Trajectory Learning for Activity Understanding: Unsupervised, Multilevel, and Long-Term Adaptive Approach

Brendan Tran Morris, *Member*, *IEEE*, and Mohan Manubhai Trivedi, *Fellow*, *IEEE*

Abstract—Society is rapidly accepting the use of video cameras in many new and varied locations, but effective methods to utilize and manage the massive resulting amounts of visual data are only slowly developing. This paper presents a framework for live video analysis in which the behaviors of surveillance subjects are described using a vocabulary learned from recurrent motion patterns, for real-time characterization and prediction of future activities, as well as the detection of abnormalities. The repetitive nature of object trajectories is utilized to automatically build activity models in a 3-stage hierarchical learning process. Interesting nodes are learned through Gaussian mixture modeling, connecting routes formed through trajectory clustering, and spatio-temporal dynamics of activities probabilistically encoded using hidden Markov models. Activity models are adapted to small temporal variations in an online fashion using maximum likelihood regression and new behaviors are discovered from a periodic retraining for long-term monitoring. Extensive evaluation on various data sets, typically missing from other work, demonstrates the efficacy and generality of the proposed framework for surveillance-based activity analysis.

Index Terms—Trajectory clustering, real-time activity analysis, abnormality detection, trajectory learning, activity prediction.

1 Introduction

The dramatic decrease in cost for quality video equipment coupled with the ease of transmitting and storing video data has led to widespread use of vision-based analysis systems. Cameras are in continuous use all around, along highways to monitor traffic, for security of airports and other public places, and even in our homes. Methods to manage these huge volumes of video data are necessary as it has almost become an impossible task to continually monitor these video sources manually. Vision researchers have been forced to develop robust real-time methods to recognize events and activities of interest as a means to compress video data into a more manageable form, and to provide annotations that can be used for search indexing.

While researchers have been long been interested in understanding human behavior, much of the work has relied on high resolution descriptors such as state space approaches [1], 3D reconstruction of body configurations [2], volumetric space-time shapes [3], or body part decomposition [4]. In the surveillance and monitoring setting, such rich descriptors may not be available or may not be necessary. Rather, coarse center-of-body motion can be reliably extracted from either rigid or deformable objects (e.g., vehicles or humans) in these far-field situations. This motion has cued human activity understanding through selective focus of attention in PTZ image capture [5] or detection of loitering individuals [6]. In the transportation setting, vehicle motion has been utilized for traffic congestions measurements [7] and detailed origin-destination information at intersections [8]. But, these techniques require domain knowledge and as more video sites have been erected, the need for automatic methods to characterize activity from data rather than by manual specification [9] has greatly increased, as these methods provide a framework for unsupervised detection of interesting or abnormal events.

This work seeks to understand suveillance scene activity with minimal constraints and limited domain knowledge by developing a probabilistic trajectory analysis framework. A 3-stage hierarchical modeling process, which characterizes an activity at multiple levels of resolution, is developed to classify and predict future activity and detect abnormal behavior. The unsupervised learning scheme is able to learn the points of interest in a scene, cluster trajectories into spatial routes, capture dynamics and temporal variations by a hidden Markov model, and provide an activity adaption methodology to characterize a novel scene over long time periods without a priori knowledge. The framework effectively handles imperfect tracking and can automatically estimate the number of typical activities in a scene. Finally, extensive evaluation on a number of data sets, which is lacking in the literature, is provided to quantify analysis performance.

2 Related Research

A surveillance activity is composed of a sequence of ordered actions and a trajectory is the set of observable motion measurements. Often the observed motion patterns in visual surveillance systems are not completely

• *The authors are with the Computer Vision and Robotics Research Laboratory, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0434.*
E-mail: {b1morris, mtrivedi}@ucsd.edu.

Manuscript received 27 July 2009; revised 9 May 2010; accepted 30 Nov. 2010; published online 16 Mar. 2011.
Recommended for acceptance by T. Darrell.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-07-0483.
Digital Object Identifier no. 10.1109/TPAMI.2011.64.

random but have some underlying structure which has been exploited to build models that allow for accurate inferencing.

Pioneering work by Johnson and Hogg [10] described outdoor motions with a flow vector and learned implicit temporal relationships using a leaky neural network. The leaky network was modified with feedback between output nodes and the leaky neurons to enable prediction by Sumpter and Bulpitt [11]. Owens and Hunter [12] extended this idea using a self-organizing feature map to further detect abnormal behavior. Unfortunately, the neural network implementations required difficult fine tuning of parameters and large amounts of data which resulted in slow convergence.

By utilizing a complete trajectory as input for clustering, Hu et al. drastically improved the learning process speed [13]. This explicitly enforeced the sequential nature of trajectory observation points and allowed researchers to extend the modeling to detect abnormalities and make predictions by utilizing previously observed motion [7], [14], [15], [16]. Multilayered learning approaches have been developed to first model the spatial extent then dynamics of an activity [15]. Junejo et al. characterized an activity by its extent, the traversal speed, as well as the amount curvature. Makris and Ellis [17] developed an online method for building up spatial envelopes for each activity. Others have learned the smaller action groups and connected them within a Markov model [18] or tree-like structure [19]. By learning actions, new activities branch out from existing models (natural for online learning) and enable efficient learning through the use of shared data.

In stark contrast, others have completely ignored the inherent ordering in a trajectory. Rather than require full trajectories, which are difficult to obtain due to scene clutter and occlusion, only interframe motion is considered for bag-of-word type learning methods. Stauffer and Grimson created a codebook of motion flows and learned the co-occurrence of motions. Similarly, drawing inspiration from document clustering research, Wang et al. [20], [21], [22] have developed hierarchical Bayesian models to group co-occurring motions into actions and activities, jointly and without supervision. Xiang and Gong [23] were able to segment videos into clips of different activities based on the spectral similarity of behaviors.

A more detailed examination of techniques is presented in the review by Morris and Trivedi [24], which highlights a wide range of applications, challenges, and common approaches to trajectory learning and modeling. They emphasize not only the learning framework but also trajectory representation [25], [26], [27] and similarity metrics for comparison [28], [29].

This work presents a new multilevel trajectory learning framework which is capable of automatically estimating the number of activities in a scene and adapts to changes over time. Thorough evaluation of performance, lacking in related studies, is presented to characterize analysis performance.

3 Trajectory Learning Framework

This paper adopts the general probabilistic trajectory analysis framework, shown in Fig. 1, to automatically learn and describe activity. At the front end, a vision-based background subtraction module detects moving objects and a tracking process creates trajectories which are the main input for activity inferencing [7]. Coarse body motion is utilized as a reliable low level feature which can describe how an agent moves through a visual scene. A trajectory is a sequence of T flow vectors:

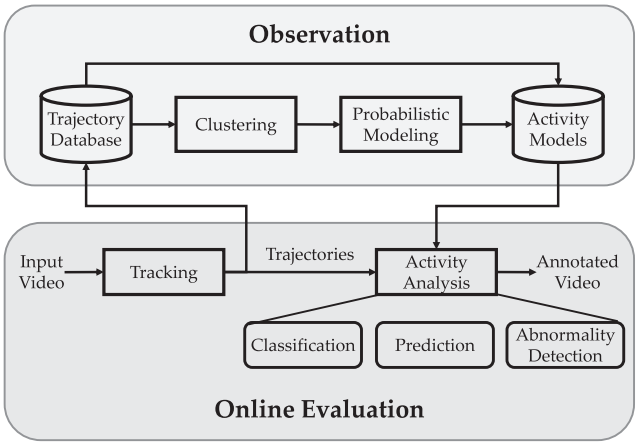


Fig. 1. The general framework for trajectory analysis. During the observation (training) phase, trajectories are clustered and modeled. The learned set of typical patterns are used for live activity analysis in the online evaluation phase.

$$F = \{f_1, \ldots, f_t, \ldots, f_T\}, \tag{1}$$

$$f_t = [x_t, y_t, u_t, v_t]^{\mathrm{T}}, \tag{2}$$

where flow vector f_t compactly represents an object's motion at time t by the (x, y) position and corresponding component velocities (u, v).

During the initial observation and learning phase, trajectories are collected to create a measurement database. The trajectory database is clustered to find similar motion patterns without specifying those of interest. Finally, the groups of similar trajectories are probabilistically modeled to populate an activity database for inferencing.

A key for designing an effective learning framework is to examine and limit the types of behaviors to be analyzed. This work is concerned with simple activities composed of smaller actions which are performed by a single agent. More complex behaviors, e.g., interactions between agents, are left for future studies. As shown in Fig. 2, an activity is defined by a sequence of atomic actions and a trajectory is the observable measurement that summarizes activity history.

An activity can be decomposed at different resolutions to answer specific analysis questions. An activity can be characterized by four main components (left side of Fig. 3) that form an explanation hierarchy with each level providing a more refined activity description. Each of the levels is briefly described to highlight its analysis importance.

Level 1 (Node). A simple compact representation of the interesting endpoints (origin and destination information is used by the transportation community to understand to and from where people travel).

Level 2 (Spatial). Different routes between nodes spatially localize differing activities and separates the straight route from a more round about approach.

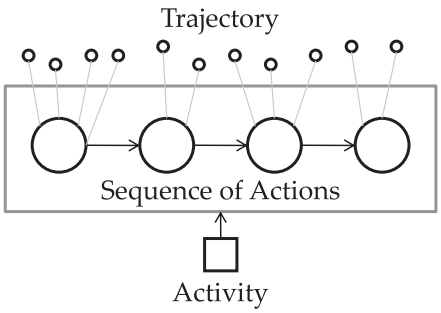


Fig. 2. A surveillance activity is defined by a set of atomic actions and composed by the action sequence. Trajectory points are samples drawn from each action and relate the duration and ordering of actions.

Level 3 (Dynamic). Further disambiguation is made by considering the dynamics in how routes are traversed to make a distinction between slowly and quickly moving activities (e.g., free-flow versus congestion on the highway).

Level 4 (Temporal). The finest level of activity characterization accounts for temporal variations. This most closely matches the activity model to account for the duration and sequencing of actions.

The 3-stage hierarchical learning process presented on the right of Fig. 3 describes each of the levels in turn. Nodes or points of interests (POI) are discovered using Gaussian mixture modeling (GMM) and expectation maximization. The spatial routes are extracted by comparing and clustering similar trajectories while simultaneously and automatically determining the number of activities in a scene. In the last stage, both the spatio-temporal dynamics that characterize an activity are encoded in the probabilistic hidden Markov model (HMM).

The evaluation phase uses the learned activity HMMs as a descriptive vocabulary for online analysis. As a new object is tracked, its activity is characterized, future behavior is predicted, and alarms are signaled if there is an abnormal or unusual action. In addition, recent observations are used to update and modify the activity models to better reflect the surveillance scene.

The analysis framework must cope with a number of real-world implementation issues in order to facilitate deployment. In a new scene, the number of typical activities is not known a priori and must be estimated automatically. In addition, the learning and evaluation algorithms must gracefully handle incomplete trajectories due to faulty tracking. Finally, when monitoring a site over long periods, the initial training period may not reflect the current scene configuration, necessitating approaches to modify the activity models. In the following sections, more details of the learning and analysis framework are provided, as well as an evaluation methodology.

4 Node Level Learning

The first activity level locates the points of interest in the image plane. These POI are represented as graph nodes in a topographical map of the scene (Fig. 4). The POI are used to filter noisy trajectories which arise from tracking failures to provide robustness and improve the automatic activity definition.

4.1 Points of Interest

There are three different types of POI in a scene: entry, exit, and stop. The entry and exit POI are the locations where objects either appear or disappear from the scene (e.g., the edge of the camera field of view) and correspond to the first, f_1, and last, f_T, tracking point, respectively.

The stop POI indicate scene landmarks where objects tend to idle or remain stationary (e.g., a person sitting at a desk). Potential stop points must have very low speed for at least τ seconds. But, unlike the inactivity regions of Dickinson and Hunter [30], stop regions are also localized. The stop POI shown in Fig. 4 is defined by $\tau/\Delta t$ consecutive points with speed less than the threshold V_{stop} and all within the small radius R. This more complex inactivity definition ensures a stationary object remains in a particular location. Just relying on a low speed threshold can contaminate the set of stop points, e.g., samples of a vehicle in congestion

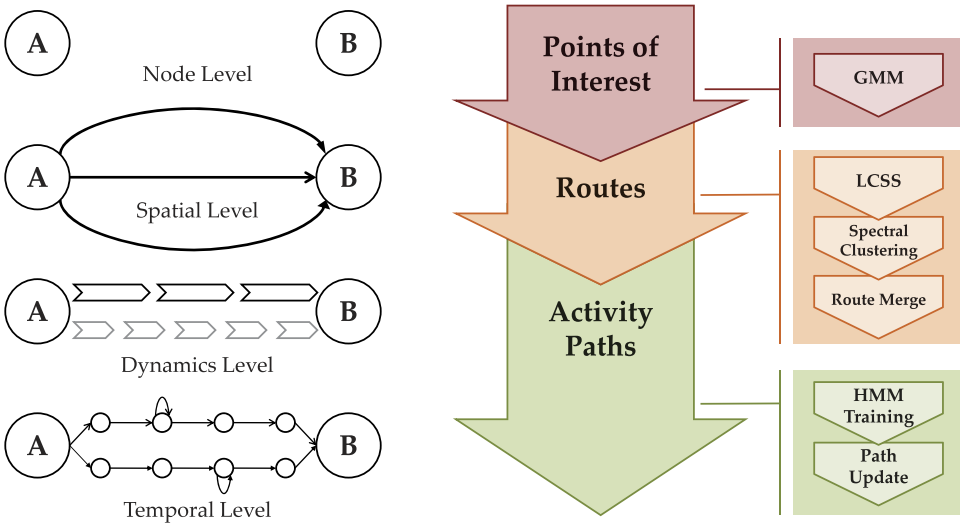


Fig. 3. In order to model activities at various resolutions, a 3-stage hierarchical learning procedure is adopted. The first level learns points of interest (nodes), the second locates spatial routes between nodes through clustering, and the final level probabilistically encodes spatio-temporal dynamics.

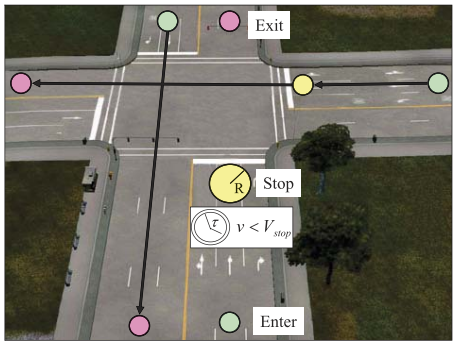


Fig. 4. Graph representation of a visual scene where nodes indicate points of interest while the edges signify the spatial connections (routes). The green nodes are the locations where objects enter the scene, red where they exit, and yellow are regions where an object stops and remains stationary.

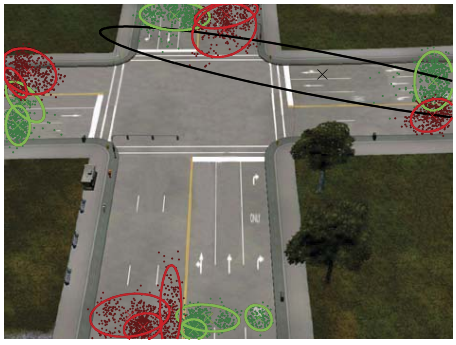


Fig. 5. Entry/Exit interest are zones learned by Gaussian mixture modeling (no stop zones in this data set).

could be consistent with the low speed check even though it travels across the the full camera field of view.

The POI are learned through a 2D mixture of Gaussian zone modeling procedure [17]. For each POI type {enter, exit, stop}, a zone consisting of Z Gaussian components,

$$\mathcal{Z} = \sum_{m=1}^{Z} z_m G([x, y]^{\mathrm{T}}, \mu_m, \Sigma_m), \tag{3}$$

is learned using expectation maximization (EM) [31], with $[x, y]^{\mathrm{T}}$ the image coordinates of a point in the POI set. Each mixture component m denotes the location of an activity node on the image plane by its mean μ_m and its size is specified by Σ_m. The density of a node,

$$d_m = \frac{z_m}{\pi\sqrt{|\Sigma_m|}}, \tag{4}$$

defines its importance.

Since it is not known a priori how many POI belong in a zone, the number must be estimated. The unimportant zones components can be automatically discarded using a density criterion based on the average mixture threshold,

$$L_d = \frac{\alpha_Z}{\pi\sqrt{|\Sigma_Z|}}, \tag{5}$$

where $0 < \alpha_Z < 1$ is a user defined weight and Σ_Z is the covariance matrix of the entire zone data set [17]. Zone components with low density (low importance) $d_m < L_d$ do not have much support in the training data and most likely model noise points in the tracking process (broken trajectories) since they are not well localized, while tight mixtures with high density indicate valid POI.

Fig. 5 shows the entry/exit zones learned for an intersection with green denoting entry and red exit POI. Noise mixtures from broken tracks, which are removed, are drawn in black.

4.2 POI-Based Filtering of Broken Tracks

The POI modeling procedure indicates important nodes in the surveillance scene which can be used to help the activity training process. Including noise tracks generated during tracking failure from things like occlusion would make activity learning more difficult because they are not adequately explained and result in wasted modeling effort.

The POI-based filtering process considers any trajectory that does not begin and end in a POI as a tracking error and removes it from the training database. Trajectories that travel through a stop zone are split into separate tracks leading into and out of the zone. The enter/exit and stop filtering provides the consistent trajectories needed for robust activity modeling.

4.3 Discussion

The Node Level learning is able to distinguish the image locations where objects should appear, disappear, and remain stationary. While it is evident that the mixture of Gaussian process is not able to perfectly place POI for each lane, the top left enter POI actually incorporates three different lanes in Fig. 5; it is effective at removing broken trajectories from the training database. The choice of Z for a zone is not critical because, when underspecified, EM will create a larger mixture component.

Better POI localization might be possible by automatically selecting the number of mixture components in conjunction with estimation by modifying the classical EM algorithm [32].

5 Spatial Level Learning

The second behavior level focuses on spatial support, differentiating tracks based on where they occur. Once the scene goals are explained by POI, the connections between nodes are described by spatial routes.

5.1 Trajectory Clustering

After POI-based filtering, a training database of clean trajectories is available for grouping into routes which separate different ways to get between nodes. These routes can be learned in unsupervised fashion through clustering. Clustering groups typical trajectory patterns and only relies on the definition of similarity between tracks. The main difficulties when trying to learn routes are the time-varying nature of activities, which leads to unequal length trajectories, and not knowing how many routes exist, necessitating unsupervised learning techniques.

5.1.1 LCSS Trajectory Distance

Since trajectories are realizations of an activity process, the length of trajectories can be of unequal length based on the properties of differing activities. Additionally, even

Spectral Clustering of Trajectories

1) Construct similarity graph $S = \{s_{ij}\}$
2) Compute the normalized Laplacian L, $L = I - D^{-1/2} S D^{-1/2}$
3) Compute the first K eigenvectors of L
4) Let $U \in \mathcal{R}^{N \times K}$ be the normalized matrix constructed with eigenvectors as columns
5) Cluster the rows of U using k-means

Fig. 6. Basic steps for spectral clustering of trajectories as presented by Ng et al. [33].

trajectories sampled from the same activity can be of unequal length due to sampling rate and speed of action. In order to compare trajectories, a distance measure must be able to handle variable sized inputs.

LCSS is a sequence alignment tool for unequal length data that is robust to noise and outliers because not all points need to be matched. Instead of a one-to-one mapping between points, a point with no good match can be ignored to prevent unfair biasing. When modified for trajectory comparison, LCSS was found to provide the best clustering performance when compared with a number of other popular distance measures [28], [34]. The LCSS distance for trajectories suggested by Vlachos et al. [35] is defined as

$$D_{LCSS}(F_i, F_j) = 1 - \frac{LCSS(F_i, F_j)}{\min(T_i, T_j)}, \tag{6}$$

where the $LCSS(F_i, F_j)$ value specifies the number of matching points between two trajectories of different length, T_i and T_j. The recursive LCSS definition

$$LCSS(F_i, F_j) = \begin{cases} 0, & T_i = 0 \mid T_j = 0, \\ 1 + LCSS(F_i^{T_i - 1}, F_j^{T_j - 1}), & d_E(f_{i,T_i}, f_{j,T_j}) \\ & \quad < \epsilon \;\&\; |T_i - T_j| < \delta, \\ \max(LCSS(F_i^{T_i - 1}, F_j^{T_j}), & \\ \quad LCSS(F_i^{T_i}, F_j^{T_j - 1})), & \text{otherwise.} \end{cases} \tag{7}$$

which can be efficiently computed with dynamic programming, matches points that are within a small euclidean distance ϵ and an acceptable time window δ. The term $F^t = \{f_1, \ldots, f_t\}$ denotes all of the sample points in F up to time t.

5.1.2 Spectral Clustering

Spectral clustering has become a popular technique recently because it can be efficiently computed and has improved performance over more traditional clustering algorithms [34]. Spectral methods do not make any assumptions on the distribution of data points and instead relies on eigendecomposition of a similarity matrix which approximates an optimal graph partition. The basic steps for normalized spectral clustering presented by Ng et al. [33] are outlined in Fig. 6

The similarity matrix $S = \{s_{ij}\}$, which represents the adjacency matrix of a fully connected graph, is constructed from the LCSS trajectory distances using a Gaussian kernel function:

$$s_{ij} = e^{-D^2_{LCSS}(F_i, F_j)/2\sigma^2} \in [0, 1], \tag{8}$$

where the parameter σ describes the trajectory neighborhood. Large values of σ cause trajectories to have a higher similarity score while small values lead to a more sparse similarity matrix (more entries will be very small). The Laplacian matrix is formed from the adjacency matrix,

$$L = I - D^{-1/2} S D^{-1/2}, \tag{9}$$

with D the diagonal degree matrix with elements the sum of the same row in S. A new $N \times K$ matrix, U, is built using the first K eigenvectors of L as columns. Finally, the rows of U, each viewed as a new feature vector representation of a training trajectory F, are clustered using fuzzy C means (FCM) [36] to form K groups of similar tracks. The initial cluster centers are chosen based on the orthogonal initialization method proposed by Hu et al. [37].

Rather than k-means, FCM is used in the last spectral clustering step to minimize the effects of outliers and obtain soft cluster membership values $u_{ik} \in [0, 1]$ which indicate the quality of training sample (how confidently trajectory i is place in cluster k). High membership means little route ambiguity or a typical realization of an activity process.

5.1.3 Route Creation

The spectral clustering process partitions the trajectory training database into groups of similar patterns. A route is defined as a prototype from each group. The route prototype is chosen as an average of trajectories in a cluster which utilizes the membership values u_{ik} returned from FCM.

In order to average trajectories of different length, each trajectory F has its velocity information ignored and is spatially resampled to a fixed size L. Rather than simple temporal subsampling, the resampled track evenly distributes points along the trajectory. A trajectory is viewed as a parameterized curve $F = \{x(t), y(t)\}$ and is reparameterized in terms of arc length $F(s)$ to consider only the trace of the curve. The sampling ensures the euclidean distances between consecutive points are equal to completely remove dynamic information (as hidden in the temporal sampling rate). This prevents regions of higher sample density from contributing bunches of points in a single area as occurs when an object is moving slowly during a section of its trajectory. Finally, a vector $\bar{F} = [x_1, y_1, \ldots, x_L, y_L]$, representing a point in the $\mathbb{R}^{2L}$ route space, is constructed for each of the N training trajectories by stacking the consecutive trace points.

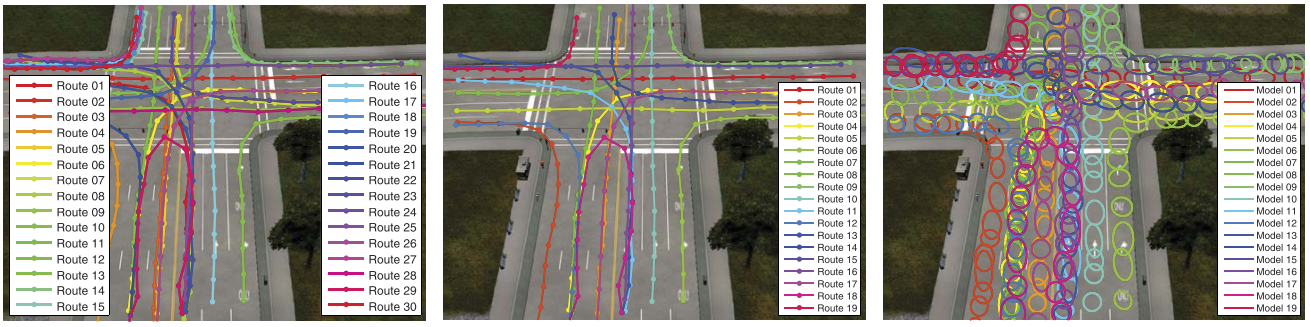
A route prototype is constructed as the weighted average of the training database

$$r_k = \frac{\sum_{i=1}^{N} u_{ik}^2 \bar{F}_i}{\sum_{i=1}^{N} u_{ik}^2}, \tag{10}$$

where u_{ik} is the FCM membership of example i to cluster k. The resulting set of K prototypes $\{r_k\}$ encode the the typical scene routes.

5.2 Cluster Validation

Spectral clustering is performed with a large K to partition the training data but the true number of routes must still be estimated because it is not known a priori. After clustering,



(a) (b) (c)

Fig. 7. (a) A large set of clusters, representing intersection maneuvers, are learned by spectral clustering with LCSS trajectory distance. (b) The correct $N_r = 19$ routes remain after merging. (c) The spatio-temporal dynamics of each route is encoded by an HMM.

the routes are subsequently refined to a smaller number ($N_r < K$) by merging similar prototypes.

The merge procedure compares routes by finding an alignment between pairs of routes using dynamic time warping (DTW) [38]. DTW is used to equally value all matches and provide a one-to-one matching between points on routes. After alignment, two routes are considered similar if the number of matching points is high or if the total distance is small. The match count is defined as

$$M(r_m, r_n) = \sum_{l=1}^{L} \mathcal{I}(d_l(r_m, r_n) < \epsilon_d), \tag{11}$$

$$d_l(r_m, r_n) = \sqrt{(x_m(l) - x_n(l))^2 + (y_m(l) - y_n(l))^2}, \tag{12}$$

where $\mathcal{I}(.)$ is the indicator function and the total distance between routes is

$$D(r_m, r_n) = \sum_{l=1}^{L} d_l(r_m, r_n). \tag{13}$$

The threshold value ϵ_d pixels was chosen experimentally for good results and is based on how closely routes are allowed to exist in the image plane. When either $M(r_m, r_n) > T_M$ or $D(r_m, r_n) < L\epsilon_d$, then routes r_m and r_n are considered similar.

A cluster correspondence list is created from these pairwise similarities, forming similarity groups. Each correspondence group is reduced to a single route by removing similar routes and transferring the membership weight of every training trajectory onto the remaining route, e.g.,

$$\tilde{u}_{mi} = u_{mi} + u_{ni} \quad \forall n \sim m, \tag{14}$$

where $\tilde{u}$ represents the membership after merging.

In practice we have found that FCM tends not to overfit the data but instead finds several very similar clusters, making the merge algorithm effective. The route prototypes are plotted in Fig. 7a for $K = 30$. The turns in the upper corners have multiple similar clusters which are removed by the match-merge procedure to retain only the true $N_r = 19$ routes shown in Fig. 7b.

5.3 Discussion

There are many options when clustering trajectories, ranging from the choice of clustering algorithm to the measure of similarity between tracks [24]. The choice of LCSS distance along with spectral clustering has been proven to be very effective for surveillance settings [28], [34] but still requires a model order choice and validation scheme such as the match-merge process presented.

Additionally, automatic route definition relies on consistent and repetitive activities; rare occurrences will be lost during clustering. A hierarchical clustering procedure can avoid active selection of model order and provides a set of solutions at varying resolutions which could prove effective for handling infrequently recurring activities.

6 Dynamic-Temporal Level Learning

The clustering procedure locates routes spatially, but this is insufficient for detailed analysis. It is necessary to know not only where objects are but also the manner in which they travel. The higher order dynamics and temporal structure of the third and fourth activity hierarchy levels are needed to completely characterize an activity path.

6.1 Activity HMM

A path provides a compact representation of an activity which incorporates location, dynamics, and temporal properties. Each activity is represented using a hidden Markov model because it naturally handles time normalization. In addition, the simplicity of training and evaluation makes it ideal for real-time implementation.

An HMM, in compact notation $\lambda = (A, B, \pi_0)$, is characterized by the following parameters:

- The number of states (actions) in the model Q. In this work, Q is fixed for simplicity but it is possible to estimate an optimal number [39].
- The $Q \times Q$ state transition probability matrix $A = \{a_{ij}\}$ where

$$a_{ij} = p(q_{t+1} = j | q_t = i). \tag{15}$$

- The observation probability distribution $B = b_j(f)$ where

$$b_j(f) = G(f, \mu_j, \Sigma_j) \tag{16}$$

  represents the Gaussian flow f distribution of each of the $j = 1, \ldots, Q$ states of unknown mean μ_j and covariance Σ_j.
- The initial state distribution $\pi_0 = \{\pi_j\}$ with

$$\pi_j = p(q_1 = j). \tag{17}$$

The temporal level of the learning hierarchy in Fig. 3 gives a graphical representation of an HMM where circles represent the states and arrows indicate the state transition.

The activity path HMM is a left-right hidden Markov model ($a_{ij} = 0$ for $j < i$) which encodes the sequential start to finish structure of trajectories by prohibiting backtracking. Each of the $k = 1, \ldots, N_r$ scene activities is represented as $\lambda_k = (A_k, B_k, \pi_0)$ with a fixed π_0 for each path of

$$\pi_0(j) = \frac{1}{C} e^{-\alpha_p j} \quad j = 1, \ldots, Q. \tag{18}$$

C is a normalization constant to ensure valid probabilities. This definition allows tracks to begin in any state, which is important when trajectories are incomplete due to occlusion or during online analysis. The exponential weighting is used to prefer action sequences that begin decoding in earlier states, though the choice of α_p is not crucial as long as there is nonzero probability for each state.

The path model is finalized when both the transition probabilities A_k and the action states which define B_k are learned from the trajectory training database.

6.2 Path Training

An HMM is trained for each activity by dividing the training set D into N_r disjoint sets, $D = \bigcup_{k=1}^{N_r} D_k$. The set

$$D_k = \{F_i | u_{ik} > 0.8\} \tag{19}$$

contains only the most representative examples of cluster k. In this stage of learning, the full trajectory, including position and velocity, $F_i = \{f_t\}$ with $f_t = [x_t, y_t, u_t, v_t]^{\mathrm{T}}$, is utilized for a precise spatio-temporal model. Using path training sets D_k, the N_r HMMs can be efficiently learned using standard methods such as the Baum-Welch method [38] to find the transition probabilities A_k and the Gaussian state observation distributions μ_{jk} and Σ_{jk}.

The set of learned activity paths which specify where lanes are located as well as how vehicles are expected to move in the lane for the traffic intersection are shown in Fig. 7c. Typically, the path models have a highly structured transition matrix which is block diagonal. For long activities with few model states ($T \gg Q$), the highest transition probability is along the main diagonal, which means it is typical to have multiple observations from a single action.

Notice that while data could be shared and states tied in regions of overlap between HMMs, doing so would discard subtle cues which distinguish activities. For example, the slowing necessary to make a right turn through an intersection is not present when passing straight through.

The resultant activity models after the third stage of the hierarchical activity learning process provide a simple probabilistic interpretation of an activity. In addition, new camera setups can be easily deployed because there was no need to manually select "good" trajectories for the modeling since they were automatically discovered.

6.3 Updating Path Models

The activity HMMs learned above accurately depict the scene at the time of training, but, in a surveillance setting, there is no guarantee that the activity processes are stationary, meaning the models must reflect changes over time. Two complementary adaption methods are used to update the HMM database. The first is an online scheme to refine the existing activity HMMs based on newly observed trajectories, while the second introduces new models through periodic relearning rounds.

6.3.1 Online Incremental Update

After training, the activity models are optimal for the training data, but, after time, might not reflect the current configuration of the scene. Small variations and perturbations could arise from various causes such as camera movement or path reconfiguration, e.g., people walking around a puddle. An activity HMM can be updated with new trajectories in an online fashion using maximum likelihood linear regression (MLLR) [40]. MLLR computes a set of linear transformations that reduce the mismatch between the initial model and new (adaption) data. The adapted HMM state mean is given by

$$\hat{\mu}_j = W_k \xi_j, \tag{20}$$

where W_k is the $4 \times (4+1)$ transformation matrix for activity k and ξ_j is the extended mean vector

$$\xi_j = \left[1, \mu_j^{\mathrm{T}}\right]^{\mathrm{T}} = [1, \mu_x, \mu_y, \mu_u, \mu_v]^{\mathrm{T}} \tag{21}$$

of state j. State j represents an individual action that makes up an activity and $\mu_j = [\mu_x, \mu_y, \mu_u, \mu_v]^{\mathrm{T}}$ summarizes the location and dynamics of the action. $W_k = [b\ H]$ produces an affine transformation for each Gaussian HMM state with H a transformation and b a bias term. The transformation matrix W_k can be found using EM by solving the auxiliary equation

$$\sum_{t=1}^{T} \sum_{j=1}^{Q} L_j(t) \Sigma_j^{-1} f_t \xi_j^{\mathrm{T}} = \sum_{t=1}^{T} \sum_{j=1}^{Q} L_j(t) \Sigma_j^{-1} W_k \xi_j \xi_j^{\mathrm{T}}, \tag{22}$$

where $L_j(t) = p(q_j(t)|F)$. The optimization is performed over all Q states of an HMM to provide an activity level regression.

Each time a new trajectory is classified into path λ_k (below in Section 7.1.1), a transformation is learned and applied to the mean of each of the HMM states for a sequential online update. The update

$$\mu_j(t+1) = (1 - \alpha_{\mathrm{MLLR}}) \mu_j(t) + \alpha_{\mathrm{MLLR}} W_k(t) \xi_j \quad \forall j, \tag{23}$$

modifies the mean of existing path λ_k to better fit new observations at time t. The learning rate $\alpha_{\mathrm{MLLR}} \in [0, 1]$ is a user-defined parameter set to control the importance of a new trajectory.

6.3.2 Incorporating New Activities

The MLLR update allows modification of existing activities but, in order to introduce new and unseen activities into the activity set, a periodic batch retraining procedure is adopted [15]. Abnormal trajectories that are not well explained by any of the activity models (defined in Section 7.1.2) are collected to form an auxiliary training database. Once the database has grown sufficiently large, it can be passed through the 3-stage learning machinery to periodically augment the activity set. Since the abnormality database is populated by trajectories which did not fit any of the existing models, any consistent patterns extracted indicate new activities, e.g., a newly opened lane on a highway. In

this way, repetitive motions initially considered abnormal can be assimilated into the scene definition.

6.4 Discussion

The HMM-based formulation does not impose global temporal alignment, unlike other probabilistic spatio-temporal activity representations [15]. Instead, a local model of temporal structure is used which encodes the probability of time-scale distortion at any particular instance in time in the transition probability. The local model provides the flexibility to accurately compare similar, though different length, trajectories. But, because there is an underlying dynamic process, the exponential state duration model implied by the transition probabilities may not accurately match the perceived dynamics.

The hidden semi-Markov model (HSMM) has been proposed to explicitly model the state duration density [41]. Yet this is often ignored due to increased computational and memory requirements along with a need for significantly more data to train a larger number of parameters [38]. By modeling state durations, the path transitions can more closely match observed dynamics and more complex activities can be included (states with multimodal stay durations, e.g., straight through or stop at an intersection).

7 Activity Analysis

After the the offline learning process, the underlying activities present in a visual scene are compactly represented by the set of HMM paths. Using these learned models, the activity of a scene agent can be accurately characterized from live video. Activity analysis includes describing actions, predicting future behavior, and detection of abnormal and unusual events.

7.1 Trajectory Summarization

When a trajectory is completed (e.g., leaves the scene), a summary of the object activity is generated and it is determined to be either typical or abnormal.

7.1.1 Trajectory Classification

Using Bayesian inference, the activity that best describes the trajectory observation sequence is determined by maximum likelihood estimation

$$\Lambda^* = \arg\max_k P(F|\lambda_k). \tag{24}$$

The likelihood can be solved efficiently for the HMM using the forward-backward procedure [38] to indicate how likely is it that activity λ_k generated trajectory F. Partially observed activities (broken tracks from occlusion) can also be evaluated using (24) because the nonzero π_0 allows any initial start state.

7.1.2 Abnormal Trajectories

Since only typical trajectories are used to learn the activity paths, abnormalities (outliers) are not well modeled and the quality of class assignment Λ^* will be low. Trajectories with low log-likelihood $\log P(F|\Lambda^*) < LLT_{\Lambda^*}$ can be recognized as abnormalities. The decision threshold is learned during training by comparing the average likelihood of samples in training set D_k to those outside:

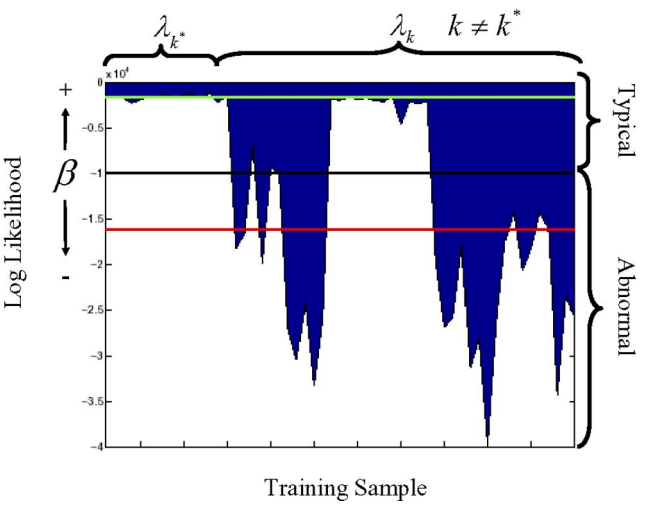


Fig. 8. A threshold is automatically set between the within class (green) and out of class (red) training examples with a tunable parameter β to control the sensitivity of abnormality detection.

$$LL_k^{in} = \frac{1}{|D_k|}\sum_{i\in D_k} \log P(F_i|\lambda_k), \tag{25}$$

$$LL_k^{out} = \frac{1}{N-|D_k|}\sum_{i\notin D_k} \log P(F_i|\lambda_k), \tag{26}$$

$$LLT_k = \beta\left(LL_k^{in} - LL_k^{out}\right) + LL_{out}. \tag{27}$$

The sensitivity factor $\beta \in [0, 1]$ controls the abnormality rate with larger β, causing more trajectories to be considered anomalous. The automatic threshold selection process is visualized in Fig. 8, where the green line signifies LL_k^{in}, red signifies LL_k^{out}, and black indicates the threshold LLT_k.

A few examples of abnormalities are displayed in Fig. 9. In Fig. 9a, although U-turns were allowed, the wide arc from the middle lane was not. The illegal loop in Fig. 9b and the off-road driving in Fig. 9c are clearly unusual, while the trajectory in Fig. 9d looks acceptable but actually came from travel in the wrong direction.

7.2 Online Tracking Analysis

Although it is interesting to provide a summary of completed tracks, it is often more important in surveillance to recognize activity as it occurs in order to assess and react to the current situation. A description of activity must be generated based on the most current information and refined with each new video frame. The online analysis engine is given the difficult tasks of making predictions with incomplete data (partial trajectory) and detecting unusual actions.

It is important to note that, within this framework, each individual scene agent is considered separately. Rather than just indicating a noteworthy event is happening, the location and individual of interest is highlighted as well.

7.2.1 Activity Prediction

Real-time analysis provides the alerts necessary for timely reaction. This response time could be improved if the

(a) (b) (c) (d)

Fig. 9. Abnormal trajectories (a) Wide u-turn. (b) Illegal loop. (c) Off-road driving. (d) Travel in the opposite direction.

monitoring system were able to infer intentions and determine what will happen before it actually occurs. Accurate prediction can help reduce reaction time or even avoid undesirable situations by providing a buffer to take countermeasures and corrective actions.

Future actions can be inferred from the current tracking information. Instead of using all of the tracking points accumulated up to time t, only a small window of data is utilized:

$$F_{w_c}^{w_p} = \{f_{t-w_c}, \ldots, f_{t-1}, f_t, \hat{f}_{t+1}, \ldots, \hat{f}_{t+w_p}\}. \tag{28}$$

The windowed track consists of w_c past measurements, the current point f_t, as well as w_p future points. The future points $\hat{f}_{t+\tau}$ are estimated by applying the tracking motion model τ time steps ahead. By utilizing only the windowed trajectory, only the recent history is considered during online evaluation because old samples may not correlate well with the current activity. This allows an agent to be monitored over very long time periods by discarding stale data.

The activity prediction is made at the current time t by evaluating (24) with F replaced by its windowed version:

$$\Lambda^*(t) = \arg\max_k P\left(F_{w_c}^{w_p} | \lambda_k\right). \tag{29}$$

This prediction has a longer time horizon than standard one step prediction (e.g., Kalman prediction) because it exploits previously observed activities rather than relying on a generic motion model. During complex maneuvers, motion models will rarely be effective much more than a few time steps into the future.

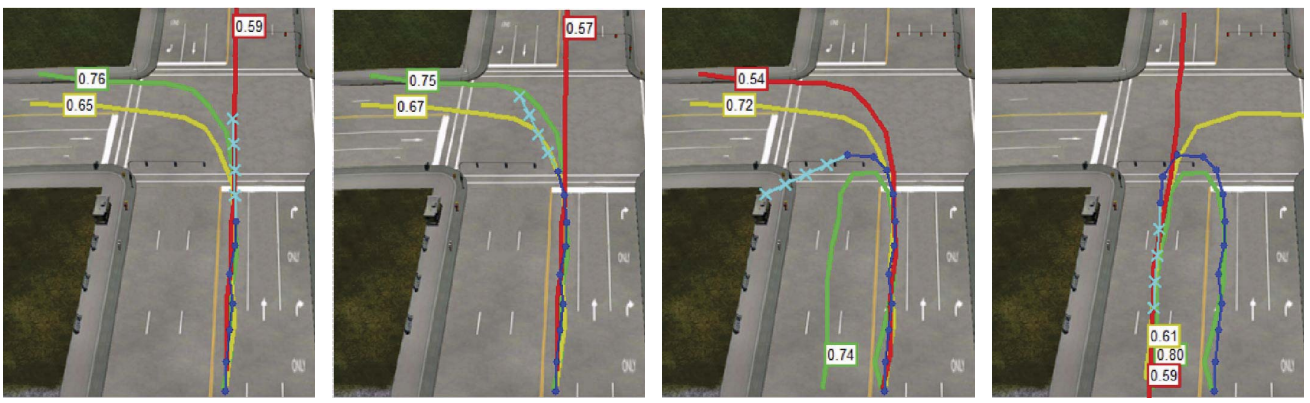
An example of the superiority of the trajectory pattern prediction is shown in Fig. 10. The top three best activity predictions are color coded as green for best match, yellow as top-2, and red as top-3 and their associated confidence is presented in the colored box. During the straight sections (Fig. 10a) the motion model, shown as light blue Xs, seems to estimate behavior. As the turn progresses in Fig. 10b, the motion model very poorly approximates the maneuver. It is quickly realized that the maneuver is a U-turn (Fig. 10c) since it was seen before while the motion model seriously falls behind. Finally, in Fig. 10d, the estimates realign in the straight section but the track memory helps distinguish the U-turn in green.

The prediction sequence which encodes an object's action history, $\{\Lambda^*(1), \ldots, \Lambda^*(t), \ldots, \Lambda^*(T)\}$, can indicate behavioral changes. A consistent activity manifests as consecutive labels which are equal, while a transition between labels indicates an activity switch (such as during a lane change).

7.2.2 Unusual Action Detection

Similarly to abnormal trajectories, unusual actions can be detected during tracking. These anomalies indicate deviations the instant they occur. Since only a windowed version of a track is used during tracking, the the log-likelihood threshold (27) needs to be adjusted. The new threshold is

$$\begin{aligned} LLT_k^t &= \gamma_k^t\left[\beta_t\left(LL_k^{in} - LL_k^{out}\right) + LL_k^{out}\right], \\ \gamma_k^t &= \frac{E[\#\{q\}|w_c]}{Q}. \end{aligned} \tag{30}$$

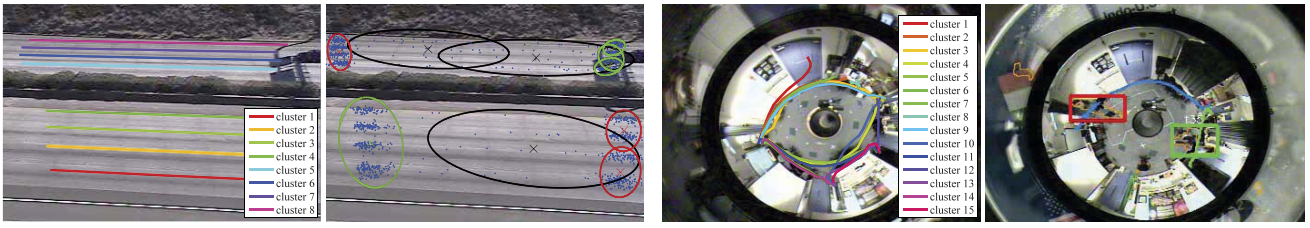


(a) (b) (c) (d)

Fig. 10. The top three {1 green, 2 yellow, and 3 red} most likely maneuvers and prediction confidence are shown during a U-turn. (a) The motion model (light blue X) fits during linear motion but (b) does not handle turns well. (c) Halfway through the u-turn, the prediction can accurately gauge the maneuver, while the motion model poorly approximates it. (d) New maneuvers explain the current situation but activity memory still indicates the U-turn in green.

(a) (b) (c)

Fig. 11. First occurrence of an unusual event during online processing. The red box indicates unusual detection, while green is deemed acceptable. (a) Illegal loop. (b) Off-road driving. (c) The illegal left turn from the middle lane is detected before realigning into the exit lane.



(a) (b) (c) (d)

Fig. 12. The other experimental data sets tested in addition to the intersection (CROSS). (a) The I5 highway traffic scene. (b) POI learning results with sparse mixtures due to broken trajectories. (c) OMNI camera inside a laboratory. (d) Analysis results show the predicted path and a red bounding box highlights an usual action from backtracking.

The abnormality threshold is adjusted with γ_t to account for the reduced probability mass associated with a partial trajectory. The term γ_t corresponds to the fraction of states visited in the evaluation window. This assumes uniform state duration for a full trajectory and averages the log-likelihood into each model state Q, while the numerator $E[\#\{q\}|w_c]$ is the expected number of states that will be visited in an observation window. The adjustment term is estimated as

$$\gamma_k^t \sim \frac{Qw_c/\bar{T}_k}{Q} = \frac{w_c}{\bar{T}_k}, \tag{31}$$

with $\bar{T}_k$ the average length of training trajectories in D_k corresponding to path λ_k. $\bar{T}_k/Q$ is the number of samples per state which results in $Qw_c/\bar{T}_k$ as the number of states in a window. Notice that the window only considers w_c and sets $w_p = 0$ to make assessments only on observed data. Here, $\beta_t \in [0, 1]$ is again chosen to ensure detection of most suspicious tracking points. In this work, β_t was automatically selected to return an unusual action detection rate of approximately 10 percent on the training set.

As soon as an object strays from an activity model, an unusual action alarm is triggered for timely detection. Examples of unusual actions are given in Fig. 11. The illegal loop and off-road driving from Fig. 9 are seen in Figs. 11a and b, respectively. The time evolution of a detection is shown in Fig. 11c, where initially a typical behavior is denoted with a green box but, in the middle of an illegal left turn from the middle lane, the box turns red to indicate an unusual action. Finally, a short time later, the action stabilizes back at green.

8 Experimental Studies and Analysis

The following section assess the performance of the activity learning and analysis framework. First, the three-staged learning process is evaluated by examining the quality of the automatically extracted activities. After, the accuracy of the trajectory summarization and online analysis modules are considered.

The results are compiled from a set of varying scenes. Three different experimental locations are analyzed: a simulated traffic intersection (CROSS), a highway road segment (I5) (Figs. 12a and 12b), and the interior of a laboratory under observation by an omnidirectional camera (OMNI) (Figs. 12c and 12d). Table 1 details the size of the data sets and experimental parameters. Further details about the experimental data sets can be found in [34]. The results of trajectory summarization and online analysis are presented in Tables 2 and 3, respectively.

TABLE 1
Experimental Parameters

	N	D	L	Q	w_c	w_p	β	β_t
CROSS	1900	1764	15	15	0.3	0.1	0.9	0.9
I5	606	261	15	15	0.2	0.1	0.9	0.95
OMNI1	117	55	15	25	0.3	0.1	0.98	0.75
OMNI2	131	98	15	25	0.3	0.1	0.9	0.75

TABLE 2
Trajectory Summarization Results

		classification		abnormalities		
	N_r	count	Acc %	count	TPR	FPR
CROSS	19	9191/9500	96.8%	163/200	81.5%	18.5
I5	8	879/923	95%	-	-	-
OMNI1	7	25/25	100%	10/16	68.8%	12.0
OMNI2	15	12/16	75.0%	15/18	83.3%	25.0

TABLE 3
Online Analysis Results

		prediction		unusual actions		
	N_r	count	Acc %	count	TPR	FPR
CROSS	19	31713/41871	75.7	364/443	0.822	0.324
I5	8	13859/14876	93.2	-	-	-
OMNI1	7	2241/3048	73.5	544/945	0.576	0.266
OMNI2	15	1719/2693	63.8	-	-	-

8.1 Cluster Validation

Perhaps the most important, and most difficult, task in trajectory learning is to robustly and accurately extract the dominant scene activities without a priori specification. This section compares different methods to estimate the number of routes a scene; this process is generally known as cluster validation.

In this work, the number of activities is automatically determined by first overclustering trajectories and then merging similar routes (Fig. 13). Two types of merge criteria are compared: membership and distance. The membership criteria utilizes the distribution of membership score u_{ik} to find similar clusters. The three membership criteria are the Bhattacharyya coefficient $BC(r_m, r_n)$, cosine angle $cos(r_m, r_n)$, and sum-of-squared difference $ssd(r_m, r_n)$:

$$BC(r_m, r_n) = \sum_{i=1}^{N} \sqrt{u_{ir_m} u_{ir_n}}, \tag{32}$$

$$cos(r_m, r_n) = \frac{\sum_{i=1}^{N} u_{ir_m} u_{ir_n}}{\sum_i \sqrt{u_{ir_m}^2} \sum_i \sqrt{u_{ir_n}^2}}, \tag{33}$$

$$ssd(r_m, r_n) = \sum_{i=1}^{N} \sqrt{(u_{ir_m} - u_{ir_n})^2}. \tag{34}$$

The route distance criteria measures, the match count (11), and total distance (13) were found to perform better than membership criteria (Fig. 13a).

Using the distance-based merge criterion, all 19 of the traffic intersections are correctly identified in the CROSS experiment. This set contained the most activities, but was distinct due to a favorable view. All eight of the highway lanes (Fig. 12a) were correctly identified in the I5 experiment as well, but there were also two false lanes. The extra lanes appeared in the southbound direction closest to the camera. Even though tracking errors were removed through POI learning (Fig. 12b), perspective distortion cause the northbound lanes to appear significantly closer than the southbound. By selecting the distance threshold ϵ_d that could resolve the north lanes, it allowed multiple responses in the closest lanes. This highlights the need for camera calibration to unify the measurement space. In the OMNI experiments, there are no physical lanes but virtual paths that people travel between doors and desks. The seven door-to-door paths were correctly extracted in the OMNI1 experiment. The second omni experiment was more complex and only 13 of 15 (Fig. 12c) paths were discovered. The two missing paths had very little support after trajectory filtering, but could be learned given more data. These OMNI experiments were much more difficult than the vehicle scenes because they are less constrained, paths do not have clear separation, and paths contain significant overlap.

The cluster validity evaluation shows the learning framework is able to accurately extract the typical scene activities with few parameters. The two parameters to specify for route clustering are the initial number of clusters K and ϵ_d. Although ϵ_d must be chosen, it is related to the desired spatial resolution between trajectories, and when doing spectral clustering with the merge process, the initial choice of K is quite forgiving (Figs. 13b and 13c).

8.2 Trajectory Summarization

After an object track finishes, trajectory summarization determines which activity likely generated the trajectory and also whether or not it is typical or something abnormal. In the following section, the quality of activity assignment and anomaly detection are evaluated, with results summarized in Table 2.

8.2.1 Trajectory Classification

Using (24), each test example is categorized by the activity that most likely generated the observation. The intersection experiment had 9,191 of 9,500 trajectories correctly labeled. The lane number for 879 of 923 manually labeled tracks

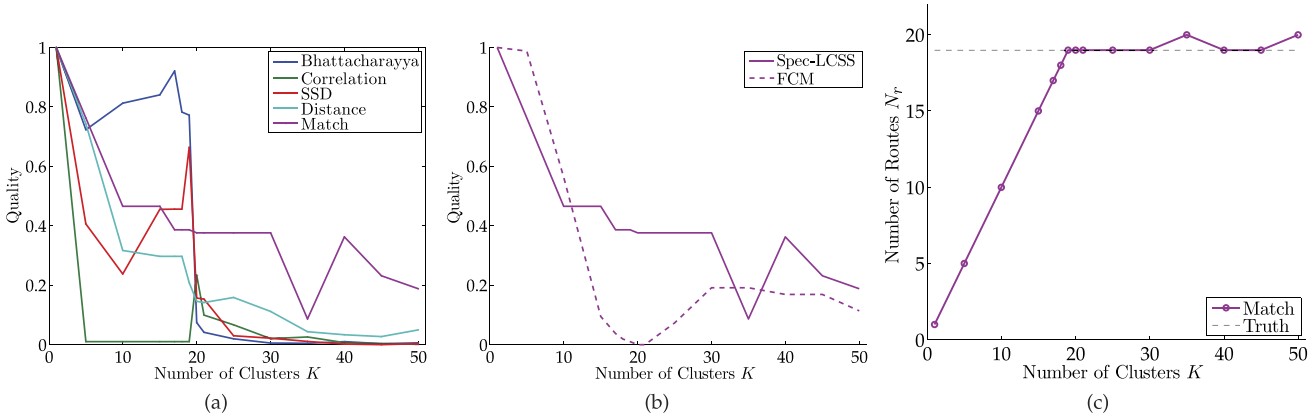


Fig. 13. (a) The point match criteria outperforms the distance and membership criteria for merging similar clusters. (b) Spectral clustering using LCSS distance has better merge performance than simply resampling tracks and FCM clustering. (c) The true number of clusters is consistently selected after the merge process. The route count N_r is automatically estimated as the mode across the range of match thresholds.

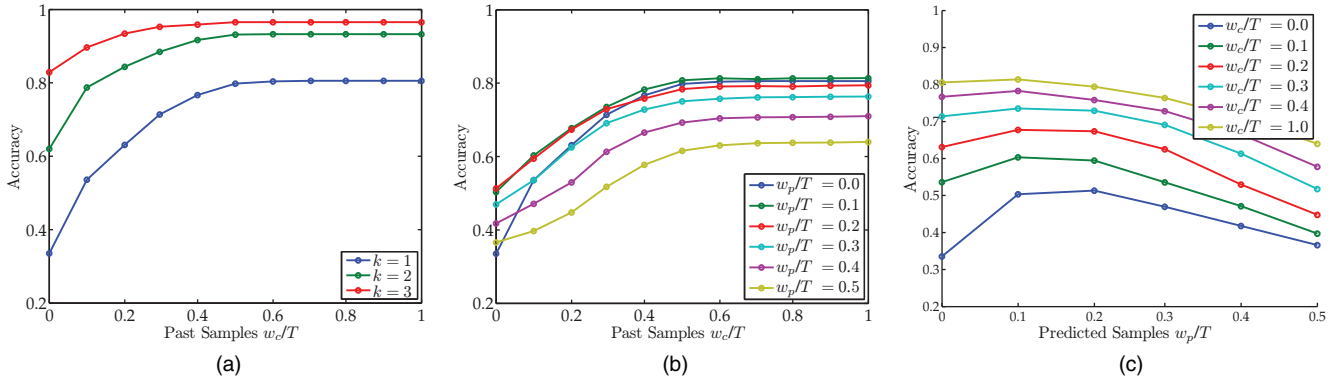


Fig. 14. OMNI1 online analysis results: (a) The true activity is often within the top-3 predictions. (b) Prediction accuracy increases with more data but saturates between 40-50 percent of track length. (c) Propagating the motion model forward a short amount of time improves prediction with the most benefit when utilizing fewer historical samples.

from I5 video was correctly determined for 95 percent accuracy. Not surprisingly, most classification errors occur in the northbound lanes (93 versus 98 percent southbound) where the lanes appear very close in the image plane. Since a trajectory was constructed from the centroid of a detection bounding box, large vehicles which cover multiple lanes had trajectories that would "float" above the true lane. In this situation, homography-based ground plane tracking would help keep trajectories consistent across detection sizes and preserve relative distances.

The test set for the OMNI1 database was collected over 24 hours on a single Saturday without participant awareness for natural tracks and all 25 of the test trajectories were correctly classified. The OMNI2 test set, in contrast, was choreographed during a 30 minute collection period. In this set only 12 of the 16 were correctly assigned to a path.

8.2.2 Abnormal Trajectories

Abnormal trajectories were determined using (27) with $\beta = 0.9$ for the CROSS test and a more lenient $\beta = 0.75$ for the OMNI1 test, reflecting the relative difficulty of the two data sets. Eighty-four percent of the CROSS abnormalities were identified with a 10 percent false positive rate. The OMNI1 and OMNI2 experiments did not perform quite as well. They had a 68.8 and 83.3 percent detection rate with 12.0 and 25.0 percent false positive rate (FPR), respectively. Fig. 9 shows abnormal trajectories in the CROSS set, while Fig. 16a shows an example from the OMNI scene.

8.3 Online Analysis

While trajectory summarization is useful for data compression and semantic query [37], it is an offline process that operates after the completion of an activity. In contrast, surveillance requires real-time up-to-date information to understand the monitoring situation. Online analysis must interpret agent behavior while under observation and express findings immediately. The following section gives the online performance. In particular, it examines how well an activity can be predicted using limited data and how often unusual actions are detected. The online analysis results are summarized in Table 3, which uses $w_c = 0.3\bar{T}$ and $w_p = 0.1\bar{T}$ in (28).

8.3.1 Activity Prediction

An emerging theme in intelligent monitoring is intention prediction. Rather than just explaining what has occurred, intent prediction tries to infer what an agent will do before it happens to provide time for preemptive acts

This section provides evaluation on how well activities can be predicted using the HMMs. The accuracy of prediction is measured at each time instant for every moving object in a scene. A correct prediction must have the same label at a time t as the true label of the full trajectory.

Looking at the prediction column in Table 3 we see that the highest prediction performance is for the I5 data set. This is not surprising because the lanes of the highway are straight and do not have any overlap. In contrast, the other data sets have 20-30 percent lower performance attributed to the overlap of activities. Using only a small window of data it is almost impossible to distinguish activities when segments are shared. At the start of the U-turn in Fig. 10, none of the top-3 predictions match so are considered incorrect even though it is not possible to make an accurate prediction with so little data.

When considering not just the best match, but the top three matches, as shown in Fig. 14a, the performance is significantly better. This shows that it is possible to have a good idea of the final activity, as demonstrated in the u-turn example (Fig. 10).

The effects effects of w_c and w_p when evaluating (29) are examined for the OMNI1 experiment in Fig. 14. As expected, the prediction accuracy improves given more data, since larger w_c values get closer to having the full trajectory, but, when w_c is between 40-50 percent performance saturates (Fig. 14b). In Fig. 14c, the effects of w_p are explored and indicate that, despite poor motion models, there is some benefit for small $w_p > 0$, especially at lower w_c values. But, w_p should not be too large since the predicted points will poorly represent the future activity far into the future.

The plots in Fig. 14 indicate it is best to only propagate the motion model forward for a few time steps and to use as much historical data as possible for accuracy. But, a balance must be made between accuracy and response delay when dealing with more past samples.

8.3.2 Unusual Action Detection

With unusual action detection, the moment of an abnormality is immediately indicated in real time (the exact time and location is highlighted by the red box seen in Fig. 12d). It is noted that these events tend to occur in groups of

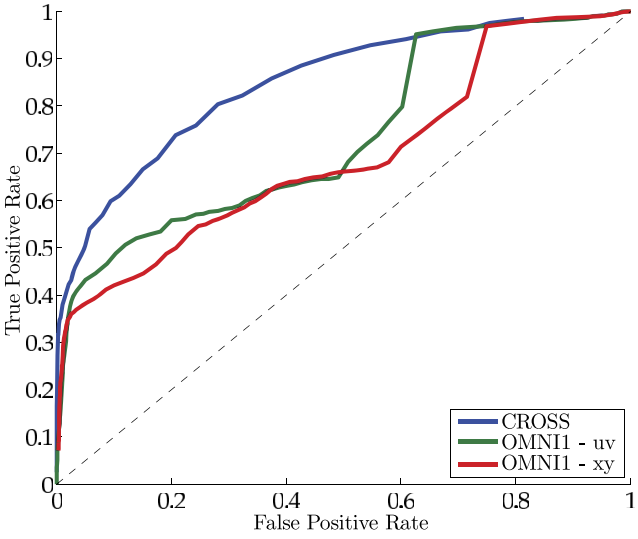


Fig. 15. Unusual action ROC for the CROSS and OMNI1 data sets. The OMNI1 experiment demonstrates significant room for improvement. Evaluation is difficult because of the large overlap between activities, which causes confusion when using only a small window of data.

successive points, where the number of points is directly proportional to the time duration of an unusual action.

The unusual action detection was performed for the CROSS and OMNI1 experiments only and each individual tracking point had to be manually labeled as abnormal or not. The unusual action column of Table 3 summarizes the detection performance on these sets. The simulated intersection had an 82.2 percent detection rate with a 32.4 percent FPR, while the OMNI1 experiment only had 57.6 percent TPR but at a lower 26.6 percent FPR. Unfortunately, the false positive rates were quite high.

Performance on the simulated data set was higher because the trajectories were more controlled. It was easy to indicate what was an abnormal trajectory point. In contrast, the natural tracks from the OMNI1 data set were given a best guess label. It was often very difficult to tell what exactly was unusual in this scene as there was little interframe motion and high overlap between routes. In addition, what might be considered abnormal given the final activity label might actually be quite typical given the best online match $\Lambda^*(t)$ based on just a window of data. For example, when moving in a direction, backtracking, then resuming forward, all of the backtrack was manually labeled as abnormal (since it is opposite the end maneuver), but during the backtrack, it is a typical action for another maneuver. An example of this behavior is demonstrated in Fig. 16b, where the red Xs correspond to the point of backtrack.

The online abnormality detection performance is better characterized in Fig. 15 by the receiver operating characteristics (ROC) curve for the two data sets. The CROSS curve looks promising and has decent detection (40-50 percent) even at very low FPR. In contrast, the real data collected in the OMNI1 set shows ample room for improvement. Work is needed to move the OMNI1 curve toward the CROSS ROC curve, which may require modification of the γ_k^t term. Notice that the curve associated with a velocity (dynamics) encoded path (OMNI1-uv) is slightly better than relying on just position (OMNI1-xy). The added velocity detail helps discriminate higher order effects.

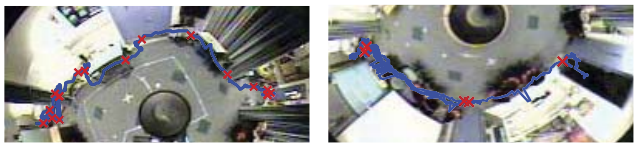


Fig. 16. Examples of automatically detected anomalies. (a) Offline: Walking along the walls of the OMNI1 room. (b) Online: Backtracking and reversing travel direction (red Xs indicate the point of unusual action).

8.4 Comparative Analysis

In order to validate our work we compare our system against the state of the art. We use the system presented by Hu et al. in 2006 [15] with their improved spectral clustering technique developed in 2007 [37]. The comparison looks at the quality of route extraction from clustering and activity recognition results on the CROSS data set.

In order to learn the routes, Hu et al. [15] defined trajectory similarity as the average distance between trajectory points (after resampling to a fixed number of points) and used spectral clustering. The optimal number of scene routes was automatically determined using the tightness and separation criterion (TSC). The plot in Fig. 17 shows the TSC values for different numbers of clusters K for the test sets. Note: A smaller TSC value indicates better clusters (tighter, more compact clusters with greater separation between different clusters). Although spectral clustering produces better separated routes than FCM, it is not readily apparent what the optimal choice for K should be. The red X indicates the true number of paths and, unfortunately, choosing the K value that minimizes the TSC does not always produce the correct number because of local minima. This is in contrast with Fig. 13c which demonstrates the stability of the match-based merge techniques.

In [15], the motion patterns are modeled as a chain of Gaussians very similar to an HMM. The main difference is time normalization is done probabilistically with the HMM, while the chain does this based on track length (dividing a trajectory into Q parts, with each part corresponding to a small set of points). Using the average distance between a trajectory and a chain, the likelihood of a path was found to behave as an exponential random variable. Their abnormal trajectory threshold was chosen as the minimal value in

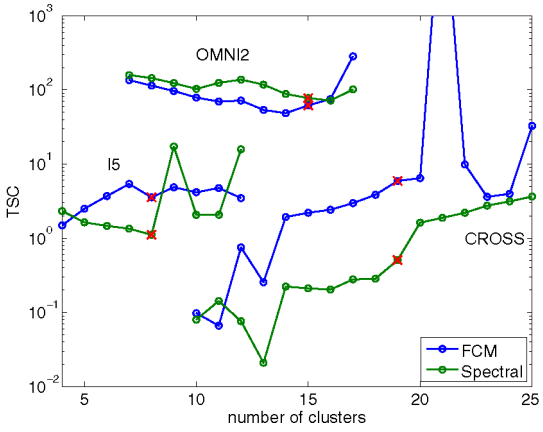


Fig. 17. The TSC cluster validity criterion used by Hu was not able to consistently elbow at N_r (denoted by the red X).

TABLE 4
Comparison with Hu [15] for the CROSS Data Set

	classify	abnormal (FPR)	predict
Hu [15]	98.01%	91.0% (23.3%)	69.3%
h CVRR	96.8%	81.5% (18.5%)	75.7%

each set D_k. Finally, they estimate the probability of an unusual point during live analysis by seeing how well the current tracking point is modeled by the best fit Gaussian state. Table 4 gives quantitative comparison between the two systems on the intersection CROSS data set.

Since a Gaussian chain and an HMM are very similar, the results for full trajectories are quite similar. Using the same abnormality threshold, Hu is able to detect more abnormal trajectories but with almost 5 percent higher FPR. In contrast to the summarization, the HMMs are much more effective during live analysis, both for prediction and unusual action detection. The HMM time normalization procedure is more robust than the length based normalization used by Hu. The ROC curve in Fig. 18 shows significant improvement when using a small window of data over just the current sample point. In fact, the lowest FPR possible with Hu's single point is approximately 30 percent and this requires β_t values up to five significant digits.

The comparative experiments suggest the three stage learning framework accurately specifies scene activities. Automatic selection of the number or routes is robustly handled through the match-based merge process and the HMM path models activity subtleties for robust online analysis.

9 Concluding Remarks

This paper has introduced a general framework for unsupervised visual scene description and live analysis, based on learning the repetitive structure inherent in motion trajectories. An activity vocabulary at differing resolutions is automatically generated through a 3-stage hierarchical learning process, which indicates important image points, connects them through spatial routes, and probabilistically models spatio-temporal dynamics with an HMM. The activity models adapt for long-term monitoring of changing environments. Comprehensive analysis, not present in previous literature, of three different experimental sites demonstrates the framework's ability to accurately predict future activity and detect abnormal actions in realtime.

Trajectory learning and analysis can be integrated into larger monitoring systems [42] to help focus and direct attention in multicamera setups. In addition, by utilizing only trajectory information, rather than higher resolution visual descriptors, the learning framework is general enough to be used for data exploration in alternate sensor modalities such as radar [43] without modification.

Acknowledgments

The authors would like to thank the reviewers for all their insightful comments, NSF-ITR and UC Discovery for support, and their CVRR colleagues for their continued assistance.

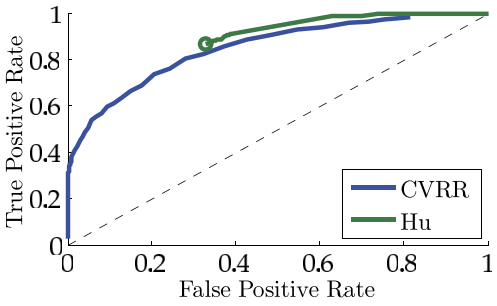


Fig. 18. The single tracking point abnormality detection technique developed by Hu resulted in a high false positive rate (even at the highest threshold).

References

[1] J.K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review," *Computer Vision and Image Understanding*, vol. 73, pp. 428-440, 1999.

[2] K. Huang and M. Trivedi, "3D Shape Context Based Gesture Analysis Integrated with Tracking Using Omni Video Array," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, June 2005.

[3] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247-2253, Dec. 2007.

[4] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Trans. Systems, Man, and Cybernetics Part C: Applications and Rev.*, vol. 37, no. 3, pp. 311-324, May 2007.

[5] M.M. Trivedi, T.L. Gandhi, and K.S. Huang, "Distributed Interactive Video Arrays for Event Capture and Enhanced Situational Awareness," *IEEE Intelligent Systems*, vol. 20, no. 5, pp. 58-66, Sept./Oct. 2005.

[6] W. Yan and D.A. Forsyth, "Learning the Behavior of Users in a Public Space through Video Tracking," *Proc. IEEE Workshop Applications of Computer Vision*, pp. 370-377, Jan. 2005.

[7] B.T. Morris and M.M. Trivedi, "Learning, Modeling, and Classification of Vehicle Track Patterns from Live Video," *IEEE Trans. Intelligent Transport Systems*, vol. 9, no. 3, pp. 425-437, Sept. 2008.

[8] S. Messelodi, C.M. Modena, and M. Zanin, "A Computer Vision System for the Detection and Classification of Vehicles at Urban Road Intersections," *Pattern Analysis and Applications*, vol. 8, nos. 1/2, pp. 17-31, Sept. 2005.

[9] D. Demirdjian, K. Tollmar, K. Koile, N. Checka, and T. Darrell, "Activity Maps for Location-Aware Computing," *Proc. IEEE Workshop Applications of Computer Vision*, pp. 70-75, Dec. 2002.

[10] N. Johnson and D. Hogg, "Learning the Distribution of Object Trajectories for Event Recognition," *Proc. British Conf. Machine Vision*, vol. 2, pp. 583-592, Sept. 1995.

[11] N. Sumpter and A.J. Bulpitt, "Learning Spatio-Temporal Patterns for Predicting Object Behavior," *Image and Vision Computing*, vol. 18, pp. 697-704, June 2000.

[12] J. Owens and A. Hunter, "Application of the Self-Organising Map to Trajectory Classification," *Proc. Third IEEE Int'l Workshop Visual Surveillance*, pp. 77-83, July 2000.

[13] W. Hu, D. Xie, T. Tan, and S. Maybank, "Learning Activity Patterns Using Fuzzy Self-Organizing Neural Network," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 3, pp. 1618-1626, June 2004.

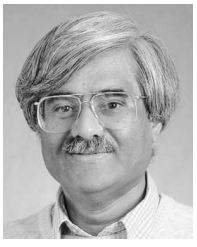
[14] I.N. Junejo, O. Javed, and M. Shah, "Multi Feature Path Modeling for Video Surveillance," *Proc. 17th Int'l Conf. Pattern Recognition*, pp. 716-719, Aug. 2004.

[15] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A System for Learning Statistical Motion Patterns," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 28, no. 9, pp. 1450-1464, Sept. 2006.

[16] B. Morris and M. Trivedi, "Learning and Classification of Trajectories in Dynamic Scenes: A General Framework for Live Video Analysis," *Proc. IEEE Int'l Conf. Advanced Video and Signal Based Surveillance*, pp. 154-161, Sept. 2008.

[17] D. Makris and T. Ellis, "Learning Semantic Scene Models from Observing Activity in Visual Surveillance," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 35, no. 3, pp. 397-408, June 2005.
[18] F.I. Bashir, A.A. Khokhar, and D. Schonfeld, "Object Trajectory-Based Activity Classification and Recognition Using Hidden Markov Models," *IEEE Trans. Image Processing,* vol. 16, no. 7, pp. 1912-1919, July 2007.
[19] C. Piciarelli and G.L. Foresti, "Online Trajectory Clustering for Anomalous Events Detection," *Pattern Recognition Letters,* vol. 27, no. 15, pp. 1835-1842, Nov. 2006.
[20] X. Wang, X. Ma, and E. Grimson, "Unsupervised Activity Perception by Hierarchical Bayesian Models," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, June 2007.
[21] X. Wang, K.T. Ma, G.-W. Ng, and W. Grimson, "Trajectory Analysis and Semantic Region Modeling Using a Nonparametric Bayesian Model," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, June 2008.
[22] X. Wang, X. Ma, and W. Grimson, "Unsupervised Activity Perception in Crowded and Complicated Scenes Using Hierarchical Bayesian Models," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 31, no. 3, pp. 539-555, Mar. 2009.
[23] T. Xiang and S. Gong, "Video Behavior Profiling for Anomaly Detection," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 30, no. 5, pp. 893-908, May 2008.
[24] B.T. Morris and M.M. Trivedi, "A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance," *IEEE Trans. Circuits and Systems Video Technology,* special issue on video surveillance, vol. 18, no. 8, pp. 1114-1127, Aug. 2008.
[25] F. Porikli, "Learning Object Trajectory Patterns by Spectral Clustering," *Proc. IEEE Int'l Conf. Multimedia and Expo,* vol. 2, pp. 1171-1174, June 2004.
[26] F. Bashir, W. Qu, A. Khokhar, and D. Schonfeld, "HMM-Based Motion Recognition System Using Segmented Pca," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 3, pp. 1288-1291, Sept. 2005.
[27] A. Naftel and S. Khalid, "Classifying Spatiotemporal Object Trajectories Using Unsupervised Learning in the Coefficient Feature Space," *Multimedia Systems,* vol. 12, no. 3, pp. 227-238, Dec. 2006.
[28] Z. Zhang, K. Huang, and T. Tan, "Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes," *Proc. IEEE Int'l Conf. Pattern Recognition,* pp. 1135-1138, 2006.
[29] S. Atev, O. Masoud, and N. Papanikolopoulos, "Learning Traffic Patterns at Intersections by Spectral Clustering of Motion Trajectories," *Proc. IEEE Conf. Intelligent Robots and Systems,* pp. 4851-4856, Oct. 2006.
[30] P. Dickinson and A. Hunter, "Using Inactivity to Detect Unusual Behavior," *Proc. IEEE Workshop Motion and Video Computing,* Jan. 2008.
[31] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc.,* vol. 39, pp. 1-38, 1977.
[32] M.A.T. Figueiredo and A.K. Jain, "Unsupervised Learning of Finite Mixture Models," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 24, no. 3, pp. 381-396, Mar. 2002.
[33] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Proc. Advances in Neural Information Processing Systems,* vol. 14, pp. 849-856, Sept. 2002.
[34] B. Morris and M. Trivedi, "Learning Trajectory Patterns by Clustering: Experimental Studies and Comparative Evaluation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 312-319, June 2009.
[35] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering Similar Multidimensional Trajectories," *Proc. IEEE Conf. Data Eng.,* pp. 673-684, Feb. 2002.
[36] W. Pedrycz, *Knowledge-Based Clustering: From Data to Information Granules.* John Wiley, 2005.
[37] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank, "Semantic-Based Surveillance Video Retrieval," *IEEE Trans. Image Processing,* vol. 16, no. 4, pp. 1168-1181, Apr. 2007.
[38] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition.* Prentice-Hall, 1993.
[39] E. Swears, A. Hoogs, and A. Perera, "Learning Motion Patterns in Surveillance Video Using HMM Clustering," *Proc. IEEE Workshop Motion and Video Computing,* pp. 1-8, Jan. 2008.
[40] M. Gales, D. Pye, and P. Woodland, "Variance Compensation within the MLLR Framework for Robust Speech Recognition and Speaker Adaptation," *Proc. IEEE Int'l Conf. Spoken Language,* pp. 1832-1835, Oct. 1996.
[41] M. Russell and R. Moore, "Explicit Modelling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* pp. 5-8, Apr. 1985.
[42] B.T. Morris and M.M. Trivedi, "Contextual Activity Visualization from Long-Term Video Observations," *IEEE Intelligent Systems,* special issue on intelligent monitoring of complex environments, vol. 25, no. 3, pp. 50-62, May/June 2010.
[43] B. Morris and M. Trivedi, "Unsupervised Learning of Motion Patterns of Rear Surrounding Vehicles," *Proc. IEEE Int'l Conf. Vehicular Electronics and Safety,* pp. 80-85, Nov. 2009.

Brendan Tran Morris received the BS degree from the University of California, Berkeley, and the PhD degree from the University of California, San Diego. He currently works as a postdoctoral researcher in the Computer Vision and Robotics Research Laboratory, where he has worked to develop VECTOR, a real-time highway transportation measurement system for congestion and environmental studies, and on-road behavior prediction for intelligent driver assistance systems. He was awarded the IEEE ITSS Best Dissertation Award in 2010. His research interests include unsupervised machine learning for recognizing and understanding activities, real-time measurement, monitoring, and analysis, and driver assistance and safety systems. He is a member of the IEEE.



Mohan Manubhai Trivedi received the BE (with honors) degree from the Birla Institute of Technology and Science, Pilani, India, and the PhD degree from Utah State University, Logan. He is a professor of electrical and computer engineering and the founding director of the Computer Vision and Robotics Research Laboratory, University of California, San Diego (UCSD). He and his team are currently pursuing research in machine and human perception, machine learning, distributed video systems, multimodal affect and gesture analysis, human-centered interfaces, and intelligent driver assistance systems. He regularly serves as a consultant to industry and government agencies in the US and abroad. He has given more than 65 keynote/plenary talks. He served as the editor-in-chief of the journal *Machine Vision and Applications*. He is currently an editor for the *IEEE Transactions on Intelligent Transportation Systems* and *Image and Vision Computing*. He served as the general chair for the IEEE Intelligent Vehicles Symposium IV 2010. His team also designed and deployed the "Eagle Eyes" system on the US-Mexico border in 2006. He served as a charter member and vice chair of the Executive Committee of the University of California system-wide UC Discovery Program. He served as an expert panelist for the Strategic Highway Research Program of the Transportation Research Board of the National Academies. He has received the Distinguished Alumnus Award from Utah State University, Pioneer Award and Meritorious Service Award from the IEEE Computer Society, and several "Best Paper" awards. Two of his students were awarded Best Dissertation Awards by the IEEE ITS Society (Dr. Shinko Cheng 2008 and Dr. Brendan Morris 2010). He is a fellow of the IEEE ("for contributions to Intelligent Transportation Systems field"), IAPR ("for contributions to vision systems for situational awareness and human-centered vehicle safety"), and the SPIE ("for contributions to the field of optical engineering").