Learning Semantic Scene Models From Observing Activity in Visual Surveillance

Dimitios Makris, *Member, IEEE,* and Tim Ellis, *Member, IEEE*

***Abstract*—This paper considers the problem of automatically learning an activity-based semantic scene model from a stream of video data. A scene model is proposed that labels regions according to an identifiable activity in each region, such as entry/exit zones, junctions, paths, and stop zones. We present several unsupervised methods that learn these scene elements and present results that show the efficiency of our approach. Finally, we describe how the models can be used to support the interpretation of moving objects in a visual surveillance environment.**

***Index Terms*—Motion analysis, site security monitoring, TV surveillance systems, unsupervised learning.**

I. Introduction

VIDEO surveillance has become a ubiquitous feature of the modern urban landscape, located in a wide variety of environments including shopping malls, railway stations, hospitals, government buildings and commercial premises. In some cases surveillance functions as a deterrent, discouraging unacceptable social behavior that can no longer be engaged in anonymously, recording and logging events for evidential purposes, or providing remote observation of sensitive locations where rigid access control is important.

These video monitoring systems typically deploy multiple video cameras, channeling the video signals to a central monitoring room, where multiplexing is used to display a subset of the images to security personnel. Event detection and recognition employ the perceptual capabilities of a human operator to observe (detect and identify) objects moving within the field-of-view (FOV) of the cameras and to infer their actions. However vigilant the operators, manual monitoring inevitably suffers from information overload, as a result of periods of operator inattention due to fatigue, distractions and interruptions. In practice, it is inevitable that a significant number of the video channels are not regularly monitored, and potentially important events are overlooked. In addition, fatigue increases dramatically as the number of cameras in the system is increased. Automating all or part of this process would obviously provide significant benefits, ranging from a capability to alert an operator to potential events of interest, through to a fully automatic detection and analysis system. However, the reliability of automated detection systems is a paramount issue, since frequent false alarms induce skepticism in the operators, who quickly learn to ignore the system.

Automation requires the development of sophisticated computer vision algorithms to detect, locate and follow a target (generally either a pedestrian or a vehicle) as it moves through the environment. Simple video motion detection (VMD) has been used to provide a means of alerting operators, typically based on manually located regions of interest, which are triggered through some perceived change in the image data. While they are widely deployed in commercially available surveillance systems, VMDs have many drawbacks. They can be falsely triggered by nonmotion events such as those due to weather-related changes or variations in the illumination; they are unable to determine the context of a particular event, e.g., a valid individual entering a restricted area or a merely boisterous group enjoying an evening out.

More advanced motion detectors attempt to track moving objects throughout the camera FOV, maintaining tracking through a variety of noise-related interference. Much of previous research has focused on determining the trajectory of objects in single camera views [1], [2]. More recently, effort has focused on robustly tracking targets through complex environments viewed by multiple cameras [3], addressing the problems of combining corresponding information where the FOVs are overlapped.

However, it is desirable that visual surveillance systems not only can detect and track objects, but also understand the activity of the scene, ideally in a way that is consistent with that of a human observer. The task of automating the interpretation of the video data is a complex one and can depend on a wide range of factors, including location, context, time, and date. This information relates to where objects are and what they may be doing as they are observed, and attempting to characterize typical behaviors. Such information can be used to enrich the raw trajectory data, and provide higher level descriptions of activity.

Our aim is to provide visual surveillance systems with an activity-based semantic scene model that supports high-level understanding and analysis of the observed activity. An unsupervised approach is preferred because it allows automatic configuration of the visual surveillance systems. This concept of learning from the data is particularly appropriate for video surveillance, since it is possible to allow the system to construct the models simply by observing activity over long periods of time.

The scene structure influences, directly or indirectly, the way that targets act. Therefore, specific types of events may be associated with specific regions. For instance, roads constrain vehicles to move along specific lanes in a particular direction; gates and doors are related to entrance/exit events where targets appear or disappear; bus stops indicate where people should wait for the bus. The activity-based semantics learnt in our framework aims at recognizing generic activities. For instance, our

Manuscript received August 11, 2003; revised August 31, 2004. This work was supported in part by the EPSRC under Grant GR/M58030. This paper was recommended by Guest Editor B. Draper.

The authors are with Kingston University, Kingston-upon-Thames, Surrey KT1 2EE, U.K. (e-mail: d.makris@kingston.ac.uk; t.ellis@kingston.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2005.846652

method does not rely on distinguishing between a pedestrian pavement and a traffic lane, but it describes both as "paths," i.e., areas within which targets move.

The input to our algorithms is motion-tracking data, derived from an online surveillance system we have been developing [2]–[5]. Using trajectories extracted over long periods of observation (hours/days), we identify regions or zones in the camera FOV where objects appear and disappear, paths that they follow through the scene, junctions where they might change their route and stopping areas, where they might wait for a friend or queue for a bus. Learning and encoding these regions allows subsequent trajectories to be classified online. A track database is used to manage and record the results of tracking and the annotations associated with the semantic activity models [5].

The paper is structured according to the following: Section II reviews previous research related to activity analysis. In Section III we briefly describe the algorithms that extract the motion-tracking data from video data. In Section IV, the scene model is described. In Sections V–VII we introduce the algorithms that are used to construct the models of each of the scene elements and in Section VIII we evaluate the operation of these algorithms. In Section IX, we describe how these models can then be utilized in a video surveillance context. Conclusions are outlined in Section X.

II. Previous Work

Extracting semantics from images has long attracted the attention of computer vision researchers. The VISIONS system by Hanson and Riseman [6] aimed to segment static images into semantically consistent regions, according to their pixel values. In the visual surveillance domain, Neumann [7] and Mohnhaupt and Neumann [8] tried to learn paths by accumulating speed and orientation information in a spatio-temporal buffer.

Howard and Buxton [9] identified the lack of an explicit semantic model of the scene and they proposed a spatial, polygon-based model. Howard [10] used these models to segment new scenes manually.

Fernyhough *et al.* [11] used the same model as a basis for a learning algorithm that can automatically determine object paths by accumulating the trace of tracked blobs. The benefits of his method are that it is unsupervised and auto-initialized. However, it requires full trajectories, it cannot handle occlusions and the results are dependent on the shape and size of the blobs, as they appear on the image plane.

Johnson and Hogg [12] proposed a method for learning behavior models using a vector quantization method to learn typical routes taken by pedestrians from representative trajectories. However, no high-level semantic information is derived and their method requires the knowledge of entry/exit areas of the scene, which are defined manually.

Koller-Meier and Van Gool [13] accumulate observed trajectories to produce average descriptions of similar trajectories, which then can be used for atypical activity detection.

Grimson *et al.* [14] describe activity in a six-dimensional (6-D) space (position, velocity, size, aspect ratio). They use two different methods to classify activities: a) clustering of observations in the 6-D space to Gaussian models by using the Numeric Iterative Hierarchical Cluster (NIHC); and b) accumulating co-occurrence statistics in a quantised 6-D space. In their former method, they retrieve activity-related areas by backprojecting Gaussian clusters with specific characteristics on the spatial (x, y) plane.

Stauffer [15] accumulated observations to reconstruct a rough depth map of the scene, using the line-of-sight constraint.

In our earlier work, we proposed path spatial modeling and learning in [16], [17]. Our path model was enriched by probabilistic information and Hidden Markov Model theory was used to analyze the pedestrian activity [18]. In [19], entry/exit zones modeling and learning is presented. More recently, Stauffer [20] has proposed a similar technique to estimate entry/exit zones.

A recent paper by Buxton [21] summarizes a number of models and methods that have been used to represent activity in the visual surveillance domain.

III. Motion Tracking

The first steps in motion tracking require the separation of objects of interest from the background. When the video data source is a static camera, background subtraction provides an efficient and sensitive method for detection. To operate reliably in an outdoor environment, with the vagaries of illumination change and weather variations, it is essential to employ a robust method for background adaptation. We use a sequence of video frames to define an adaptive pixel-wise model for the background based on a Gaussian mixture model for each pixel, in either the intensity, RGB [22] or normalized rgb space [4].

At each frame, pixels are classified as either foreground or background, according to the most likely Gaussian model. A connectivity algorithm is then applied to identify possible objects in motion. Undersegmentation (e.g., due to low contrast) can cause the detected objects to fragment, so morphological operations can be applied to improve the connectivity of the foreground objects. A detected foreground object is known as blob and is characterized by its boundary, position (centroid) and mean color. Velocity information is added during the matching process. Fig. 1 summarizes the motion detection process.

At each frame, new blobs are matched to blobs detected in previous frames to describe a tracked object (also referred to as a target). The matching process is assisted by a Kalman filter[1] which models the position, the velocity and the size of each blob [2]. Unmatched blobs may indicate new objects that have just appeared in the view. Previously detected blobs that do not match new blobs may indicate objects exiting the scene, or objects in occlusion (i.e., hidden from view).

The output of the motion tracking process is a set of trajectories. A trajectory aims to describe the location history of a target moving through the scene. Information for the position, velocity, size and color of the corresponding blob is held for each frame over which the object was tracked.

Unfortunately, motion detection and tracking in a cluttered environment experience many problems due to a variety of reasons: Illumination changes (local-global, slow-fast), static occlusions, self-occlusions, and "semistationary" motion may re-

[1]Strictly speaking, the targets' motion cannot be assumed linear due to the perspective effect of the camera and Kalman filter is not appropriate. However, the performance of the Kalman filter is satisfactory because of the high frame rate that allows us to assume that motion is linear for short time intervals.

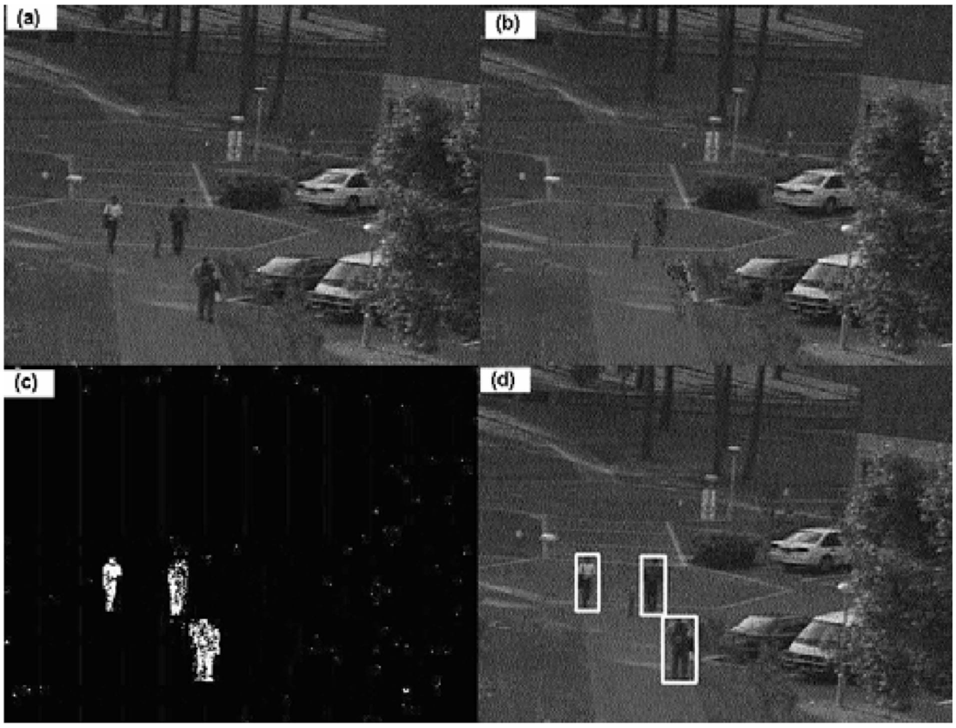

Fig. 1. Motion detection. (a) Original frame. (b) Background model. (c) Foreground pixels. (d) Detected objects.

sult in either falsely detected blobs, or undetected objects. Additionally, the presence of multiple targets may cause errors to the blob matching process. We can identify different types of system noise that are manifested as incomplete trajectories, false trajectories and trajectories corresponding to apparent motion (e.g., associated with moving vegetation, curtains, computer screens, reflections on windows and other surfaces, background motion) that are of no direct interest. Many techniques have been developed to improve the reliability of motion detection and tracking (see, for example, [2]). However, because a surveillance system is expected to operate for extended periods of time, under very different conditions, it seems that there is no guarantee for perfect tracking.

We can visualize a trajectory on the image plane as a sequence of points, where the points indicate the positions of the object centroid (Fig. 2). Alternatively, the object positions can be converted from image coordinates to ground plane coordinates by using geometric camera models and the ground plane constraint. In this case, trajectories are visualized on a ground plane map. Ground-plane trajectories have the advantage that they are not affected by the viewpoint and the perspective of the camera.

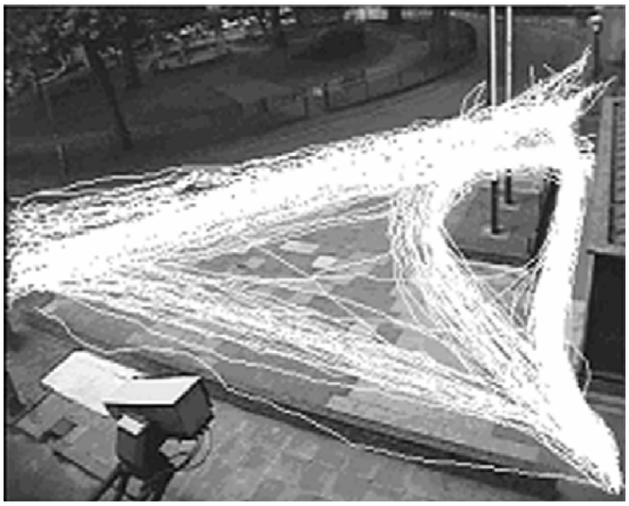

Fig. 2. Set of 752 trajectories for a particular scene visualized on the image plane.

IV. Scene Model

In many surveillance applications (see for example [23]), activity analysis is based on a manual segmentation of the scene. However, this implies that each surveillance system must be specially configured to allow event analysis. In multicamera systems, configuration of every camera of the network would be a tedious task and must be repeated each time that either the view of the camera changes (because the camera has been moved) or the scene itself has been changed. Therefore, automatic, unsupervised learning of a semantic model of the scene is desirable.

We require a scene model that can represent the spatial nature of the scene, so that events can be localized with respect to actual scene features (e.g., doorways, seats). Additionally, the model must have a probabilistic nature, so it can support a probabilistic framework for the activity analysis.

We introduce an activity-based semantic scene model that consists of the following primitive elements (examples are taken from Figs. 3 and 4):

1) entry/exit zones, e.g., A, C, E, G, H;
2) junctions, e.g., B, D, F;
3) paths, e.g., AB, CB, BD, DE, FH;
4) routes, e.g., ABDFH, EDFG, CBDE;
5) stop zones, e.g., I, J.

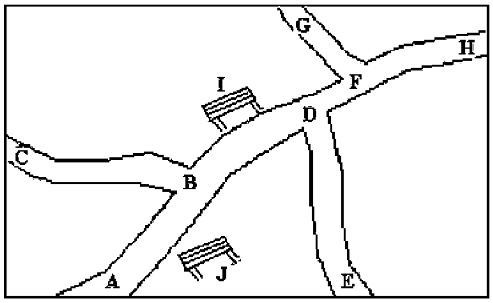

Fig. 3. Topographical representation of the scene model.

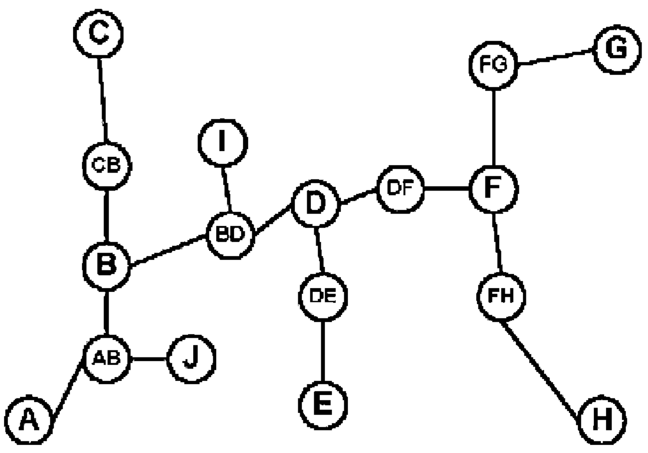

Fig. 4. Topological representation of the scene model.

We use two levels of representation for the scene level:

1) a topographical representation that shows the spatial extent and the location of the scene elements (see Fig. 3);
2) a topological representation that shows the scene elements as nodes of a network (see Fig. 4). The topographical representation focuses on the spatial nature of the model, whereas the topological representation is associated with the probabilistic nature of the model which can be used as the basis of a Bayesian belief network (BBN).

The structure of the scene affects the behavior of the targets around the scene. Therefore, in our approach, the scene is learnt by observing typical patterns of activity. In the following sections, we describe how the elements of the scene model are learnt using an unsupervised approach from observed trajectories extracted by the motion-tracking algorithm.

V. Entry/Exit Zones

A. Modeling

Entry/exit zones represent areas of the scene where targets normally appear and disappear. They can be distinguished as either scene-related features, such as doors and gates, or they can be view-related, for instance, occurring at the borders of the camera FOV. We distinguish between these different types of entry/exit zones because in multicamera systems, the scene-related zones remain fixed with respect to the environment, while the view-related zones may change according to the camera location. Entry and exit zones may be spatially coincident, e.g., in most pedestrian environments where paths are bidirectional, or sepatated, such as in road traffic environments, where vehicles are constrained to drive on one side of the road (i.e., either on the left or right).

We choose to encode both the spatial and probabilistic characteristics of the zones using two-dimensional (2-D) Gaussian mixture models (GMMs), which provides a more compact representation than say, a simpler, purely spatial model using polygons.

B. Learning

Our learning method is multistep and uses the expectation-maximization (EM) algorithm [24]. An entry-point dataset consists of the start points of each trajectory, as derived by the motion-tracking algorithm. Similarly, the end points of a trajectory form the exit-point set.

One of the benefits of EM is that it can successfully model overlapped distributions. Therefore, if noise is overlapped onto signal, then it is possible to generate separate clusters for the signal and the noise, if they have different statistics. Two types of noise are recognized in the dataset.

1) Tracking failure noise: Tracking failure noise is due to the failure of the motion-tracking algorithm to track a target successfully for its whole activity in the scene. It may appear in the form of false trajectories (trajectories where the motion history of more than one target have been mixed), or split trajectories (trajectories that represent only a portion of the motion history of a target). In the entry-point/exit-point datasets, tracking failure noise appears as false positive points, distributed over all the activity areas. In general, the greater the activity in an area, the more false positive points will be generated.
2) Semistationary motion noise: If sources of semistationary motion noise are present (such as trees, curtains, window reflections), then an apparent high activity is detected in the vicinity of the source of the semistationary motion noise. The characteristic of this apparent activity is that the trajectories start and end in the same local region. In the entry-point/exit-point datasets, semistationary motion noise appears as a dense distribution of false positive points.

We model both types of noise, as they appear in the entry-point/exit-point datasets, using Gaussian distributions. The tracking failure noise is represented by wide Gaussian distributions over the activity areas, whereas the semistationary motion noise is usually represented by narrow Gaussian distributions at the noise sources. Because of the different nature of the two types of noise, different methods are used to filter them out.

We developed a multistep learning algorithm that first discards the semistationary noise and then the tracking failure noise. Since we have no information on the actual number of entry/exit zones in the scene, we overestimate the number of entry/exit zones in the scene then estimate the number of the

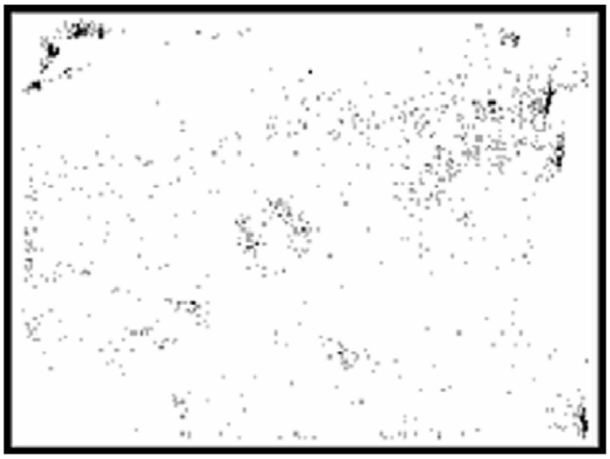
Fig. 5. Entry-point dataset (4250 samples) with both types of noise present.

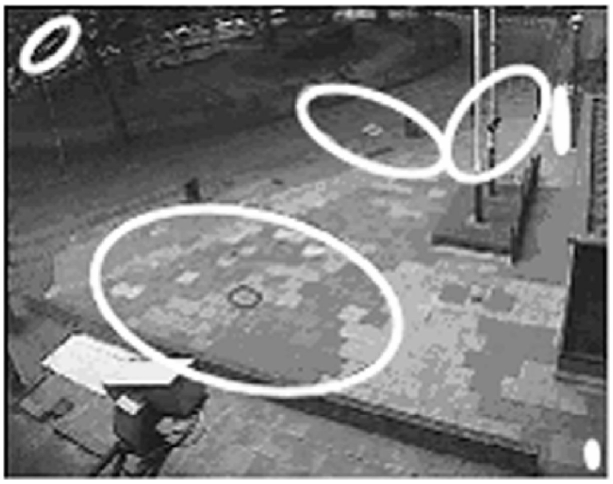
Fig. 6. GMM derived by first run of EM ($N = 6$). The upper left cluster is caused by the semistationary of the tree branches.

signal clusters by eliminating the noisy ones. We describe the algorithm for the entry-point dataset; the same method is used for the exit-point dataset.

1) The EM algorithm with model order N is applied to the entry-point dataset $\mathbf{E}$ (Fig. 5) and a GMM is derived (Fig. 6). High-density Gaussian clusters correspond to either entry zones or semistationary motion noise. Low-density clusters correspond to tracking failure noise.
2) All the trajectories are checked if they are semistationary, according to the GMM extracted in the previous step. If so, they are discarded from the dataset and a new cleaned entry-point dataset $\mathbf{E}'$ (Fig. 7) is created.
3) The EM algorithm with model order N' is applied to the clean entry-point data-set E' (Fig. 8).
4) Gaussian clusters are classified as either signal clusters or noise clusters, according to a density criterion (Fig. 9). More specifically, if w_i is the prior probability of a cluster i and $\mathbf{\Sigma_i}$ is its covariance matrix, where

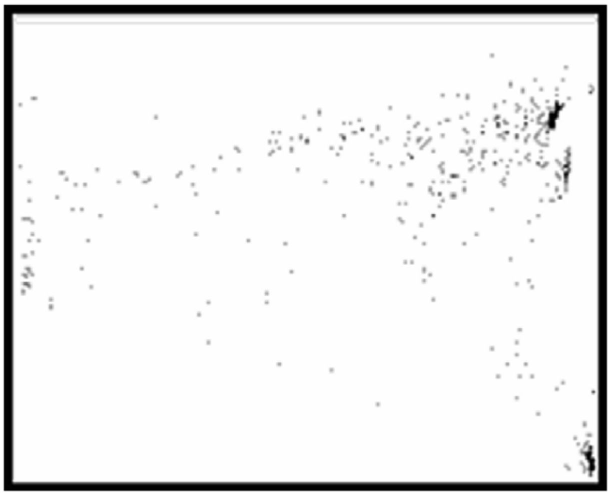
Fig. 7. Clean entry-point dataset (1767 samples) after discarding stationary motion noise.

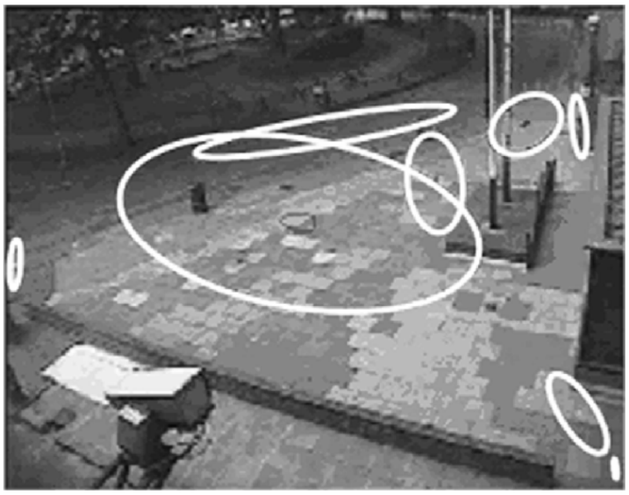
Fig. 8. GMM after second run of EM ($N' = 8$).

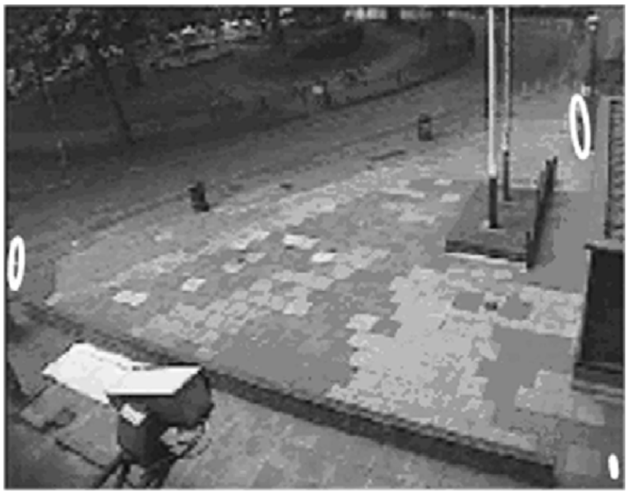
Fig. 9. Three entry zones derived by the multistep algorithm.

TABLE I
CLUSTERS OF FIG. 8

Cluster id	w_i (%)	d_i (10^{-6} pixels^{-2})	d_i/T	Signal?
1	10.2	2	0.03	No
2	13.2	1622	21.41	Yes
3	19.0	61	0.80	No
4	4.3	134	1.77	Yes
5	11.7	23	0.31	No
6	3.3	13	0.17	No
7	11.2	24	0.32	No
8	27.0	417	5.51	Yes

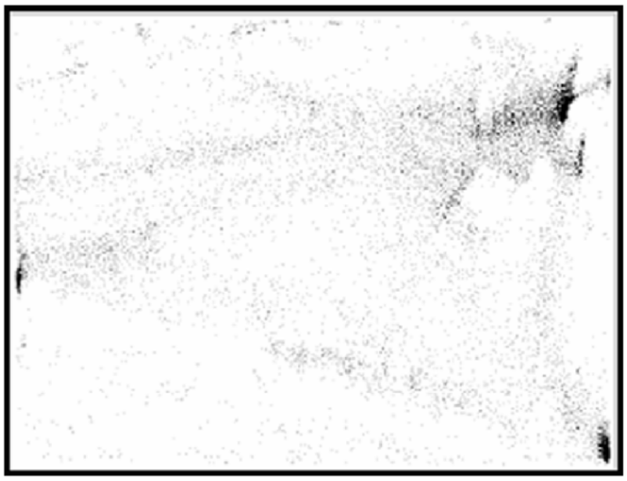
Fig. 10. Entry-point dataset (13223 samples).

$i = 1 \ldots N'$, then a measure of the density d_i is given by

$$d_i = \frac{w_i}{\pi \cdot \sqrt{||\Sigma_i||}}. \tag{1}$$

A threshold value T is defined by the clean entry-point dataset $\mathbf{EW}'$ and the model order $\mathbf{N}'$

$$T = \frac{\alpha}{\pi \cdot \sqrt{|\Sigma|}} \tag{2}$$

where α is a user-defined weight $(0\langle\alpha\langle 1)$, (which we have experimentally set to 0.1) and $\mathbf{\Sigma}$ is the covariance matrix of the dataset $\mathbf{E}'$.

The selection of the threshold T is justified as follows: $\pi \cdot \sqrt{|\Sigma|}$ is a measure of the area over which dataset $\mathbf{E}'$ is distributed. Therefore, $\pi \cdot \sqrt{|\Sigma|}/N'$ is a measure of the area of an "average" cluster. If a/N' indicates an acceptable popularity for a signal "average" cluster, then T indicates the density of an "average" signal cluster. Table I lists the densities d_i of the clusters of Fig. 8 and separate them to signal and noise clusters according to the threshold $T = 7.87 \times 10^{-6}$ pixels^{-2}, as estimated by (2).

Our method is relatively insensitive to the model order selection at steps 1) and 2). N is selected to ensure that any possible motion source noise is not multimodeled. Similarly, N' must be large enough to ensure all the zones can be modeled, while not over-modeling the data. Therefore, if N_e is the number of real entry zones, N_{sm} is the number of the semistationary motion sources and we allow $N_{\mathtt{tf}}$ (usually 1–5) clusters to model the tracking failure noise, then $N = N_e + N_{\mathrm{sm}} + N_{\mathtt{tf}}$ and $N' = N_e + N_{\mathtt{tf}}$. The values of N, N' are quite flexible, thanks to the fact that in most cases, extra clusters tend to model tracking failure noise.

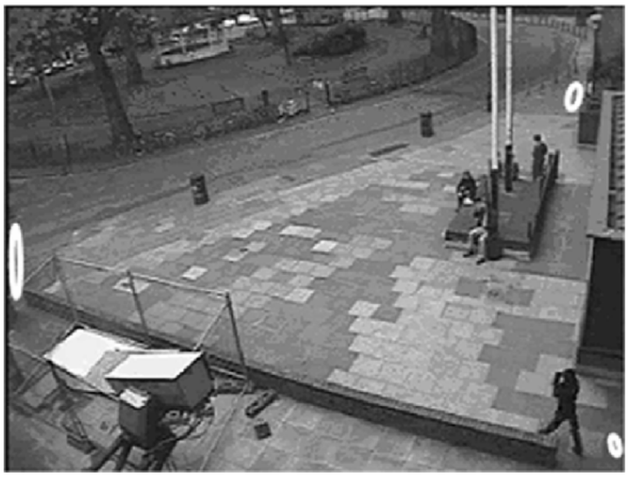
Fig. 11. Three detected entry zones.

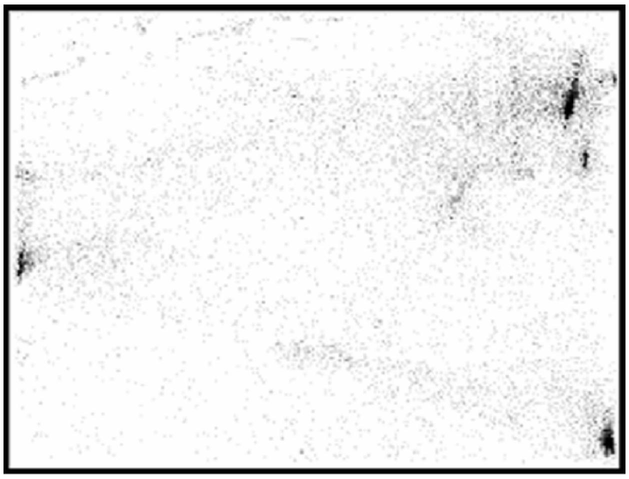
Fig. 12. Exit-point dataset (13223 samples).

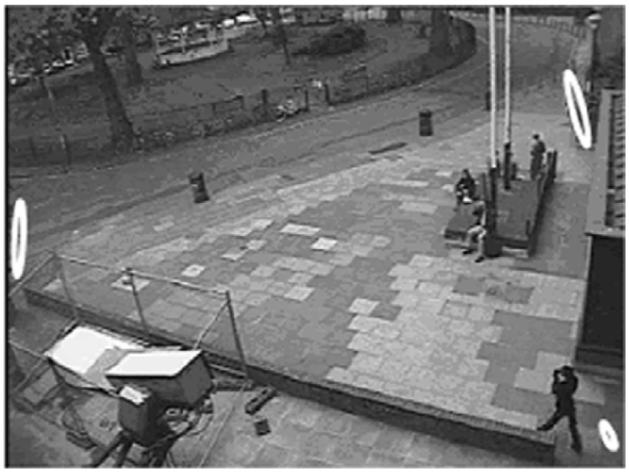
Fig. 13. Three detected exit zones.

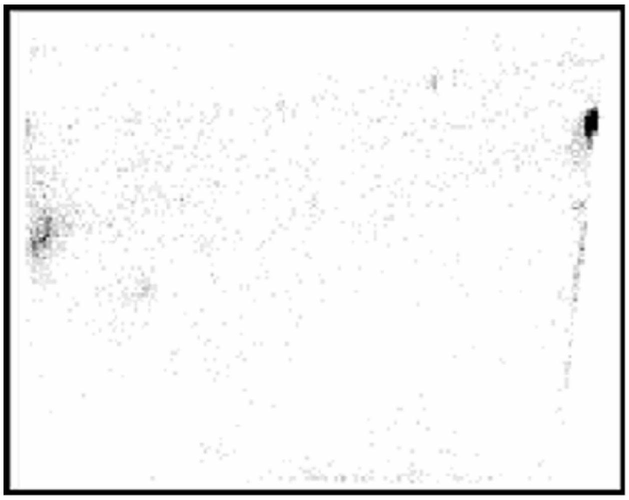
Fig. 14. Entry-point (12746 samples) dataset in a road traffic environment.

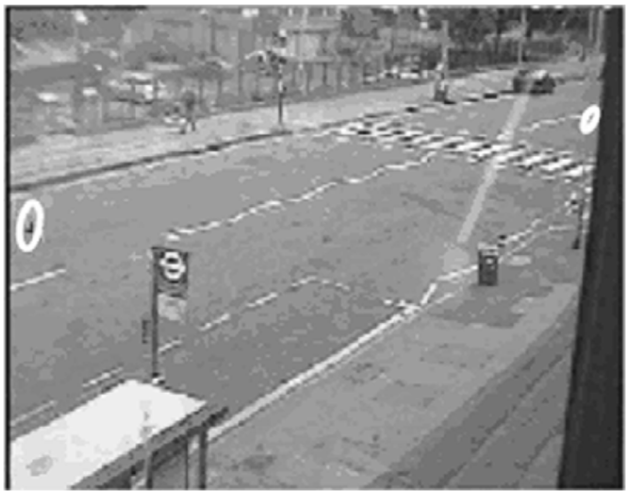
Fig. 15. Two detected entry zones in a road traffic environment.

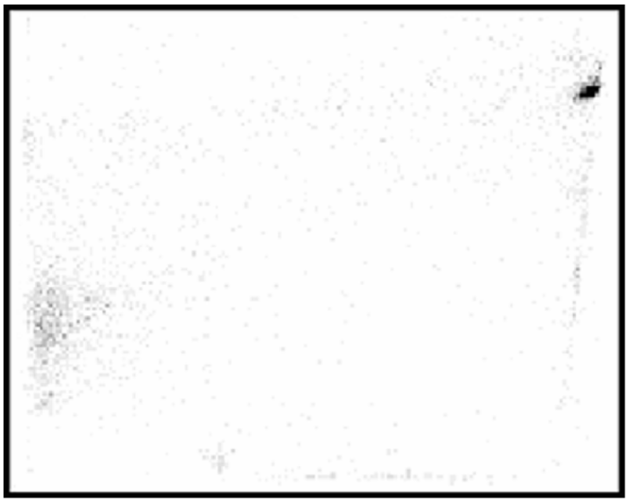
Fig. 16. Exit-point dataset (12746 samples) in a road traffic environment.

The algorithm has been tested on data extracted from two outdoor scenes, containing both pedestrian and vehicle trajectory

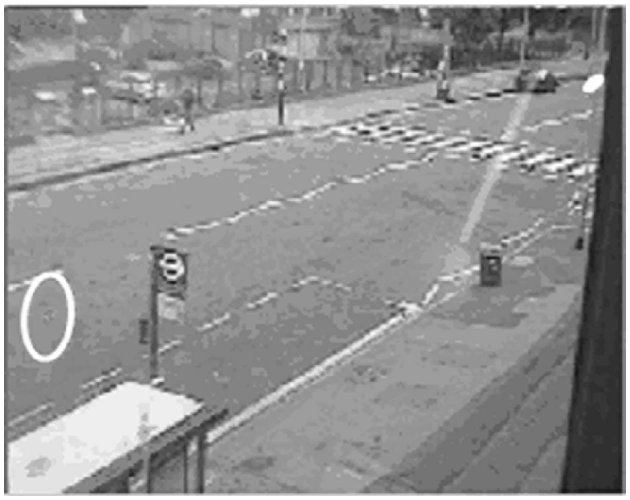
Fig. 17. Two detected exit zones in a road traffic environment.

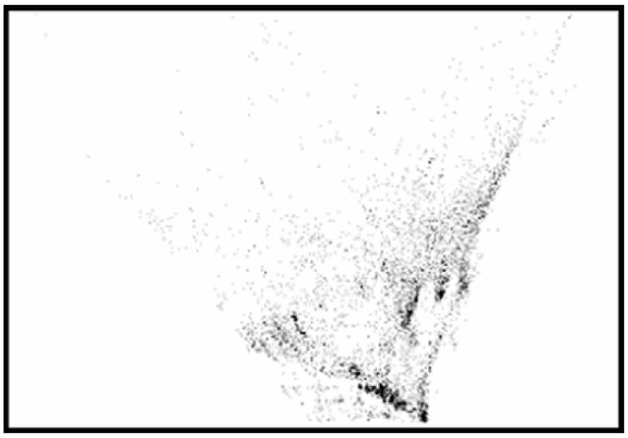
Fig. 18. Stop events (9455) on the ground plane.

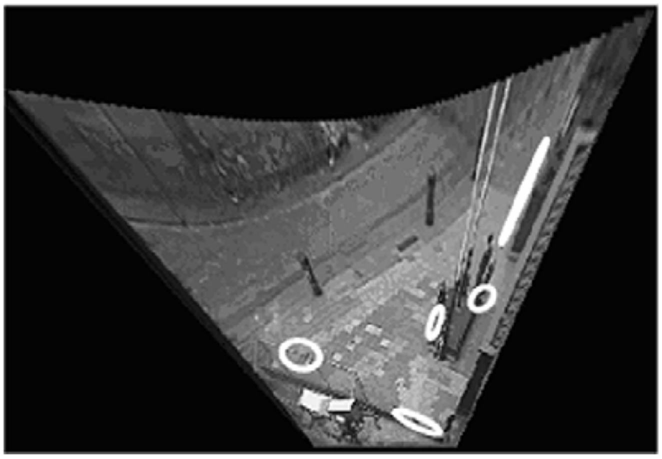
Fig. 19. Five stop zones as derived by the EM algorithm.

data. For example, Figs. 10 and 12 show entry/exit point datasets for the scene of Fig. 2 and the extracted entry/exit zones are illustrated in Figs. 11 and 13. Similarly, Figs. 14 and 16 show entry/exit point datasets for a road scene (note that "driving on the left" regulations are applied to the specific scene) and Figs. 15 and 17 show the extracted entry/exit zones.

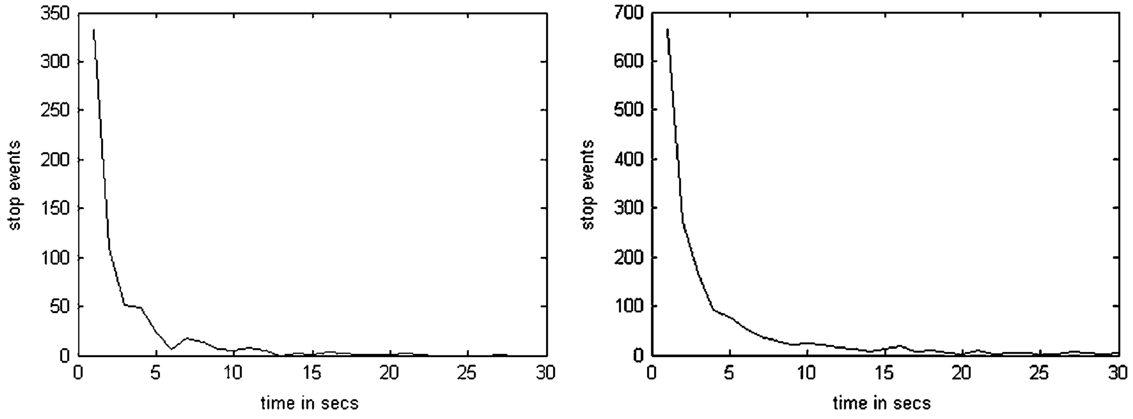

Fig. 20. Stop event duration diagrams. The x axis indicates the duration of a stop event and the y axis the number of detected stop events with the specific duration. Results are given for the two most popular stop zones.

VI. Stop Zones

Stop zones are areas where the targets appear stationary for some time. Pedestrians are stationary when they stop to sit, rest, queue, wait to access a resource, merely observe the scene or just wonder around. Although the majority of research in video surveillance has focused on detecting and tracking motion, it is when objects stop and interact with some fixed element of the scene that the system is more likely to be interested in them, for instance at a bus stop, an ATM machine, a park seat, or a shop window.

We detect a stop event when a targets speed is lower than a predefined threshold. A stop-event dataset is formed by checking the target trajectories for stop events. We form the stop-event dataset, using ground plane coordinates (see Fig. 18) because apparent speed may be strongly affected by the perspective of the camera viewpoint; therefore, the image coordinate system is less reliable.

As with the entry/exit zones, we use a GMM to model the spatio-probabilistic characteristics of the stop zones and EM to learn them (Fig. 19). However, stop zones have an additional property that we want to model: the duration of the stop events. We want to know for how long an object may be stopped. From our experiments (see Fig. 20), it seems that stop event duration can be adequately approximated by an exponential function; therefore, we can cheaply model the stop event duration of each stop zone by adding an additional parameter.

VII. Routes

A. Modeling

In road traffic environments, vehicles must follow specific predefined routes. In pedestrian environments, people normally walk on well-prescribed pathways. Even in cases where no predefined routes exist, the structure of the scene affects the behavior of pedestrians and route-patterns of activity can be estimated.

A model was required to represent routes. We differentiate between paths and routes in order to represent where tracked objects may change their predicted destinations as they approach a junction. This model must represent the physical extent of the route and its usage by the targets. More specifically, the model

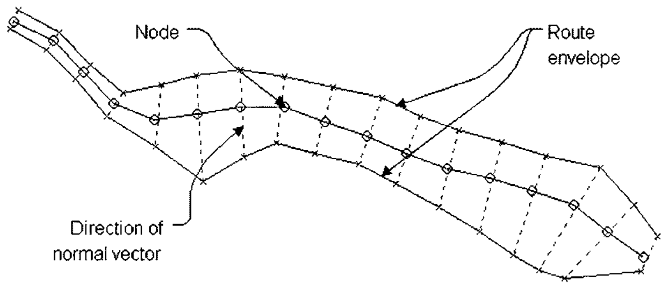

Fig. 21. Depiction of the route model.

represents the main axis of the route, its borders (or envelope) along the route and its usage along and across the route.

The model that we propose is depicted in Fig. 21. It uses a spline-like representation and consists of a sequence of equidistant nodes, where each node is characterized by:

1) a 2-D position vector $\vec{\mathbf{x}}_i = [x_i, y_i]$, which is part of the main axis;
2) a weight factor w_i that shows the usage of the node;
3) a normal vector $\hat{\mathbf{n}}_i = [nx_{i,}ny_i]$ defined as the unit vector perpendicular to the local spline direction, as defined by the sequence of the nodes;
4) two bound 2-D points along the normal vector line, the left boundary $\vec{\mathbf{l}}_i = [lx_i, ly_i]$ and the right boundary $\vec{\mathbf{r}}_i = [rx_i, ry_i]$. The two bound points encode the width of the route at the specific position. They can be set according to either the extremes of the matching trajectories, or a Gaussian distribution around the node position;
5) a probability density function (pdf) $g_i(x)$ of the usage of the node, across the route, where x is the signed distance from the node position. $g_i(x)$ could be represented as a set of samples. However, as results indicate (Fig. 22), $g_i(x)$ can be adequately approximated by a Gaussian distribution.

B. Learning

We have developed an unsupervised algorithm for learning a set of route models from trajectories. The algorithm has two parameters: a) a resample factor r of the route model, which

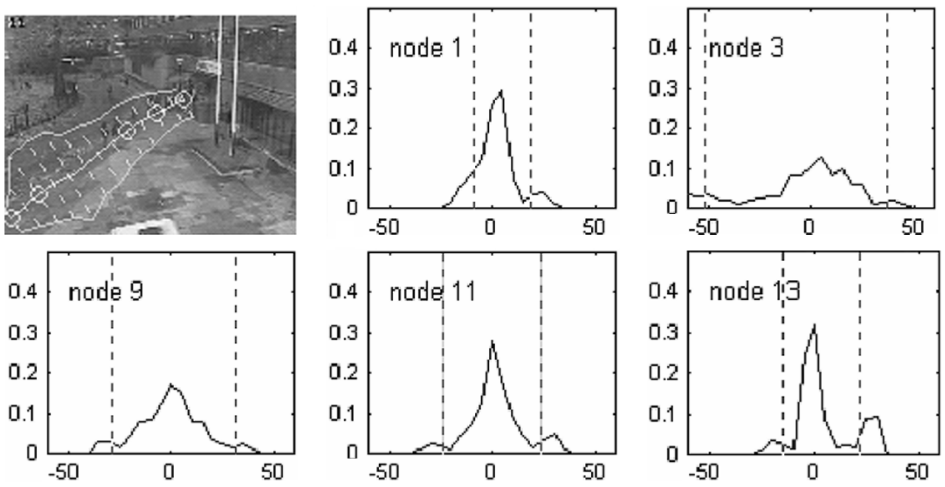

Fig. 22. Sample-based pdfs of the usage across the route, for some selected nodes. The dotted vertical lines indicate the location of the route borders at the node.

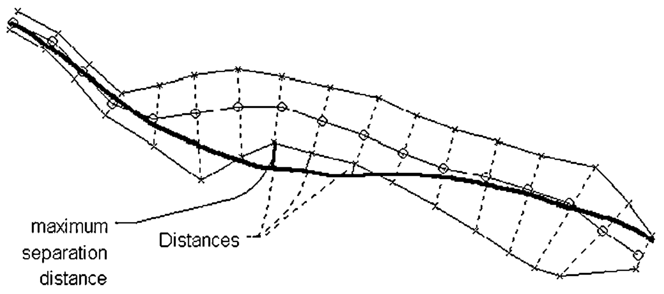

Fig. 23. Matched trajectory. The MaxSD of the trajectory from the route model is smaller than the MinAD of the algorithm.

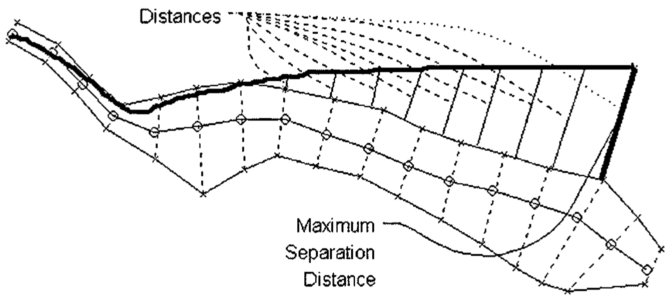

Fig. 24. Unmatched trajectory. The MaxSD is too large.

defines the distance between two consecutive nodes of a route and b) the minimum allowed distance (MinAD) $d_{\min}$ between two routes.

The resample factor r defines the spatial resolution of the route model, whereas MinAD defines when two route models should be merged into one. MinAD does not define model order explicitly, but implicitly.

A summary of the learning algorithm is as follows:

1) First trajectory of the dataset initializes the first route model
2) Each new trajectory is compared with the existing route models.
   a) If a trajectory matches a route model (Fig. 23), then the route model is updated.
   b) If a trajectory does not match to any route model (Fig. 24), a new model is initialized.
3) The updated route model is resampled, so internode distances are kept equal to r.
   a) Each updated route model is compared with the other route models.
   b) If two route models are sufficiently overlapped, they are merged.

A trajectory matches a route model according to the following criteria: a) if the trajectory is within the route model envelope; and b) the maximum separation distance (MaxSD) between the trajectory and the route model envelope is smaller than MinAD.

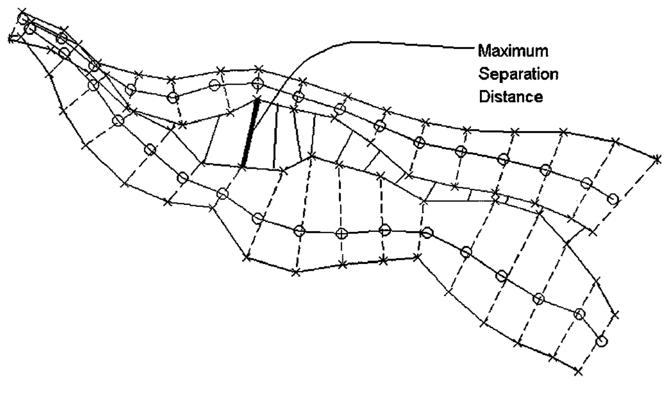

Fig. 25. Case for merging a pair of route models. The MaxSD between the two routes is smaller that the MinAD.

To update a route model, a sample $\vec{\mathbf{p}}_i$ of the matching trajectory for each model node i is derived along the direction of $\hat{n}_i$. The node characteristics are updated cumulatively according to the sample $\vec{\mathbf{p}}_i$ and the weight factor is increased by one. Similarly, two route models are merged if the MaxSD of their envelopes is smaller than the MinAD (Fig. 25). When the route is resampled, all the nodes characteristic values are re-estimated using linear or bicubic interpolation of the old values.

Because the algorithm was developed to deal with data that may contain a considerable amount of tracking failure noise, it is possible for a trajectory to match a shorter route (Fig. 26).

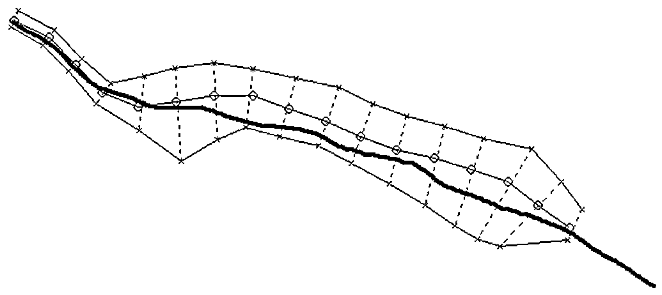
Fig. 26. Case for extension. The route model is shorter than the matching trajectory.

TABLE II
AVERAGE LEARNING TIME PER TRAJECTORY WITH RESPECT TO THE RESAMPLE FACTOR (r)

r (pixels)	20	30	40	50	60
Time (secs)	8.46	4.85	3.85	2.83	2.55

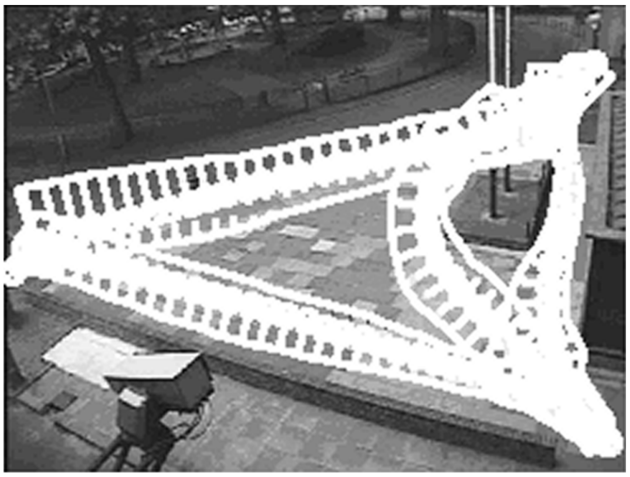
Fig. 27. Route models learnt from 24 h of trajectory data (752 samples).

In this case, when updating the route model, it is extended according to the trajectory. However, if entry/exit zones are known and tracking failure noise is discarded from the dataset, then trajectories are restricted not to match shorter routes and no route extension step is required.

Table II shows the average learning time per trajectory for different values of r. The route learning algorithm was implemented in Matlab and running on a Pentium 3–550 Mhz Linux system with 256 MB RAM. The current implementation can serve real-time learning of routes with rates of 0.2–0.4 trajectories per second. Implementation of the algorithm in C and deployment in faster processing units would allow real-time learning of routes, even for high-activity scenes, subject to satisfactory extraction of data from the motion tracking component.

The route-learning algorithm can be used in either image plane coordinates (Fig. 27) or ground plane coordinates (Fig. 28). Working in ground plane coordinates may be beneficial, because results are not distorted by the perspective effect and models from different camera views can be integrated.

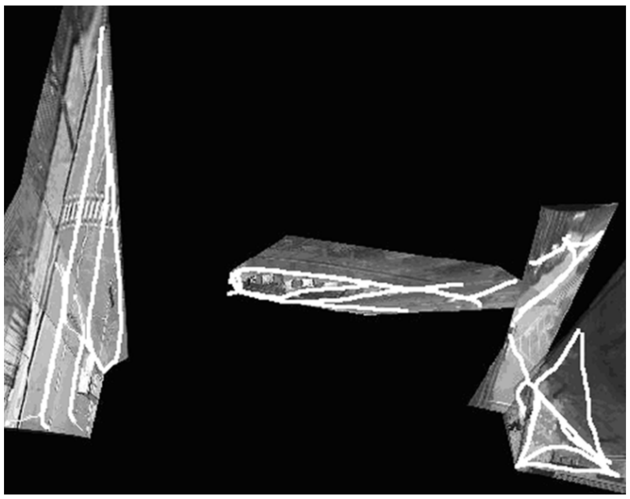
Fig. 28. Route models learned from ground plane trajectory data, derived by six different cameras.

TABLE III
SUCCESS RATE OF IDENTIFYING ENTRY ZONES FROM NOISE

N'	3	4	5	6	7	8	9	10
%	100	100	100	100	100	100	80	60
N'	11	12	13	14	15	16	Overall	
%	80	80	80	60	100	40	85.7	

VIII. EVALUATION

We ran a set of 70 experiments to test the algorithm that detects the entry/exit zones, based on the dataset shown in Fig. 10. Specifically, the algorithm was tested for different values of $N' = 3, \ldots, 16$ and for each value, it was run five times with different initialization data. The output Gaussian ellipses were manually labeled as "signal" and "noise" and these labels were compared to the ones that our method automatically provides. The outcome of the automatic method in each experiment was characterized successful only when all the labels were coincident with the manual ones. Table III illustrates the success rate for different values of N' and the overall success rate which is 85.7%.

We ran the algorithm for a variety of scenes and datasets and in most cases the results are successful. The algorithm fails, if $N' <= N_e$. If $N' > 2 \times N_e$, then some of the entry/exit zones may be represented as multimodal. The algorithm fails to discard the stationary motion noise if its source is coincident to a real entry zone, or if its source is multimodal.

The dataset of Fig. 2 was used to evaluate the route-learning algorithm. The algorithm was run for different values of r and MinAD. Because the algorithm is incremental, we re-order the trajectory dataset to test against any dependency on the dataset order. In the majority of the experiments, the output of the route learning algorithm was consistent with a semantic interpretation of the routes given manually. However, if the MinAD becomes low (e.g., $\mathrm{MinAD} = 10$), the algorithm tends to produce a large number of narrow routes.

For further validation, a test dataset of 100 previous unseen trajectories was labeled both manually and using the route matching algorithm and a specific set of routes, which is pictured in Fig. 27. The results were compared and our automatic method classified correctly 97 out of 100 trajectories.

IX. Applications

The scene model we have presented can have many applications in the area of the visual surveillance ([5], [16]–[19]) and we provide a brief summary in this section.

Knowledge of a semantic scene model allows a high-level description of the observed activity. A hierarchical database has been designed that contains three levels of information: a) low-level video data; b) middle-level trajectory data; and c) high-level conceptual description of the activity using terms derived from the scene model. High-level description of the activity is beneficial for a surveillance database: it provides compact representation of the data, allows processing of conceptual queries and can respond to queries much faster than to queries based on raw trajectory data.[2]

Elements of the scene model are used to enhance the motion tracking process. For example, entry/exit zones allow the motion tracker to localize where objects should be initialized and terminated respectively. Entry/exit zones that are caused by stationary noise are indications of stationary motion noise sources, which can subsequently be ignored. Also, knowledge of the route models can be used to provide short-term predictions to support blob matching.

Conceptual descriptions of the activity are used to automatically annotate the surveillance video, e.g., "pedestrian #445 enters the scene at entry zone A, moves along the route AB and exits the scene at exit zone B." If the scene elements are labeled manually according to real scene features, then the descriptions are more meaningful, e.g., "A pedestrian enters the scene from the Student Union, walks toward the main entrance and exits the scene at the main entrance."

Online classification of trajectories to routes can provide long-term predictions about targets expected destination. More specifically, a trajectory is classified online, as soon as it is derived by the motion tracker. For each possible matching of the incomplete trajectory to a route model, a probability is assigned which shows the likelihood that the target will follow the specific route.

As a consequence of the discrete nature of our scene model, as expressed in the topological representation, activity analysis can be performed using a BBN, overlaid on the network of the scene elements. We use a Markovian field, overlaid onto our network and activity analysis is performed using a hidden Markov model (HMM) theory [25]. However, in our probabilistic model, the states are not hidden, as they have already been learnt. This is rather a benefit, as learning the HMM parameters is considerably simplified.

Activity cannot be assumed as a stationary process, as it is highly dependent on the time of the day. Therefore, we use a time-variant HMM to represent this variation. Using Observation Evaluation of the HMM theory, trajectories can be characterized as typical or atypical (see, for example, Figs. 29 and 30). Atypical trajectory identification can be used to alert the security personnel to incidents that may require a closer look.

[2]The conceptual descriptors can be generated online, in real time, or offline at times that the system is idle. When a conceptual query is set, the system exploits the existing conceptual labels to facilitate fast search of the database.

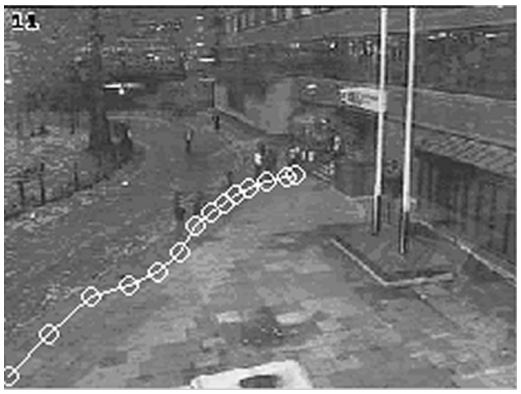

Fig. 29. Typical trajectory according to the HMM model.

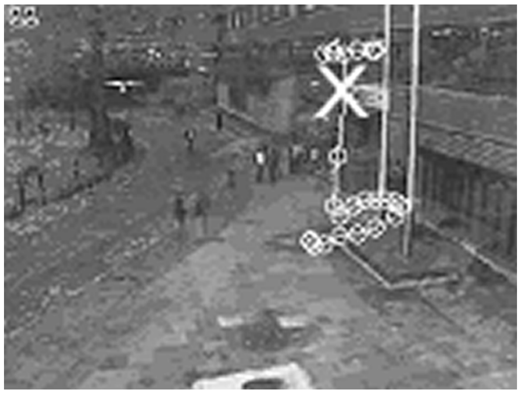

Fig. 30. Atypical trajectory, according to the HMM model. Large X notes where atypicality is detected.

X. Conclusion

We have developed an activity-based semantic scene model for an area that is viewed by a video surveillance system. Semantics of our model include entry/exit zones, paths, routes, and stop zones.

A set of methods is presented that allow learning of the scene elements from observations, automatically. The unsupervised nature of the proposed algorithms allows the implementation of a visual surveillance system that "observes" and "learns" its environment.

While the methods have been applied individually to cameras in a multicamera surveillance system (see Fig. 28), we describe elsewhere [26] how we can automatically learn the connectivity of paths between cameras.

The knowledge of such semantic models can have many applications in the visual surveillance domain, like conceptual databases, tracking process enhancement, video annotation, long-term predictions and atypical activity detection.

Acknowledgment

The authors would like to thank J. Black and M. Xu for supplying the trajectory data.

References

[1] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[2] M. Xu and T. Ellis, "Partial observation versus blind tracking through occlusion," in *Proc. BMVC*, Sep. 2002, pp. 777–786.

[3] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *IEEE Workshop Motion and Video Computing*, Orlando, FL, Dec. 2002, pp. 169–174.

[4] M. Xu and T. Ellis, "Illumination-invariant motion detection using color mixture models," in *Proc. BMVC*, Manchester, U.K., Sep. 2001, pp. 163–172.

[5] J. Black, T. J. Ellis, and D. Makris, "A hierarchical database for visual surveillance applications," in *IEEE Int. Conf. Multimedia and Expo*, vol. 3, Jun. 2004, pp. 1571–1574.

[6] A. R. Hanson and E. M. Riseman, "The VISIONS image-understanding system," *Proc. Advances in Computer Vision*, vol. 1, pp. 1–114, 1988.

[7] B. Neumann, "Natural language description of time-varying scenes," *Semantic Structures: Advances in Natural Language Processing*, 1989.

[8] M. Mohnhaupt and B. Neumann, "Understanding object motion: Recognition, learning, and spatio-temporal reasoning," *J. Robot. Auton. Syst.*, vol. 8, pp. 65–91, 1991.

[9] R. J. Howard and H. Buxton, "Analogical representation of spatial events, for understanding traffic behavior," in *Proc. 10th Eur. Conf. AI*, 1992, pp. 785–789.

[10] R. J. Howarth, "Spatial representation and control for a surveillance system," Ph.D. dissertation, Queen Mary and Westfield College, Univ. London, London, U.K., 1994.

[11] J. H. Fernyhough, A. G. Cohn, and D. C. Hogg, *Generation of Semantic Regions from Image Sequences*, B. Buxton and R. Cipolla, Eds. New York: Springer-Verlag, 1996, pp. 475–478.

[12] N. Johnson and D. C. Hogg, "Learning the distribution of object trajectories for event recognition," in *Proc. BMVC*, Birmingham, UK, Sep. 1995, pp. 583–592.

[13] E. B. Koller-Meier and L. Van Gool, "Modeling and recognition of human actions using a stochastic approach," in *Proc. Eur. Workshop n Advanced Video-Based Surveillance Systems*, Kingston, U.K., Sep. 2001, pp. 17–28.

[14] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proc. CVPR*, Santa Barbara, CA, Jun. 1998, pp. 22–31.

[15] C. Stauffer, "Scene reconstruction using accumulated line-of-sight," M.S. dissertation, Mass. Inst. Technol., Cambridge, 1997.

[16] D. Makris and T. Ellis, "Finding paths in video sequences," in *Proc. Brit. Machine Vision Conf.*, vol. 1, Manchester, U.K., Sep. 2001, pp. 263–272.

[17] ——, "Path detection in video surveillance," *Image Vis. Comput.*, vol. 20/12, pp. 895–903, Oct. 2002.

[18] ——, "Spatial and probabilistic modeling of pedestrian behavior," in *Proc. Brit. Machine Vision Conf.*, vol. 2, Cardiff, U.K., Sep. 2002, pp. 557–566.

[19] ——, "Automatic learning of an activity-based semantic scene model," in *Proc. IEEE Int. Conf. Advanced Video and Signal Based Surveillance*, Miami, FL, Jul. 2003, pp. 183–188.

[20] C. Stauffer, "Estimating tracking sources and sinks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Madison, WI, Jul. 2003, pp. I-259–I-266.

[21] H. Buxton, "Generative models for learning and understanding dynamic scene activity," in *Eur. ConF. Computer Vision, Workshop on Generative Model-Based Vision*, Copenhagen, Denmark, Jun. 2002, pp. 77–81.

[22] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. CVPR*, Fort Colins, CO, 1999, pp. 246–252.

[23] F. Cupillard, F. Bremond, and M. Thonnat, "Behavior recognition for individuals, groups of people, and crowd," presented at the *IEE Seminar Intelligent Distributed Surveillance Systems*, London, U.K., Mar. 2003.

[24] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.*, vol. B-39, pp. 1–38, 1977.

[25] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[26] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition* , Washington D.C., Jun. 2004, pp. 205–210.

Dimitrios Makris (M'03) was born in Johannesburg, South Africa, in 1975. He received the Diploma degree in computer and electronic engineering from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1999 and the Ph.D. degree in computer vision from City University, London, U.K., in 2004.

He joined Kingston University, London, as Research Assistant in 2003, and is currently a Lecturer in the School of Computing and Information Systems. His previous experience includes his internships at OTE (Hellenic Telecommunications Organization), Greece (1997), and at AHEPA Hospital, Greece (1998), and a visiting lectureship at City University (2001–2004), London. His research interests are in visual surveillance, machine learning, and cognitive vision.

Dr. Makris is member of the British Machine Vision Association.

Tim Ellis (M'03) received the B.Sc. degree in physics from the University of Kent, Canterbury, U.K., in 1974 and the Ph.D. degee in biophysics from University of London, London, U.K., in 1981.

He joined City University, London, in 1979 as a Research Assistant, and was awarded a five-year advanced fellowship from 1982 to 1987 by the Science and Engineering Research Council. He was appointed to a lectureship in 1987 and to Reader in 1995. In 2003, he joined the School of Computing and Information Systems at Kingston University, London, where he is currently Professor and Head of School. His research interests are in visual surveillance, industrial inspection, color image analysis, and vision systems hardware.

Prof. Ellis is a member of the IEE and is a past Chairman of the British Machine Vision Association.