Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2006-008 February 10, 2006

Learning Semantic Scene Models by Trajectory Analysis

Xiaogang Wang, Kinh Tieu, and Eric Grimson

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Learning Semantic Scene Models by Trajectory Analysis

Xiaogang Wang, Kinh Tieu, and Eric Grimson

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology,
Cambridge, MA 02139, USA
{xgwang, tieu, welg}@csail.mit.edu

Abstract. In this paper, we describe an unsupervised learning framework to segment a scene into semantic regions and to build semantic scene models from long-term observations of moving objects in the scene. First, we introduce two novel similarity measures for comparing trajectories in far-field visual surveillance. The measures simultaneously compare the spatial distribution of trajectories and other attributes, such as velocity and object size, along the trajectories. They also provide a comparison confidence measure which indicates how well the measured image-based similarity approximates true physical similarity. We also introduce novel clustering algorithms which use both similarity and comparison confidence. Based on the proposed similarity measures and clustering methods, a framework to learn semantic scene models by trajectory analysis is developed. Trajectories are first clustered into vehicles and pedestrians, and then further grouped based on spatial and velocity distributions. Different trajectory clusters represent different activities. The geometric and statistical models of structures in the scene, such as roads, walk paths, sources and sinks, are automatically learned from the trajectory clusters. Abnormal activities are detected using the semantic scene models. The system is robust to low-level tracking errors.

1 Introduction

The visual surveillance task is to monitor the activity of objects in a scene. In far-field settings (*i.e.*, wide outdoor areas), the majority of visible activities are objects moving from one location to another. Monitoring activity requires low-level detection, tracking, and classification of moving objects. Both high-level activity analysis and low-level vision can be improved with knowledge of scene structure (*e.g.*, roads, paths, and entry and exit points). Scene knowledge supports activity descriptions with spatial context, such as "person crossing *road*," and "person waiting at *bus stop*." Anomalous activity, such as, "car moving off *road*," also depends on scene context. Scene information can also improve low-level tracking and classification [1]. For example, if an object disappears, but not at an exit point, then it is likely a tracking failure instead of a true exit. In classification, we can leverage the fact that vehicles are much more likely than pedestrians to move on the road.

Complementary to the geometric description are the statistics of the scene. A statistical scene model provides an *a priori* probability distributions on where, when, and what types of activities occur. It also places priors on the attributes of moving objects, such as velocity and size. Figure 1(d), shows distributions of location and direction of vehicles on three paths. Statistical information tells us whether the road is one lane or two. It enables concepts such as "vehicle on wrong side of road" and "u-turn in forbidden area". Furthermore scene statistics such as velocity distributions help us predict likely future positions of a tracked vehicle.

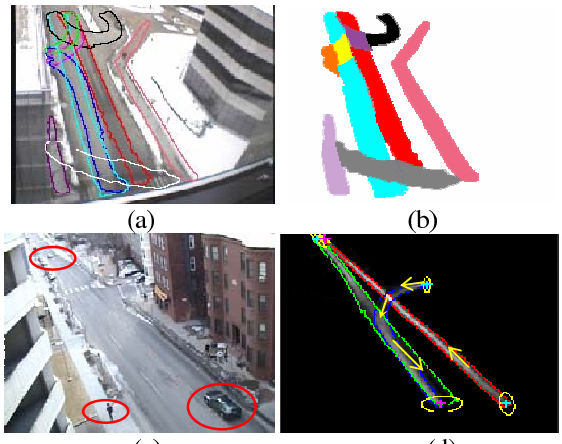


Figure 1. Examples of far-field scene structures. (a): Far-field scene S1; (b): Semantic regions automatically learned in S1. (c): Far-field scene S2. Images of objects undergo substantial projective distortion so that nearby pedestrians appear larger than far vehicles. (d): Automatically learned spatial layout of three vehicle paths showing distributions of location and moving direction, sources marked by cyan cross and sinks marked by magenta cross in S2.

One way to formally model a scene is to represent it as an attributed graph. Vertices as regions and edges as paths represent the coarse structure and topology of the scene. Attributes on vertices and edges further describe the geometry and statistics of

the scene. For example, a source (entry) vertex can be attributed with a mean location and covariance, along with a birth probability. An edge joining a source and sink (exit) can be attributed with the spatial extent of the path and its velocity distribution. In far-field settings, we primarily deal with sources, sinks, and paths between them.

A scene model may be manually input, or possibly automatically extracted from the static scene appearance. However, manual input is tedious if many scenes require labeling, and static scene appearance has large variation and ambiguity. In addition, it is difficult to handcraft the statistics of a scene, or to estimate them from static appearance alone. An example is shown in Figure 1(a)(b). From the image of scene S1, we see one road. However, the road is composed of two lanes of opposing traffic (cyan and red paths). The black path is a one-way u-turn lane. There are two entrances on the left. Vehicles from these entrances wait in the orange region in Figure 1(b) and cross the yellow region on the cyan lane in order to enter the red lane. Pedestrians cross the road via the gray region. In this paper we show how this information can be automatically learned by passive observation of the scene. Our method is based on the idea that because scene structure affects the behavior of moving objects, the structure of the scene can be learned from observing the behavior of moving objects.

1.1. Our algorithm

Gross positions and sizes of moving objects can be obtained from a blob tracker. A moving object traces out a trajectory of locations and sizes from entry to exit. From long-term observation we can obtain thousands of trajectories in the same scene. We propose a framework to cluster trajectories based on types of activities, and to learn scene models from the trajectory clusters. In each cluster, trajectories are from the same class of objects (vehicle or pedestrian), spatially close and have similar directions of motion. In Section 3, we first describe two novel trajectory similarity measures insensitive to low-level tracking failures, which compare:

(I) both spatial distribution and other features along trajectories: two trajectories are similar if they are close in space and have similar feature distribution, e.g. velocity.

(II) only particular features along trajectories, and augment trajectory similarity with a comparison confidence measure. This is used to separate vehicle and pedestrian trajectories by comparing object size. Under this measure, two trajectories are similar if they have similar features, but need not be close in space. A low comparison confidence means the observed similarity may not reflect true similarity in the physical world. In far-field visual surveillance, images of objects undergo large projective distortion in different places as shown in Figure 1(c). It is difficult to compare the size of the two objects when they are far apart. The comparison confidence measure captures this uncertainty.

In Section 4, we propose novel clustering methods which use both similarity and confidence measures, whereas traditional clustering algorithms assume certainty in the similarities. Based on the novel trajectory similarity measures and clustering methods, we propose a framework to learn semantic scene models summarized in Figure 2. The method is robust to tracking errors and noise.

Input: set of trajectories obtained by the Stauffer-Grimson tracker [2] from raw video (trajectories may be fragmented because of tracking errors).

1. Cluster trajectories into vehicles and pedestrians based on size using trajectory similarity measure II and clustering methods in Section 4.
2. Detect and remove outlier trajectories which are anomalous or noisy.
3. Further subdivide vehicle and pedestrian trajectories into different clusters based on spatial and velocity distribution using trajectory similarity I.
4. Learn semantic scene models from trajectory clusters. In particular, sources and sinks are estimated using local density-velocity maps from each cluster, which is robust to fragmented trajectories.
5. Real-time detection of anomalous activity using the learned semantic scene models.

Figure 2. Summary of the scene model learning process.

2. Related Work

Two path detection approaches can be found in [3][4]. Both iteratively merge trajectories into an expanded path. In many settings where observed trajectories are noisy and there are objects roaming between paths, the path regions will become increasingly broader, finally merging into a single large path after long observation. In our framework, trajectories can be well clustered even with the existence of noisy and outlier trajectories. The previous approaches in [3][4] also ignored attributes along the trajectories. This makes it difficult to distinguish vehicles and pedestrians and to separate nearby or overlapping paths with different travel directions, such as the two driving sides of a road. Our approach is able to perform this more fine-grained analysis with a single basic framework.

A straightforward way to learn sources and sinks is to build Gaussian mixture models from the start and end points of the trajectories [5][6]. However, tracking sequences are often fragmented because of object interaction, occlusion, and scene clutter. False entry/exit points caused by broken trajectories will bias the estimation of sources and sinks. Stauffer [7] simultaneously attacked the trajectory fragmentation and source/sink problem by iteratively optimizing the pairwise transition likelihood matrix and estimating sources and sinks. We solve this problem using the local density-velocity map in a trajectory cluster. Sources and sinks can only appear at the two ends of a path. False entry/exit points between the path endpoints are detected and removed by inspecting the local density-velocity distribution in a small neighborhood.

There is a large literature on vehicle vs. pedestrian classification. Our work is related to [8][9] which used object positions in the scene to normalize object features with projective distortion. [8] was a supervised learning approach. [9] required the feature space for each scene to be properly quantized before classification. If some observations of different classes fall into the same cell after quantization, the error will propagate to the entire multiple observation set. In both previous approaches,

spatial location was treated as extra features for similarity, while in out method spatial location is used to calculate comparison confidence.

3 Trajectory similarity

A trajectory is a sequence of observations $A=\{\vec{a}_i\}$, where $\vec{a}_i =< x_i^a, y_i^a, \beta_i^a >$, (x_i^a, y_i^a) are the spatial coordinates of the ith observation, and β_i^a is its feature vector, such as object size and velocity. Recall that our goal is to learn the structure of a scene by clustering trajectories. To this end we propose two trajectory similarity measures.

3.1 Trajectory similarity I

Considering two trajectories $A=\{\vec{a}_i\}$ and $B=\{\vec{b}_i\}$, for a observation $\vec{a}_i$ on A, its nearest observation on B is

$$\psi(i)=\arg\min_{j\in B}\left\|\left(x_i^a-x_j^b, y_i^a-y_j^b\right)\right\|.$$

The directed spatial distance between A and B is

$$h(A,B)=\frac{1}{N_A}\sum_{\vec{a}_i\in A}\left\|\left(x_i^a-x_{\psi(i)}^b, y_i^a-y_{\psi(i)}^b\right)\right\|, \tag{1}$$

where N_A is the observation number in A. This is similar to the modified Hausdorff distance [10]. It is small when A is close to B in space. However, in some cases, we want to distinguish two trajectories even though they are close in space. For example, to separate a road and a walkway beside it, we need to distinguish vehicles and pedestrians by their size difference. If we want to separate two lanes in opposite moving directions, we have to distinguish trajectories with different velocities. Therefore, we further compare other features along the trajectories, and the directed distance is,

$$f(A,B)=\frac{1}{N_A}\sum_{\vec{a}_i\in A}\left(\left\|x_i^a-x_{\psi(i)}^b, y_i^a-y_{\psi(i)}^b\right\|+\gamma d\left(\beta_i^a,\beta_{\psi(i)}^b\right)\right), \tag{2}$$

where $d(\beta_i^a,\beta_{\psi(i)}^b)$ is the dissimilarity measure between features β_i^a and $\beta_{\psi(i)}^b$, and γ is a weighting parameter. The symmetric distance between A and B is

$$F(A,B)=\begin{cases} f(A,B) & if \quad h(A,B)<h(B,A) \\ f(B,A) & if \quad h(A,B)>h(B,A) \end{cases}. \tag{3}$$

It is transformed to a similarity measure

$$S_f(A,B)=\exp\left(-F(A,B)/\sigma\right). \tag{4}$$

Under this measure, two trajectories are similar only if they are close in space and their observations in nearby locations have similar attributes. In (3), we use a minimum instead of the maximum used in the Hausdorff distance. Thus this measure can handle broken trajectories caused by tracking errors. If A is a short broken trajectory beside a long trajectory B as shown in Figure 3(c1), $h(A, B)$ is small while $h(B, A)$ is large. Under (3), the dissimilarity between A and B could be small. It satisfies our

expectation that all broken trajectories on the same path should be grouped into the same cluster.

3.2 Trajectory similarity II

The above similarity measure is inadequate for clustering all trajectories into two classes, vehicles and pedestrians, by comparing size differences. Trajectories of the same class are not necessarily close in space. Furthermore, features on the trajectories cannot be directly compared because of different geometric and photometric transformations in the scene. For example, vehicles are much larger than pedestrians, and thus should be easily distinguished by size. However, as shown in Figure 1(c), because of projective distortion, some pedestrians close to the camera appear larger than vehicles far away in the scene. Without knowledge of camera geometric parameters, we only have the sense that if two objects are close in space, their observed image size similarity reflects their true size similarity, since both objects undergo the same geometric transform in the same place.

If two pedestrian trajectories are far apart or they are only close at some points, such as *A* and *B* in Figure 3(b1), their similarity will be small using the measure in Section 3.1. In the former case, it is difficult to ascertain the true similarity because of projective distortion. In the latter case, we can obtain similarity by comparing the trajectories at intersection points, and ignoring points which are far apart. This leads us to augment the trajectory similarity measure with a comparison confidence.

We first define the comparison confidence between two observations as

$$c(\vec{a}_i,\vec{b}_i)=\exp\left(-\left\|\left(x_i^a-x_j^b,y_i^a-y_j^b\right)\right\|/\sigma_1\right). \tag{5}$$

To compare trajectories *A* and *B*, the directed similarity $S_{A\to B}$ and comparison confidence $C_{A\to B}$ are:

$$S_{A\to B}=\sum_{\vec{a}_i\in A}c(\vec{a}_i,\vec{b}_{\psi(i)})s(\vec{a}_i,\vec{b}_{\psi(i)})\Big/\sum_{\vec{a}_i\in A}c(\vec{a}_i,\vec{b}_{\psi(i)}). \tag{6}$$

$$C_{A\to B}=\sum_{\vec{a}_i\in A}c(\vec{a}_i,\vec{b}_{\psi(i)})^2\Big/\sum_{\vec{a}_i\in A}c(\vec{a}_i,\vec{b}_{\psi(i)}) \tag{7}$$

Here, $s(\vec{a}_i,\vec{b}_{\psi(i)})=\exp\left(-d(\vec{\beta}_i^a,\vec{\beta}_{\psi(i)}^b)/\sigma_2\right)$ is the feature similarity between observations $\vec{a}_i$ and $\vec{b}_{\psi(i)}$. For each observation $\vec{a}_i$ on trajectory *A*, we find its spatially nearest observation $\vec{b}_{\psi(i)}$ on *B*, and compute the feature similarity $s(\vec{a}_i,\vec{b}_{\psi(i)})$ and comparison confidence $c(\vec{a}_i,\vec{b}_{\psi(i)})$. Along trajectory *A*, feature similarities of observations are averaged, weighted by the comparison confidences to get $S_{A\to B}$. The similarity of observations close in space has larger weight for computing trajectory similarity. $C_{A\to B}$ indicates how far apart *A* is from *B*. The symmetric similarity $S(A,B)$ and comparison confidence $C(A,B)$ for trajectories are,

$$S(A,B)=\begin{cases}S_{A\to B} & if \quad C_{A\to B}>C_{B\to A}\\ S_{B\to A} & if \quad C_{A\to B}<C_{B\to A}\end{cases}, \tag{8}$$

$$C(A,B)=\begin{cases} C_{A\to B} & if \quad C_{A\to B} > C_{B\to A} \\ C_{B\to A} & if \quad C_{A\to B} < C_{B\to A} \end{cases}. \tag{9}$$

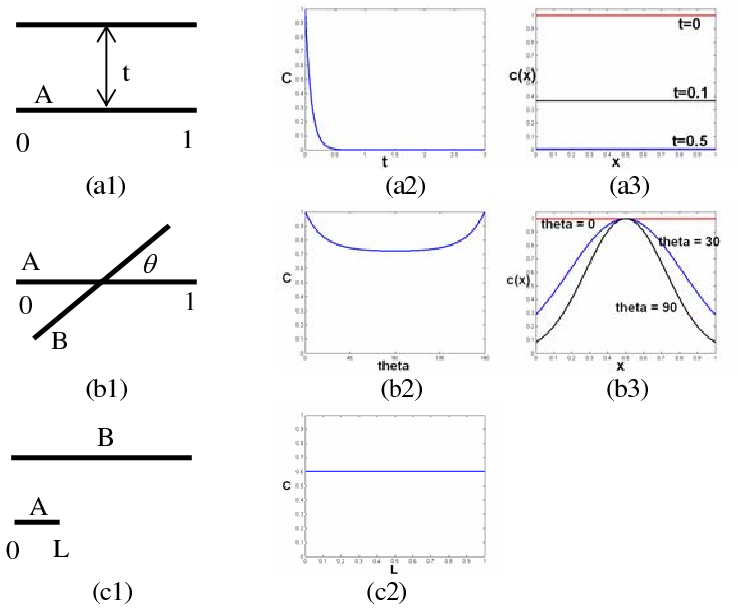


Figure 3. Analyze trajectory comparison confidence $C(A,B)$, and observation comparison confidence $c(x)$ along trajectory A in several typical cases. Find detailed description in text.

In Figure 3, we analyze the comparison confidence measure in several typical cases. In Figure 3(a1), A and B are two parallel lines with equal length 1. As the increase of their distance t, the trajectory comparison confidence $C(A,B)$ reduces in (a2) as expected, and the observation comparison confidence $c(x)$ ($0 \le x \le 1$) is uniform along A as shown in (a3). In (b1), A and B intersect at the middle point. However, as their angle θ changes from 0° to 180°, $C(A,B)$ does not reduce much in (b2). As shown in (b3), the observation comparison confidence $c(x)$ is high around the intersection point, where we have enough information to correctly estimate the true feature similarity between two trajectories. In (c1), A is a short trajectory near the long trajectory B. In (c2), $C(A,B)$ is insensitive to the length L of A. Our measure can handle short broken trajectories caused by tracking failures. This property is the same as that of (3), explain in Section 3.1.

4 Clustering with confidences

Our clustering method is based on pairwise similarity. As mentioned in Section 3.2, some measured similarities between samples may not well approximate the true similarity in the physical world. This makes traditional clustering methods inadequate

because they assume uniformly confident similarity values. For example, in Figure 4, *A*, *B* and *C* are three trajectories in the same class. The observed similarity between *A* and *B* may be low because they are far apart and there is projective distortion, and comparison confidence is also low. *C* has high similarity with both *A* and *B* under our similarity measure, since *C* intersects *A* and *B*. We should emphasize similarities with high confidence, while ignore similarities with low confidence in the cost function. The clustering problem with comparison confidences is formulated as following. The sample set is represented as a graph $G=(V, E)$, where vertices V are the samples and each edge $e_{ij} = < S_{ij}, C_{ij} >$ between any pair of samples has two components: similarity S_{ij} and confidence C_{ij}. The task is to partition V into two subsets V_1 and V_2. There are two ways to augment clustering methods using both similarity and comparison confidence measures: (a) map similarity and confidence measures to a new weight measure, and then apply traditional clustering methods, such as spectral clustering, to the new weight; (b) modify the clustering cost function.

4.1 Remapping weights

Let g be a function mapping S_{ij} and C_{ij} to a new weight, $W_{ij} = g(S_{ij}, C_{ij})$. The key is to preserve similarities with high confidence and leave low confidence similarities uncertain. If the confidence is small, the weight should be set to a median value. We compute W_{ij} as

$$W_{ij} = \frac{{S_{ij}}^{C_{ij}}}{(1-S_{ij})^{C_{ij}} + {S_{ij}}^{C_{ij}}} . \tag{10}$$

The transform functions from similarity to weight given different confidence values from 0 to 1 are shown in Figure 5. When we have no confidence in the similarity ($C = 0$), the weight is 0.5, providing little information for clustering. When we have full confidence ($C = 1$), the weight is exactly the similarity measure. When C changes from 0 to 1, the transform function has a gradual change between the two extremes. Before doing the transform, we first perform histogram equalization on the distribution of similarity values of all the samples in the data set, so that similarities have a uniform distribution from 0 to 1. This normalization makes 0.5 a reasonable value for zero confidence in similarity. Then we apply spectral clustering using the new weights.

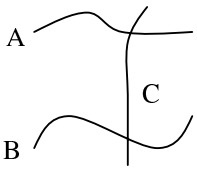


Figure 4. A, B, C are three trajectories in the same class. Because of projective distortion, A and B has low similarity, while C has high similarity with both A and B.

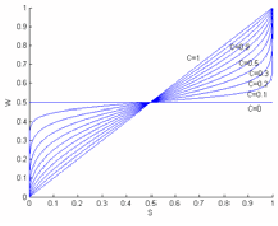


Figure 5. Transform functions from S to W, setting $C = 0, 0.1, 0.2, \ldots, 0.9, 1$.

4.2 Modify the clustering criterion

Traditional clustering methods also can be augmented by including the comparison confidence measure in the cost function. In this work we modify the *average cut*. Let z be an $N = |V|$ dimensional indicator vector, $z_i = 1$ if sample i is in V_1, and $z_i = 0$ if sample i is in V_2. We propose the cost function as the average similarity of the edges connecting V_1 and V_2, weighted by the confidence measures:

$$ave\_cut(V_1,V_2) = \frac{\sum_{i\in V_1, j\in V_2} C_{ij}S_{ij}}{\sum_{i\in V_1, j\in V_2} C_{ij}} = \frac{z^T(D-Q)z}{z^T(T-C)z} \tag{11}$$

The goal is to find the optimal z minimizing $ave\_cut(V_1,V_2)$. Here, $Q = [Q_{ij}]_{N\times N}$, $Q_{ij} = C_{ij}S_{ij}$, $C = [C_{ij}]_{N\times N}$. D and T are two $N\times N$ diagonal matrix with d and t on their diagonal, $d(i) = \sum_j Q_{ij}$, $t(i) = \sum_j C_{ij}$. When all the edges have the same confidence 1, (11) reduces to

$$ave\_cut(V_1,V_2) = \frac{\sum_{i\in V_1, j\in V_2} S_{ij}}{|V_1|\cdot|V_2|} = \frac{1}{N}\left[\frac{cut(V_1,V_2)}{|V_1|} + \frac{cut(V_1,V_2)}{|V_2|}\right]. \tag{12}$$

It is the same as the cost function of the original average cut clustering method [12].

However, *D-Q* and *T-C* have an eigenvector $\vec{1}_{N\times 1}$ with zero eigenvalue, $\vec{1}^T(T-C)\vec{1} = 0$ and $\vec{1}^T(D-Q)\vec{1} = 0$. So the Reyleigh quotient (11) is undefined. When all the edges with $C_{ij} > 0$ form one connected component in this graph, $\vec{1}_{N\times 1}$ is the only eigenvector with zero eigenvalue for *T-C*, $z^T(T-C)z = 0 \Leftrightarrow z = \alpha\vec{1}$ where α is a scalar. The remaining *N-1* eigenvectors $L_{N\times(N-1)}$ all have positive eigenvalues $\Lambda_{(N-1)\times(N-1)} > 0$, $T - C = L\Lambda L^T$. It can also be proved that (11) remains unchanged when z shifts by $\alpha\vec{1}$,

$$\frac{z^T(D-Q)z}{z^T(T-C)z} = \frac{(z+\alpha\vec{1})^T(D-Q)(z+\alpha\vec{1})}{(z+\alpha\vec{1})^T(T-C)(z+\alpha\vec{1})}. \tag{13}$$

Any indicator vector z can be expressed as $z = Lx + \vec{1}\alpha$. According to (13), we only consider the indicator vector in the form $y = Lx$, setting α to 0. From (11),

$$\frac{y^T(D-Q)y}{y^T(T-C)y} = \frac{x^TL^T(D-Q)Lx}{x^T\Lambda x}. \tag{14}$$

If z is relaxed to real value, (13) can be minimized by solving the generalized eigenvalue system,

$$L^T(D-Q)Lx = \lambda\Lambda x. \tag{15}$$

(14) can be transformed to a standard eigenvalue system,

$$\Lambda^{-1/2}L^T(D-Q)L\Lambda^{-1/2}x' = \lambda x', \tag{16}$$

where $x' = \Lambda^{1/2} x$. (11) is minimized by computing the eigenvector x' with the smallest eigenvalue in (15), and $y = L\Lambda^{-1/2} x'$. Notice that y is a continuous vector while z should be discrete vector on {1, 0}, and there is a shift difference $\alpha\vec{1}$ between y and z. We partition y into two pieces by further clustering y using k-means. An alternative way is to search for the split point as in [12].

5 Trajectory Clustering

5.1 Clustering different types of trajectories (vehicles vs. pedestrians)

Scene structures and activities are often related to the class of objects. For example, cars drive on roads and pedestrians often appear on walk paths. Vehicles and pedestrians may have different sources and sinks. So we cluster trajectories into vehicles and pedestrians using the similarity and confidence measure proposed in Section 3.2 and the clustering methods in Section 4. We add a small disturbance 0.05 to each confidence measure making it positive. The feature similarity between observations in (6), is defined as

$$s(\vec{a}_i, \vec{b}_j) = \exp\left(-\left(\frac{(r_i^a - r_j^b)^2}{r_i^a r_j^b} \right) / \sigma_2 \right), \tag{16}$$

where r_i^a and r_j^b are the sizes of observations $\vec{a}_i$ and $\vec{b}_j$. We set parameter $\sigma_1 = \sigma_2 = 0.01$ in (5) and (16).

5.2 Clustering activity groups

Each class of trajectories, vehicles or pedestrians, is further clustered according to different spatial and velocity distributions. We define the trajectory similarity as described in Section 3.1, considering velocity direction along the trajectories. Dissimilarity between observation features in (2) is

$$d(\beta_i^a, \beta_{\psi(i)}^b) = 1 - \frac{\vec{v}_i^a \cdot \vec{v}_{\psi(i)}^b}{\left\|\vec{v}_i^a\right\| \cdot \left\|\vec{v}_{\psi(i)}^b\right\|} \tag{17}$$

$\vec{v}_i^a$ and $\vec{v}_{\psi(j)}^b$ are the velocities of $\vec{a}_i$ and $\vec{b}_{\psi(i)}$. The width and the height of the scene is normalized to 1 and parameter γ in (2) is set to 0.25. Spectral clustering is applied using the defined trajectory similarity.

Before clustering, we first remove outlier trajectories. Usually these are noisy trajectories caused by tracking errors, anomalous trajectories, *e.g.*, a car drives out of the way, or some pedestrians roaming between different paths. In visual surveillance, they may be of particular interest, and it is nice that our algorithm can detect them by comparing trajectories. Because they are not strongly constrained by scene structures, the scene structure models will be learnt more accurately by removing them. For each trajectory A, we find its N nearest trajectories $B_i \ (i = 1, \ldots, N)$, and compute the average distance

Table 1. Results of clustering trajectories into vehicles and pedestrians. I: compare average observation size along the trajectory and use spectral clustering; II: compare more observation features, (size, speed, size variation, aspect ratio and percentage occupancy of silhouette), also averaged along the trajectory; III: size similarity defined in (2)(3)(4) without considering comparison confidence; IV: compare trajectory distance in space as define in (1); V: combine size similarity and comparison confidence as described in Section 3.2 and 4.

Method	Scene	Cluster	Vehicle	Pedestrian
I	S1	Cluster 1	127	0
		Cluster 2	42	368
	S2	Cluster 1	55	2
		Cluster 2	14	16
II	S1	Cluster 1	162	154
		Cluster 2	7	214
	S2	Cluster 1	65	0
		Cluster 2	4	18
III	S1	Cluster 1	152	0
		Cluster 2	17	368
	S2	Cluster 1	61	0
		Cluster 2	8	18
IV	S1	Cluster 1	166	242
		Cluster 2	3	126
	S2	Cluster 1	40	8
		Cluster 2	29	10
V	S1	Cluster 1	167	0
		Cluster 2	2	368
	S2	Cluster 1	69	0
		Cluster 2	0	18

$$E(A) = \frac{1}{N} \sum_{i=1}^{N} f(A, B_i). \tag{18}$$

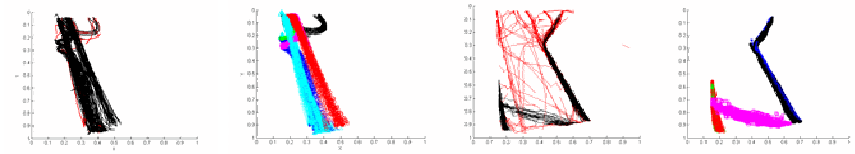
We reject trajectories with large average distance to neighbors as outliers.

5.3 Experiments

In Table 1, we report the results of clustering trajectories into vehicles and pedestrians using different clustering methods and similarity measures. There are two data sets from the two scenes shown in Figure 1. We show the numbers of vehicle and pedestrian trajectories in each cluster. The average observation size along the trajectory cannot separate vehicles and pedestrians, since there is overlap between the size distributions of the two classes. In method II, we add more features, such as speed, size variation, aspect ratio and percentage occupancy of silhouette, which proved effective in vehicle/pedestrian classification [8], to compute the similarity. Although these discriminative features work well in supervised classification using some complex classifiers, they are not effective in clustering. Our two clustering approaches in Section 4.1 and 4.2 using both similarity and confidence measures give the same result

on this data set. They perfectly separate vehicle and pedestrian trajectories in Scene S2, and incorrectly cluster only two among 537 trajectories in Scene S1. If we only use the size similarity measure as define in (2)(3)(4), or only compare spatial distance as defined in (1), the result is worse. Note that our method is essentially unsupervised and only requires labeling a cluster as vehicle or pedestrian.

The separated vehicle trajectories and pedestrian trajectories are further clustered into different activity groups. Some results from scene S1 are shown in Figure 6. The vehicle trajectories are clustered into six clusters. Because the road has two opposite driving directions, the trajectories on the two lanes are separated into cyan and red clusters. The vehicles from the two entrances on the left of the scene enter the road along three different paths. The black clusters detect the one-way road and u-turn in the upper center of the scene. Most of the pedestrian trajectories crossing the road and roaming between the two walk paths are first removed as outlier trajectories. The remaining pedestrian trajectories are well clustered into five clusters on the two walk paths aside the road and one path crossing the road, because there are two opposite moving directions on each walk path aside the road.



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

Figure 6. Clustering vehicle and pedestrian trajectories in Scene S1. (a): outlier vehicle trajectories in red; (b): six vehicle trajectory clusters (c): outlier pedestrian trajectories in red; (d): five pedestrian clusters.

6 Learning semantic scene models

6.1 Road and walk path models

For each cluster Ω, we detect its spatial extent in the scene, and estimate the density and velocity direction distributions in the region. The density at position (x, y) is estimated as,

$$p_{\Omega}(x, y) = \sum_{\vec{a} \in A} \sum_{A \in \Omega} \phi^{(x,y)}_{(x_i^a, y_i^a)}, \tag{19}$$

where $\phi^{(x,y)}_{(x_i^a, y_i^a)} = \exp\left(-\left\| (x - x_i^a, y - y_i^a) \right\|^2 / \sigma_3 \right)$. The velocity direction distribution at (x, y) is modeled as a circular normal (von Mises) distribution [13],

$$p(\theta) = \frac{e^{\cos\theta - \alpha_{\Omega}(x,y)}}{2\pi I_0(\kappa)} \tag{20}$$

with mean $\alpha_{\Omega}(x, y)$ computed by

$$\alpha_\Omega(x,y) = \arctan \frac{\sum_{\vec{a}_i \in A} \sum_{A \in \Omega} \phi^{(x,y)}_{(x_i^a, y_i^a)} \sin\theta_i^a}{\sum_{\vec{a}_i \in A} \sum_{A \in \Omega} \phi^{(x,y)}_{(x_i^a, y_i^a)} \cos\theta_i^a} \tag{21}$$

where θ_i^a is the angle of velocity direction at $\vec{a}_i$.

The path region is obtained by thresholding the density distribution, using $\max P_\Omega(x,y)/10$. Experimental results on scene S1 are shown in Figure 1(a)(b). The vehicles and pedestrian paths are shown in Figure 1(a). Using some logical operations on the path regions of different clusters, some semantic regions are obtained. In Figure 1(b), the cyan and red regions are two lanes on the main road in the scene. The black color marks a u-turn. When the vehicles merge from two entrances on the left of the scene, they wait in the orange region before entering the road, and cross the yellow region on the cyan road in order to be on the red road. The purple region has a similar semantic explanation. Pedestrians cross the road via the gray region.

6.2 Sources and Sinks

Two interesting scene structures are locations where vehicles or pedestrians enter or exit the scene. They are called sources and sinks. Trajectories are often broken because of inevitable tracking failures. There are false entry/exit points biasing the estimation of sources and sinks as shown in Figure 8(a). We remove them using the local density-velocity map. Sources and sinks should be on the two ends of the path regions. A false entry/exit point inside the path region has high density around its neighborhood, since there are many other trajectories passing through this point. In each trajectory cluster, starting from a start/end point of the trajectory, we find its local path by searching forward and backward L steps. On the local path, the next point is decided by the average velocity direction of the current position as shown in Figure 7 (a). In Figure 7 (b), we sample an entry point (A), an exit point (C), and a false entry/exit point (B) on the density map of one trajectory cluster. We can clearly see their difference on density distribution along the local path. The entry point has a very low density along the path behind it. The exit point has a very low density along the local path ahead of it. A false entry/exit point has little change on density along the whole local path, since trajectories in the same cluster have similar moving directions and they do not diverge. We distinguish them comparing the average densities of the two halves of the local path. Results are shown in Figure 8.

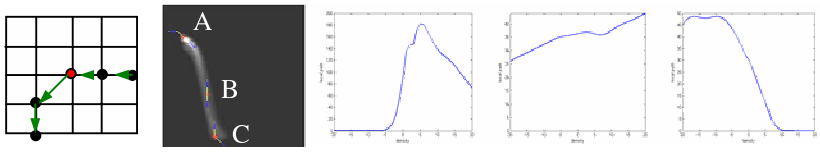
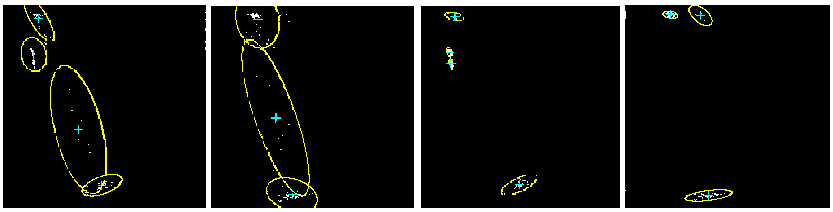


Figure7. Removing break points in trajectory clusters. (a): Find local path of the red point based on velocity distribution; (b): Examples of entry point (A), exit point (C), false entry/exit point (B) on cluster density map; (c): Density distributions along local paths of A, B, C.



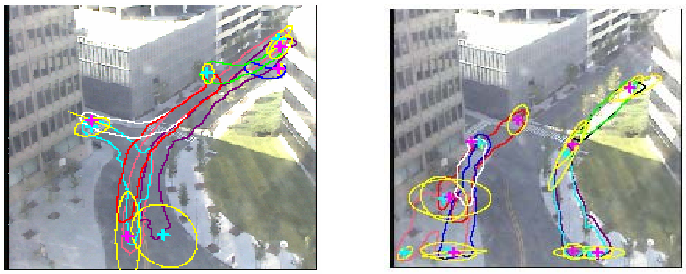
(a) Gaussian mixture models of sources and sinks directly learnt from the start and end points of trajectories.

(b) Gaussian mixture models of sources and sinks learnt from the trajectory clusters using the local density-velocity map.

Figure 8. Learning vehicle sources and sinks models in Scene S1.

6.3 More experimental results

More experimental results of learning semantic scene models in scene S2 and S3 are shown in Figure 1(c)(d) and Figure 9. In S3, there is a red pedestrian path crossing the road, however, it is not the crosswalk beside it. People tend to take a short cut instead of using the crosswalk. This is one illustration of how our learnt scene models can provide additional information unavailable from the static image.



(a) Vehicles (b) Pedestrians

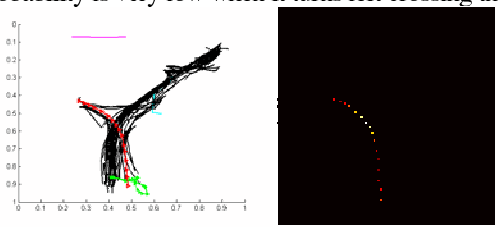
Figure 9. Extract paths, sources and sinks of vehicles and pedestrians in Scene S3. Path boundaries are marked by different color, the source and sink centers are marked by cyan and magenta crosses. The yellow ellipses indicate the estimated extent of sources/sinks.

7. Anomalous trajectory detection

As mentioned in Section 5, anomalous trajectories can be detected as outlier samples. In Figure 10 (a), outlier vehicle trajectories in S3 are marked by different colors. The green trajectory is a car backing up in the middle of the road. The car on the red trajectory first drives along the purple path in Figure 10(a), then it turns left, crosses the red path on its left side, and has opposite moving direction with the trajectories in the cyan cluster. So it is detected as an anomalous trajectory.

We further develop the system to real-time detect anomalous activity. When an object enters the scene, we classify it into vehicle or pedestrian. For each vehicle/pedestrian class, we model the density and velocity direction distributions in the scene as mixture models, since we have built the statistical model for each cluster in Section 7. When the object passes a location, a likelihood is computed, so we can

monitor the object at each position without requiring the whole trajectory data. In Figure 10 (b) we plot the log likelihood of the red trajectory in Figure 10(a) at different locations. The probability is very low when it turns left crossing the red path.



(a) (b)

Figure 10. Detect anomalous trajectories in S3. (a): outlier trajectories; (b): transform the log-likelihood into density map. The white color indicates low probabilities (highly anomalous).

8 Discussion

We described a framework to learn semantic scene models by trajectory analysis. Trajectories related to different kinds of activities are separated into different clusters using novel trajectory similarity measures, and clustering methods with similarity and comparison confidences. The scene semantic models are applied to anomalous activity detection. Our system can also handle to errors in low-level tracking. We plan to investigate what other scene structures can also be automatically learned by observing trajectories. In addition we believe there are further applications of our learned scene model such as more complex activities across longer time scales and involving multiple objects. Finally, our notion of clustering with confidences deserves further study and may be applicable to other areas of computer vision and statistical modeling.

Reference

[1] 1. R. Kaucic, A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs, "A Unified Framework for Tracking through Occlusions and across Sensor Gaps," in Proceedings of CVPR 2005.

[2] C. Stauffer and E. Grimson, "Learning Patterns of Activity Using Real-Time Tracking," IEEE Trans. on PAMI, Vol. 22, No. 8, pp. 747-757, 2000.

[3] D. Makris and T. Ellis, "Path Detection in Video Surveillance," Image and Vision Computing, Vol. 20, pp. 859-903, 2002.

[4] J. H. Fernyhough, A. G. Cohn, and D. C. Hogg, "Generation of Semantic Regions from Image Sequences," in Proc. of ECCV, 1996.

[5] D. Makris and T. Ellis, "Automatic Learning of an Activity-Based Semantic Scene Model," in Proc. of IEEE Conference on Advanced Video and Signal Based Surveillance 2003.

[6] S. J. Mckenna and H. Nait-Charif, "Learning Spatial Context from Tracking Using Penalized Likelihood," in Proc. of ICPR, 2004.

[7] C. Stauffer, "Estimating Tracking Sources and Sinks," in Proc. of Event Mining Workshop, July 2003.

[8] B. Bose and E. Grimson, "Improving Object Classification in Far-Field Video," in Proc. CVPR, 2004.

[9] C. Stauffer, "Minimally-Supervised Classification using Multiple Observation Sets," ICCV 2003.
[10] M. P. Dubuisson and A. K. Jain, "A Modified Hausdorff distance for Object Matching," in Proc. of ICPR, 1994.
[11] M. Meila and J. Shi, "A Random Walk View of Spectral Segmentation," in Proc. of AISTATS, 2001.
[12] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," in Proc. of CVPR, 1997.
[13] E. J. Gumbel and J. A. Greenwood "The Circular Normal Distribution: Theory and Tables," J. Amer. Stat. Soc., Vol. 48, No. 261„ pp. 131-152, 1953.