Trajectory Learning and Analysis Based on Kernel Density Estimation

Jianying Zhou, Kunfeng Wang, Shuming Tang, Member, IEEE, Fei-Yue Wang, Fellow, IEEE

Abstract—This paper presents a novel kernel density estimation approach to vehicle trajectory learning and motion analysis. The framework comprises a training stage and a testing stage. In the training stage, vehicle trajectories are first clustered by the hierarchical spectral clustering method. Then, through the proposed kernel density estimation approach, the average kernel density of one point on a trajectory can be estimated. In the testing stage, the compactness estimated by a Gaussian kernel function is introduced. Abnormal trajectories are detected with compactness lower than expected for a few consecutive frames. Vehicle motions are identified into multiple activities with their respective trajectory compactness.

I. INTRODUCTION

A KEY GOAL of video surveillance is to understand the interactions and behaviors present in a scene. It is important to detect abnormal activities or behaviors for surveillance system that need continually monitor a site. Unfortunately, large amounts of video data are generated, making it a tedious and tiring job for human to manually process accurately and quickly. Computer vision technology helps to automate the process. Using computer vision techniques, in activity analysis systems, normal activity patterns in the video are filtered out automatically, and attention is focused on the data in which the activity is abnormal.

By observing and collecting tracking data, which is required in the activity analysis systems, normal motion patterns corresponding to lanes in the road can be learned offline or on-line. Then the normal motion patterns can be used to detect anomalous actions if the activity happens with a low probability to match any normal activity.

A training set of trajectories is acquired by collecting tracking data for a period of time. Then the set is clustered to find the major spatial routes, which can be probabilistically modeled by several statistical models, such as hidden Markov model (HMM), Gaussian mixture models and so on. Over the past few years, based on trajectory clustering, a lot of work has been devoted to understanding activity patterns of vehicles [1], [2], [3], [4], [5], [6]. On the other hand, statistical models can be used to complement object tracking or shadow detection [7], [8].

Wang et al. [9] used spectral clustering method to build semantic regions for two scenes through trajectory clustering. They first used the size of objects and spatial information to cluster trajectories into vehicles and pedestrians. Then spatial and temporal information were used to cluster trajectories into more clusters. Wang et al. [10], [11] also used LDA (latent Dirichlet allocation) and HDP (hierarchical Dirichlet process) models to do activity analysis separately. LDA and HDP are two methods based on Bayesian models; they have been used very well for word-document analysis by supposing that words often co-existing in the same documents are clustered into the same topic. Supposing documents were trajectories and words were observations and topics were semantic regions, Wang et al used their improved HDP and LDA individually to analyze the traffic scenes.

Hu et al. [3] used FCM (fuzzy c-means) to analyze traffic scenes and modeled the probability density as Gaussian function to make prediction of the activity.

After clustering the trajectories, Morris et al. [5] and Bashir et al. [12] used HMM to describe the transitions between states and obtained good experimental results. They supposed that each activity path can be represented by several states in HMM, after learning the transition matrix by standard methods, such as the Expectation Maximum (EM) algorithm, the probability of moving from one state to another could be predicted. Instead of finding the transition probabilities, Saleemi et al. [7] used the learned density distribution, which is obtained in the training stage, as a generative model and sample feature points from the model to construct a sequence of tracks in the testing stage. In their work, kernel density estimation (KDE) was used to describe the transition distribution, and then the Metropolis Hastings Sampling was used to sample the trajectories to help object tracking or activity understanding. In [13], 4-D histograms were built to describe the scene. After learning the histogram of each class using their proposed kernel which was similar to KDE, the obtained statistical descriptions of motion patterns can be used to detect and classify the trajectory in the testing stage.

As depicted in [14], the work in the training stage can be summarized in Fig. 1. In the tracking stage, vehicles are detected and tracked between consecutive frames, and at the same time the spatial and velocity attributes for each frame of the vehicle are extracted. Before cluster, the trajectories...
should be preprocessed to remove noises or may be normalized with the same length. Based on the spatial and velocity attributes, the similarity measure between trajectories will be built by Euclidean distance or Hausdorff distance etc. Trajectory clustering could be done by unsupervised learning algorithms, such as k-means clustering, SOM, and spectral clustering. After trajectory clustering, semantic regions of each activity pattern are depicted by statistical distributions. After obtaining the statistical distributions, we can use the distributions to detect and understand vehicle motion in the testing stage or turn back to improve low-level tracking or detection as in [7], [8].

The rest of this paper is organized as follows. In section II, some theoretical background is introduced, and the details of spectral clustering algorithm we used are first given. Using the hierarchical spectral clustering method, the trajectories are clustered into five clusters. Section III describes our proposed method to build statistical description of each cluster and introduce a factor of compactness for trajectory classification and abnormal detection in the testing stage. In Section IV, some experimental results are shown and analyzed. Finally, in Section V, some conclusions and discussions bring the paper to completion.

II. THEORETICAL BACKGROUND AND TRAJECTORY CLUSTERING

A. Object Detection and Tracking

A lot of work of object detection in traffic surveillance has been done in the lab I am working for [15]. Gaussian mixture models of background modeling method [16], is used in our work, in each frame after background subtraction, objects are detected and in two frames matched according to their position and velocity attributes. Then vehicle detection and tracking have been completed. The trajectory of each observed object is a sequence of tracking states obtained in every frame, A = {a_i}, where a_i can depict things like position, velocity, appearance, shape, or other object attributes. In our work we select the attributes with a_i = <x_i^a, y_i^a, u_i^a, v_i^a>, where (x_i^a, y_i^a) are the spatial position of the i-th observation, and (u_i^a, v_i^a) are the velocity attribute in x and y direction separately of the image. This trajectory information forms the basic block for further activity analysis.

Although tracking is well studied, there are still many difficulties due to various effects, such as occlusion, camera twitting and real-time adaptability to changing conditions. These cause errors in the form of noise measurements or trajectories breaking, which will be accounted for in the activity learning process.

B. Spectral Clustering Algorithm and Trajectories Clustering

Many kinds of clustering algorithm need the trajectories to be set with the same length. This will introduce noises in the clustering. Spectral clustering method avoid these problems.

In the past few years, spectral clustering method has become one of the most popular modern clustering algorithms. Spectral clustering is simple to implement and very often outperforms traditional clustering algorithms such as k-means [17]. In spectral clustering, we just need to get the similarity or dissimilarity matrix between different observations. Hence the length of every observation is not necessary to be equal differing from that of k-means or SOM. In spectral clustering, similarity graph matrix W between each two observations is first built, from which the Laplacian matrix L can be derived, and then the first K eigenvectors of L can be found. After that let the eigenvectors v_1, ..., v_K be matrix V’s columns and {y_i, i = 1, ..., n}, be the rows of matrix V. Subsequently, k-means clustering algorithm is used to cluster the rows {y_i, i = 1, ..., n} of matrix V. After clustering the rows of matrix V into K clusters, C_1, ..., C_K, the clusters of observations A_1, ..., A_K, can be obtained with A_i = {j|y_j ∈ C_i}.

In this paper, hierarchical spectral clustering is used to cluster trajectories into several clusters. The trajectories are first clustered into three clusters and then the three clusters are divided into five final clusters as shown in Fig. 2.

As in [9], considering any two trajectories, A = {a_i} and B = {b_i}, for an observation a_i on A, its nearest observation on B is b_{ψ(i)} with,

\[
ψ(i) = \arg \min_{j \in B} \|x_i^a - x_j^b, y_i^a - y_j^b\| \tag{1}
\]

The directed spatial distance between A and B is

\[
h(A, B) = \frac{1}{N_A} \sum_{a_i \in A} \|x_i^a - x_{ψ(i)}^b, y_i^a - y_{ψ(i)}^b\| \tag{2}
\]

where N_A is the total observation number on trajectory A. Considering the influence of velocity, the directed distance between A and B is

\[
f(A, B) = \frac{1}{N_A} \sum_{a_i \in A} (\|x_i^a - x_{ψ(i)}^b, y_i^a - y_{ψ(i)}^b\| + γd(u_i^a, v_{ψ(i)}^b, u_{ψ(i)}^b, v_{ψ(i)}^b)) \tag{3}
\]

where \(d(u_i^a, v_{ψ(i)}^b, u_{ψ(i)}^b, v_{ψ(i)}^b)\) is the dissimilarity measure between velocities of A and B. And γ is a weight coefficient to make sure that the spatial distance and velocity distance have similar scale before adding them. The systematic distance between A and B is

\[
F(A, B) = \begin{cases} 
f(A, B) & \text{if } h(A, B) < h(B, A) \\ f(B, A) & \text{if } h(A, B) > h(B, A) \end{cases} \tag{4}
\]
Then the systematic distance can be transformed to a similarity measure:

$$S(A, B) = \exp(-F(A, B))$$  \hspace{1cm} (5)

In the experiments, $d(u^a_i, v^a_i, u^b_i, v^b_i)$ is chosen as,

$$d(u^a_i, v^a_i, u^b_i, v^b_i) = 1 - \frac{u^a_i \cdot u^a_i + v^a_i \cdot v^a_i}{\sqrt{(u^a_i)^2 + (v^a_i)^2}(u^b_i)^2 + (v^b_i)^2}}$$  \hspace{1cm} (6)

with $\gamma = 0.02$ in our experiments, and $d(u^a_i, v^a_i, u^b_i, v^b_i)$ ranges from 0 to 2. To make sure that the spatial distance and the velocity distance is similar in scale, the height and width of the image in the experiment are normalized to 1 before clustering.

Under the similarity measure detailed up, two trajectories are similar only if they are similar both in spatial position and velocity. Note that before clustering we do not need to resample trajectories with fixed equal length, and we also can deal with broken trajectories. In the training stage, as in [9], before clustering we first detect and remove noises in the trajectories. These noises may come from occlusion or camera twitting. Using directed distance, for each trajectory, after finding its nearest trajectories, we compute the average distance and then reject trajectories with large average distance to neighbors as noises as shown in Fig. 2(a) with white color curves. In our experiments, we select $N = 10$, and the average distance is 0.04.

After noises removal, similarity matrix for spectral clustering could be built. Then the trajectory clustering results are shown in Fig. 2(a) and Fig. 2(b). Fig. 2(a) shows the clustering results with noises shown and Fig. 2(b) shows the clustering results without noise shown.

### C. Density Estimation

As in [3], [5], [12], after trajectory clustering, each cluster of trajectories are assumed to be distributed according to mixtures of Gaussian probability density function. Hence, each cluster $k$ is characterized by its mean vector $\mu_k$ and covariance matrix $C_k$. Using the density distribution we can complete the detection and prediction of the activities. But this method will be failed when the trajectories can not be described in mixtures of Gaussian functions. Some nonparametric density estimation method, such as kernel density estimation (KDE) method, can be used to avoid the limitation.

Kernel density estimation is a nonparametric density estimation method [18]. In KDE, a kernel centered at each observation is used to obtain a continuous probability density function (PDF) of the data. In fact, we do not need the continuous probability density function in traffic surveillance, as the vehicles moving with regular patterns appear at similar positions on the image. In our paper, we proposed a method to reflect the regular pattern in each cluster avoiding the computation of continuous PDF. Our method considers the average density of one point in each cluster.

### III. PROPOSED METHOD

Vehicle tracking, noises detection and trajectories clustering are detailed in previous section and the results are shown in Fig. 2. In all our experiments, we set manually an entry zone and an exit zone in the region of interesting (ROI). Only trajectories that begin in the entry zone and end in the exit zone are retained in the training stage for further processing. But all the trajectories which are detected in ROI are considered in the testing stage, even though the track is not begin in the entry zone or end in the exit zone.

Then, using the trajectories in each cluster we will estimate the density distribution of one point on trajectory to build the statistical description models. We introduce a compactness function to evaluate the compactness of the trajectory with the regular pattern in each cluster, then activity classification and abnormal detection will be completed according to the compactness. These details are given next.

#### A. Kernel Density Estimation

We use exponential kernel function to evaluate the density of one point on trajectories. For a point $(x^a_i, y^a_i, u^a_i, v^a_i)$, the kernel density can be computed using its nearest point on each trajectory in the same cluster as the kernel. So we call our method as kernel density function even though we do not compute the continuous probability density function. Our kernel density function is based on the separable sum of individual kernels in the spatial dimensions and the orientation dimensions and the speed magnitude dimensions. On the other hand, we just select the nearest point as the kernel to estimate the density. Our method can be described as follows.

Considering a trajectory $A$ in class $j$ and any trajectory $B$ in the same class, the kernel density of the $i$th point $(x^a_i, y^a_i, u^a_i, v^a_i)$ on trajectory $A$ based on $B$ can be evaluated as

$$kde(i, B) = (\cos{vel}(u^a_i, v^a_i, u^b_i, v^b_i), v^b_i) + 2spatial_kde(x^a_i, y^a_i, u^b_i, v^b_i) + vel_ratio(u^a_i, v^a_i, u^b_i, v^b_i))/4.0$$  \hspace{1cm} (7)

where the first item in the right of (7) describes the operation of the velocity orientation at point $(x^a_i, y^a_i, u^a_i, v^a_i)$. This item...
is assessed as
\[
\cos_{vel}(u_i^a, v_i^a, u_i^b, v_i^b) = \exp(-0.5d(u_i^a, v_i^a, u_i^b, v_i^b)) \tag{8}
\]
where the \(d(u_i^a, v_i^a, u_i^b, v_i^b)\) is as same as Part B in Section II and \((x_{\psi(i)}^b, y_{\psi(i)}^b, u_{\psi(i)}^b, v_{\psi(i)}^b)\) is the nearest point on \(B\) of the \(i\)th point \((x_i^a, y_i^a, u_i^a, v_i^a)\) on \(A\). This item depicts the similarity between the velocity orientation of two points.

The spatial kernel is evaluated as
\[
\text{spatial kde}(x_i^a, y_i^a, x_i^b, y_i^b) = \exp(-\|x_i^a - x_i^b, y_i^a - y_i^b\|) \tag{9}
\]

We double the second item to make sure that the spatial attribute have similar scale with the temporal information, which is denoted by the sum of the first and the third items in (7).

The third item of (7) describes the kernel in the speed magnitude, or the similarity between the magnitude of velocity at point \((x_i^a, y_i^a, u_i^a, v_i^a)\) and the magnitude of the velocity of its nearest point on trajectory \(B\). This item is
\[
\text{vel ratio}(u_i^a, v_i^a, u_i^b, v_i^b) = \exp(-\max(|1 - r_1|/r_2, |1 - r_2|/r_1)) \tag{10}
\]
\[
r_1 = \sqrt{(u_i^a)^2 + (v_i^a)^2}/\sqrt{(u_i^b)^2 + (v_i^b)^2} \tag{11}
\]
\[
r_2 = \sqrt{(u_i^b)^2 + (v_i^b)^2}/\sqrt{(u_i^a)^2 + (v_i^a)^2} \tag{12}
\]

Here \(r_1\) and \(r_2\) are the two ratios between the velocity magnitude of the \(i\)th point on trajectory \(A\) and the velocity magnitude of its nearest point on trajectory \(B\). As normal activity patterns, the velocity direction and magnitude of the \(i\)th point will be similar with that of its nearest point on trajectory \(B\). Thus \(r_1\) and \(r_2\) should be close to 1. We select the larger one between \(|1 - r_1|/r_2\) and \(|1 - r_2|/r_1\) to describe the dissimilarity between the \(i\)th point on trajectory \(A\) and its nearest point on trajectory \(B\).

So the kernel density for the \(i\)th point on \(A\) in class \(j\), based on all the trajectories in class \(j\) can be evaluated by
\[
k(i) = \sum_{B \in \text{class}_j} kde(i, B) \tag{13}
\]

Then the average kernel density for one point on trajectory \(A\) is computed by
\[
kp(A) = \frac{(\sum_{i \in A} k(i))}{poin_num}
= \frac{(\sum_{i \in A} (\sum_{B \in \text{class}_j} kde(i, B)))}{poin_num} \tag{14}
\]
where, the \(poin_num\) is the length of trajectory \(A\).

For every trajectory we can get the average kernel density of one point in each class and its distribution is shown in Fig. 3.

Then the results of Fig. 3 are used to compute the mean kernel density of a point in each class. The results of the means are shown in table I. Where the mean \(\mu_j\) of class \(j\) is evaluated by
\[
\mu_j = \frac{\sum_{A \in \text{class}_j} kp(A)}{\text{Traj num}} \tag{15}
\]

Here \(\text{Traj num}\) is the total number of trajectories in class \(j\).

In table I, \(\sigma\) is the parameter selected according to experiments. These parameters will be used to evaluate the compactness of each trajectory in the testing stage.

According to the distribution of the average kernel density in each class, we can see that the kernel density distribution is very compact in the same class and have different ranges in different class. Each trajectory is composed by several points, and as normal trajectories should have similar spatial distribution with similar velocities when they move in the same region. Therefore we select the nearest point to evaluate the average kernel density for every point is feasible. To do this, we do not need to resample the trajectory with equal length, and in the testing stage, this make sure that the density is only related to the points before.

Along with section II and this part of section III, we gives the details of all the work in the training stage. Then, in the next part, introducing a factor of compactness, we will give the details of the work in the testing stage.

B. TRAJECTORY CLASSIFICATION AND ABNORMAL DETECTION

In the testing stage, in order to classify the trajectory we introduce the compactness to measure the similarity of kernel density of the trajectory with the mean of kernel density in each class.

When a vehicle moves in the ROI, through tracking, the trajectory \(T\) with spatial and velocity attributes can be got. Given that the vehicle moves with activity pattern of class \(j\), we
then employing our kernel density function described previously, the average kernel density for one point $kp_{(t,j)}(T)$ at time $t$ of the trajectory $T$ is computed. Here $t$ denotes time (in frames). Since the vehicle enter ROI, $t$ will increase until the vehicle vanishes from tracking, then $kp_{(t,j)}(T)$ at time $t$ is,

$$kp_{(t,j)}(T) = \frac{\sum_{i=1}^{t} \sum_{B \in \text{class}j} kde(i, B)}{t} (16)$$

After computing the average kernel density of time $t$, the compactness of the trajectory $T$ with class $j$ is $P_T(t,j)$ which is computed as,

$$P_T(t,j) = \exp(-kde_{(t,j)}(T) - \mu_j^2/2\sigma_j^2) (17)$$

$P_T(t,j)$ is the compactness of the average kernel density at time $t$ with $\mu_j$, the mean of kernel density of class $j$ computed in the training stage. $P_T(t,j)$ is similar with the role of probability that this trajectory belongs to class $j$, but the sum of $P_T(t,j)$ of all classes is not 1.

After computing the average kernel density $kp_{(t,j)}(T)$ and the corresponding compactness $P_T(t,j)$, for $j = 1, \ldots, 5$, the most compact class is,

$$c = \arg \max_j (P_T(t,j)) (18)$$

So the vehicle at time $t$ is moving with activity pattern of class $c$, $c \in \{1, \ldots, 5\}$, and if the maximum of compactness, $\max_j (P_T(t,j))$, is less than 0.05 for consecutive 15 frames, we set $c = 6$ as abnormal activity, when a vehicle moves with very high speed or in the opposite direction or very far away from the road.

**IV. EXPERIMENTS**

In the experiments, the data of video lasting more than one hour with size $320 \times 240$ is captured by an off-the-shelf CCD camera. The camera is fixed on a tall building and overlooks the traffic scene. The computer is equipped with Intel® Core™ 2 Duo processor P8700 (2.26GHz processor, 1.98GHz RAM). The algorithm is only applied in the ROI set manually shown in Fig. 4.

The clustering results of trajectories in the training stage are shown in Fig. 2(a) and Fig. 2(b) separately. In all the figures we use different color curves to represent different activity patterns. Blue curves represent class 1 where vehicles move straight along the lane near which the right-turn

![Fig. 5. Experiments results. The first three images are examples for vehicles move in different straight roads. And last two graphs show a car which is turning right.](image)

![Fig. 6. An experimental result of abnormal activity detection. In the figure a bike with two people, as they are large enough to deal with, we detect the abnormal of the activity, as it moves in the opposite direction.](image)

![Fig. 7. Another example for right-turning detection.](image)
The probability distribution of each class, we detect and classify the vehicle activity successfully, obtaining a good understanding of activity patterns.

In the future, more work will be focused on the prediction of activities and more semantic explanation would be given to activity patterns. Also, we can use the high-level information to help to resolve the occlusion which we have not considered here.

**References**


