# Patterns of Motion in Non-Overlapping Networks Using Vehicle Tracking Data



by

Chaowei Niu

B.S., University of Science and Technology of China (1999)
M.E., University of Science and Technology of China (2002)
M.S., Massachusetts Institute of Technology (2006)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010



Author .....
Department of Electrical Engineering and Computer Science
October 23, 2009

Certified by .....
W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

# Patterns of Motion in Non-Overlapping Networks Using Vehicle Tracking Data

by

Chaowei Niu

Submitted to the Department of Electrical Engineering and Computer Science
on October 23, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

## Abstract

We present a systematic framework to learn motion patterns based on vehicle tracking data captured by multiple non-overlapping uncalibrated cameras. We assume that the tracks from individual cameras are available. We define the key problems related to the multi-camera surveillance system and present solutions to these problems: learning the topology of the network, constructing tracking correspondences between different views, learning the activity clusters over global views and finally detecting abnormal events.

First, we present a weighted cross correlation model to learn the topology of the network without solving correspondence in the first place. We use estimates of normalized color and apparent size to measure similarity of object appearance between different views. This information is used to temporally correlated observations, allowing us to infer possible links between disjoint views, and to estimate the associated transition time. Based on the learned cross correlation coefficient, the network topology can be fully recovered.

Then, we present a MAP framework to match two objects along their tracks from non overlapping camera views and discuss how the learned topology can reduce the correspondence search space dramatically. We propose to learn the color transformation in $l\alpha\beta$ space to compensate for the varying illumination conditions across different views, and learn the inter-camera time transition and the shape/size transformation between different views. After we model the correspondence probability for observations captured by different source/sinks, we adopt a probabilistic framework to use this correspondence probability in a principled manner. Tracks are assigned by estimating the correspondences which maximize the posterior probabilities (MAP) using the Hungarian algorithm. After establishing the correspondence, we have a set of stitched trajectories, in which elements from each camera can be combined with observations in multiple subsequent cameras generated by the same object.

Finally, we show how to learn the activity clusters and detect abnormal activities using the mixture of unigram model with the stitched trajectories as input. We adopt a $bag-of-words$ presentation, and present a Bayesian probabilistic approach

in which trajectories are represented by a mixture model. This model can classify trajectories into different activity clusters, and gives representations of both new trajectories and abnormal trajectories.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering

# Acknowledgments

First of all, I would like to thank my advisor, Professor Eric Grimson, for his constant wise guidance through my graduate research, and for maintaining an open and interactive environment for me to thrive in. His ever-positive attitude toward work and life sets the standards that I always strive to achieve. Not only as a research mentor, Eric also gives me endless life support while I was having a baby. Without his understanding and encouragement, I could not have completed this work.

I also thank Professor Leslie Kaelbling and Professor Antonio Torralba for being on my thesis committee as readers and for their valuable comments and suggestions. Leslie also taught my first graduate class (AI) at MIT, which opened the door to a wonderful research life.

Many Thanks to Chris Stauffer for providing the tracker, Kinh Tieu for providing the map plotting code, Gerald Dalley for collecting videos for me. I would also like to thank the fellow graduate students in the group: Xiaogang Wang, Xiaoxu Ma, Biswajit Bose, Tomas Izo, Joshua Migdal. We had many discussions on life, the universe and everything, and on research too. These discussions were not only fun but also very productive.

Yajun, I want to thank you for those fun discussions about life and culture, for moon cakes I got from you every year. You have been such a good friend to me.

Big thanks go to Maysoon Hamdiyyah and Donna Kaufman for helping with making and canceling any appointments, answering all the questions and ensuring things could get done.

I want to thank my parents for their love and support, and especially for encouraging me all along to pursue my own independent development. Finally, thanks to my husband for his kind understanding and eternal support.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Video Surveillance

In recent years, governments, corporations, and public service organizations have spent increasing amounts of money on video surveillance systems to protect borders, critical infrastructure, public transportation, malls, office buildings, and parking lots. According to SearchSecurityAsia.com, research shows that the video surveillance market maintained a 10 percent growth despite the recession [1]. This rising interest also leads computer vision researchers to focus on visual surveillance applications, which has been one of the most active research topics in computer vision [2][3][4] in recent years.

Traditional, most visual surveillance depends on a human operator to sift through videos. It is a tiring, expensive, and tedious job, monitoring for unusual events that rarely occur. Experiments run at Sandia National Laboratories for the US Department of Energy study found that: "... such a task, even when assigned to a person who is dedicated and well-intentioned, will not support an effective security system. After only 20min, human attention to video monitors degenerates to an unacceptable level"[4] , see Fig. 1-1. The sheer volume of these data impedes easy human analysis, necessitating computer vision solutions to help automate the process and assist operators.

The goal of an automatic visual surveillance system is to detect all people and

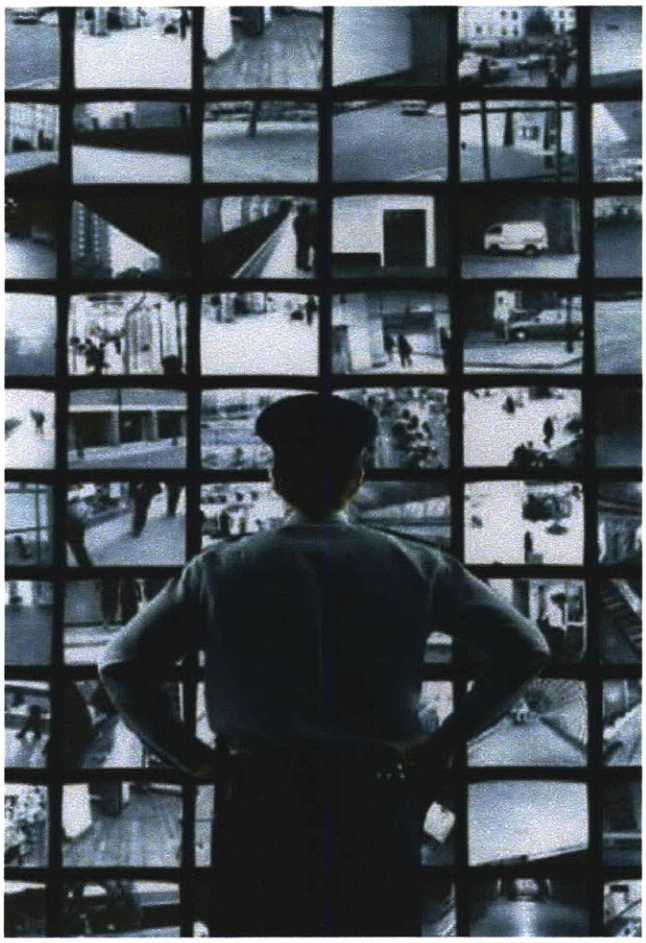

Figure 1-1: The spread of surveillance cameras leads to video information overload [4].

meaningful objects in the monitored area, track them over time, and infer all the relationships between them. It should be able to associate observations of an individual from videos taken days, months, or even years apart. It should be able to easily detect individual activities like running, excessive loitering, or entering unauthorized regions. It should also be able to detect activities involving multiple actors and/or objects such as theft, violence, surreptitious coordination, or chasing. Further, it should be able to characterize and detect larger scale events like crowd formation, a panic, or shifting traffic patterns. Based on where the interesting areas are, how many cameras are needed, and how far the cameras should be deployed from the monitored sites, the visual surveillance systems can be categorized as:

- *indoor* versus *outdoor*: Based on the system's purpose, cameras can be deployed indoor (such as stores, train stations, labs, shopping malls, etc.) or outdoor (such as parking lots, highways, airports, etc.). An indoor surveillance system is more focused on detecting, and recognizing people. For example, in some security sensitive locations such as a governmental unit, an indoor camera will be installed at the entrance. When somebody is entering, the surveillance system will obtain the visitor's features, such as height, facial appearance and walking gait from the video, and then decide whether the visitor can be cleared for entry. An outdoor surveillance system is more focused on detecting, and tracking moving objects (e.g. people and vehicles), and learning the long time activity types. For example, a highway surveillance system can monitor the traffic flow and the status of road congestion, which are of great importance for traffic management.

- *single camera* versus *multiple cameras*. Based on the system's purpose, either a single camera or multiple cameras will be used for surveillance. For the indoor surveillance example mentioned above, one camera could fulfill the task. However, the field of view (FOV) of a single fixed camera, or the field of a single moving Pan-Title-Zoom camera are limited in large environments. The straightforward goal of distributed systems with no overlapped cameras or

video sensor networks is to allow an extended view of the scene. With the growing availability of cheap sensors and processors, cooperative surveillance using multiple cameras could be used to ensure the security of an entire community. Considering a surveillance system located at a busy harbor, where are usually about 1,000 ships in the port at any one time, we not only want to monitor the dock areas, we also want to monitor the loading areas, and the ship activities on the sea. One single camera in this case couldn't fulfill the task. And in order to use minimum number of cameras with maximum coverage, multiple non overlapping cameras should be deployed.

- *near field* versus *far field*: Far-field tracking is primarily interested in extracting the position of many objects in situations where the objects are far from the sensor and usually do not visually interact. Here the most significant problems are determining the number of objects and learning correspondence between multiple observations of the same objects. Near-field tracking is primarily interested in modeling the configuration of the object over time. It is often assumed that there is at most one object and the object is always completely visible. For example, consider a site on campus monitoring system versus a face recognition system located at the entrance of a security building. For the first case, cameras are usually deployed up high to monitor the site, and the size of the moving objects is very small compared with their distances to the camera, which will be considered as a far field system. However, for the second case, in order to extract specific features such as facial expressions, a camera is usually placed very near the people to get a high resolution of the face, and will be treated as a near field system.

Sometimes, a visual surveillance system can belong to one or more of these categories. For example, an indoor surveillance system, might be an indoor, single camera, near field system. While for the highway surveillance system, it would be an outdoor, multiple cameras, far field system. Our thesis work is concerned with surveillance in an outdoor urban setting. In such cases, it is not possible for a single camera to

observe the complete area of interest because sensor resolution is finite and structures in the scene limit the visible areas. Thus multiple cameras are required to observe large environments. Even then it is usually not possible to completely cover large areas with cameras. The number of cameras required increases exponentially with the decrease in distance between the cameras. Therefore, there is a requirement for handling non-overlapping fields of view (FOV) of the cameras. In this thesis, we discuss and propose techniques for development of an automated non-overlapping multi-camera surveillance system for outdoor environments.

## 1.2 Non-overlapping Multi-camera Surveillance Tasks, Associated Challenges and Proposed Methods

A significant amount of research has been done on detecting, tracking and recognize moving objects of interest [5][6][7][8][10][11][18][12][13], and understanding and describing the behaviors of objects [14][76][82][77][78][87][83] for single camera surveillance systems. Besides these problems associated with single camera networks, there are some distinguished tasks that need to be solved for non-overlapping multi-camera networks.

### 1.2.1 Learning the Topology of the Network

Because of the development of technology, multi-camera visual surveillance applications are rapidly increasing in interest. Those applications include tracking moving objects throughout a set of views, classifying those moving objects into different categories (i.e. cars, people, animals), learning the network topology, getting statistics about the moving objects, and finally detecting and interpreting uncommon activities of the moving objects. In this thesis we are assuming tracking and classifying moving objects in a single scene have been solved. In such kinds of camera networks, to discover the relationship between the cameras is one of the most important issues in developing the intelligent surveillance system.

Consider the problem of wide-area surveillance, such as traffic monitoring and activity classification around critical assets (e.g. an embassy, a troop base, critical infrastructure facilities like oil depots, port facilities, airfield tarmacs). We want to monitor the flow of movement in such a setting from a large number of cameras, typically without overlapping fields of view (FOV). To coordinate observations in these distributed cameras, first we need to know the connectivity of movement between fields of view (i.e. when an object leaves one camera, it is likely to appear in a small number of other cameras with some probability). In some instances, one can carefully site and calibrate the cameras so that the observations are easily coordinated. In many cases, however, cameras must be rapidly deployed and may not last for long periods of time. It is preferable that the system does not require camera calibration. Also, maintaining calibration between a large network of sensors is a daunting task, since a slight change in the position of a sensor will require the calibration process to be repeated. Hence we seek a passive way of determining the topology of the camera network. That is, we want to determine the graph structure relating cameras, and the typical transitions between cameras, based on noisy observations of moving objects in the cameras.

If we can in fact determine the "blind" links (i.e. links that connect the disjoint views which cannot be observed directly) between camera sites, we can gather statistics about patterns of usage in this distributed camera setting. We can then record site usage statistics, and detect unusual movements. To determine the network topology and to answer these questions, we must first solve the tracking problem, i.e. we must maintain a moving object's identity from frame to frame, through the same camera view, through overlapping camera views, and through non-overlapping camera views, as shown in Figure 1-2. The bounding box shows the moving object. Within the field of view (FOV), vehicles tend to appear and disappear at certain locations. These locations may correspond to garage entrances, or the edge of a camera view, and have been called sources and sinks, respectively [15]. Based on the visible tracking trajectories, one can easily learn the links between each source and sink[16].

As we discussed above, tracking through the same view and through overlapping

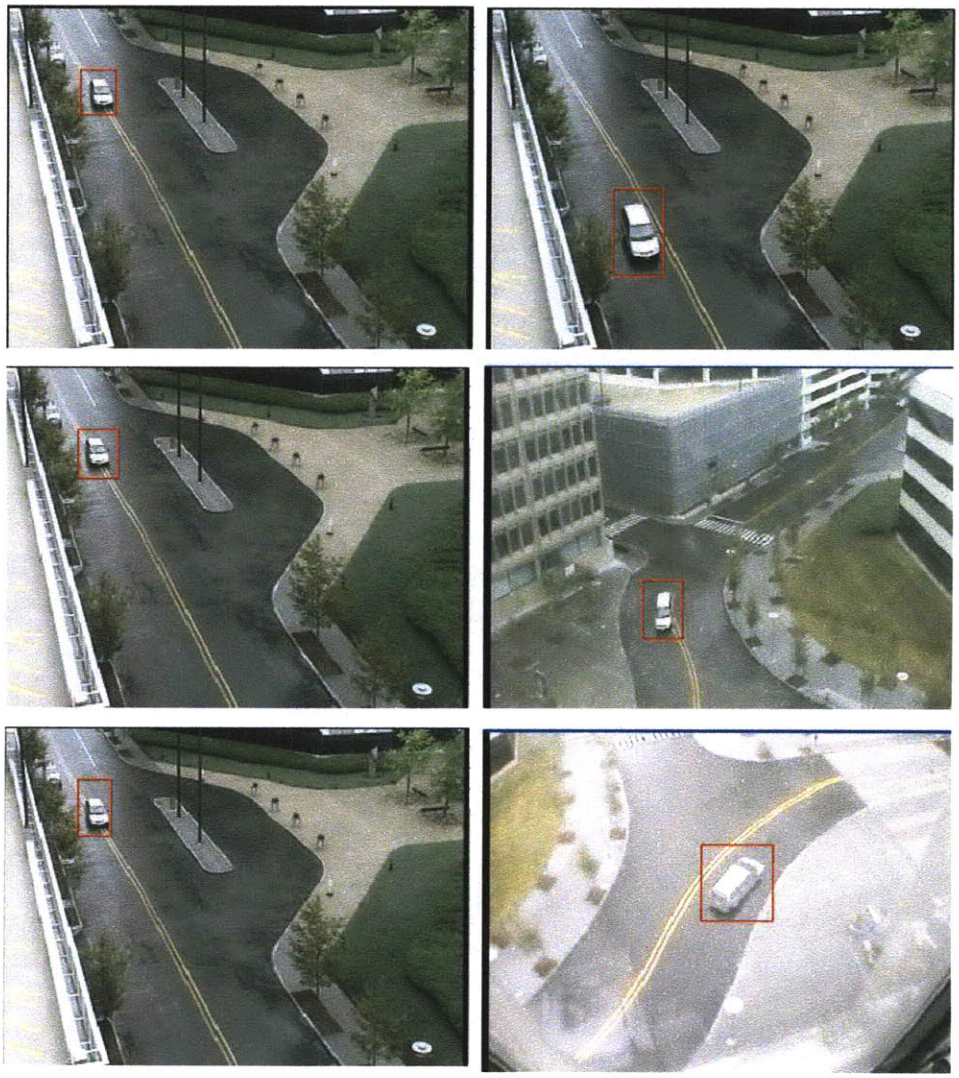

Figure 1-2: Tracking examples. The bounding box shows the moving object. The first row shows tracking through the same view, the middle row shows tracking through overlapping camera views, and the bottom row shows tracking through non-overlapping camera views.

views has been widely studied [5][6][7][8][10][17] [18] [19] [20]. However, little attention has been paid to the non-overlapping tracking correspondence problem. Good understanding of an activity requires knowledge of the trajectories of moving objects. For the field out of view, however, the tracking correspondences are unavailable, even the tracking trajectories are unavailable, which makes this problem harder. On the other hand, the information on how the cameras are connected (i.e. the prior probabilities of objects moving between cameras) will aid in solving the tracking correspondence through different views, which make the problem interesting and challenging.

After observing the input vehicle tracking data, we have found the following facts: first, physical characteristics (i.e. appearance) of moving objects do not change. Second, vehicles running on the same route roughly share the same speed and other trajectory characteristics. Finally, the trajectories of moving objects are highly correlated across non-overlapping views (i.e. vehicles are not randomly moving between different views. If people want to drive from one location to another location, they will usually chose the shortest and easiest path). With these observations in mind, we proposed a weighted cross correlation technique. First, an coarse appearance model is constructed by the combination of the normalized color and overall size model to measure the moving objects appearance similarity across the non-overlapping views. Then based on the similarity in appearance, votes are weighted to exploit the temporally correlating information between different views. The use of the moving objects' appearance similarity will help to narrow down the voting space for the correlation. From the learned correlation function, the possible links between disjoint views can be detected and the associated transition time can be estimated. Finally, based on the learned cross correlation coefficient, the network topology can be fully recovered.

### 1.2.2 Tracking through Different Cameras

The efficient tracking of multiple objects is a challenging and an important task in computer vision. It has many applications in surveillance, video communication and human computer interaction. The success of high lever system description (e.g. event detection, and trajectory interpretation) relies on accurate tracking of moving

objects throughout the camera networks. A typical vehicle surveillance system of far field wide areas requires the ability to track moving vehicles while observing them through multiple cameras with non-overlapping field of views, known as the vehicle re-identification problem. However, to re-establish a match of the same vehicle over different camera views located at different physical sites is a challenging problem because of the following reasons.

- First, the observations of an object are often widely separated in time and space, when viewed from non-overlapping non calibrated cameras. Thus, unlike conventional single camera tracking approaches, proximity in space and time cannot be used to constrain possible correspondences.

- Second, the appearance of an object in one camera view might be very different from its appearance in another camera view due to the differences in illumination, pose and camera properties.

- Third, in far-field settings, objects are small in size and the captured videos are of low resolution and poor quality. It is difficult to compute more complicated features, such as poses, gestures, and appearance of objects to facilitate the correspondence problem.

In order to deal with these problems, we assume that the tracks of individual cameras are available, and find the correspondences between these tracks in different cameras such that the corresponded tracks belong to same object in the real world. We propose a Maximum A Posteriori (MAP) estimation framework to combine multiple cues (e.g. space-time, color, and shape/size) to model transitions between different views for moving vehicles. The correspondence probability, i.e., the probability that two observations originate from the same object, depends on the following information:

- Color is a commonly used cue for tracking moving objects. As we will discuss in Chapter 2, the color distribution of an object is a function of scene illumination, object geometry, object surface material properties (e.g. surface albedo) and

camera parameters. Among all these, only the object surface material properties remain constant as the object moves across cameras. Thus, the observed color distribution of an object can be fairly different when viewed from two different cameras. We use comprehensive color normalization to compensate color change in different views to help solve the topology of the network. However, if we aim to construct exact correspondence between observations, this method doesn't suffice. We need to model the color transformation from one camera to another camera more precisely. Unfortunately, for a given pair of cameras, this transformation is not unique and also depends upon the scene illumination and camera parameters. In this chapter, we show that despite these large number of parameters, for a given pair of cameras, all such transformations can be modeled as a linear transformation in $l\alpha\beta$ space ($l\alpha\beta$ space is a color space in which each of the channels is decorrelated with others, see more details in Chapter 3). Then, we can fit an multivariate Gaussian model on parameters of the linear transformation for $l$, $\alpha$, and $\beta$ respectively. Hence, we can estimate the probability of the color transformation between cameras.

- Another commonly used cue for tracking is local features (i.e. edges, corners or SIFT features [21]). As we will discuss in Chapter 2, for far-field surveillance, however, even after successful detection, there are often very few image pixels per object, which makes it difficult to model the local feature change throughout cameras (see examples shown in Figure 1-3). However, we know for sure that a sedan in one scene cannot be a truck in another scene, which means overall size information still plays an important role in correspondence. Give the objects are fairly small in far field settings, it is unlikely that we will be able to recover the shape detail, so all we rely on is overall size measures. Here, we fit a best ellipse to the shape using minimum volume enclosing ellipsoid method [73][75][74], to model the shape/size change between different views. Alternatively, we could match the templates of the models, but in general, given the small image size of objects, the best ellipse suffices.

Figure 1-3: Tracking examples from different cameras. The tracking results are shown in their bounding boxes. The resolution is so low that it is difficult to model the local features throughout different cameras.

- The topology of the camera network we have learned (see Chapter 2) will aid in learning the correspondence problem. We already know that there exists an average transition time from one sink/source in one camera view to another sink/source in another camera view. This average transition time can be used to constrain correspondences. We refer to this cue as a space-time or an inter camera time cue. We propose to use a log normal distribution to model the transition time between different views which we will discuss in details in Chapter 3.

After we know how to model the correspondence probability for observations captured by different source/sinks, we adopt a probabilistic framework to use this correspondence probability in a principled manner. Once again, the information about the network topology will help us here. We only model the correspondence probability of observations that are from the connected source/sinks (i.e. from the learned topology, a pair of connected source/sinks means there exists a link between this pair of source/sinks), which will dramatically reduce the search space and resolve ambiguities arisen from similar observations presented by different objects. Tracks are assigned by estimating the correspondences which maximize the posterior probabilities (MAP). This is achieved by using Hungarian algorithm to solve the association matrix. After establishing the correspondence, we have a set of stitched trajectories,

in which elements from each camera can be combined with observations in multiple subsequent cameras generated by the same object.

### 1.2.3 Activity Analysis and Abnormal Event Detection

The final step is to automatically determine when an observed scene contains unusual or unexpected activity (for example, vehicle makes an illegal turn, people steal suitcases, etc.) which is the key and ultimate goal of any visual surveillance system. In the past, this task was mostly performed by a human expert: someone familiar with the scene who was able to recognize when something out of the ordinary occurred (e.g. human operators). In recent years, there has been a growing trend in both federal agencies and private firms to employ multiple video cameras for monitoring and surveillance purposes. The system's effectiveness and response is largely determined, not by the technological capabilities or placement of the cameras but by the vigilance of the person monitoring the camera system. The number of cameras and the area under surveillance is limited by the number of personnel available. Also even well trained people can't maintain their attention span for extended periods of time. Thus, machine vision systems are needed to mine the collected data for any potentially interesting activity. This has fostered a new area of machine vision research, often broadly referred as surveillance, aiming at the statistical modeling of scenes and activities.

Many methods have been introduced in the literature for event detection in surveillance videos from single camera views[76][82][77][78][87][83]. Multi-camera event detection still remains an emerging problem in real-life applications. Correlated activities across multiple camera views should be modeled collectively. This is because by utilizing visual evidence collected from different views, global activity modeling is more robust to noise and visual ambiguities than modeling activities separately within individual camera views. Hence not only motion patterns (such as the common path for moving objects) in single cameras interest us, but how are the objects moving between cameras (e.g. global motion patterns) interests us more. Especially, when our research concerns detecting unusual activities, we need to focus on how to

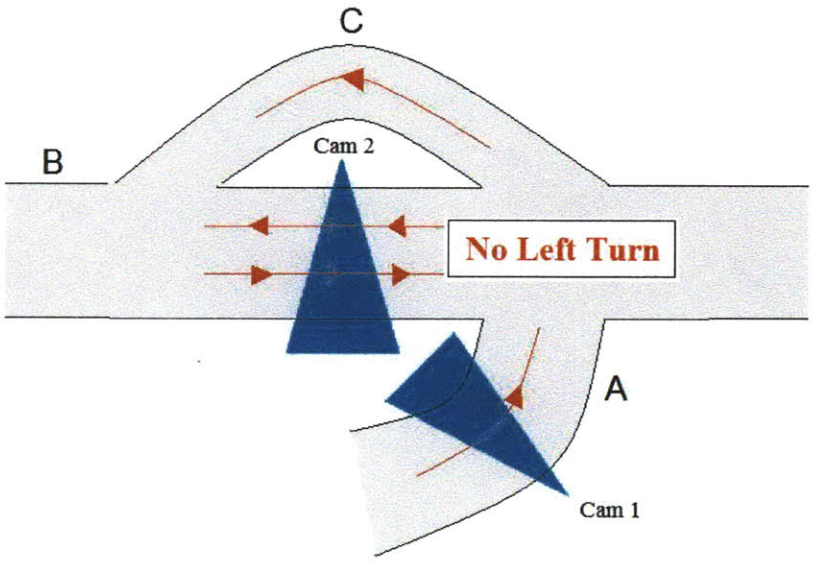


Figure 1-4: If a vehicle wants to move from Road A to Road B, it cannot make a left turn at A and B's intersection which is prohibited, and must move through Road C to Road B instead. Thus, an activity which a vehicle makes a direct left turn at A and B's intersection would be an abnormal activity and should be flagged. If we only study the activities captured by Camera 1 and 2 separately, this illegal activity will not get attention, because the segments of this activity in Camera 1 and 2 are perfect normal in each view. This example shows the necessity of studying the global motion patterns.

monitor activities of objects in multiple scenes without overlap of views. Sometimes, a combination of normal activities in their own view, does not necessarily lead to a normal activity. Figure 1-4 shows an example. If a vehicle wants to move from Road A to Road B, it cannot make a left turn at A and B's intersection which is prohibited, and must move through Road C to Road B instead. Thus, an activity which a vehicle makes a direct left turn at A and B's intersection would be an abnormal activity and should be flagged. If we only study the activities captured by Camera 1 and 2 separately, this illegal activity will not get attention, because the segments of this activity in Camera 1 and 2 are perfect normal in each view. This example shows the necessity of studying the global motion patterns.

As we discussed above, we assume we have already solved the correspondence problem, and stitched together the trajectories generated by every moving vehicle. Given these trajectories, the last part is to incorporate all the sensor information into the system (data fusion), learn the motion patterns, and be able to detect unusual events. More specifically, given the long time observation data, we want to model the overall scene (including the regions which cannot be covered by the sensors) which includes, learning the common trajectories for the different types of moving objects, building classifiers based on the trajectories and detecting anomalies. In our wide-scale network setting, the whole tracks have been segmented into different pieces of observed and un-observed tracks. Figure 1-5 gives one example. Rectangles show three different observed views, view A, B and C respectively. Track A has been divided into two observed segments, in View A and View B. Track B has been divided into two observed segments as well, in View A and View C. Given this situation, how can we compare these two tracks? It is not straightforward to extend from single camera to multiple cameras. If we have world coordinates which can be given by calibrated cameras, we can put these two tracks into the world coordinate system, and use the traditional distance measurements such as Euclidean distance [76], Hausdorff distance and its variations [77][78], and hidden Markov model [79], to group the trajectories into different activity categories by some standard clustering techniques such as spectral clustering [82], and graph-cuts [83]. However, if such information is

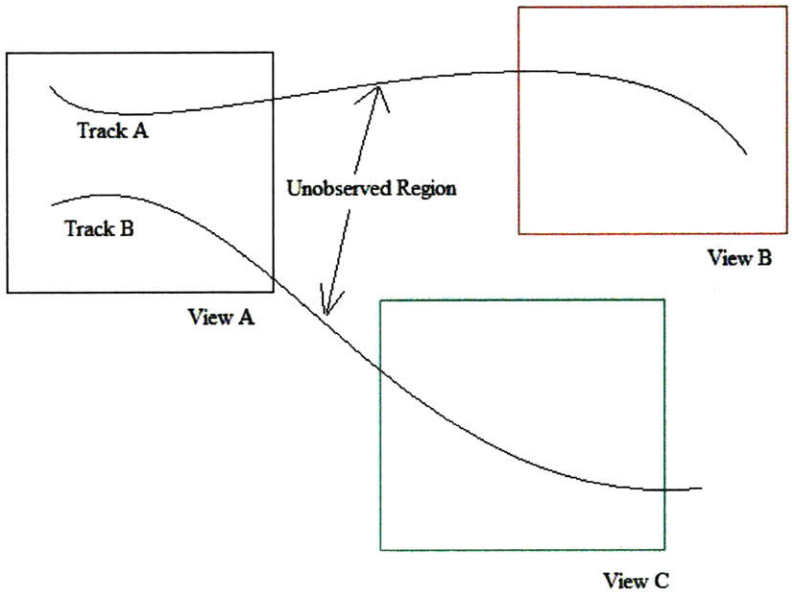


Figure 1-5: Examples of the whole tracks been segmented into different pieces of observed and unobserved tracks. Rectangles show three different observed views, view A, B and C respectively. Track A has been divided into two observed segments, in View A and View B. Track B has been divided into two observed segments as well, in View A and View C.

unavailable, distance measurements can not be applied directly.

In order to solve this problem, we adopt a $bag-of-words$ approach, and present a Bayesian probabilistic method in which trajectories are represented by a mixture model. This model can classify trajectories into different activity clusters, and gives a representation of both new trajectories and abnormal trajectories. First, our method defines a global codebook of observations that are representative of the entire set of observations captured by different cameras. Using this codebook to represent our continuous observations through different views, we can represent the likelihood of a trajectory by a mixture of unigram model. By using the EM algorithm, we can learn the model parameters, and label trajectories into different activity clusters. Finally, we can detect abnormal activities if they do not fit any learned activity model well.

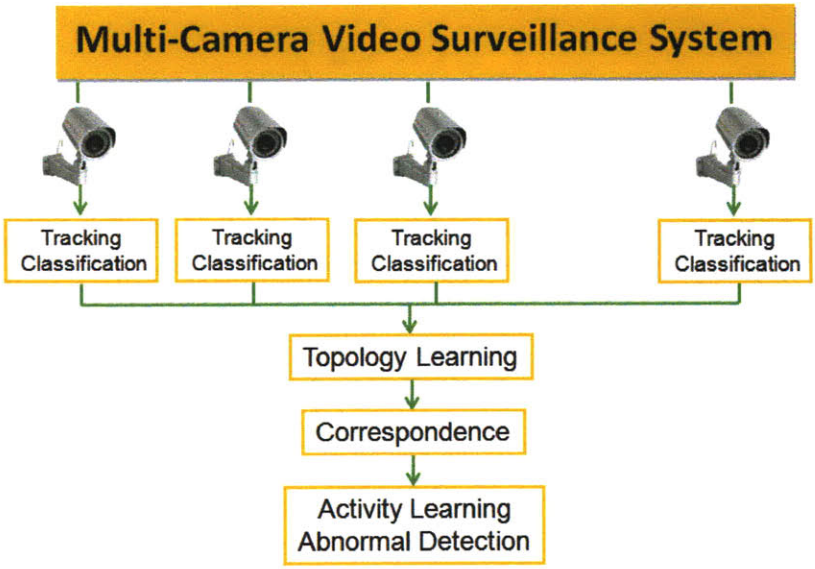


Figure 1-6: Proposed multiple non-overlapping camera network surveillance framework.

## 1.3 Contributions and Thesis Organization

In our thesis, we present a systematic framework to learn motion patterns based on vehicle tracking data captured by multiple non-overlapping uncalibrated cameras. Figure 1-6 gives an illustration about our proposed multiple non-overlapping camera network surveillance framework. We assume that the tracks of individual cameras are available. We define the key problems related to the multi-camera surveillance system and present solutions to these problems: learning the topology of the network, constructing tracking correspondence between different views, learning the activity clusters over global views, and finally detecting abnormal events.

In Chapter 2, we present a weighted cross correlation model to learn the topology of the network without solving correspondence in the first place. First, an coarse appearance model is constructed by the combination of the normalized color and overall size model to measure the moving object's appearance similarity across the non-overlapping views. Then based on the similarity in appearance, the votes for cross correlation are weighted to exploit the temporally correlating information between different views. The use of the moving objects' appearance similarity will help

to narrow down the voting space for the correlation. From the learned correlation function the possible links between disjoint views can be detected and the associated transition time can be estimated. Finally, based on the learned cross correlation coefficient, the network topology can be fully recovered.

In Chapter 3, we present a MAP framework to match two objects along their tracks from non overlapping camera views and discuss how the learned topology can reduce the correspondence search space dramatically. We propose to learn the color transformation in $l\alpha\beta$ space to compensate for the varying illumination conditions across different views, and learn the inter-camera time transition and the shape/size transformation between different views.

In Chapter 4, we show how to learn the activity clusters and detect abnormal activities using the mixture of unigram model with the stitched trajectories as input. We adopt a $bag-of-words$ representation, and present a Bayesian probabilistic approach in which trajectories are represented by a mixture model. This model can classify trajectories into different activity clusters, and gives a representation of both new trajectories and abnormal trajectories. First, our method defines a global codebook of observations that are representative of the entire set of observations captured by different cameras. Using this codebook to represent our continuous observations through different views, we can represent the likelihood of a trajectory by a mixture of unigram model. By using the EM algorithm, we can learn the model parameters, and label trajectories into different activity cluster. Finally, we can detect abnormal activities if they does not fit any learned activity model well.

We will discuss the limitations, future work and give concluding remarks in Chapter 5.

# Chapter 2

# Topology Learning

Because of the development of technology, multi-camera visual surveillance applications are rapidly increasing in interest. Among these applications, discovering the relationship between the cameras is one of the most important issues. This Chapter will discuss how to learn the network's topology using a weighted cross correlation method.

## 2.1 Background Research

Most intelligent video surveillance systems deploy a network of cameras to monitor a wide-area scene, e.g. underground station, airport, or shopping complex. In order to use a minimum number of cameras, these cameras are usually deployed without overlapping views. For global activity monitoring and situation awareness, it is crucial to detect and model correlations among object activities observed across camera views. Specifically, discovering multi-camera activity correlations will lead to understanding of both the spatial topology (i.e. between-camera spatial relationships) and more importantly the temporal topology of a camera network, that is, we wish to discover if an activity takes place in one camera view, what other activities it may cause in different camera views and after what time delay. Discovering and modeling such activity correlations among multiple camera views directly from data can facilitate moving object re-identification across disjoint camera views and global activity

analysis.

One possible approach to learn the connectivity or spatial adjacency of the camera network is to use calibrated camera networks [22] [23]. Jain et al. [22] used calibrated cameras and an environmental model to obtain the 3D location of a person. Collins et al. [23] developed a system consisting of multiple calibrated cameras and a site model, and then used region correlation and location on the 3D site model for tracking. This kind of method usually requires detecting the same landmarks with known 3D coordinates from different cameras and using a complex site model.

Another possible approach is to solve the tracking correspondence problem directly. Ali et al.[24] uses MAP estimation over trajectories and camera pose parameters to calibrate and track with a network of non-overlapping cameras. Huang and Russell [25] present a Bayesian foundation for computing the probability of identity, which is expressed in terms of appearance probabilities. Their appearance model is treated as the product of several independent models, such as: lane of traffic, size, color and arrival time. They have used a simple Gaussian model to measure the transition probability between two disjoint views.

Javed et al. [26] adopted Huang and Russell's method[25] and used Parzen windows to estimate the inter-camera space-time (i.e., transition time between two views) probabilities and then solved the correspondence problem by maximizing the posterior probability of the space-time and appearance.

Kang et al. [27] used a combination of appearance model and motion model to track the moving objects continuously using both stationary and moving cameras, then learned the homography between the stationary cameras, the moving cameras, and finally estimated the affine transformation.

The above methods require that we establish the correspondence for individual tracks between non-overlapping views. The correspondence assignment problem can be found in time $O(n^3)$ by formulating the problem as a weighed bipartite graph matching problem (i.e. finding maximum weight pathes in the graph), which is difficult and time consuming. However, appearance information between different views is still quite useful and should not be discarded.

Other approaches to estimate the spatio-temporal information use statistical models[28] [29] [30] [31]. Petty et al.[29] proposed to estimate transition time from aggregate traffic parameters in a freeway scenario. Westerman et al. [30] used cumulative arrivals at successive detector sites to estimate vehicle arrivals. Ellis[31] proposed a two stage algorithm to learn the topology: first detecting entry and exit zones in each view, then temporally correlating the disappearance and reappearance of tracked objects between those views to detect possible links. For these statistical methods, the performance is only based on information of appearing and leaving times of the detected moving objects at each soure/sink. It will not perform well under heavy traffic conditions.

In this thesis, we first focus on how to learn the non-overlapping network topology, which means to detect the possible "blind" links between disjoint views, and how to determine the transition time (i.e., the time between disappearing at one location and reappearing at the other location). Our learning is based on the following observations:

1. Physical characteristics of moving objects should not change. For example, a red sedan in one view is still a red sedan in another disjoint view, it cannot become a white SUV.

2. Vehicles running on the same route roughly share the same speed and other trajectory characteristics. Based on real road traffic, most vehicles on roads are just following traffic. They will slow down and stop with a red light and will speed up when the green light turns on. This will make the assumption that the transition time from one location to another location is Gaussian distributed reasonable.

3. The trajectories of moving objects are highly correlated across non-overlapping views (i.e. vehicles are not randomly moving between different views). To be more illustrative, suppose a vehicle wants to go from location A to location C through location B. It will go directly from A to B and then to C, instead of

doing loops between A and B (i.e. from A to B, then to A, then to B) and finally goes to C.

## 2.2 Cross Correlation

In statistics, the term cross correlation is sometimes used to refer to the covariance $cov(X, Y)$ between two random vectors $X$ and $Y$, in order to distinguish that concept from the "covariance" of a random vector $X$, which is understood to be the matrix of covariances between the scalar components of $X$.

In signal processing, the cross correlation (or sometimes "cross-covariance") is a standard method of estimating the degree to which two series are correlated, commonly used to find features in an unknown signal by comparing it to a known one[32]. Consider two discrete series $x(i)$ and $y(i)$ where $i = 0, 1, 2...N - 1$. The cross correlation $R$ at delay $d$ is defined as:

$$R(d) = \sum_{i=0}^{i=N-1} x_i * y_{i+d} \tag{2.1}$$

If the above is computed for all delays d=0,1,2,...N-1 then it results in a cross correlation series of twice the length as the original series.

There is the issue of what to do when the index into the series is less than 0 or greater than or equal to the number of points ($i + d < 0$ or $i + d \geq N$). The most common approaches are to either ignore these points or assuming the series $x$ and $y$ are zero for $i < 0$ and $i \geq N$. In many signal processing applications, the series is assumed to be circular in which case the out of range indexes are "wrapped" back within range, ie: $x(-1) = x(N - 1), x(N + 5) = x(5)$ etc. The range of delays d and thus the length of the cross correlation series can be less than N, for example the aim may be to test correlation at short delays only.

### 2.2.1 Cross Correlation Model

As mentioned in the Introduction, there are two observations: transition time from one location to another location is Gaussian distributed; and the trajectories of moving objects are highly correlated across non-overlapping views. Under these two observations, we can see that the sequences of appearing vehicles under the connected cameras (i.e. there exist routes directly connecting those cameras) are highly correlated. Since the cross correlation function can capture the degree of correlation between two signals, we present a simple cross-correlation model to estimate the existence of possible blind links and the associated transition time between different cameras.

For each traffic source/sink (i.e. locations where objects tend to appear in a scene and locations where objects tend to disappear from a scene), traffic can be represented as a discrete flow signal $V_i(t)$, which is defined as the list of observations (see Figure 2-2) appearing in a time interval around time $t$ at source/sink $i$. For each observation, there are time, location and appearance information associated with it.

The cross-correlation function between signals $V_i(t)$ and $V_j(t)$ can indicate the possibility of a link, and be used to estimate the transition time if there exists such a link:

$$R_{i,j}(T) = \sum_{t=-\infty}^{t=\infty} ||V_i(t)|| * ||V_j(t+T)|| \tag{2.2}$$

If there is a possible link between source/sink $i$ and $j$, there should exist a clear peak in $R_{i,j}(T)$ at time $T = t$, where $t$ denotes the typical transition time from location $i$ to location $j$. In this sense, a possible "blind" link from location $i$ to location $j$ has been learned.

However, there are some limitations to this method. For example, it would not perform well under heavy traffic conditions. To illustrate this problem, we present an extreme situation as shown in Figure 2-1. Suppose at source/sink A, an yellow school bus leaves every 5 minutes starting at 8am, while at source/sink B, a blue police car appears every 5 minutes starting 8:01am, and there is no possible link between A and

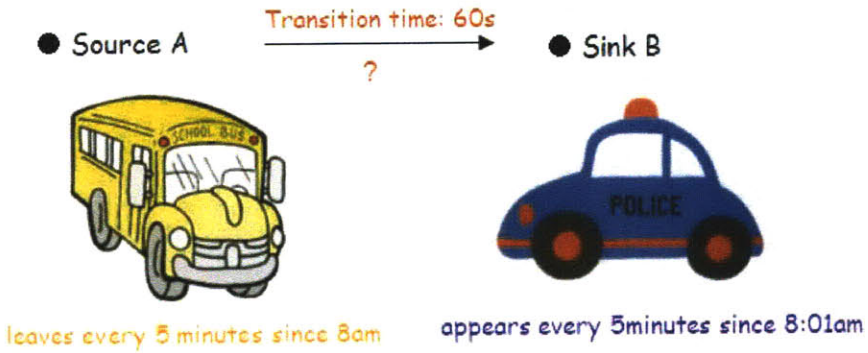


Figure 2-1: Example of the case which cross correlation doesn't work

B. However, if we use the cross correlation method directly, a possible link will be learned and the learned transition time would be 60 seconds.

Intuitively, at different source/sinks, only those observations which look similar in appearance should be counted to derive the spatio-temporal relation. In order to fix this problem, we propose a weighted cross correlation technique (i.e. a technique that counts appearance similarity information in it to narrow down the voting space), and will discuss the details next.

## 2.3 Coarse appearance model

In order to measure appearance similarity between different views, first we need to select what appearance features to use, and then to construct an appearance model. The far field vehicle tracking system we have been using is provided by Chris Stauffer[18]. The input to the tracking system is the video sequence, and the output of the tracking system is a set of tracking sequences, where each track is a sequence of observations of the same object (supposely) in the field of view. These tracks are provided as input to our topology learning system. Some sample observations are shown in Figure 2-2.

In different views, the same object can appear dramatically different, not only the size, but the color as well. In order to relate the appearance of an object from view to view, the appearance model (i.e. color model, and size model) should be

Figure 2-2: Examples of observations captured at different views.

learned first. Learning the appearance model is carried out by assuming that there exists some known correspondences between disjoint views. One way to achieve the correspondence is by driving the same car around the environment. Another possible way is to manually detect interesting vehicles (i.e. yellow cab, Fedex truck, blue bus) across the disjoint views. Since we only need to model color and overall size, unlike the traditional appearance-based correspondence method, which requires a significant amount of known correspondence, only some small number of the best matches are needed in the training phase.

## 2.4 Normalized Color Model

Various methods have been proposed to model the color change of moving objects from one camera to another. For far-field vehicle surveillance, since a vehicle is the only moving object and usually contains one color, a single color model per vehicle would be sufficient. However, under different views, the same color may appear dramatically different due to the lighting geometry and illuminant color. Based on this consideration, we adopt a normalized color model. First, we use the comprehensive color normalization(CCN) algorithm proposed by Finlayson et al. [33] to reprocess the input color images.

### 2.4.1 Comprehensive Color Normalization Algorithm

This section gives details about comprehensive normalization algorithm and how this algorithm works in our vehicle tracking data. The normalization algorithm is directly

replicated from the work of Finlayson [33]. To motivate this method, and to indicate why it is well suited to our problem, we begin with a description of the method, which we paraphrase from Finlayson et al. [33].

The light reflected from a surface depends on the spectral properties of the surface reflectance and of the illumination incident on the surface. In the case of Lambertian surfaces, the light is simply the product of the spectral power distribution of the light source with the percent spectral reflectance of the surface. Assuming a single point source light, combining together with illumination, surface reflection and sensor function, forms a sensor response:

$$\bar{p}^{\hat{x},E} = \bar{e}^x \cdot \bar{n}^x \int_\omega S^x(\lambda) E(\lambda) \bar{F}(\lambda) d\lambda \tag{2.3}$$

where $\lambda$ is wavelength, $\bar{p}$ is a 3-vector of sensor responses ($rgb$ pixel value), $\bar{F}$ is the 3-vector of response functions (red, green and blue sensitivity), $E$ is the illumination striking surface reflectance $S^x$ at location $x$. Integration is over the visible spectrum $\omega$. Bar denotes vector quantities. The light reflected at $x$ is proportional to $E(\lambda)S^x(\lambda)$ and is projected onto $\bar{x}$ on the sensor array. The precise power of the reflected light is governed by the dot-product term $\bar{e}^x \cdot \bar{n}^x$. Here, $\bar{n}^x$ is the unit vector corresponding to the surface normal at $x$ and $\bar{e}^x$ is in the direction of the light source. The length of $\bar{e}^x$ models the power of the incident light at $x$. Note that this implies that the function $E(\lambda)$ is actually constant across the scene. Substituting $\bar{q}^{x,E}$ for $\int_\omega S^x(\lambda)E(\lambda)\bar{F}(\lambda)$ allows us to simplify the above formula into:

$$\bar{p}^{\hat{x},E} = \bar{q}^{x,E} \bar{e}^x \cdot \bar{n}^x \tag{2.4}$$

It is now understood that $\bar{q}^{x,E}$ is that part of a scene that does not vary with lighting geometry (but does change with illuminant color). Equation 2.4, which deals only with point-source lights is easily generalized to more complex lighting geometries. Suppose the light incident at $x$ is a combination of $m$ point source lights with lighting direction vectors equal to $\bar{e}^{x,i}$ $(i = 1, 2, \cdots, m)$. In this case, the camera response is equal to:

$$\bar{p}^{\hat{x},E} = \bar{q}^{x,E} \sum_{i=1}^{m} \bar{e}^{x} \cdot \bar{n}^{x} \tag{2.5}$$

Of course, all the lighting vectors can be combined into a single effective direction vector:

$$\bar{e}^{x} = \sum_{i=1}^{m} \bar{e}^{x,i} \Rightarrow \bar{p}^{\hat{x},E} = \bar{q}^{x,E} \bar{e}^{x} \cdot \bar{n}^{x} \tag{2.6}$$

This equation conveys the intuitive idea that the camera response to $m$ light sources equals the sum of the responses to each individual light. Since we now understand the dependency between camera response and lighting geometry is a scalar relationship dependent on $\bar{e}^{x} \cdot \bar{n}^{x}$, it is straightforward to normalize it:

$$\frac{\bar{p}^{\hat{x},E}}{\sum_{i=1}^{3} p_i^{\hat{x},E}} = \frac{\bar{q}^{x,E} \bar{e}^{x} \cdot \bar{n}^{x}}{\sum_{i=1}^{3} \bar{q}_i^{x,E} \bar{e}^{x} \cdot \bar{n}^{x}} = \frac{\bar{q}^{x,E}}{\sum_{i=1}^{3} \bar{q}_i^{x,E}} \tag{2.7}$$

when $\bar{p}^{\hat{x},E} = (r, g, b)$ then the normalization returns: $(\frac{r}{r+g+b}, \frac{g}{r+g+b}, \frac{b}{r+g+b})$.

Hence, we can define function $R()$:

$$R(I)_{i,j} = \frac{I_{i,j}}{\sum_{k=1}^{3} I_{i,k}} \tag{2.8}$$

where $I$ is an $N \times 3$ image matrix with $N$ image pixels, whose columns contain the intensity of 3 RGB color channels.

Let us now consider the effect of illuminant color. If we hold lighting geometry, the vectors $\bar{e}^{x}$, fixed and assume the camera sensors are delta functions: $F(\lambda) = \delta(\lambda - \lambda_i),\ i = (1, 2, 3)$. Under $E(\lambda)$ the camera response is equal to:

$$\bar{p}_i^{\hat{x},E} = \bar{e}^{x} \cdot \bar{n}^{x} \int_{\omega} S^{x}(\lambda) E(\lambda) \delta(\lambda - \lambda_i) d\lambda = \bar{e}^{x} \cdot \bar{n}^{x} S^{x}(\lambda_i) E(\lambda_i) \tag{2.9}$$

and under a different $E_1(\lambda)$:

$$\bar{p}_i^{\hat{x},E_1} = \bar{e}^{x} \cdot \bar{n}^{x} \int_{\omega} S^{x}(\lambda) E_1(\lambda) \delta(\lambda - \lambda_i) d\lambda = \bar{e}^{x} \cdot \bar{n}^{x} S^{x}(\lambda_i) E_1(\lambda_i) \tag{2.10}$$

Combining the above two equations together we can get:

$$\bar{p}_i^{\hat{x},E_1} = \frac{E_1(\lambda_i)}{E(\lambda_i)} \bar{p}_i^{\hat{x},E} \tag{2.11}$$

This equation informs us that, as the color of light changes, the values recorded in each color channel scale by a factor (one factor per channel). It is straightforward to remove the image dependence on illuminate color by function $C()$:

$$C(I)_{i,j} = \frac{N/3 I_{i,j}}{\sum_{k=1}^{N} I_{k,j}} \tag{2.12}$$

where $I$ is an $N \times 3$ image matrix with $N$ image pixels, whose columns contain the intensity of 3 RGB color channels. The $N/3$ here is to ensure that the total sum of all pixels after the column normalization is N which is the same as that after the row normalization. The comprehensive normalization procedure is defined as a loop:

1. $I_0 = I$

2. do $I_{i+1} = C(R(I_i))$ until $I_{i+1} = I_i$

where function $R(I)$ is defined in Equ.2.8, and function $C(I)$ is defined in Equ.2.12 Note that after the iteration, we get a lighting geometry and illuminant color independent image [33].

We applied this comprehensive color normalization algorithm to our vehicle tracking data. Two examples are shown in figure 2-3. Because HSV color model is more similar to the way humans tend to perceive color, examples are shown in the HSV color model. (a) is the two observations the same vehicle from two different views. (b) is the color histograms before comprehensive color normalization. (c) is the color histograms after the color normalization. Because of the huge illuminate difference between the two views, we can see that the two histograms for Hue and Saturation are quite different. After the normalization, however, the histograms for Hue and Saturation match well.

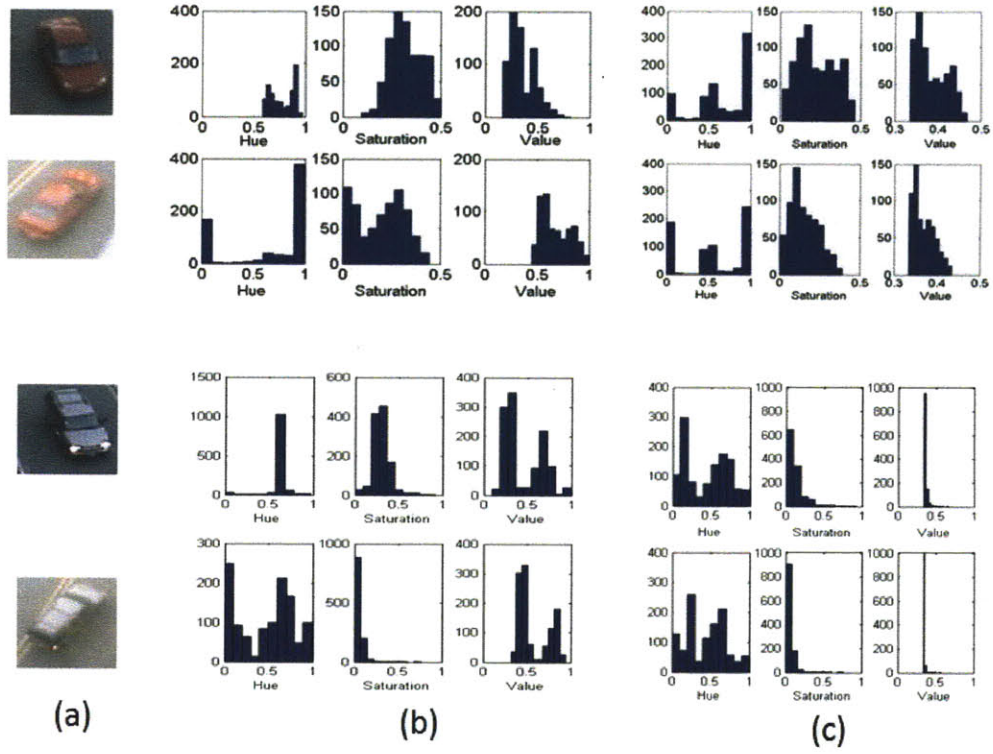


Figure 2-3: (a) are the two examples of two observations of the same vehicle from two different views. (b) are the color histograms before comprehensive color normalization. (c) are the color histograms after the color normalization. Because of the huge illuminate difference between the two views, we can see that the two histograms for Hue and Saturation are quite different. After the normalization, however, the histograms for Hue and Saturation match well.

### 2.4.2 Color Model

After the color normalization procedure, we can define the color change$P_{color}$ throughout any two different scenes in terms of the quadratic distance between the normalized histograms of the observations [40]:

$$Dist(hist1, hist2) = |hist1_i - hist2_i| \times sim(i,j) \times |hist1_j - hist2_j| \qquad (2.13)$$

where $i$ and $j$ are indexes into the two histograms (hist1 and hist2). $sim(i,j)$ denotes the similarity between the quantized colors, represented by the histogram indexes

$i$ and $j$. The color histogram quadratic distance is summed for all of the possible combinations of $i$ and $j$ (depending on the histogram size), in order to obtain an overall measure of histogram difference. Although this computation is somewhat expensive, it allows for the testing of color closeness, as opposed to merely considering exact color matches.

Then we can fit a multivariate Gaussian distribution on the quadratic distance for H and S channels, to model the color change $P_{color}$:

$$\begin{aligned} P_{color} &= P(h^{c_1}, s^{c_1}, h^{c_2}, s^{c_2} | O^{c_1} = O^{c_2}) \\ &= N_{\mu_{h,s}, \Sigma_{h,s}}(h^{c_1} - h^{c_2}, s^{c_1} - s^{c_2}) \end{aligned} \tag{2.14}$$

where $c_1$, $c_1$ are the camera 1 and 2. $O^{c_1}, O^{c_2}$ are the detected observation under camera 1 and camera 2 respectively. $h, s$ are H and S information included in the observation. $\mu_{h,s}$ and $\Sigma_{h,s}$ are the mean and variance respectively. And $O^{c_1} = O^{c_2}$ means those two observations are actually generated by the same object. For each pair of different views, there is a multivariate Gaussian distribution associated with it.

## 2.5 Size Model

For far-field surveillance, even after successful detection, there are often very few image pixels per object, which makes it difficult to model the shape change between cameras. However, we know for sure that a sedan in one scene cannot be a truck in another scene, which means overall size information still plays an important role in correspondence. Here we use width and length of the bounding box to measure the overall size. This estimate of size is somewhat simplistic. However, given that objects are fairly small in far field settings, it is unlikely that we will be able to recover the shape detail, so all we rely on is overall size measures. Ideally, we should fit a best ellipse to the shape, to account for orientation relative to the camera, but in general given the small image size of objects, we find width and length to suffice.

We also adopt a multivariate Gaussian distribution to model the size change $P_{size}$.

$$
\begin{aligned}
P_{size} &= P(w^{c_1}, l^{c_1}, w^{c_2}, l^{c_2} | O^{c_1} = O^{c_2}) \\
&= N_{\mu_{w,l}, \Sigma_{w,l}}(w^{c_1} - w^{c_2}, l^{c_1} - l^{c_2})
\end{aligned}
\tag{2.15}
$$

where $w^{c_1}$,$l^{c_1}$ are the detected vehicle's width and length under camera 1. $\mu_{w,l}$ and $\Sigma_{w,l}$ are the mean and variance respectively. The imaging transformation of a perspective camera leads to distortion of a number of geometric scene properties. As a result, objects appear to grow larger as they approach the camera center and become smaller when they are far away from the camera[34]. So in the sense of simple normalization, the average size over the whole trajectory has been adopted, when we do the size model.

## 2.6 Joint Probability Model

Given two observations $o_a^i$ and $o_b^j$, where $o_a^i$ is the observation $a$ from camera $i$ and $o_b^j$ is the observation $b$ from camera $j$, the similarity in appearance between those two observations can be calculated as the probability that these two observations are actually generated by the same object, which is called "appearance probability", denoted by $P(o_{a,i}, o_{b,j} | a = b)$. It is important to note that the appearance probability is *not* the probability $a = b$.

Assuming that color and size information of each observation is independent, the similarity in appearance between two observations can be described as the product of the color and size similarity:

$$
\begin{aligned}
&P_{similiarity}(o_{a,i}, o_{b,j}) \\
&= P(o_{a,i}, o_{b,j} | a = b) \\
&= P(color_{a,i}, color_{b,j} | a = b) P(size_{a,i}, size_{b,j} | a = b) \\
&= P_{color} P_{size}
\end{aligned}
\tag{2.16}
$$

Now we know how to model the appearance change of objects from view to view, and how to measure the similarity in appearance for two observations. This result will be used to help explore the statistical spatio-temporal information (see next section).

## 2.7 Weighted cross correlation

After we know how to measure the appearance similarity of objects from different views, we can count this information into the cross correlation function, and we name it a weighted cross correlation function.

The weighted cross correlation technique is defined as :

$$R_{i,j}(T) = \sum_{t=-\infty}^{t=\infty} \sum_{O_{a,i} \subseteq V_i(t)} \sum_{O_{b,j} \subseteq V_j(t+T)} P_{similarity}(O_{a,i}, O_{b,j}) \tag{2.17}$$

Specifically, for a pair of disappearing vehicles at source/sink $i$ at time $t$ and appearing vehicles at source/sink $j$ at time $t+T$, calculate the similarity in appearance between those two observations and update $R_{i,j}(T)$. Then peak values can be detected using the threshold estimated as:

$$threshold = mean(R_{i,j}(T)) + w * std(R_{i,j}(T)) \tag{2.18}$$

where $w$ is a user-defined constant.

In this work, we assume there is only one popular transition time if there is a link between $i$ and $j$. People in real life tend to choose the shortest path between the start location and the destination, which makes the single transition time reasonable with the assumption of constant velocity. Although we assume there is only one popular transition time between two disjoint views, this weighted cross correlation model can be applied to the cases with multiple transition times which will result in multiple peaks in $R(T)$. For our implementation, transition time is assigned with the time associated with the highest detected peak. Figure 2-4 gives an example when weighted cross correlation can detect a valid link, while general cross correlation fails. After applying the general and weighted cross correlation function on the data from two cameras located at an intersection, the results are shown in Figure 2-4 (a) and (b), respectively. (b) has a clear peak which suggests a possible link with transition time 11 seconds between those cameras, which (a) does not.

In this part, we learned how to use the weighted cross correlation model to estimate

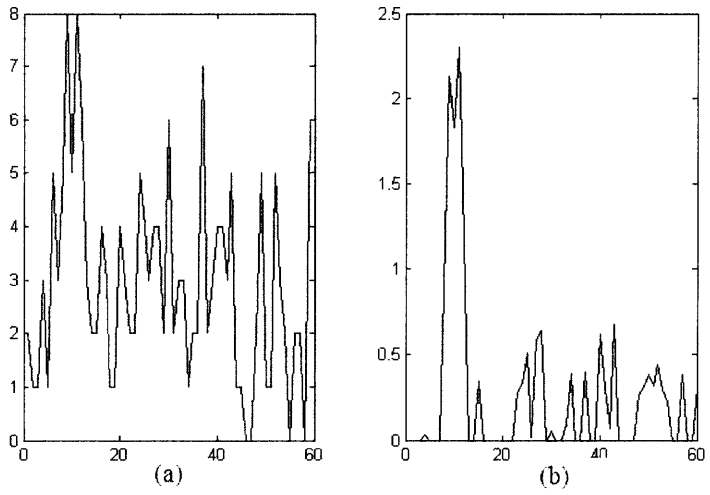


Figure 2-4: After applying the general and weighted cross correlation function on the data from two cameras located at an intersection, the results are shown in Figure (a) and (b), respectively. (b) has a clear peak which suggests a possible link with transition time 11 seconds between those cameras, which (a) doesn't.

the possible blind links and the associated transition time between disjoint views. We will present experimental results in the next section using both real tracking data and synthetic tracking data.

## 2.8 Experiments and Problems

In order to evaluate the proposed weighted cross correlation method, we have tested it both on real data and synthetic data.

### 2.8.1 Real Data

For the real data experiment, we used three non-overlapping cameras distributed around several buildings. The layout of the environment and the cameras' location are shown in Figure 2-5. For each camera, we have 1 hour of vehicle tracking data obtained from a tracker based on [18] every day for six days. There are total of 213 observations in camera(a), 1056 observations in camera (b), 1554 observations in

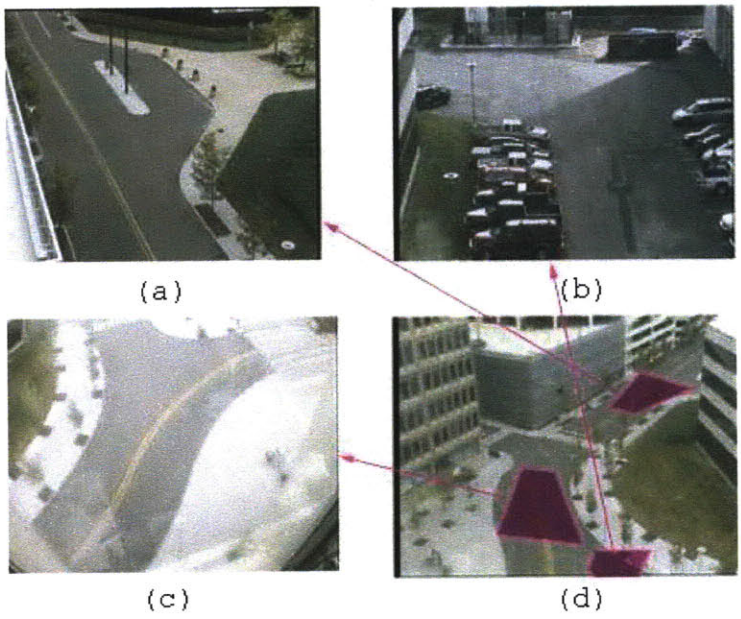


Figure 2-5: (a),(b),(c) are the three non-overlapping cameras we have used. The cameras' relative location is shown in (d) using the shaded quadrangle.

camera (c).

In our cameras, all the streets are two way streets, i.e. each source is also a sink. For simplicity, we merge sources and sinks into groups of source/sinks. The detected source/sinks in each camera are learned by clustering the spatial distribution of each observation's trajectory's beginning and ending points (i.e. the appearing coordinate and disappearing coordinate) using EM algorithm [15]. The detected source/sinks are shown in Figure 2-6. For each source/sink, there is an associated Gaussian distribution with mean and variance. From the cameras' spatial relationship, we know that there exists direct links between source/sink 3 and source/sink 4, source/sink 6 and source/sink 7, and there is no other direct link among those sources/sinks. Visible links can be easily learned using trajectory information. Our goal is to learn such "blind" links.

Because we only focus on learning the "blind" link between disjoint views, we know that the transition time must be non-negative which is determined by the nature of traffic flow, i.e, the same vehicle must first disappear at one specific location, then

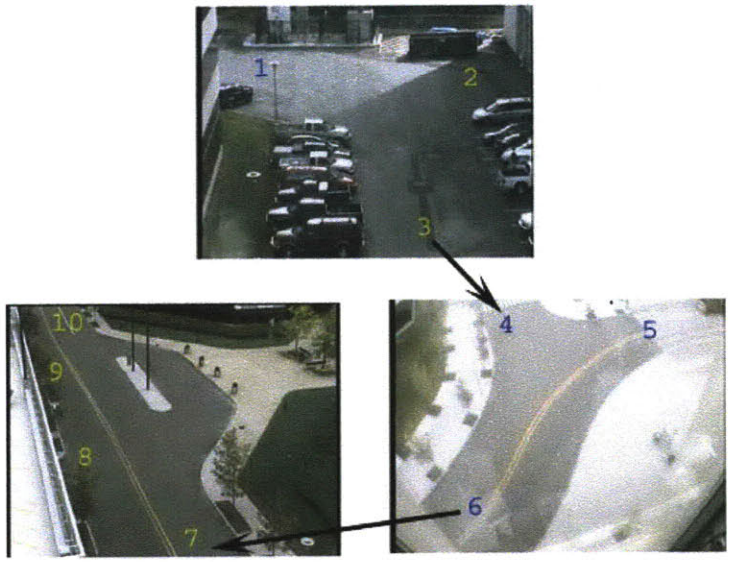


Figure 2-6: Detected sources/sinks. Black arrows indicate direct links between source/sink 3 and source/sink 4, source/sink 6 and source/sink 7

can reappear at the other different location. However, if overlapping views have been considered, the transition time may be negative.

For any pair of source/sinks, we can use the disappearing vehicles at one sink and the appearing vehicles at the other source to calculate the weighted cross correlation function. A possible link has been detected if there exists a significant peak in the cross-correlation function (See equation 2.18, in our experiments, $w$ is set to 2). Only two possible links have been detected as shown in Figure 2-7. The left one gives the cross correlation between camera $b$, source/sink 3 and camera $c$, source/sink 4, with transition time 3 seconds. The right one shows correlation between camera $c$, source/sink 6 and camera $a$, source/sink 7, with transition time 4 seconds. Figure 2-8 gives examples that do not indicate possible links. Notice that the detected "blind" links don't include the links like the one between source/sink 10 to source/sink 6 through source/sink 7. The reason is that we have used the visible trajectory's information. If we want to check the possible "blind" link between source/sink 10 and source/sink 6, we would use the observations that leave the scene through source/sink

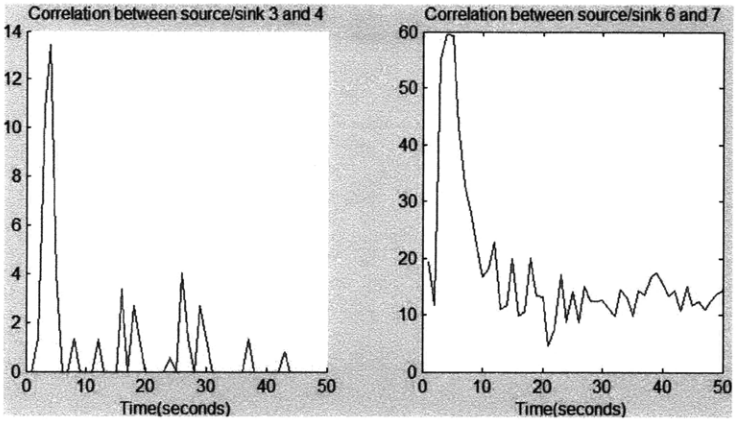


Figure 2-7: Cross correlation functions between different views. Left one gives the cross correlation between camera $b$,source/sink 3 and camera $c$, source/sink 4, with transition time 3 seconds; Right one shows correlation between camera $c$, source/sink 6 and camera $a$, source/sink 7, with transition time 4 seconds.

10 and the ones that enter the scene through source/sink 6, which wouldn't give the link through source/sink 7. So the cameras' topology can be fully recovered.

### 2.8.2 Simulated Data

We also tested our algorithm on a simulated network. This simulator synthetically generates the traffic flow in a real city street set (we couldn't disclose the city map because of security issue), allowing for stop signs, traffic lights, and differences in traffic volume (i.e. morning rush hours and afternoon rush hours have a higher volume, as well as lunch traffic). The network includes 101 cameras which are located at roads' intersections (including cross and T intersections). For each camera, there are two observers that look in the opposite directions of the traffic flow (i.e. Observer1 and 2 belong to camera 1 , Observer 3 and 4 belong to camera 2, etc). Every observer can be treated as a source/sink. Tracking data has been simulated 24 hours every day for 7 week days, including 2597 vehicles (Fig. 2-9).

Transition time may change with the road condition. For example, it will be larger during rush hour than during non-rush hour. So in our experiment, we only pick one particular hour of data (10am to 11am) each day for 5 days. For each camera, the only information we have is that vehicles appear then disappear from this location

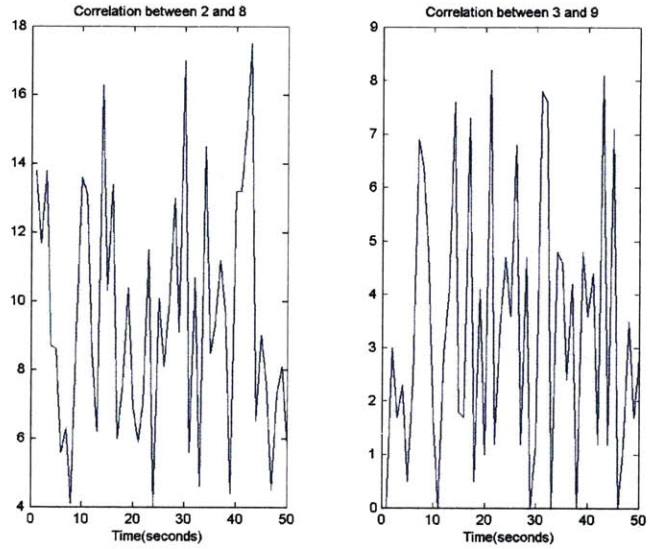


Figure 2-8: Cross correlation functions that indicate no possible links.

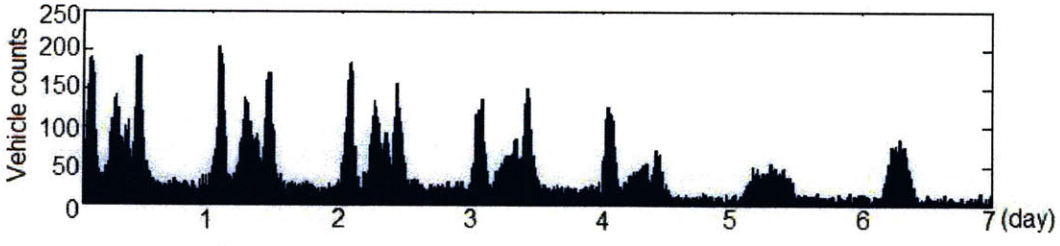


Figure 2-9: Statistics of the simulated data

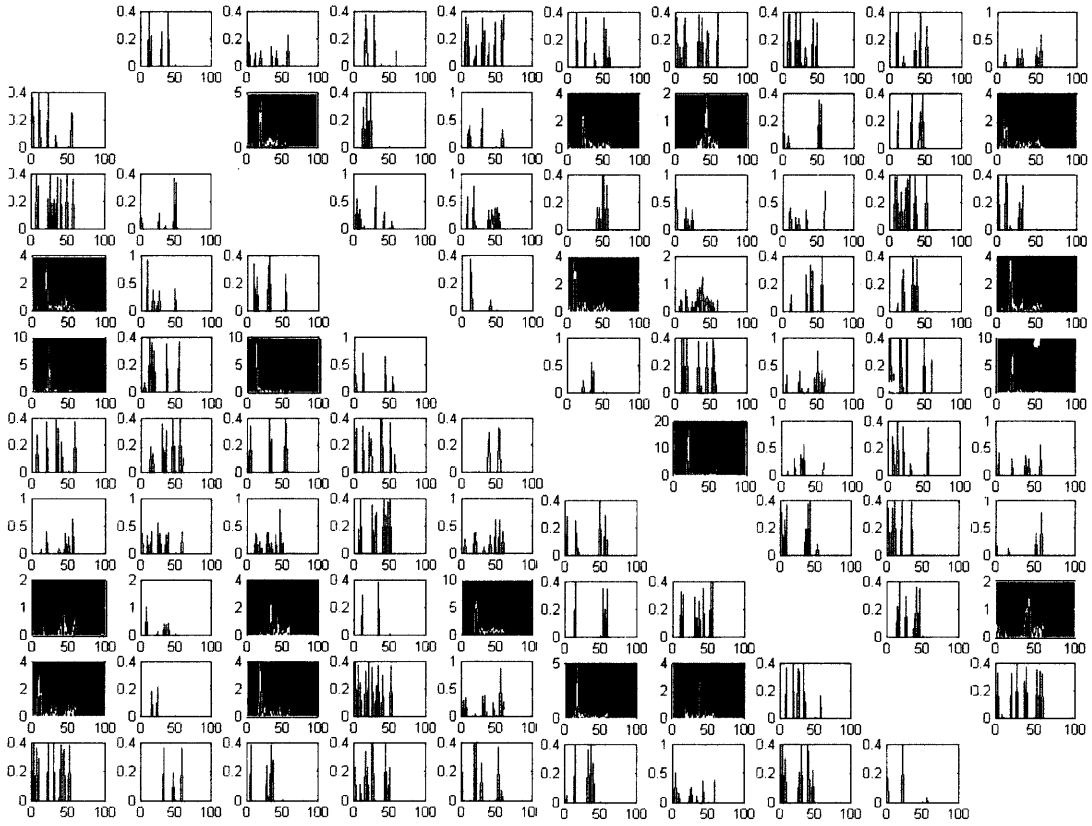

Figure 2-10: Cross correlation for each pair of the observers from 17,18,...,to 26. The column index from left to right is: observer 17, observer 18, ...., observer 26; The row index from up to bottom is: observer 17, observer 18, ...., observer 26.

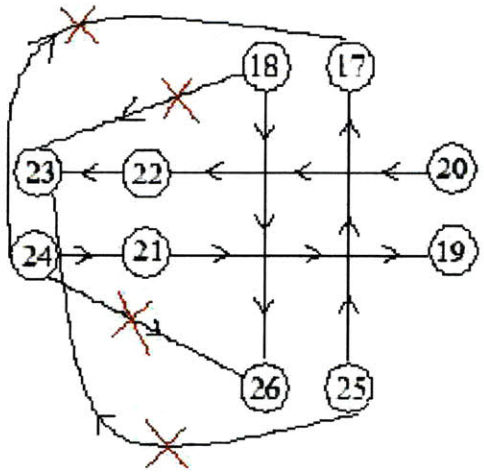


Figure 2-11: The recovered topology based on the weighted cross correlation,the red cross indicates the false link based on the group truth.

roughly at the same time (i.e. the duration is very short), so we can treat it like a delta function.

For each pair of the observers, we first calculate the cross correlation function that has been learned for each pair of the observers. A possible link has been detected if there exists a significant peak in the cross-correlation function (See equation 2.18, in our experiments, $w$ is set to 2). Figure 2-10 shows the cross correlation results for each pair of the observers from 17, 18,..., to 26, each row is the parent observer, each column is the child observer, detected possible links are highlighted in black background figures. From the detected links, however, the topology wasn't correctly recovered(see Figure 2-11). For example, there are detected links from observer 25 to 22, from observer 22 to 23 and from observer 25 to 23. We don't know if the link from observer 25 to 23 is actually through observer 22, or if there exists another link between them.

### 2.8.3 Problems

Unlike the real data, this camera view has only one source/sink and we have no information of any visible links, so we don't know where the vehicles are coming from

and where they are going. Hence all the vehicles have been used to calculate the cross correlation function. Hence, some "fake" links which are actuary connections between direct links have been detected. In order to get rid of those "fake" links and recover the true topology, we further explore the cross correlation functions.

### 2.8.4 Data Processing Inequality and Cross Correlation Coefficient

Mutual information is also a measure of the dependence between two variables [39]. If the two variables are independent, the mutual information between them is zero. If the two are strongly dependent, the mutual information between them is large. In the information theory, for a Markov chain type topology between three random variables $X \rightarrow Y \rightarrow Z$, we have $I(X;Y) \geq I(X;Z)$, where $I(X;Y)$ and $I(X;Z)$ mean mutual information between $X$ and $Y$, and $X$ and $Z$. This is called data processing inequality. Considering our camera network problem, the mutual information of neighboring cameras should be greater than non-neighboring cameras. It has been shown that data processing inequality also holds true for the cross correlation coefficient [40], which means the cross correlation coefficient of neighboring cameras should be greater than that of non-neighboring cameras. We can use this property to refine the network topology.

We already know how to estimate the weighted cross correlation $R_{i,j}(T)$. So if there exists a clear peak in $R_{i,j}(T)$ at time $T = T_{peak}$, the correlation coefficient can be estimated as:

$$\rho_{i,j}^2 = \frac{R_{i,j}(T_{peak}) - median(R_{i,j}(T))}{\sigma_{V_i}\sigma_{V_j}} \tag{2.19}$$

Because the cross correlation function is under the assumption that the signals are transient, which is not accurate for our case, we have used median of $R_{i,j}(T)$ instead of mean of $R_{i,j}(T)$.

### 2.8.5 Overall Review of The Algorithm

From the data processing inequality, we know that the cross correlation coefficient of neighboring cameras should be greater than that of non-neighboring cameras. Thus we can cluster the cross correlation coefficients of all the detected links into two categories based on the magnitude of the coefficients using k-means [?]. To implement the proposed algorithm, four steps must proceed sequentially:

1. For each possible pair of source/sinks, learn the cross correlation function;
2. Detect the possible links using the peak detection algorithm;
3. For the detected links, estimate the cross correlation coefficients, otherwise, set the cross correlation coefficient to 0;
4. Cluster the detected links into two categories: true links with higher coefficients and false links with lower coefficients.

## 2.9 More Experiment on Simulated Network

We will recover the simulated network topology based on the weighted cross correlation coefficients.

As we discussed before, for the simulated network (there is only one source/sink per camera view), only using the weighted cross correlation function to detect a peak, the topology cannot be correctly recovered.

So after the cross correlation function has been learned, cross correlation coefficients can be estimated as shown in Figure 2-12(a) with intensities corresponding to the magnitude of the coefficient information between their pair of observations. The brighter the figure, the higher the coefficient. From the data processing inequality, we know that cross correlation coefficient for the neighboring cameras is higher than that of the non-neighboring cameras. We then cluster the cross correlation coefficients of all the detected links into two categories based on the magnitude of the coefficients

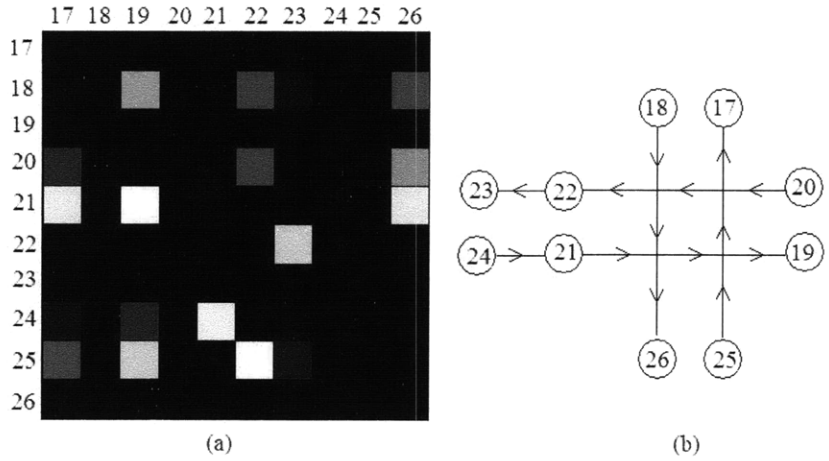


Figure 2-12: (a) The adjacency matrix of the cross correlation coefficient information. (b) The recovered corresponding topology.

using k-means. The cluster with higher coefficient would be used to recover the network topology. Figure 2-12(b) is the recovered topology for observer 17 to observer 26. We can see that the link from observer 25 to 23 is actually through 22 which is consistent with the ground truth. Table 6-1 shows the learned associated transition time for each link. Finally, the fully recovered topology of the simulated network is shown in Figure 2-13. Number means the index of the observers.

For the real data, since there are multiple source/sinks per camera view, which means we can get information of visible trajectories, we can successfully recovery the topology without calculating the cross correlation coefficient. If there is only one source/sink per camera view (i.e. zooming in), or every camera view is treated as one large source/sink, however, the cross correlation coefficient will be needed to learn the network topology.

## 2.10 Summary

In this chapter, we have studied how to recover the network's topology given vehicles' tracking data for non overlapping cameras. In order to solve this problem, we proposed a weighted cross correlation technique. First, an appearance model is

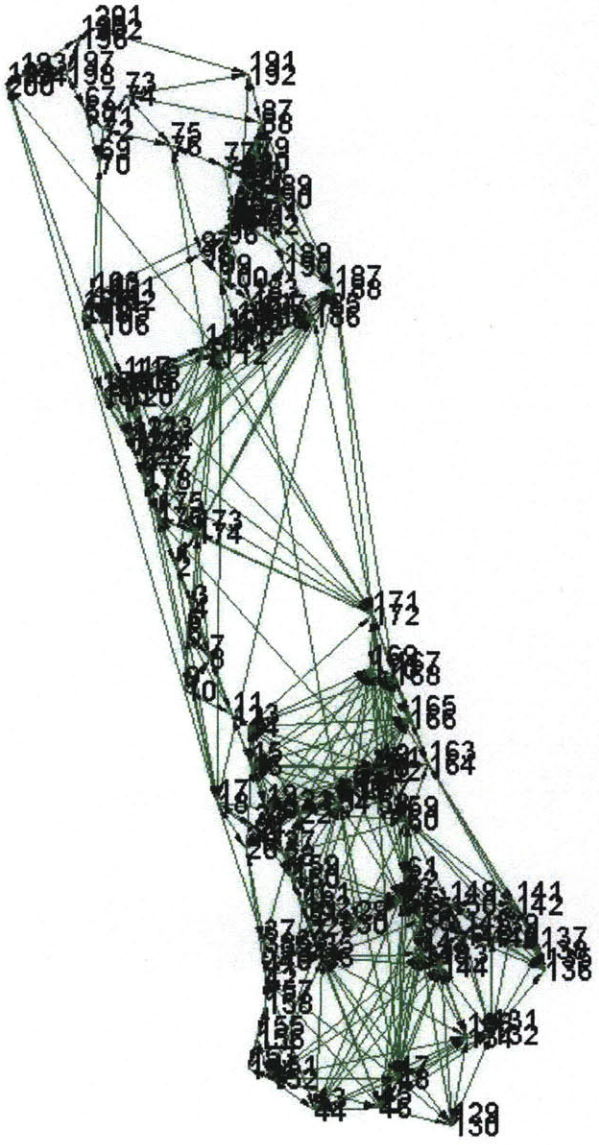

Figure 2-13: The fully recovered simulated network topology

| Parent observer | Child observer | Tran. time(Seconds) |
|---|---|---|
| 18 | 19 | 21 |
| 18 | 22 | 23 |
| 18 | 26 | 11 |
| 20 | 17 | 20 |
| 20 | 22 | 10 |
| 20 | 26 | 17 |
| 21 | 17 | 24 |
| 21 | 19 | 14 |
| 21 | 26 | 20 |
| 22 | 23 | 21 |
| 24 | 21 | 20 |
| 25 | 17 | 10 |
| 25 | 19 | 19 |
| 25 | 22 | 18 |

Table 2.1: The learned associated transition time

constructed by the combination of the normalized color and overall size model to measure the moving objects appearance similarity across the non-overlapping views. Then based on the similarity in appearance, the votes are weighted to exploit the temporally correlating information. From the learned correlation function the possible links between disjoint views can be detected and the associated transition time can be estimated. Based on the learned cross correlation coefficient, the network topology can be recovered.

This method combines the appearance information and statistics information of the observed trajectories, which can overcome the disadvantages of the approaches which only use one of them. This method also avoid doing the camera calibration, and will help to solve the tracking correspondence between disjoint views, which we will discuss in next Chapter.

# Chapter 3

# Correspondence between disjoint views

This Chapter will discuss how to use a probabilistic framework to establish observation correspondence between disjoint views (i.e. tracking objects through disjoint views).

## 3.1 Background Research

A large amount of work on multi-camera surveillance assumes overlapping views. The earlier efforts to solve the correspondence in overlapping views is to use calibrated cameras directly [43][44][47][45]. Researchers later focused on how to use tracking data to calibrate the multiple camera views [46][48][49]. Lee et al. [46] proposed to recover the camera calibration information by matching motion trajectories obtained from different views and to compute plane homographies from the most frequent matches. Their method assumed that the the topological arrangement of camera views was known[48]. Stauffer et al.[49] proposed to use an linear model to calibrate the cameras automatically. All these methods assumed that camera views had significant overlap and that objects moved on the same ground plane. This requirement is usually prohibitive in terms of cost and computational resources for surveillance of wide areas.

Huang and Russell [25] present a Bayesian foundation for computing the probability of identity, which is expressed in terms of appearance probabilities. Their appearance model is treated as the product of several independent models, such as: lane of travel, size, color and arrival time. They have used a simple Gaussian model to measure the transition probability between two disjoint views. Our work is different from this approach in that, Huang and Russel presented an application specific solution i.e. vehicles traveling in one direction, vehicles being in one of three lanes and solution formulation for only two calibrated cameras. We present a general solution, which allows movement in all directions for arbitrary numbers of un-calibrated cameras. Moreover, appearance is modeled by just the mean color value of the whole object, which is not robust enough to count for the illumination change between different views. Ali et al.[24] used a joint MAP estimation over trajectories and camera pose parameters to calibrate and track with a network of non-overlapping cameras. They modeled the dynamics of the moving object as a Markovian process. Given the location and velocity of the object from the multiple cameras, they estimated the most compatible trajectory with the object dynamics using a non-linear minimization scheme. The authors assumed that the objects move on a ground plane and that all trajectory data of the object is available, which is not suitable for online implementation. Their scheme also assumes that the correspondence of the trajectories in different cameras is already known.

Javed et al. [26][50] adopted Huang and Russell's method[25] and used Parzen windows to estimate the inter-camera space-time (i.e., transition time between two views) probabilities and then solved the correspondence problem by maximizing the posterior probability of the space-time and appearance. They proposed a subspace based color brightness transfer function (BTF) which is also called histogram equalization, then use probabilistic PCA to calculate the subspace of BTF's for a set of training data to determine the correspondence. Cheng et al. [53] proposed to learn a cumulative color histogram transformation of the disjoint views. Then they use an incremental major color spectrum histogram representation (IMCSHR). Prosser et al. [52] also use a BTF-based approach but accumulate training data before computing

the BTF. This cumulative BTF enables sparse color transformation to be preserved through the BTF calculation process. All these methods are based on learning BTFs between different views. BTFs are initially introduced in grey scale space. However, with extension into color images, all these methods apply BTF independently to the different bands of the color image, R, G, and B channel respectively, which do not consider the correlation between these different bands [55][56][57][69].

Another important branch of object recognition methods is local feature based methods [58][59][60][61][62]. Shan et al. [59] solved the vehicle correspondence problem by using multiple edge-based measures (distance, angular and magnitude difference) to compute the discriminative match scores between different vehicle images. Shan et al.[62] continue to propose a method to match vehicles through an embedding system without directly computing similarity between the two vehicle images. The key idea of their work is to use non-metric distance embeddings of vehicle observations within a camera and their corresponding embeddings in another camera as a means of characterizing similarities and differences between vehicles across cameras. Guo et al. [58] extended this idea and claimed that their method doesn't require a mapping function to align the matching scores. Guo et al. [60] utilized a compact set of 3D models to provide geometry constraints and transfer appearance features for object matching. Ferencz et al. [61] proposed a patch-based representation, where they model the distribution of comparison metrics defined on the patches, then used an online algorithm to select the most salient patches based on a mutual information criterion to label the given image pairs as matching or not. Arth et al. [63] presented a system to reacquire and track vehicles. They use PCA-SIFT to extract features from a large set of samples, build the feature vocabulary tree, and match the samples based on the tree. All these methods need to compute images' local features (i.e. image edges, corners or SIFT feature), and wouldn't be robust and are not feasible in our far field vehicle correspondence scenario because of the low resolution.

We propose a Maximum A Posteriori (MAP) estimation framework to combine multiple cues (i.e. space-time, color, and shape/size) to model the transition between different views for moving vehicles. Especially, we propose to model the color transfor-

mation in $l\alpha\beta$ apace. After we know how to model the correspondence probability for observations captured by different source/sinks, we adopt a probabilistic framework to use this correspondence probability in a principled manner. The information about the network topology will help us here. We only model the correspondence probability of observations that are from the connected source/sinks, which will dramatically reduce the search space and resolve ambiguities arising from similar observations presented by different objects. Tracks are assigned by estimating the correspondences which maximize the posterior probabilities (MAP). This is achieved by using the Hungarian algorithm to solve the association matrix. After establishing the correspondence, we have a set of stitched trajectories, in which elements from each camera can be combined with observations in multiple subsequent cameras generated by the same object.

## 3.2 Probabilistic Formulation of Correspondence Problem

Suppose we have a system composed of a set $C = \{c_1, c_2, \ldots, c_m\}$ of $m$ cameras, which cameras are not overlapped with each other. Unlike the approaches in [26][25][51] which consider whole camera views, we only consider the connected source/sink views which we already computed with the methods from Chapter 2.

Our goal is to construct a correspondence between these views. Specifically, for each of the camera views we define its set of $n$ source/sink regions as $E^1_{C_i}, E^2_{C_i}, \ldots, E^n_{C_i}$. We then simplify this by describing the global set of $g$ source/sink regions as $E_1, E_2, \ldots, E_g$ as shown in Figure 3-1. Assume that the task of single camera tracking is already solved, and let $O_j = \{O_{j,1}, O_{j,2}, \ldots, O_{j,k}\}$ be the set of $k$ object observations that were observed in source/sink region $E_j$ . For each observation, there is a location, time of observation and set of appearance features such as color, shape and size associated with it. It is reasonable to assume that appearance and time of observation features are independent of each other, i.e., the appearance of an object does not depend on

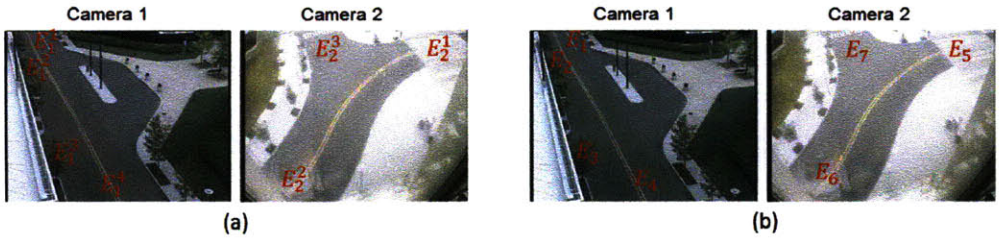


Figure 3-1: Source/Sink illustration. (a) shows source/sink regions in Camera 1 $(E_1^1, \ldots, E_1^4)$ and Camera 2 $(E_2^1, \ldots, E_2^3)$, respectively. (b) shows the global set of source/sink regions in Camera 1 and Camera 2 from $E_1$ to $E_7$.

the time when is observed and vice versa. The problem of multi-camera correspondence is to find which of the observations in the system of cameras belong to the same object.

For a formal definition of the above problem, we let a correspondence $h_{i,a}^{j,b}$ be an ordered pair $(O_{i,a}, O_{j,b})$, which defines the hypothesis that the observations $O_{i,a}$ and $O_{j,b}$ are consecutive observations of the same object in the environment, with the observation $O_{i,a}$ preceding the observation $O_{j,b}$. The problem of multi camera correspondence is to find a set of correspondences $H = \{h_{i,a}^{j,b}\}$ such that $h_{i,a}^{j,b} \in H$ if and only if $O_{i,a}$ and $O_{j,b}$ correspond to consecutive observations of the same object in the environment.

Let $\Sigma$ be the solution space of the multi camera correspondence problem as described above. Assuming that an object in $E_i$ is seen no more than once in $E_j$, we aim to find the subset of $\Sigma$, $H$, where each $h_{i,a}^{j,b} \in H$, if and only if $O_{i,a}$ and $O_{j,b}$ correspond to consecutive observations of the same object. We define the solution of the multi camera correspondence problem to be a hypothesis $H'$ in the solution space $\Sigma$ that maximizes the posteriori probability, and is given by:

$$H' = \arg\max_{H \in \Sigma} P(H|O) \tag{3.1}$$

Assuming that each correspondence, i.e., a matching between two observations, is conditionally independent of other observations and correspondences, we have:

$$P(H|O) = P(H|O_1, O_2, \ldots, O_k) = \prod_{h_{i,a}^{j,b} \in H} P(h_{i,a}^{j,b}|O_{i,a}, O_{j,b}) \tag{3.2}$$

where $P(h_{i,a}^{j,b}|O_{i,a}, O_{j,b})$ is the conditional probability of the correspondence $h_{i,a}^{j,b}$ given the observation $O_{i,a}$ and $O_{j,b}$ for two source/sinks $E_i$ and $E_j$ in the system. From Bayes Rule, we get,

$$P(h_{i,a}^{j,b}|O_{i,a}, O_{j,b}) = \frac{P(O_{i,a}, O_{j,b}|h_{i,a}^{j,b})P(h_{i,a}^{j,b})}{P(O_{i,a}, O_{j,b})} \tag{3.3}$$

where prior $P(h_{i,a}^{j,b})$ is the transition probability $P(E_i, Ej)$ from source/sink $E_i$ to $E_j$, which we already got from Chapter 2. With this equation, we have,

$$P(H|O) = \prod_{h_{i,a}^{j,b} \in H} (\frac{1}{P(O_{i,a}, O_{j,b})} P(O_{i,a}, O_{j,b}|h_{i,a}^{j,b})P(E_i, Ej)) \tag{3.4}$$

Moreover, we assume that the observation pairs are uniformly distributed and hence, $P(O_{i,a}, O_{j,b})$ is a constant scale factor. Thus, the problem is reduced to the solution of following term:

$$H' = \arg\max_{H \in \Sigma} \prod_{h_{i,a}^{j,b} \in H} (P(O_{i,a}, O_{j,b}|h_{i,a}^{j,b})P(E_i, Ej)) \tag{3.5}$$

As we mentioned before, for each observation, there are a location, time of observation $O(time)$ and appearance features such as color $O(color)$, shape and/or size $O(s)$ associated with it, and these features are independent. We can factorize these terms into the above form, and we have,

$$\begin{aligned} H' = \arg\max_{H \in \Sigma} \prod_{h_{i,a}^{j,b} \in H} & (P(O_{i,a}(color), O_{j,b}(color)|h_{i,a}^{j,b}) \\ & P(O_{i,a}(s), O_{j,b}(s)|h_{i,a}^{j,b})P(O_{i,a}(time), O_{j,b}(time)|h_{i,a}^{j,b})P(E_i, Ej)) \end{aligned} \tag{3.6}$$

This is equivalent to minimizing the following term (where the product is replaced

by summation by taking the log of the above term):

$$H' = \arg\min_{H\in\Sigma} \sum_{h_{i,a}^{j,b}\in H} log(P(O_{i,a}(color), O_{j,b}(color)|h_{i,a}^{j,b}) P(O_{i,a}(s), O_{j,b}(s)|h_{i,a}^{j,b}) P(O_{i,a}(time), O_{j,b}(time)|h_{i,a}^{j,b}) P(E_i, Ej)) \quad (3.7)$$

By taking the negated logarithm, the maximization of the product turns into the minimization of a sum. This makes it possible to express the maximization of the posterior as a linear program. More specifically, it becomes a weighted assignment problem for which very efficient algorithms exist, for example, the Hungarian algorithm that we currently use to compute a solution [54]. The input to the Hungarian algorithm is a matrix called the *association matrix*, $M$, for source/sink $E_i$ to $E_j$, where each entry is given by

$$M_{a,b} = -log(P(O_{i,a}(color), O_{j,b}(color)|h_{i,a}^{j,b}) P(O_{i,a}(s), O_{j,b}(s)|h_{i,a}^{j,b}) P(O_{i,a}(time), O_{j,b}(time)|h_{i,a}^{j,b}) P(E_i, Ej)) \quad (3.8)$$

In order to construct the association matrix $M$, we need to find the appearance (i.e. color and shape/size) and the space-time probability density functions. This issue is discussed in the next sections.

## 3.3 Color correspondence

A commonly used cue for tracking moving objects in different views is the color of the objects. However, unconsistent color between different views is a serious problem in multi-camera systems. Such a system may contain identical cameras that are operating under various lighting conditions, e.g. indoor cameras under fluorescent lamps or outdoor cameras in daylight, etc. and different cameras that have dissimilar radiometric responses. Even between identical cameras all working outdoors, it is possible to have color deviations due to different scene illumination and optical

materials. Images of the same objects acquired under these variants usually show dissimilar color characteristics, and this makes the correspondence problem or other related computer vision tasks more challenging.

In Chapter 2, we have discussed how to compensate for color differences using a comprehensive color normalization algorithm. However, in order to construct a fine appearance model to solve the correspondence problem, we find this method no longer suffices[64]. Instead, we seek to find a global histogram transformation that maps the color of an object in one camera image to its color in another camera image.

### 3.3.1 Histogram Transformation for Grey Level Images

The most common approach to match the histograms of two images is histogram specification[66]. Histogram specification is a technique that transforms the histogram of one image into the histogram of another image through histogram equalization [66][1].

Let variable $r$ and $z$ represent the gray levels of the input and output images, respectively. And let $p_r(r)$ and $p_z(z)$ denote their corresponding continuous probability density functions. We can estimate $p_r(r)$ from the given input image, while $p_z(z)$ is the specified probability density function that we wish the output image to have.

Let $s$ be a random variable with the property

$$s = T(r) = \int_0^r p_r(w)dw \tag{3.9}$$

where $w$ is a dummy variable of integration. The right side of this equation is recognized as the cumulative distribution function(CDF) of random variable $r$. This equation is known as the continuous version of histogram equalization. Suppose next we define a random variable $z$ with the property

$$G(z) = \int_0^z p_z(t)dt = s \tag{3.10}$$

[1]Much of the material in this section is based on the presentation in [66], and provides background for understanding our variation of this approach to the specific problem addressed in this thesis

where $t$ is a dummy variable of integration. It then follows from these two equations that $G(z) = T(r)$ and, therefore, that $z$ must satisfy the condition

$$z = G^{-1}(s) = G^{-1}[T(r)] \tag{3.11}$$

The transformation $T(r)$ can be obtained once $p_r(r)$ has been estimated from the input images. Similarly, the transformation function $G(z)$ can be obtained because $p_z(z)$ is given.

The above equations (Eq.3.9 - Eq.3.11) show that an image with a specified probability density function can be obtained from an input image by using the following procedure:

1. Obtain the transformation function $T(r)$ using Eq.3.9.
2. Obtain the transformation function $G(z)$ using Eq.3.10
3. Obtain the inverse transformation function $G^{-1}$.
4. Obtain the output image by apply Eq.3.11 to all the pixels in the input image.

The result of this procedure will be an image whose gray levels, $z$, have the specified probability density function $p_z(z)$.

Although the procedure just described is straightforward in principle, it is seldom possible in practice to obtain analytical expressions for $T(r)$ and for $G^{-1}$. Fortunately, this problem is simplified considerably in the discrete case. The discrete formulation of Eq.3.9 is

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) \quad k = 0, 1, \ldots, L-1 \tag{3.12}$$

where $p_r(r)$ is the input image's grey level histogram and $L$ is the number of discrete gray levels. Similarly, the discrete formulation of Eq.3.10 is obtained from the target histogram $p_z(z)$, and has the form

$$r_k = G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k \quad k = 0, 1, \ldots, L-1 \tag{3.13}$$

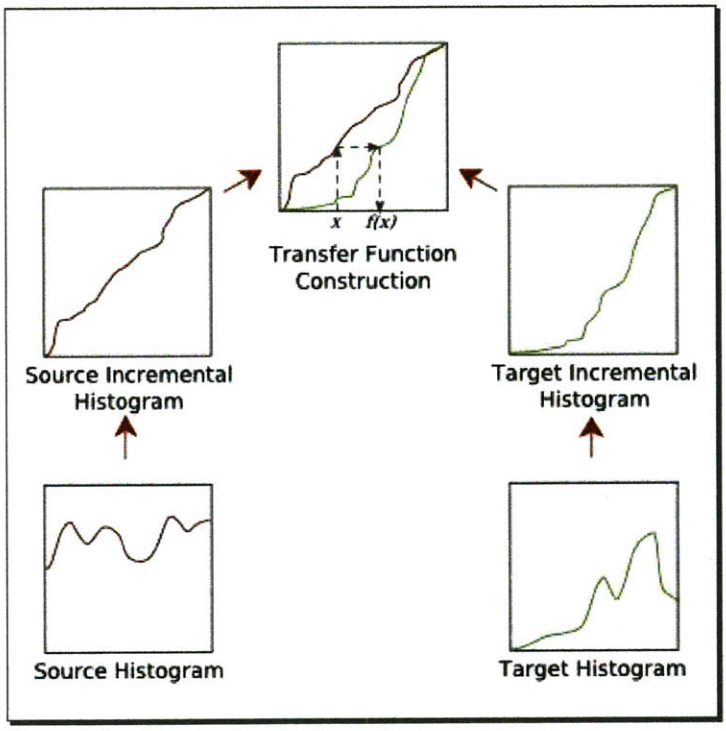


Figure 3-2: Histogram matching illustration

Finally, the discrete version of Eq.3.11 is given by

$$z_k = G^{-1}(s_k) = G^{-1}[T(r_k)] \quad k = 0, 1, \ldots, L-1 \tag{3.14}$$

Given Eq.3.12, Eq.3.13 and Eq.3.14, we know histogram matching or histogram specification in grey scale images can be implemented in the following steps (as illustrated in Fig.3-2):

1. Calculate the histogram of the given image.

2. Calculate the cumulative sum of the histogram of the given image $s_k$ using Eq.3.12.

3. Calculate the cumulative sum of the target histogram $r_k$ using Eq.3.13

4. Map pixels from one bin to another according to the rules of histogram equal-

ization. More specifically, for each pixel in the original image, if the value of that pixel is $r_k$, map this value to its corresponding level $s_k$; then map level $s_k$ into the final level $z_k$ based on the precomputed values from step 2 and 3 (more detail can be found in [66].

Although histogram matching in grey scale images has been proven to be the most simple and effective method, the extension from gray-scale images to color images is not trivial and straightforward. The common method of histogram matching of a color image is to treat each of the red, green, and blue channels as a grey level image matching each of these independently. However, since this method doesn't consider the statistical dependence between these color axes, it will produce unwanted color artifacts. Removing the correlation between the color axes prior to independently modifying the histogram of each color axis has proven useful in texture synthesis[67] and remote sensing[68]. Next, we will discuss how to do the histogram matching by selecting a color space which exhibits minimal statistical dependence.

### 3.3.2 RGB to $l\alpha\beta$ Space

Color image decorrelation is useful for applying color image processing operations independently on each image channel[69]. Recently, Ruderman et al.[71] developed a color space, called $l\alpha\beta$, which minimizes correlation between channels for many natural scenes. This space is based on data-driven human perception research that assumes the human visual system is ideally suited for processing natural scenes [2].

$l\alpha\beta$ space is a transform of LMS cone space, which is represented by the response of the three types of cones of the human eye, named after their responsivity (sensitivity) at long, medium and short wavelengths. It is common to use the LMS color space when performing chromatic adaptation (estimating the appearance of a sample under a different illuminant). In [71] Ruderman shows that in LMS space, there are three expected facts. First, signals between all pairs of cone types are strongly correlated. This occurs because overall fluctuations in light intensity will tend to increase all cone

[2]Much of the materia in this section is based on [69][71]

responses simultaneously. Second, the correlation between the **L** and **M** photoreceptors is much larger than that between the **L** and **S** photoreceptors. This is primarily due to the large overlap of the **L** and **M** cone spectral sensitivities. Finally, receptor response distributions show a great deal of skew and are highly asymmetrical. Ruderman proposed to eliminate the skewness by converting the data to logarithmic space. Then they proceed to decorrelate these axes in the new logarithmic space using principal components analysis (PCA). Next, we will show how to decorrelate the color space step by step[69].

Since $l\alpha\beta$ space is based on LMS, we first convert the image to LMS space in two steps. The first is a conversion from RGB to XYZ tristimulus values (this conversion is derived from the International Telecommunications Union standard matrix, and more details can be found in [69]):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.15}$$

then we can convert the image to LMS space using the following conversion called Hunt-Pointer-Estevez (HPE) transformation [72]:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.16}$$

Combining these two matrices gives the following transformation between RGB and LMS cone space:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.17}$$

The data in this color space shows a great deal of skew, which we can largely eliminate by converting the data to logarithmic space:

$$\begin{aligned} \mathbf{L} &= log(L) \\ \mathbf{M} &= log(M) \\ \mathbf{S} &= log(s) \end{aligned} \tag{3.18}$$

The next step is to decorrelate the three axes by principal components analysis [70], which effectively rotates them and makes them orthogonal. The analysis is straightforward, and the three resulting orthogonal principal axes have simple forms and are close to having integer coefficients. Moving to those nearby integer coefficients, the following transform has been adopted [71]:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix} \tag{3.19}$$

If we think of the **L** channel as red, the **M** channel as green, and the **L** channel as blue, we can see that this is a variant of many opponent-color models[72]:

$$\begin{aligned} l &\propto Achromatic &&\propto r+g+b \\ \alpha &\propto Yellow-Blue &&\propto r+g-b \\ \beta &\propto Red-Green &&\propto r-g \end{aligned} \tag{3.20}$$

Thus the $l$ axis represents an achromatic channel, while the $\alpha$ and $\beta$ channels are chromatic yellow-blue and red-green opponent channels. In this color space, however, data are symmetrical and compact, and the channels are independent, which will simplify the histogram transformation method. The conversion from $l\alpha\beta$ to RGB space is very straightforward. First, we convert from $l\alpha\beta$ to LMS using this matrix multiplication:

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} \tag{3.21}$$

Then, after raising the pixel values to the power ten to go back to linear space,

we can convert the data from LMS to RGB using:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \tag{3.22}$$

### 3.3.3 Color Transform in $l\alpha\beta$ space

Reinhard [69] showed that the mean and standard deviation along each of the three axes in $l\alpha\beta$ space suffice if we intend to transfer the given image's color into the target image's color. First, we subtract the mean from the data points:

$$\begin{aligned} l_s^* &= l_s - \overline{l_s} \\ \alpha_s^* &= \alpha_s - \overline{\alpha_s} \\ \beta_s^* &= \beta_s - \overline{\beta_s} \end{aligned} \tag{3.23}$$

where the subscript s represents the source image. $\overline{l_s}$, $\overline{\alpha_s}$ and $\overline{\beta_s}$ are the means of the data in respective channels.

Then, we scale the data points with the ratio of the respective standard deviations of the source and target images, and add the means of the target image respectively:

$$\begin{aligned} l' &= \frac{\sigma_t^l}{\sigma_s l} l_s^* + \overline{l_t} \\ \alpha' &= \frac{\sigma_t^\alpha}{\sigma_s \alpha} \alpha_s^* + \overline{\alpha_t} \\ \beta' &= \frac{\sigma_t^\beta}{\sigma_s \beta} \beta_s^* + \overline{\beta_t} \end{aligned} \tag{3.24}$$

where $\sigma_t^l$, $\sigma_t^\alpha$, $\sigma_t^\beta$ are the standard deviations of the target image in respective channels, and $\sigma_s^l$, $\sigma_s^\alpha$, $\sigma_s^\beta$ are the standard deviations of the source image in respective channels.

After this transformation, the resulting data has the same means and standard deviations to the target image in $l\alpha\beta$ space. Finally, we transform the result data back into RGB space using Eq.3.21 and Eq.3.22, so that the resulting image has a similar color appearance with the target image.

Figure 3-3 gives examples of color transformation both in RGB and $l\alpha\beta$ spaces. Transformation in $l\alpha\beta$ space yields better results than that in RGB space. Color of

the transformed images in $l\alpha\beta$ spaces is more smooth and real.

### 3.3.4 Color model

Now we know how to model the color transformation for a given pairs of images in $l\alpha\beta$ space. Our goal is to learn a global color transformation for any pair of views. We study the distributions of slopes and intercepts for pairs of images taken at two different views. Figure 3-4 gives an illustration of slope's and intercept's in Eq.3.24 distributions for a pair of given views. From this distribution, we can fit a multivariate Gaussian on the slope and intercept for $l$, $\alpha$ and $\beta$ channels to model the color $P(O_{i,a}(color), O_{j,b}(color)|h_{i,a}^{j,b})$ throughout any two learned connected source/sinks of different views.

## 3.4 Shape/Size correspondence

Local features (i.e. edges, corners or SIFT features) have shown to be very helpful in solving the multi-camera correspondence problem [58][59][60][61][62]. As we discussed in Chapter 2, however, for far-field surveillance, even after successful detection, there are often very few image pixels per object, which makes it difficult to model the local feature change throughout cameras (see detected moving vehicle examples in Fig. 3-5). However, as we argued in Chapter 2, we know for sure that a sedan in one scene cannot be a truck in another scene, which means overall size information still plays an important role in correspondence.

Give the objects are fairly small in far field settings, it is unlikely that we will be able to recover the shape detail, so all we rely on is overall size measures. Given this setting, where the sizes of the observed objects are much smaller than the distance to the camera, the bounding ellipse area is a good representation of the object size, as most objects appear merely as rigid blobs and the ellipse will account for orientation relative to the camera. However, in a near-field setting, where the articulation and detailed shape of objects can be observed, more descriptive attributes such as the shape or area of the silhouette might be useful. Hence, we fit a best ellipse to the

Figure 3-3: Color transformation in RGB and $l\alpha\beta$ spaces. The first column is the original image, the second column is the target image, the third column is the color transformation in RGB space and the fourth column is the transformation in $l\alpha\beta$ space. We can see that the results in $l\alpha\beta$ are more smooth and real space.

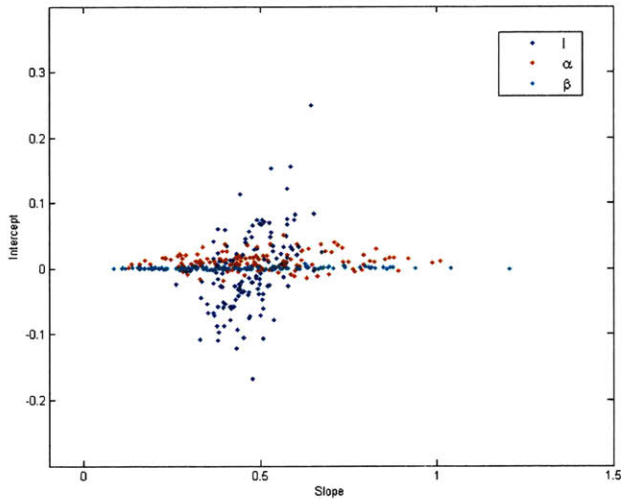


Figure 3-4: slope and interception distributions for a pair of given views in $l\alpha\beta$ space

Figure 3-5: Tracked vehicle samples from different views.

shape using the minimum volume enclosing ellipsoid method [73][75][74], to model the shape/size change between different views.

We also adopt a multivariate Gaussian distribution to model the size change $P(O_{i,a}(s), O_{j,b}(s)|h_{i,a}^{j,b})$.

$$\begin{aligned} P(O_{i,a}(s), O_{j,b}(s)|h_{i,a}^{j,b}) &= P(w_{i,a}, l_{i,a}, w_{j,b}, l_{j,b}|h_{i,a}^{j,b}) \\ &= N_{\mu_{w_i/w_j,l_i/lj},\Sigma_{w_i/w_j,l_i/lj}}(w_i/w_j, l_i/l_j) \end{aligned} \tag{3.25}$$

where $w_{i,a}$,$l_{i,a}^{l}$ are the detected vehicle's ellipse's semimajor axis and semiminor axis under entry/exit i. $\mu_{w_i/w_j,l_i/lj}$ and $\Sigma_{w_i/w_j,l_i/lj}$ are the mean and variance of the semimajor axis ratio and semiminor axis ratio for source/sink $E_i$ and source/sink $E_j$, respectively. The imaging transformation of a perspective camera leads to distortion of a number of geometric scene properties. As a result, objects appear to grow larger as they approach the camera center and become smaller when they are far away from the camera [34]. So in the sense of simple normalization, the average size over the whole trajectory has been adopted, when we do the size model. The parameters of this Gaussian distribution can be estimated using the same procedure as described in Chapter 2.5.2.

## 3.5 Inter Camera Time Correspondence

The observations of an object exiting from one camera and entering into another camera will be separated by a certain time interval. We refer to this interval as inter-camera travel time. As we discussed in Chapter 2, we already know that there exits an average transition time from one source/sink in one camera view to another source/sink in another camera view. This average transition time can be used to constrain correspondences.

Figure 3-6 gives an example of a transition time distribution for a pair of source/sinks. This data shows a great deal of skewness because of the nature of the traffic. For a

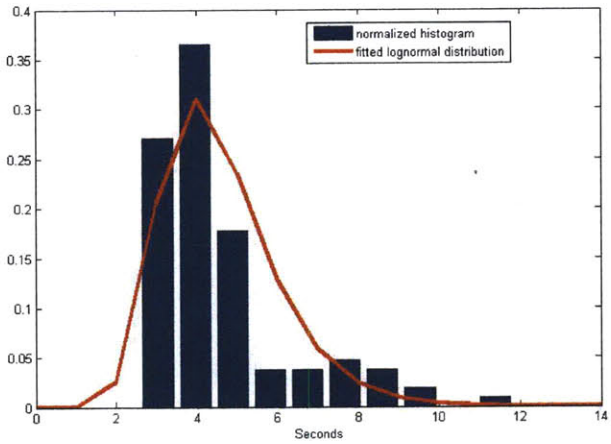


Figure 3-6: A log normal fitting of transition times for a pair of connected source/sinks.

given pair of source/sinks, although there is an average transition time between them, it is more common for a car moving from one sounce/sink to the other source/sink using more time than using less time. For example, as shown by Fig. 3-6, with the average transition time around 4 seconds, it is not possible for a vehicle to use 0 seconds to make the transition between the source/sinks unless there is a timing error. However, it is quite normal for a vehicle to use 8 seconds (e.g. waiting for a pedestrian to pass by). Thus, the distribution is asymmetrical. In order to compensate for this skewness, we propose to use a log normal distribution to model the inter camera transition.

## 3.6 Experiment Results and Discussion

In this section, we demonstrate how our proposed algorithm works in two different multi-camera scenes: a street scene and a parking lot scene respectively. The street scene has 2 camera views as shown in Fig. 3-7 and the parking lot scene has 3 camera views as shown in Fig. 3-8, which are all non overlapping. The topologies have already been learned in Chapter 2. In these settings, scene illumination conditions are quite different, the size of vehicles is very small (especially in our parking lot


Figure 3-7: Street Scene

scene), and a vehicle's pose changes dramatically, which all make the data sets very challenging. For each setting, every experiment consists of a training phase and a testing phase. In the training phase, the correspondences are assumed to be known and this information is used to compute the color, shape/size and time transformation for every pair of connected source/sinks. In the testing phase, these correspondences are computed using the proposed algorithm. The performance of the algorithm is analyzed by comparing the resulting tracks to the ground truth. We also compare our algorithm to other color transformation/calibration approaches: basic RGB transformation, the comprehensive color normalization algorithm proposed in Chapter 2 and the brightness transformation function (BTF) subspace [50] algorithm.

### 3.6.1 Data Sets of the Experiment

The first experiment was conducted with Street Scene, with two cameras 1 and 2. The camera topology is shown in Figure 3-7. It can be seen from the figure that there is a significant illumination difference between the two camera views, and matching the color is considerably difficult without accurate modeling of the changes in color across the cameras. We hand labeled 30 minutes of tracking data, with 5 minutes of data as the training set, and 25 minutes of data as the testing set. Testing data contains a total of 749 individual tracks from each camera, of which there are 293 vehicles appearing at both views, and 163 vehicles only appearing at one view.

The second experiment was conducted with Parking Lot Scene, with three cameras

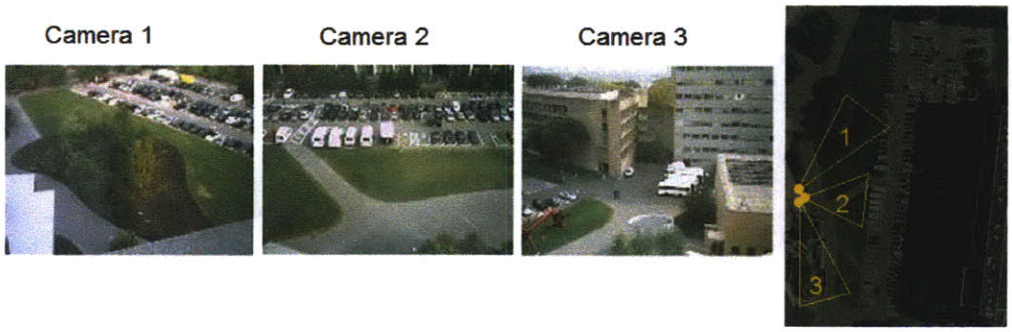


Figure 3-8: Parking Lot Scene

1 and 2 and 3. The camera topology is shown in Figure 3-8. From these three different views, we can see that the illumination difference isn't as significant as that of Street Scene, however, the pose and size of the vehicles change dramatically between different views. We also hand labeled 30 minutes of tracking data, with 5 minutes of data as the training set, and 25 minutes of data as the testing set. Testing data contains a total of 182 individual tracks from each camera, of which there are 73 vehicles appearing at the views of 1 and 2, 5 vehicles appearing at the views of 1 and 2 and 3, and 5 vehicles appearing at the views of 2 and 3.

### 3.6.2 Experiment results

First, we study how each of our correspondence feature models (i.e. color, inter-camera time and shape/size models) contributes to solve the tracking problem, and how the combination of these models improves the tracking result. For each scene, we examine four different cases separately, 1) only color model (i.e. $l\alpha\beta$ transformation), 2) only inter-camera time model, 3) $l\alpha\beta$ transformation and inter-camera time models, and 4) all three models. The results of each of these cases are analyzed by using correspondence accuracy (i.e. the ratio of the number of objects tracked through different views correctly to the total number of objects that passed through the scene) as the evaluation measure. These results are summarized in Figure 3-9 and are explained below.

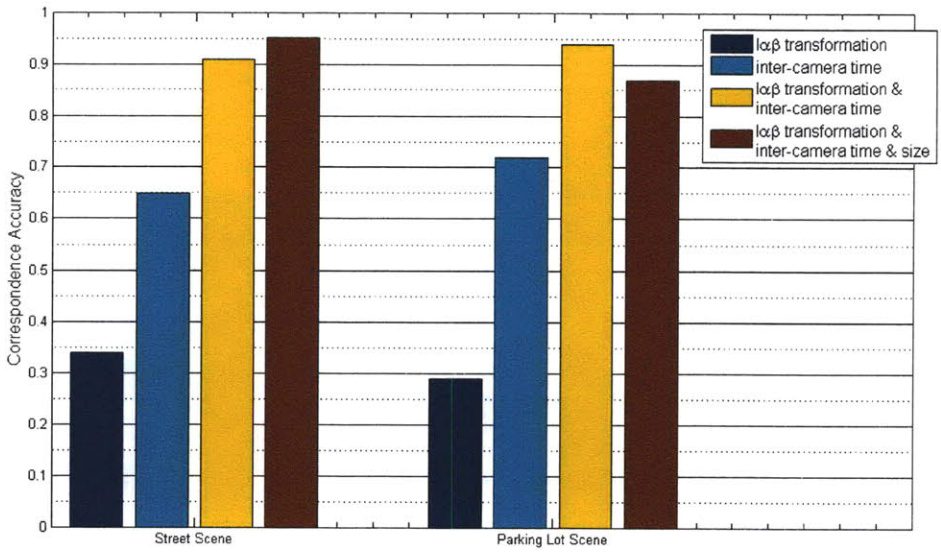


Figure 3-9: Performance comparison of using different feature models in street and parking lot scenes.

It is clear from Figure 3-9 that both the appearance and space-time models are important sources of information as the tracking results improve significantly when both the models are used jointly in both scenes. Because the parking lot scene isn't as busy as the street scene, the space-time model alone performs better in the parking lot scene than in the street scene. With additional shape/size information, the tracking accuracy is improved by 3.7% in the street scene. In the parking lot scene, however, the added shape/size model impairs the tracking accuracy. The reason is that, in the parking lot scene, there are a lot of pedestrians moving around, which causes the tracker to group together the moving vehicle with moving pedestrians. Hence, the detected moving vehicle's bounding box changes through every view dramatically. Figure 3-10 gives an example. When a vehicle entered the view of camera 2, it was tracked perfectly. Then it passed the crosswalk, where several people passed by, and they were grouped together. After that, the vehicle separated from them and was tracked back to normal. Then it captured another person, which was tracked

Figure 3-10: An example of a tracked moving vehicle's bounding box changing throughout the view of Camera 2 in the parking lot scene. When there was people passing by, the vehicle was grouped with the people by the tracker.

together and separated. This kind of activity happens frequently in the parking lot scene, which makes the shape/size of every vehicle unstable throughout the view and this information doesn't help to construct the correspondence between different views. This problem will be solved, however, if we can improve the tracking accuracy for each individual view. For instance, we can separate the moving vehicle from the moving people by carefully choosing the connected component parameters [49] in the tracking model. Since it is not the focus of our research, we won't discuss here. Although the shape/size information didn't help in the parking lot scene, it did help in the street scene. Therefore, in the real multi camera tracking system, we need to carefully decide which feature to use: color, inter-camera time, shape/size, or any combination of them. Ideally, all these three should be beneficial for the tracking through different views.

In order to demonstrate how our $l\alpha\beta$ color transformation helps to improve the tracking results compared with other color histogram-based techniques, for each scene, we examine four different approaches, 1) naive RGB histogram matching with our inter-camera time model, 2) color model used in Chapter 2–Comprehensive Color Normalization (CCN) with our inter-camera time model, 3) brightness transfer function (BTF) subspace with our inter-camera time model, and 4)$l\alpha\beta$ color transformation with inter-camera time model. The results of each of these cases are analyzed by using correspondence accuracy defined previously as the evaluation measure. These results are summarized in Figure 3-11 and are explained below.

It can been that in the street scene, since the illumination condition differs significantly between the two camera views, any color correlation method would dramatically improve the tracking accuracy. Among our choices, the BTF subspace and our

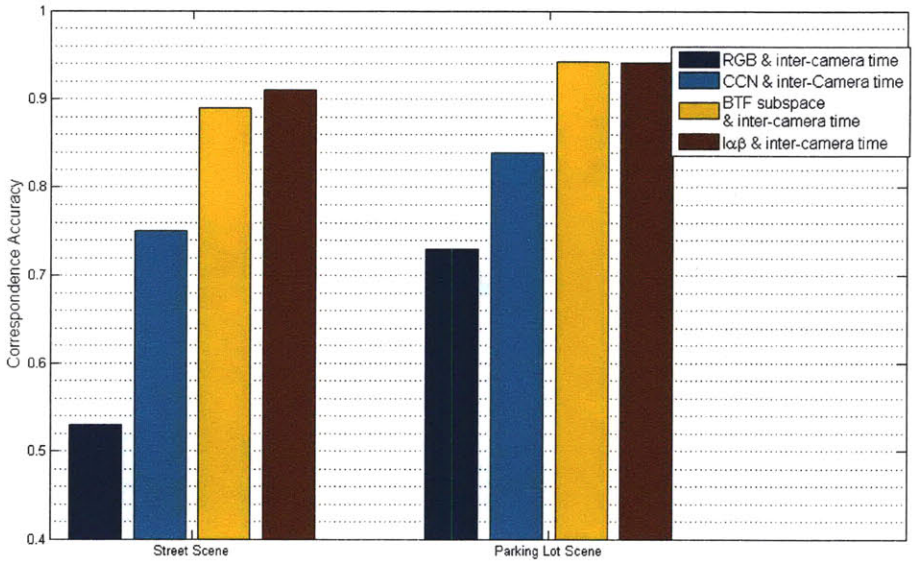


Figure 3-11: Tracking comparison with other color transformation models

$l\alpha\beta$ space models perform the best. This also holds true for the parking lot scene although the improvement from the RGB histogram matching model isn't as big as that of the street scene. The performance of the BTF subspace model and our $l\alpha\beta$ space in both scenes are quite similar. However, the complexity of our color model is less than that of the BTF subspace model. As we recall, in our $l\alpha\beta$ model, we only need to model the slopes and intercepts for the $l$, $\alpha$ and $\beta$ channel respectively, hence, the dimensionality is only 6. However, in the BTF subspace model, in order to achieve the same performance, we choose 6 of the principle components of R, G and B channel, respectively. These 6 principle components account for more than 95% of the variance in the BTF's. This gives us a total of 18 dimensions, which is significantly larger than the dimensionality of our $l\alpha\beta$ model. With the lower dimensionality, our model shows the same performance as the BTF subspace model, which suggests our model is better than the BTF subspace model with fewer parameters, and less computational cost.

## 3.7 Summary

In this chapter, we have proposed a MAP framework to match two objects along their tracks from non overlapping camera views. We showed how to learn the color transformation in $l\alpha\beta$ space, inter-camera time transition and shape/size transformation between different views. The main contribution is the proposal to learn the color transformation in $l\alpha\beta$ space to compensate for the varying illumination conditions across different views. The experiment results show the combination of the color and inter-camera time models improves the tracking accuracy significantly. And shape/size information can help or jeopardize the tracking results depends on the accuracy of the individual camera tracker. Hence, in the real multi camera network, we need to carefully choose which feature to use, color, inter-camera time, shape/size, or any combination of them. And finally we show that our $l\alpha\beta$ space model can give the same performance as of the BTF subspace model, however, with fewer parameters, and less computational cost.

# Chapter 4

# Activity learning

In the last Chapter, we discussed how to solve the correspondence problem, and stitch the trajectories together for every moving vehicle. In this Chapter, we will show how to learn the common trajectories throughout the overall scene and be able to detect unusual activities, given these learned trajectories.

## 4.1 Background Research

Many of the existing activity analysis approaches cluster moving objects' trajectories and detect abnormal trajectories by defining the pairwise similarities/distances between trajectories. The proposed trajectory similarities/distances include Euclidean distance [76], Hausdorff distance and its variations [77][78], hidden Markov model [79], and Dynamic Time Warping (DTW) [80]. A comparison of these different similarity/distance measures can be found in [81]. Based on the computed similarity matrix, some standard clustering algorithms such as spectral clustering [82], graph-cuts [83], agglomerative and divisive hierarchical clustering [84][106], and fuzzy c-means [86][87] were used to group trajectories into different activity categories. Finally, abnormal trajectories can be detected as those with a larger distance to other trajectories. These similarity/distance-based approaches have several drawbacks. First, there is no global probabilistic framework to model activities happening in the scene. They have an ad hoc nature especially on the definitions of distance measures. Abnormal

trajectories are usually detected as those with a larger distance to other trajectories. Their abnormality detection lacks a probabilistic explanation. Moreover, these methods all focused on one particular scene captured by one camera. For large area surveillance, we are faced with hundreds of non-overlapping sensors. The information of how objects are moving between cameras interests us more. As we discussed before, in our wide-scale network setting, the whole tracks have been segmented into different pieces of observed and unobserved tracks. Without the world coordinate system, there is no easy way to measure the distance between trajectories observed under different views. This kind of methods will fail in this situation.

In recent years, there have been some interesting work developed for detecting events or activities across multi-cameras [88][89][90][91][92]. Ahmedali and Clark [88] introduced a framework for a distributed network of collaborating and intelligent surveillance cameras. They proposed an unsupervised calibration technique which allows each camera module to represent its spatial relationship with other cameras in the network. For each camera, a person detector was trained using the Winnow algorithm with automatically extracted training samples. To improve detection performance, multiple cameras with overlapping FOVs collaborate to confirm results. Kim and Davis [90] proposed a multi-view multi-hypothesis approach for segmenting and tracking multiple people on a ground plane. To precisely locate the ground location of a person, all center vertical axes of the person across different views are mapped to the top view plane and their intersection point on the ground is estimated. Iterative segmentation-searching was incorporated into the tracking framework to deal with the explosive state space due to multiple targets and views. Remagnino et al. [91] proposed a multi-agent architecture for the understanding of scene dynamics merging the information streamed by multiple cameras. However, their multi-agent architecture can only deal with overlapping views.

Generative topic models[93] in language processing have drawn much attention with computer vision researchers recently, and have been applied to many vision applications, such as object recognition[97][96][95][94], scene categorization[98][99], and human action recognition[100][101]. The generative model also can be called a

$bag-of-words$ model because the model does not make any assumption about the order of words as they appear in documents. In the generative framework, each pixel on the trajectories can be treated as an independent motion feature vector (those features usually include location, speed, direction) to do the activity analysis, where the location feature is the relative image location. Stauffer et al.[27] proposed to generate a codebook of prototype representations using on-line Vector Quantization(VQ). Then for each sequence, it can be represented by the codebook labels. Their system leveraged this information by accumulating joint co-occurrences of the representations within a sequence. These joint co-occurrence statistics were then used to create a hierarchical binary-tree classification of the representations. This method is useful for classifying sequences as well as individual instances of activities in a site. Although Stauffer et al. only addressed this method in a single scene setting, we can see its potential to be extended to multiple non overlapping settings. Wang et al.[102] proposed to use the locations and moving directions of observations as features, and quantize them to visual words according to predefined codebook of its camera view. Then, they adopted an unsupervised hierarchical Bayesian model derived from LDA [104], which can cluster trajectories in different camera views into activities and models paths across camera views. The Dual-HDP model they proposed adds an extra layer of clustering to the original HDP clustering. Since we already know the correspondence from the last Chapter, we choose to use a mixture of unigram model instead, because it has fewer parameters, is easier to analyze and can be learned more efficiently.

## 4.2 Mixture of Unigram Activity Model

The $bag-of-words$ is an effective representation of documents, and the Naïve Bayes(NB) classifier based on this model works very well on many text classification and other clustering tasks [105][106]. Hence, we present a probabilistic framework (i.e. mixture of unigram model) which defines a generative process to model trajectories, and this framework has four elements:

Figure 4-1: Examples of concatenated trajectories between two different views. Trajectories generated by the same objects use the same color.

1. the data are produced by a mixture model.

2. there is a one to one correspondence between mixture components and clusters.

3. observations on any trajectory are independent (i.e. observations are generated independently).

4. the length of trajectories is independent of clusters.

Our probabilistic model will categorize concatenated trajectories in different camera views into activity clusters and model paths across camera views. We define an activity cluster as a collection of observations that commonly co-occur within a trajectory. Each activity cluster has a distribution over locations and moving directions in different camera views, and corresponds to a path.

### 4.2.1 Representation of Observations

After solving the correspondence problem as discussed in Chapter 3, we have concatenated trajectories over disjoint views as system input. Examples are shown in Figure 4-1 The locations and moving directions of observations of the moving vehicles are computed as features. For each camera view, we define a codebook as follows. The space of the view is uniformly quantized into small cells. Similarly, the velocity of the moving vehicles is quantized into several directions as well. Then, a global codebook concatenates the codebooks of all the camera views. Thus, for each concatenated

trajectory $i$, there are $J$ observations described as:

$$O_i = [o_{i,1}, \ldots, o_{i,j}, \ldots, o_{i,J}] \tag{4.1}$$

where

$$o_{i,j} = (x_{i,j}, y_{i,j}, v_{i,j}, c_{i,j}) \tag{4.2}$$

$(x_{i,j}, y_{i,j})$ and $d_{i,j}$ are the quantized moving vehicle's image position(i.e. coordinates) and its corresponding moving direction. $c_{i,j}$ is the camera view index in which the moving vehicle is observed. In this representation frame, each observation $j$ on trajectory $i$ has a word index of the global codebook associated with it.

### 4.2.2 The Generative Model of New Activity

In the mixture of unigram setting, every trajectory is represented by a probability distribution over the global codebook, defined by a set of parameters $\theta$. The probability distribution consists of a mixture of components (i.e. clusters) $c_j \in C = \{c_1, \ldots, c_{|C|}\}$. where $|C|$ denotes the number of clusters. So each activity cluster is parameterized by a disjoint subset of $\theta$ and modeled as a distribution over space and moving directions in all the camera views.

To generate a new trajectory $t_i$, we first sample an activity cluster index, $c$, according to the cluster prior probabilities $P(c_j)$. Then having this selected cluster fixed, we generate the trajectory according to its own parameters with distribution $P(t_i|c_j, \theta)$. To achieve this, we independently sample $|t_i|$ observations. For each single observation, we sample it from the corresponding activity cluster with distribution $P(o_{t_i,k}|c_j, \theta)$. From this process, we can characterize the likelihood of trajectory $t_i$ with a sum of total probability over all activity clusters:

$$\begin{aligned} P(t_i|\theta) &= \sum_{j=1}^{|C|} P(c_j) P(t_j|c_j, \theta) \\ &= \sum_{j=1}^{|C|} P(c_j) \prod_{k=1}^{|t_i|} P(o_{t_i,k}|c_j, \theta) \end{aligned} \tag{4.3}$$

Since we assume that for all clusters, the length of the trajectories is identically distributed, we do not need to parameterize for clustering. The only model parameters are the cluster prior probabilities $P(c_j)$, the multinomial distribution over the global codebook $P(o_{t_i,k}|c_j,\theta)$, and the number of clusters. Next, we will talk about how to learn the model parameters and set the number of clusters.

### 4.2.3 Learning Model Parameters

The model parameters can be estimated by a Maximum Likelihood method. As discussed in the previous section, by introducing latent variables, activity clusters, we can write the log-likelihood of the joint likelihood of the joint distribution of trajectories and model parameters as:

$$\begin{aligned} l(\mathbf{T};\theta) &\triangleq log(P(\mathbf{T}|\theta)) \\ &= log(\prod_{t_i \in \mathbf{T}} P(t_i|\theta)) \\ &= \sum_{t_i \in \mathbf{T}} log(\sum_{j=1}^{|C|} P(c_j) \prod_{k=1}^{|t_i|} P(o_{t_i,k}|c_j,\theta)) \end{aligned} \tag{4.4}$$

where $\mathbf{T}$ represents the collection of all the trajectories.

The Expectation-Maximization (EM)[107][108] algorithm is generally applied to maximize log likelihood. By using Jensen's inequality (i.e. $E[log(x)] \geq log(E[X])$ ). The parameters could be estimated by a hill climbing procedure defined by E-step and M-step:

- In the E-step, we compute the posterior probability $P(c_j|t_i,\theta)$ by using the current estimate of the parameters:

$$\begin{aligned} P(c_j|t_i,\theta)^{t+1} &= \frac{P(c_j)^t P(t_i|c_j,\theta)^t}{P(t_i|\theta)^t} \\ &= \frac{P(c_j)^t \prod_{k=1}^{|t_i|} P(o_{t_i,k}|c_j,\theta)^t}{\sum_{r=1}^{|C|} P(c_r)^t \prod_{k=1}^{|t_i|} P(o_{t_i,k}|c_j,\theta)^t} \end{aligned} \tag{4.5}$$

  where the upper script $t$ indicates the $t$-th iteration.

- In the M-step, by maximizing the complete likelihood equation, we update the model parameters using the current estimate of $P(c_j|t_i,\theta)$:

$$P(o_l|c_j,\theta)^{t+1} = \frac{1+\sum_{i=1}^{|\mathbf{T}|} P(c_j|t_i,\theta)^{t+1} * tf(o_l,t_i)}{N+\sum_{i=1}^{|\mathbf{T}|} P(c_j|t_i,\theta)^{t+1} * \sum_{s=1}^{N} tf(o_s,t_i)} \tag{4.6}$$

$$P(c_j)^{t+1} = \frac{\sum_{i=1}^{|\mathbf{T}|} P(c_j|t_i,\theta)^{t+1}}{|C|+|\mathbf{T}|} \tag{4.7}$$

  where $N$ is the vocabulary size of the global codebook, and $tf(o_l,t_i)$ is the count of the number of times $o_l$ occurs in trajectory $t_i$. Since the co-occurrence matrix is very sparse, we apply Laplace smoothing [105] to prevent zero probabilities for infrequently occuring observations.

The EM algorithm will increase the log likelihood consistently, while it will stop at a local maximum.

### 4.2.4 Estimating the Number of Clusters

Our clustering model requires choosing one additional parameter: the number of activity clustersn $k$. Though no measure is definitive, intuitively a good clustering is one in which the clusters are tight and well separated. In another word, we want the variance of inter-cluster to be large and the variance of intra-cluster to be small. In our probabilistic model, there is no straight way to define the variance of clusters. Intuitively, however, just like how to choose the number of eigen values in PCA, we can choose the number of clusters which could explain majority of the activities.

As illustrated in Figure 4-2, only a limited number of clusters are salient ones, so we could start with relative large number of clusters, run the EM algorithm, and learn the activity clusters. Then we choose the number of clusters whose prior distribution could account for a certain mount of the activities (in our experiment, we choose 90%). With this fixed number $k$, we run the algorithm one more time learn the activity cluster models.

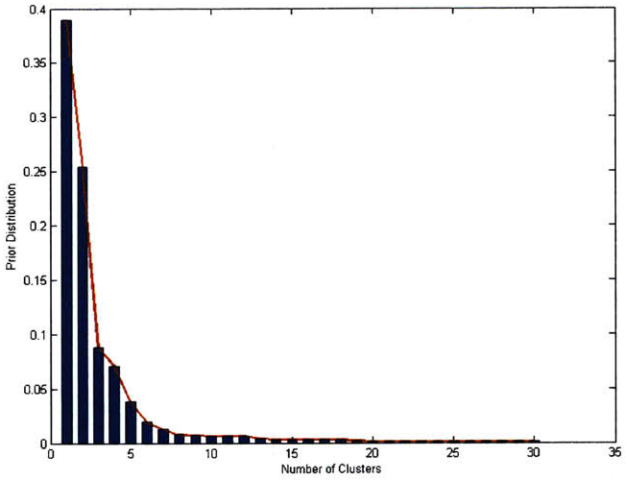


Figure 4-2: Prior distribution of the learned 30 activity clusters.

### 4.2.5 Using the Classifier to Label Trajectories into Activity Clusters

Given estimates of those model parameters (e.g. prior probability and multinomial distribution over the global codebook for every cluster), we can turn the generative model backwards and calculate the posterior probability that a particular activity cluster generated a given trajectory based on Eq.4.5.

We will use the Maximum Posterior principle (MAP) to label trajectories (including an unseen new trajectory). A trajectory will be assigned into an activity cluster with the highest posterior probability:

$$y_i = \underset{j}{\text{argmax}}(P(c_j|t_i, \theta)) \tag{4.8}$$

where $y_i$ is the label of the trajectory $t_i$.

### 4.2.6 Detecting Unusual Trajectories

In visual surveillance, detecting abnormal activities in the videos is of great interest. Under the Bayesian models, abnormality detection has a nice probabilistic explanation by the data likelihood of every trajectorie rather than by comparing similarity between samples. A trajectory is detected as an abnormality if it does not fit any activity model well. Given the estimation of model parameters, we can calculate the likelihood of a trajectory under the learned activity models using Eq.4.5. If the likelihood is smaller that a preset threshold, the trajectory will be flagged as an abnormal activity.

## 4.3 Experiment Results

We evaluate our algorithm on two data sets: the street scene and the parking lot scene again, respectively. The street scenes have 2 camera views and the parking lot scenes have 3 camera views, which are all non overlapping. The size of the views is 320 by 240. To build the codebook, each camera view is quantized into $64 \times 48$ cells. Each cell is of size 5 by 5. The moving directions of moving pixels are quantized into four directions (i.e. north east, north west, south east and south west). For each of the data sets, we first show the learned activity clusters, then plot the trajectories clustered into its corresponding cluster, and finally show the top 5 abnormal activities ranking from low to high based on the data likelihood. Also, we compare our results to the activity cluster results learned from LDA model [104] to demonstrate that our model gives a better performance.

### 4.3.1 Parking Lot Scene

The parking lot data was collected for 9 hours during the day time over 3 days (e.g. 3 hours per day for 3 days). It is a one way parking lot. The topology of these three camera views (see Figure 3-7) was learned using the technique presented in Chapter 2 and the tracking through different views was solved using the technique presented

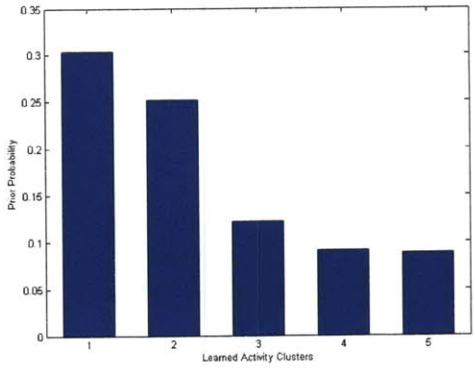


Figure 4-3: Prior distribution of the learned 5 activity clusters for the parking lot scene.

in Chapter 3. The views of these three cameras are not overlapped. The views of Camera 1 and 2 have a small gap. Hence, most vehicles appearing in Camera 1 will enter Camera 2, and vice versa. However, Camera 2 and 3 have a relative large gap. Camera 3 only captures a small area of the parking lot. Therefore, most vehicles appearing in Camera 2 will not reappear in Camera 3. Five different activity clusters are learnt from this data set (see Fig. 4-4 to 4-8). The prior distribution of these clusters is shown in Fig. 4-3. For each activity, we plot its distribution over space and moving directions in the three camera views and show the trajectories clustered into this activity. When visualizing activity clusters, moving directions are represented by different colors (red, yellow, blue and green), 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 Figure 4-4, the first row shows the learned activity cluster 1 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles entering the parking lot. These vehicles appear in Camera 1 and 2 , and may reappear in Camera 2 again later. However, they are not captured by Camera 3 because of the large gap between the views of Camera 2 and 3.

In Figure 4-5, the first row shows the learned activity cluster 2 and the second

Figure 4-4: Learned cluster 1: the first row shows the learned activity cluster 1 and the second row shows the trajectories clustered into this activity. When visualizing activity clusters, moving directions are represented by different colors (red, yellow, blue and green), 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. This cluster represents vehicles entering the parking lot. These vehicles appear in Camera 1 and 2 , and may reappear in Camera 2 again later. However, they are not captured by Camera 3 because of the large gap between the views of Camera 2 and 3.

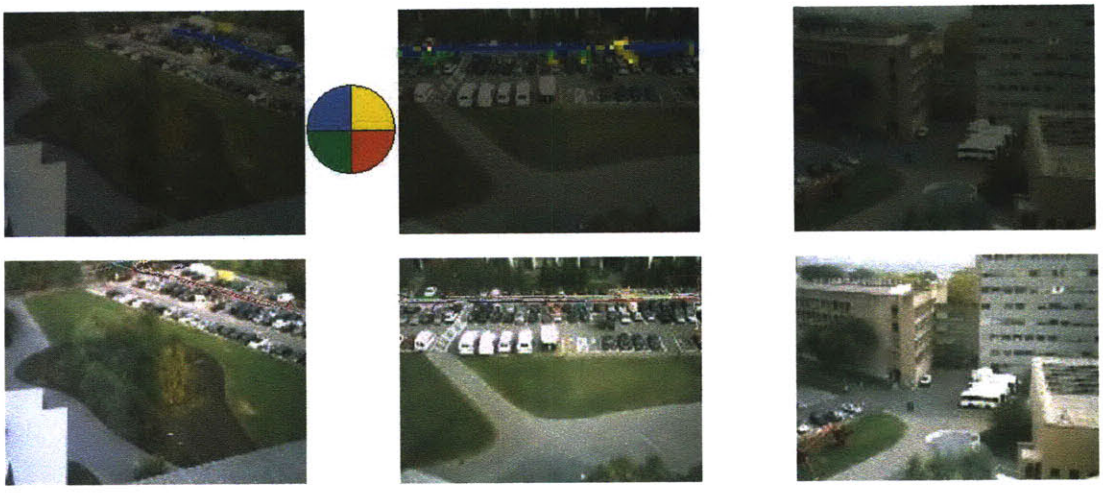

Figure 4-5: Learned cluster 2:the first row shows the learned activity cluster 2 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles leaving the parking lot. Again, these vehicles appear in Camera 2 and 1, and are not captured by Camera 3.

Figure 4-6: Learned cluster 3: the first row shows the learned activity cluster 3 and the second row shows the trajectories clustered into this activity. This cluster captures vehicles entering the parking lot and parking immediately without appearing at Camera 2.

row shows the trajectories clustered into this activity. This cluster represents vehicles leaving the parking lot. Again, these vehicles appear in Camera 2 and 1, and are not captured by Camera 3.

In Figure 4-6, the first row shows the learned activity cluster 3 and the second row shows the trajectories clustered into this activity. This cluster captures vehicles entering the parking lot and parking immediately without appearing at Camera 2.

In Figure 4-7, the first row shows the learned activity cluster 4 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles appearing in Camera 3 from the parking lot. Although most vehicles in Camera 2 will reappear in Camera 2 later, either parking or leaving the parking lot, there are small portion of them will enter camera 3 and go to the tech shuttle parking space. Cluster 4 just captures this kind of activity.

In Figure 4-8, the first row shows the learned activity cluster 5 and the second row shows the trajectories clustered into this activity. This cluster captures vehicles moving in the opposite direction compared with cluster 4.

Figure 4-9 to 4-13 show the top 5 abnormal activities. As we discussed above, the

Figure 4-7: Learned cluster 4:the first row shows the learned activity cluster 4 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles appearing in Camera 3 from the parking lot. Although most vehicles in Camera 2 will reappear in Camera 2 later, either parking or leaving the parking lot, there are small portion of them will enter camera 3 and go to the tech shuttle parking space.

Figure 4-8: Learned cluster 5

abnormality can be modeled by the low magnitude of data likelihood. The lower the data likelihood, the more abnormal the activity. Since in the parking lot, the moving path and moving direction for vehicles are well defined, most abnormal activities are the activities where vehicles move in the wrong directions (The car lane in the parking lot is one way) or use a path that isn't allowed. So we rank the detected abnormal activities from low data likelihood to high data likelihood, plot the trajectories in blue, and mark the starting and ending points of a trajectory by green plus and dot to indicate the moving directions. The top 5 are:

1. Figure 4-9 shows the most abnormal activity. A vehicle left Camera 3, entered Camera 2 with the wrong moving direction and then stopped. In this case, the moving direction in Camera 2 is unusual.

2. Figure 4-10 shows the activity with the second lowest data likelihood. A vehicle started in the view of Camera 2, drove through the parking lot in the wrong direction, entered the view of Camera 3, and headed to the tech shuttle parking area. Again, the moving direction of this activity is unusual.

3. Figure 4-11 shows a vehicle taking the path that it shouldn't take. In this case, it is the moving space which is unusual.

4. Figure 4-12 shows a vehicle driving through the parking lot, then making a right turn towards the building by the parking lot. In this case, it is the moving space which is unusual.

5. Figure 4-13 shows a vehicle backed in the parking lot. Again, it is the moving direction which is unusual.

Then we use the LDA model to learn the activity clusters, in order to compare the results with our mixture of unigram model directly, we set the number of clusters for LDA model to also be 5. The learned acitivity clusters are shown at Figure 4-14. The LDA model learns the same activity clusters as our mixture of unigram model. However, with the same performance, our model uses fewer parameters and can be learned more efficiently.

Figure 4-9: Abnormal activity 1: A vehicle left Camera 3, entered Camera 2 with the wrong moving direction and then stopped. In this case, the moving direction in the Camera 2 is unusual. The trajectories is ploted in blue, and the starting and ending points of the trajectory are marked by green plus and dot.

Figure 4-10: Abnormal activity 2: A vehicle started in the view of Camera 2, drove through the parking lot in the wrong direction, entered the view of Camera 3, and headed to the tech shuttle parking area. Again, the moving direction of this activity is unusual.

Figure 4-11: Abnormal activity 3: a vehicle took an unusual path.

Figure 4-12: Abnormal activity 4: a vehicle drove through the parking lot, then make a right turn towards the building by the parking lot. In this case, it is the moving space which is unusual.

Figure 4-13: Abnormal activity 5: a vehicle backed in the parking lot. Again, it is the moving direction which is unusual.

Figure 4-14: Learned 5 activity clusters using LDA model for the parking lot scene. Each row represents an activity cluster. The LDA model learns roughly same meaning activity clusters as our mixture of unigram model.

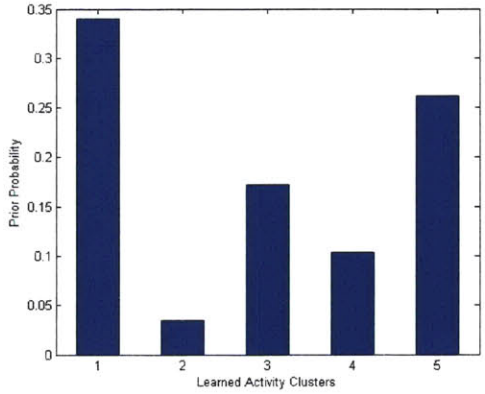


Figure 4-15: Prior distribution of the learned 5 activity clusters for the street scene.

### 4.3.2 Street Scenes

The street scenes data was collected for 1 hour in the morning over 5 days (i.e.9 AM from Monday to Friday). This scene has two camera views and the topology was learned using the technique presented in Chapter 2, as shown in Figure 3-6 and the tracking through different views was solved using the technique presented in Chapter 3. The views of these two cameras are not overlapped. There is a gap between the views of Camera 1 and Camera 2. Most vehicles appearing in Camera 1 will enter Camera 2, and vice versa. Five different activity clusters are learnt from this data set (see Fig. 4-16 to 4-20). The prior distribution of these clusters is shown in Fig. 4-15. Again, for each activity, we plot its distribution over space and moving directions in the two camera views and show the trajectories clustered into this activity. When visualizing activity clusters, moving directions are represented by different colors (red, yellow, blue and green), 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 Figure 4-16, the first row shows the learned activity cluster 1 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles

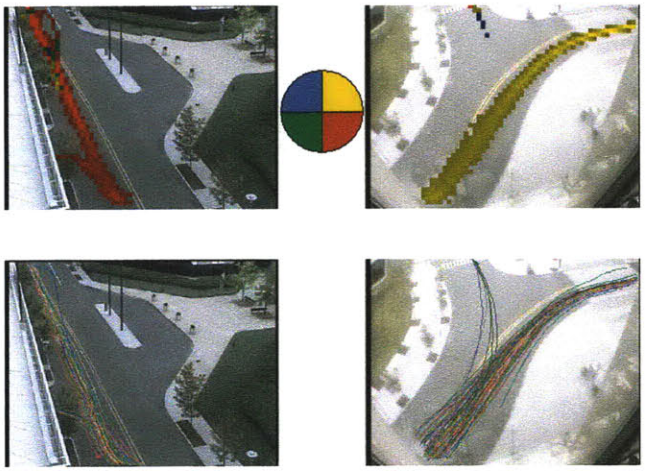

Figure 4-16: Learned activity cluster 1: the first row shows the learned activity cluster 1 and the second row shows the trajectories clustered into this activity. This cluster represents vehicle entering the view of Camera 1 from the upper left corner, driving through the view of Camera 1, then reappearing at the lower left corner of Camera 2' view, and finally leaving the scene.

entering the view of Camera 1 from the upper left corner, driving through the view of Camera 1, then reappearing at the lower left corner of Camera 2's view, and finally leaving the scene. This is the most common activity for this street scene.

In Figure 4-17, the first row shows the learned activity cluster 2 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles entering the view of Camera 1 from the upper left corner, making a U turn at the island, and then leaving the scene without reentering the view of Camera 2.

In Figure 4-18, the first row shows the learned activity cluster 3 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles entering the view of Camera 1 from the upper left corner, either driving into the garage with higher probability, or driving through the view of Camera 2 with lower probability, and then reappearing at the view of Camera 2 with very low probability. Comparing with the learned activity cluster 1, this cluster most describes the activities happening only at the view of Camera 1. Since the probability of reentering the view of Camera 2 is pretty low, it shows that there may exist a hidden sink for Camera 1, which means vehicles will leave the view of Camera 1 and go to another place without

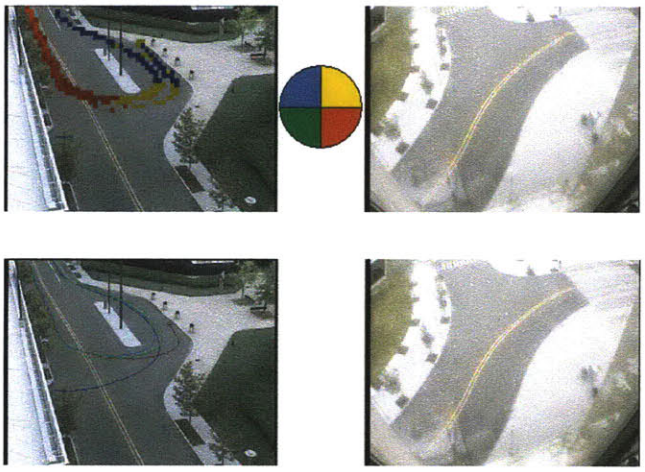

Figure 4-17: Learned activity cluster 2: the first row shows the learned activity cluster 2 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles entering the view of Camera 1 from the upper left corner, making a u turn at the island, and then leaving the scene without reentering the view of Camera 2.

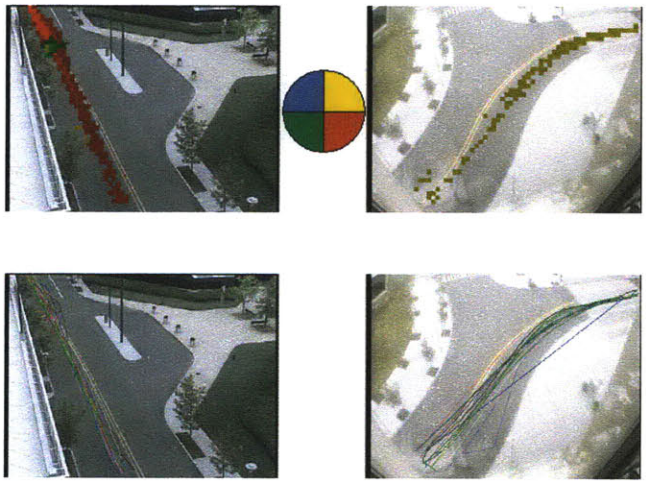

Figure 4-18: Learned activity cluster 3: the first row shows the learned activity cluster 3 and the second row shows the trajectories clustered into this activity. This cluster represents vehicles entering the view of Camera 1 from the upper left corner, either driving into the garage with high probability, or driving through the view of Camera 2 with lower probability, and then reappearing at the view of Camera 2 with very low probability.

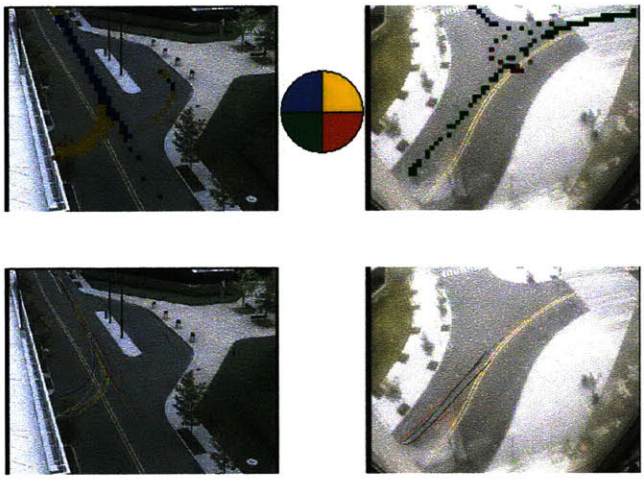

Figure 4-19: Learned activity cluster 4: this cluster represents activities similar to cluster 3 only in the opposite moving direction. Vehicles pull out from the garage, drive through the view, and finally leave the scene. Again, this cluster most captures the activities happening only at the view of Camera 1.

reentering the view of Camera 2.

Figure 4-19 shows the learned activity cluster 4 and the trajectories clustered into this activity. This cluster represents activities similar to cluster 3 only in the opposite moving direction. Vehicles pull out from the garage, drive through the view, and finally leave the scene. Again, this cluster most captures the activities happening only at the view of Camera 1.

Figure 4-20 shows the learned activity cluster 5 and the trajectories clustered into this activity. This cluster represents activities similar to cluster 1 only in the opposite moving direction. Vehicles enter the view of Camera 2 from the upper right corner, driving through the view of Camera 2, then reappearing at the lower right corner of Camera 1' view, and finally leaving the scene.

Figure 4-21 to 4-25 show the top 5 abnormal activities happening in the street scene. Also in this scene, the moving path and moving direction for vehicles are well defined, most abnormal activities are the activities that vehicles move in the wrong directions (e.g. in the wrong lane) or take a path not allowed. We rank the detected abnormal activities from low data likelihood to high data likelihood, plot

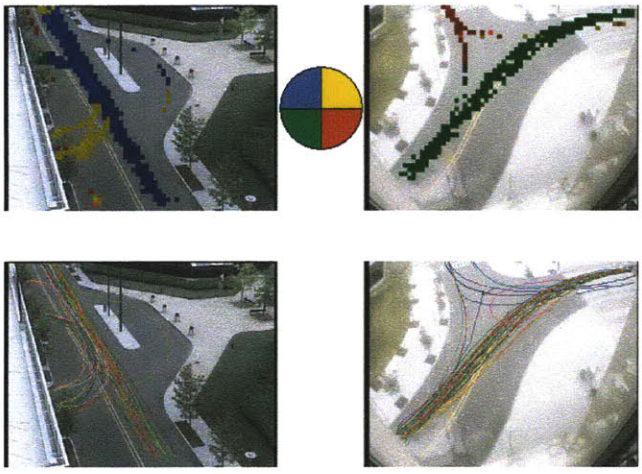

Figure 4-20: Learned activity cluster 5: this cluster represents activities similar to cluster 1 only in the opposite moving direction. Vehicles enter the view of Camera 2 from the upper right corner, driving through the view of Camera 2, then reappearing at the lower right corner of Camera 1' view, and finally leaving the scene.

the trajectories in blue, and mark the starting and ending points of a trajectory by green plus and dot to indicate the moving directions. The top 5 are:

1. Figure 4-21 shows the most abnormal activity. A vehicle left the view of Camera 2, entered Camera 1, then made an interesting U turn at the dropping area, and finally left the scene. The learned moving direction around the dropping area is north west, however, this vehicle moved in the opposite direction around the area, which makes it unusual.

2. Figure 4-22 shows the activity with the second lowest data likelihood. This abnormality is caused by the tracker. When a vehicle stopped and dropped off a person, the tracker didn't separate the person's trajectory from the vehicle's trajectory. Hence, it looked like a vehicle made an illegal right turn. In this case, it is the moving space which is unusual.

3. In Figure 4-23, when a vehicle entered the view of Camera 2, it didn't stay at the right lane, and shifted to the opposite lane which is very danger. It is the moving direction which is unusual.

Figure 4-21: Abnormal activity 1: a vehicle left the view of Camera 2, entered Camera 1, then made an interesting U turn at the dropping area, and finally left the scene. The learned moving direction around the dropping area is north west, however, this vehicle moved in the opposite direction around the area, which makes it unusual. The trajectory is plot in blue, and the starting and ending points of the trajectory are marked by green plus and dot.

Figure 4-22: Abnormal activity 2: when a vehicle stopped and dropped off a person, the tracker didn't separate the person's trajectory from the vehicle's trajectory. Hence, it looked like a vehicle made an illegal right turn. In this case, it is the moving space which is unusual. The trajectory is plot in blue, and the starting and ending points of the trajectory are marked by green plus and dot.

4. Figure 4-24 shows a vehicle driving through the view of Camera 1, making an U turn in the view of Camera 2, and finally leaving the scene. It is the moving direction causing the abnormality.

5. Figure 4-25 shows another example of vehicles shifting to the opposite lane.

We should notice that in our model, an abnormal activity is defined as an activity that couldn't fit to the learned activity clusters. Sometimes, a seldom occuring activity doesn't necessarily mean an abnormal activity. For example, in our street scene case, although detected abnormal activity 1 and 3 are flagged as abnormal activities by our model, they are not illegal operations and quite normal. In this circumstance, human intervention is needed to supervise.

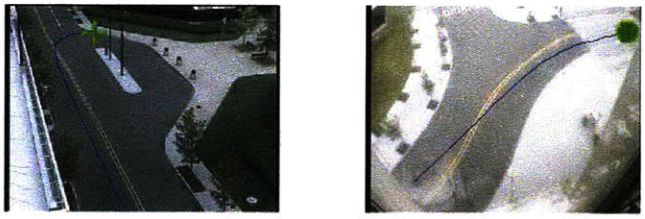

Figure 4-23: Abnormal activity 3: when a vehicle entered the view of Camera 2, it didn't stay at the right lane, and shifted to the opposite lane which is very danger. It is the moving direction which is unusual. The trajectory is plot in blue, and the starting and ending points of the trajectory are marked by green plus and dot.

Figure 4-24: Abnormal activity 4: a vehicle drove through the view of Camera 1, made an U turn in the view of Camera 2, and finally left the scene. It is the moving direction causing the abnormality. The trajectories is plot in blue, and the starting and ending points of the trajectory are marked by green plus and dot.

Figure 4-25: Abnormal activity 5: another example of vehicles shifting to the opposite lane.

Then we use the LDA model to learn the activity clusters, in order to compare the results with our mixture of unigram model directly, we also set the number of clusters for LDA model to be 5. The learned acitivity clusters are shown at Figure 4-26. Compared with the mixture of unigram model, LDA model doesn't capture the activities that vehicles make U turns at the view of camera 1. The other four activity clusters are both successfully learned by our mixture of unigram model and LDA model. With fewer parameters and less computational cost, our model outperforms the LDA model.

## 4.4 Summary

In this Chapter, we showed how to learn the activity clusters and detect abnormal activities using the mixture of unigram model with the stitched trajectories as input. We adopt a $bag - of - words$ presentation, and present a Bayesian probabilistic approach in which trajectories are represented by a mixture model. This model can classify trajectories into different activity clusters, and gives representation of both new trajectories and abnormal trajectories.

First, our method define a global codebook of observations that are representative of the entire set of observations captured by different cameras. Using this codebook to represent our continuous observations through different views, we can represent the likelihood of a trajectory by a mixture of unigram model. By using EM algorithm, we can learn the model parameters, and label trajectories into different activity cluster. Finally, we can detect abnormal activities if they does not fit any learned activity model well.

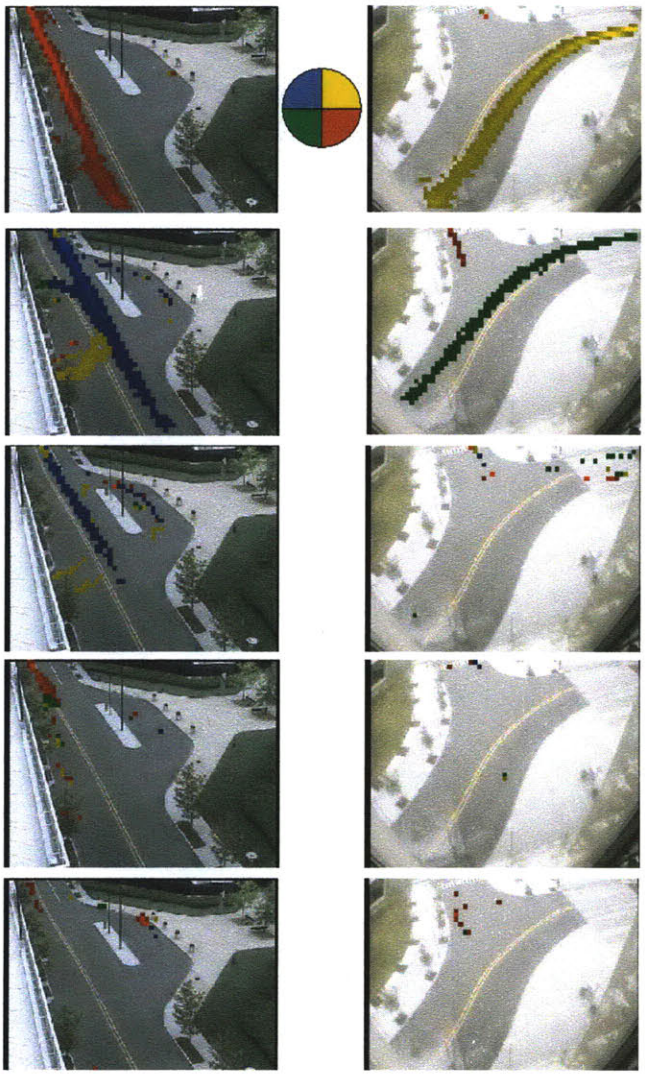

Figure 4-26: Learned 5 activity clusters using LDA model. Each row represents an activity cluster. Compared with the mixture of unigram model, LDA model doesn't capture the activities that vehicles make U turns at the view of camera 1. The other 4 activity clusters are both successfully learned by our mixture of unigram model and LDA model.

# Chapter 5

# Conclusions

The motion pattern learning framework introduced in this thesis shows promise in bootstrapping perceptual intelligence for multi camera surveillance systems. Given a primitive system that detects and tracks the presence of moving objects in any single sensor, this system can learn the topology of the network, build models of appearance change in different views, track objects in multiple sensors, build models of activities, and determine events that are uncharacteristic of the environment.

## 5.1 Contributions

The primary contribution of this thesis was to motivate this problem and to advocate our data driven framework. Our framework exploits different aspects of the real world learning problem in a particular order to boost the tracking accuracy between different views and help to learn the activity clusters.

Secondary contributions include each individual component of the motion pattern learning framework. First, we present a weighted cross correlation model to learn the topology of the network without solving correspondence in the first place. A coarse appearance model is constructed by the combination of the normalized color and overall size model to measure the moving objects appearance similarity across the non-overlapping views. Our general correspondence framework incorporated both appearance and inter camera time correspondence and shows promise for long-term

correspondence. The most essential element is the $l\alpha\beta$ space color transformation model. This general modeling technique is quantitatively shown be to be very stable and can improve the tracking accuracy result dramatically with fewer parameters and less computational cost. Finally, we adopt a $bag-of-words$ presentation, and present a Bayesian probabilistic approach in which trajectories are represented by a mixture model. This model not only can classify trajectories into different activity clusters, but also gives representation of both new trajectories and abnormal trajectories.

## 5.2 Applications

The most intriguing aspect of the motion pattern learning framework is the potential applications to the real world:

1. Data summary and retrieval. Imagine you are given hundreds of videos, possibly thousands of hours in duration for each one, depicting every day scenes like shopping malls, train stations, or security sensitive buildings. You are asked to give the summary of the data (i.e. what are the common activities) and be able to do data retrieval. Having built a statistical model of activities in the scene, we can use the model to find other activities that are similar, i.e. activities that fit the same cluster. For instance, given a particular example of activity, we can retrieve other examples that have been classified as belonging to the same cluster. Additionally, we could also flag any future activity that is similar to a particular example or that comes from a specific cluster in our model.

2. For many applications the unusual activities detection are of most interest. For example, a vehicle leaves the scene at an unusual place (e.g. in the middle of the road), or a vehicle speeds up towards a government building. As we discussed before, most visual surveillance depends on a human operator to sift through videos. However, it is a tiring, expensive, and tedious job, monitoring for interesting events that rarely occur. The methods from this thesis could be applied to filter out activities that are common based on long-term observation

of the scene and flag any possible abnormal activity. As a result, a single operator would be able to oversee a much larger network of sensors that has previously been impossible.

## 5.3 Limitations and Future Work

In our topology learning step, one limitation is that it can only learn the topology with one popular transition time between disjoint views. If the transition time is multi-modal, one possible way to solve it is to estimate the mutual information directly, which means to estimate the joint distribution and marginal distribution of variables, which we will explore in future work.

In Chapter 3, we showed that although the shape/size information is helpful to determine the correspondence, it will jeopardize the tracking accuracy when the shape/size information isn't stable. Especially, in a crowded scene, when there are substantial amount of people and vehicle moving around, how to accurately tracking individual moving objects without grouping them together becomes very essential and the performance of the tracker will impact our motion learning framework directly. Although, how to track moving objects in a single view isn't the focus of our system, we would like to explore it in the future and to build a more reliable tracker.

Another limitation of our work is that we can only model the motion patterns of individual moving objects, and cannot handle the interactions between them. For instance, if there is a vehicle want to make a left turn, it should yield in the middle of the intersection until the road is clear and no more vehicles comes toward it. Our future work will be focus on how to incorporate interaction modeling into our system. Olive et. al. [109] proposed to use Coupled HMM to learn human interactions, and Wang et. al. [103] used a Dual-DHP model to model the interactions of multi agents. which we will further investigate.

# Bibliography

[1] SearchSecurityAsia editors, "Research shows video surveillance market maintains 10% growth despite recession", SearchSecurityAsia.com, April 2, 2009.

[2] W. Hu, T. Tan, L. Wang, and S. Maybank. "A Survey on Visual Surveillance of Object Motion and Behaviors". IEEE Transactions on Systems, Man, Cybernetics-Part C: Applications and Reviews, 34:334352, 2004.

[3] B. T. Morris and M. M. Trivedi. "A survey of Vision-based Trajectory Learning and Analysis for Surveillance". IEEE Trans. on Circuits and Systems for Video Technology, 18:11141127, 2008.

[4] N. Haering, P. L. Venetianer, and A. Lipton, "The evolution of video surveillance: an overview", Machine Vision and Applications, 19:279290, 2008.

[5] I. Haritaoglu, D. Harwood, and L. S. Davis, "W : Real-time Surveillance of People and their Activities", IEEE Trans. Pattern Anal. Machine Intell., vol. 22, pp. 809830, Aug. 2000.

[6] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving Target Classification and Tracking from Real-time Video", in Proc. IEEE Workshop Applications of Computer Vision, 1998, pp. 814.

[7] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking Groups of People", Comput. Vis. Image Understanding, 80(1):4256, 2000.

[8] D. Meyer, J. Denzler, and H. Niemann, "Model Based Extraction of Articulated Objects in Image Sequences for Gait Analysis", in Proc. IEEE Int. Conf. Image Processing, pp. 7881, 1998.

[9] D. Meyer, J. Psl, and H. Niemann, "Gait Classification with HMMs for Trajectories of Body Parts Extracted by Mixture Densities", in Proc. British Machine Vision Conf., pp. 459468, 1998.

[10] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time Tracking of the Human Body", IEEE Trans. Pattern Anal. Machine Intell., 19:780785, 1997.

[11] W. F. Gardner and D. T. Lawton, "Interactive Model-Based Vehicle Tracking", IEEE Trans. Pattern Anal. Machine Intell., 18:11151121, 1996.

[12] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y.Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L.Wixson, "A System for Video Surveillance and Monitoring", Carnegie Mellon Univ., Tech. Rep., CMU-RI-TR-00-12, 2000.

[13] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving Target Classification and Tracking from Real-time Video", in Proc. IEEE Workshop Applications of Computer Vision, pp. 814, 1998.

[14] J. G. Lou, Q. F. Liu, W. M. Hu, and T. N. Tan, "Semantic Interpretation of Object Activities in a Surveillance System", in Proc. Int. Conf. Pattern Recognition, 2002.

[15] Chris Stauffer. "Estimating Tracking Sources and Sinks," Conference on Computer Vision and Pattern Recognition Workshop, 2003.

[16] Chris Stauffer, Eric Grimson,"Learning Patterns of Activity Using Real-Time Tracking", IEEE Transactions on Pattern Recognition and Machine Intelligence (TPAMI), 22(8):747-757, 2000.

[17] Reid D B. "An algorithm for tracking multiple targets", AC, 24(6):843-854, December 1974.

[18] Chris Stauffer, "Adaptive Background Mixuture Models for Real-Time Tracking", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, July, 1999.

[19] Isard M, MacCormick J P, "Bramble: A Baysian Multiple-Blob Tracker", IEEE International Conference on Computer Vision, October, 2001.

[20] Williams O, Blake A and Cipolla R. "A Sparse Probabilistic learning algorithm for real-time tracking". IEEE International Conference on Computer Vision, October, 2003.

[21] D. Lowe, "Distinctive Image Features from Scale Invariant Features", International Journal of Computer Vision, 60(2):91-110, 2004.

[22] Jain R, Wakimoto K, "Multiple perspective interactive video", IEEE International Conference on Multimedia Computing and System, 1995.

[23] Collins R, Lipton A, Fujiyoshi H, and Kanade T, "A system for video survillance and monitoring", In Proc. American Nuclear Society(ANS) Eighth International Topical Meeting on Robotic and Remote Systems, April 1999.

[24] Ali Rahimi, Brian Dunagan, Trevor Darrell,"Simultaneous Calibration and Tracking with a Network of Non-overlapping Sensors", IEEE International Conference on Computer Vision and Pattern Recognition, June, 2004.

[25] Huang T, Russell S, "Object identification in a Bayesian context",in Proc. of IJCAI, 1997.

[26] Omar Javed, Zeeshan Rasheed, Khurram Shafique, Mubarak Shah, "Tracking Across Multiple Cameras With Disjoint Views", IEEE International Conference on Computer Vision, October, 2003.

[27] Kang J, Cohen I, Medioni G,"Continuous Tracking Within and Across Camera Streams",Proc. Computer Vision and Pattern Recognition, July, 2003.

[28] Dailey D, "Travel Time Estimation Using Cross Correlation Techniques", Transportation Research Part B, vol 207B, No 2, pp97-107, 1993

[29] Petty, K, et al, "Accurate Estimation of Travel Times From Single Loop Detectors", paper presented at the 76th annual Transprotation Research Part meeting, 1997.

[30] Westerman M, Immers L, "A Method for Determining Real-Time Travel Times on Motorways", Road Transport Informatics/Intelligent Vehicle Highways Systems, ISATA, pp 221-228, 1992.

[31] Makis D, Ellis T, et al, "Learning a Multicamera Topology", Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, October, Nice, France, 2003.

[32] Http://en.wikipedia.org/wiki/Cross-correlation

[33] Finlayson G D, Schiele B, and Crowley J L, "Comprehensive Colour Image Normalization". Proc. the European Conf. on Computer Vision, 1998.

[34] Biswajit Bose, Eric Grimson, "Ground Plane Rectification by Tracking Moving Objects". Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), Nice, France, October 2003.

[35] Cover T M, Thomas J A, "Elements of Information Theory". Wiley, 1991.

[36] Singhal M, "A Dynamic Information-Structure Mutual Exclusion Algorithm for Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, 1992.

[37] Brillinger D R,"Second-order moments and mutual information in the analysis of time series", Recent Advances in Statistical Methods. Imperial College Press, London, 2002.

[38] Finn Jensen. "An Introduction to Bayesian Networks". Spring, 1996.

[39] Kullback S, "Information Theory And Statistics". Dover, 1968.

[40] Wentian Li, "Mutual Information Functions Versus Correlation Functions". Journal of Statistical Physics, 60(5-6):823-837 1990.

[41] Smith J.R., Chang S.F, "Visual SEEk: a fully automated content-based image query system", 1996. http://www.ctr.columbia.edu/Bjrsmith/html/pubs/acmmm96/acmfin.htmlS (April 11 2001).

[42] S. P. LLOYD, "Least Squares Quantization in PCM", IEEE Transaction in Information Theroy, 28(2):129-137, 1982.

[43] R. Jain and K. Wakimoto. "Multiple perspective interactive video". In IEEE International Conference on Multimedia Computing and Systems, 1995.

[44] Q. Cai and J.K. Aggarwal. "Tracking human motion in structured environments using a distributed camera system". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(11):12411247, 1999.

[45] R.T. Collins, A.J. Lipton, H. Fujiyoshi, and T. Kanade. "Algorithms for cooperative multisensor surveillance". Proceedings of IEEE, 89(10):14561477, 2001.

[46] L. Lee, R. Romano, and G. Stein. "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame". IEEE Trans. on PAMI, 22:758768, 2000.

[47] T. H. Chang, and S. Gong. "Tracking Multiple People with a Multi-Camera System". In IEEE Workshop on Multi-Object Tracking, 2001.

[48] Y. A. Sheikh, and M. Shah. "Trajectory Association Across Multiple Airborne Cameras". IEEE Trans. on PAMI, 30:361367, 2008.

[49] C. Stauffer, and K. Tieu. "Automated Multi-camera Planar Tracking Correspondence Modeling". In Proceedings of IEEE International Conference of Computer Vision and Pattern Recognition, 2003.

[50] Javed O, Shafique K, Shah N, "Appearance Modeling for Tracking in Multiple Non-overlapping Cameras", Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2:26-33, 2005.

[51] Kettnaker V, Zabih R, "Bayesian Multi-camera Surveillance", Proceddings of the International Conference on Computer Vision and Pattern Recognition, 1999.

[52] Prosser B, Gong S, Xiang T, "Multi-camera matching using bi-directional cumulative brightness transfer functions". British Machine Vision Conference, 2008.

[53] Cheng E, Piccardi M, "Matching of Objects Moving Across Disjoint Cameras", Proceedings of the International Conference on Image Processing, 2006.

[54] Kuhn H W, "The Hungarian Method for the Assignment Problem", Naval Research Logistics, 2:83-97, 1955.

[55] Bassiou N, Kotropouls C, "Color Image Histogram equalization by Absolute Discounting Back-off", Computer Vision and Images Understanding, 107:108-122, 2007.

[56] Grundland M, Dodgson N A, "Color Histogram Specification by Histogram Warping", Proceedings of SPIE, 5667:610-621, 2005.

[57] Weeks A R, Sartor L J, Myler H R, "Histogram specification of 24-bit color images in the color difference(C-Y) color space", Journal of Electronic Imaging, 8(3):290-300, 1999.

[58] Guo Y, Shan Y, Sawhney H, Kumar R, "PEET: Prototype Embedding and Embedding Transition for Matching Vehicles over Disparate Viewpoints", Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2007.

[59] Shan Y, Sawhney H, Kumar R, "Unsupervised Learning of Discriminative Edge Measures for Vehicle Matching between Non-Overlapping Cameras", IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(4):700-711, 2008.

[60] Guo Y, Rao C, Samarasekera S, Kim J, Kumar R, Sawhney, "Matching vehicles under large pose transformations using approximate 3D models and piecewise MRF model", Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2008.

[61] Ferencz A, Learned-Miller E, Malik J, ""Learning to Locate Informative Features for Visual Identification", International Journal of Computer Vision, 77:3-24, 2008.

[62] Shan Y, Sawhney H, Kumar R, "Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching", IEEE International Conference on Computer Vision, 2005.

[63] Arth C, Leistner C, Bischof H, "Object Reacquistion and Tracking in Large-Scale Smart Camera Networks ", In Proceddings of the First ACM/IEEE International Conference on Distributed Smart Cameras,2007.

[64] Csink L, Paulus D, Ahlrichs U, Heigl B, "Color Normalization and Object Localization", http://www.uni-koblenz.de/ agas/Documents/Csink1998CNA.pdf.

[65] Ilie A, Welch G, "Ensuring color consistency across multiple cameras", IEEE International Conference on Computer Vision, 2:1268-1275, 2005.

[66] Gonzalez R C, Woods R E, "Digital Image Processing", Second Edition, 2002.

[67] Heeger D J, Bergen J R, "Pyremid-Based Texture Analysis/Synthesis", In Proceedings of SIGGRAPH, 1995.

[68] Gillespie A R, Kahle A B, Walker R E, "Color Enhancement of Highly Correlated Images. Decorrelation and Hsi Contrast Stretches", Remote Sensing of Environment, 20(3):209-235, 1986.

[69] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, "Color Transfer between Images", IEEE Computer Graphics and Applications, 21(5):34-41, 2001.

[70] I. T. Jolliffe, "Principal Component Analysis", SpringerVerlag, New York, 1986.

[71] D.L.Ruderman, T.W.Cronin, and C.C.Chiao, "Statistics of Cone Responses to Natural Images: Implications for Visual Coding", Jounal of Optical Soc. of American, 15(8):2036-2045, 1998.

[72] Wyszecki G, Stiles W S, "Color Science: Concepts and Methods, Quantitative Data and Formulae", 2nd edtion, New York, 1982.

[73] Sun P, Freund R M, "Computation of Minimum-Volume Covering Ellipsoids", Operations Research, 52(5):690-706, 2004.

[74] Moshtagh N, "Minimum Volume Enclosing Ellipsoids", GRASP Laboratory, University of Pennsylvania. Available online: http://www.seas.upenn.edu/$\sim$ nima/index$_f iles/Mim_v ol_e llipse.pdf$.

[75] Kumar P, Yildirim E A, "Minimum Volume Enclosing ellipsoids and core sets", Journal oF Optimization Theory and Applications, 2005.

[76] Fu Z, Hu W, and Tan T, "Similarity Based Vehicle Trajectory Clustering and Anomaly Detection". In Proceedings of IEEE International Conference of Image Processing, 2005.

[77] Junejo I, Javed O, and Shah M. "Multi Feature Path Modeling for Video Surveillance". In Proceedings of Internatinal Conference of Pattern Recognition, 2004.

[78] Wang X, Tieu K, and Grimson W E. "Learning Semantic Scene Models by Trajectory Analysis". In Proceedings of European Conf. Computer Vision, 2006.

[79] Porikli F and Haga T. "Event Detection by Eigenvector Decomposition Using Object and Frame features". In Proceedings of IEEE International Conference of Computer Vision and Pattern Recognition Workshop, 2004.

[80] Keogh E, and Pazzani M. "Scaling Up Dynamic Time". In Proceedings of ACM Special Interest Group on Knowledge Discovery and Data Mining, 2000.

[81] Z. Zhang, K. Huang, and T. Tan. "Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes". In Proceedings of International Conference of Pattern Recognition, 2006.

[82] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On Spectral Clustering: Analysis and an Algorithm". In Proceedings of Neural Information Processing Systems Conference, 2002.

[83] J. Shi, and J. Malik. "Normalized Cuts and Image Segmentation". IEEE Trans. on PAMI, 22:888905, 2000.

[84] D. Biliotti, G. Anotonini, and J. P. Thiran. "Multi-layer Hierarchical Clustering of Pedestrian Trajectories for Automatic Counting People in Video Sequences". In Proceedings of IEEE Workshop on Motion and Video Computing, 2005.

[85] X. Li, W. Hu, and W. Hu. "A Coarse-to-Fine Strategy for Vehicle Motion Trajectory Clustering". In Proceedings of International Conference of Pattern Recognition, 2006.

[86] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. "A System for Learning Statistical Motion Patterns". IEEE Trans. on PAMI, 28:14501464, 2006.

[87] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank. "Semantic-based Visual Surveillance". IEEE Trans. on Image Processing, 16:11681181, 2007.

[88] T. Ahmedali and J.J. Clark, "Collaborative Multi-Camera Surveillance with Automated Person Detection", Canadian Conference on Computer and Robot Vision, 2006.

[89] S. Lim, L.S. Davis and A. Elgammal, "A Scalable Image-Based Multi-Camera Visual Surveillance System", Conference on Advanced Video and Signal Based Surveillance, 2003.

[90] K. Kim and L.S. Davis, "Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering", ECCV, 2006.

[91] P. Remagnino, A.I. Shihab and G.A. Jones, "Distributed Intelligence for Multi-Camera Visual Surveillance", Pattern recognition, 2004.

[92] H. Zhou and D. Kimber, "Unusual Event Detection via Multi-camera Video Mining", Intl Workshop on ICPR, 2006.

[93] M. Steyvers, "Probabilistic Topic Model", In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors, Latent Semantic Analysis: A Road to Meaning. 2006.

[94] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. "Discovering object categories in image collections". In Proceeding of International Conference on Computer Vision, 2005.

[95] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. "Using Multiple Segmentations to Discover Objects and Their Extent in Image Collections." In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 2006.

[96] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. "Describing Visual Scenes Using Transformed Objects and Parts". International Journal of Computer Vision, 77:291330, 2007.

[97] X. Wang and E. Grimson. "Spatial latent dirichlet allocation". In Proc. Neural Information Processing Systems Conference, 2007.

[98] L. Fei-Fei, and P. Perona. "A bayesian Hierarchical Model for Learning Natural Scene Categories". In Proc. IEEE International Conference of Computer Vision and Pattern Recognition, 2005.

[99] C. Wong, D. Blei, and L. Fei-Fei, "Simultaneous Image Classification and Annotation", In Proceedings of IEEE Intrernation Conference of Computer Vision and Pattern Recognition, 2009.

[100] J. C. Niebles, H. Wang, and F. Li. "Unsupervised Learning of Human Action Categories using Spatial-Temporal Words". In Proc. British Machine Vision Conference, 2006.

[101] S. Wong, T. Kim, and R. Cipolla, "Learning motion categories using both semantic and structural information", in Proceedings of IEEE Internation Conference of Computer Vision and Pattern Recognition. 2007.

[102] X. Wang, T. Kieu, and E. Grimson, "Correspondence-Free Multi-Camera Activity Analysis and Scene Modeling", In Proceedings of IEEE Computer Society Conference on Computer Vision and Patter Recognition, 2008.

[103] X. Wang, X. Ma and E. Grimson, "Unsupervised Activity Perception by Hierarchical Bayesian Models", in Proceedings of IEEE Computer Society Conference on Computer Vision and Patter Recognition, 2007.

[104] D. M. Blei, A. Y. Ng, and M. J. Jordan, "Latent Dirichlet Allocation". In Journal of Machine Learning Research, 3:993-1022, 2003.

[105] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents using EM", Maching Learning, 2000.

[106] Z. Li, B. Wang, M. Li, and W. Ma, "A Probabilistic Model for Retrospective News Event Detection", In Proceedings of SIGIR, 2005.

[107] A. P. Dempster, N.M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society, Series B, 39(1):1-38, 1977.

[108] R.O. Duda, P.E. Hart, and D.G. Stork. "Pattern Classification", section 3.9, Expectation-Maximization (EM), Wiley-Interscience, New York, NY, second edition, 2001.

[109] N. Oliver, B. Rosario, and A. Pentland. "A bayesian Computer Vision System for Modeling Human Interactions;;. IEEE Trans. on PAMI, 22:831843, 2000.