Correspondence-Free Activity Analysis and Scene Modeling in Multiple Camera Views

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

| Citation | Xiaogang Wang, Kinh Tieu, and E.L. Grimson. “Correspondence-Free Activity Analysis and Scene Modeling in Multiple Camera Views.” Pattern Analysis and Machine Intelligence, IEEE Transactions on 32.1 (2010): 56-71. © 2009 Institute of Electrical and Electronics Engineers. |
| As Published | http://dx.doi.org/10.1109/tpami.2008.241 |
| Publisher | Institute of Electrical and Electronics Engineers |
| Version | Final published version |
| Accessed | Wed Apr 17 11:16:59 EDT 2019 |
| Citable Link | http://hdl.handle.net/1721.1/53523 |
| Terms of Use | Article is made available in accordance with the publisher’s policy and may be subject to US copyright law. Please refer to the publisher’s site for terms of use. |
| Detailed Terms | |
Correspondence-Free Activity Analysis and Scene Modeling in Multiple Camera Views

Xiaogang Wang, Kinh Tieu, and W. Eric L. Grimson, Fellow, IEEE

Abstract—We propose a novel approach for activity analysis in multiple synchronized but uncalibrated static camera views. In this paper, we refer to activities as motion patterns of objects, which correspond to paths in far-field scenes. We assume that the topology of cameras is unknown and quite arbitrary, the fields of views covered by these cameras may have no overlap or any amount of overlap, and objects may move on different ground planes. Using low-level cues, objects are first tracked in each camera view independently, and the positions and velocities of objects along trajectories are computed as features. Under a probabilistic model, our approach jointly learns the distribution of an activity in the feature spaces of different camera views. Then, it accomplishes the following tasks: 1) grouping trajectories, which belong to the same activity but may be in different camera views, into one cluster; 2) modeling paths commonly taken by objects across multiple camera views; and 3) detecting abnormal activities. Advantages of this approach are that it does not require first solving the challenging correspondence problem, and that learning is unsupervised. Even though correspondence is not a prerequisite, after the models of activities have been learned, they can help to solve the correspondence problem, since if two trajectories in different camera views belong to the same activity, they are likely to correspond to the same object. Our approach is evaluated on a simulated data set and two very large real data sets, which have 22,951 and 14,985 trajectories, respectively.

Index Terms—Visual surveillance, activity analysis in multiple camera views, correspondence, clustering.

1 INTRODUCTION

In visual surveillance, a key task is to monitor activities in the scene. People have interest in discovering typical and abnormal activities, detecting activities of some categories, and knowing some structures of the scenes, such as paths commonly taken by objects, sources, and sinks, where objects appear and disappear. Because visual surveillance systems collect a huge amount of data from many different scenes, people expect the algorithms to be unsupervised with little human labeling effort as possible. When modeling activities, while some approaches [1], [2], [3], [4], [5], [6], [7], [8] directly extract motion and appearance features from video streams without relying on tracking and object detection, in many surveillance systems [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], especially in far-field settings, objects are first detected and tracked and the activity of an object is then treated as sequential movements along its trajectory. Through tracking, an activity executed by a single object can be separated from other co-occurring activities and features related to the activity can be integrated as a track. In many far-field surveillance settings, the captured videos are of low resolution and poor quality and it is difficult to compute more complicated features, such as gestures, local motions, or appearance of objects within the tracks. Usually, only positions of objects are recorded along tracks, which are called trajectories. Although quite simple, the motion patterns of trajectories can distinguish many different activity categories, especially in far-field settings. Examples can be found in Fig. 1a. The goal of this work is to model activities by trajectory analysis: clustering trajectories into different activities, detecting abnormal trajectories, and modeling paths commonly taken by objects.

Many approaches were proposed by the authors of [10], [20], [21], [22], [23], [24], [25], [17], [19] to cluster or classify trajectories into activities. They used the spatial proximity between a pair of trajectories, measured in different ways, for clustering. Activities are often closely related to the structures of scenes, e.g., roads, paths, and entry and exit points, which can help not only high-level description of activities [17] but also low-level tracking and classification [26]. The models of paths commonly taken by objects can be obtained by finding the spatial extents of trajectory clusters [21], [27], [28], [17], [29]. Entry and exit points are detected at the ends of paths [17].

All of these clustering and modeling approaches assumed a single camera view whose visible area is finite and limited by the structures of the scene. In order to monitor activities in a wide area, video streams from multiple cameras have to be used. Examples of activities observed in different camera views can be found in Fig. 1b. Many systems [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48] using multiple cameras for visual surveillance have been developed in recent years and these are based on various assumptions on the number of cameras, the topology and geometry of camera views, and camera calibration. Most of these approaches focused on tracking
The camera settings in our approach are as follows:

- The cameras are static and synchronized but do not have to be calibrated.
- The fields of view covered by these cameras may have no overlap or any amount of overlap. However, we assume that, when an object exits a camera view, it is already in or will enter one of the other camera views within time $T$.
- Objects may move on different ground planes.

Examples of multicamera settings are shown in Fig. 2.

We briefly explain several basic concepts and assumptions held in this paper. There are paths in the physical world. Objects move along these paths and, thus, have different moving patterns (examples of different moving patterns can be found in Fig. 1), which are called activities. A path may be observed in multiple camera views and has spatial distributions in these views. Although some paths, such as roads of vehicles, can be recognized by their physical features, some paths cannot be. For example, pedestrians take a shortcut on a grass field. A trajectory which only records the positions of an object is a history of the movement of an object in a camera view. The points on trajectories are called observations. In this work, trajectories are clustered into different activities based on their spatial distributions and moving directions. A cluster of trajectories is related to a path in the physical world. The scene of a camera view is quantized into small cells. When an object moves, it connects two cells far apart in a camera view by its trajectory. Our probabilistic model is based on some simple, general assumptions on the spatial and temporal features related to activities: 1) Cells located on the same path are likely to be connected by trajectories; 2) trajectories passing through the same path belong to the same activity; and 3) it is likely for trajectories of the same object observed in different camera views to be on the same path in the real world and belong to the same activity; otherwise, objects might switch paths when crossing different camera views.

In our approach, a network is first built by connecting trajectories that are in different camera views and whose temporal extents are close by edges. Then, a probabilistic model in which different kinds of activities have distributions in low-level feature spaces of different camera views is built. A trajectory is treated as a set of observations that belongs to different activities. A smoothness constraint requires that two neighboring trajectories connected by an edge have similar distributions over activities. Trajectories are clustered according to the assigned major activities among their observations. The distributions of activities over feature spaces of different camera views model the regions of paths across cameras.

1.1 Related Work

Many similarity-based trajectory clustering methods have been proposed in the past years. The spatial proximity
views had significant overlap and that objects moved on the data from moving objects. They also assumed that camera features between different views is difficult. In general, automatically finding correspondence of static objects on their spatial proximity on the common ground plane. In order to extend these approaches to multiple camera views, trajectories observed in different camera views have to be stitched together.

Considerable work has been done to solve the challenging correspondence problem of trajectories observed in multiple camera views. One way is to manually label salient points in the scene and record their coordinates in the 3D world. After mapping 2D image planes to the 3D world, objects can be tracked in multiple camera views. When the camera views overlap, static features can be selected to compute an homography between two camera views and calibrate camera views to a single global ground plane. Trajectories in different camera views can be stitched based on their spatial proximity on the common ground plane. In general, automatically finding correspondence of static features between different views is difficult.

Lee et al. [33], Sheikh and Shah [47], and Stauffer and Tieu [38] calibrated multiple camera views using tracking data from moving objects. They also assumed that camera views had significant overlap and that objects moved on the same ground plane. Lee et al. [33] and Sheikh and Shah [47] assumed that the topological arrangement of camera views was known. Stauffer and Tieu [38] could automatically infer it, but with high complexity (\(O(N^2)\), where \(N\) is the number of camera views).

When the camera views are disjointed or their overlap is small, automatic calibration is difficult and the appearance of objects is often used as a cue to correspondence and the topology has to be known [47]. Stauffer and Tieu [38] inferred the topology of disjoint camera views using the transition time between cameras.

Even given similarities between trajectories observed in different camera views, solving the correspondence problem is still difficult because of the large search space, especially when there are many trajectories and cameras. It requires searching in the solution space of \(N\)-partite graphs, where \(N\) is the number of cameras. In general, if there are more than two cameras, the problem is \(NP\)-hard in the number of trajectories. It has solution in polynomial time only with some particular topologies of camera views and the topology has to be known [37].

In summary, all these trajectory correspondence approaches had various assumptions on the topology and geometry of camera views, and they faced difficulties of camera calibrations, appearance matching, inference on the topology of camera views, and high computational cost to search for the optimal solution. The contributions of this paper are that we directly cluster trajectories into activities and model regions of paths over a multicamera network without solving the correspondence problem. Given a general setting of a camera network, solving the correspondence problem is difficult. So, our method has less restriction on the topology of camera views, the structures of the scene,
and the number of cameras. Furthermore, in our approach, each trajectory cluster is represented by a parametric probabilistic model. It does not require computing the similarity between each pair of trajectories. It has much lower space complexity compared with those similarity-based approaches. So it is more appropriate to process huge data sets often required in visual surveillance applications.

The paper is organized as follows: Section 2 explains how to compute low-level features from trajectories and quantize them into visual words. In Section 3, a trajectory network is built by connecting trajectories, which are in different camera views. Section 4 introduces the models of activities (paths). In Section 5, our algorithms are evaluated on a simulated data set and two real data sets, a parking lot scene and a street scene, each of which has four cameras. The views and topology of these cameras are shown in Fig. 2.

2 Feature Space

Objects are tracked in each of the camera views independently using the Stauffer-Grimson tracker [10]. A trajectory is treated as a set of observations. The locations and moving directions of observations of an object are computed as features and quantized to visual words according to a codebook of its camera view. In each camera view, the space of the view is uniformly quantized into small cells and the velocity of objects is quantized into several directions. A global codebook concatenates the codebooks of all the camera views. Thus, the word value of an observation \(i\) is indexed by \((c_i, x_i, y_i, d_i)\) in the global codebook, \(c_i\) is the camera view in which \(i\) is observed, \((x_i, y_i)\) and \(d_i\) are the quantized coordinates and moving direction of observation \(i\) in camera \(c_i\). The set of visual words on the trajectory is modeled as exchangeable (i.e., the distribution is invariant to a permutation of the observations). The moving direction encoded in the word value captures the first order temporal information among observations. Although quite simple, the position and velocity features can distinguish many different activity patterns, especially in far-field settings.

3 Trajectory Network

A network is built connecting trajectories observed in multiple camera views based on their temporal extents. Each trajectory is a node on the network. Let \(t_{si}\) and \(t_{ei}\) be the starting and ending times of trajectory \(i\). \(T\) is a positive temporal threshold. It is roughly the maximum transition time of objects crossing the gap between adjacent camera views. If trajectories \(a\) and \(b\) are observed in different camera views and their temporal extents are close,

\[
(t_{sa} \leq t_{sb} \leq t_{ea} + T) \lor (t_{sb} \leq t_{sa} \leq t_{eb} + T),
\]

then \(a\) and \(b\) will be connected by an edge on the network. This means that \(a\) and \(b\) may be the same object since they are observed by cameras around the same time. There is no edge between two trajectories observed in the same camera view. An example can be found in Fig. 3. As shown in Fig. 3a, the views of cameras 1 and 2 overlap and are disjoint with the view of camera 3. Trajectories 1 and 2 observed by cameras 1 and 2 correspond to the same object moving across camera views. Their temporal extents overlap as shown in Fig. 3b, so they are connected by an edge on the network as shown in Fig. 3d. Trajectories 3 and 4 observed by cameras 1 and 3 correspond to an object crossing disjoint views. Their temporal extents have no overlap but the gap is smaller than \(T\), as shown in Fig. 3c, so they are also connected. Trajectories 3 and 6, 5 and 7 do not correspond to the same objects, but their temporal extents are close, so they are also connected on the network. A single trajectory 3 can be connected to multiple trajectories (4 and 6) in other camera views. An edge on the network indicates a possible correspondence candidate only based on the temporal information of trajectories. But, we do not really solve the correspondence problem when building the trajectory network since many edges are actually false correspondences. The network simply keeps all of the possible candidates.

4 Probabilistic Model

In this section, we describe our probabilistic model that clusters trajectories in different camera views into activities and models paths across camera views. Our work is related to topic models, such as Probabilistic Latent Semantic Analysis (pLSA) [54] and Latent Dirichlet Allocation (LDA) [55], which were used for word document analysis. These topic models assume that a document is a mixture of topics and cluster words, such as “professor” and “university,” that often
co-occur in the same documents into one topic, such as “education.” In our domain, documents are trajectories, words are observations, and topics are activities (paths). Each activity has a distribution over locations and moving directions in different camera views, and corresponds to a path. If two word values which are indexes of locations and moving directions often co-occur on the same trajectories, they are on the same path. Trajectories passing through the same path belong to the same activity. In previous topic models, documents are generated independently. However, we assume that if two trajectories in different camera views are connected by an edge in the network, which means that they may correspond to the same object since they are observed by cameras around the same time, they tend to have similar distributions over activities. Thus, the distributions of an activity (a path of observations) in different camera views can jointly be modeled. In Fig. 4, we describe the high-level picture of our model. Trajectories a and b are observed in different camera views and connected by an edge on the network. Points on trajectories are assigned to activity categories by fitting activity models. Thus, both a and b have distributions over activities. The smoothness constraint requires that their distributions over activities are similar in order to have small penalty. In this example, both trajectories a and b have a larger distribution on activity 1, so the models of activity 1 in two different camera views can be related to the same activity.

Let $M$ be the number of trajectories. Each trajectory $j$ has $N_j$ observations. Each observation $i$ on trajectory $j$ has a visual word value $w_{ji}$, which is an index of the global codebook. Observations will be clustered to one of the $K$ activity categories. Let $z_{ji}$ be the activity label of observation $i$ on trajectory $j$. Each activity $k$ has a multinomial distribution $\phi_k$ over the global codebook. So, an activity is modeled as distributions over space and moving directions in multiple camera views. $\phi_k$ is sampled from a Dirichlet prior

$$p(\phi_k|\beta) = \text{Dir}(\phi_k; \beta),$$

where $\text{Dir}(\cdot; \cdot)$ is Dirichlet distribution and $\beta$ is a flat hyperparameter. If a visual word $w_{ji}$ has activity label $z_{ji}$, its data likelihood is

$$p(w_{ji}|z_{ji}, \phi_k) = \phi_k w_{ji}.$$

Equations (2) and (3) are the same as modeled in LDA.

Each trajectory has a random variable $\theta_j$, which is the parameter of a multinomial distribution over $K$ activities. Activity labels $\{z_{ji}\}$ of observations are sampled from $\theta_j$. If two trajectories $j_1$ and $j_2$ are connected by an edge on the network, they are neighbors and the smoothness constraint requires that $\theta_{j_1}$ and $\theta_{j_2}$ are similar and the distributions of $\{z_{j_1}\}$ and $\{z_{j_2}\}$ are similar. The joint distribution of $\{\theta_j\}$ and $\{z_{ji}\}$ is modeled as

$$p(\{\theta_j\}, \{z_{ji}\}|\alpha, \gamma) \propto \prod_{j=1}^M \prod_{k=1}^K (\theta_{jk})^\alpha - 1 \prod_{\{j, j\} \in E} \prod_{k=1}^K (\theta_{jk})^{n_{jk}} (\theta_{jk})^{\gamma n_{jk}} \prod_{j=1}^M \prod_{i=1}^{N_j} \theta_{z_{ji}} = \prod_{j=1}^M \prod_{k=1}^K \Gamma(\alpha + \gamma \sum_{\{j, j\} \in E} n_{jk}) \prod_{j=1}^M \prod_{i=1}^{N_j} \theta_{z_{ji}} \prod_{j=1}^M \prod_{k=1}^K \Gamma(K \cdot \alpha + \gamma \sum_{\{j, j\} \in E} n_{jk}) \prod_{j=1}^M \prod_{i=1}^{N_j} \theta_{z_{ji}}$$

$$\Gamma(\cdot)$$ is the Gamma function. $n_{jk}$ is the number of observations assigned to activity $k$ on trajectory $j$. $E$ is the set of pairs of neighboring trajectories which are connected. $\Omega_j$ is the set of trajectories connected with $j$. $\alpha$ is a flat Dirichlet prior as a hyperparameter. $(\sum_{j \in \Omega_j} n_{j_1}, \ldots, \sum_{j \in \Omega_j} n_{j_K})$ is the histogram of observations assigned to $K$ activity categories on the neighboring trajectories of $j$. It is used as the Dirichlet parameter for $\theta_j$, after being weighted by a positive scalar $\gamma$ and added to a flat prior $\alpha$. Let $\rho_k = \alpha + \gamma \sum_{j \in \Omega_j} n_{j_k}$. According to the properties of the Dirichlet distribution, if $\theta_j \sim \text{Dir}(\rho_1, \ldots, \rho_K)$, the expectation of $\theta_j$ is $(\rho_1 / \sum \rho_k, \ldots, \rho_K / \sum \rho_k)$ and its variation is small if $\sum \rho_k$ is large. Note that $z_{ji}$ is sampled from $\theta_j$ and $\theta_j$ has a constraint added by $z_{j'}$ on its neighboring trajectories. So trajectory $j$ tends to have a similar distribution over activities as its neighboring trajectories, which means that they are smooth. A larger $\gamma$ puts a stronger smoothness constraint. If $\gamma = 0$, (4) is the same as in LDA.
where \( \{ \theta_i \} \) are sampled from a Dirichlet prior \( \text{Dir}(\gamma; \alpha) \) independently.

Given (2), (3), and (4), finally the joint distribution of \( \{ \phi_k \} \), \( \{ \theta_i \} \), \( \{ z_{ji} \} \), and \( \{ w_{ji} \} \) is

\[
p(\{ \phi_k \}, \{ \theta_i \}, \{ z_{ji} \}, \{ w_{ji} \}) = p(\{ \theta_i \}, \{ z_{ji} \}|\alpha, \gamma) \prod_{k=1}^{K} p(\{ w_{ji} \}|\{ z_{ji} \}, \{ \phi_k \})
\]

\[
= \prod_{j=1}^{M} \prod_{k=1}^{K} p(\{ w_{ji} \}|\{ z_{ji} \}, \{ \phi_k \}) \prod_{i=1}^{N_i} p(\{ \theta_i \}, \{ z_{ji} \}|\alpha, \gamma)
\]

\[
\times \prod_{k=1}^{K} \text{Dir}(\phi_k; \beta) \prod_{j=1}^{M} \prod_{i=1}^{N_i} (\theta_{zi}, \phi_{zji} w_{ji}).
\]

\[\tag{5}\]

\[\text{4.1 Learning and Inference}\]

Our goal is to estimate activity labels \( \{ z_{ji} \} \) and activity models \( \{ \phi_k \} \). We do inference by Gibbs sampling. It turns out that \( \{ \theta_i \} \) and \( \{ \phi_k \} \) can be integrated out during the Gibbs sampling procedure:

\[
p(\{ z_{ji} \}, \{ w_{ji} \}|\alpha, \beta, \gamma)
\]

\[
= \int_{\{ \phi_k \}} \int_{\{ \theta_i \}} p(\{ \theta_i \}, \{ z_{ji} \}, \{ w_{ji} \}|\alpha, \beta, \gamma) d\{ \theta_i \} d\{ \phi_k \}
\]

\[
= \int_{\{ \phi_k \}} \prod_{j=1}^{M} \prod_{k=1}^{K} p(\{ \theta_i \}|\alpha, \gamma) \prod_{j=1}^{M} \prod_{i=1}^{N_i} p(\{ w_{ji} \}|\{ z_{ji} \}, \{ \phi_k \}) d\{ \phi_k \}
\]

\[
\times \prod_{i=1}^{N_i} p(\{ \theta_i \}, \{ z_{ji} \}|\alpha, \gamma) d\{ \theta_i \}
\]

\[
\propto \int_{\{ \phi_k \}} \prod_{j=1}^{M} \prod_{k=1}^{K} \prod_{i=1}^{N_i} \phi_{zji} w_{ji} d\{ \phi_k \}
\]

\[
\times \prod_{i=1}^{N_i} \prod_{j=1}^{M} \prod_{k=1}^{K} \phi_{jki} \prod_{j=1}^{M} \prod_{i=1}^{N_i} \theta_{zi} d\{ \theta_i \}
\]

\[
= \int_{\{ \phi_k \}} \prod_{j=1}^{M} \prod_{k=1}^{K} \prod_{i=1}^{N_i} \phi_{zji} w_{ji} d\{ \phi_k \}
\]

\[
\times \prod_{j=1}^{M} \prod_{i=1}^{N_i} \phi_{jki} \prod_{j=1}^{M} \prod_{i=1}^{N_i} \theta_{zi} d\{ \theta_i \}
\]

\[\tag{6}\]

\[\text{4.2 Labeling Trajectories into Activities}\]

A trajectory is labeled as activity \( k \), if most of its observations are assigned to \( k \). The activity label of an observation can be obtained during the Gibbs sampling procedure based on (7). However, there may be an oversmoothing effect, since in some cases, most of the trajectories being the neighbors of trajectory \( j \) do not correspond to the same object as \( j \). In this work, we adopt an alternative labeling approach, which actually achieves better performance in experiments. As shown by the experimental results in Section 5, the activity models learned from Gibbs sampling are distinctive enough to label trajectories. After the activity models have been learned and fixed at the end of Gibbs sampling, which uses (7) and (8), we ignore the smoothness constraint among trajectories and label the observation as

\[
z_{ji} = \arg \max_k l_{k_{w_{ji}}},
\]

\[\tag{9}\]

This is also used to label an unseen new trajectory.

\[\text{4.3 Detection of Abnormal Trajectories}\]

When detecting abnormal trajectories, we also ignore the smoothness constraint and fix the learned activity models \( \{ \phi_k \} \). A trajectory is detected as an abnormality if it does not fit any activity model well. Then abnormality detection is reduced to the Latent Dirichlet Allocation model proposed in [55]. The likelihood of a trajectory \( j \) under the learned activity models \( \{ \phi_k \} \) is

\[
p(\{ w_{ji} \}|\alpha, \{ \phi_k \})
\]

\[
= \int_{\{ \theta_i \}} p(\{ \theta_i \}|\alpha) \prod_{j=1}^{M} \sum_{z_{ji}} p(z_{ji}|\theta_j) p(w_{ji}|\phi_{zji}),
\]

\[\tag{10}\]

where \( p(\theta_j|\alpha) \) is a Dirichlet distribution, and both \( p(z_{ji}|\theta_j) \) and \( p(w_{ji}|\phi_{zji}) \) are discrete distributions. Since the computation of (10) is intractable, in [55], a variational approach was used to compute a lower bound of (10). A trajectory is flagged as abnormal if its lower bound is small.
4.4 Complexity
In order to simplify the notation, we assume that all the trajectories have the same number of observations, which is a fixed constant. The space complexity of our approach is $O(WK) + O(MK)$, while that of similarity-based approaches is at least $O(M^2)$. The storage of similarity-based approaches is unmanageable when $M$ is huge. $W$ is the size of the codebook, $K$ is the number of activity categories, and $M$ is the number of trajectories. In our approach, the time complexity of each Gibbs sampling iteration is $O(M)$; however, it is difficult to provide theoretical analysis on the convergence of Gibbs sampling. Similarity-based approaches have to compute the similarity of $O(M^2)$ pairs of trajectories and if spectral clustering is used, it is quite challenging to compute the eigenvectors of a huge $M \times M$ similarity matrix when $M$ is large. The time complexity of our approach to label a new trajectory into one of the activity categories or detect a new trajectory as abnormal is $O(K)$, while the time complexity of similarity-based approaches is at least $O(M)$. So our approach is much more efficient when the number of trajectories is huge.

5 Experimental Results
We evaluate our approach on two data sets, a parking lot scene and a street scene. Each has four camera views. Each camera view is of size $320 \times 240$. To build the codebook, each camera view is quantized into $64 \times 48$ cells. Each cell is of size $5 \times 5$. The moving directions of moving pixels are quantized into four directions. There are tracking errors in both of the two data sets. For example, a track may break into fragments because of interactions among objects. In order to obtain more quantitative evaluation, we simulate some trajectories and if spectral clustering is used, it is quite challenging to compute the eigenvectors of a huge $M \times M$ similarity matrix when $M$ is large. The time complexity of our approach to label a new trajectory into one of the activity categories or detect a new trajectory as abnormal is $O(K)$, while the time complexity of similarity-based approaches is at least $O(M)$. So our approach is much more efficient when the number of trajectories is huge.

5.1 Learning Activity Models
5.1.1 Parking Lot Scene
The parking lot data set has 22,951 trajectories, collected in 10 hours during the daytime over three days. Inspections shows that it is a fairly busy scene. The topology of its four camera views is shown in Fig. 2a. The view of camera 1 has no overlap with other camera views. However, the gap between views of cameras 1 and 2 is small. The views of cameras 2 and 3 have small overlap. The views of cameras 3 and 4 have large overlap. Our approach does not require the knowledge of the topology of cameras. Fourteen different activities are learned from this data set. Six of them are shown in Fig. 5. For each activity, we plot its distribution over space and moving directions in the four camera views and the trajectories clustered into this activity. When visualizing activity models, moving directions are represented by different colors and the density of distributions over space and moving directions is proportional to the brightness of colors (high brightness means high density). When plotting trajectories, random colors are used to distinguish individual trajectories.

In Fig. 5, activity 1 captures vehicles and pedestrians entering the parking lot. It has a large extent in space and is observed by all four cameras. Activity 2 captures vehicles and pedestrians leaving the parking lot. In activities 3 and 5, pedestrians are walking in the same direction but on different paths. From the distributions of their models, it is observed that the two paths are side by side but well separated in space. The path of activity 4 occupies almost the same region as that of activity 3. However, pedestrians are moving in opposite directions in these two activities, so the distributions of their models are plotted in different colors. In activity 6, pedestrians appear from behind the trees and a building as observed by cameras 3 and 4 and disappear from a gate of the parking lot in the view of camera 2.

5.1.2 Street Scene
The topology of the four camera views of the street scene is shown in Fig. 2b. Camera 1 has a distant view of the street. Camera 2 zooms in on the top-right part in the view of camera 1. The view of camera 3 has overlap with the views of cameras 1 and 2. It extends the top-right part of the view in camera 1 along the street. The view of camera 4 partially overlaps with the bottom region of the view in camera 1. There are 14,985 trajectories in this data set, collected in 30 hours during the daytime in four days. Seventeen activities are learned in this scene. Six of them are shown in Fig. 6.

Activity 1 (Fig. 6) captures vehicles moving on the road. It is observed by all of the four cameras. Vehicles first move from the top-right corner to the bottom-left corner in the view of camera 4. Then, they enter the bottom region in the view of camera 1 and move upward. Some vehicles disappear at the exit points observed in the views of cameras 2 and 3, and some move further beyond the view of camera 3. In activities 3, 5, and 6, pedestrians first walk along the sidewalk in the view of camera 1 and then cross the street as observed by camera 4. The paths of activities 5 and 6 occupy similar regions in the view of camera 1, but their paths diverge in the view of camera 4. The paths of activities 2 and 3, 4 and 5 occupy the same regions but pedestrians are moving in opposite directions on them.

As shown in Figs. 5 and 6, the models of activities reveal some structures, such as paths commonly taken by objects, and entrance and exit points in the scene. Some paths are less related to the appearance of the scene. For example, some paths cross the street outside the crosswalk in the street scene. Usually paths have spatial extents in multiple camera views. These regions can be detected by simply thresholding the density of the distributions of activities ($\phi_k$ in (5)). As observed, in these two very large data sets, there are many outlier trajectories, which do not fit any activity model well, such as those crossing the grass fields in the parking lot scene. They are finally assigned some activity at random or because part of the trajectory fits a particular activity.

5.1.3 Negative Log Likelihood on Testing Data
Since clustering trajectories into activities is unsupervised learning, we compute the negative log likelihood on testing data to evaluate its performance. It is the log of perplexity proportion to the number of bits required to encode the testing data. It measures how unseen testing data fit the model learned from training data. Two hundred trajectories randomly sampled from each camera serve as the test set; the remaining trajectories are used for training. To compare models with different trajectory networks, the activity models $\{\phi_k\}$ are learned with smoothness constraint added.

1. In abnormality detection, a variational approach [55] is used to compute a lower bound of the data likelihood (10) in an iterative process. We assume that the number of iterations is small.
by the trajectory network. Once \( \{ \phi_k \} \) are learned and fixed, the negative log likelihood is computed on the test data ignoring the smoothness constraint.

First, we compare our approach with two alternatives: 1) unconnected network and 2) network with random correspondences. The former completely abandons the smoothing constraint, so it cannot jointly model the distributions of a single activity in multiple camera views. The latter simulates the case when correspondence is poor. Both alternatives result in higher negative log likelihood as shown in Table 1.

We also compare against models learned with trajectories from a single to all of the camera views. Models learned from a subset of the cameras will necessarily have lower negative log likelihood for trajectories within those

![Fig. 5. Distributions of activity models (1-6) and clusters of trajectories of the parking lot scene. When plotting the distributions of activity models (in the four red windows on the top), different colors are used to represent different moving directions: \( \rightarrow \) (red), \( \leftarrow \) (cyan), \( \uparrow \) (blue), and \( \downarrow \) (magenta). When plotting trajectories clustered into different activities (in the four green windows at the bottom), random colors are used to distinguish individual trajectories.](image)
camera views; however, they are limited to modeling joint activities only in a subset of the camera views. Our model captures joint activities in all cameras simultaneously and only exhibits a small increase in the negative log likelihood as shown in Table 2.

5.1.4 Temporal Threshold
The temporal threshold \( T \) in (1) determines the connectivity on the trajectory network. If a camera view \( A \) is disjoint from other views and it takes objects more than \( T \) seconds to cross the smallest gap between \( A \) and other views, then there is no way to extend a path in \( A \) to other views. If \( T \) is large and the scene is busy, the network will have too many “noisy” edges, which connect two trajectories actually corresponding to

| Table 1 Negative Log Likelihood under Our Approach and Two Alternative Trajectory Networks |
|-----------------------------------------------|-----------------|-----------------|-----------------|
| Parking Lot Street                           | Our approach    | Unconnected     | Random          |
| 130.3                                        | 200.3           | 176.8           |
| 85.7                                         | 228.8           | 135.2           |

Fig. 6. Distributions of activity models (1-6) and clusters of trajectories of the street scene. The meaning of colors is the same as Fig. 5.
different objects. Undersmoothing could lead to the same activity separated into different clusters, while oversmoothing could lead to different activities joining into the same cluster. Empirically, we achieved similar results with a wide range of values for $T$: for the street scene data set, good results are achieved when $T$ varies between 0 and 30 seconds; for the parking lot data set, the range of good values of $T$ is roughly from 3 to 15 seconds because the parking lot scene is busier and the view of camera 1 is disjoint from other camera views. There is quantitative evaluation of $T$ on a simulated data set in Section 5.5.

5.2 Correspondence

Although our activity analysis approach does not require correspondence among trajectories in different camera views, after the models of activities have been learned in an unsupervised way, they can help to solve the correspondence problem, since if two trajectories belong to the same activity and are connected by an edge, they are likely to correspond to the same object. For example, see Fig. 7. We pick a query trajectory from one of the camera views and mark it using green color and a star. All the trajectories in other camera views satisfying (1) are plotted in random colors. The red color and red stars mark the trajectories with the same activity category as the query trajectory. They are likely to correspond to the same object. So the information on activity category can dramatically reduce the search space when solving the correspondence problem. There is a quantitative evaluation in Section 5.5.2.

5.3 Abnormality Detection

In Figs. 8 and 9, we plot some trajectories with low data likelihoods, which have been normalized by the length of trajectories and are detected as abnormality from the parking lot scene and the street scene. All of the trajectories are sorted by abnormality and the top 30 are shown. Some very short trajectories most likely caused by tracking errors are not shown here. In the parking lot scene, most of the detected abnormal trajectories are pedestrians walking on the grass field. In the street scene, abnormal activities include pedestrians walking on the grass fields, pedestrians crossing the street, pedestrians walking in the middle of the street, and vehicles moving along a wrong lane.

5.4 Computational Cost

Running on a computer with 2 GHz CPU, it takes about two hours to learn the activity models from 22,951 trajectories of the parking lot data set and 40 minutes to learn the activity models from 14,985 trajectories from the street scene. When the activity models are learned and fixed, it takes less than 0.03 second to compute the likelihood of a trajectory in

---

**TABLE 2**

Negative Log Likelihood with Models Trained on a Variable Number of Cameras

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking Lot</td>
<td>120.9</td>
<td>121.3</td>
<td>122.8</td>
<td>123.3</td>
<td>425</td>
</tr>
<tr>
<td>Street</td>
<td>40.0</td>
<td>41.5</td>
<td>44.9</td>
<td>42.2</td>
<td>168</td>
</tr>
</tbody>
</table>

The test data are 200 trajectories from a single camera. The activity models in that camera are jointly learned with different numbers of cameras (from 1 to 4). The last column is a baseline model trained on data whose cluster labels of trajectories are randomly assigned.
order to detect abnormality and is much faster to label a new trajectory as some activity category.

5.5 Simulated Data

In order to quantitatively evaluate our algorithm, we simulate data used as the ground truth. As shown in Fig. 1c, we choose a scene that covers almost the same area of the street scene we used in Section 5.1.2. On a satellite image, we manually draw the fields covered by four camera views. The fields are convex four-sided polygons. These fields are converted into a standard camera view of size 240 × 360 through projective transformation. The views observed by four cameras are shown in Fig. 1b. We manually draw the central lines of eight paths on the satellite image (Fig. 10a) and simulate 8,000 trajectories. We assume that trajectories have almost the same speed, since speed does not play an important role in our algorithm. The starting points of trajectories are generated sequentially as follows:

\[ t_{s(i+1)} = t_{s(i)} + \Delta t_{i+1}, \]

\[ \Delta t_{i+1} \sim \text{Exponential}(\lambda). \]

\( t_{s(i)} \) is the starting time of the \( i \)th trajectory. The temporal difference \( \Delta t_{i+1} = t_{s(i+1)} - t_{s(i)} \) between two successive trajectories is sampled from an exponential distribution with mean \( \lambda \). A trajectory \( i \) is randomly assigned to one of the eight predefined activities \( k \). Trajectory \( i \) samples the location of its starting point from a Gaussian distribution centered at the starting point of path \( k \) with variance \( \sigma_1 \) (\( \sigma_1 = 5 \) in this simulation). Then \( i \) samples the remaining points sequentially with the velocity specified by path \( k \) and being added to Gaussian noise with variance \( \sigma_2 \) (\( \sigma = 2 \) in this simulation). The simulated trajectories in the global views and each of the four camera views are shown in Figs. 10b and 10c.

5.5.1 Learning Activity Models

\( \lambda \) is the parameter controlling how busy the scene is. When \( \lambda \) is smaller, more objects coexist in the scene at the same time, which means that there are more edges on the trajectory network and it is harder for our algorithm to jointly learn the models of activities in different camera views. In our experiments, we change the value of \( \lambda \) from 5 to 40 seconds. Based on the speed set for this experiment, the time an object spent to pass through a path varies from 170 to 410 seconds. It depends on the length of the path. When \( \lambda \) takes values from 5 to 40 seconds, the averaged number of objects coexisting in the scene varies from 57.5 to 7.1 (see Fig. 11). After the trajectories are clustered by our algorithm, we manually specify each of the eight clusters as an activity category, so each trajectory is assigned an activity label by our algorithm. By comparing with the ground truth, the accuracy of activity classification is computed. The accuracies when choosing different \( \lambda \) values are shown in Fig. 11. The accuracy is high (>97.8 percent) when \( \lambda \geq 30 \) seconds. The models of activities in a single global view and four camera views learned from the simulated data when \( \lambda = 30 \) seconds are shown in Figs. 12 and 13. Note that, when \( \lambda = 30 \) seconds, if we randomly sample a time point, there are around 9.6 objects coexisting in the scene, on average. Each trajectory is connected to 12.4 trajectories by edges on the network, on average. When \( \lambda \) decreases, some trajectories of different activities merge into one cluster. When \( \lambda = 5 \) seconds, the scene is very busy (there are 57.5 objects coexisting in the scene, on average), all of the trajectories are merged into one cluster and our algorithm cannot learn any useful activity models from this data set. Each trajectory is connected to 73.0 trajectories by edges on the network, on average.

We further look into the structure of the trajectory network constructed according to the temporal extents of trajectories in the data set when \( \lambda = 30 \) seconds. Fig. 14 shows the number of edges, which are related to different combinations of activities and camera views according to

![Fig. 11. The accuracies of classifying trajectories into different activities when \( \lambda \) takes different values and \( \gamma \) is fixed at 0.](image-url)
the ground truth. The entry of \((k_1, k_2)\) in the table of camera views \(i_1\) and \(i_2\) is the number of edges connecting two trajectories, one of which is in camera view \(i_1\) and belongs to activity \(k_1\), and the other is in camera view \(i_2\) and belongs to activity \(k_2\). There are six \(8 \times 8\) tables. As mentioned before, the edges on the trajectory network indicate possible correspondence candidates based on the temporal extents of trajectories. If the correspondence can be solved just using temporal information, all of the nonzero numbers in the table will be on diagonal. Actually, many off diagonal entries have nonzero numbers, which indicate false correspondences, whose ambiguity cannot be solved by only using temporal information. The ratio between the numbers of edges on diagonal and off diagonal is 0.2732. This ratio can be understood as signal-to-noise ratio in some sense. There are many more false correspondences than true correspondences. However, these false correspondences almost uniformly distribute among different combinations of activities and work as background noise. So if a trajectory of activity \(k_1\) is connected with another trajectory of activity \(k_2\), \(k_2\) is more likely to be the same as \(k_1\) than any one of the other activities. When the scene is busier, the signal-to-noise ratio is lower. When \(\lambda = 5\), the ratio is 0.1692 and our algorithm fails. Note that the signal-to-noise ratio is \(1/7 = 0.1429\), if trajectories are randomly connected without using any temporal information.

Fig. 15 plots the classification accuracies when \(\lambda\) is fixed at 40 seconds and the temporal threshold \(T\) in (1) changes from 0 to 300 seconds. The results stay at a high accuracy when \(T\) varies in a large range between 0 and 40 seconds. There is some interesting correlation between Figs. 11 and 15. The performance of our algorithm drops if there are too many edges on the trajectory network, which means that the "signal-to-noise" ratio is low. The number of edges increases if \(\lambda\) decreases, which means that the scene is busier and there are more objects coexisting in the scene, or \(T\) increases. In Fig. 11, when \(T\) is fixed at 0 second, \(\lambda = 30\) seconds seems to be a turning point on the accuracy curve. On average, there are around two more objects co-occurring when \(\lambda = 30\) seconds compared with \(\lambda = 40\) seconds. In Fig. 15, when \(\lambda\) is fixed at 40 seconds, \(T = 40\) seconds seems to be a turning point on the accuracy curve. Compared with \(T = 0\) second, the temporal window in (1) extends for \(2 \times T = 80\) seconds. In 80 seconds, there are around two more objects appearing, on average, when \(\lambda = 40\) seconds. So there are approximately the same number of edges under two settings. For \((\lambda = 30, T = 0)\), on average, each trajectory is connected to 12.4 trajectories by edges on the network, and for \((\lambda = 40, T = 40)\), this number is 13.0.

5.5.2 Using Activity Models to Solve the Correspondence Problem

As mentioned in Section 5.2, the learned activity models can help to solve the correspondence problem. We evaluate the performance on simulated data. When there are more than two camera views, the correspondence problem is NP-hard in the number of trajectories. Finding an approximate solution to this NP-hard problem is not the focus of this paper. So we demonstrate the capability of our activity models by doing correspondence among trajectories in two camera views. Given the distances between trajectories, correspondence of trajectories in two camera views can be solved by the Hungarian algorithm [56] in polynomial time.
The distance $D(a, b)$ between two trajectories $a$ and $b$ that are in different views are defined as follows. Each point on trajectories is assigned to one of the activities according to (9). Thus, each trajectory $j$ has a distribution $p_j$ over activities. If trajectories $a$ and $b$ satisfy the temporal constraint (1),

$$D(a, b) = \sum_{k=1}^{K} p_a(k) \log \left( \frac{p_a(k)}{p_b(k)} \right) + \sum_{k=1}^{K} p_b(k) \log \left( \frac{p_b(k)}{p_a(k)} \right),$$

which is Jensen-Shannon divergence; otherwise $D(a, b) = \infty$.

We choose camera views 1 and 4, which are shown in Fig. 1. One thousand trajectories are simulated and they are not in the data set of 8,000 trajectories used to learn the activity models. One thousand trajectories are observed in camera view 1 and around 880 trajectories are observed in camera view 4. Some trajectories of activity 7 (see Fig. 10) observed in camera view 1 have no corresponding trajectories in camera view 4. We simulate different sets of data by changing the parameter $\lambda$. The accuracies of correspondence are plotted in Fig. 16. It achieves very good correspondence accuracy (higher than 97 percent) when $\lambda \geq 30$. The accuracy drops when the scene is busier because of two reasons: the scene is busy or the temporal threshold $T$ is large. A large $T$ allows a large transition gap between camera views. So, if a scene is busy, the transition gap between cameras has to be small, which limits the topology of camera views in some sense. In this paper, only temporal information is used to build the trajectory network. That is why the algorithm is sensitive to how busy the scene is. Some other features, such as appearance, can also be used to eliminate some edges. If two objects observed in different camera views are poorly matched by appearance, their trajectories are not connected by an edge even though their temporal extents are close. Thus, activity models may be well learned even in a busy scene. However, in this case, the problem of matching appearance across camera views has to be addressed. It is a direction of our future study.

In this paper, only positions and moving directions are computed as features of trajectories, since in our application, they are enough to model paths of objects. Some other features, such as size of objects and speed, can also be
added into this framework. They can be used to cluster motion patterns into more categories, such as separating vehicles and pedestrians moving on the same path. However, the size of the codebook will increase quickly as more features are included, and thus, the computational cost will increase.

In our clustering method, the number of clusters $K$ has to be manually chosen. Some nonparametric models, such as Hierarchical Dirichlet Processes [57], [8], can learn the number of clusters from data. They could be used to improve our clustering method in the future work.

7 CONCLUSION

We propose a framework to model activities and cluster trajectories over a multicamera network. The models of activities can be used to detect scene structures. It is unsupervised and does not require first solving the challenging multicamera correspondence problem. When the activity models have been learned without supervision, it can help to solve the correspondence problem. Experiments on a simulated data set and two data sets including a very large number of trajectories demonstrate the effectiveness of this approach.
References


Xiaogang Wang received the BS degree in electrical engineering and information science from the University of Science and Technology of China in 2001, the MS degree in information engineering from Chinese University of Hong Kong in 2003, and the PhD degree from the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology in 2009. He is currently an assistant professor in the Department of Electronic Engineering at The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Kinh Tieu received the PhD degree in computer science from the Massachusetts Institute of Technology in 2006. His research interests include pattern recognition, computer vision, and machine learning. He has worked on image retrieval, visual surveillance, and medical imaging.

W. Eric L. Grimson received the BSc (hons) degree in mathematics and physics from the University of Regina in 1975 and the PhD degree in mathematics from the Massachusetts Institute of Technology (MIT) in 1980. He is currently the head of the Department of Electrical Engineering and Computer Science at MIT. He is the Bernard Gordon Professor of Medical Engineering in MIT’s Department of Electrical Engineering and Computer Science. He is a member of MIT’s Computer Science and Artificial Intelligence Laboratory and head of the Computer Vision Group. He also holds a joint appointment as a lecturer in radiology at Harvard Medical School and Brigham and Women’s Hospital. Prior to this position, he has served as an associate head of the Department of Computer Science and as education officer for the department. He has research interests in computer vision and medical image analysis. Since 1975, he and his Research Group have pioneered state-of-the-art methods for activity and behavior recognition, object and person recognition, image database indexing, site modeling, stereo vision, and many other areas of computer vision. Since the early 1990s, his Research Group has been applying vision techniques in medicine for image-guided surgery, disease analysis, and computational anatomy. He is a fellow of the IEEE and the AAAI, a member of the IEEE Computer Society, and a recipient of the Bose Award for Undergraduate Teaching at MIT.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.