26.43	**17.80**
		ϵ_r	-	18.00	15.00	10.00	**14.33**
	PBBA	ϵ_t	-	0.27	0.93	1.12	**0.77**
		ϵ_r	-	15.00	13.00	9.00	**12.33**
	PABA	ϵ_t	-	0.14	0.49	0.50	**0.38**
		ϵ_r	-	14.00	8.00	7.00	**9.67**
c_3 fails	MAP	ϵ_t	8.95	-	18.10	26.15	**17.33**
		ϵ_r	15.00	-	19.00	12.00	**15.33**
	MMAP	ϵ_t	8.91	-	18.07	26.43	**17.80**
		ϵ_r	15.00	-	17.00	10.00	**14.00**
	PBBA	ϵ_t	0.64	-	0.11	1.11	**0.41**
		ϵ_r	15.00	-	16.00	9.00	**13.33**
	PABA	ϵ_t	0.06	-	0.11	0.49	**0.22**
		ϵ_r	7.00	-	6.00	7.00	**6.67**
c_4 fails	MAP	ϵ_t	8.92	12.76	-	27.15	**16.28**
		ϵ_r	13.00	17.00	-	5.00	**11.67**
	MMAP	ϵ_t	8.89	13.04	-	27.15	**16.36**
		ϵ_r	7.00	11.00	-	5.00	**7.67**
	PBBA	ϵ_t	0.66	1.82	-	0.35	**0.94**
		ϵ_r	4.00	11.00	-	5.00	**6.67**
	PABA	ϵ_t	0.08	0.79	-	0.21	**0.36**
		ϵ_r	4.00	3.00	-	2.00	**3.00**

therefore it can handle a sharp turn between a pair of cameras efficiently; (b) in terms of number of observations, PABA does not give any preference to cameras based on the number of its observations (although more observations help in better localisation of that camera), therefore it is possible to localise a camera with fewer observations as compared to MAP which inherently gives higher preference to the cameras with more observations; and (c) in terms of processing time, PABA requires approximately 1/3 less processing time compared to MAP; therefore can be more effective in case of network scalability.

In the next section, we provide results of object correspondence to reconstruct complete trajectories across multiple calibrated cameras.

6.2 Global trajectory reconstruction

6.2.1 Datasets

To evaluate the performance of the proposed hybrid multi-feature association (HMFA) approach, we use two real world datasets. The first dataset ($S1$) is an indoor basketball video sequence, which consists of 500 frames (RGB 24 bit images at 25 frames/sec and 1200x1600 pixels), describing a scene simultaneously recorded by 4 cameras located at different viewpoints (see Fig. 6.4). The second dataset ($S2$) is a more complex soccer match footage, which consists of 3000 frames (RGB 24 bit images at 25 frames/sec and 1920x1080 pixels), describing a scene simultaneously recorded by 6 cameras located at different viewpoints (see Fig. 6.5). In both datasets, the closeness of players' movement and similarity in team colours make the association task even more challenging. As these sequences are captured during real games, therefore, no constraints were imposed on objects' trajectories.

6.2.2 Performance evaluation

For both datasets, we used visual data (manual) to generate the ground truth for association, we perform objective evaluation of association and fusion results using *Recall* (R_a) and *Precision* (P_a). R_a is the fraction of accurate associations to the

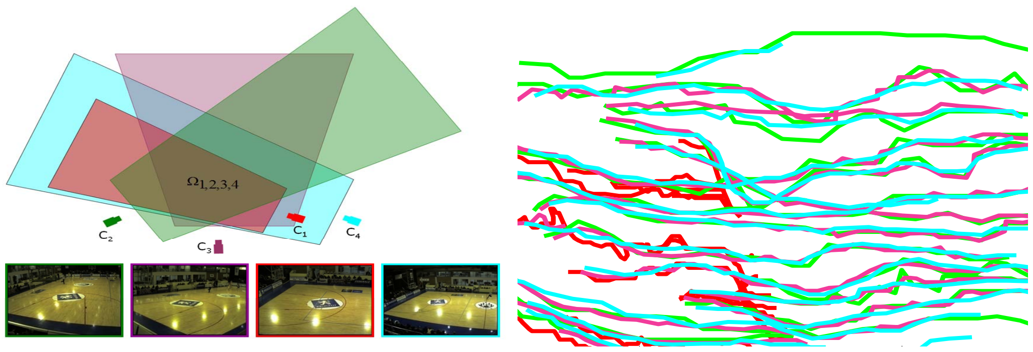


Figure 6.4: Trajectory segments accumulated over 500 frames from the 4 partially overlapping cameras. (Left) configuration of the cameras; (right) accumulated trajectory segments (segment's colour corresponds to the camera's colour).

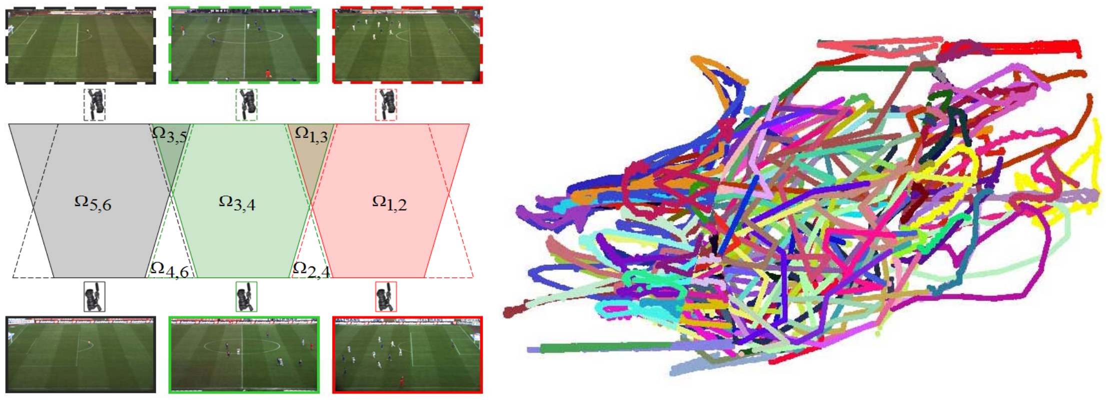


Figure 6.5: Trajectory segments accumulated over 3000 frames from the 6 partially overlapping cameras of Figure 3. (Left) configuration of the cameras; (right) trajectory segments (colour-coded). Note the visibility of the limits of the fields of view of each camera.

true number of associations. P_a is the fraction of accurate associations to the total number of achieved associations. Let ξ_Ω be the ground truth for pairs of trajectories on the overlapping region Ω (marked in Fig. 6.6 and Fig. 6.7) and let E_Ω be the estimated results. Then R_a and P_a are calculated as:

$$R_a = \frac{|\xi_\Omega \cap E_\Omega|}{|\xi_\Omega|}, \tag{6.1}$$

$$P_a = \frac{|\xi_\Omega \cap E_\Omega|}{|E_\Omega|}, \tag{6.2}$$

Table 6.4: Evaluation and comparison of trajectory association results on $S1$ and $S2$.

	$S1$		$S2$															
	$\Omega_{1,2,3,4}$		$\Omega_{1,2}$		$\Omega_{3,4}$		$\Omega_{5,6}$		$\Omega_{1,3}$		$\Omega_{2,4}$		$\Omega_{3,5}$		$\Omega_{4,6}$		**Avg.**	
Alg.	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a	R_a	P_a
M1	.75	.83	.60	.70	.76	.85	.73	.78	.71	.87	.72	.76	.68	.80	.73	.71	**.71**	**.79**
M2	.95	.93	.80	.90	.88	.90	.81	.98	.90	.90	.82	.96	.82	.90	.78	.81	**.84**	**.91**
M3	1.00	1.00	.76	.81	.81	.92	.78	.99	.89	.92	.84	.97	.87	.96	.82	.85	**.85**	**.93**
HMFA	1.00	1.00	.96	.98	.95	1.00	.96	1.00	.93	.95	.85	.98	.87	.98	.82	.85	**.92**	**.97**

where $|.|$ is the cardinality of a set. We compare the performance of the *HMFA* with standard Dynamic Time Warping (DTW) [90] (*M1*) and two state-of-the-art approaches presented in [9] (*M2*) and [8] (*M3*) in terms of P_a and R_a for both sequences.

6.2.3 Discussion

Figure 6.6 and Fig. 6.7 show the complete global trajectories of all the objects for both sequences. The objective results are compiled in Tab. 6.4. We initially evaluate the performance of the algorithms only on the overlapping regions in both sequences; followed by the overall average analysis of the results. For $S1$, there is only one overlapping region $\Omega_{1,2,3,4}$. The results for R_a are 0.75, 0.95, 1.00 and 1.00 with *M1*, *M2*, *M3* and *HMFA*, respectively. Similarly, the results for P_a are 0.83, 0.93, 1.00 and 1.00 with *M1*, *M2*, *M3* and *HMFA*, respectively. During this short duration of game, most players exhibit similar motion patterns; therefore it is difficult to distinguish players using DTW (*M1*) and using only the shape information (*M2*); thus there are association errors. However, *M3* and *HMFA* accurately associate the trajectories from multiple views.

For $S2$, we divide overlapping regions into two: dense ($\Omega_{1,2}$, $\Omega_{3,4}$, $\Omega_{5,6}$) and sparse ($\Omega_{1,3}$, $\Omega_{2,4}$, $\Omega_{3,5}$, $\Omega_{4,6}$). For dense regions, the average R_a are 0.70, 0.83, 0.78 and 0.96 with *M1*, *M2*, *M3* and *HMFA*, respectively. Similarly, the average

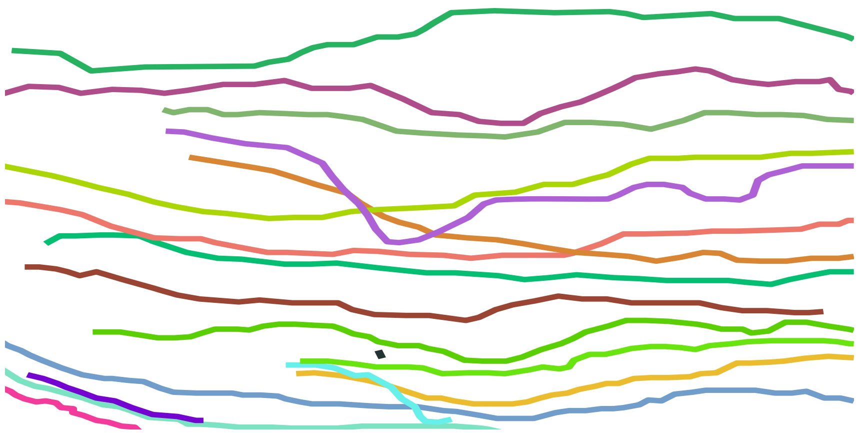


Figure 6.6: Trajectory association and fusion results across the cameras of $S1$. Each complete trajectory is shown with different colour.

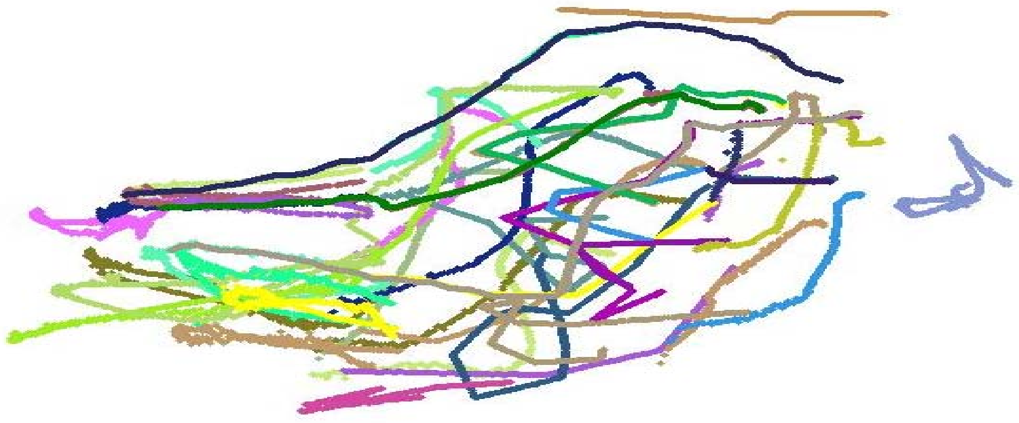


Figure 6.7: Trajectory association and fusion results across the cameras of $S2$. Each complete trajectory is shown with different colour.

P_a are 0.78, 0.93, 0.91 and 0.99 with *M1*, *M2*, *M3* and *HMFA*, respectively. On the other hand, for sparse regions, the average R_a are 0.71, 0.83, 0.86 and 0.87 with *M1*, *M2*, *M3* and *HMFA*, respectively. Similarly, the average P_a are 0.78, 0.89, 0.92 and 0.94 with *M1*, *M2*, *M3* and *HMFA*, respectively. The analysis reveals that for these sequences, DTW in its standard form has more association errors than other approaches primarily due to spatio-temporal closeness of trajectories. Furthermore, for dense regions, *M2* has better precision and recall compared to

M3. However, for sparse regions the performance of *M3* is better than *M2*. This shows that reprojection-based approach works well when the number of trajectories is small and trajectories are not so close to each other; however, for dense regions only trajectory feature based analysis can provide more reliable results. Therefore, for both types of regions, the *HMFA* approach is better in associating trajectories than its counterparts. On average the *HMFA* approach is better by 8% and 6% for R_a and P_a respectively, compared to *M2*. Compared to *M3*, the *HMFA* approach outperforms it by 7% and 4% for R_a and P_a, respectively.

In summary, in complex datasets such as $S1$ and $S2$, where objects are very close in time and space, trajectory statistics help in better association. In particular, for dense segments' regions, *HMFA* outperformed the other approaches because of the more generic feature-set used and built-in ground plane and image plane verification method. On the other hand, *M2* covers very limited features and lacks a procedure to resolve conflicting situations. This results in lower P_a and R_a scores. Similarly, if the segments are too close on the image plane, they cannot be separated using the reprojection error criterion. Therefore, *M3* fails to distinguish the segments that in fact belong to different physical objects exhibiting similar motion patterns.

In the next section, we discuss results of trajectory clustering given that the object association is already performed and complete trajectories are already reconstructed.

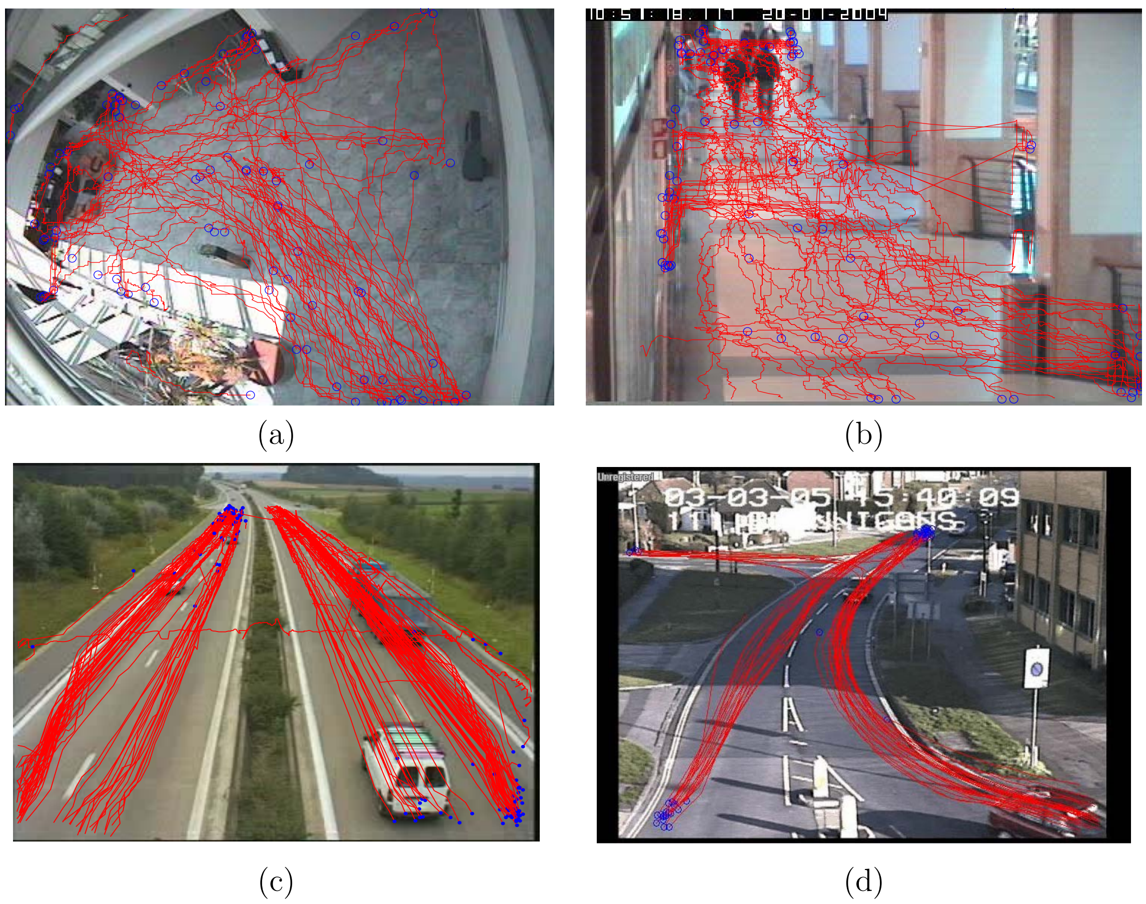


Figure 6.8: Trajectory datasets: CAVIAR video sequences (a) S1 and (b) S2, Highway traffic video sequences (c) S3 and (d) S4.

6.3 Trajectory clustering

6.3.1 Datasets

To evaluate the performance of the proposed multi-feature based trajectory clustering (*MFTC*) approach, four standard video sequences are used . The first two sequences are from the *CAVIAR* dataset[1]. These videos are captured in a lobby ($S1$, 90 trajectories, 384 x 288 pixels, 25 Hz) and in a corridor of a shopping centre ($S2$, 84 trajectories, 384 x 288 pixels, 25 Hz). Two traffic sequences are from the

[1]http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

MPEG–7 ($S3$, 134 trajectories[2], 352 x 288 pixels, 25 Hz) and from the CLEAR [91] ($S4$, 47 trajectories, 720 x 480 pixels, 25 Hz) datasets, respectively. Figure 6.8 shows the cumulated trajectories superimposed on a key-frame of each test sequence.

For $S1$, we are interested in clustering the trajectories into four main groups consisting of 18, 13, 22 and 10 trajectories and representing, respectively, the trajectories belonging to the upper-region on the image plane (starting from the top-right and ending at top-left), the trajectories starting and ending at the bottom-right of the image plane, the trajectories from the centre to the bottom-right of the image and, finally, the trajectories starting and ending at the centre-left region of the image. For $S2$, we are interested in finding five dominant regions consisting of 15, 28, 10, 9 and 10 trajectories and representing, respectively, the trajectories from the centre-left to the bottom-right on the image plane, the trajectories on the top-left corner, the trajectories from the bottom-right to the centre-left, the trajectories from the top-left corner on the image plane to the bottom region and, finally, the trajectories that start and end in the bottom-half of the image. For the $S3$ traffic sequence the objective is to determine two dominant patterns consisting of 55 and 47 trajectories. For $S4$, the goal is to cluster trajectories into three groups consisting of 20, 17 and 7 trajectories. The remaining trajectories are considered as outliers. Note that in S3 and S4 there are true outliers e.g., a person crossing the highway (S3) and a vehicle stopping on the road (S4). In S1 and S2, the outliers are short trajectories generated by window shoppers (S1) or people which are very far from the cameras (S4).

[2]http://www.tele.ucl.ac.be/PROJECTS/MODEST/

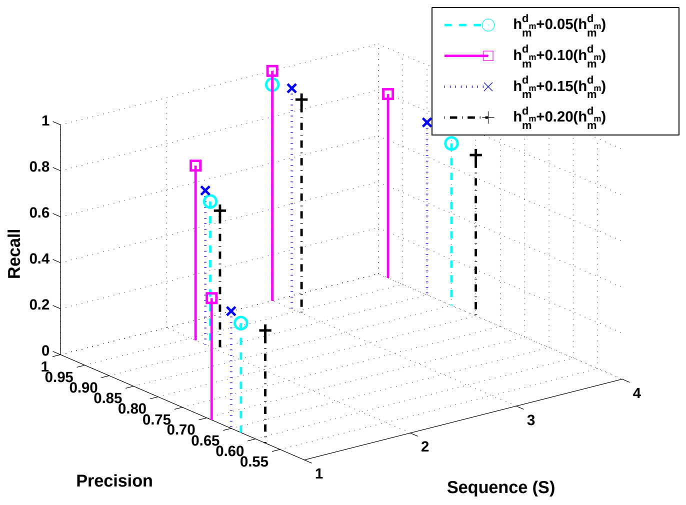


Figure 6.9: Sensitivity analysis at the variation of the cluster merging criterion.

6.3.2 Performance evaluation

To objectively assess the clustering performance we use *Precision* (P_c) and *Recall* (R_c). For the i^{th} final cluster, C_i^f, P_c is calculated as

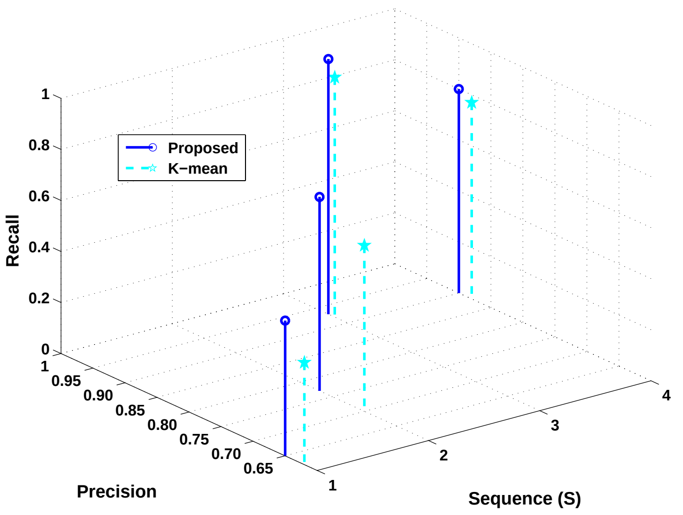
$$P_c = \frac{|C_i^f \cap \Gamma_i|}{|C_i^f|}, \tag{6.3}$$

and R_c as

$$R_c = \frac{|C_i^f \cap \Gamma_i|}{|\Gamma_i|}, \tag{6.4}$$

where $|.|$ is the cardinality of a cluster.

In addition to the evaluation of the results obtained using the original trajectories, we perform a series of robustness tests in which we evaluate the clustering results after corrupting the input data (missing data and noisy data). In the first test, we reduce the trajectory sampling rate by 2, 3, 4 and 5 steps. In the second test, we add Gaussian noise with standard deviation equal to 5%, 10%, 15% and 20% of the longest trajectory of each set. In all further experiments proximity measure



(a)

Figure 6.10: Comparison of the MFTC approach and K-mean.

Table 6.5: Clustering precision and recall comparison for $S1$, $S2$, $S3$ and $S4$.

	MFTC										SOM [13]										TDH [16]									
	C_1		C_2		C_3		C_4		C_5		C_1		C_2		C_3		C_4		C_5		C_1		C_2		C_3		C_4		C_5	
	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c
S1	.62	.50	.56	.77	1	.54	.43	.30	-	-	.36	.44	.33	.54	.23	.23	.13	.20	-	-	.45	.50	.36	.61	.44	.18	.21	.40	-	-
↓ 2	.60	.50	.54	.76	1	.52	.38	.29	-	-	.34	.41	.30	.52	.21	.21	.11	.18	-	-	.44	.48	.36	.56	.42	.18	.21	.40	-	-
↓ 3	.60	.49	.54	.76	1	.52	.36	.29	-	-	.33	.39	.28	.50	.19	.19	.11	.15	-	-	.40	.45	.34	.52	.40	.16	.20	.36	-	-
↓ 4	.54	.47	.52	.75	.95	.51	.36	.28	-	-	.31	.37	.26	.48	.18	.16	.09	.13	-	-	.38	.42	.32	.43	.38	.14	.19	.30	-	-
↓ 5	.50	.47	.48	.75	.90	.47	.35	.27	-	-	.29	.35	.24	.43	.15	.15	.06	.10	-	-	.38	.39	.29	.40	.36	.12	.18	.28	-	-
5%	.58	.49	.53	.71	.89	.45	.36	.29	-	-	.30	.38	.27	.50	.17	.20	.10	.14	-	-	.40	.41	.32	.42	.40	.14	.19	.36	-	-
10%	.50	.48	.50	.70	.81	.42	.32	.26	-	-	.28	.24	.24	.49	.14	.18	.08	.11	-	-	.37	.36	.29	.39	.37	.11	.17	.30	-	-
15%	.47	.46	.48	.68	.80	.39	.29	.26	-	-	.25	.30	.20	.45	.14	.15	.06	.09	-	-	.34	.32	.27	.36	.35	.09	.15	.28	-	-
20%	.43	.45	.46	.67	.80	.37	.29	.24	-	-	.20	.25	.17	.42	.12	.14	.06	.06	-	-	.31	.28	.25	.31	.32	.46	.14	.22	-	-
Avg	**.54**	**.48**	**.51**	**.73**	**.91**	**.46**	**.35**	**.28**	-	-	**.30**	**.35**	**.25**	**.48**	**.17**	**.18**	**.09**	**.13**	-	-	**.38**	**.40**	**.31**	**.40**	**.38**	**.11**	**.19**	**.32**	-	-
S2	.80	.57	.80	.90	.86	.64	.63	.94	.75	.75	.54	.50	.12	.10	.46	.54	.45	.78	.17	.25	.42	.36	.26	.90	.54	.64	.07	.06	.25	.25
↓ 2	.80	.53	.80	.90	.84	.63	.61	.94	.74	.75	.52	.50	.12	.10	.46	.52	.45	.78	.16	.24	.38	.35	.24	.90	.51	.60	.07	.05	.23	.24
↓ 3	.79	.51	.78	.89	.84	.60	.61	.94	.72	.73	.52	.48	.10	.08	.42	.48	.42	.76	.14	.23	.35	.32	.21	.86	.48	.58	.05	.04	.21	.20
↓ 4	.76	.49	.76	.86	.80	.57	.59	.90	.70	.69	.48	.46	.09	.07	.40	.46	.39	.73	.12	.21	.31	.26	.19	.80	.44	.55	.03	.02	.18	.16
↓ 5	.75	.45	.75	.83	.80	.54	.56	.87	.68	.67	.44	.42	.06	.06	.36	.44	.36	.69	.10	.20	.27	.21	.17	.76	.40	.50	.03	.02	.17	.12
5%	.78	.50	.78	.88	.85	.61	.62	.90	.73	.72	.50	.44	.10	.09	.41	.51	.34	.65	.13	.21	.38	.28	.20	.73	.44	.58	.06	.05	.16	.20
10%	.76	.47	.74	.85	.83	.58	.60	.84	.79	.72	.47	.36	.08	.07	.39	.46	.28	.62	.09	.18	.27	.21	.17	.68	.38	.52	.04	.04	.14	.16
15%	.74	.44	.71	.81	.79	.58	.57	.87	.68	.68	.40	.30	.07	.06	.37	.42	.26	.59	.08	.17	.24	.18	.14	.58	.36	.48	.01	.02	.10	.14
20%	.68	.39	.69	.78	.74	.51	.53	.84	.64	.61	.38	.26	.06	.01	.34	.41	.22	.55	.06	.14	.20	.15	.12	.48	.31	.40	.01	.01	.06	.10
Avg	**.76**	**.48**	**.76**	**.86**	**.82**	**.58**	**.59**	**.89**	**.71**	**.70**	**.47**	**.41**	**.09**	**.07**	**.40**	**.47**	**.35**	**.68**	**.12**	**.20**	**.31**	**.26**	**.18**	**.71**	**.43**	**.54**	**.04**	**.03**	**.17**	**.17**
S3	.96	1	.90	1	-	-	-	-	-	-	.72	1	.98	.91	-	-	-	-	-	-	.83	1	.86	.79	-	-	-	-	-	-
↓ 2	.94	1	.90	1	-	-	-	-	-	-	.72	.98	.98	.91	-	-	-	-	-	-	.82	.90	.86	.76	-	-	-	-	-	-
↓ 3	.91	1	.90	1	-	-	-	-	-	-	.71	.98	.98	.90	-	-	-	-	-	-	.81	.86	.86	.73	-	-	-	-	-	-
↓ 4	.87	.97	.89	1	-	-	-	-	-	-	.71	.97	.97	.90	-	-	-	-	-	-	.81	.82	.83	.73	-	-	-	-	-	-
↓ 5	.75	.96	.89	1	-	-	-	-	-	-	.70	.88	.98	.89	-	-	-	-	-	-	.84	.68	.89	.71	-	-	-	-	-	-
5%	.92	1	.90	1	-	-	-	-	-	-	.72	1	.98	.90	-	-	-	-	-	-	.81	.90	.81	.63	-	-	-	-	-	-
10%	.87	1	.89	1	-	-	-	-	-	-	.75	.96	.98	.88	-	-	-	-	-	-	.79	.86	.81	.57	-	-	-	-	-	-
15%	.81	.95	.89	1	-	-	-	-	-	-	.74	.91	.97	.85	-	-	-	-	-	-	.79	.80	.80	.51	-	-	-	-	-	-
20%	.75	.91	.85	1	-	-	-	-	-	-	.72	.87	.96	.81	-	-	-	-	-	-	.67	.77	.77	.42	-	-	-	-	-	-
Avg	**.86**	**.98**	**.89**	**1**	-	-	-	-	-	-	**.72**	**.95**	**.97**	**.88**	-	-	-	-	-	-	**.79**	**.84**	**.83**	**.64**	-	-	-	-	-	-
S4	.87	1	.84	1	1	.40	-	-	-	-	.70	1	.87	.65	.35	.12	-	-	-	-	.93	.65	.90	1	1	.21	-	-	-	-
↓ 2	.86	1	.96	1	.99	.31	-	-	-	-	.68	.89	.86	.65	.31	.11	-	-	-	-	.81	.42	.86	.81	1	.19	-	-	-	-
↓ 3	.86	1	.94	1	1	.25	-	-	-	-	.68	.76	.81	.38	.21	.11	-	-	-	-	.75	.39	.83	.63	1	.15	-	-	-	-
↓ 4	.86	1	.90	1	1	.21	-	-	-	-	.56	.62	.80	.12	.19	.10	-	-	-	-	.71	.28	.78	.51	1	.10	-	-	-	-
↓ 5	.85	1	.84	1	1	.19	-	-	-	-	.26	.32	.75	.05	.19	.08	-	-	-	-	.67	.20	.75	.38	1	.10	-	-	-	-
5%	.98	1	.89	1	.97	.41	-	-	-	-	.70	.80	.67	.60	.41	.10	-	-	-	-	.93	.65	.92	.81	1	.12	-	-	-	-
10%	.97	1	.87	1	.96	.39	-	-	-	-	.61	.62	.59	.51	.15	.08	-	-	-	-	.92	.63	.90	.65	.99	.10	-	-	-	-
15%	.96	1	.86	1	1	.33	-	-	-	-	.55	.31	.51	.43	.13	.06	-	-	-	-	.89	.63	.90	.61	1	.10	-	-	-	-
20%	.96	1	.81	1	.99	.29	-	-	-	-	.44	.08	.52	.34	.12	.05	-	-	-	-	1	.60	.89	.59	1	.09	-	-	-	-
Avg	**.92**	**1**	**.88**	**1**	**.99**	**.31**	-	-	-	-	**.58**	**.60**	**.71**	**.41**	**.23**	**.09**	-	-	-	-	**.84**	**.42**	**.86**	**.66**	**1**	**.12**	-	-	-	-

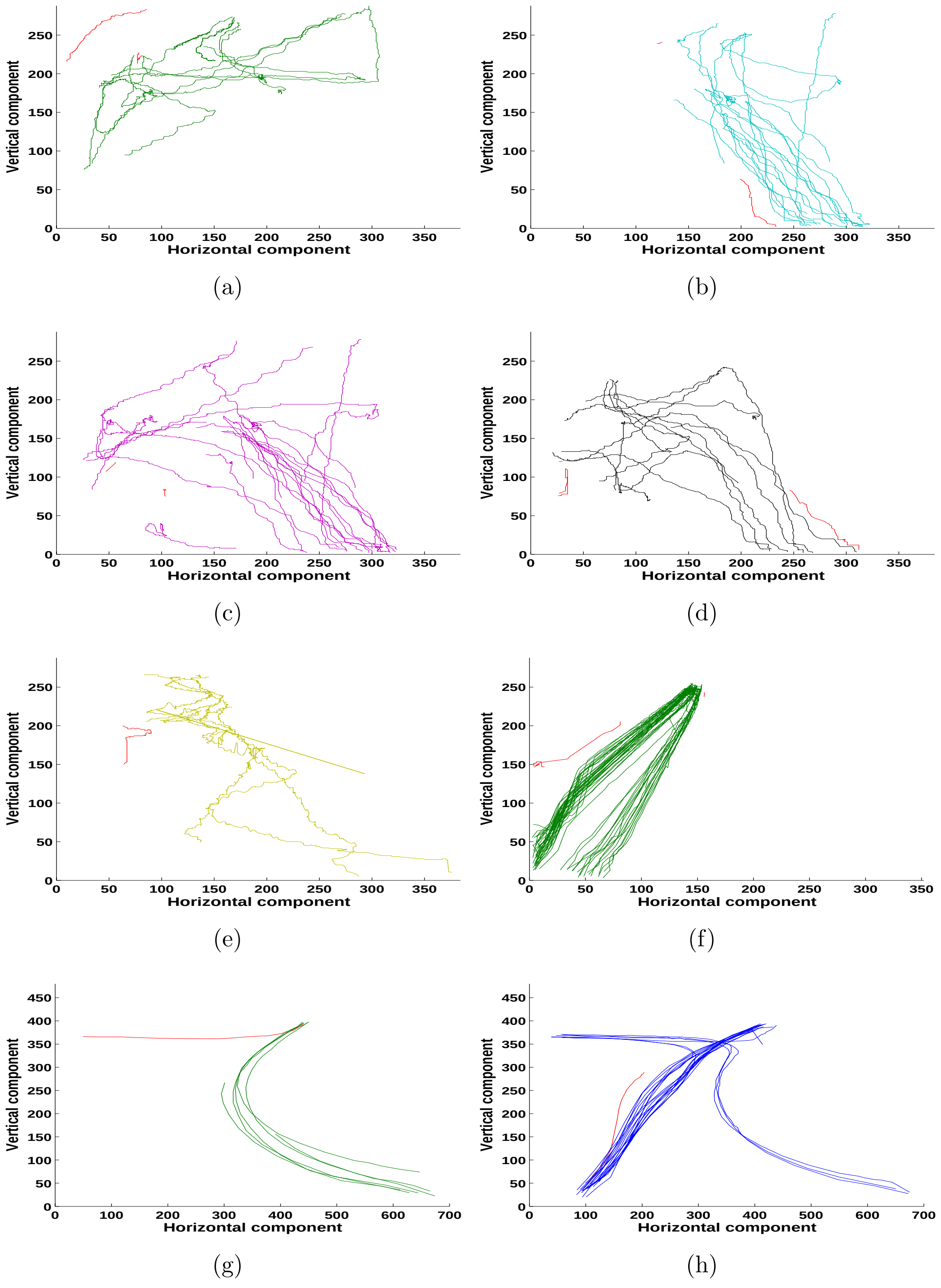


Figure 6.11: Sample outlier detection improvements by applying the proposed outlier detection criteria on the SOM clustering results (red: new outliers identified in a cluster).

for cluster merging is set to $h_m^{d_m} + 0.1(h_m^{d_m})$ (See Fig. 6.9).

As a baseline test, initially, we compare the performance of the MFTC approach with the standard K-mean clustering [58]. Afterward, we compare it with two state-of-the-art trajectory clustering algorithms, based on Self-Organizing Maps (SOM) [13] and on Trajectory Directional Histograms (TDH) [16].

6.3.3 Discussion

Figure 6.10 shows the comparison of the MFTC approach and K-mean. The graph shows the average P_c and R_c results for the four test sequences. The results show that the MFTC approach has 3% and 13% higher precision and recall for $S1$. For $S2$, it has 7% and 9% higher precision and recall compared to K-mean. Similarly, its P_c and R_c results are better by 1% and 6% for $S3$. Finally, for $S4$ the MFTC approach has higher P_c and R_c results by 2% and 3%, respectively. Better P_c and R_c results for all sequences show that MFTC is better in trajectory clustering both in terms of exactness and completeness. The subsequent discussion focuses on the performance comparison of the MFTC approach with SOM and TDH based approaches.

Figure 6.12 compares the clustering results of the MFTC approach, SOM and TDH on the four test sequences ($S1$, $S2$, $S3$ and $S4$). To facilitate the visualisation, the trajectory clusters are shown in 3D, where each vertical layer corresponds to a separate cluster. In each plot, the top-most layer corresponds to the detected outliers. In the figure, the first column shows the ground truth, second, third and fourth columns show the results of MFTC, SOM and TDH, respectively. Furthermore, the P_c and R_c results for the three approaches are also compiled in Table 6.5 (hereinafter "the Table"). The Table is organised in four sections corresponding to four datasets (separated by double horizontal lines). Each section is further divided into two (separated by single horizontal line), where the upper-half compiles the clustering results on original and down-sampled version of input trajectories and the bottom-half compiles the clustering results on the noisy version of input trajectories. The overall results are also compiled in the last row of each section by taking average of P_c and R_c of each approach on original, down-sampled and noisy trajectories. The detailed discussion on the results is provided in rest of this section.

In Fig. 6.12 (b-d), for the first group of trajectories in $S1$, unlike the MFTC approach, SOM and TDH generate clusters which contain additional trajectories that either starts from the centre or bottom regions on the image plane. For the second group, the clustering results for the MFTC approach outperform SOA and TDH. In particular, the clustering results of TDH and SOM for the third group are contaminated by additional trajectories that either start from the bottom or the centre-left image plane regions. From the Table it is possible to notice that for the first three clusters, the MFTC approach not only clusters the trajectories accurately (R_c values) compared to SOM and TDH, but also avoids the additions of other trajectories into a group (P_c values). However, for C_4, the accuracy of TDH is 10% better than that of the MFTC approach, though it adds more outliers and trajectories belonging to other groups. The precision values are important as they give an indication of how a particular algorithm treated the outliers: the larger P_c, the higher the relevant information contained and therefore fewer outliers are part of a cluster. P_c shows that in C_4 TDH has 20% extra trajectories compared to the MFTC approach. The Table also shows that, compared to SOM and TDH, the MFTC approach is more robust to missing data and noisy trajectories. On average, the overall degradation in accuracy (recall value) is 3% for the MFTC

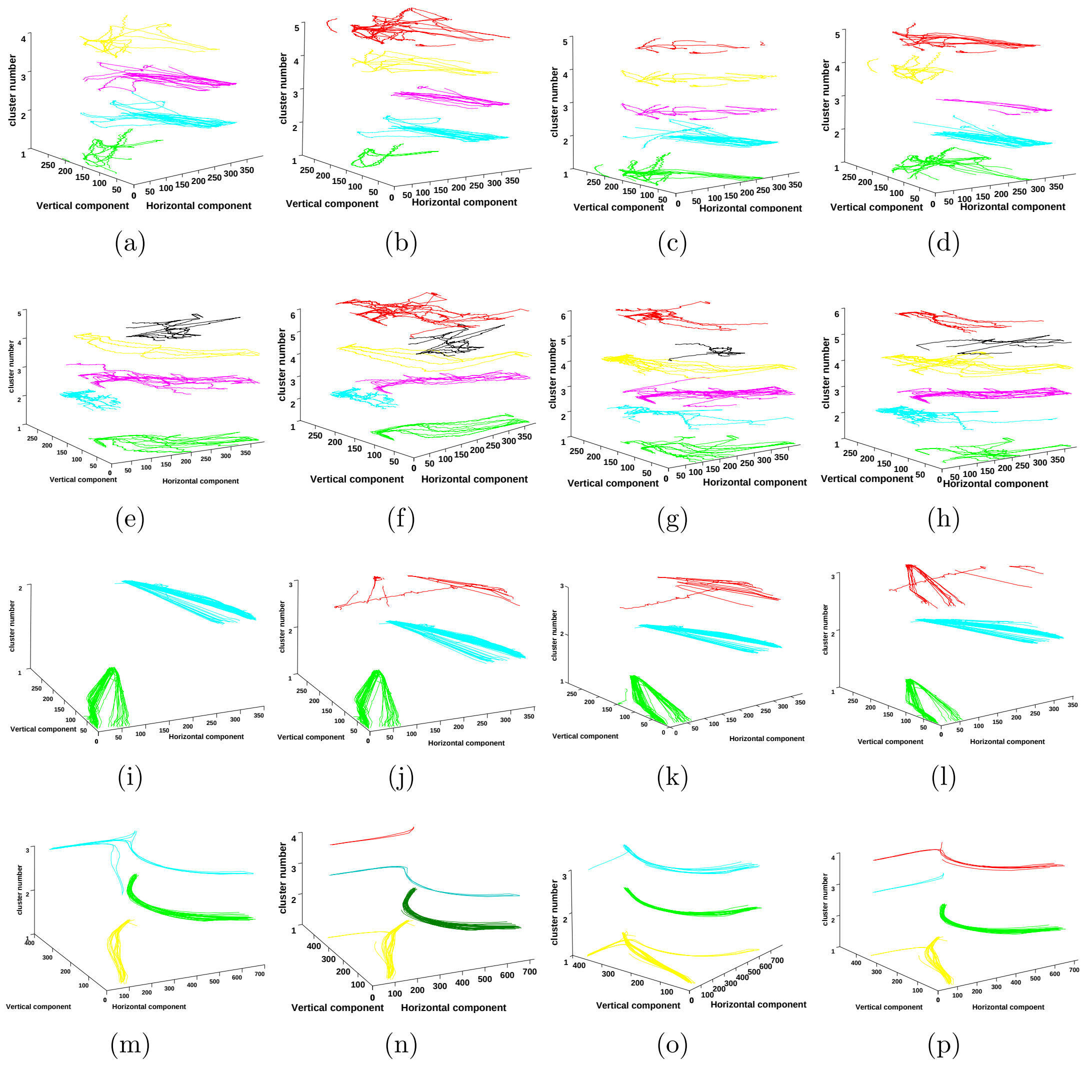


Figure 6.12: Comparison of trajectory clustering results for the test sequences $S1$ (first row), $S2$ (second row), $S3$ (third row) and $S4$ (fourth row). First column: ground truth. Second column: results of the MFTC approach. Third column: SOM results. Fourth column: TDH results. The top-most layer ('red') shows the detected outliers for each approach.

approach when the trajectory is reduced by 5 sampling steps. On the other hand, the degradations for SOM and TDH are 8% and 12.5%, respectively. Similarly, for noisy trajectories, on average the MFTC approach degrades by 9.5% when 20% noise is added to the trajectories. For SOM and TDH, the degradation is 13.5% and 20.5%, respectively. The complexity of the trajectories affects SOM negatively. The reduced sampling and noisy trajectories affect TDH the most, as its performance degrades substantially compared to the other two approaches.

For $S2$, Fig. 6.12 (g-h) shows that for the first type of motion the clustering results of both SOM and TDH contain extra trajectories that either start from the bottom or top regions. Similar results can be observed for other clusters. On average, the MFTC approach (Fig. 6.12 (f)) outperforms SOM and TDH by 32.6% and 31.8%, respectively. The robustness tests also reveal that the MFTC approach efficiently clusters the trajectories compared to SOM and TDH, especially in C_2 and C_5 for SOM, and in C_4 and C_5 for TDH.

For $S3$, Fig. 6.12 (j) shows that the MFTC approach correctly identifies the clusters. There are noticeable errors in the SOM results (Fig. 6.12 (k)), for example the left-most outlier trajectory is considered as part of the 1^{st} cluster. Similarly, for TDH (Fig. 6.12 (l)), there are several trajectories that should be part of the 1^{st} cluster, but are treated as outliers. For C_1, the precision of the MFTC approach is 96%. Both SOM and TDH have a lower precision for the same cluster. SOM tends to accept outliers as part of a normal cluster compared to the MFTC approach and TDH. For C_2 the precision and recall values for the MFTC approach are 90% and 100%, respectively. For SOM, 9% of the trajectories in C_2 are misclassified, whereas for TDH the misclassification is 21%. As for the robustness test results, the MFTC approach misclassifies 4% of trajectories in C_1 when the sampling rate is reduced from 2 to 5 steps and all the trajectories are successfully grouped together for C_2. By comparison, the clustering results of SOM degrade by 12% and 2% for C_1 and C_2, respectively. Likewise, the degradation of the results for TDH is 32% and 8% for C_1 and C_2. In the case of the maximum added noise, +20%, the trajectory classification results are reduced by 9% in C_1 and 0% in C_2 for the MFTC approach.

However, this reduction is 13%, and 10% in C_1, and C_2 for SOM, and 23%, and 37% in C_1, and in C_2 for TDH. The MFTC approach is more robust to outliers up to $\downarrow 4$ (subsampled by a factor 4) and +15% (15% added noise) in the generation of the final clusters.

Figure 6.12 (n-p) shows the results for $S4$. SOM could not detect any outlier. If the proposed outlier detection criteria are applied on the SOM results, its performance can be improved. Figure 6.11 shows few examples where the proposed outlier detection criteria has improved the clustering results. The Table shows that although for C_1 and C_2, the MFTC approach identified the patterns correctly, for C_3 its performance is poorer due to varying behaviour of the trajectories in that cluster and also to a strong similarity between the trajectories and the outliers. However, the comparison with the other two approaches reveals that the results of the MFTC approach are better by 35% for C_2 and 43% for C_3 over SOM, and by 35% and 31% for C_1 and C_2 over TDH. The robustness test results for $S4$ show that the MFTC approach does not modify its results for C_1 and C_2, in case of sampling rate reduction. However, there is a 21% degradation for C_3. On the other hand, the degradation for SOM is 68%, 60%, 4% in C_1, C_2 and C_3, respectively. Furthermore, this degradation for TDH is 45%, 62% and 11%, respectively. Moreover, for the noisy data, the MFTC approach is more stable and the only reduction in performance is 14% for C_3. In the same conditions, there are relevant classification errors for SOM (92%, 31%, 19%) and TDH (5%, 41%, 3%).

In summary, MFTC outperforms the other approaches in three aspects: (a) due to the use of multiple complementary features, the approach is successfully able to separate two trajectories which partially overlap in time and space for small duration but belong to different clusters, (b) the approach is better in highlighting the outlier trajectories due to trajectory- and cluster-level analysis, and (c) the approach is more stable to miss detections and to noisy observations.

Chapter 7

Conclusions

7.1 Summary of achievements

In this thesis, we addressed an important problem of video understanding in multi-camera setups, without imposing any constraints on placement of cameras and motion of the objects. We provided a framework that used trajectory segments from multiple cameras and summarised the results in the form of common (or normal) patterns and outliers. To this end, we assumed that scene structures affected the behaviour of moving objects; therefore, from the analysis of trajectories, we learned the common motion paths and any trajectory that deviated significantly from the common path is considered as an outlier.

However, to understand the information from multiple cameras, it is needed to know the relative configuration of the cameras. This knowledge is useful in establishing object correspondence across multiple views. The object association is required to reconstruct complete object trajectories across the entire scene. These trajectories are then analysed together for summarisation. Major contributions of this thesis are that it uses trajectory segments observed in multiple cameras to:

(a) learn the configuration of a camera network,

(b) establish object correspondence across multiple views,

(c) identify common patterns and outliers for activity analysis.

To learn the configuration of the camera network, we considered both partially overlapping and non-overlapping cameras. For partially overlapping cameras, we used existing homography-based approach; however, for non-overlapping cameras, we proposed an algorithm that allowed us to relax the traditional linearity or smooth turn constraint on the motion of objects. The algorithm recovers the position and the rotation of each camera based on two main steps: the estimation of the unobserved trajectory in regions not covered by the cameras' fields of view and the estimation of the relative rotations of the cameras. Kalman filtering is initially employed on the available trajectory segments to extrapolate their value in the unobserved regions, with the modification that the forward motion model provides measurements for the backward trajectory estimation and vice versa. This helps in improving the accuracy of the trajectory estimation in the unobserved regions even if there is a sharp turn. The relative rotation of the cameras is then obtained by using the estimated and the observed exit and entry point information in each camera field of view. Refinements are then applied using bundle adjustment. The proposed algorithm is tested on a real and two simulated sequences and compared with existing approach. The comparison showed that the proposed approach not only efficient in localising the camera network but also requires at least 1/3 less processing time compared to existing approach.

For object correspondence, we proposed an algorithm that works in an unsupervised fashion and does not impose any constraints on the camera placement. Local trajectory segments from each camera are projected on a common ground plane. Multiple spatio-temporal features are then analysed to find the degree of proximity among the trajectories. The matching is validated via ground plane to image plane reprojections. The proposed approach generates a complete trajectory belonging to a physical object in an unsupervised way without requiring learning (a computationally complex process) of motion parameters. We tested the performance of the proposed approach on two real world datasets and found that it outperforms state-of-the-art approaches by at least of 4% in precision and 8% in recall. Furthermore, it is also noticed that the proposed algorithm is efficient in establishing correspondence not only for regions with sparse trajectory segments, but also for regions with dense trajectory segments; on the other hand, other approaches work mainly for regions with sparse trajectory segments.

Finally, we presented a framework for scene context learning and outlier detection by clustering video object trajectories. The input trajectories are transformed into distinct feature spaces to represent the complementary characteristics of motion patterns. Next, Mean-shift was used to estimate clusters in each feature space and adjacent clusters in a feature space were merged to refine the initial results. A fuzzy membership of a trajectory to the final clusters was estimated, and crisp clusters were then obtained based on the maximum membership using the information from all the feature spaces. Finally, the clusters with only few associated trajectories and trajectories far from the clusters centre were considered outliers. The proposed approach was validated on standard datasets and compared with state-of-the-art approaches. The results showed that the proposed algorithm is not only able to cluster trajectories for indoor e.g., shopping mall and outdoor e.g., highway sequences with higher precision and recall but also more stable to missed detections and noisy trajectories.

7.2 Future work

In future, we plan to extend this work in the following directions: (a) control point's selection, (b) complex motion modelling, (c) feature space's selection and (d) contextual outlier's detection. The details of possible extensions are as follows:

(a) The choice of control points for both self-calibration (in the case of single camera) and for homography (in case of partially overlapping cameras) is completely user dependent. However, it is desirable to define a mechanism to assess the quality of these control points [92]. Furthermore, it is possible to have scenarios were it is difficult to select reliable control points manually e.g., the regions where there is no structural information available. To completely automate the process, it would be interesting to use object tracks as control points. To this end it is possible to derive transformation matrices by selecting (or identifying) few objects in each view [93]

(b) The proposed unobserved trajectory estimation algorithm primarily assumes that an object takes one sharp turn in unobserved region between a pair of cameras. However, if a complex environment is observed by a few non-overlapping cameras, it is possible that the object may take more than one sharp turns in unobserved regions. To this end, the contextual information (e.g., map of the environment) can be used within motion models to handle this problem [94].

(c) While clustering using multiple features, we consider each individual feature equally good. Therefore, during cluster fusion we do not give any preference to a particular feature space based on its importance. However, by employing cluster validity [95] before fusion would help in assigning dynamic weights to feature based on its ”goodness” for a particular application data.

(d) This thesis only considered point anomalies i.e., sparse trajectories that have attribute different from the rest of trajectories. However, another dimension of the work can be attained by considering contextual anomalies i.e., a trajectory would be anomalous in a specific context, but not otherwise [96].

Bibliography

[1] N. Anjum and A. Cavallaro. *Multi-camera calibration and global trajectory fusion.* Intelligent Video Surveillance Systems and Technology, Auerbach Publications, 2010.

[2] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *Technical report(00-729), University of Southern California, Computer Science Department*, pages 28–34, 2000.

[3] E. Polat and M. Ozden. A nonparametric adaptive tracking algorithm based on multiple feature distributions. *IEEE Trans. on Multimedia*, 8(6):1156–1163, 2006.

[4] W. Qu and D. Schonfeld. Real-time decentralized articulated motion analysis and object tracking from videos. *IEEE Trans. on Image Processing*, 16(8):2129–2138, 2007.

[5] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. Knight: A real time surveillance system for multiple overlapping and non-overlapping cameras. In *Proc. of Intl. Conf. on Multimedia and Expo*. Baltimore, USA, 2003.

[6] D. Makris, T. Ellis, and J. Black. Bridging the gaps between cameras. In *Proc. of Intl. Conf. on Computer Vision and Pattern Recognition.* Washington, DC, USA, 2004.

[7] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*. Washington-DC, USA, 2004.

[8] Y. Sheikh and M. Shah. Trajectory association across multiple airborne cameras. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(2):361–367, 2008.

[9] G. Kayumbi, N. Anjum, and A. Cavallaro. Global trajectory reconstruction from distributed visual sensors. In *Proc. of ACM / IEEE Intl. Conf. on Distributed Smart Cameras*. California, USA, 2008.

[10] F. Amasyali and S. Albayrak. Fuzzy c-means clustering on medical diagnostic systems. In *Proc. of Intl. twelfth Turkish Symp. on Artificial Intelligence and Neural Networks (TAINN2003)*. Canakkale, Turkey, 2003.

[11] G. Antonini and J. Thiran. Counting pedestrians in video sequences using trajectory clustering. *IEEE Trans. on Circuit and Systems for Video Technology*, 16(8):1008–1020, 2006.

[12] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proc. of Intl. Conf. on Knowledge Discovery and Data Mining*. San Diego, CA, USA, 1999.

[13] S. Khalid and A. Naftel. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. In *Proc. of third ACM Intl. workshop on Video Surveillance and Sensor Networks*. Singapore, 2005.

[14] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Trans. on Multimedia Systems*, 12(3):227–238, 2006.

[15] Z. Fu, W. Hu, and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Proc. of IEEE Intl. Conf. on Image Processing*. Genova, Italy, 2005.

[16] X. Li, W. Hu, and W. Hu. A coarse-to-fine strategy for vehicle motion trajectory clustering. In *Proc. of Intl. Conf. on Pattern Recognition*. Hong Kong, China, 2006.

[17] F. Bashir, A. Khokhar, and D. Schonfeld. Real-time motion trajectory-based indexing and retrieval of video sequences. *IEEE Trans. on Multimedia*, 9:58–65, 2007.

[18] S. Andradottir. A new algorithm for stochastic optimization. In *Proc. of 22nd Conf. on Winter simulation*, pages 364 – 366. USA, 1990.

[19] O. Faugeras. Stratification of 3-dimensional vision: Projective, affine, and metric representations. *JOSA-A*, 12:465–484, 1995.

[20] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second ed. Cambridge University Press, UK, 2004.

[21] P. R. S. Mendonca. *Multiview geometry: Profiles and self-calibration*. Ph.D. dissertation, University of Cambridge, Cambridge, UK, 2001.

[22] B. Triggs. Autocalibration and the absolute quadric. In *Proc. of IEEE Intl. Conf. on Computer vision and pattern recognition*. Sanjuan, Puerto Rico, 1997.

[23] M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *Intl. J. Comput. Vis.*, 32(1):7–25, 1999.

[24] L. D. Agapito, E. Hayman, and I. Reid. Self-calibration of rotating and zooming cameras. *Intl. J. Comput. Vis.*, 22(11):1330–1334, 2000.

[25] Y. Seo and K. Hong. About the self-calibration of a rotating and zooming camera: Theory and practice. In *Proc. of IEEE Intl. Conf. on Computer Vision.* Kerkyra, Greece, 1999.

[26] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc of IEEE Intl. Conf. on Computer vision and pattern recognition.* FL, USA, 1986.

[27] O.D.Faugeras and G.Toscani. The calibration problem of stereo. In *Proc of IEEE Intl. Conf. on Computer vision and pattern recognition.* FL, USA, 1986.

[28] R. Collins and R. Beveridge. Matching perspective views of coplanar structures using projective unwarping and similarity matching. In *Proc. of Conf. on Computer Vision and Pattern Recognition.* Washington, USA, 1994.

[29] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proc. of Conf. on Computer Vision and Pattern Recognition.* Santa Barbara, CA, USA, 1998.

[30] N. Anjum and A. Cavallaro. Single camera calibration for trajectory-based behavior analysis. In *Proc of IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance.* London, UK, 2007.

[31] K. Tieu, G. Dalley, and W. E. L. Grimson. Inference of non-overlapping camera network topology by measuring statistical dependence. In *Intl. Conf. Comput. Vis.* Beijing, China, 2005.

[32] I. N. Junejo, X. Cao, and H. Foroosh. Autoconfiguration of a dynamic nonoverlapping camera network. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 37:803–816, 2007.

[33] R. B. Fisher. Self-organization of randomly placed sensors. In *Proc. of Eur. Conf. on Computer Vision*. Copenhagen, Denmark, 2002.

[34] N. Anjum, M. Taj, and A. Cavallaro. Relative position estimation of non-overlapping cameras. In *Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*. Honolulu, USA, 2007.

[35] D. Marinakis, G. Dudek, and D. J. Fleet. Learning sensor network topology through monte-carlo expectation maximization. In *Proc. of Intl. Conf. on Robotics and Automation*. Barcelona, Spain, 2005.

[36] V. Kettnaker and R.Zabih. Bayesian multi-camera surveillance. In *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition*. Fort Collins, CO, USA, 1999.

[37] A. R. Dick and M. J. Brooks. A stochastic approach to tracking objects across multiple cameras. *Book Series Lecture Notes in Computer Science*, 3339/2005:160–170, 2004.

[38] T. Huang and S. Russell. Object identification in a bayesian context. In *Proc. of Intl Joint Conf. on Artificial Intelligence*. NAGOYA, Japan, 1997.

[39] X. Wang, K. T. Ma, G. W. Ng, and W. E. L. Grimson. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*. Anchorage, Alaska, USA, 2008.

[40] F. Porikli. Trajectory pattern detection by HMM parameter space features and eigenvector clustering. In *Proc. of 8th European Conf. on Computer Vision (ECCV)*. Prague, Czech Republic, 2004.

[41] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion

time-series data. In *Proc. of IEEE Conf. on Computer vision and pattern recognition*. Madison, Wisconsin, 2003.

[42] A. D. Wilson and A. F. Bobick. Recognition and interpretation of parametric gesture. In *Proc. of IEEE 6th Intl. Conf. on Computer Vision*. Bombay, India, 1998.

[43] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[44] D. Chudova, S. Gaffney, and P. Smyth. Probabilistic models for joint clustering and time warping of multi-dimensional curves. In *Proc. of 19th Conf. on Uncertainty in Artificial Intelligence*. Acapulco, Mexico, 2003.

[45] N. Sumpter and A. J. Bulpitt. Learning spatio-temporal patterns for predicting object behaviour. In *Proc. of British Conf. on Machine vision*. Southampton, UK, 1998.

[46] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615, 1996.

[47] J. Baras and S. Dey. Combined compression and classification with learning vector quantization. *IEEE Trans.on Information Theory*, 45(6):1911–1920, 1999.

[48] W. Hu, D. Xie, T. Tan, and S. Maybank. Learning activity patterns using fuzzy self-organizing neural network. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 34(3):334–352, 2004.

[49] F. Bashir, A. Khokhar, and D. Schonfeld. Object trajectory-based activity

classification and recognition using hidden markov models. *IEEE Trans. on Image Processing*, 16:1912–1919, 2007.

[50] F. Bashir, A. Khokhar, and D. Schonfeld. Segmented trajectory based indexing and retrieval of video data. In *Proc. of Intl. Conf. on Image Processing.* Barcelona, Catalonia, Spain, 2003.

[51] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999.

[52] H. Wilson, B. Boots, and A. Millward. A comparison of hierarchical and partitional clustering techniques for multispectral image classification. In *Proc. of IEEE Intl. Geoscience and Remote Sensing Symp.* Totonto, Canada, 2002.

[53] D. Buzan, S. Sclaroff, and G. Kollios. Extraction and clustering of motion trajectories in video. In *Proc. of IEEE Intl. Conf. on Pattern Recognition (ICPR).* USA, 2004.

[54] M. Li, C. Wu, Z. Han, and Y. Yue. A hierarchical clustering method for attribute discretization in rough set theory. In *Proc. of Intl. Conf. on Machine Learning and Cybernetics.* Shanghai, China, 2004.

[55] D. Biliotti, G. Antonini, and J. Thiran. Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences. In *Proc. of IEEE Workshop on Motion and Video Computing.* Washington, USA, 2005.

[56] A. K. Jain, M. N. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[57] J. Melo, A. Naftel, A. Bernardino, and J. Victor. Retrieval of vehicle trajec-

tories and estimation of lane geometry using non-stationary traffic surveillance cameras. In *Proc. of Advanced Concepts for Intelligent Vision Systems.* Brussels, Belgium, 2004.

[58] G. A. F. Seber. *Multivariate Observations.* John Wiley and Sons, New York, 1984.

[59] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Trans. Systems Man Cybernet - Part B*, 35(3):397–408, 2005.

[60] C. Piciarelli and G. L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27(15):1835–1842, 2006.

[61] T. Kwok, R. Smith, S. Lozano, and D. Taniar. Parallel fuzzy c-means clustering for large data sets. In *Proc. of 8th Intl. Euro-Par Conf. on Parallel Processing.* Paderborn, Germany, 2002.

[62] Y. Zhou, S. Yan, and T. Huang. Detecting anomaly in videos from trajectory similarity analysis. In *Proc. of Intl. Conf. on Multimedia and Expo.* Beijing, China, 2007.

[63] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Trans. on Multimedia Systems*, 12(3):227–238, 2006.

[64] X. Li, J. Han, S. Kim, and H. Gonzalez. Roam: Rule- and motifbased anomaly detection in massive moving object data sets. In *Proc. of the Intl Conf. on Data Mining.* Minnesota, USA, 2007.

[65] I. C. J. Kang and G. Medioni. Tracking people in crowded scenes across multiple cameras. In *Proc of Asian Conf. on Computer Vision.* Jeju Island, Korea, 2004.

[66] N. Anjum and A. Cavallaro. Trajectory association and fusion across partially overlapping cameras. In *Proc. of IEEE Intl. Conf. Advanced Video and Signal Based Surveillance.* Genoa, Italy, 2009.

[67] D. Marinakis and G. Dudek. Self-calibration of a vision-based sensor network. *Image and vision computing*, 27(1-2):116–130, 2009.

[68] R. Pflugfelder and H. Bischof. Localization and trajectory reconstruction in surveillance cameras with non-overlapping views. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(4):709–721, 2010.

[69] S.J.Maybank and O.D.Faugeras. A theory of self-calibration of moving camera. *Intl. Journal of Computer Vision*, 8(2):123–151, 1992.

[70] A. Basu. Active calibration: alternative strategy and analysis. In *Proc. of Proc. of IEEE Conf. on Computer Vision and Pattern Recognition.* New York, USA, 1993.

[71] Q. Ji and S. Dai. Self-calibration of a rotating camera with a translational offset. *IEEE Trans. on Robotics and Automation*, 20(1):1–14, 2004.

[72] N. Anjum and A. Cavallaro. Single camera calibration for trajectory-based behavior analysis. In *Proc. of IEEE Intl. Conf. Advanced Video and Signal Based Surveillance.* London, UK, 2007.

[73] R. A. Johnson and D. W. Wichern. *Applied multivariate statistical analysis.* Fifth ed. Pearson Education, UK, 2006.

[74] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. on ASME. Series D: Journal of Basic Engineering*, 82:3545, 1960.

[75] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment-a

modern synthesis. *Springer Lecture Notes on Computer Science*, 1883:298–375, 2000.

[76] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar. Distributed localization of networked cameras. In *Proc of IEEE Intl. Conf. on Information Processing in Sensor Networks*. Nashville, TN, USA, 2006.

[77] C. Yinghao, C. Wei, K. Huang, and T. Tan. Continuously tracking objects across multiple widely separated cameras. *Book Series Lecture Notes in Computer Science*, 4843:843–852, 2007.

[78] M. Taj, E. Maggio, and A. Cavallaro. Multi-feature graph-based object tracking. In *CLEAR, Springer LNCS 4122*, pages 190–199. Southampton, UK, 2006.

[79] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Second ed. Cambridge University Press, UK, 2004.

[80] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *Proc. of sixth British Conf. on Machine vision.* Birmingham, UK, 1996.

[81] R. Rosales and S. Sclaroff. Trajectory guided tracking and recognition of actions, computer science department. university of boston. Technical Report 1999-002, 1999. (online: http://citeseer.ist.psu.edu/rosales99trajectory.html, last accessed: 28 Feb, 2010).

[82] A. R. Webb. *Statistical Pattern Recognition, 2nd Edition.* John Wiley and Sons Ltd., 2002.

[83] Y. Zeng and J. Starzyk. Statistical approach to clustering in pattern recognition. In *Proc. of 33rd southeastern Symp. on System Theory.* Athens, Ohio, USA, 2001.

[84] Y. Gdalyahu, D. Weinshall, and M. Werman. Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.

[85] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603 – 619, 2002.

[86] D. Comaniciu and P. Meer. Distribution free decomposition of multivariate data. *Trans. on Pattern Analysis and Applications*, 2(1):22 – 30, 1999.

[87] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. of IEEE Intl Conf. on Computer Vision.* Vancouver, Canada, 2001.

[88] H. Wang and D. Suter. False-peaks-avoiding mean shift method for unsupervised peak-valley sliding image segmentation. In *Proc. of Intl. Conf. on Digital Image Computing: Techniques and Applications.* Sydney, Australia, 2003.

[89] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In *Proc. of the workshop on application and computer vision.* Princeton, New Jersey, USA, 1998.

[90] E. Keogh. Exact indexing of dynamic time warping. In *Proc. of Intl. Conf. on Very Large Data Bases.* Hong Kong, China, 2002.

[91] R. Kasturi. Performance evaluation protocol for face, person and vehicle detection tracking in video analysis and content extraction. *VACE-II. University of South Florida, USA*, 2006.

[92] D. S. Kumar and C. V. Jawahar. Robust homography-based control for camera

positioning in piecewise planar environments. In *Indian Conference on Computer Vision, Graphics and Image Processing*. Madurai, India, 2006.

[93] B. Bose and E. Grimson. Plane rectification by tracking moving objects. In *Proc. of the workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. Nice, France, 2003.

[94] B. Pannetier, J. Dezert, and E. Pollard. Improvement of multiple ground targets tracking with gmti sensor and fusion of identification attributes. In *IEEE Aerospace Conference*. Montana, USA, 2008.

[95] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3):645–678, 2005.

[96] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys*, 41(3):1–58, 2009.