Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance

by

Frederick Tung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Video surveillance systems are playing an increasing role in preventing and investigating crime, protecting public safety, and safeguarding national security. In a typical surveillance installation, a human operator has to constantly monitor a large array of video feeds for suspicious behaviour. As the number of cameras increases, information overload makes manual surveillance increasingly difficult, adding to other confounding factors like human fatigue and boredom.

The objective of an intelligent vision-based surveillance system is to automate the monitoring and event detection components of surveillance, alerting the operator only when unusual behaviour or other events of interest are detected. While most traditional methods for trajectory-based unusual behaviour detection rely on low-level trajectory features, this thesis improves a recently introduced approach that makes use of higher-level features of intentionality. Individuals in a scene are modelled as intentional agents instead of simply objects. Unusual behaviour detection then becomes a task of determining whether an agent's trajectory is explicable in terms of learned spatial goals. The proposed method extends the original goal-based approach in three ways: first, the spatial scene structure is learned in a training phase; second, a region transition model is learned to describe normal movement patterns between spatial regions; and third, classification of trajectories in progress is performed in a probabilistic framework using particle filtering. Experimental validation on three published third-party datasets demonstrates the validity of the proposed approach.

Acknowledgements

I would like to thank Hannah Dee and Andrew Naftel for generously providing copies of their trajectory datasets. I would also like to thank Claudio Piciarelli and colleagues for making their synthetic dataset available online, and X.C. He and N.H.C. Yung for making a MATLAB implementation of their CSS corner detector available online.

This work is sponsored by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Contents

List of Tables **vii**

List of Figures **ix**

List of Symbols **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem definition 2

1.3 Thesis objectives and organization 3

2 Related Work **5**

2.1 Trajectory-based methods for unusual behaviour detection 5

2.2 Unusual behaviour detection in dense crowds 12

2.3 Summary 14

3 Proposed Method **16**

3.1 Contribution details and broader significance 16

3.2 Modelling the scene structure 18

3.2.1 Overview 18

3.2.2 Fitting GMMs using Expectation Maximization 19
3.2.3 Determining an appropriate number of mixture components 20
3.2.4 Detecting turning points 21
3.2.5 Examples 22
3.3 Learning a region transition model 22
3.4 Classifying observed trajectories 26
3.4.1 Particle filtering 26
3.4.2 Inference details 32
3.5 Summary 36

4 Experimental Results **38**
4.1 Methodology 38
4.2 Piciarelli *et al.*'s synthetic dataset [42] 39
4.3 Naftel and Khalid's laboratory dataset [39] 43
4.4 Dee and Hogg's carpark dataset [9, 10, 11] 44
4.5 Discussion and limitations 48

5 Conclusion and Future Work **50**

References **51**

List of Tables

2.1 Summary of recurring issues in trajectory-based unusual behaviour detection. A checkmark indicates successful inclusion in the work. 15

3.1 An iteration of the Bayes filtering algorithm [49] 27

3.2 An iteration of the particle filtering algorithm [44, 49] 28

3.3 Low-variance resampling method for particle filtering [49] 31

3.4 An iteration of the proposed inference method using particle filtering . . . 35

List of Figures

3.1 Sample corner detection results on trajectories from Dee and Hogg's carpark dataset [9, 10, 11]. Detected corners are marked with green asterisks. . . . 23

3.2 Sample learned scene models for training sets from Piciarelli *et al.*'s synthetic dataset [42]. Entry, turning, and exit points are indicated by blue, green, and red points respectively. Asterisks denote the component means of the entry, turning, and exit GMMs. 24

3.3 Illustrative example of the particle filtering algorithm (see text for details): (a) initial particles at time t - 1; (b) particles after prediction using state transition model; (c) particle weights based on observation likelihood; (d) final particles after weighted resampling. 29

3.4 Illustration of low-variance resampling, adapted from [49]. 30

3.5 Summary flowchart of state transition model 33

4.1 ROC curves for Piciarelli *et al.*'s synthetic dataset [42], varying the number of normal trajectory clusters from 1 to 6 (a-f, respectively). 41

4.2 ROC curves for Piciarelli *et al.*'s synthetic dataset [42], varying the number of normal trajectory clusters from 7 to 10 (a-d, respectively). 42

4.3 Examples of false negatives in Piciarelli *et al.*'s synthetic dataset [42], with two missed unusual trajectories highlighted in red and highly similar normal trajectory clusters highlighted in yellow. Trajectory entry and exit points are indicated by blue and red points, respectively. 43

4.4 Overlays of the trajectories in Naftel and Khalid's laboratory dataset [39]; the top four subplots highlight the pre-planned trajectory classes and the fifth subplot highlights the unusual trajectories. 45

4.5 ROC curve for Naftel and Khalid's laboratory dataset [39]. 46

4.6 Overlay of the trajectories in Dee and Hogg's carpark dataset [9, 10, 11], with the unusual trajectories highlighted in red. 47

4.7 ROC curve for Dee and Hogg's carpark dataset [9, 10, 11]. 48

List of Symbols

d_t	Current path segment end position at time t
e_t^0	Previous path segment at time t
e_t^1	Current path segment at time t
κ	Curvature
m	Sample mean
M	Number of particles
μ_i	Mean of Gaussian component i in a GMM
p_{ij}	Probability that data point x_j is generated by Gaussian component i in GMM
ψ	Angle subtended by the tangent of a curve with the x-axis
Q	Measurement noise covariance in probabilistic filtering
R	Process noise covariance in probabilistic filtering
s	Arclength
s_t	Position at time t
σ	Standard deviation
S	Sample covariance
Σ_i	Covariance of Gaussian component i in a GMM

t	Time
u	Direction vector
v_t	Speed at time t
w_i	Weight for component/particle i
x	Generic data point generated by a mixture distribution
x_t	State at time t, used in probabilistic reasoning over time
x_t^i	Particle i at time t (in particle filtering, a particle is an instantiation of the state at time t)
X_t	Set of particles at time t
$(x(s), y(s))$	Curve (in spatial plane) in parametric form with arclength parameter s
$\dot{x}(s)$	First derivative of $x(s)$
$\ddot{x}(s)$	Second derivative of $x(s)$
$x_s(s, \sigma)$	Smoothed $x(s)$ - smoothed by a Gaussian with standard deviation σ
z_t	Measurement at time t

Chapter 1

Introduction

1.1 Motivation

Intelligent surveillance is an application of computer vision that has attracted much research interest in recent years. Video surveillance systems have played an increasing role as tools for preventing and investigating crime, protecting public safety, and safeguarding national security. However, the operation of these systems presents significant challenges.

In a typical traditional surveillance setting, a human operator has to constantly monitor a large array of video feeds for suspicious behaviour. In many cases, there are more cameras than screens, requiring the cameras to be cycled through the screens. A 2003 survey of CCTV in UK railway systems reported screen to camera ratios varying from 1:4 to 1:78 in eight railway stations [27]. In their informal survey of four UK installations, Dee and Velastin reported screen to camera ratios from 1:4 to 1:30, and operator to screen ratios as high as 1:16 [12]. As the number of security cameras increases, information overload makes the task of detecting occurrences of suspicious behaviour increasingly difficult, adding to other confounding factors such as human fatigue and boredom [12, 33]. These conditions make surveillance very challenging in large installations such as airports.

A human operator may also make biased judgements based on appearance instead of actual statistical behaviour [12]. Not only can this bias hamper overall performance, it can

also raise discrimination and privacy issues.

All of these factors make a compelling case for automating the visual surveillance process, as much as practicable, using computer vision and pattern recognition techniques.

1.2 Problem definition

The objective of an intelligent surveillance system is to minimize the involvement required of the human operator. The system should automate the monitoring and event detection components of video surveillance, alerting the operator only when unusual behaviour or other events of interest are detected. The general framework of intelligent surveillance comprises several stages: environment modeling, motion detection, object classification, tracking, behaviour understanding and description, and possibly personal identification [18]. An alternative formulation identifies the similar stages of low-level image processing, tracking, and high-level scene and/or behaviour analysis [12]. This thesis is concerned with the stage of high-level scene and behaviour analysis (i.e., behaviour understanding and description in the former framework), and more specifically, with the detection of suspicious behaviour given trajectory data from the tracking stage.

Fundamentally, methods that focus on this stage of the intelligent surveillance framework attempt to extract semantic meaning from tracker observations. Given an observed trajectory, we aim to draw some inferences about the behaviour of the individual - in particular, whether the individual's behaviour might be considered unusual. The degree to which an observed behaviour is unusual depends on the specific context. The scene's spatial structure can influence the types of normal behaviour: for example, an individual taking a shortcut through a restricted area should be identified as behaving unusually. Temporal considerations can also be a factor: for example, an individual sprinting down a corridor through which people normally walk is behaving unusually. Hence, an effective detector of unusual behaviour requires context-specific training or learning. In most surveillance scenarios, to learn explicit models for every possible form of suspicious behaviour is impractical. A more realistic approach, adopted by the vast majority of methods in the literature

(described later in Section 2.1), is to learn what constitutes normal behaviour and then classify new observations based on how novel they are with respect to the learned model.

The problem can be summarized as follows. Given a set of training trajectories, the algorithm is required to learn what constitutes normal movement behaviour in the scene. During system operation, the algorithm is required to classify new observed trajectories as normal or unusual.

1.3 Thesis objectives and organization

As described in more detail in Section 2.1, there are several methods in the literature for detecting unusual behaviour based on trajectories. However, most of these methods use low-level features such as flow vectors (vectors containing position and velocity data) or control points (e.g., for a spline), and ignore the fact that the trajectories are generated by humans with specific intentions and objectives. Recently, a novel approach that makes use of higher-level features of intentionality was proposed by Dee and Hogg [9, 10, 11]. Dee and Hogg introduced the idea of modelling individuals in the scene as intelligent agents instead of simply "objects". Detection of unusual behaviour then becomes a task of evaluating the goal-directedness of an agent's behaviour - in other words, how explicable the trajectory is with respect to the known spatial goals in the scene. More details are presented in Section 2.1.

This thesis builds upon Dee and Hogg's goal-based paradigm and extends it in three important ways. First, the proposed method learns the spatial scene structure in a training phase. In Dee and Hogg's method, the inference of goals and subgoals depends on an obstacle map that is specified manually. The proposed method's automatic learning of spatial goals in the scene reduces the human involvement required in training the system. Second, the proposed method learns a transition model that describes normal agent movement between subgoals/goals. This is a unique contribution that is not present in Dee and Hogg's method. The feature adds richer semantic information about the scene in terms of the relative frequency of normal subgoal/goal transitions and the normal speeds

of travel. Third, the proposed method classifies trajectories in progress in a probabilistic framework using particle filtering. The probabilistic approach is also novel with respect to the original method. Particle filtering enables multiple hypotheses about the goals of the agents in the scene to be efficiently maintained. Both Dee and Hogg's method and the proposed method naturally evaluate trajectories in progress (i.e., partial trajectories), allowing unusual behaviour to be detected as it is occurring instead of only after it has completed. Further details are presented in Chapter 3.1.

The experimental validation of this thesis aims to resolve two main questions. *First, can the proposed learning approach achieve the same classification accuracy as the original approach that uses a manually specified obstacle map? Second, can the goal-based method provide good classification performance in comparison to traditional methods based on low-level trajectory features?*

The rest of this thesis is organized as follows. Section 2 provides an overview of related work in trajectory-based unusual behaviour detection. Section 3 describes the proposed method. Section 4 reports the results of experiments conducted on one synthetic and two real-world datasets. Finally, conclusions and directions for future research are presented in Section 5.

Chapter 2

Related Work

This thesis focuses on the stage of behaviour understanding and analysis in the intelligent visual surveillance framework, which takes as input the results of the object tracking stage. Object tracking is a challenging problem, especially in the presence of occlusion, noise, camera motion, and changes in illumination [55]; however, tracking issues are beyond the scope of this work. The object tracking literature contains a wide variety of well-established algorithms and extensions of those algorithms. For example, the mean-shift tracker [7, 8] has been the basis of numerous extensions [15, 28, 54]. The interested reader is referred to the review by Yilmaz *et al.* [55] for a good treatment of the state-of-the-art in object tracking.

2.1 Trajectory-based methods for unusual behaviour detection

Several researchers have proposed trajectory-based methods for recognizing unusual behaviour patterns in a surveillance context. In most surveillance scenarios, to learn explicit models for all forms of suspicious behaviour is impractical. The more realistic approach, and the approach adopted by the methods described in this section, involves learning what constitutes normal behaviour in a particular scene, and then classifying new observed be-

haviours based on how novel they are with respect to the learned model. The methods described in this section perform online behaviour analysis, instead of identifying instances of unusual behaviour given an entire video sequence. Algorithms of the latter type [56] may be useful for post-mortem offline analysis; however, they are not suitable for real-time surveillance.

Johnson and Hogg developed one of the earliest works in trajectory-based detection of unusual behaviour [25]. Johnson and Hogg represented an object's trajectory using a sequence of flow vectors, which consist of the object's instantaneous position and velocity in the image plane. Clustering is performed on training flow vectors using a competitive neural network. The output of this network is input to a layer of leaky neurons. The leaky neurons retain a short-term memory of activation, allowing partial trajectory information to be encoded. The partial trajectories, as represented by the outputs of the leaky neuron layer, are then clustered using a second competitive neural network. In this way, clusters of normal instantaneous movements and partial trajectories are learned. However, the number of clusters is required to be specified a priori in both clustering steps. This requirement poses an implementation issue since it is not clear how an appropriate number of clusters should be determined. In addition, the experimental results demonstrate the performance of Johnson and Hogg's method in learning normal trajectory patterns but no explicit validation is presented for the task of unusual trajectory detection.

Owens and Hunter proposed a method for detecting unusual trajectories in greyscale surveillance video [40]. The authors build on the flow vector representation, adding second order information and applying temporal smoothing to encode a short-term history of motion. A self-organizing map neural network is used to learn normal trajectories. To classify the novelty of an observed feature vector, the Euclidean distance is calculated between the observed vector and the prototype vector of the winning neuron when the observed vector is input to the self-organizing map. If the distance is above a threshold, then the observed vector is classified as unusual. Owens and Hunter's method uses a simpler neural network structure than Johnson and Hogg's, but determining an appropriate configuration for the output layer to achieve effective clustering remains an implementation issue. The

trade-off between false negatives and false positives is also unclear, as experimental results are reported only for a single threshold value.

Stauffer and Grimson performed online k-means clustering on flow vectors augmented with object size information [48]. The joint co-occurrence statistics of the prototypes are used to learn a hierarchical classifier of trajectories. The novelty of a trajectory is determined based on the novelty of the constituent augmented flow vectors and the overall co-occurrence statistics. Similar to Johnson and Hogg's method [25], determining an appropriate number of clusters a priori for the clustering step is an implementation issue. The trajectory classifier is demonstrated within a fully integrated object tracking system; however, unusual behaviour detection is not experimentally validated.

Hu *et al.* developed a combined tracking and behaviour understanding system [19]. Trajectories are represented by sequences of "point feature vectors": flow vectors augmented with size information. Training trajectories are clustered using fuzzy k-means clustering in a two-stage approach. The first stage clusters trajectories based on only spatial information. Trajectories are resampled and linearly interpolated to obtain the fixed-length vectors needed for clustering. The number of clusters k is determined by finding a local optimum of the Tightness and Separation Criterion [52]; however, if the number of samples in a cluster falls below a fixed threshold, the cluster is merged with its closest neighbour. The second stage adds temporal information (velocities in the point feature vector) to further cluster the trajectories within the clusters found by the first stage. The output of this two-step clustering is cluster prototypes that are represented using "motion patterns". Each motion pattern is a sequence of Gaussian distributions. To learn these Gaussian distributions, each trajectory belonging to the cluster is uniformly partitioned using a common width, and point feature vectors falling into each partition are used to learn a Gaussian distribution corresponding to that partition. The probability of a point feature vector given a Gaussian distribution is then defined as a function of its Mahalanobis distance to the distribution. During system operation, a trajectory in progress is classified as normal or unusual by finding the motion pattern with maximum posterior probability, and then calculating the probability of the current point feature vector given this motion

pattern. If this probability remains low enough for several frames, then the trajectory is flagged as unusual. The experimental validation of the unusual trajectory detection could be improved by performing quantitative validation; only examples of correct detections are presented in the experimental results. The trade-off between false negatives and false positives could also be included in the quantitative validation.

Makris and Ellis proposed an algorithm in which normal trajectories are learned by presenting training trajectories in sequence and comparing them to the current prototype trajectories, or "routes" [32]. A route is modelled using a pair of entry and exit points, a central axis consisting of control points, and an envelope describing deviation from the axis. A training trajectory is assigned to the closest route according to a maximum separation distance measure, and the parameters of the matched route are updated based on the trajectory. If no route is sufficiently close, a new route is created. As training progresses, routes may be merged or split according to maximum separation distances. In later work, the authors improved the learning and modelling of entry and exit regions using 2-D Gaussian mixture models, and detected unusual trajectories using hidden Markov models (HMMs) [33]. A key contribution of the algorithm is the learning of a scene model that describes semantically important spatial regions in the scene. However, since unusual behaviour detection is considered more as an application than as a core component of the algorithm, experimental validation of unusual behaviour detection is limited and not quantitative.

Fernyhough *et al.* represented an object's trajectory using the convex hull of the object over time, partitioned according to the object's position sampled at regular time intervals [14]. Like Makris and Ellis [32, 33], prototype paths are learned by presenting training trajectories in sequence and updating the parameters of the matching prototype. Trajectories are then used to learn qualitative event sequences, where an event refers to a spatial interaction between two objects as defined by relative position and direction of motion (e.g., "following" or "travelling alongside left") [14]. No experimental validation is performed for the task of unusual behaviour detection, as unusual behaviour detection is presented as an application instead of as a core component of the algorithm.

Piciarelli and Foresti [41] took a similar approach to Makris and Ellis [32, 33] and proposed an online clustering algorithm for trajectories in which cluster prototypes consist of a sequence of points and local variances from those points. The distance from a trajectory to a cluster is defined as the mean normalized Euclidean distance between a point in the trajectory and the nearest point in the prototype within a sliding window. Clusters are organized in a tree structure and continually updated, split, and merged as tracking observations arrive. Accumulated frequency information is used to calculate the likelihood of an observed trajectory and decide whether it is normal or unusual. However, the method operates on complete trajectories and is not designed to detect unusual trajectories in progress. In addition, no experimental validation is presented for unusual trajectory detection.

In a later work, Piciarelli *et al.* proposed using a single-class support vector machine (SVM) to detect unusual trajectories [42]. Trajectories are subsampled into fixed-length vectors for training and classification. Outliers are detected and removed from the SVM training set based on the change in the hypervolume in SVM feature space containing the training samples as samples are removed: the hypervolume should shrink more markedly when removing outliers than normal trajectories. In contrast to many of the previous works, extensive quantitative validation is performed for unusual trajectory detection, including validation on a large synthetic dataset developed by the authors. However, how to select an appropriate subsampling (fixed vector length) that preserves all important spatial features of the trajectories is not obvious. In addition, the method requires complete trajectories and cannot detect unusual trajectories in progress.

Junejo *et al.* clustered raw trajectories using a min-cut graph algorithm [26]. Nodes in a fully connected weighted graph correspond to trajectories, and edges are weighted by the Hausdorff distance between trajectories. Each cluster is represented by a prototype trajectory and an envelope based on the maximum spatial deviation from the prototype. In addition, representative velocities and curvatures are modelled by fitting Gaussian distributions to the instantaneous velocities and curvature measures in the cluster's trajectories. An observed trajectory is classified as unusual if, for every learned cluster, the observed

trajectory is sufficiently different in terms of spatial extent, velocity, or curvature. A limitation of this approach is the reliance on Hausdorff distance, which does not consider sequential ordering: as a result, trajectories going in opposite directions are viewed as the same in clustering. The number of clusters is also required to be specified a priori; how to determine an appropriate number of clusters is unclear.

Naftel and Khalid represented trajectories using function approximations, including least square polynomial, Cheybyshev polynomial, and Discrete Fourier Transform [39]. Trajectories are compared using their Euclidean distance in the coefficient feature space. Training trajectories are clustered using a self-organizing map neural network followed by an agglomerative hierarchical clustering step. An observed trajectory is classified as unusual if its Mahalanobis distance to the nearest cluster is sufficiently large according to a T^2 statistic test. Similar to several of the previous methods, the number of trajectory clusters must be specified a priori, which poses an implementation issue. The authors suggest a possible workaround in which the agglomerative merging is controlled by a distance threshold, but how to select an appropriate threshold is not obvious. The method also operates primarily on complete trajectories and does not easily evaluate trajectories in progress. The authors do present an experiment in which trajectories in progress are simulated by removing 10%, 20%, 30%, 40%, and 50% of the points from the end of each trajectory. These partial trajectories are then passed to the classifier as though they are regular, complete trajectories. However, there are two problems if partial trajectories are simply treated as complete trajectories in practice: first, there is no means to distinguish between a truly unusual trajectory and a normal trajectory that has completed only a small proportion of its sequence (say, 50% complete); second, there is no means to estimate the proportion of the trajectory that is complete.

Jiang *et al.* modelled trajectories using hidden Markov models (HMMs) and adopted a similarity measure based on the Bayesian Information Criterion [24]. Training trajectories are clustered using a modified hierarchical clustering algorithm with 2-depth search and additional reclassification and HMM retraining steps in each iteration. The prior probability of each HMM is estimated using Expectation Maximization (EM), considering the

trajectories to be generated by a mixture of the HMMs. HMMs with above average prior probabilities are considered to model normal behaviour. An observed trajectory is classified as unusual if its probability of being generated by a mixture of the normal models falls below a threshold; the threshold is determined by the minimum probability of a normal training trajectory being generated by a mixture of the normal models. Quantitative experimental results are presented. However, the method requires complete trajectories and cannot detect unusual trajectories in progress.

Sillito and Fisher proposed a semi-supervised method for learning normal trajectories and detecting unusual behaviour [47]. A trajectory is represented by the control points of an approximating cubic spline, plus the elapsed time. During the training phase, a one-class classifier based on a Gaussian mixture model is incrementally learned. The incremental learning is semi-supervised: when a training trajectory is classified as unusual by the mixture model learned so far, the human operator is prompted to decide whether or not the trajectory is normal. Training trajectories that are classified as normal by the model learned so far do not trigger human intervention. A trajectory is classified as unusual if its Mahalanobis distance to the closest component of the Gaussian mixture model exceeds a threshold conditioned on the number of samples used to train the mixture model. Quantitative experimental results are presented, including Receiver Operating Characteristic (ROC) curves to illustrate the trade-off between false positives and false negatives. However, the classifier requires complete trajectories and cannot detect unusual trajectories in progress.

The work in this thesis is inspired by the recent approach of Dee and Hogg, in which individuals in the scene are modelled as intelligent agents with intentionality instead of simply "objects" [9, 10, 11]. Detection of unusual behaviour then becomes a problem of determining whether an agent's trajectory is explicable in terms of known spatial goals. The greater the goal-directedness of a trajectory, the more likely it represents normal behaviour. In [9], the goal-directedness of a trajectory is determined by the degree to which its constituent flow vectors are geometrically oriented towards spatial goals and subgoals, taking manually specified obstacles into account. Dee and Hogg later extended

this work to incorporate two human navigational strategies: shortest path and simplest path [10]. The shortest path to a goal is the one that minimizes total travel distance; the simplest path minimizes the total number of subgoals, or changes in direction. An observed trajectory is compared with the shortest and simplest paths to all possible spatial goals. A monotonic Hausdorff distance measure is used to determine the closest simplest or shortest path, and the goal-directedness of the observed trajectory is determined by the angular disparity between the observed trajectory and this ideal path.

Invasion of privacy is always a potential concern when deploying surveillance systems in public areas. To protect the privacy of individuals as far as possible, a surveillance system should collect the minimum amount of personal information necessary to achieve its objectives [5]. An advantage of trajectory-based methods for detecting suspicious behaviour is that they do not require any appearance information of the individuals in the scene. Trajectory-based methods should also integrate easily with privacy-enhancing surveillance technologies that encrypt the output of tracking algorithms [34].

2.2 Unusual behaviour detection in dense crowds

Trajectory-based methods for unusual behaviour detection are applicable in settings in which individuals can be tracked with reasonable accuracy. However, these methods may be unsuitable in scenes with very high-density crowds in which traditional tracking algorithms cannot be used. The detection of unusual behaviour in dense crowds is an emerging research area, and although beyond the scope of this work, a brief description of some of the work in this area is presented for completeness. In general, these methods use optical flow or other holistic scene features to track overall crowd movement without explicitly tracking individuals in the crowd.

Boghossian and Velastin proposed an early optical flow based method to detect crowd-related emergencies [4]. Circular flows near exits are used to detect crowd evacuation, as exits become bottlenecks during an evacuation. Diverging flows outwards from a region are indicative of local danger. Both the circular and diverging flows are detected using Hough

transforms. Motion-free regions isolated within homogeneous flows indicate obstructions. Flows in the scene are segmented using a region-growing segmentation method.

Xiang and Gong took a similarly holistic approach analyzing temporal patterns in foreground blobs or "scene-events" to detect normal and unusual events [50]. Scene-events are represented by feature vectors comprising blob centroid, shape, and motion features, and fitted to a Gaussian mixture model. The "behaviour pattern" in a video segment is encoded by a sequence of vectors, one per frame, of the posterior probability of each scene-event class (component in the Gaussian mixture model). Each behaviour pattern is then used to train a Multi-Observation HMM (MOHMM). The MOHMMs are used to calculate a pairwise similarity matrix over the behaviour patterns, allowing the behaviour patterns to be grouped using spectral clustering. Finally, the resulting clusters of behaviour patterns are used to train another mixture of MOHMMs that constitutes the normal behaviour model. During system operation, an observed behaviour pattern is classified as unusual if its likelihood given the normal behaviour model falls below a threshold.

For the task of detecting unusual crowd behaviour, Ihaddadene and Djeraba defined a measure based on optical flow features [20]. A KLT tracker [46] is used to track Harris corners in areas of high motion. A measure of entropy is then calculated based on the area of moving blobs, the direction and magnitude variances of optical flow vectors, and the dominant directions of optical flow. This measure is thresholded to classify crowd behaviour as normal or unusual.

Ali and Shah proposed a crowd flow segmentation algorithm based on Lagrangian particle dynamics [2]. A grid of particles is overlaid on the scene and advected using optical flow. The motion of the particles is used to construct flow maps in the horizontal and vertical directions. Spatial gradients in the flow maps are used to calculate a Finite Time Lyapunov Exponent (FTLE) field. Ridges in the FTLE field segment the crowd flow into regions with similar flow dynamics. In addition, instabilities or abnormalities in the crowd flow are detected by changes in the number of flow segments.

Mehran *et al.* detected unusual crowd behaviour by analyzing "interaction forces" over time [35]. Their approach is inspired by the Social Force model of crowd behaviour [17], in

which the motion of individuals is determined by both their desired motion and interaction forces from the environment or the crowd. Instead of tracking individuals in a dense crowd, Mehran *et al.*'s method applies Ali and Shah's particle advection framework [2] to capture the crowd flow. The interaction force on a particle is defined as a function of both the current optical flow and a spatially and temporally smoothed optical flow. The current optical flow represents the particle's desired motion under the Social Force model, and the spatially and temporally smoothed optical flow represents the particle's actual motion under the Social Force model. A classifier is trained on spatio-temporal volumes of normal interaction force magnitudes.

2.3 Summary

In summary, several methods for trajectory-based unusual behaviour detection have been proposed in the literature. Although each method has application and merit, a few recurring issues can be observed to varying degree: the requirement of the number of clusters in a clustering step to be known a priori; inability to evaluate trajectories in progress, and hence to detect unusual behaviour as it is occurring instead of only after it has completed; lack of quantitative experimental validation; and lack of characterization of the trade-off between false positives and false negatives. Table 2.1 shows a summary of these issues.

Most traditional approaches to trajectory-based detection of unusual behaviour also rely on low-level trajectory features such as flow vectors or control points. As a result, they ignore the fact that the trajectories are generated by humans with specific goals and intentions. No attempt is made to model the higher-level psychology of the individuals in the scene, which could be instrumental in the task of detecting unusual behaviour. In contrast, Dee and Hogg's recent goal-based approach makes use of higher-level features of intentionality. Individuals in the scene are modelled as intelligent agents and trajectories are evaluated based on their explicability with respect to spatial goals. The following chapter elaborates on the contributions of this thesis to the goal-based approach.

Table 2.1: Summary of recurring issues in trajectory-based unusual behaviour detection. A checkmark indicates successful inclusion in the work.

	Evaluates trajectories in progress?	Number of clusters *not* required a priori?	Quantitative validation of unusual trajectory detection?	Trade-off between false positives and false negatives?
Johnson and Hogg [25]	✓			
Owens and Hunter [40]	✓		✓	
Stauffer and Grimson [48]	✓			
Hu *et al.* [19]	✓	✓		
Makris and Ellis [32, 33]	✓	✓		
Fernyhough *et al.* [14]	✓	✓		
Piciarelli and Foresti [41]		✓		
Piciarelli *et al.* [42]		✓	✓	
Junejo *et al.* [26]			✓	
Naftel and Khalid [39]			✓	
Jiang *et al.* [24]		✓	✓	
Sillito and Fisher [47]		✓	✓	✓
Dee and Hogg [11]	✓	✓	✓	✓

Chapter 3

Proposed Method

This chapter begins by detailing the contributions of this thesis and the broader significance of the research. Subsequent sections then elaborate on the proposed method in terms of the three main contributions.

3.1 Contribution details and broader significance

As mentioned in Section 1.3, the key objectives of this thesis are to further develop the goal-based approach of Dee and Hogg in three important ways.

First, as described in Section 3.2, the proposed method learns a spatial scene model in a training phase. Dee and Hogg's scene model comprises goals, which correspond to exit regions, and subgoals, which correspond to intermediate turning regions induced by the obstacles in the scene. Dee and Hogg's obstacles are manually specified using a polygonal model, and subgoals are recursively calculated based on tangential obstacle vertices assuming linear travel from one subgoal to the next. An advantage of their approach is that learning subgoals requires no training. However, the manual crafting of polygonal obstacle models for every scene is time-consuming from a usability perspective and may be subject to bias. Moreover, if the camera is moved, the entire scene model must be re-specified by hand. The proposed method's automatic learning of spatial regions in the scene reduces the human involvement required in training the system.

Second, as described in Section 3.3, the proposed method learns a transition model that describes normal agent movement between spatial regions. In particular, the relative frequency of region transitions and the normal speeds of travel between regions are encoded. This is a unique contribution that is not present in the original method. These additions incorporate richer semantic information about the scene and extend the scope of the types of unusual behaviour that the system can detect. For example, an agent using a common spatial route but moving at an unusually high speed may be identified as behaving unusually. Hence, the proposed method incorporates not only the spatial characteristics but also the temporal characteristics of the agent's behaviour.

Third, as described in Section 3.4 the proposed method classifies trajectories in progress in a probabilistic framework using particle filtering. The probabilistic approach is also novel with respect to the original method. Particle filtering is a well-established method for probabilistic reasoning over time. The framework allows the proposed method to efficiently maintain multiple hypotheses about the goals of the agents in the scene over time, and offers a natural way to integrate the learned transition model.

Developing and extending the goal-based paradigm has the potential to open promising new avenues in intelligent surveillance research. For example, researchers have recently started to focus on natural language description of surveillance events [18]. The aim of this branch of research is to take the results of the behaviour analysis and translate them into a form that a human can readily understand. Traditional methods relying on low-level trajectory features such as flow vectors or control points risk producing black-box results that are not easy to interpret by human surveillance operators. On the other hand, describing behaviour in terms of spatial goals is natural and intuitive for humans. For instance, a human readily understands a statement such as "The individual is walking from the main entrance towards the elevators." In contrast, to interpret an individual's trajectory following a spline, set of control points, or flow vectors is not easy.

The semantic-rich description of behaviour using goals is also useful for content-based surveillance video retrieval [18]. A surveillance operator may need to retrieve all instances of individuals entering a restricted zone or leaving at a particular exit, for example.

This ability to describe trajectories in terms of spatial goals has potential applications beyond security surveillance as well. For example, it could be used by architects in evaluating a designed space. Architectural assessment typically involves analysing the paths people take with respect to design elements in the environment (e.g., fountains, walkways, steps). The task has traditionally required manual observation and annotation, but recently a system that incorporates simple computer vision based tracking has been proposed [53].

3.2 Modelling the scene structure

3.2.1 Overview

The proposed method models the spatial scene structure by learning three separate Gaussian mixture models (GMMs) from the training trajectories: one GMM each for the sets of all entry points, all turning points, and all exit points. The entry, turning, and exit points are 2-D coordinates, so each mixture component is a 2-D Gaussian distribution. Entry and exit points are straightforward: these are simply the first and last points in the trajectories. Turning points refer to locations at which agents make significant changes in direction, such as at road bends or intersections. Turning locations are detected using image processing techniques described later in this section. Modelling entry and exit regions using GMMs has precedent in the works of Makris and Ellis [32, 33] as well as Dee and Hogg [9, 10, 11].

In the current implementation, all training trajectories are assumed to correspond to normal behaviour. However, in future work, investigating ways to automatically detect and remove outlier trajectories from the training set will be valuable.

3.2.2 Fitting GMMs using Expectation Maximization

In general, a mixture distribution is a weighted combination of multiple distributions, or components, and is given by [44]

$$P(x) = \sum_i w_i \, P(x|C = i) \tag{3.1}$$

where x is a generic data point, C denotes a component, and $w_i = P(C = i)$ is the weight or prior probability of component i. A GMM in particular consists of Gaussian components, and in the case of 2-D Gaussians is given by

$$P(x) = \sum_i w_i \, \frac{1}{2\pi|\Sigma_i|^{1/2}} \exp(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)) \tag{3.2}$$

where μ_i is the mean and Σ_i the covariance of Gaussian component i.

The GMMs are fit to the training data using the Expectation Maximization (EM) algorithm [44]. The EM algorithm is an iterative method for estimating model parameters from incomplete sample data. Each iteration involves an E (expectation) step and an M (maximization) step. In the E step, the current estimated model parameters are used to calculate the expected values of the hidden variables, with which we would have the complete data to describe the problem. In the M step, the model parameters are updated to maximize the log likelihood of the data. The process repeats until convergence. For fitting a GMM to the training data, the E and M steps are as follows [44]:

E: For each data point and Gaussian component, calculate the probability p_{ij} that data point x_j is generated by Gaussian component i:

$$p_{ij} = P(C = i|x_j) = \alpha P(x_j|C = i)P(C = i) \tag{3.3}$$

by Bayes' rule, where α is a normalization constant and the other terms are evaluated according to the current estimated GMM.

M: Update the model parameters by

$$\mu_i = \frac{\sum_j p_{ij} x_j}{\sum_j p_{ij}} \tag{3.4}$$

$$\Sigma_i = \frac{\sum_j p_{ij}(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_j p_{ij}} \tag{3.5}$$

$$w_i = \sum_j p_{ij} \tag{3.6}$$

3.2.3 Determining an appropriate number of mixture components

For Makris and Ellis [32, 33] as well as Dee and Hogg [9, 10, 11], the number of entry and exit regions, and hence the number of mixture components in the GMMs, is manually specified. The proposed method reduces the human involvement required to train the system by automatically determining an appropriate number of mixture components using the Bayesian Information Criterion (BIC) [45]. The automatic estimation of an appropriate number of mixture components is especially important for fitting the turning region GMM, as this information would be difficult to obtain a priori in most practical settings.

BIC is a penalized likelihood criterion used for model selection. In general, when fitting a model to training data, the likelihood of the training data can be increased by increasing the model's complexity in terms of the number of model parameters. However, as the number of parameters in the model increases, so does the risk of over-fitting the data. The purpose of a penalized likelihood criterion is to penalize model complexity (large numbers of parameters) [6]. Another established penalized likelihood criterion is the Akaike Information Criterion [1], which has been used, for example, to determine the appropriate number of histogram bins in the context of object tracking [23].

BIC can be applied to determine an appropriate number of components for a GMM. The BIC value is evaluated for candidate models with varying number of mixture components, and the model corresponding to the maximum BIC value is selected. In the case of the proposed method, GMMs are fit to the data (entry points, turning points, or exit points) with incrementally increasing number of mixture components, and the model corresponding to the first local maximum BIC value is selected. GMM selection based on the maximum BIC value has been previously used in the pattern recognition literature with success [51].

3.2.4 Detecting turning points

In an early implementation of the proposed method, turning points were detected based on the cosine of the angle between the direction vectors before and after a point. Denoting the direction vectors as u_1 and u_2, the cosine can be calculated by

$$\cos\theta = \frac{u_1 \cdot u_2}{|u_1||u_2|} \tag{3.7}$$

If the cosine value falls below a threshold, then the point is considered a turning point. To reduce the effects of noise in the trajectory, v_1 and v_2 can be averaged over a fixed number of points, or window. This simple approach is sensitive to the window size and is appropriate only when turns occur at the same scale for all trajectories. The approach is not very effective when turns occur at different scales, which can be expected in most real-world surveillance situations.

To robustly detect turning points at varying scales, the proposed method adapts an existing algorithm from the image processing literature. Many image processing researchers have worked on the problem of detecting corners in curves, which in the context of the proposed method corresponds to detecting turning points in trajectories. Curvature scale space (CSS) [36, 43] is one popular tool for detecting corners at varying scales. CSS methods find candidate corner points by smoothing the curve at different scales, typically by convolving the curve with Gaussians of different standard deviations, and finding local maxima of curvature. The curvature κ at a point P is defined as the differential change in the angle ψ with respect to arclength s at that point [43]:

$$\kappa = \frac{d\psi}{ds} \tag{3.8}$$

where ψ is the angle subtended by the tangent at P with the x-axis. Given a curve in the spatial plane, denoted in parametric form $(x(s), y(s))$, the curvature can be calculated by [43]

$$\kappa(s) = \frac{\dot{x}(s)\ddot{y}(s) - \ddot{x}(s)\dot{y}(s)}{(\dot{x}^2(s) + \dot{y}^2(s))^{3/2}} \tag{3.9}$$

where $\dot{x}(s)$ and $\dot{y}(s)$ denote the first derivatives of x and y, and $\ddot{x}(s)$ and $\ddot{y}(s)$ denote the second derivatives. The curvature of a curve smoothed by a Gaussian with standard

deviation σ can be calculated by [36, 43]

$$\kappa(s,\sigma) = \frac{\dot{x}_s(s,\sigma)\ddot{y}_s(s,\sigma) - \ddot{x}_s(s,\sigma)\dot{y}_s(s,\sigma)}{(\dot{x}_s^2(s,\sigma) + \dot{y}_s^2(s,\sigma))^{3/2}} \tag{3.10}$$

After finding the candidate corner points, different CSS methods use varying techniques for identifying and removing false corners.

The proposed method adapts He and Yung's CSS corner detector [16], which removes false corners based on an adaptive threshold and a dynamic region of support. He and Yung's CSS corner detector normally removes wide, or "rounded", corners from consideration by using an adaptive local threshold. However, in the case of the proposed method, wide turns are valid and should be kept. This modification is easily achieved by setting He and Yung's constant C to 1 (as pointed out in Section 3.1 of [16]). The rest of He and Yung's CSS corner detector is unmodified. In particular, false corners are identified by checking the angle based on a dynamic region of support. Using the terminology from the previous discussion of the cosine approach, the region of support is essentially the window around a candidate corner point. In He and Yung's algorithm, the region of support is defined dynamically by the nearest corner candidates.

Figure 3.1 shows a few sample corner detection results. The detected corners are marked with green asterisks.

3.2.5 Examples

Figure 3.2 illustrates the overall scene modelling approach on some sample training sets from Piciarelli *et al.*'s synthetic dataset [42]. The training trajectories are overlaid in gray. Entry, turning, and exit points are indicated by blue, green, and red points respectively. The asterisks denote the mixture component means of the entry, turning, and exit GMMs.

3.3 Learning a region transition model

To encode information about travel patterns between entry, turning, and exit regions, a transition model is learned in terms of region-to-region segments, which will be referred

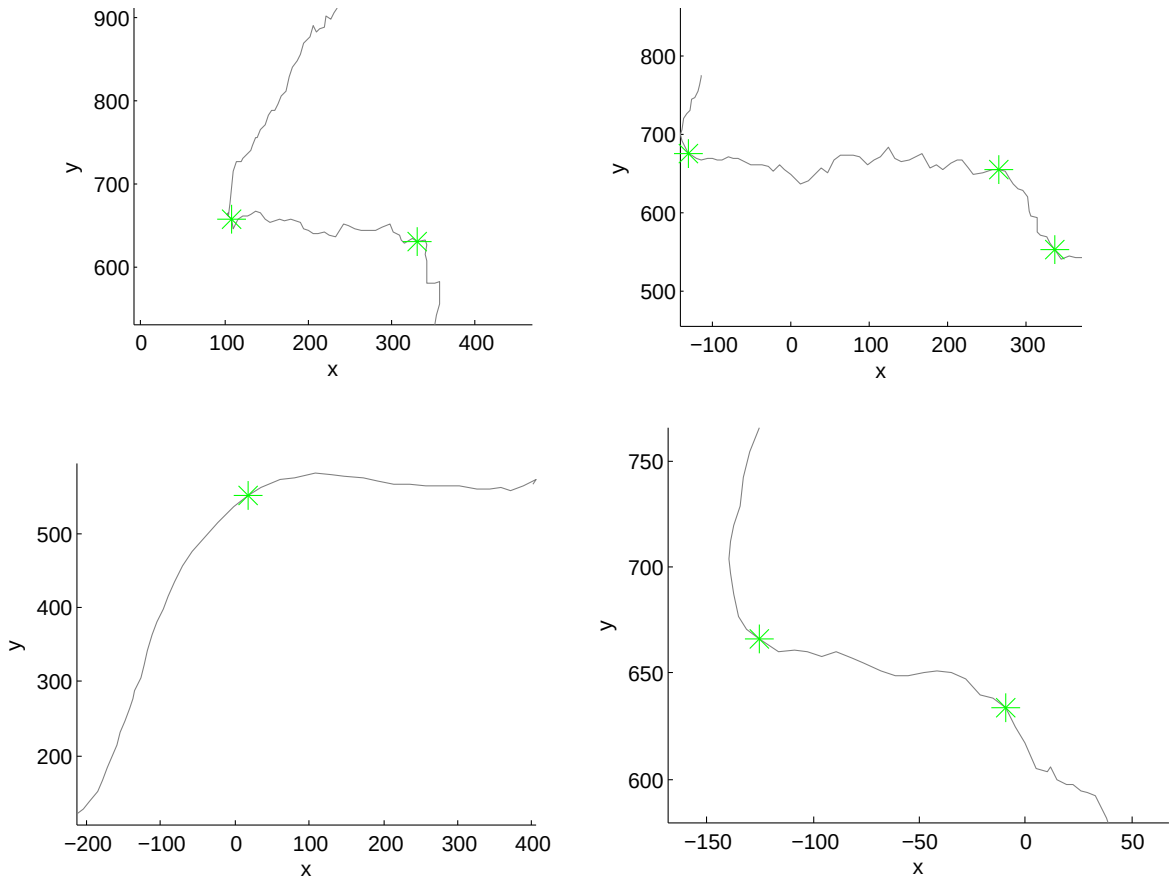

Figure 3.1: Sample corner detection results on trajectories from Dee and Hogg's carpark dataset [9, 10, 11]. Detected corners are marked with green asterisks.

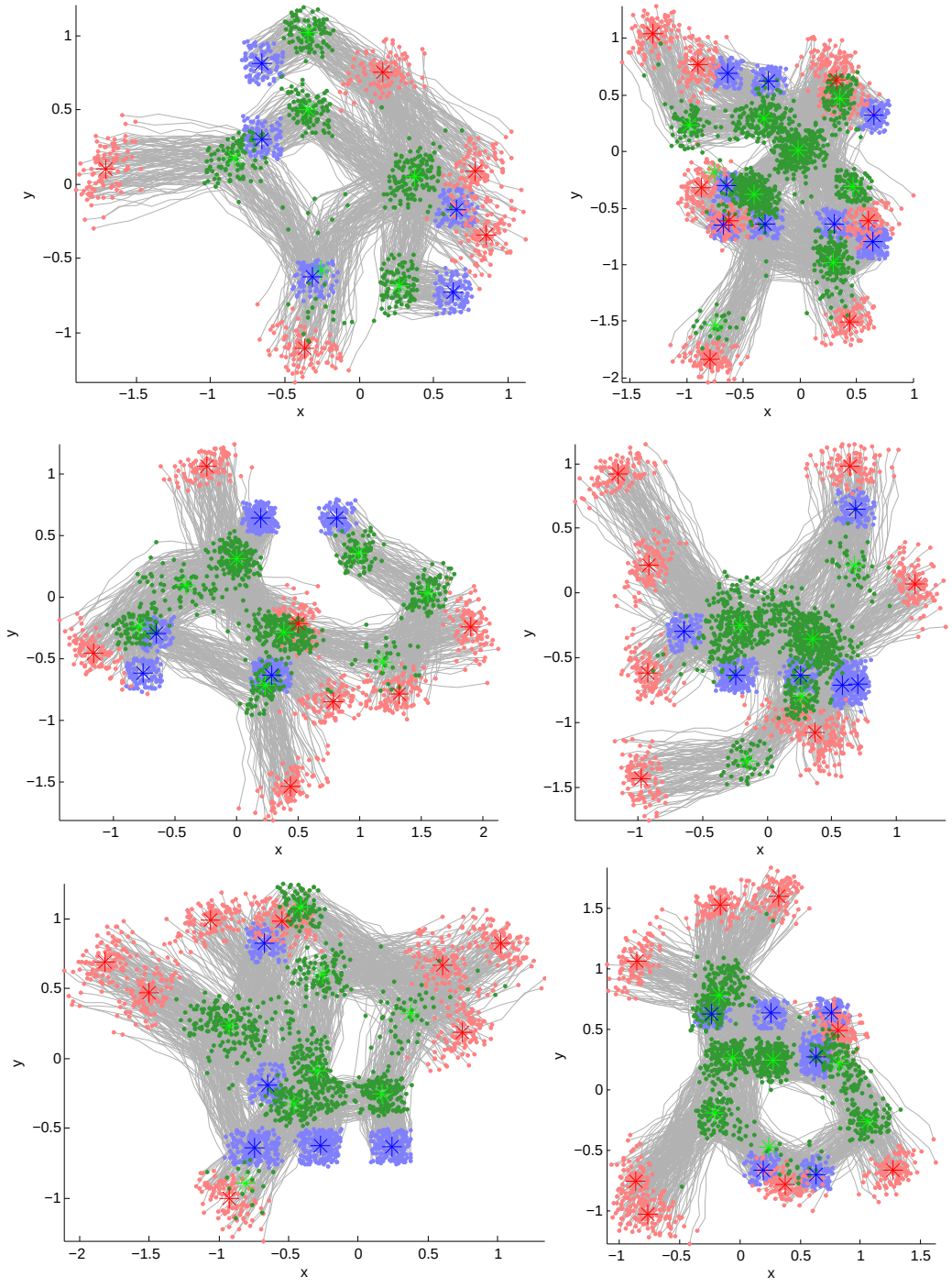

Figure 3.2: Sample learned scene models for training sets from Piciarelli *et al.*'s synthetic dataset [42]. Entry, turning, and exit points are indicated by blue, green, and red points respectively. Asterisks denote the component means of the entry, turning, and exit GMMs.

to as *path segments*. For example, if an agent enters the scene at entry region 1, proceeds to turning region 5, continues to turning region 7, and finally exits at exit region 10, then the agent traverses the path segments 1-5, 5-7, and 7-10. The transition model is used to predict the next path segment (and hence the next destination region) that the agent will choose upon arriving at his/her current destination region. As will be discussed in Section 3.4, this prediction does not have to be made deterministically: it can be made probabilistically in a particle filtering framework, which allows multiple hypotheses to be efficiently maintained and evaluated.

To keep the prediction tractable, the Markov assumption is made: the current state depends on a finite history of previous states [44]. The proposed method adopts the transition model for a simple second-order Markov process. In other words, the probability of an agent choosing a particular path segment depends only on the current and the previous path segments.

The path segment transition probabilities are estimated by performing frequency counting on the training trajectories. As an extremely simplistic example, suppose there are five training trajectories. A training trajectory contains an entry point, an exit point, and zero or more turning points, each of which can be associated with one of the learned regions in the scene model as described in Section 3.2. Call the sequence of regions corresponding to these significant points the *region sequence* of the trajectory. Recalling the previous example of an agent traversing regions 1, 5, 7, and 10, the region sequence of that trajectory would be 1-5-7-10. Suppose the five training trajectories have region sequences of 1-5-7-10, 1-5-7-10, 1-5-7-9, 1-3-7-10, and 1-3-7-9. The resulting transition model would estimate the conditional probability of an agent choosing to take path segment 7-10 to be $^2/_3$ given that the current path segment is 5-7 and the previous path segment is 1-5; or to be $^1/_2$ given that the current segment is 3-7 and the previous path segment is 1-3. A *null* region is used as necessary if there is no previous or current path segment: for example, the conditional probability of an agent choosing to take path segment 1-5 is $^3/_5$ given that the previous and current path segments are both *null*.

The mean and standard deviation of agent speeds are also calculated for travel within

each path segment. To reduce the effect of noisy outliers due to tracking errors, the top and bottom 5% of speeds are removed before calculating the mean and standard deviation measures. This speed information is used in the inference engine described in the following section, and will be referred to as the *speed model*. The purpose of calculating speed statistics on a per path segment basis instead of globally is to allow for the possibility of different speed expectations in different parts of the scene; that is, normal speed behaviour may be non-stationary in the scene.

3.4 Classifying observed trajectories

3.4.1 Particle filtering

Particle filtering is a well-established and efficient algorithm for probabilistic reasoning over time [44]. Before elaborating on the algorithm, this subsection briefly reviews the key terminology in probabilistic inference over time. Most of the following discussion is based on the treatment of the topic in [44] and [49].

The *state* comprises the random variables of interest; the state at time t is denoted x_t. An *observation* z_t is a measurement that provides information about the state at time t.

The evolution of the state over time is governed by the *state transition model* $p(x_t|x_{0:t-1})$, which specifies the conditional probability of x_t given all the prior states $x_{0:t-1}$. If the state is assumed to be *complete* - that is, if we assume each state x_t contains all information required to predict the next state, without need of previous states - then the transition model simplifies to $p(x_t|x_{t-1})$. Using the terminology in Section 3.3, assuming completeness implies making the Markov assumption and treating the state evolution over time as a first-order Markov process.

The conditional probability $p(z_t|x_t)$ is referred to as the *sensor model* or *measurement model*. The sensor model describes how measurements are affected by the actual state. Measurements can be thought of as "noisy projections of the state" [49].

Finally, the posterior probability distribution over the current state, conditioned on all

Table 3.1: An iteration of the Bayes filtering algorithm [49]

Input: $bel(x_{t-1}), z_t$

for all x_t **do**

$\bar{bel}(x_t) = \int p(x_t|x_{t-1})bel(x_{t-1})dx_{t-1}$

$bel(x_t) = \eta p(z_t|x_t)\bar{bel}(x_t)$

end for

return $bel(x_t)$

---

observations so far, is referred to as the *belief*: $bel(x_t) = p(x_t|z_{1:t})$. The objective of a probabilistic *filtering* algorithm is to compute the belief.

The most general probabilistic filtering algorithm is the Bayes filter, which is shown in Table 3.1. The Bayes filter recursively calculates the current belief based on the previous belief. Two steps are involved in each iteration: a prediction step and a correction step. The prediction step computes $\bar{bel}(x_t)$, which predicts x_t based on the previous state x_{t-1} according to the state transition model. The prediction is then corrected by incorporating the observation z_t according to the sensor model. The constant η is a normalization constant to ensure $bel(x_t)$ integrates to 1.

The Bayes filter as stated above is computationally feasible for only very simple problems. It is the theoretical basis for a number of approximations that are more computationally practical, including the Kalman filter, the information filter, the histogram filter, and the particle filter.

The particle filtering algorithm approximates the belief distribution using a finite number of samples drawn from the belief distribution. The samples are referred to as particles. Each particle encapsulates an instantiation of the state. A particle can also be interpreted as a hypothesis about the true state [49]. This sample-based representation of the belief gives the particle filtering algorithm the flexibility to approximate arbitrary, multi-modal belief distributions, as well as handle complex, non-linear state transitions over time.

Table 3.2: An iteration of the particle filtering algorithm [44, 49]

Inputs: particle set X_{t-1}, observation z_t

Denote the number of particles in the particle set as M.

for each particle $x_{t-1}^i \in X_{t-1}$ **do**

  Sample $x_t^i \sim p(x_t|x_{t-1})$

  Calculate particle weight $w^i = p(z_t|x_t)$

end for

Create X_t by drawing (with replacement) M particles, in which the probability of drawing particle i is proportional to w^i

return X_t

The particle filtering algorithm is summarized in Table 3.2. Similar to the basic Bayes filter, the particle filter involves a prediction step followed by a correction step. In the prediction step, the particles are propagated forwards according to the state transition model $p(x_t|x_{t-1})$. The resulting set of particles is an approximation of $\bar{bel}(x_t)$. In addition, a weight or importance factor is computed for each particle. The weight of a particle is given by its observation likelihood $p(z_t|x_t)$. Next, in the correction step, the observation is incorporated by resampling the particles based on the computed weights: a new particle set is created by drawing particles with replacement, in which the probability of a particle being drawn is proportional to its weight. The new particle set contains the same number of particles as the original particle set and is not weighted.

Figure 3.3 illustrates a simplistic example of the particle filtering algorithm in action. Suppose we are tracking an object in 2-D space. Figure 3.3(a) shows the particle approximation of the belief of the object's position at a particular time $t-1$. The actual position of the object is $(2, 2)$ but there is some spread representing the uncertainty of the localization. At time t, we wish to update the belief distribution. First, the particles are propagated forwards according to a state transition model. Depending on the application, the state transition model might be as simple as a constant velocity model, or it can be

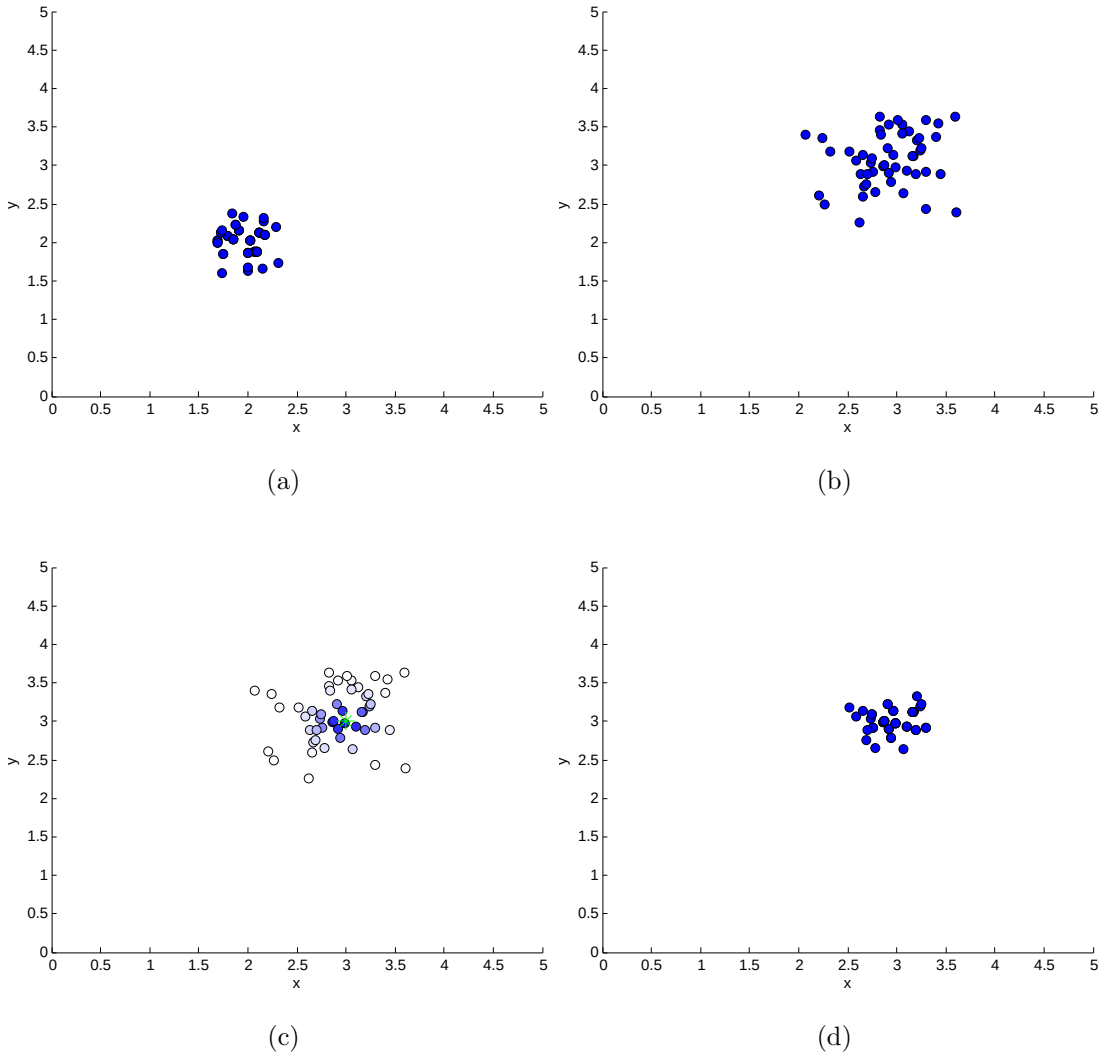

Figure 3.3: Illustrative example of the particle filtering algorithm (see text for details): (a) initial particles at time t - 1; (b) particles after prediction using state transition model; (c) particle weights based on observation likelihood; (d) final particles after weighted resampling.

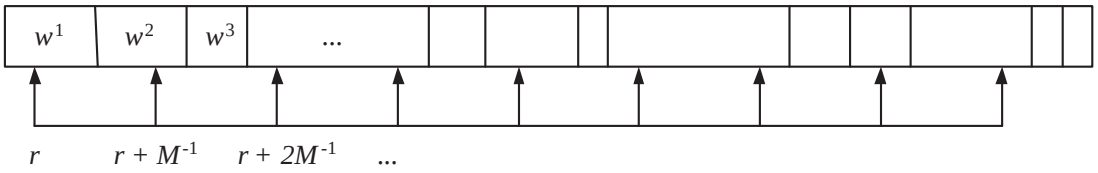

Figure 3.4: Illustration of low-variance resampling, adapted from [49].

a more complex non-linear model. Figure 3.3(b) shows the updated particle set after the prediction. The new observation z_t is incorporated by assigning weights to the particles and performing a weighted resampling. Figure 3.3(c) shows the observation z_t and the resulting weights that are assigned to the particles. Darker particles indicate higher weights. Particles that are in closer agreement with the observation are more heavily weighted. Figure 3.3(d) illustrates the new particle set after resampling. This new particle set represents the updated belief at time t.

The algorithm as discussed so far is the basic form of the particle filtering algorithm. The implementation of the proposed method adopts a slightly extended version of the basic algorithm in which low-variance resampling is performed. One of the inherent limitations of any random sampling method is sampling variance, or sampling error: statistics derived from the samples will vary slightly from the true statistics [49]. Repeated resampling, as performed in each iteration or time step of the particle filtering algorithm, adds to the sampling variance. Sampling variance can be reduced by using a low-variance sampling method, such as the one summarized in Table 3.3. The idea is illustrated in Figure 3.4. Instead of generating M numbers at random, a single random number is generated and used to systematically cover the sample space, while still sampling particles with probability proportional to their weights. An additional advantage of this sampling method is computational efficiency: the sampling of M particles requires only $O(M)$ time.

The particle filtering algorithm and its variants have been applied successfully in a variety of research domains, including in object tracking [21, 22, 31], as well as in robotics for simultaneous localization and mapping [37, 38].

Table 3.3: Low-variance resampling method for particle filtering [49]

Input: particle set X, weights W

Denote the number of particles in the particle set as M. Denote the weight of particle i as w^i

Sample r from the interval $[0, M^{-1}]$

$c = w^1$

$i = 1$

for $m = 1$ to M **do**

  $U = r + (m - 1) \times M^{-1}$

  while $U > c$ **do**

    $i = i + 1$

    $c = c + w^i$

  end while

  Add x^i to the new particle set X'

end for

return X'

3.4.2 Inference details

In the proposed method, the state x_t at time t consists of the agent's position s_t, speed v_t, current path segment e_t^1, current path segment end position d_t, and previous path segment e_t^0:

$$x_t = \left(s_t,\ v_t,\ e_t^1,\ d_t,\ e_t^0\right)^T \tag{3.11}$$

Recall that a path segment is defined by a start region and an end region. The specific role of the current path segment end position d_t is discussed later in the description of the state transition model, but intuitively it represents the location within the current path segment's end region to which the agent is predicted to be heading. Agent velocity is implicit given the speed: following Dee and Hogg's goal-based paradigm, the agent is assumed to travel linearly between regions. This linearity assumption might seem restrictive at first glance, but at least three reasons can be given for its validity. First, the turning points by definition capture significant changes in direction; hence, travel between pairs of entry, turning, or exit points should be reasonably linear. Second, transportation research suggests that shortest distance and fewest turns are among the top path planning strategies for humans [10, 11]. Third, the incorporation of a process noise covariance in the state transition model accounts for small deviations from linear travel.

The state transition model $p(x_t|x_{t-1})$ is defined as follows; a summary in flowchart form can be found in Figure 3.5.

If the distance to the path segment end position d_{t-1} is greater than the speed v_{t-1}, the new position s_t is calculated by projecting the particle a distance of v_{t-1} towards d_{t-1}. Otherwise, the current path segment is completed and the next path segment is sampled according to the path segment transition model, conditioned on the current and previous path segments e_{t-1}^1 and e_{t-1}^0. Note that this sampling involves selecting both a path segment and a path segment end position. The end position is sampled based on the GMM component of the selected path segment's end region. Hence, e_t^1, e_t^0, and d_t are all updated. Any remaining speed is then used to project the particle towards d_t. When updating the position s_t in either case, random zero-mean Gaussian noise based on a process noise covariance R is added to s_t, to account for deviations from the linear travel

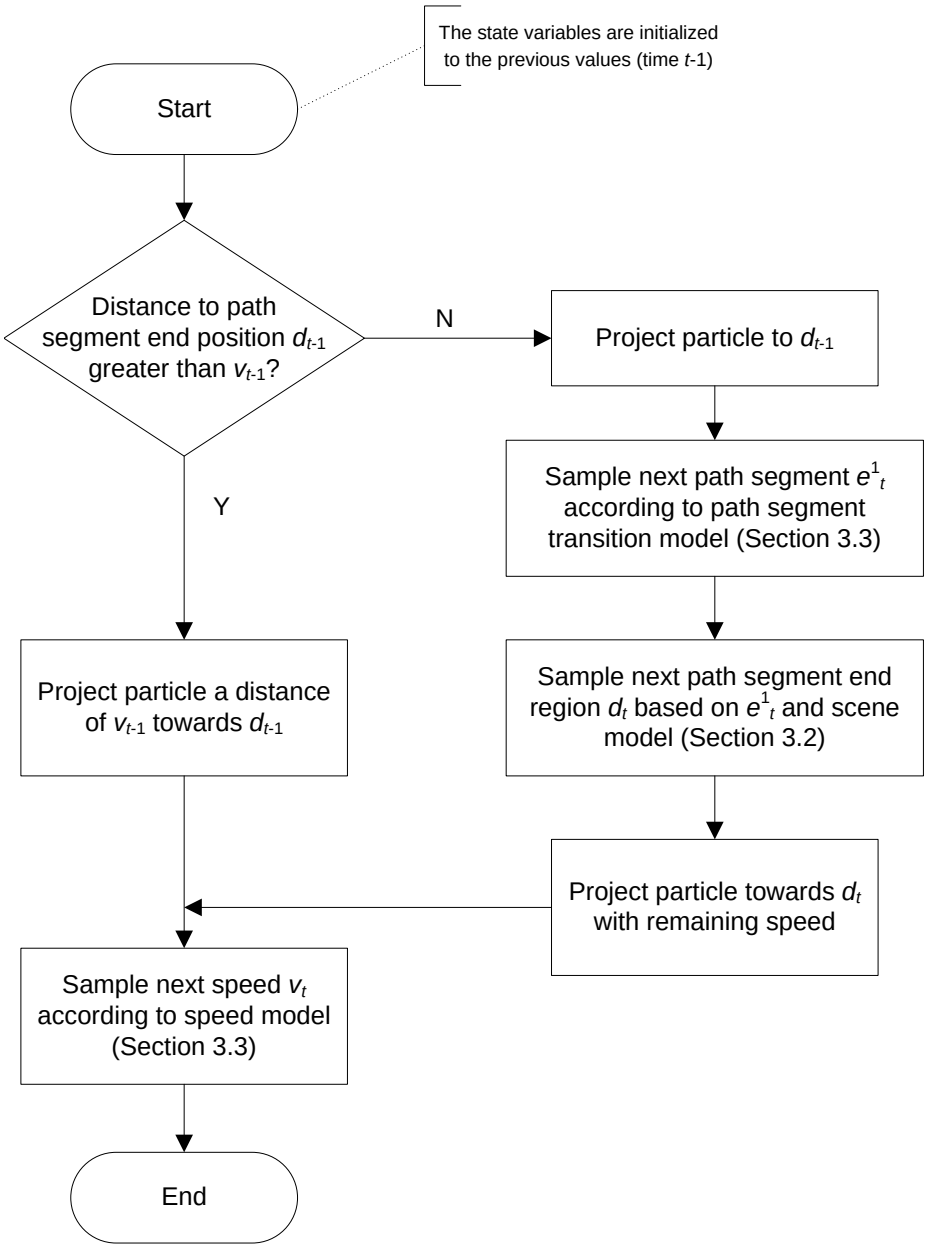

Figure 3.5: Summary flowchart of state transition model

assumption.

The speed v_t is sampled from the learned speed model conditioned on the current path segment e_t^1. Recall that the speed model specifies the mean and standard deviation of travel speed for each path segment. The speed is sampled from the Gaussian distribution with this mean and standard deviation.

Any state variables that are not explicitly changed as indicated above remain unmodified by the state transition.

The particle weights are assigned according to the observation likelihood function $p(z_t|s_t) = N(0, Q)$, where Q is the measurement noise covariance. In the current implementation, the measurement noise covariance is set empirically. The notation $N(\mu, \Sigma)$ refers to the Gaussian probability distribution with mean μ and covariance Σ. The particles are then resampled using the low-variance resampling method previously described.

An observed trajectory in progress is classified as unusual if the maximum observation likelihood (or particle weight) in the particle set falls below a threshold. This threshold can be adjusted to shift the balance between false negatives and false positives, and is varied in the experiments to generate the Receiver Operator Characteristic (ROC) curves (more details follow in Section 4.1).

Table 3.4 summarizes the proposed inference method as described above.

The inference is initialized as follows. The starting point of the trajectory is classified into one of the entry regions according to the learned entry model. Specifically, a Maximum a Posteriori (MAP) classification is performed on the starting point to determine the appropriate mixture component in the entry GMM. A MAP classifier selects the most likely class (in this case, a component of the entry GMM) given a data point. More formally, the MAP classification rule can be stated as follows:

Assign data point x to class C_i iff $P(C_i|x) > P(C_j|x) \quad \forall j, i \neq j$

Using Bayes' rule, this condition is equivalent to $p(x|C_i)P(C_i) > p(x|C_j)P(C_j) \quad \forall j, i \neq j$, which can be readily evaluated according to the learned GMM.

To identify unusual entries, the Mahalanobis distance to the classified GMM component

Table 3.4: An iteration of the proposed inference method using particle filtering

---

Inputs: particle set X_{t-1}, observation z_t

for each particle $x_{t-1}^i \in X_{t-1}$ **do**

  if $|d_{t-1} - s_{t-1}| < v_{t-1}$ **then**

    if e_{t-1}^1 is an exit region **then**

      $s_t = \infty$

    else

      Sample $e_t^1 \sim p(e|e_{t-1}^1, e_{t-1}^0)$

      Sample d_t based on scene model and end region of e_t^1

      $e_t^0 = e_{t-1}^1$

      $s_t = d_{t-1} + (d_t - d_{t-1})/|d_t - d_{t-1}| \times (v_{t-1} - |d_{t-1} - s_{t-1}|) + \epsilon_R$

    end if

  else

    $s_t = s_{t-1} + (d_{t-1} - s_{t-1})/|d_{t-1} - s_{t-1}| \times v_{k-1} + \epsilon_R$

  end if

  Sample $v_t \sim p(v|e_t^1)$

  Calculate particle weight $w^i = p(z_t|s_t) = N(z_t : s_t, Q)$

end for

Create X_t by performing low variance resampling (Table 3.3)

return X_t

---

is also computed. The Mahalanobis distance, also known as the Generalized Euclidean distance, is a distance measure that takes into account the shape of the cluster and is defined by

$$d^2_{GED}(x, m) = (x - m)^T S^{-1}(x - m) \tag{3.12}$$

where m is the sample mean and S the sample covariance of the cluster. Intuitively, the Mahalanobis distance is scaled based on the spread of the cluster as characterized by its covariance. A Mahalanobis distance of 2, for example, may be interpreted as 2σ from the cluster mean.

To enforce the spatial constraints of the entry regions, a trajectory is immediately flagged as unusual if the Mahalanobis distance between the entry point and the MAP-classified entry region exceeds 4 (i.e., the entry point is more than 4σ from the mean of the MAP-classified entry GMM component). Otherwise, the particle set is created and uniformly initialized to $e^1 = (null, i), e^0 = (null, null), d = (\text{entry point}), s = (\text{entry point}), v = \epsilon$ (where ϵ is a very small positive value). This initialization causes the particle filter to sample the path segments starting from the entry region in the first iteration.

If the trajectory is still classified as normal by its conclusion, then the exit point is also checked for consistency with the spatial constraints of the exit regions. If the exit point is not within a Mahalanobis distance of 4 of an exit region represented in the particle set, then the trajectory classification is changed to unusual.

3.5 Summary

In summary, the proposed method learns context-specific patterns of normal behaviour from training trajectories. A spatial scene model consisting of three GMMs is created: one each for all entry, all turning, and all exit regions. BIC is used to determine an appropriate number of Gaussian components in each GMM. Turning points are detected using a curvature scale space corner detector. A transition model is learned to encode statistics of travel between regions. During system operation, a probabilistic inference method using particle filtering is used to efficiently maintain multiple hypotheses about

the goals of the agents in the scene. This framework is used to classify trajectories in progress, allowing unusual behaviour to be detected as it is occurring instead of only after it has completed.

Chapter 4

Experimental Results

The experimental validation of this thesis aims to resolve two main questions. First, can the proposed learning approach achieve the same classification accuracy as the original approach that uses a manually specified obstacle map? Second, can the goal-based method provide good classification performance in comparison to traditional methods based on low-level trajectory features?

The first question is addressed by validating the proposed method using Dee and Hogg's carpark dataset and comparing to prior results [11]. The second question is addressed through experiments with Piciarelli *et al.*'s synthetic dataset [42] and Naftel and Khalid's real-world laboratory dataset [39]. Hence, all three datasets used in these experiments are published by third-party researchers working on trajectory-based unusual behaviour detection. The following subsections describe the experimental methodology and illustrate the performance of the proposed method on the three datasets.

4.1 Methodology

The proposed method classifies observed trajectories as normal or unusual. In general, a classifier can be validated by partitioning the data into two sets, and using one to train the classifier and the other to test the classifier. In many practical real-world situations, the amount of data is limited. For example, data collection may be expensive or the data

of interest may occur infrequently. When data is limited, classifier performance can be assessed using m-cross validation or the leave-one-out technique [13]. In m-cross validation, m separate training and testing sets are generated from the data and used to train and test the classifier. Results are averaged over the m cases. The leave-one-out method, also known as the jack-knifing method, is the limiting case of m-cross validation in which m equals the number of data points N. Each classifier is trained on $N-1$ data points and tested on one data point, and the results are averaged over the N cases. Leave-one-out is used in the experiments with the two real-world datasets. The synthetic dataset comprises data partitioned into large training and testing sets, so basic (1-cross) validation is appropriate for those experiments.

The parameters of classification algorithms can often be adjusted to shift the balance between false negatives and false positives. In many real-world situations, the cost of a false negative may be different from the cost of a false positive. In the particular case of an intelligent surveillance system, the cost of missing a suspiciously behaving individual may be different from the cost of unnecessarily drawing the attention of the human operator. In such situations, to simply minimize the probability of error is not appropriate. Rather, it is more appropriate to characterize the trade-off between false negatives and false positives. A standard technique to communicate this characterization is to use a Receiver Operating Characteristic (ROC) curve. A ROC curve plots the probability of detection (true positive) P_D on the vertical axis against the probability of false alarm (false positive) P_F on the horizontal axis. Points on the ROC curve represent attainable (P_F, P_D) combinations. An ideal classifier approaches the$(P_F, P_D) = (0, 1)$ point representing no false alarms and 100% probability of detection; a mediocre classifier approaches the diagonal $(0, 0)$ to $(1, 1)$. ROC curves are used in the reporting of experimental results for all three datasets below.

4.2 Piciarelli *et al.*'s synthetic dataset [42]

Since instances of suspicious behaviour are uncommon in real-world data, synthetic trajectory data are a good starting point for validating the proposed method. Experiments were

conducted using the synthetic dataset published by Piciarelli *et al.* in their work applying single-class SVM to detect unusual trajectories [42]. The objective of this set of experiments is to verify that the proposed method can identify anomalies in sets of relatively idealized trajectories.

Piciarelli *et al.*'s synthetic dataset contains training and testing sets for various combinations of the number of normal trajectory clusters (1 though 10) and the number of unusual trajectories (1 through 10). Figure 3.2 in Section 3.2.5 previously showed several sample training sets. The proposed algorithm is tested on 10 combinations, in which the number of normal trajectory clusters is varied from 1 to 10 and the number of unusual trajectories is kept fixed at the maximum 10. Since each combination is represented by 10 separate training and testing sets in the synthetic dataset, the proposed method is tested on a total of 100 training and testing sets.

Figures 4.1 and 4.2 shows the ROC curves for the ten combinations. A direct comparison with Piciarelli *et al.*'s reported results is not strictly possible, as their single-class SVM method includes a mechanism to detect and remove unusual trajectories from the training set, whereas in these experiments only the normal trajectories are used in training. Nevertheless, to provide a general benchmark for performance, the experimental result reported by Piciarelli *et al.* is indicated by an asterisk in the ROC plots. Recall that an ideal classification result approaches the $(P_F, P_D) = (0, 1)$ point representing no false alarms and 100% probability of detection. As of the time of writing, no other papers appear to have used this dataset for validation.

Overall, the proposed method produces very encouraging classification results. For example, a detection rate of 95% can be achieved with the following false alarm rates: 0%, 0.2%, 0.1%, 1.5%, 0.3%, 0.1%, 0.7%, 0.3%, 0.2%, and 1.8%, respectively, for 1 to 10 normal trajectory clusters.

The ROC curves show a trend of slight decline in classification performance as the number of normal trajectory clusters increases. This trend is expected: as the range of normal trajectories increases, synthetic trajectories generated from random parameters are less likely to appear unusual. The result is a slight drop in the probability of detection. The

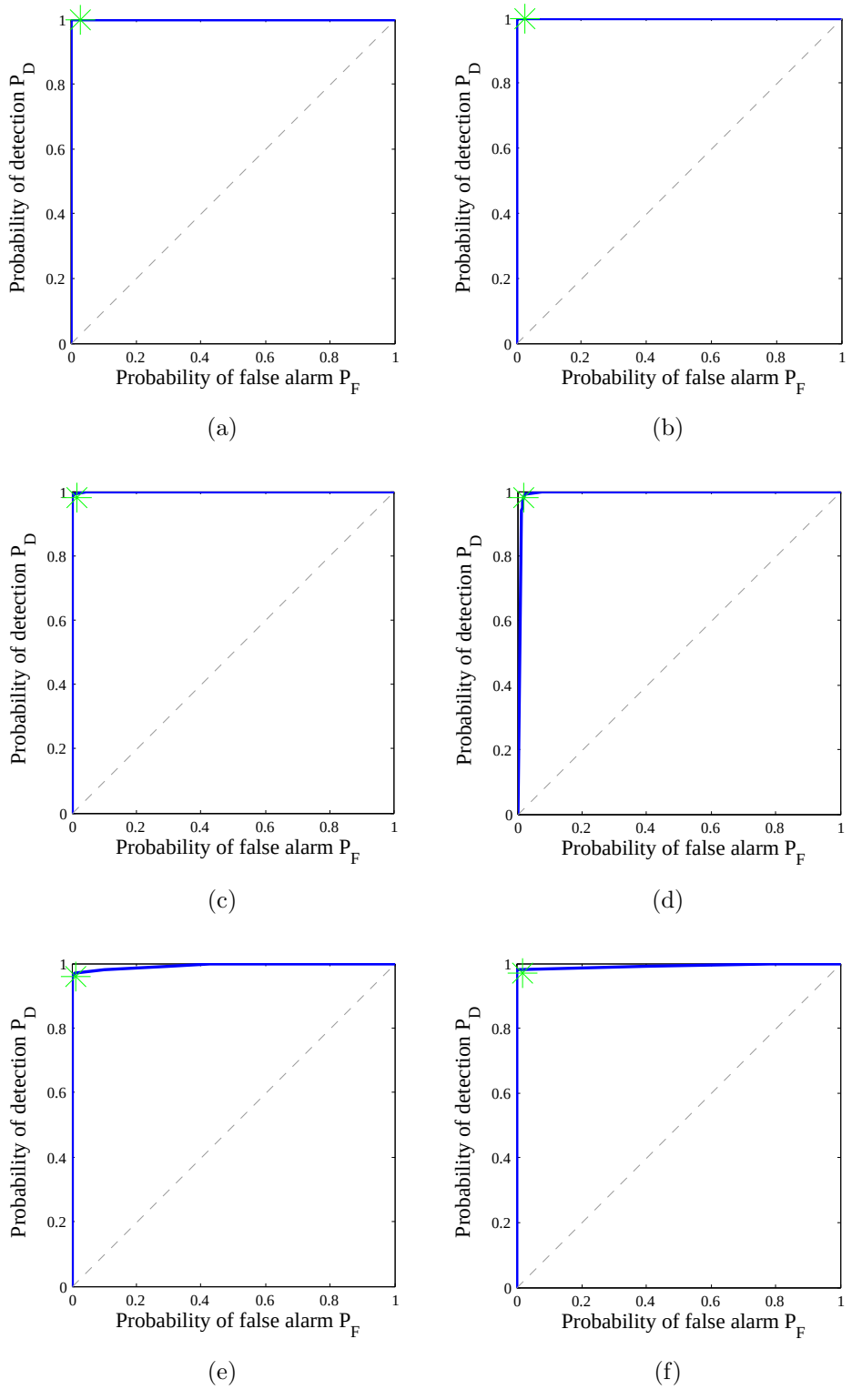

Figure 4.1: ROC curves for Piciarelli *et al.*'s synthetic dataset [42], varying the number of normal trajectory clusters from 1 to 6 (a-f, respectively).

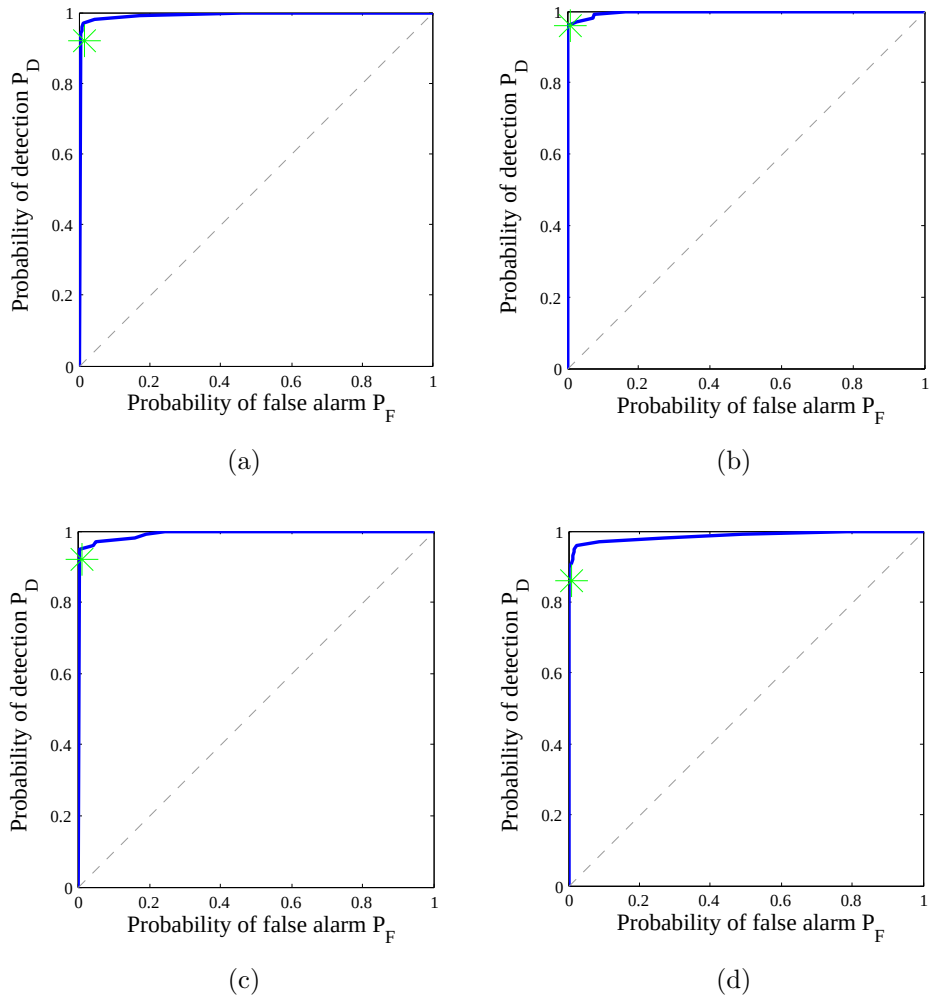

Figure 4.2: ROC curves for Piciarelli *et al.*'s synthetic dataset [42], varying the number of normal trajectory clusters from 7 to 10 (a-d, respectively).

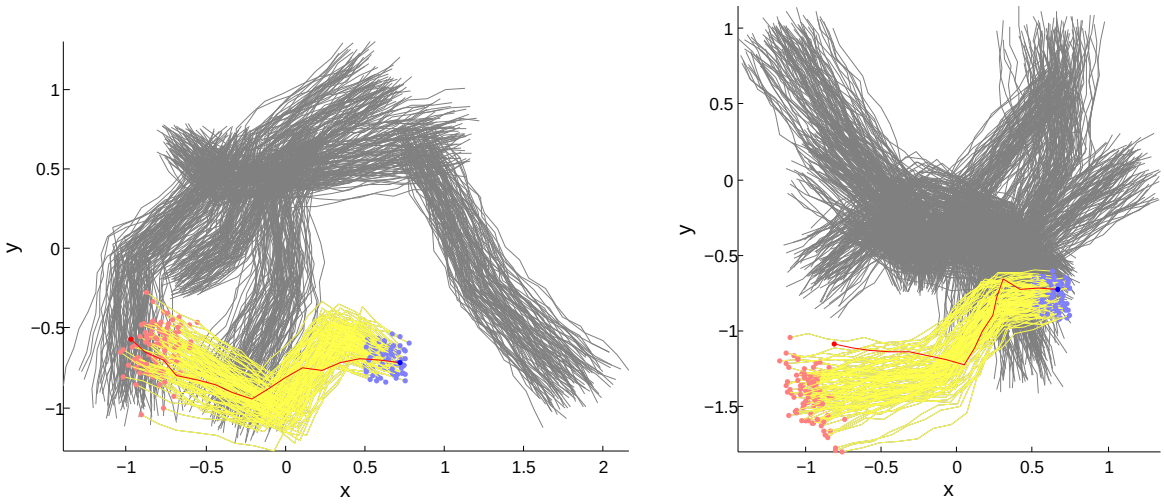

Figure 4.3: Examples of false negatives in Piciarelli *et al.*'s synthetic dataset [42], with two missed unusual trajectories highlighted in red and highly similar normal trajectory clusters highlighted in yellow. Trajectory entry and exit points are indicated by blue and red points, respectively.

same trend can be seen in the experimental results reported by Piciarelli *et al.* [42]. Figure 4.3 shows two examples of synthetic unusual trajectories (unusual according to ground truth) that the proposed method misclassifies as normal due to their high similarity to many normal trajectories. In each example, the misclassified unusual trajectory is highlighted in red, and a highly similar cluster of normal trajectories is shown in yellow. Following the colour conventions in Section 3.2, trajectory entry and exit points are indicated by blue and red points, respectively.

4.3 Naftel and Khalid's laboratory dataset [39]

Further validation of the proposed method was performed using the real-world dataset developed by Naftel and Khalid [39]. The trajectories in Naftel and Khalid's dataset are derived from video of people walking around their laboratory. The dataset consists of 152 normal trajectories falling into four pre-planned underlying classes and eight unusual trajectories. Although the trajectory classes are pre-planned, there is still significant spatial

and temporal variation in the trajectories, including instances of loitering and sudden changes in movement. Figure 4.4 shows the trajectories after pre-processing with a Kalman filter. The first four subplots highlight the four pre-planned trajectory classes, and the fifth subplot highlights the unusual trajectories. The objective of this set of experiments is to validate the proposed method on a trajectory dataset derived from real-world surveillance video but in a somewhat controlled environment.

The leave-one-out technique is applied to validate this dataset. Only the normal trajectories are used in the training phase: as mentioned in Section 3.2, all training trajectories are assumed to reflect normal behaviours.

Figure 4.5 shows the ROC curve for this dataset. The proposed method detects all unusual trajectories with eight false positives (5% probability of false alarm). In comparison, Piciarelli *et al.* also validated their method on this dataset and reported detecting all unusual trajectories with only two false positives [42]. Naftel and Khalid do not explicitly report the number of false positives. Therefore, in comparison, the proposed method has a higher tendency to issue false alarms on this dataset. However, the proposed method can offer at least two advantages. First, the proposed method is able to evaluate trajectories in progress, enabling suspicious behaviour to be detected as it is occurring instead of only after it has completed: both Piciarelli *et al.*'s method and Naftel and Khalid's method operate on complete trajectories. Second, the proposed method's goal-based approach allows for more intuitive natural language description of observed behaviours (as discussed in Chapter 3.1). For example, an individual in the scene might be described as "walking from the main entrance towards the work station".

4.4 Dee and Hogg's carpark dataset [9, 10, 11]

Since the proposed method is built on the foundation works of Dee and Hogg, validation using the carpark dataset used by Dee and Hogg in their experiments is appropriate. The carpark dataset is a challenging real-world dataset in which, in contrast to Naftel and Khalid's laboratory dataset, the normal trajectories are completely unscripted. The

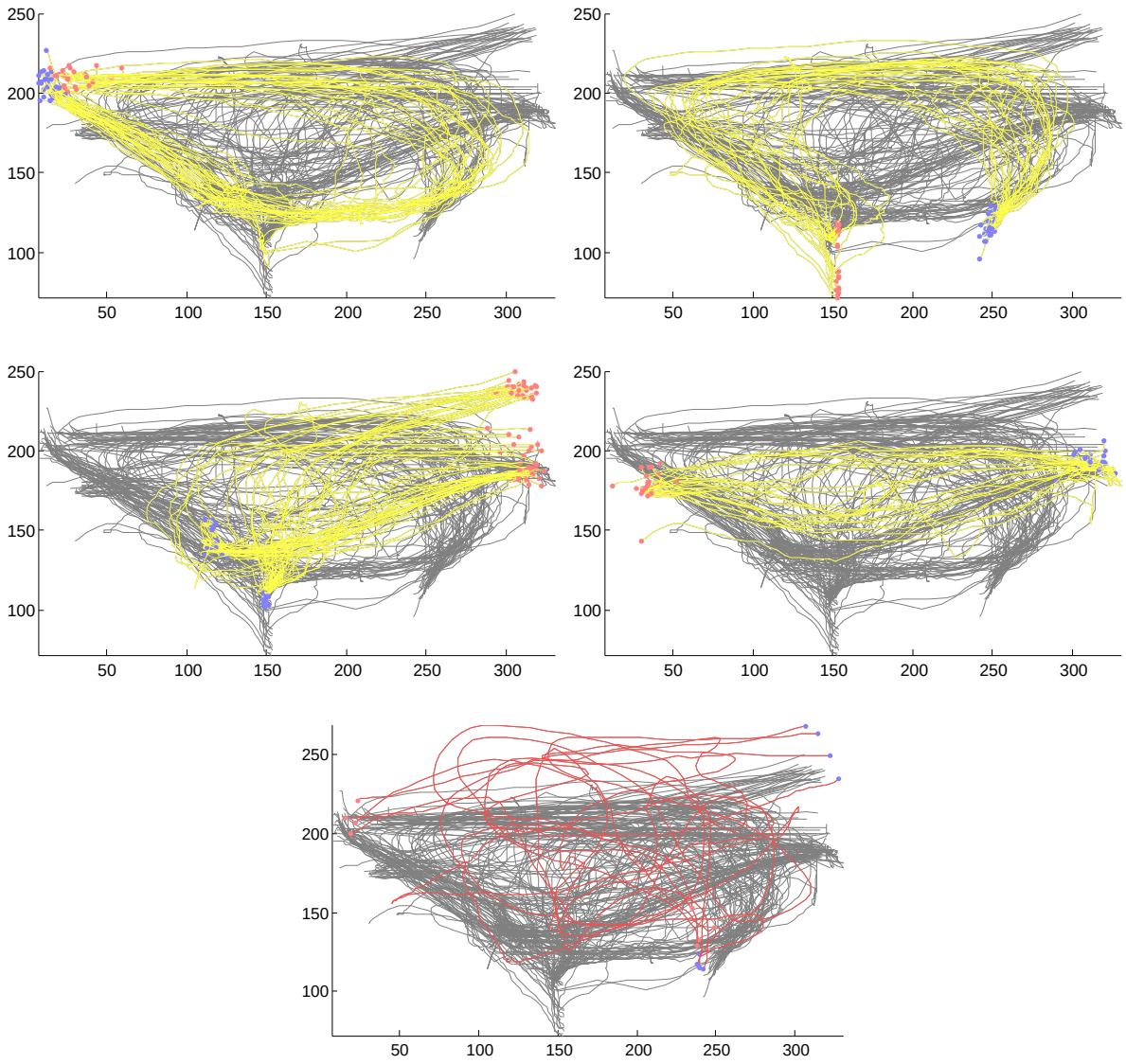

Figure 4.4: Overlays of the trajectories in Naftel and Khalid's laboratory dataset [39]; the top four subplots highlight the pre-planned trajectory classes and the fifth subplot highlights the unusual trajectories.

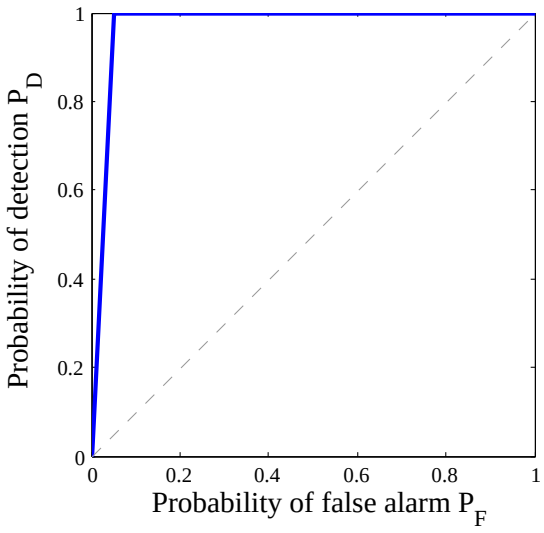

Figure 4.5: ROC curve for Naftel and Khalid's laboratory dataset [39].

dataset consists of 262 normal trajectories from surveillance video captured over an hour and six trajectories of volunteers instructed to "behave in a suspicious fashion" though without any knowledge of Dee and Hogg's goal-based method of analysis [11]. Ground plane coordinates are used. The objective of this set of experiments is to validate the proposed method on a trajectory dataset derived from challenging real-world surveillance video that has also been used in the development of the foundation.

The trajectories are subsampled at 3 frames per second and as in Dee and Hogg's experiments, smoothed using a Kalman filter to reduce the effects of noise. Manual editing of trajectories to correct tracking errors was not performed as in [9, 10, 11]. Figure 4.6 shows the filtered carpark trajectories in the ground plane. The unusual trajectories are highlighted in red. Similar to the Naftel and Khalid dataset, the leave-one-out technique is applied and only the normal trajectories are used in the training phase.

Figure 4.7 shows the ROC curve for this dataset. Besides Dee and Hogg, the carpark dataset has also been used for experimental validation by Sillito and Fisher [47]. Sillito and Fisher's work was described earlier in Section 2.1. The proposed method detects all unusual trajectories with 22% probability of false alarm, which is comparable to both Dee and Hogg's method [11] and Sillito and Fisher's method [47] (approximately 25% and 24%, respectively).

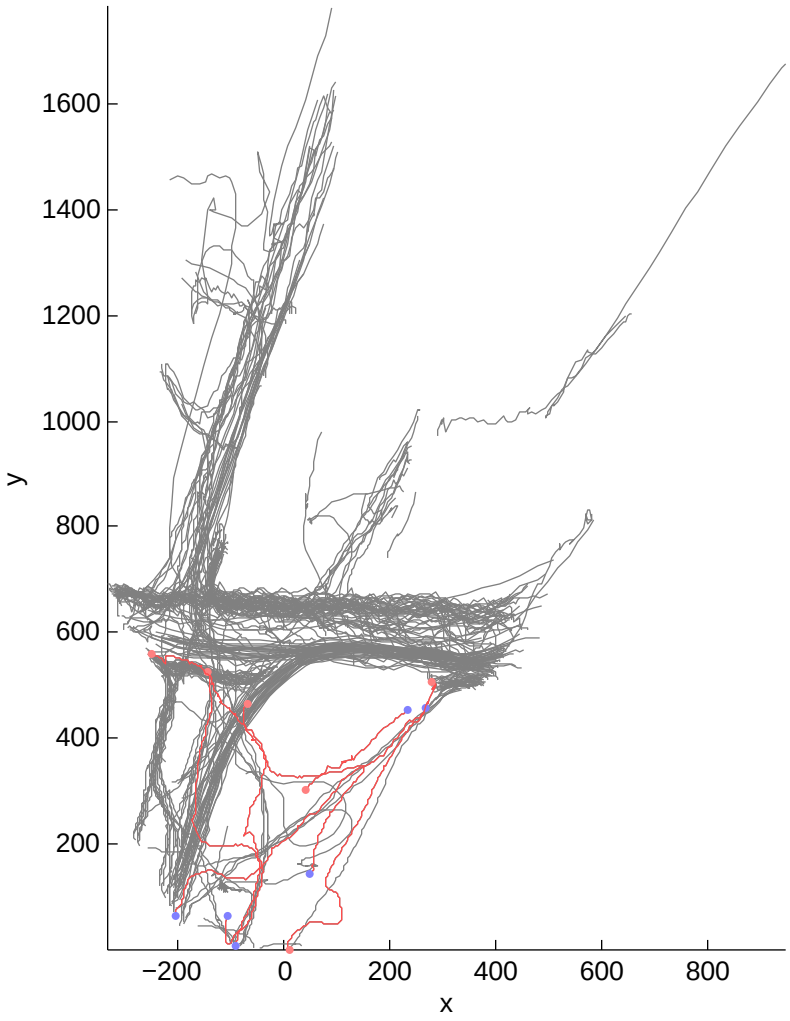

Figure 4.6: Overlay of the trajectories in Dee and Hogg's carpark dataset [9, 10, 11], with the unusual trajectories highlighted in red.

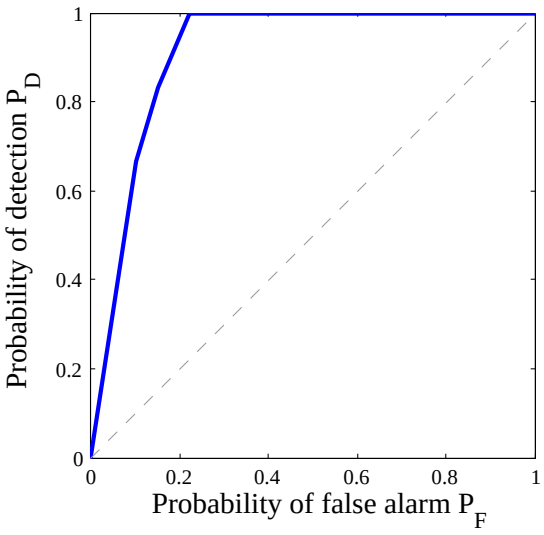

Figure 4.7: ROC curve for Dee and Hogg's carpark dataset [9, 10, 11].

4.5 Discussion and limitations

The results of the carpark experiments indicate that the performance of the classifier depends on the completeness of the training set. In many cases, the false positives correspond to trajectories that do actually appear novel with respect to the rest of the carpark dataset. A human might view the agent's trajectory as normal based on prior domain knowledge about how people generally behave in carparks or similar outdoor settings. However, an intelligent surveillance system has to build most of this domain knowledge from scratch using training samples. If the training samples are not sufficiently representative of the types of behaviour that should be considered normal, then the system is likely to issue false alarms. This limitation pertains to any machine learning or pattern recognition algorithm that uses exemplar-based learning. The collection of more training data is a straightforward strategy to mitigate this risk. In the case of intelligent surveillance, this strategy is practical assuming the cost of monitoring the scene using a tracking algorithm does not increase significantly with duration. The assumption is reasonable since tracking algorithms do not normally require human intervention beyond initialization. In the general case, the strategy may not be practical when obtaining training samples is expensive. A second way to address the limitation would be to encode additional domain knowledge by

incorporating a rule-based knowledge base, as in an expert system [44]. For example, in the context of intelligent surveillance, a heuristic rule based on domain knowledge might be to issue an alarm if groups of agents suddenly stop, or if an agent loiters in front of a bank machine [12].

One of the limitations of any trajectory-based method for unusual behaviour detection is that some forms of suspicious behaviour cannot be detected using trajectory features alone. Events of surveillance interest such as theft or abandoned baggage may not be accompanied by overall unusual trajectories. Detecting these events or actions may require other techniques, ranging from simple background subtraction [48] to specialized human action recognition methods [3]. Trajectory analysis can still contribute in many cases. For example, a common pattern seen with theft is that the perpetrator will approach the target, circle or examine it, and then retreat before striking [12]. A robust intelligent surveillance system will likely require the fusion of both trajectory and action analyses.

Although not included in this work, some methods in the literature incorporate stopping regions in their scene model [33]. In many surveillance settings, constructing a map of normal or expected stopping regions may provide useful scene knowledge. For instance, in an outdoor courtyard it may be expected for individuals to sit down at a fountain or on benches. Temporal constraints may be applicable: an individual waiting at a train platform for excessive time - even after several trains have passed - should probably be flagged as suspicious [12].

Goal-based trajectory analysis has applications beyond intelligent surveillance. Liao *et al.*'s work [29, 30] demonstrates the broader applicability of this research in such areas as geomatics and intelligent transportation systems. Liao *et al.* applied a geographic goal-based approach to analyze user trajectories at the city scale. Agent location is estimated using portable GPS and motion is constrained to the streets in a street map. Probabilistic inference is performed using Rao-Blackwellized particle filtering. The authors envision their work to be applied in personal guidance systems and just-in-time transportation information services.

Chapter 5

Conclusion and Future Work

Many traditional methods for trajectory-based unusual behaviour detection rely on low-level features such as flow vectors or control points. In contrast, Dee and Hogg's goal-based approach models individuals in the scene as intelligent agents with intentionality, and detects unusual behaviour by evaluating the explicability of the agent's trajectory with respect to known spatial goals. This thesis advances the goal-based framework in three main ways: the spatial scene model is learned in a training phase; a region transition model is learned to incorporate statistics of movement between spatial regions; and trajectories in progress are classified in a probabilistic framework using particle filtering. Experimental results on three datasets published by third-party researchers in trajectory-based unusual behaviour detection demonstrate the validity of the proposed approach.

In the present implementation of this thesis, training trajectories are assumed to correspond to normal behaviour. As future work it would be valuable to investigate methods for automatically detecting and removing outlier trajectories from the training set. As discussed in Section 4.5, the incorporation of stopping regions in the scene model can also be investigated. Finally, the fusion of trajectory and action recognition analyses is an important research direction in the long term for robust behaviour analysis in intelligent surveillance systems.

References

[1] H. Akaike. A new look at the statistical identification model. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. 20

[2] S. Ali and M. Shah. A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–6, Minneapolis, MN, 2007. 13, 14

[3] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001. 49

[4] B.A. Boghossian and S.A. Velastin. Motion-based machine vision techniques for the management of large crowds. In *Proc. IEEE International Conference on Electronics, Circuits and Systems*, volume 2, pages 961–964, Pafos, Cyprus, 1999. 12

[5] A. Cavoukian. Privacy and video surveillance in mass transit systems: a special investigation report. Technical Report MC07-68, Information and Privacy Commissioner of Ontario, 2008. 12

[6] S.S. Chen and P.S. Golapalakrishnan. Clustering via the Bayesian Information Criterion with applications in speech recognition. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 645–648, Seattle, WA, 1998. 20

[7] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 142–149, Hilton Head Island, SC, 2000. 5

[8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, 2003. 5

[9] H.M. Dee and D.C. Hogg. On the feasibility of using a cognitive model to filter surveillance data. In *Proc. IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 34–39, Como, Italy, 2005. vi, viii, ix, 3, 11, 18, 20, 23, 44, 46, 47, 48

[10] H.M. Dee and D.C. Hogg. Navigational strategies and surveillance. In *Proc. IEEE International Workshop on Visual Surveillance*, pages 73–81, Graz, Austria, 2006. vi, viii, ix, 3, 11, 12, 18, 20, 23, 32, 44, 46, 47, 48

[11] H.M. Dee and D.C. Hogg. Navigational strategies in behaviour modelling. *Artificial Intelligence*, 173(2):329–342, 2009. vi, viii, ix, 3, 11, 15, 18, 20, 23, 32, 38, 44, 46, 47, 48

[12] H.M. Dee and S.A. Velastin. How close are we to solving the problem of automated visual surveillance? *Machine Vision and Applications*, 19(5-6):329–343, 2008. 1, 2, 49

[13] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, second edition, 2001. 39

[14] J. Fernyhough, A. G. Cohn, and D.C. Hogg. Constructing qualitative event models automatically from video input. *Image and Vision Computing*, 18:81–103, 2000. 8, 15

[15] I. Haritaoglu and M. Flickner. Detection and tracking of shopping groups in stores. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I431–I438, Kauai, HI, 2001. 5

[16] X.C. He and N.H.C. Yung. Curvature scale space corner detector with adaptive threshold and dynamic region of support. In *Proc. International Conference on Pattern Recognition*, volume 2, pages 791–794, Cambridge, UK, 2004. 22

[17] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995. 13

[18] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviours. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 34(3):334–352, 2004. 2, 17

[19] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, 2006. 7, 15

[20] N. Ihaddadene and C. Djeraba. Real-time crowd motion analysis. In *Proc. International Conference on Pattern Recognition*, pages 1–4, Tampa, FL, 2008. 13

[21] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 30

[22] M. Isard and A. Blake. A mixed-state CONDENSATION tracker with automatic model-switching. In *Proc. International Conference on Computer Vision*, pages 107–112, Bombay, India, 1998. 30

[23] A. Jacquot, P. Sturm, and O. Ruch. Adaptive tracking of non-rigid objects based on color histograms and automatic parameter selection. In *Proc. IEEE Workshop on Motion and Video Computing*, volume 2, pages 103–109, Breckenridge, CO, 2005. 20

[24] F. Jiang, Y. Wu, and A.K. Katsaggelos. A dynamic hierarchical clustering method for trajectory-based unusual video event detection. *IEEE Transactions on Image Processing*, 18(4):907–913, 2009. 10, 15

[25] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996. 6, 7, 15

[26] I.N. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. In *Proc. International Conference on Pattern Recognition*, volume 2, pages 716–719, Cambridge, UK, 2004. 9, 15

[27] Kingston University, M. MacDonald, and Ipsotek Ltd. Maximising benefits from CCTV on the railway existing systems. Technical Report T061b, Rail Safety and Standards Board, 2003. 1

[28] P. Li. An adaptive binning color model for mean shift tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(9):1293–1299, 2008. 5

[29] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *Proc. AAAI National Conference on Artificial Intelligence*, pages 348–353, San Jose, CA, 2004. 49

[30] L. Liao, D.J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007. 49

[31] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000. 30

[32] D. Makris and T. Ellis. Path detection in video surveillance. *Image and Vision Computing*, 20:895–903, 2002. 8, 9, 15, 18, 20

[33] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(3):397–408, 2005. 1, 8, 9, 15, 18, 20, 49

[34] K. Martin and K.N. Plataniotis. Privacy protected surveillance using secure visual object coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1152–1162, 2008. 12

[35] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, pages 935–942, Miami, FL, 2009. 13

[36] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, 1998. 21, 22

[37] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. AAAI National Conference on Artificial Intelligence*, pages 593–598, Edmonton, AB, 2002. 30

[38] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. International Joint Conference on Artificial Inteligence*, pages 1151–1156, Acapulco, Mexico, 2003. 30

[39] A. Naftel and S. Khalid. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Multimedia Systems*, 12(3):227–238, 2006. vi, ix, 10, 15, 38, 43, 45, 46

[40] J. Owens and A. Hunter. Application of the self-organizing map to trajectory classification. In *Proc. IEEE International Workshop on Visual Surveillance*, pages 77–83, Dublin, Ireland, 2000. 6, 15

[41] C. Piciarelli and G.L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27:1835–1842, 2006. 9, 15

[42] C. Piciarelli, C. Micheloni, and G.L. Foresti. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1544–1554, 2008. vi, viii, 9, 15, 22, 24, 38, 39, 40, 41, 42, 43, 44

[43] A. Rattarangsi and R.T. Chin. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(4):430–449, 1992. 21, 22

[44] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, second edition, 2003. vii, 19, 25, 26, 28, 49

[45] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. 20

[46] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, 1994. 13

[47] R.R. Sillito and R.B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proc. British Machine Vision Conference*, pages 1035–1044, Leeds, UK, 2008. 11, 15, 46

[48] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000. 7, 15, 49

[49] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2006. vii, viii, 26, 27, 28, 30, 31

[50] T. Xiang and S. Gong. Video behaviour profiling and abnormality detection without manual labelling. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 1238–1245, Beijing, China, 2005. 13

[51] T. Xiang and S. Gong. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41(3):1012–1029, 2008. 20

[52] X.L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991. 7

[53] W. Yan and D.A. Forsyth. Learning the behavior of users in a public space through video tracking. In *Proc. IEEE Workshop on Applications of Computer Vision*, pages 370–377, Breckenridge, CO, 2005. 18

[54] A. Yilmaz. Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, Minneapolis, MN, 2007. 5

[55] A. Yilmaz, O. Javed, and M. Shah. Object tracking: a survey. *ACM Computing Surveys*, 38(4), 2006. 5

[56] H. Zhong, J. Shi, and M. Visontai. Detecting unusual events in video. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 819–826, Washington, DC, 2004. 6