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

Frederick Tung[*], John S. Zelek, David A. Clausi

Vision and Image Processing Lab, Systems Design Engineering, University of Waterloo, 200 University Ave. West, Waterloo, Ontario, Canada, N2L 3G1

Abstract

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 such as 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 such as flow vectors or control points, this paper builds upon a recently introduced approach that makes use of higher-level features of intentionality. Individuals in the scene are modelled as intentional agents, and unusual behaviour is detected by evaluating the explicability of the agent's trajectory with respect to known 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.

Keywords: video surveillance, behaviour understanding, trajectory

[*]Corresponding author. Tel.: +1 519 888-4567 x35342; fax: +1 519 746 4791
Email addresses: `ftung@uwaterloo.ca` (Frederick Tung), `jzelek@uwaterloo.ca` (John S. Zelek), `dclausi@uwaterloo.ca` (David A. Clausi)

analysis, anomaly detection

1. Introduction

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 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. 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 [1, 2]. These conditions make surveillance challenging in large installations such as airports.

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 [3]. An alternative formulation identifies the similar stages of low-level image processing, tracking, and high-level scene and/or behaviour analysis [1]. This work 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.

As described in Section 2, 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 [4, 5, 6]. 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, the explicability of the trajectory with respect to the known spatial goals in the scene.

This paper builds upon Dee and Hogg's goal-based paradigm and extends it in three ways: first, the spatial scene structure is learned in a training phase; second, a transition model that describes normal agent movement between spatial regions is learned; and third, trajectories in progress are classified in a probabilistic framework using particle filtering. 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.

The rest of this paper 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 three published third-party datasets. Discussion and limitations are presented in Section 5. Finally, conclusions and directions for future research are outlined in Section 6.

2. Related work

This paper focuses on the high-level scene and behaviour analysis stage in the intelligent video surveillance framework, which takes as input the results of the object tracking stage [1, 3]. Object tracking is a challenging problem, especially in the presence of occlusion, noise, camera motion, and changes in illumination [7]; however, tracking issues are beyond the scope of this work. The interested reader is referred to the review by Yilmaz *et al.* [7] for a comprehensive treatment of the state-of-the-art in object tracking.

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 behaviours 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 [8] may be useful for post-event 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 [9]. 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.

Owens and Hunter proposed a method for detecting unusual trajectories in greyscale surveillance video [10]. 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.

Stauffer and Grimson performed online k-means clustering on flow vectors augmented with object size information [11]. 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.

Hu *et al.* developed a combined tracking and behaviour understanding system [12]. 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 [13]; however, if the number of samples in a cluster falls below a fixed threshold, the cluster is merged with its closest neighbour in the feature space. 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.

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" [14]. 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) [2].

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 [15]. Like Makris and Ellis [2, 14], 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") [15].

Piciarelli and Foresti [16] took a similar approach to Makris and Ellis [2, 14] 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.

In a later work, Piciarelli *et al.* proposed using a single-class support vector machine (SVM) to detect unusual trajectories [17]. 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.

Junejo *et al.* clustered raw trajectories using a min-cut graph algorithm [18]. 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.

Naftel and Khalid represented trajectories using function approximations, including least square polynomial, Cheybyshev polynomial, and Discrete Fourier Transform [19]. 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.

Sillito and Fisher proposed a semi-supervised method for learning normal trajectories and detecting unusual behaviour [20]. 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.

Most of the previous approaches to trajectory-based detection of unusual behaviour 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 [4, 5, 6]. Individuals in the scene are modelled as intelligent agents instead of simply "objects" [4, 5, 6], and trajectories are evaluated based on their explicability with respect to known spatial goals. The greater the goal-directedness of a trajectory, the more likely it represents normal behaviour. In [4], 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 [5]. 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. In a later work [6], a weighted sum of Euclidean distance and angular disparity is used as the distance measure, with point correspondences found using dynamic programming.

3. Proposed method

This work builds upon the goal-based framework of Dee and Hogg [4, 5, 6] in three main ways.

First, as described in Section 3.1, 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 polygon 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 annotation of polygonal obstacle models for every scene is time-consuming from a usability perspective and may be subject to bias. The proposed method's automatic learning of spatial regions in the scene reduces the human involvement required in training the system, at the cost of losing the ground-truth accuracy of a manually specified obstacle map.

Second, as described in Section 3.2, 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. 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.

Third, as described in Section 3.3, the proposed method classifies trajectories in progress in a probabilistic framework using particle filtering. The probabilistic approach is novel with respect to the original goal-based 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.

3.1. Learning the scene structure

The proposed method models the spatial scene structure using three separate Gaussian mixture models (GMMs) learned 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. The GMMs are fit to the training data using the Expectation Maximization (EM) algorithm [21]. Learning of the scene structure is performed in an offline training phase and is not updated online during system operation.

Modelling entry and exit regions using GMMs has precedent in the works of Makris and Ellis [2, 14] as well as Dee and Hogg [4, 5, 6]. In addition, McKenna and Nait Charif proposed modelling entry, exit, and inactivity regions using GMMs for the detection of unusual inactivity in a supportive home environment for seniors [22]. Falls in the home are inferred by prolonged inactivity in an area that is not a normal inactivity region. In the context of tracking people in crowded scenes, Ali and Shah modelled regions of interest using floor fields from evacuation dynamics [23]. Attractive regions, such as exit regions and densely travelled paths, are captured in a static floor field; repulsive regions, such as walls and opposing crowd flows, are captured in a boundary floor field.

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. 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) [24, 25] 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 [25]:

$$\kappa = \frac{d\psi}{ds} \tag{1}$$

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 [25]

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

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 [24, 25]

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

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 [26], 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 [26]). The rest of He and Yung's CSS corner detector is unmodified.

For Makris and Ellis [2, 14] as well as Dee and Hogg [4, 5, 6], 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) [27]. 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) [28].

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 [29].

Figure 1 illustrates the overall scene modelling approach on sample training sets from Piciarelli *et al.*'s synthetic dataset [17], Dee and Hogg's carpark dataset [4, 5, 6], and Naftel and Khalid's laboratory dataset [19]. The training trajectories are overlaid in gray. Entry, turning, and exit points are indicated by blue, green, and red points respectively. The asterisks denote

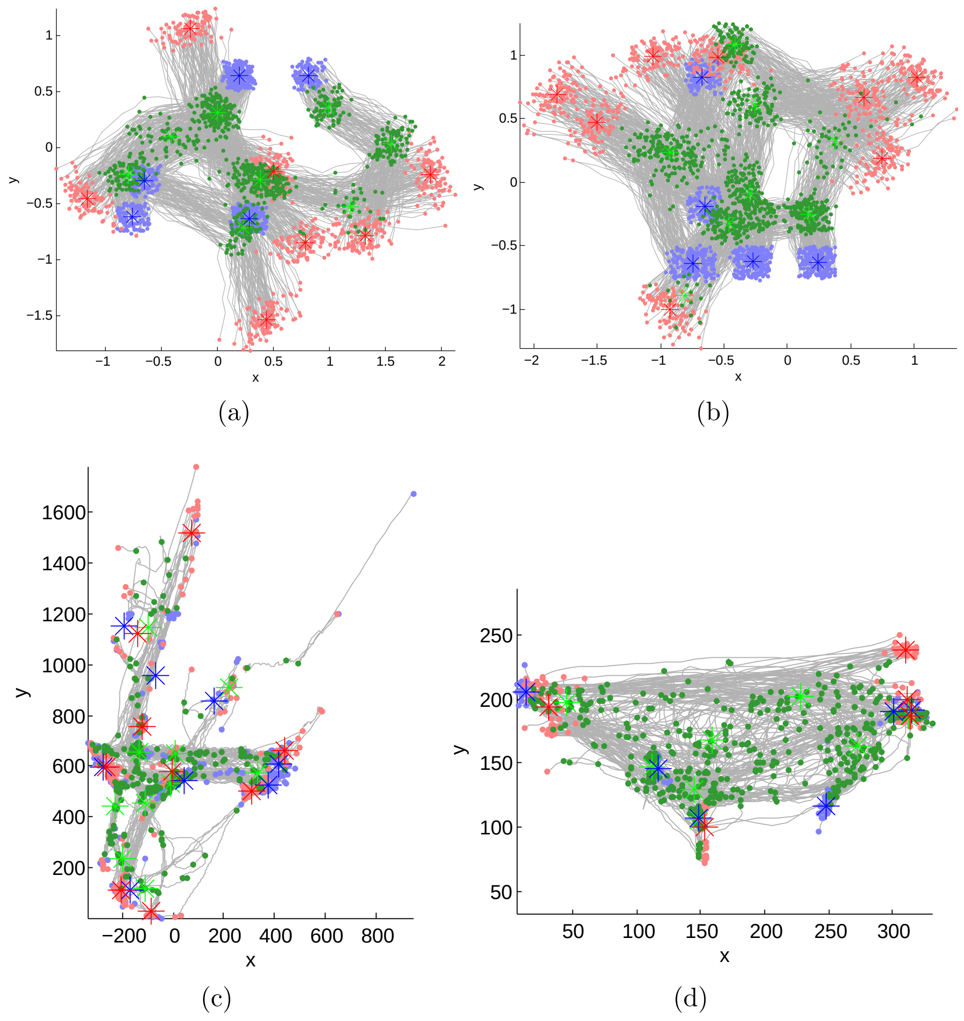

Figure 1: Sample learned scene models for training sets from (a, b) Piciarelli *et al.*'s synthetic dataset [17]; (c) Dee and Hogg's carpark dataset [4, 5, 6]; (d) Naftel and Khalid's laboratory dataset [19]. Entry, turning, and exit points are indicated by blue, green, and red points respectively. Asterisks denote the component means of the entry, turning, and exit GMMs.

the mixture component means of the entry, turning, and exit GMMs.

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. 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 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 that the agent will choose upon arriving at his/her current destination region. As will be discussed in Section 3.3, 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 [21]. 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 simple 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 3.1. 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.

3.3. Classifying trajectories in progress

Particle filtering is a well-established and efficient algorithm for probabilistic reasoning over time. We review it briefly here; the interested reader is referred to more complete treatments of the topic in [21] and [30], from which most of the following discussion is based.

The objective of a probabilistic filtering algorithm is to compute the belief distribution: the posterior probability distribution over the current state, conditioned on all observations so far: $bel(x_t) = p(x_t|z_{1:t})$. The particle filtering algorithm approximates the belief distribution using a finite number of samples, referred to as particles, drawn from the belief distribution. Each particle encapsulates an instantiation of the state. A particle can also be interpreted as a hypothesis about the true state [30]. 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.

The particle filtering algorithm is summarized in Table 1. Two steps are involved in each iteration: a prediction step and a correction step. In the prediction step, the particles are propagated forwards according to the state transition model $p(x_t|x_{t-1})$. 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

Table 1: An iteration of the particle filtering algorithm [21, 30]

```
Inputs: particle set $X_{t-1}$, observation $z_t$ ($t$ is the time index)
Denote the number of particles in the particle set as $M$.
**for** each particle $x^i_{t-1} \in X_{t-1}$ **do**
    Sample $x^i_t \sim p(x_t|x_{t-1})$
    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** $X_t$
```

new particle set contains the same number of particles as the original particle set and is not weighted.

The particle filtering algorithm and its variants have been applied successfully in a variety of computer vision domains, including in object tracking [31, 32, 33], as well as in robotics for simultaneous localization and mapping [34, 35].

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^1_t, current path segment end position d_t, and previous path segment e^0_t:

$$x_t = \left(s_t,\ v_t,\ e^1_t,\ d_t,\ e^0_t\right)^T \tag{4}$$

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 [5, 6]. 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 2. If the distance to the path segment end position d_{t-1} is greater than the speed v_{t-1} (in distance per time step), 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^1_{t-1} and e^0_{t-1}. 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^1_t, e^0_t, 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 assumption.

The speed v_t is sampled from the learned speed model conditioned on the current path segment e^1_t. 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.

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. To select an appropriate threshold in practice, the human operator can generate an ROC curve by running experiments similar to the ones described in Section 4 before system operation. The threshold corresponding to the desired performance measure (e.g. 95% detection rate) can then be used. An interesting direction for future study would be the online adaptation of the threshold based on human operator feedback during system operation; such a mechanism would require less human involvement in setting up the system.

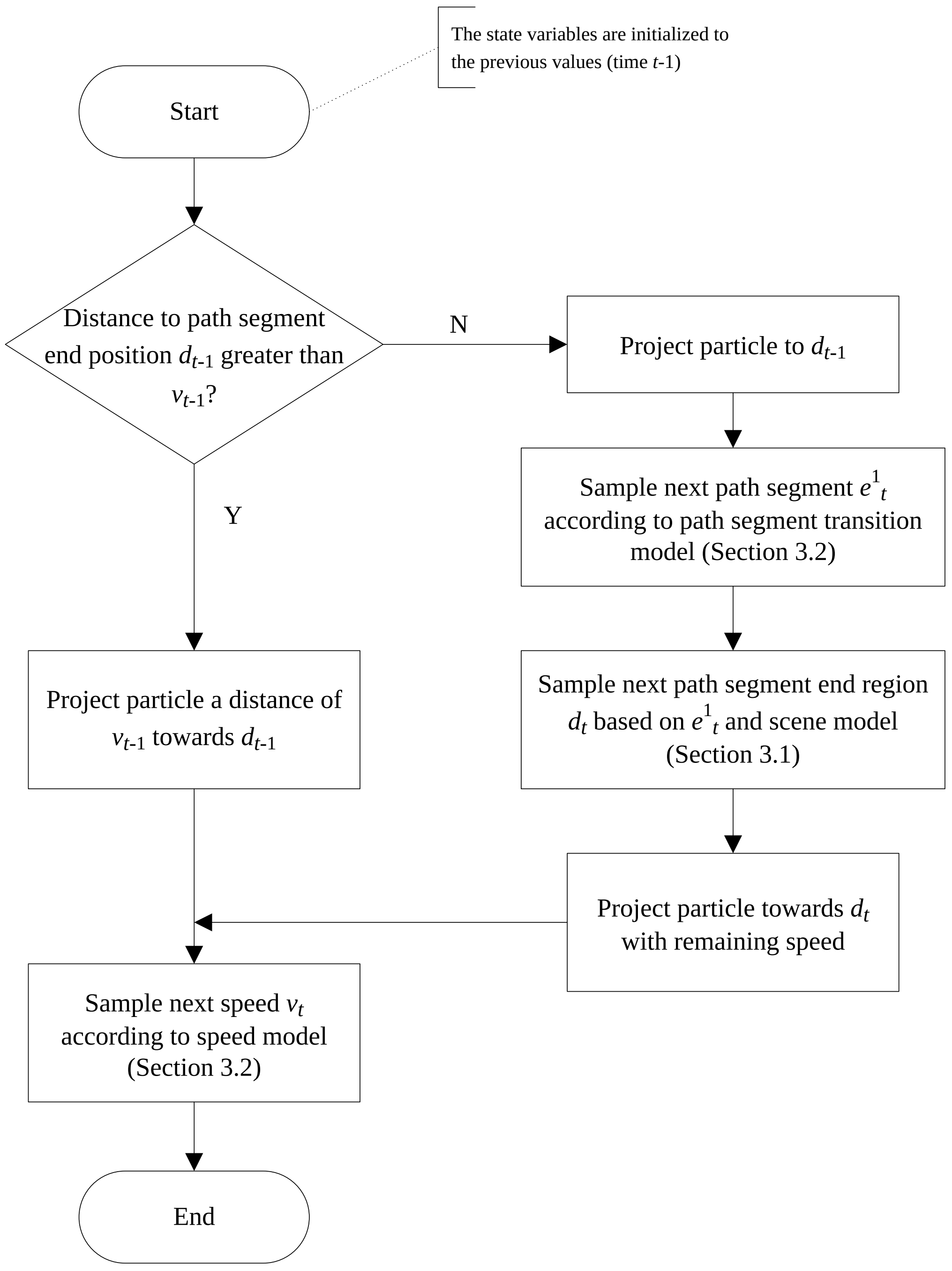
The state variables are initialized to the previous values (time t-1)
Start
Distance to path segment end position $d_{t\text{-}1}$ greater than $v_{t\text{-}1}$?
N
Project particle to $d_{t\text{-}1}$
Sample next path segment e^1_t according to path segment transition model (Section 3.2)
Y
Project particle a distance of $v_{t\text{-}1}$ towards $d_{t\text{-}1}$
Sample next path segment end region d_t based on e^1_t and scene model (Section 3.1)
Project particle towards d_t with remaining speed
Sample next speed v_t according to speed model (Section 3.2)
End

Figure 2: Summary flowchart of the state transition model $p(x_t|x_{t-1})$. The state transition model governs the forward propagation of particles in the prediction step of particle filtering.

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

$$P(C_i|x) > P(C_j|x) \quad \forall j, i \neq j \tag{5}$$

Using Bayes' rule, this condition is equivalent to

$$p(x|C_i)P(C_i) > p(x|C_j)P(C_j) \quad \forall j, i \neq j \tag{6}$$

which can be readily evaluated according to the learned GMM.

To enforce the spatial constraints of the entry regions, a trajectory is immediately flagged as unusual if the Mahalanobis distance between the starting point and the MAP-classified entry region exceeds 4 (intuitively, the entry point is more than 4σ from the mean of the entry region Gaussian). Otherwise, the particle set is created and uniformly initialized to $e^1 = (null, i), e^0 = (null, null), d =$ (entry point), $s =$ (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.

3.4. 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.

4. Experimental results

The experimental validation of the proposed method 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 [6]. The second question is addressed through experiments with Piciarelli *et al.*'s synthetic dataset [17] and Naftel and Khalid's real-world laboratory dataset [19]. Hence, all three datasets used in these experiments are published by third-party researchers working on trajectory-based unusual behaviour detection.

In intelligent surveillance, the cost of missing a suspiciously behaving individual may be different from the cost of unnecessarily drawing the attention of the human operator. Therefore, characterization of the trade-off between false negatives and false positives is appropriate. Receiver Operating Characteristic (ROC) curves, which plot the probability of detection (true positive) P_D against the probability of false alarm (false positive) P_F, are used in the reporting of experimental results for all three datasets.

4.1. Piciarelli et al.*'s synthetic dataset [17]*

Since instances of suspicious behaviour are uncommon in real-world data, synthetic trajectory data are a good starting point for validation. Experiments were conducted using the synthetic dataset published by Piciarelli *et al.* in their work applying single-class SVM to detect unusual trajectories [17].

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 1 previously showed two 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

Table 2: False alarm rates for 95% detection rate on Piciarelli *et al.*'s synthetic dataset [17]

# normal trajectory clusters	1	2	3	4	5	6	7	8	9	10
False alarm rate for 95% detection	0%	0.2%	0.1%	1.5%	0.3%	0.1%	0.7%	0.3%	0.2%	1.8%

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 3 and 4 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. 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. Table 2 lists the false alarm rates for a detection rate of 95%, for 1 to 10 normal trajectory clusters.

4.2. Naftel and Khalid's laboratory dataset [19]

Further validation of the proposed method was performed using the real-world dataset developed by Naftel and Khalid [19]. 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.

The leave-one-out (or jack-knifing) technique [36] is applied to validate this dataset. Only the normal trajectories are used in the training phase: as mentioned in Section 3.1, all training trajectories are assumed to reflect normal behaviours. Trajectories are pre-processed using a Kalman filter to reduce the effects of tracker noise. On average, six entry regions, seven turning regions, and five exit regions are learned in the scene model.

Figure 5 shows the ROC curve for this dataset. The proposed method detects all unusual trajectories with eight false positives (5% probability of

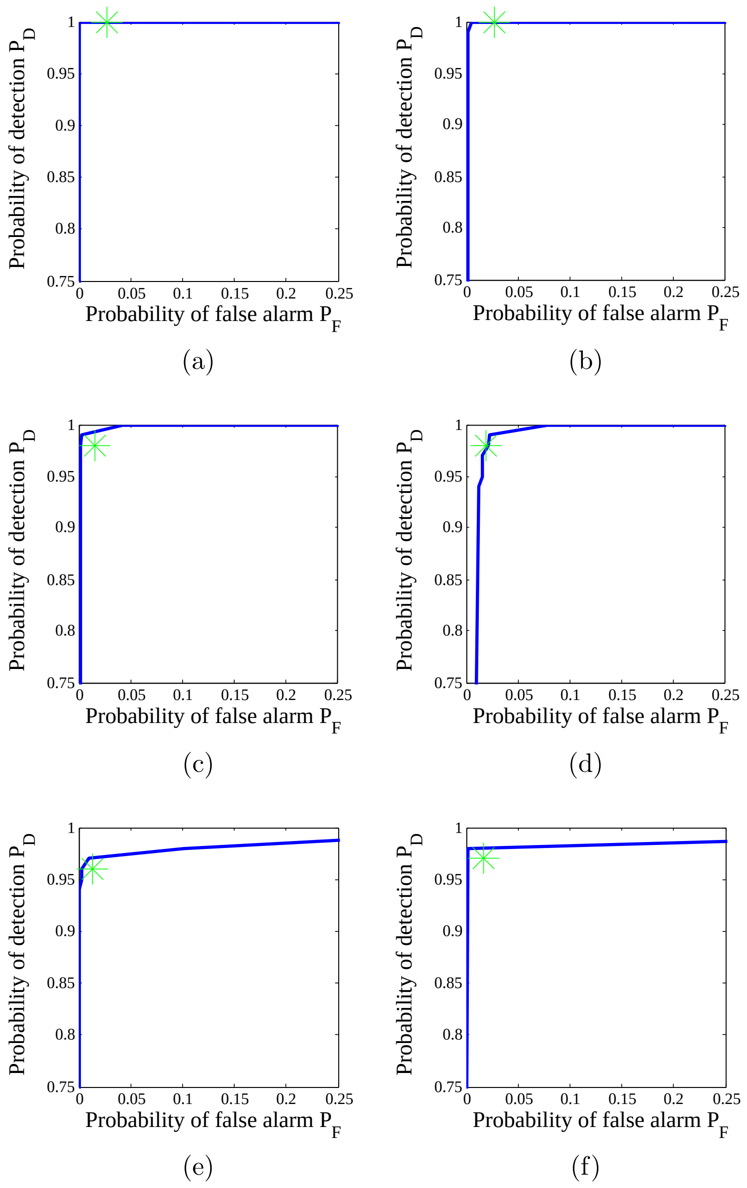

Figure 3: ROC curves for Piciarelli *et al.*'s synthetic dataset [17], varying the number of normal trajectory clusters from 1 to 6 (a-f, respectively). The experimental result reported by Piciarelli *et al.* is indicated by an asterisk. For visual clarity, the axes have been zoomed in.

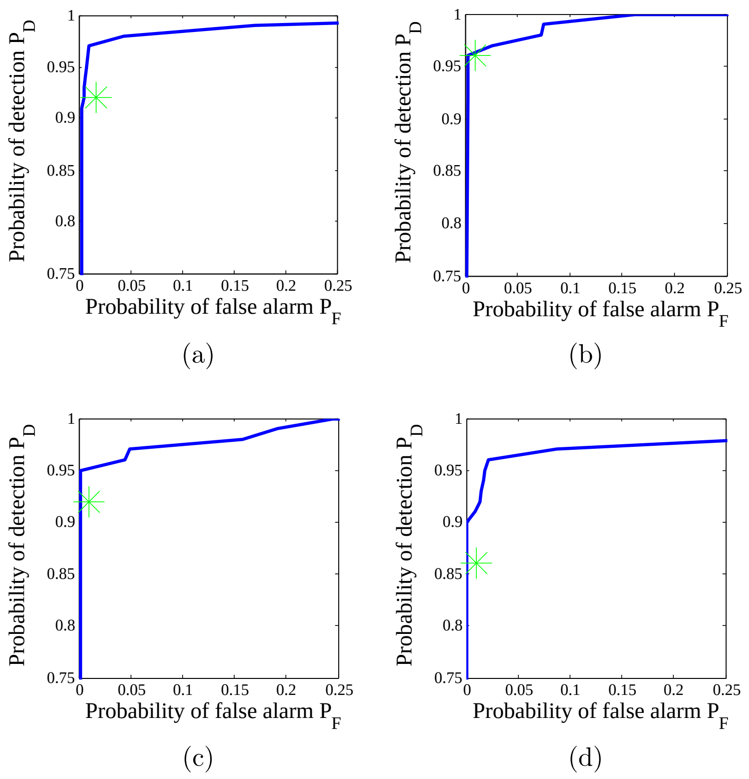

Figure 4: ROC curves for Piciarelli *et al.*'s synthetic dataset [17], varying the number of normal trajectory clusters from 7 to 10 (a-d, respectively). The experimental result reported by Piciarelli *et al.* is indicated by an asterisk. For visual clarity, the axes have been zoomed in.

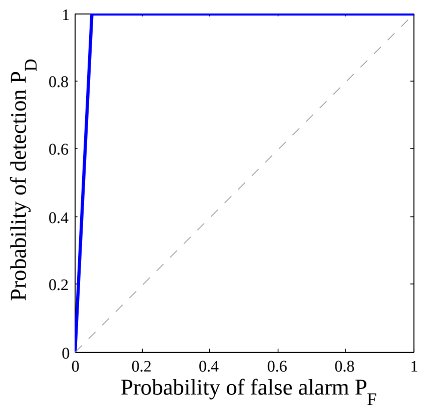

Figure 5: ROC curve for Naftel and Khalid's laboratory dataset [19].

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 [17]. 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. Firstly, 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. Secondly, the proposed method's goal-based approach allows for more intuitive natural language description of observed behaviours (as discussed later in Section 5). For example, an individual in the scene might be described as "walking from the main entrance towards the work station".

4.3. Dee and Hogg's carpark dataset [4, 5, 6]

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 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 [6]. Ground plane coordinates are used.

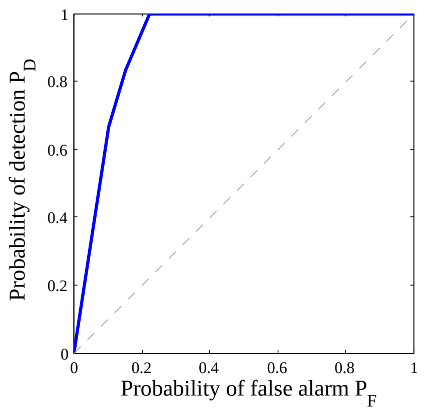

Figure 6: ROC curve for Dee and Hogg's carpark dataset [4, 5, 6].

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 [4, 5, 6], reducing the human involvement required to train the system. 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. On average, seven entry regions, nine turning regions, and eight exit regions are learned in the scene model.

Figure 6 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 [20]. Sillito and Fisher's work was described earlier in Section 2. The proposed method detects all unusual trajectories with 22% probability of false alarm, which is comparable to both Dee and Hogg's method [6] and Sillito and Fisher's method [20] (approximately 25% and 24%, respectively).

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 appear novel with respect to the rest of the carpark dataset. 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.

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 [11] to specialized human action recognition methods [37]. 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 [1]. A robust intelligent surveillance system will likely require the fusion of both trajectory and action analyses.

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 [3]. 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 [3]. A surveillance operator may need to retrieve all instances of individuals entering a restricted zone or leaving at a particular exit, for example.

Goal-based trajectory analysis has applications beyond intelligent surveillance. Liao *et al.*'s work [38, 39] 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.

Goal-based trajectory analysis can also potentially be applied in the evaluation of 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 tracking techniques has been proposed [40].

6. 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. The proposed method 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, training trajectories are assumed to correspond to normal behaviour. As future work, investigating methods to automatically detect and remove outlier trajectories from the training set will be valuable. The incorporation of stopping regions in the scene model can also be investigated: in many surveillance settings, constructing a map of normal or expected stopping regions can provide useful scene knowledge [2]. For instance, in an outdoor courtyard individuals may be expected to loiter at a fountain or on benches. 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.

Acknowledgements

We would like to thank Hannah Dee and Andrew Naftel for generously providing copies of their trajectory datasets. We 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 and by GEOIDE (Geomatics for Informed Decisions, a Network of Centres of Excellence).

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

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

[3] W. Hu, T. Tan, L. Wang, 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) (2004) 334–352.

[4] H. M. Dee, 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, Como, Italy, 2005, pp. 34–39.

[5] H. M. Dee, D. C. Hogg, Navigational strategies and surveillance, in: Proc. IEEE International Workshop on Visual Surveillance, Graz, Austria, 2006, pp. 73–81.

[6] H. M. Dee, D. C. Hogg, Navigational strategies in behaviour modelling, Artificial Intelligence 173 (2) (2009) 329–342.

[7] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Computing Surveys 38 (4).

[8] H. Zhong, J. Shi, M. Visontai, Detecting unusual events in video, in: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, Washington, DC, 2004, pp. 819–826.

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

[10] J. Owens, A. Hunter, Application of the self-organizing map to trajectory classification, in: Proc. IEEE International Workshop on Visual Surveillance, Dublin, Ireland, 2000, pp. 77–83.

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

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

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

[14] D. Makris, T. Ellis, Path detection in video surveillance, Image and Vision Computing 20 (2002) 895–903.

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

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

[17] C. Piciarelli, C. Micheloni, G. L. Foresti, Trajectory-based anomalous event detection, IEEE Transactions on Circuits and Systems for Video Technology 18 (11) (2008) 1544–1554.

[18] I. N. Junejo, O. Javed, M. Shah, Multi feature path modeling for video surveillance, in: Proc. International Conference on Pattern Recognition, Vol. 2, Cambridge, UK, 2004, pp. 716–719.

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

[20] R. R. Sillito, R. B. Fisher, Semi-supervised learning for anomalous trajectory detection, in: Proc. British Machine Vision Conference, Leeds, UK, 2008, pp. 1035–1044.

[21] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, Upper Saddle River, NJ, 2003.

[22] S. J. McKenna, H. Nait Charif, Summarising contextual activity and detecting unusual inactivity in a supportive home environment, Pattern Analysis and Applications 7 (4) (2005) 386–401.

[23] S. Ali, M. Shah, Floor fields for tracking in high density crowd scenes, in: Proc. European Conference on Computer Vision, Vol. 2, Marseille, France, 2008, pp. 1–14.

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

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

[26] X. C. He, N. H. C. Yung, Curvature scale space corner detector with adaptive threshold and dynamic region of support, in: Proc. International Conference on Pattern Recognition, Vol. 2, Cambridge, UK, 2004, pp. 791–794.

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

[28] S. S. Chen, 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, Vol. 2, Seattle, WA, 1998, pp. 645–648.

[29] T. Xiang, S. Gong, Spectral clustering with eigenvector selection, Pattern Recognition 41 (3) (2008) 1012–1029.

[30] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge, MA, 2006.

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

[32] M. Isard, A. Blake, A mixed-state CONDENSATION tracker with automatic model-switching, in: Proc. International Conference on Computer Vision, Bombay, India, 1998, pp. 107–112.

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

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

[35] M. Montemerlo, S. Thrun, D. Koller, 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, Acapulco, Mexico, 2003, pp. 1151–1156.

[36] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, John Wiley & Sons, New York, 2001.

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

[38] L. Liao, D. Fox, H. Kautz, Learning and inferring transportation routines, in: Proc. AAAI National Conference on Artificial Intelligence, San Jose, CA, 2004, pp. 348–353.

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

[40] W. Yan, D. A. Forsyth, Learning the behavior of users in a public space through video tracking, in: Proc. IEEE Workshop on Applications of Computer Vision, Breckenridge, CO, 2005, pp. 370–377.