Probabilistic Modeling of Scene Dynamics for Applications in Visual Surveillance

Imran Saleemi, Khurram Shafique, and Mubarak Shah

Abstract—We propose a novel method to model and learn the scene activity, observed by a static camera. The proposed model is very general and can be applied for solution of a variety of problems. The motion patterns of objects in the scene are modeled in the form of a multivariate non-parametric probability density function of spatio-temporal variables (object locations and transition times between them). Kernel Density Estimation is used to learn this model in a completely unsupervised fashion. Learning is accomplished by observing the trajectories of objects by a static camera over extended periods of time. It encodes the probabilistic nature of the behavior of moving objects in the scene and is useful for activity analysis applications, such as persistent tracking and anomalous motion detection. In addition, the model also captures salient scene features, such as, the areas of occlusion and most likely paths. Once the model is learned, we use a unified Markov Chain Monte-Carlo (MCMC) based framework for generating the most likely paths in the scene, improving foreground detection, persistent labelling of objects during tracking and deciding whether a given trajectory represents an anomaly to the observed motion patterns. Experiments with real world videos are reported which validate the proposed approach.

Index Terms—Vision and Scene Understanding, Machine Learning, Tracking, Markov Processes, Nonparametric statistics, Kernel Density Estimation, Metropolis Hastings, Markov Chain Monte Carlo.

1 Introduction

1.1 Problem Description

Recently, there is a major effort underway in the vision community to develop fully automated surveillance and monitoring systems [1], [2]. Such systems have the advantage of providing continuous 24 hour active warning capabilities and are especially useful in the areas of law enforcement, national defence, border control and airport security. The current systems are efficient and robust in their handling of common issues, such as illumination changes, shadows, short-term occlusions, weather conditions, and noise in the imaging process [3]. However, most of the current systems have short or no memory in terms of the observables in the scene. Due to this memory-less behavior, these systems lack the capability of learning the environment parameters and intelligent reasoning based on these parameters. Such learning, prior modeling, and reasoning is an important characteristic of all cognitive systems that increases the adaptability and thus the practicality of such systems. A number of studies have provided strong psychophysical evidence of the importance of prior knowledge and context for scene understanding in humans, such as, handling long term occlusions, detection of anomalous behavior, and even improving the existing low-level vision tasks of object detection and tracking [4], [5].

Recent works in the area of scene modeling are limited to the detection of entry and exit points and finding the likely paths in the scene [6], [7], [8], [9]. We argue that over the period of its operation, an intelligent tracking system should be able to model the scene from its observables and be able to improve its performance based on this prior model. The high-level knowledge necessary to make such inferences derives from domain knowledge, past experiences, as well as scene geometry, learned traffic and target behavior patterns in the area, etc. For example, consider a scene that contains bushes that only allow partial or no observation while the targets pass behind them. Most existing systems only detect the target when it comes out of the bushes and are unable to link the observations of the target before and after the long term occlusion. Given that this behavior of targets disappearing and appearing after a certain interval at a certain place is consistently observed, an intelligent system should be able to infer the correlation between these observations and to use it to correctly identify the targets at reappearance. We believe that the identification, modeling and analysis of target behavior in the scene is the key to achieving autonomous intelligent decision making capabilities. The work presented here is a step forward in this direction. Specifically, we present a framework to automatically learn a probabilistic model of the traffic patterns in a scene. We show that the proposed model can be applied towards various visual surveillance applications that include, behavior prediction, detection of anomalous patterns in the scene, improvement of foreground detection, and persistent tracking.

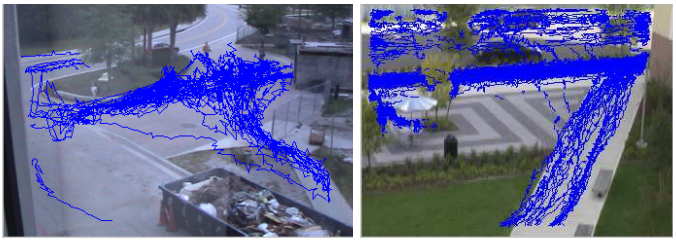
1.2 Proposed Approach

In this paper, we present a novel framework to learn traffic patterns in the form of a multivariate non-parametric

I. Saleemi and M. Shah are with University of Central Florida, Orlando FL. Email: imran@cs.ucf.edu, shah@eecs.ucf.edu. K. Shafique is with ObjectVideo Inc., Reston VA. Email: kshafique@objectvideo.com.

probability density function of spatio-temporal variables that include the locations of objects detected in the scene and their transition times from one location to another. The model is learned in a fully unsupervised fashion by observing the trajectories of objects by a static camera over extended periods of time. The model also captures salient scene features, such as, the usually adapted paths, frequently visited areas, occlusion areas, entry/exit points, etc.

We learn the scene model by using the observations of tracks over a long period of time. These tracks may have errors due to clutter and may also be broken due to short term and long term occlusions, however by observing enough tracks, one can get a fairly good understanding of the scene and infer above described scene properties and salient features. We assume that the tracks of the moving objects are available for training (We use KNIGHT system for generating these tracks [3]). Two scenes and the observed tracks are shown in Fig. 1. We use these tracks in a training phase to discover the correlation in the observations by learning the motion pattern model in the form of a multivariate probability density function (pdf) of spatio-temporal parameters (i.e., the joint probability density of pairs of observations of an object occurring within certain time intervals, $(x, y, x', y', \Delta t)$). Instead of imposing assumptions about the form of this pdf, we estimate the pdf using kernel density estimators. Each track on the image lattice can then be seen as a random walk where the probabilities of transition at each state of the walk is given by the learned spatio-temporal kernel density estimate. Once the model is learned, we use a unified Markov Chain Monte-Carlo (MCMC) sampling based scheme to generate the most likely paths in the scene, to decide whether a given path is an anomaly to the learned model, and to estimate the probability density of the next state of the random walk based on its previous states. The predictions based on the model are then used to improve the detection of foreground objects as well as to persistently track targets through short-term and long-term occlusions. We show quantitatively that the proposed system improves both the precision and recall of the foreground detection and can handle long term occlusions that cannot be handled using constant dynamics models commonly used in the literature. We also show that the proposed algorithm can handle occlusions that were not observed during training. The examples of these type of occlusions include vehicles parked in the scene after training, or a bug sitting on the camera lens.



(a) (b)

Fig. 1. Two scenes used for the testing of proposed framework, with some of the tracks observed by the tracker during training.

2 Related Work

Trajectory and path modeling is an important step in various applications, many of which are crucial to surveillance and monitoring systems. Such models can be used to filter tracking algorithms, generate likely paths, find locations of sources and sinks in the scene, and detect anomalous tracks, etc. This kind of modeling can be directly used as a feedback to the initial stages of the tracking algorithm, and applied to solve short and long term occlusions. In recent years, a number of different methods and features for trajectory and path modeling of traffic in the scene have been proposed. These methods differ by their choice of features, models, learning algorithms, applications, and training data. A detailed review of these models is presented in [10]. We now describe some of these methods.

Neural Network based approaches for learning of typical paths and trajectory modeling are proposed in [11], [12], [13]. Other than the computational complexity and lack of adaptability of Neural Networks, a major disadvantage of these methods is their inability to handle incomplete and partial trajectories. Fernyhough et al. [14] use the spatial model presented in [15] as a basis for a learning algorithm that can automatically learn object paths by accumulating the traces of targets. Koller-Meier et al. [16] use a node-based model to represent the average of trajectory clusters. A similar technique is suggested by Lou et al. [17]. Although both methods successfully identify the mean of common trajectory patterns, no explicit information is derived regarding the distribution of trajectories around the mean. A hierarchical clustering of trajectories is proposed in [18], [19], where trajectories are represented as a sequence of states in a quantized 6D space for trajectory classification and the method is based on a co-occurrence matrix that assumes that all trajectory sequences are equally long. However, this assumption is usually not true in real sequences.

Detection of sources and sinks in a scene as a pre-step allows robust tracking. In [20], a Hidden Markov Model based scheme of learning sources and sinks is presented, where all sequences are two-state long. The knowledge of sources and sinks is used to correct and stitch tracks in a closed-loop manner. Another HMM-based approach is to model trajectories as transitions between states representing Gaussian distributions on the 2D image plane [21], [22]. Galata et al. [23] propose a mechanism that automatically acquires stochastic models of any behavior. Unlike HMMs, the proposed variable length markov model can capture dependencies that may have a variable time scale.

Another body of work for modeling motion patterns in a scene lies in the literature of multi-camera tracking [9], [24], [25], where the objective is to use these models to solve for hand-off between cameras. The motion patterns are modeled only between the field of views of cameras. These models can be seen as special cases of the proposed model, which is richer in the sense that it captures the motion pattern both in visible and occluded regions. This richness of the proposed model allows it to handle dynamic occlusions, such as person to person occlusions as well as occlusions that occur after the model is learned. Tekalp[26] uses Bayesian networks for tracking and occlusion reasoning across calibrated cameras with overlapping views, where sparse motion estimation and appearance are used as features. Tieu et al.[6] infer the topology of a network of non-overlapping cameras by computing the statistical dependence between observations in different cameras.

Another interesting piece of work in this area is by Ellis et al.[7] who determined the topology of a camera network by using a two stage algorithm. First the entry and exit zones of each camera are determined using an Expectation-Minimization technique to cluster the start and end points of object tracks. The links between these zones across cameras are then found using the co-occurrence of entry and exit events. The proposed method assumes that correct correspondences will cluster in the feature space (location and time) while the wrong correspondences will generally be scattered across the feature space. Stauffer [8] proposes an improved linking method that tests the hypothesis that the correlation between exit and entry events is similar to the expected correlation when there are no valid transitions. This allows the algorithm to handle the cases where exit-entrance events may be correlated, but the correlation is not due to valid object transitions. Both of these methods assume a fixed set of entry and exit locations after initial training and hence cannot deal with newly formed occlusions without retraining of the model.

Hoiem et. al. [27] take a step forward in image and scene understanding and proposed improvement in object recognition by modeling the correlation between objects, surface geometry and camera viewpoints. Rosales et al. [28] estimate motion trajectories using Extended Kalman Filter to enable improved tracking before, during and after occlusions. Kaucic et al. [29] present a modular framework that handles tracking through occlusions and the blind regions between cameras by the initialization, tracking and linking of high-confidence smaller track sections. Wang et al. [30] propose measures for similarity between tracks that take into account spatial distribution, velocity, and object size. The trajectories are then clustered based on object type and its spatial and velocity distributions. Perera et. al. [31] present a method to reliably link track segments during the linking stage. Splitting and merging of tracks is handled to achieve multi-object tracking through occlusion. Recently, Hu et al. [32] have presented an algorithm for learning motion patterns where foreground pixels are first clustered using fuzzy K-means algorithm and trajectories are then hierarchically clustered based on the results of previous step. Trajectory clustering is performed in two layers for spatial and temporal based clustering. Each pattern of clustered trajectories is then assumed to be a link in a chain of gaussian distributions, the parameters for which are estimated using features of each clustered trajectory. Results of experiments on anomaly detection and behavior prediction are reported.

As opposed to explicitly modeling trajectories or smaller sections of trajectories of objects in the scene, we propose modeling of joint distribution of initial and final locations of every possible transition and the transition times. Our approach is original in the following ways:

- *We propose a novel motion model that not only learns the scene semantics but also the behavior of traffic through arbitrary paths. This learning is not limited like other approaches that work best with well defined paths like roads and walkways.*
- *The learning is accomplished using a joint five dimensional model unlike pixel-wise models and mixture or chain models. The proposed model represents the joint probability of a transition from any point in the image to any other point, and the time taken to complete that transition.*
- *The temporal dimension of traffic patterns is explicitly modeled, and is included in the feature vector, thus enabling us to distinguish patterns of activity that correspond to the same trajectory cluster but have high deviation in the temporal dimension. This is a more generalized method as compared to modeling pixel-wise velocities.*
- *Instead of fitting parametric models to the data, we propose the idea of learning tracks information using Kernel Density Estimation. It allows for a richer model and the density retained at each point in the feature space accurately reflects the training data.*
- *Rather than exhaustively searching for predictions in the feature space based on their probabilities, we propose to use stochastic methods to sample from the learned distribution and use it as prediction with a computed probability. Sampling is thus used as the process propagation function in our state estimation framework.*
- *Unlike most of the previous work reported in this section, which is targeted towards one or two similar applications, we apply the proposed probabilistic framework to solve a variety of problems that are commonly encountered in surveillance and scene analysis.*

3 Probabilistic Traffic Model

We now propose a model that learns traffic patterns as joint probability of the initial and final locations, and duration of object transitions and describe our method of learning and sampling from the distribution. In this section, we discuss the feature to be learned and explain how it is computed. The sampling process and its use in the state estimation framework is also explained.

3.1 Learning the Transition Distribution using KDE

We use the surveillance and monitoring system, KNIGHT[3] to collect tracking data. This data consists of time stamps and object labels with locations of their centroid for each frame of video. KNIGHT assigns a unique label to each object present in the scene and attempts to persistently track the object using a joint motion and appearance model. A frame of video may contain multiple objects. Given this data, we associate an observation vector $O(x, y, t, l)$ with each detected object. In this vector, x and y are the spatial coordinates of the object centroid on image lattice, t is the time instant at which O is observed (accurate to a millisecond) and l is the label for object in O as per tracking. We seek to build a five dimensional estimate of the transition probability density function $p\,(X, Y, \tau)$, where X and Y are the initial and final states representing the object centroid locations in $2d$ image coordinates and τ is the time taken to make the transition from X to Y in milliseconds.

In the proposed work, the analysis is performed on a feature space where the n transitions are represented by $\mathbf{z}_i \in \mathbb{R}^5$, $i = \{1, 2, \ldots, n\}$. The vector $\mathbf{z}$ consists of a pair of $2d$ locations of the centroid of object before and after transition, and time taken to execute the transition. Any two observations O_i and O_j in frames i and j respectively, comprise a feature point $Z(x_i, y_i, x_j, y_j, t_j - t_i)$ for the probability estimate if they satisfy the following:

- $0 < t_j - t_i = \tau < T$, thus the implicit assumption is that O_i and O_j are uncorrelated if occurring simultaneously or beyond a time interval of T. For our experiments, we chose $T = 5000$ms.
- $l_i = l_j$, both O_i and O_j are the observations of the same object as per the tracker's labelling.
- If $l_j \notin L_k$, where L_k is the set of all objects (labels) in frame k, then *all* labels in frame k make valid data points with l_j provided $0 < \tau < T$.

Kernel Density Estimation ([33], [34]) is used to estimate the pdf $p\,(X, Y, \tau)$ non-parametrically. Suppose we have a sample consisting of n, d dimensional, data points $\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_n}$ from a multi-variate distribution $p(\mathbf{z})$, then an estimate $\hat{p}(\mathbf{z})$ of the density at $\mathbf{z}$ can be computed using

$$\hat{p}(\mathbf{z}) = \frac{1}{n}|\mathbf{H}|^{-\frac{1}{2}} \sum_{i=1}^{n} K(\mathbf{H}^{-\frac{1}{2}}(\mathbf{z} - \mathbf{z_i})), \tag{1}$$

where the d variate kernel $K(\mathbf{z})$ is a bounded function satisfying $\int K(\mathbf{z})dz = 1$, $K(\mathbf{z}) = K(\mathbf{-z})$, $\int \mathbf{z}K(\mathbf{z})dz = 0$, $\int \mathbf{z}\mathbf{z^T}K(\mathbf{z})dz = \mathbf{I_d}$ and $\mathbf{H}$ is the symmetric positive definite $d \times d$ bandwidth matrix. The multivariate kernel $K(\mathbf{z})$ can be generated from a product of symmetric univariate kernel K_u, i.e.,

$$K(\mathbf{z}) = \prod_{j=1}^{d} K_u(\mathbf{z}_{\{j\}}). \tag{2}$$

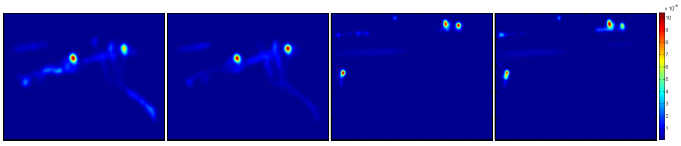
The selection of the kernel bandwidth $\mathbf{H}$ is the single most important criterion in kernel density estimation. Asymptotically, the selected bandwidth $\mathbf{H}$ does not affect the estimate but in practice sample sizes are limited. Theoretically, the ideal or optimal $\mathbf{H}$ that balances the bias and variance globally can be computed by minimizing the mean-squared error, $MSE\{\hat{p}_{\mathbf{H}}(\mathbf{z})\} = E\{[\hat{p}_{\mathbf{H}}(\mathbf{z}) - p_{\mathbf{H}}(\mathbf{z})]^2\}$, where $\hat{p}$ is the estimated density, p is the true density, and the subscript $\mathbf{H}$ indicates the use of the bandwidth $\mathbf{H}$ in computing the density. However, the true density p is not known in practice. Instead, various heuristic approaches have been proposed for finding $\mathbf{H}$ (See [35] for a survey of these approaches). We use the *Rule of Thumb* method [36], which is a fast standard deviation based method that uses the Asymptotic Mean Integrated Squared error (AMISE) criterion,

$$AMISE\{\mathbf{H}\} = \frac{1}{n}R(K)|\mathbf{H}|^{-\frac{1}{2}} + \int_{R^5} [(K_{\mathbf{H}} * p)(\mathbf{z}) - p(\mathbf{z})]^2 dz, \tag{3}$$

where $R(K) = \int_{R^5} K(\mathbf{z})^2 dz$ and $*$ is the convolution operator. The data-driven bandwidth selector is then, $\mathbf{H} = \arg\min_{\mathbf{H}} \widehat{AMISE}\{\mathbf{H}\}$. To reduce the complexity, $\mathbf{H}$ is assumed to be a diagonal matrix, i.e., $\mathbf{H} = diag[h_1^2, h_2^2, \ldots, h_d^2]$. We use a product of univariate Gaussian kernels to generate $K(\mathbf{z})$, i.e.,

$$K_u(z_{\{j\}}) = \frac{1}{h_j\sqrt{2\pi}} exp\left(-\frac{z_{\{j\}}^2}{2{h_j}^2}\right), \tag{4}$$

where, h_j is the j^{th} non-zero element of $\mathbf{H}$, and K_u is centered at $z_{\{j\}}$. It is emphasized here, that using a Gaussian kernel does not make any assumption on the scatter of data in the feature space. The kernel function only defines the effective region in $5d$ space in which each data point will have an influence while computing the probability estimate. Each time a pair of observations satisfying the described criteria are available during training, the observed feature $\mathbf{z}$ is added to the sample. $p(X, Y, \tau)$ then gives the probability of a transition from point X to point Y in τ milliseconds.



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

Fig. 2. These maps represent the marginal probability of any object, (a) reaching each point in the image, and (b) starting from each point in the image, in any arbitrary duration of time for the scene in Fig. 1a. (c) and (d) show similar maps for the scene in Fig. 1b.

Fig. 2 illustrates the maps of $p(X, Y, \tau)$ marginalized over $2d$ vectors X and Y, i.e., Fig. 2a represents the probability of an object reaching a point in the map from any location, i.e., $\int_X \int_\tau p(X, Y, \tau)\; d\tau dX$. It is important to note that the areas with higher densities (brighter colors) cannot be labelled as entry or exit points. The intensity at a point in the image only illustrates the

marginal probability that an observed state transition has, of *starting* or *ending* at that point. The dominant points in the maps only mean that a higher number of object observations were made at these points. For example, in practice, this could possibly mean that these areas are places where people stand or sit while exhibiting nominal motion but significant enough to be noticed by the tracker. Similarly, each point in Fig. 2b represents the probability of an object starting from that particular point and ending anywhere in the image, i.e., $\int_Y \int_\tau p(X, Y, \tau)\ d\tau dY$. Notice that the similarity in the two maps manifests a correlation between incoming and outgoing object transitions at the same locations. It should also be pointed out that since $p(X, Y, \tau)$ is a high dimensional space, the probabilities from a particular location to another location in a specific interval can be very low and a high dynamic range is required to display these maps correctly. So the dark blue regions in these maps are not necessarily absolutely zero. Many dark blue regions in these maps will have relatively higher probabilities of transition, if the pdf is marginalized over specific time intervals or a specific starting or ending point, as discussed later. Figures 2c and 2d show similar maps for the scene in Figure 1b.

The probability density function $p(X, Y, \tau)$ now represents the distribution of transitions from any point X in the scene to any other point Y given the time taken to complete the transition as τ. Our next step is to be able to predict the *next* state of an object, given the *current* state and an approximate time duration of the jump. This is achieved by sampling the marginal of this distribution function as described in the following subsection.

3.2 Construction of Predicted Trajectories

The learned model can be used to find a likely next state given the current state of an object. The motivation is that we want to construct a likely trajectory from the current object position onwards, which would act as the predicted behavior of the object. Instead of finding the probabilities of transitions to different locations, we use the learned distribution as a generative model and sample feature points from the model to construct a sequence of tracks based on the training observations.

Once the distribution has been learned, each track Ω can be viewed as a Markov process, $\Psi_k : \Omega \longrightarrow I$, on the image lattice I, where $k \in \{1, 2, \ldots\}$. By marginalizing out τ or integrating $p(X, Y, \tau)$ over an appropriate value of τ, and factoring by $p(X)$, the learned distribution can then be seen as the transition probability model, i.e., $\hat{p}\left(X_{t+1}|X_t\right)$, where X_t is the current state and X_{t+1} is the next state. Fig. 3 illustrates the marginal probability of reaching any state Y in the image starting from state X = G, in any arbitrary duration of time, and can be written as $\int_\tau p(X = G, Y, \tau)\ d\tau$. The relatively bright spots further away show the possible states that a series of transitions can lead to. These correspond to higher values of τ, the time dimension of the distribution, for which the probability of transition from G is obviously low. These maps do not represent the whole scene, but just small regions of it.

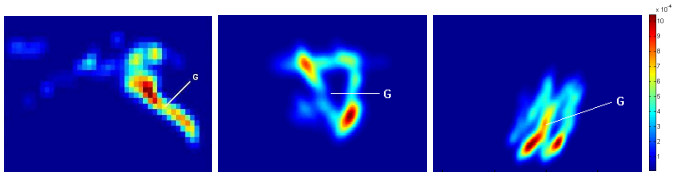


Fig. 3. Regions of maps that represent the probability of reaching any state in the image from the state G.

To compute a next state given the current state, we propose sampling from the learned transition distribution. Instead of sampling only one prediction, we propose sampling multiple distinct future states from the distribution so that the multi-modal nature of the learned density is fully utilized.

In real world scenarios, the assumption that tracks are first order Markov processes is not usually valid. It can however be noted that in our scene traffic model, the choice of different values of τ (the time duration required to complete a transition), can be used to achieve higher order Markov chains. Furthermore, while dealing with applications such as track filtering, anomaly detection and tracking through occlusions, it must be noted that sampling high probability future states is not always enough, for example in human traffic scenarios where it is possible for people to take the less travelled but possible paths, which we do not want to ignore in our algorithm. So instead of only one prediction for next location, multiple distinct future states are sampled from the distribution. The track is initialized with any location

Initialize X_0; *set* $i = 0$.
Repeat,
Let τ *be the time elapsed between* X_{i-1} *and* X_i.
Repeat for each sample u_{i-1}^k, *where* $k \in \{1, 2, \ldots, N\}$,
- *Let* $\acute{Y} \sim q(.|\theta_{i-1}, \tau)$, *q being the Gaussian distribution* $\mathcal{N}(\theta_{i-1}, \Sigma)$, *where* θ_{i-1} *is the mean and covariance matrix,* $\Sigma = diag[\tau^2, \tau^2]$.
- *Let* $r \sim U(0, 1)$.
- *Let* $\hat{p}(\acute{Y}) = \int_X \int_\tau p(X, Y = \acute{Y}, \tau)\ d\tau dX$ *and* $\hat{p}(u_{i-1}^k) = \int_X \int_\tau p(X, Y = u_{i-1}^k, \tau)\ d\tau dX$.
- *Compute* $\alpha(u_{i-1}^k, \acute{Y}, \tau) = \min\left(1, \frac{\hat{p}(\acute{Y}) q(u_{i-1}^k|\acute{Y}, \tau)}{\hat{p}(u_{i-1}^k) q(\acute{Y}|u_{i-1}^k, \tau)}\right)$.
- *If* $r \leq \alpha(u_{i-1}^k, \acute{Y}, \tau)$, *set* $u_i^k = \acute{Y}$, *otherwise, set* $u_i^k = u_{i-1}^k$.
Set $X_i = \sum_{m=1}^N u_i^m w_i^m$, *where* w_i^m *is the weight associated with* u_i^m.
Set $i = i + 1$.

Fig. 4. Metropolis Hastings Sampling. See section 6 for details about the choice of q, the proposal density.

and the representative future states are initially chosen

with uniform distribution in neighborhood of the current state with equal weights. At any given step the weighted mean of all these will give the next candidate state. The transitions of this random walk will not always be the most likely ones but rather possible ones, that are in accordance with the training data. A sample set containing N future states is sampled from the distribution to represent $P(X_0)$ using the algorithm described in Fig. 4. These samples can be written as, $\left\{\left(u_0^{k,-}, w_0^{k,-}\right)\right\}$, where $u_0^{k,-} \sim \mathcal{N}(X_0, \Sigma)$, $w_0^{k,-} = \frac{1}{N}$, N is the number of possible future states, $\Sigma = diag[\tau_{max}^2, \tau_{max}^2]$, and $k = \{1, \ldots, N\}$. τ_{max} is the maximum possible time, which we seek to sample transitions from. The value of τ_{max} is set to the average duration in which a person takes a step. This will correspond to maximum time allowed for a sampled transition to occur. The symbols '$-$' and '$+$' indicate the representation of the state before and after the measurement has arrived respectively. Since only N states are used to represent the whole distribution given X_0 as starting state, weights $w^{1,\ldots,k}$ are normalized such that $\sum_i w_i^{k,+} = 1$, thus $w_0^{k,-} = \frac{1}{N}$, initially. At each step i, we have the representation of the probability density function $P(X_i|X_0, \ldots, X_{i-1})$ in the form of a set of future states $\left\{\left(u_i^{k,-}, w_i^{k,-}\right)\right\}$, where $u_i^{k,-} = f\left(u_{i-1}^{k,+}\right)$, and $w_i^{k,-} = w_{i-1}^{k,+}$, f is a function that samples from state transition distribution according to the conditional probability $P(.|u_{i-1}^{k,+})$ using the Metropolis-Hastings algorithm, which is described in detail in Fig. 4. Given this representation of $P(X_i|X_0, \ldots, X_{i-1})$, we update it to $\left\{\left(u_i^{k,+}, w_i^{k,+}\right)\right\}$ as, $u_i^{k,+} = u_i^{k,-}$ and,

$$w_i^{k,+} = w_i^{k,-} \int_0^{\tau_{\max}} p\left(X_{i-1}, u_i^{k,-}, \tau\right) d\tau. \tag{5}$$

An estimate of X_i is now given by the weighted mean of all the sampled future states as, $X_i = \sum_{m=1}^{N} u_i^{m,+} w_i^{m,+}$, where X_i serves as the distribution's prediction until the measurement from the tracker arrives which is denoted by ω_i (omega, not to be confused with w_i, which is the weight of a particular sample). The update step then involves the computation of the final estimate of the object's new location, θ_i as a weighted mean of X_i and ω_i as,

$$\theta_i = \frac{X_i p(X_i|\theta_{i-1}) + \omega_i p(\omega_i|\theta_{i-1})}{p(X_i|\theta_{i-1}) + p(\omega_i|\theta_{i-1})}, \tag{6}$$

where the probabilities of transition from the previous state estimate θ_{i-1} to the new potential states X_i and ω_i, serve as the weights.

After update, the weights are normalized again so that $\sum_i w_i^{k,+} = 1$ and their variance σ_w^2 is calculated. If $\sigma_w^2 > \sigma_{th}^2$, a predefined maximum threshold, then a new set of samples is constructed by drawing (sampling) without replacement from the distribution using θ_i as current position. The probability of the sample being drawn is assigned as the initial weights. Thus the predicted states with relatively lower weights are discarded and others are retained to represent the next state. This process is continued to propagate the trajectory.

The choice of $\tau_{\max}$ depends on desired maximum duration between two consecutive points on a track. Assuming first order Markov process in a human traffic environment, $\tau_{\max}$ corresponds to the average time taken by a person to take one step. For most of the experiments reported in this paper, the value of $\tau_{\max}$ was chosen to be the inverse of the frame rate of the tracker. The importance of this value is illustrated in Fig. 5. Different values of τ can be used to specify the time interval in transition from X to Y. Fig. 5a, 5b and 5c depict the marginal probabilities of an object starting from G shown by a $*$, and reaching any state in the image in 0-1.5, 1.5-3 and 3-5 seconds respectively. These maps can be represented mathematically as $\int_0^{1500} p(X = G, Y, \tau) d\tau$, $\int_{1500}^{3000} p(X = G, Y, \tau) d\tau$, and $\int_{3000}^{5000} p(X = G, Y, \tau) d\tau$. The maps show the paths that objects usually follow in these areas. So the time dimension of the learned distribution encodes additional information that is vital to solving short-term occlusions using the distribution as described in Section 4.4. Notice that the modes or the areas of dominant probability in general are not equidistant from G. Furthermore, the locations of these modes and their distances from G can be different for different choices of G depending on the usual speeds of objects in that area, which is in contrast to tracking using Gaussian or other symmetric distributions as process noise functions.

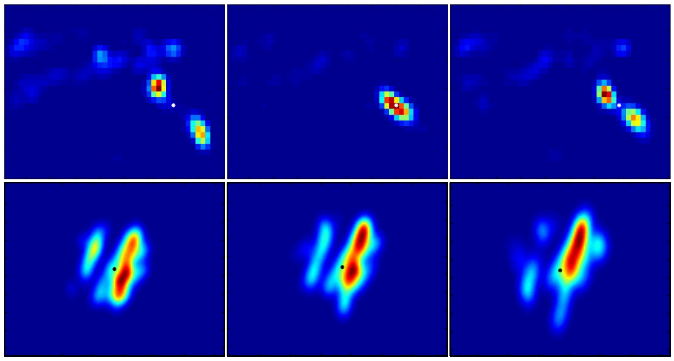
We now have a mechanism of sampling from the distribution that enables us to predict object behavior in space and time, which is then used for likely (typical) path generation, anomaly detection, and persistent tracking of objects through occlusions as described in the next section. The learned transition probabilities are also used to improve Foreground detection.

4 Applications of Proposed Model

It is important that motion patterns of objects be modeled in a way that enables solution of multiple problems that are encountered in scene analysis and surveillance. We now describe some of these problems and application of the state estimation framework for their solution.

4.1 Generating Likely Paths

An important aspect of modeling traffic patterns is generation of likely paths. Given the current location of an object, such a simulated path amounts to a prediction of future behavior of the object. We seek to sample from the learned model of transition patterns to generate behavior predictions. We expect that only a few number of paths should adequately reflect observations of trajectories through walkways and roads, etc. The sampling method is used as described in section 3.2 except that there are no measurements available which translates



(a) (b) (c)

Fig. 5. Marginal probability maps of the learned pdf at different time intervals. The maps represent the probability of reaching any state in the image from the state G (shown by white $*$ in first and black $*$ in second row) in (a) 0-1.5 seconds, (b) 1.5-3 seconds (c) 3-5 seconds. Each row shows a different starting state in the image. These maps show only small regions of the actual scene.

to acceptance of the weighted mean of all samples at each step as the next state. More specifically, equation 6 transforms to $\theta_i = X_i$. For our experiments we start by manually choosing X_0 as the initial state in a region of significant traffic. The trajectory is propagated by accepting $predicted$ states as $next$ states. The chain is stopped whenever $\sigma_w^2 > \sigma_{th}^2$, that is, there is a need to resample candidate predictions or when the Metropolis-Hastings algorithm chain fails to move for more than a specified number of iterations, e.g., we used a maximum of 200 iterations for this purpose.

4.2 Improvement of Foreground Detection

The intensity difference of objects from the background has been a widely used criterion for object detection, but it can be noted that temporal persistence is also an intrinsic property of the foreground objects, i.e. unless an object exits from the scene or becomes occluded, it has to either stay at the same place or move to a location within the spatial vicinity of the current observation. Since our proposed transition model incorporates the probabilities of movement of objects from one location to another, it can be used to improve the foreground models. We now present the formulation for this application. It should be noted however that this method alone cannot be used to model the foreground. Instead it needs to be used in conjunction with an appearance based model like mixture of Gaussians [19]. We seek to find the likelihood of a pixel u belonging to the foreground at time instant t. Let $\mathbf{u}$ be a random variable which is true if and only if the pixel u is a foreground pixel. Also, let Λ be the feature set used to model the background (for example, color/gray scale in appearance based model), and $\Phi = \{\phi_1, \phi_2, \ldots \phi_Q\}$ be the set of pixels detected as foreground at time instant $t-1$, where Q is the total number of the foreground pixels in the previous frame. Then according to Bayes rule,

$$P(\mathbf{u} = true|\Lambda, \Phi) \propto P(\Lambda, \Phi|\mathbf{u} = true)P(\mathbf{u} = true). \quad (7)$$

Assuming independence of appearance Λ and the foreground history Φ, we may write,

$$P(\mathbf{u} = true|\Lambda, \Phi) \propto P(\Lambda|\mathbf{u} = true)P(\Phi|\mathbf{u} = true)P(\mathbf{u} = true). \quad (8)$$

Here we use the framework of aggregating expert opinions [37] to combine the decisions from different features. Specifically, we consider logarithmic opinion pooling [38], i.e.,

$$P(\Lambda, \Phi|\mathbf{u} = true) \propto P(\Phi|\mathbf{u} = true)^{\beta} P(\Lambda|\mathbf{u} = true)^{1-\beta}. \quad (9)$$

The weight β represents the expert's reliability. We chose $\beta = \frac{Q}{Q+Q_a}$, where Q is the number of pixels classified as foreground in the previous frame, while Q_a is the number of foreground pixels detected in the last frame using only appearance based model.

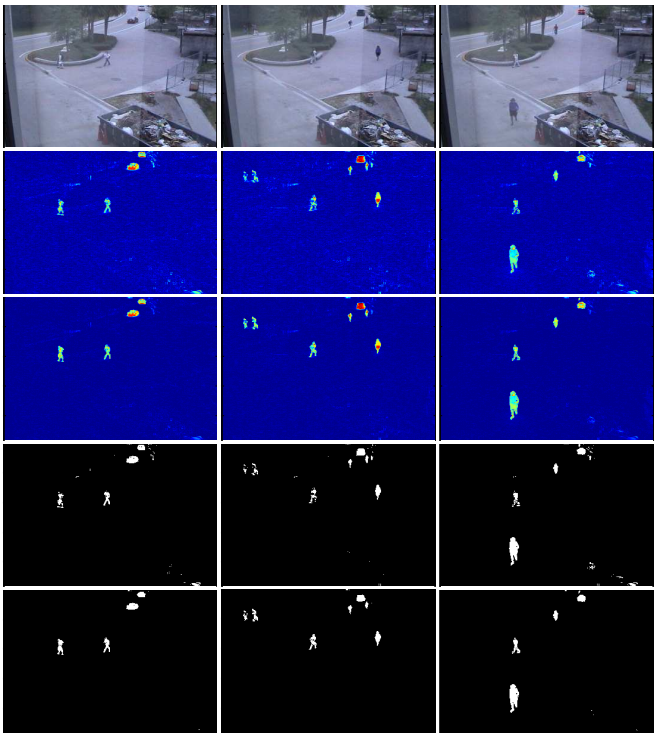


Fig. 6. Foreground Modeling: Each column shows an example of improvement on foreground probability estimation. (Row 1) original images, (Row 2) probability maps using only the mixture of Gaussians method and, (Row 3) foreground probability maps using proposed model in conjunction with the mixture of Gaussians model. Rows 4 and 5 show foreground masks obtained using the maps in rows 2 and 3 respectively.

The KNIGHT object detection and tracking system ([3]) is used to extract appearance based likelihood, $P(\Lambda|\mathbf{u} = true)$. KNIGHT performs background subtraction at multiple levels using statistical models of gradient and color data. Specifically, a mixture of Gaussians

model is used with 4 Gaussians per pixel. The learned pdf $p(X, Y, \tau)$ is used to estimate the remaining terms as follows:

$$P(\Phi|\mathbf{u} = true) = \sum_{j=1}^{Q} p(\phi_j, u, \Delta t), \tag{10}$$

and the marginal (prior) of u being a foreground pixel is,

$$P(\mathbf{u} = true) = \int_X \int_\tau p(X, Y = u, \tau) d\tau dX. \tag{11}$$

The value of Δt represents the length of time between two consecutive frames. It should be pointed out here that although not all previous foreground pixels should contribute significantly to the pixel u, the value of Δt is typically small enough to inhibit contributions from far away pixels. In other words, only the pixels in Φ, spatially close to u, chip in substantial values to the sum.

Fig. 6 shows the results of the proposed foreground model. It can be seen that the number of false positives has decreased and the true silhouettes are much more pronounced. The scheme presents a simple and effective way of reducing errors and improving background subtraction. Many of the commonly encountered false detections like motion of static background objects are removed using this approach. More experiments and a quantitative analysis are reported in Section 5.

4.3 Anomaly Detection

If tracking data used to model the state transition distribution spans sufficiently large periods of time, it is obvious that a sampling algorithm will not be able to sample a track that is anomalous considering the usual patterns of activity and motion in that scene. This observation forms the basis of our anomaly detection algorithm. The anomaly detection algorithm generates its own predictions for future states using the method described in section 3.2 without using the current observation of the tracker. It then compares the actual measurements of objects with the predicted tracks and computes a difference measure between them. Let the set of predicted states of an object be Θ and the actual measurements as observed by the tracking algorithm be Ω. Using the framework as described earlier, we can compute $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{i+1}\}$, which is the predicted or estimated state. Notice that at step i, we have a prediction for location at step $i + 1$. Then the observed track described by $\Omega = \{\omega_1, \omega_2, \ldots, \omega_i\}$ is labelled as anomalous if, $d_i > d_{th}$, where,

$$d_i = \frac{1}{n+1} \sum_{j=i-n}^{i} \left((\omega_j - \theta_j)^T \Sigma_\Theta^{-1} (\omega_j - \theta_j) \right)^{\frac{1}{2}}, \tag{12}$$

where Σ_Θ is covariance matrix of x and y coordinates of all the potential (candidate) next states, n is the number of previous states included in calculation of Mahalanobis distance, and d_{th} is a predefined maximum difference between observed and predicted tracks. Furthermore, the plot of unnormalized distances between observed anomalous and predicted trajectories versus the number of states, n, is sublinear with respect to n.

This proposed approach is sufficient to find a sequence of transitions significantly different than the predictions from the state transition distribution, and can easily identify an anomalous event in terms of motion patterns. Using this formulation trajectories that are spatially incoherent, or temporally inconsistent with normal behavior can be identified, e.g., presence of objects in unusual areas or significant speed variation respectively. Increasing the value of n (Eq. 12) helps detect suspicious behavior where activity is not necessarily unusual in either spatial or temporal domains but the object has been in view for a long time and error (distance between actual and predicted paths) has accumulated and crossed the d_{th} barrier. Hence, this approach is able to handle several different kinds of anomalous behavior.

4.4 Persistent Tracking through Occlusions

Finally, we present application of the proposed scene model for persistent tracking through occlusions and completion of the missing tracks. Persistent tracking requires modeling of spatio-temporal and appearance properties of the targets. Traditionally, parametric motion models such as, constant velocity or constant acceleration, are used to enforce spatio-temporal constraints. These models usually fail when the paths adapted by objects are arbitrary. The proposed model of learning traffic parameters is able to handle these shortcomings when occlusions are not permanently present in the scene and the patterns of motion through these occlusions have previously been learned, e.g., person to person occlusions, large objects like vehicles that hide smaller moving objects from view. We use the proposed distribution to describe a solution to these problems.

Let $\hat{p}(Y|X) = \frac{\int_\tau p(X,Y,\tau) d\tau}{\int_\tau \int_Y p(X,Y,\tau) dY d\tau}$ and Ω be an observed track. A sample set containing N samples is generated from the learned pdf to represent $\hat{p}(X_0)$ where X_0 is the initial location of the track. These samples are represented as $\left\{ \left(u_0^{k,-}, w_0^{k,-} \right) \right\}$, where $u_0^{k,-} \sim \mathcal{N}(X_0, \Sigma)$, $\mathcal{N}$ represents Gaussian distribution with mean X_0 and covariance matrix, $\Sigma = diag[\tau^2, \tau^2]$, and $k = \{1, \ldots, N\}$. For the initial set of samples, the weight is assigned as, $w_0^{k,-} = \frac{\int_X \hat{P}(u_0^{k,-}|X) dX}{P_\gamma(\Gamma = u_0^{k,-})}$, where, Γ is a random variable that represents Gaussian distribution in the neighborhood of X_0. The symbols '$-$' and '$+$' indicate the representation of the state before and after availability of the measurement respectively. Since only N samples are used to represent the whole distribution, weights $w^{\{1,\ldots,k\}}$ are normalized such that $\sum_i w_i^{k,+} = 1$. At each step i, a representation of the probability density function $\hat{p}(X_i|\omega_0, \omega_1, \ldots, \omega_{i-1})$ is retained in the form of weighted candidate samples $\left\{ \left(u_i^{k,-}, w_i^{k,-} \right) \right\}$, where $u_i^{k,-} = f\left(u_{i-1}^{k,+} \right)$ and $w_i^{k,-} = w_{i-1}^{k,+}$. f is a function that samples from the state transition distribution according to the conditional probability $\hat{P}(.|u_{i-1}^{k,+})$ and the smooth-

ness of motion constraint using the Metropolis-Hastings algorithm, which is described in detail in Fig. 4.

At any time, we want to be able to find correspondences between successive instances of objects. This task proves to be very difficult when the color and shape distributions of objects are very similar or when the objects are in very close proximity of each other, including in person to person occlusion scenarios. However, in the proposed model, a hierarchical use of transition density enables our scheme to assign larger weights to more likely paths. Assume now that the observed measurements of m objects at time instant i are given by $\Omega = \{\omega_i^1, \omega_i^2, \ldots, \omega_i^m\}$, and the predicted states of previously observed s objects are $\Theta = \{\theta_i^1, \theta_i^2, \ldots, \theta_i^s\}$ respectively. We must find the mapping function from the set $\{\Omega_i^k\}$, $1 \leq k \leq m$ to the set $\{\Theta_i^l\}$, $1 \leq l \leq s$. To establish this mapping function, we use the graph based algorithm proposed in [39]. The weights of the edges corresponding to the mapping Θ_i^l and Ω_i^k are given by the Mahalanobis distance between the estimated position θ_i^l and the observed position ω_i^k, for each l and k. Once the correspondences are established, for each object, given the representation of $\hat{p}\left(\theta_i|\theta_0, \ldots, \theta_{i-1}\right)$, and the corresponding new observation ω_i, we update it to $\left\{\left(u_i^{k,+}, w_i^{k,+}\right)\right\}$ as $u_i^{k,+} = u_i^{k,-}$ and,

$$w_i^{k,+} = w_i^{k,-} \sum_{k=1}^{N} \hat{P}\left(\omega_i|u_i^{k,-}\right) = \prod_{j=0}^{i} \sum_{k=1}^{N} \hat{P}\left(\omega_j|u_j^{k,-}\right). \tag{13}$$

The product of old weight with the sum of probabilities of reaching ω_i from each sample state $u_i^{k,-}$ for all k, translates into a product of transition probabilities for all jumps taken by an object. This results into assignment of larger weights for higher probability tracks and in the next time step, this weight helps establish correspondences such that the probability of an object's track is maximized. Once updated, candidate locations are re-sampled as described in section 3.2. This approach enables simultaneous solution of short-term, and person to person occlusions, as well as persistent labeling.

5 Results

In this section, we present our experimental results. The collected tracking data is derived from two video sequences taken at the scenes shown in Fig. 1. Some figures related to these datasets and their density estimates are summarized in Fig. 7. The video sequences were acquired by static cameras, and the scenes consist of high human traffic with occlusions and both commonly and sparingly used paths. Initial tracking is done using the algorithm in [3]. As can be seen in Fig. 1, there are numerous entry and exit points in the scenes and multiple paths can be chosen between different pairs of entry and exit points. This section is organized into subsections showing results for each of the applications.

Sequence	Scene 1	Scene 2
Resolution	360 x 240	320 x 240
Duration	3 days (approx.)	6 hr 20 min
Observations	3.9 million	253,697
Samples	936,739	66,498
Learning Time	194 hrs (approx.)	13 hrs (approx.)

Fig. 7. Datasets: Observations list the number of times any object was detected and feature samples are the number of 5d features computed and added to the density estimate (regardless of resampling). Learning Time is the time taken to compute the samples and generate their kernel density estimate.

Fig. 8. Examples of likely paths generated by the proposed algorithm using Metropolis-Hastings sampling. Tracks are initialized from random locations selected manually.

5.1 Likely Paths Generation

As described in Section 4.1, starting at random initial states in the image, sampling from the distribution gives possible paths that are usually followed by the traffic. Fig. 8 shows some of these random walks. It should be noted that no observations from the tracking algorithm have been used in this experiment. The likely paths generated are purely simulated based on the learned distribution. It can however be seen in Fig. 8 that some tracks are not very smooth and loop at some locations. But we want to point out here that it is not our purpose to generate smooth paths, rather a depiction of actual tracking scenarios. A comparison of Fig. 8 with figure 1b, that shows actual observed tracks used during training, proves the similarity of both and validates the proposed model and sampling method.

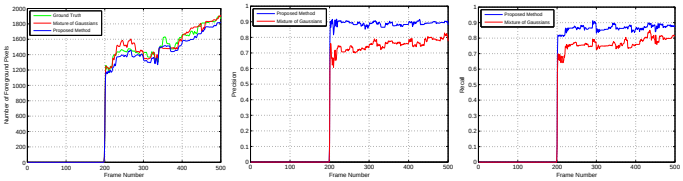
5.2 Foreground Detection

A scheme for improving detection of foreground objects using the proposed model was described in section 4.2. Fig. 9 presents the results of that scheme. The top row of columns (a) and (b) show images that represent the results of the object detection algorithm of [3]. It can be seen in these images that parts of trees, garbage dump, and grass have been detected as objects due to nominal motion owing of wind. The false detections are shown by red bounding boxes. On the other hand, during the training phase, there were very few object detections in these areas, thus giving low probability of object presence at these locations. The bottom row of columns (a) and (b) show the results after mis-detections have been removed using the combination of appearance and proposed motion model. The scheme has removed these false positives owing to their low probability of

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

Fig. 9. Foreground Modeling Results: Columns (a) and (b) show results of object detection . The images in top row are without and bottom row are with using the proposed model. Green and red bounding boxes show true and false detections respectively. (c) and (d) show results of blob tracking. Red tracks are original broken tracks and black ones are after improved foreground modeling.

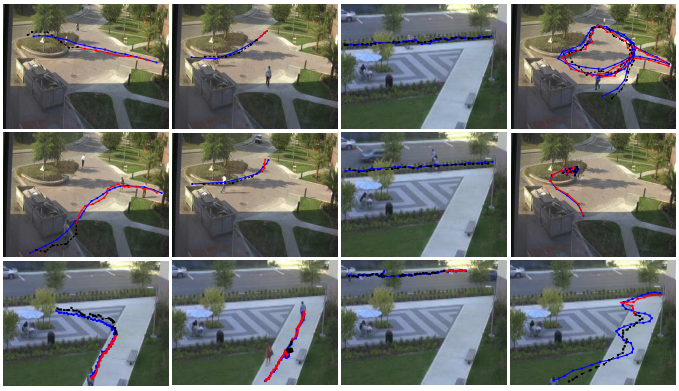
transition or existence or both. Columns (c) and (d) show tracks from the tracking algorithm of [3] which are broken in different places because of a significant number of false negatives in foreground detection. Again these missed detections were corrected after incorporating the motion based foreground model.



(a) (b) (c)

Fig. 10. Foreground Modeling Results: (a) Number of foreground pixels detected by the mixture of Gaussians method, the logarithmic pooling method, and the ground truth. The pixel-level foreground detection recall and precision using the mixture of Gaussians approach only, and the proposed formulation are shown in (b) and (c) respectively.

For a quantitative analysis of the foreground modeling scheme, we manually segmented a 500 frames video sequence into foreground and background regions. The first 200 frames were used to initialize the mixture of Gaussians based background model. Foreground was estimated on the rest of the frames using, i) mixture of Gaussians model, and ii) the proposed logarithmic pooling of appearance and motion models. Pixel level precision and recall were computed for both cases. The results of this experiment are shown in Fig. 10. It can be seen from the plots that both the precision and recall of the combined model are consistently better than mixture of Gaussians approach alone. The number of false positives for the combined model was reduced by 79.65% on average. The average reduction in the number of false negatives was 58.27%.



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

Fig. 11. Results of Anomaly detection: (a) Spatially anomalous, (b) and (c) Temporally anomalous, and (d) Suspicious behavior due to presence over large distance or extended period. Blue track represents the actual (observed) track. Red and black tracks correspond to typical and atypical (anomalous) predicted paths respectively.

5.3 Anomaly Detection

Fig. 11 shows results of our anomaly detection algorithm described in Section 4.3. The normal tracks adapted in each scene can be seen in Fig. 1 and Fig. 8. In the top row of Fig. 11a, a person walks through the paved region and then into the grass. The algorithm separates this trajectory into typical and atypical parts. The blue track shown in the figure is the track of concern, red track is the area where the prediction follows observations while dotted black shows the prediction once an anomaly is detected. Notice how the blue track is closely followed by the red one as long as it is typical behavior. At the time of training, the dump trailer was at a different position as seen in Fig. 1a, resulting in classification of the track observed in middle row of Fig. 11a as anomalous. The bottom row of Fig. 11a shows another spatially incoherent track through an area where no motion is usually observed.

The top row of Fig. 11b shows anomaly due to the unusual speed of a person riding a bicycle in an area where people usually only walk. The middle row of Fig. 11b shows the results of our algorithm on another example where a person is running in the same paved area. Again, this behavior has not been observed in the tracks used for training because people usually do not run in this area. The bottom row of Fig. 11b shows a stopping action where a person sits down to tie their shoelaces. Three more examples of the second type of anomaly (temporal inconsistency) are shown in Fig. 11c. The top and middle rows in the third column show two persons skating and riding a bicycle respectively while the bottom row shows a person walking on the road where usually only cars are observed. Since the speeds of objects in Fig. 11b and Fig. 11c are significantly different from the average speed of objects during training, the prediction either lags behind as compared to the actual

measurements from the tracker, or hurries ahead resulting in the observed trajectory being labelled anomalous.

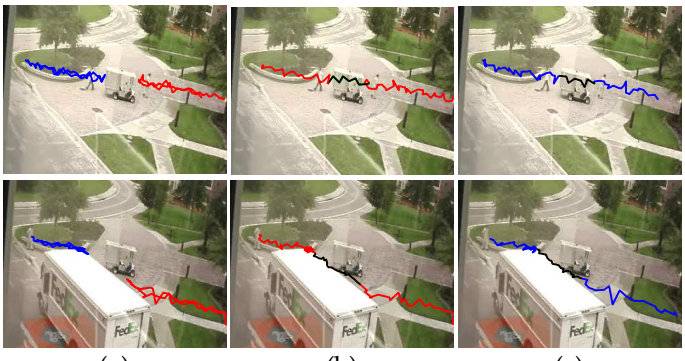
Three examples of the third type of anomalies are shown in Fig. 11d. In the top row of Fig. 11d a person is walking slowly in a circular path for a few minutes. The motion pattern itself is not anomalous to the learned distribution, but the presence of the object in the scene for a relatively longer period of time results in accumulation of error, calculated as d_i using larger values of n in Eq. 12, and eventually becomes greater than d_{th} resulting in classification of the sequence as an anomaly. The middle row of Fig. 11d shows a person jumping over an area which people usually only sit on. Even though the person is not present in the scene for very long, the error d_i quickly becomes significant due to the incoherence of his actions relative to the training examples. The bottom row shows a a zigzag pattern of walking, something not observed before, resulting in the classification of the trajectory as anomalous.

Since the decision as to whether a given trajectory is anomalous is subjective and differs from one observer to another, we asked three human observers to classify 19 sequences as either normal or anomalous. The proposed method of anomaly detection was run on these 19 sequences which labelled 14 of them as anomalous. The quantitative results of these experiments are summarized in Fig. 12.

Ground Truth	Human 1	Human 2	Human 3
Anomalous	13	15	11
Normal	6	4	8
Precision	92.86	100	78.57
Recall	100	93.34	100

Fig. 12. Quantitative analysis of anomaly detection results using classification by 3 human observers as the ground truth.

5.4 Persistent Tracking



(a) (b) (c)

Fig. 13. For each row, (a) shows the observed tracks in blue and red that have been labelled wrong, and (b) and (c) show the stitched part of tracks in black, and actual tracks in red and blue respectively.

In Section 4.4, we described our method to track objects persistently through occlusions. In absence of measurements, our algorithm propagates the track based on samples from the distribution. Fig. 13 shows two examples of tracks that have undergone erroneous labeling by the tracking algorithm because both of them had considerable missing portions due to total occlusion. In the top row of Fig. 13, the tracker assumes that two objects walked towards the golf car and then returned to their original locations, which is not correct. Each of these tracks are propagated through predicted locations to join the correct track. The result of propagation based on weighted mean of these samples is shown in top row of Fig. 13b and Fig.13c, where both tracks are separately shown for clarity. Red and blue colors show two different tracks and the black color shows the part where trajectory has been stitched. In the bottom row of Fig. 13, a truck is obstructing the camera's view.

(a) (b)

Fig. 14. Example of persistent tracking for multiple simultaneous objects with overlapping or intersecting tracks undergoing occlusion. (a) Actual original tracks (ground truth) (b) Broken tracks due to simulated occlusion shown as black region.

Another example of persistent tracking through occlusions is shown in Figs. 14 and 15, where we created a considerably large simulated occlusion in a small video sequence as shown in Fig. 14b as a black box in the middle of the image. The location of this occlusion makes the task challenging since most of the likely paths in the scene pass through this region. This sequence contains four tracks that have significant portions under occlusion. For comparison with other algorithms, we also tested Kalman filter based tracking on the same sequence where the process noise function is assumed to be Gaussian, and trajectories are assumed to be locally linear, as in [28]. The results on the four tracks are shown separately in Fig. 15. The direction of movement in all tracks is from the top of the image downwards except in Fig. 15c where it is from left to right. The results of Kalman filter tracking is shown as white track, the proposed approach as yellow and the ground truth as green tracks. The ground truth trajectories are available since the occlusion is only simulated. The reason for the sharp turns at the end of the predicted tracks is the recovery from error, once the object comes out of occlusion. It is fairly obvious that Kalman filter based tracks tend to follow linear constant motion dynamics, which actually works better than our algorithm in Fig. 15b, but in the other three tracks, the proposed approach works

significantly better than the Kalman filter. The reason for the deviation of the track generated using motion patterns, from the ground truth (in Fig. 15b) is that the length of the track $before$ the occlusion is very small. As a result, the algorithm is unable to differentiate between the actual track that is coming from the top, and a track which goes from that position to the left; hence, the predicted track goes left, as the small track length makes it seem that the object's direction is towards the left.

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

Fig. 15. Results for scenario shown in figure 14. Green track is the ground truth. Tracking through occlusion using Kalman filter is shown in white and yellow tracks are generated using the proposed approach. Notice that both methods can recover well once the measurement is available again, but during occlusion the proposed method stays closer to ground truth.

Track	Fig. 15a	Fig. 15b	Fig. 15c	Fig. 15d
Proposed Approach	74.34	119.20	142.74	52.06
Kalman Filter	165.22	59.14	472.08	628.69
% Reduction	55.01%	-50.39%	69.76%	91.72%

Fig. 16. Mahalanobis distances of tracks generated using proposed approach and kalman filter based tracking, from the ground truth.

Finally, we report some of the run time performance data. All the experiments were ran on a Pentium-4 2.53 GHz machine and the coding was done in Matlab without any optimizations for speed. Fig. 17 compares the performance of kernel density estimation to a histogram approximation in terms of different metrics.

Metric	KDE	Histogram	% Improvement
a	0.422	1.451	243.84 %
b	11.72 %	6.93 %	40.87 %
c	35.78	10.07	71.86 %
d	0.637	2.762	333.59 %

Fig. 17. Performance of the proposed algorithm when using histogram approximation of kernel density. The metrics used are: (a) Track generation (fps), (b) Metropolis Hastings Failure rate, (c), Time taken to compute probability (ms), and (d) Foreground detection (fps).

6 Discussion

In this section, we discuss some of the questions and concerns raised by the anonymous reviewers and the associate editor.

Comment 1: The proposed technique to handle problems of object association across short / long-term occlusions is not convincing especially when there are multiple objects within the scope of this association problem. The underlying track breaks may require sophisticated appearance matching over and above the kinematic constraints learned from the track distributions.

Response: We agree that appearance is a key feature for object association and do fully understand that successful object association requires both spatio-temporal and appearance features to complement each other. Traditionally, appearance models are used along with kinematic constraints (imposed by an assumed motion model, such as constant velocity or constant acceleration) for object association. The contribution of the paper is to learn these kinematic constraints directly from the track distribution instead of assuming some motion model to be valid everywhere in a scene. The kinematic model learned this way not only includes the above mentioned motion properties but also embodies scene-specific knowledge, such as presence of paths, entry/exit points, and people preferences. Similar to traditional kinematic models, the proposed spatio-temporal model can be complemented with more sophisticated appearance matching to solve complex occlusions, e.g., occlusions due to group motion.

Comment 2: Are you assuming ideal tracking? How does imperfect tracking influence the performance?

Response: No, we are not assuming the input tracks to be ideal and like other realtime systems, KNIGHT system is also susceptible to errors due to illumination changes, shadows, and severe occlusions. However, the errors do not greatly affect the performance of the proposed algorithm because the system is trained for video sequences collected over extensive periods of time. Over time, the feature samples in the density estimate are re-sampled and outliers are rejected. Consequently, the true patterns of object dynamics are retained.

Comment 3: Although you state that the proposed algorithm is resilient to failures of the underlying tracker, it is not clear from the formulation that this is so. If a tracker, such as KNIGHT, tends to fragment tracks a lot, then the transition times beyond a small value will almost never get good support. This will not reflect the true distributions. It is not clear from the paper how this aspect is handled? Or is it handled at all?

Response: The proposed framework does not rely on tracking information. As a matter of fact, the distributions can be generated by using the detections alone (by assigning a link between each pair of detection that occurs within a given time period and using it as a data point for the model). Over time, the actual links will persist and the noisy links will average out. The tracking information is used only when it is available to reduce the influence of noise on the model. For example, consider a high precision tracker that links two observations with high precision but may fragment tracks a lot. If the tracker establishes a link between two observations then we ignore other possible links. However, if such a link for an observation is not present then all possible

pairs are considered (for example, for time interval of n frames, the first and last n points in a track do not have links established by the tracker). This way, the algorithm is resilient to failures of the underlying tracker.

Comment 4: If there is a static object, like a truck or a building, that occludes objects consistently over the course of their motion, how can the proposed algorithm connect these objects especially in situations when objects close to each other are moving?

Response: As mentioned in response to Comment 3, the algorithm does not fully rely on the track links and thus can connect objects across long term occlusions due to a static object. The algorithm however cannot solve occlusions caused by objects moving together in groups. Resolving such occlusions would require use of appearance features along with the proposed model.

Comment 5: Are there some anomalous trajectories in your training dataset?

Response: There are some anomalous trajectories in our training sets, for example, people walking through the grassy areas. However, by definition, the anomalous trajectories are very sparse and hence have very low probability in the model as compared to normal trajectories or events.

Comment 6: The determination of d_{th} is tricky. How can it be determined automatically?

Response: The threshold d_{th} depends on several factors including the distance of an object from the camera. In other words, the farther the object, lesser should be the value of d_{th}, especially for scenes where the sizes of objects vary greatly from one region to another within the image lattice. The threshold can be determined automatically if either the camera calibration is known or if the object sizes are also used in the feature vector.

Comment 7: The system has a high computational cost.

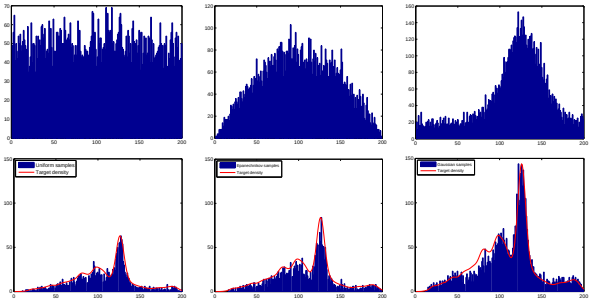
Response: The system has high computational cost for training, but it can be performed off-line. The run-time performance of the system can be significantly improved by using software optimization and using well known approximations of KDE, such as histograms, Fast Gauss Transform [40], [41], and mixture models. In our experiments, the use of histogram approximation (in an un-optimized MATLAB code) greatly improved the run-time performance. (See Fig. 17).

Comment 8: How to apply incremental learning to update the model adaptively.

Response: Traditionally, the kernel density estimation is made adaptive to the changes in the data by keeping a fixed size queue of n data points. Once k new data points are observed, the oldest k points in the queue are replaced with the new data points and the model is updated. The model updates can be done on a low priority thread, although online updating will not have any significant effect on performance. Note that the proposed system is based on static scenes and the availability of large training data, and hence does not have to be updated very frequently.

Comment 9: Why use the Gaussian distribution as proposal density for Metropolis-Hastings sampling?

Response: It is often the case that the true conditional density of object transition given the current state resembles Gaussian distribution. In addition to Gaussian, we experimented with Uniform and Epanechnikov densities centered on the current location. Assuming that the initial location X of a transition can be written as $X = (x_1, x_2)$, the results of sampling from the target density $\int_Y \int_\tau \int_{x_1} p(X, Y, \tau) dx_1 d\tau dY$ are shown in Fig. 18. The 1 dimensional marginal density is chosen for ease of visualization. As can be seen in Fig. 18, the *quality* of samples is not affected by the choice of proposal density, however, the performance in terms of speed is significantly reduced due to a bad choice as more samples are rejected before each acceptance.



(a) Uniform (b) Epanechnikov (c) Gaussian

Fig. 18. A comparison of different proposal densities. The first row shows histograms of 10,000 samples from the respective proposal distributions. The second row shows histograms of samples that were accepted for the target density plotted in red. All histograms contain 200 bins. The acceptance rates for the three densities are 0.2%, 0.25%, and 0.54% respectively.

In conclusion, we have introduced a simple and intuitive novel method for modeling scene dynamics. The proposed model is rich and accurately reflects the transition likelihoods in the scene. We have shown the effectiveness and validity of the method experimentally for diverse surveillance applications.

Acknowledgments

This research was funded in part by the U.S. Government VACE program. The authors are grateful to Xin Li, Yaser Sheikh, and Marshall Tappen for their valuable comments throughout this research.

References

[1] "Special issue on video communications, processing, and understanding for third generation surveillance systems". *Proceedings of the IEEE*, 89(10), Oct 2001.

[2] R. Collins, J. Lipton, and T. Kanade. "Introduction to the special section on video surveillance". *IEEE Trans. PAMI*, 22(8), Aug 2000.

[3] O. Javed, K. Shafique, and M. Shah. "Automated Visual Surveillance in Realistic Scenarios ". In *IEEE Multimedia*, pages 30–39, January - March 2007.

[4] I. Biederman. "On the semantics of a glance at a scene". In *Perceptual Organization*, pages 213–253. Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

[5] A. Torralba. "Contextual influences on saliency". In *Neurobiology of Attention*.

[6] K. Tieu, G. Dalley, and W.E.L. Grimson. "Inference of Non-overlapping Camera Network Topology by Measuring Statistical Dependence". In *IEEE ICCV*, 2005.

[7] T.J. Ellis, D. Makris, and J.K. Black. "Learning a multi-camera topology". In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003.

[8] C. Stauffer. "Learning to track objects through unobserved regions". In *Proceedings of the IEEE Workshop on Motion and Video Computing*, volume 2, pages 96–102, 2005.

[9] O. Javed, K. Shafique, Z. Rasheed, and M. Shah. "Modeling inter-camera space-time and appearance relationships for tracking across non-overlapping views". *Computer Vision and Image Understanding: CVIU*, 109(2), Feb 2008.

[10] H. Buxton. "Generative models for learning and understanding dynamic scene activity". In *Generative Model Based Vision Workshop*, 2002.

[11] A. Hunter, J. Owens, and M. Carpenter. "A neural system for automated CCTV surveillance". In *IEE Intelligent Distributed Surveillance Systems*, 2003.

[12] J. Owens and A. Hunter. "Application of the self-organising map to trajectory classification". In *3rd IEEE International Workshop on Visual Surveillance*, 2000.

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

[14] J.H. Fernyhough, A.G. Cohn, and D.C. Hogg. "Generation of semantic regions from image sequences". In *ECCV*, 1996.

[15] R.J. Howard and H. Buxton. "Analogical representation of spatial events, for understanding traffic behaviour". In *10th European Conference On Artificial Intelligence*, 1992.

[16] E.B. Koller-Meier and L. Van Gool. "Modeling and recognition of human actions using a stochastic approached". In *2nd European Workshop on Advanced Video-Based Surveillance Systems*, 2001.

[17] J. Lou, Q. Liu, T. Tan, and W. Hu. "Semantic interpretation of object activities in a surveillance system". In *ICPR*, 2002.

[18] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. "Using adaptive tracking to classify and monitor activities in a site". In *IEEE CVPR*, 1998.

[19] C. Stauffer and W.E.L. Grimson. "Learning patterns of activity using real time tracking". *IEEE Trans. PAMI*, 22(8):747–767, 2000.

[20] C. Stauffer. "Estimating tracking sources and sinks". In *Second IEEE Event Mining Workshop*, 2003.

[21] P. Remagnino and G.A. Jones. "Classifying surveillance events from attributes and behaviour". In *BMVC*, 2001.

[22] M. Walter, A. Psarrou, and S. Gong. "Learning prior and observation augmented density models for behaviour recognition". In *BMVC*, 1999.

[23] A. Galata, N. Johnson, and D. Hogg. "Learning variable length markov models of behaviour". *Computer Vision and Image Understanding: CVIU*, 81(3):398–413, 2001.

[24] T. Huang and S. Russell. "Object identification in a bayesian context". In *Proceedings of IJCAI,*, 1997.

[25] V. Kettnaker and R. Zabih. "Bayesian multi-camera surveillance". In *IEEE CVPR*, 1999.

[26] S. Dockstader and A. Tekalp. "Multiple camera fusion for multi-object tracking". In *IEEE Workshop on Multi-Object Tracking*, 2001.

[27] D. Hoiem, A. Efros, and M. Hebert. "Putting Objects in Perspective". In *IEEE CVPR*, 2006.

[28] R. Rosales and S. Sclaroff. "Improved tracking of multiple humans with trajectory prediction and occlusion modeling". *In CVPR Workshop on the Interpretation of Visual Motion*, 1998.

[29] R. Kaucic, A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. "A unified framework for tracking through occlusions and across sensor gaps". In *IEEE CVPR*, 2005.

[30] X. Wang, K. Tieu, and E. Grimson. "Learning semantic scene models by trajectory analysis". In *ECCV*, 2006.

[31] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. "Multi-object tracking through simultaneous long occlusions and split-merge conditions". In *IEEE CVPR*, 2006.

[32] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. "A system for learning statistical motion patterns". *IEEE Trans. PAMI*, 28(9):1450–1464, September 2006.

[33] E. Parzen. "On the estimation of a probability density function and mode". *Ann. Math. Stati.*, 33:1065–1076, 1962.

[34] R. Duda, P. Hart, and D. Stork. *"Pattern Classification"*. Wiley Interscience, 2nd edition, 2001.

[35] B. Turlach. "Bandwidth selection in kernel density estimation: A review". *Institut de Statistique*, 1993.

[36] B. W. Silverman. *"Density Estimation for Statistics and Data Analysis"*. Chapman and Hall, 1986.

[37] J. A. Benediktsson and P. H. Swain. "Consensus theoretic classification methods". In *IEEE Trans. Sys. Man and Cybernetics*, volume 22, pages 688–704, 1992.

[38] G. Hinton. "Products of experts". In *ICANN*, pages 1–6, 1999.

[39] K. Shafique and M. Shah. "A non-iterative greedy algorithm for multi-frame point correspondence". *IEEE Trans. PAMI*, Jan 2005.

[40] L. Greengard and J. Strain. "The fast Gauss transform". *SIAM J. Sci. Statist. Comput.*, 12(1):79–94, 1991.

[41] A. Elgammal, R. Duraiswami, and L. Davis. "The fast Gauss transform for efficient kernel density evaluation with applications in computer vision". *IEEE Trans. PAMI.*, 25, Nov 2003.

Imran Saleemi received the BS degree in Computer System Engineering from the Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan in 2004, and MS in Computer Science from the University of Central Florida in 2008. He is currently working towards the PhD degree at the Computer Vision Laboratory at University of Central Florida. His research interests include visual tracking and surveillance, probabilistic graphical models, and multiview object detection and categorization.

Khurram Shafique received the BE degree in computer systems engineering from NED University of Engineering and Technology, Pakistan, in 1998 and the MS and PhD degrees, both in computer science, from the University of Central Florida in 2001 and 2004, respectively. He is currently a research scientist at the Center for Video Understanding Excellence at ObjectVideo, Reston, VA. His research interests include graph theory, discrete optimization, tracking, correspondence across multiple cameras, and real-time surveillance systems. He was a recipient of Hillman Fellowship award for excellence in research in Computer Science in 2003. He is a member of the IEEE.

Mubarak Shah, Agere Chair Professor of Computer Science, is the founding director of the Computer Vision Lab at UCF. He is a co-author of two books (Motion-Based Recognition (1997) and Video Registration (2003)) both by Kluwer Academic Publisher. Dr. Shah is a fellow of IEEE, IAPR and SPIE. In 2006, he was awarded the Pegasus Professor award, the highest faculty award at UCF. He was an IEEE Distinguished Visitor speaker for 1997-2000 and received IEEE Outstanding Engineering Educator Award in 1997. He received the Harris Corporations Engineering Achievement Award in 1999, the TOKTEN awards from UNDP in 1995, 1997, and 2000; Teaching Incentive Program award in 1995 and 2003, Research Incentive Award in 2003, Millionaires Club awards in 2005 and 2006, University Distinguished Researcher award in 2007, honorable mention for the ICCV 2005 Where Am I? Challenge Problem, and was nominated for the best paper award in ACM Multimedia Conference in 2005. He is an editor of international book series on Video Computing; editor in chief of Machine Vision and Applications journal, and an associate editor of ACM Computing Surveys journal. He was an associate editor of the IEEE Transactions on PAMI, and a guest editor of the special issue of International Journal of Computer Vision on Video Computing.