UNIVERSITY OF CALIFORNIA, SAN DIEGO

Understanding Activity from Trajectory Patterns

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical and Computer Engineering

by

Brendan Tran Morris

Committee in charge:

Professor Mohan Trivedi, Chair
Professor Serge Belongie
Professor Vistasp Karbhari
Professor Kenneth Kreutz-Delgado
Professor Nuno Vasconcelos

2010

The dissertation of Brendan Tran Morris is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2010

DEDICATION

To the Faithful Departed.

EPIGRAPH

Tommy: Did you hear I finally graduated?

Richard Hayden: Yeah, and just a shade under a decade too. All right.

Tommy: You know a lot of people go to college for seven years.

Richard Hayden: I know, they're called doctors.

Tommy Boy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Coming to UCSD has been one of the most amazing experiences in my life. When I started graduate school I was not sure what I wanted to do, just that I was not ready to go to work for the rest of my life. I never could have imagined how fulfilling and rewarding the process would be.

Over the course of my graduate work I have come across so many fantastic people that have helped shape my life. There is no way I could single out every one and it is impossible for me to articulate my gratitude and how much each person has affected me on this PhD journey. I am a forgetful person and will leave out many who deserve much credit (and after seven years there are a great many) but this is for everyone who stood by me.

In particular, I want to extend a heartfelt thanks to my respected committee. Dr. Belongie, Dr. Vasconcelos, and Dr. Kreutz-Delgado thank you for keeping an open door and a critical ear. Your questions and guidance have significantly contributed to this work. Dr. Karbhari receives special distinction for remaining on my committee even after moving thousands of miles across the country to bigger and better things. Finally, my advisor Dr. Mohan Trivedi has stood by me these last seven years as my greatest advocate and champion. I can finally appreciate your decision to let me find my own interest and motivation. Your enthusiasm and expectation of excellence is infectious. The moment I went to sleep then woke up thinking about research, I got it.

I will be forever grateful for all my San Diego, Berkeley, and San Jose friends. You are the ones who help keep me grounded and maintain my sanity. In particular, I am indebted to the House members. Through Sorrento Valley to Brassica, you have been there for me all these years, providing both technical and emotional support. I rely on you to keep me connected, and this is one group I am sure I will not forget.

Most importantly, I want to thank my family for patiently waiting for me to finish. It might have seemed like I was just enjoying the sun and the beach but we finally have a doctor in the family. This dissertation is for my parents who have always remained proud of all my decision and allowed me the freedom to pursue academics without hesitation. I can never repay all the love and support you have shown over the years. I hope you know this could never have happened without you.

Chapter 2 is in part a reprint of material that appears in the IEEE Transactions on Circuits and Systems for Video Technology, 2008, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 3 is in part a reprint of material that appears in the IEEE Transactions on Intelligent Transportation Systems, 2008, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 4 is in part a reprint of material in submission to the IEEE Transactions on

Pattern Analysis and Machine Intelligence, 2010, and material that appears in the Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2009, the Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), 2008, and the Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), 2008, all by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of these papers.

Chapter 5 is in part a reprint of material that appears in the IEEE Transactions on Intelligent Transportation Systems, 2008, the Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2007, the Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), 2006, the Proceedings of ITSC, 2006, all by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 6 is in part a reprint of material that will appear in the IEEE Intelligent Systems Magazine, 2010, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of these papers.

Chapter 7 is in part a reprint of material that appears in the Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2009, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Appendix B is a reprint of material that appears in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, by Brendan Morris and Mohan Trivedi. The dissertation author was the primary investigator and author of these papers.

VITA

2002	B. S. in Electrical Engineering and Computer Science with *honors*, University of California, Berkeley
2006	M. S. in Electrical and Computer Engineering, University of California, San Diego
2010	Ph. D. in Electrical and Computer Engineering, University of California, San Diego

PUBLICATIONS

B. Morris and M. Trivedi, "Robust classification and tracking of vehicles in traffic video streams," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Toronto, Canada, Sep. 2006, pp. 1078–1083.

B. Morris and M. Trivedi, "Improved vehicle classification in long traffic video by cooperating tracker and classifier modules," in *Proc. IEEE Inter. Conf. on Advanced Video and Signal based Surveillance*, Sydney, Australia, Nov. 2006, pp. 9–14.

B. Morris and M. Trivedi, "Real-time video based highway traffic measurement and performance monitoring," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Seattle, Washington, Sep. 2007, pp. 59–64.

B. T. Morris and M. M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1114–1127, Aug. 2008, Special Issue on Video Surveillance.

B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 425–437, Sep. 2008.

B. Morris and M. Trivedi, "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis," in *Proc. IEEE Inter. Conf. on Advanced Video and Signal based Surveillance*, Santa Fe, New Mexico, Sep. 2008, pp. 154–161.

B. Morris and M. Trivedi, "An adaptive scene description for activity analysis in surveillance video," in *Proc. IEEE Inter. Conf. on Pattern Recog.*, Tampa, Florida, Dec. 2008.

B. Morris and M. Trivedi, "Learning trajectory patterns by clustering:experimental studies and comparative evaluation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami, Florida, jun 2009, pp. 312–319.

B. T. Morris and M. M. Trivedi, "Contextual activity visualization from long-term video observations," *IEEE Intell. Syst.*, to be published.

B. T. Morris and M. M. Trivedi, "Unsupervised multilevel trajectory learning for long-term adaptive activity understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010, in review.

ABSTRACT OF THE DISSERTATION

Understanding Activity from Trajectory Patterns

by

Brendan Tran Morris

Doctor of Philosophy in Electrical and Computer Engineering

University of California, San Diego, 2010

Professor Mohan Trivedi, Chair

A fundamental goal of Computer Vision is to provide scene understanding and situational awareness. In order to deliver on this promise, traditional monitoring systems were designed for specific environmental situations, such as a specific time, place, or activity scenario. A well versed expert defines the events of interest by hand for the particular application. While effective, these techniques do not scale well, they typically have poor generalization, are inflexible to behavioral changes, and the analysis rules may not reflect the true nature of the scene but a priori expectations. The conventional methods to understand activities must be scaled to match growing need. Society's rapid acceptance of video use in a wide variety of locations and applications has promoted the deployment of large camera networks. These networks monitor complex scenes and deliver volumes of video data that can not be digested without automated assistance.

This dissertation investigates unsupervised activity understanding by analyzing patterns of motion trajectories. A practical approach is introduced and carefully developed to overcome the difficulties with trajectory learning, namely the definition of a simple activity model that can be robustly inferred from crude measurements, the automatic determination of the number of typical activities in a scene, and methods to observe dynamic scenes over long periods. The activity analysis framework is able to process and analyze activity, providing activity characterization, prediction, and abnormality detection, in real-time for real world utility.

The efficacy of the trajectory learning framework is demonstrated in three distinct arenas.

Highway traffic is monitored using a single camera with the VECTOR system, multiple sensors are integrated and combined in a unified space with CANVAS, and driving maneuvers analyzed from within a moving automobile. This extension of the trajectory learning paradigm to a broad range of (untouched) application spaces further highlights the dissertation contributions. Finally, extensive performance evaluation and characterization is conducted to provide a missing benchmark for the field.

Chapter 1

Introduction

A fundamental goal of Computer Vision is to provide understanding of activities. Traditionally, monitoring systems were designed for specific environmental situations, such as a specific time, place, or activity scenario. The knowledge structures used for analysis were designed by hand by a well versed expert who defined the events of interest for the particular application. While effective, it is prohibitive to completely specify everything of interest. The rules may be difficult to develop and have poor generalization and be inflexible to behavioral changes. Finally, as applications become more complex, the design of rule-based systems might not be sufficient because they may not provide a realistic portrayal of behavior. The activity definitions might not actually reflect the true nature of behavior but a priori expectations.

The conventional methods to understand activities must be scaled to fit today's growing needs. Society is rapidly accepting the use of cameras in a wide variety of locations and applications. The dramatic decrease in cost for quality video equipment coupled with the ease of transmitting and storing video data has led to widespread use of vision based analysis systems. Cameras are in continuous use all around, along highways to monitor traffic, for security of airports and other public places, and even in our homes. Large networks of cameras monitoring complex scenes stream loads of data on a daily basis. This data must be analyzed quickly for real comprehension. The sheer volume of this video data impedes accurate human analysis. Monitoring for interesting events, which rarely occur, is a tedious and tiring job which necessitates computer vision solutions to help automate the process and assist operators.

To this end, intelligent cameras are desired. Cameras that through observation, can automatically build an internal description of a the scene. The camera's memory (historical observations) allows the it to learn and predict what is interesting or important (activities and predict future behaviors). And for long term use, the cameras should be adaptable to changes, whether environmental or behavioral.

Although automatically understanding activity is a very challenging problem with a large

space of possible activities each defined differently for different scenes, for different objects, and by the specific monitoring requirements, learning from trajectory patterns provides a method imbue intelligence to cameras in an unsupervised manner. My observing motion patterns over time it is possible to extract the underlying activity processes that generate object trajectories. By relying on experience, as opposed to complex models, activities can be generally described. Once activities are learned then the behaviors of every scene agent can be understood for description and summarization, prediction, and in order to detect unusual occurrences.

1.1 Challenges

There are many challenges for real world implementation of trajectory based activity analysis. The main difficulties encountered in activity analysis from trajectory patterns are the following:

- Difficulties with detection and tracking lead to broken and incomplete trajectories.
- There is wide variation among different scenes due to types of objects observed and activity complexity.
- Understanding over long time periods presents non-stationary activity processes.
- For maximum utility, activity analysis must occur in real-time on any camera setup with little supervision and minimal constraints.

These challenges provide opportunities for engineering solutions for practical application.

1.2 Contributions

This dissertation introduces an analysis framework that enables the understanding of activity from observation of trajectory patterns. A three stage hierarchical modeling process in developed to determine the typical activities in an unsupervised fashion for a novel camera scene without a priori knowledge. Using the learned activity models, the activity of each detected object can be classified, predicted, or deemed unusual during live operation in real-time. The list of main contributions in trajectory based activity analysis are as follows:

- Introduction of a three stage hierarchical learning process to model typical and recurrent activity.
- A method to automatically estimate the number of activities in a scene.
- A process to adapt activity models to better fit data for long term usage.

- The activity analysis occurs in real-time on live video for immediate notification.
- Extensive evaluation, which is missing from the literature, of the activity analysis is provided.
- The learning framework's generality is demonstrated on a wide variety of scenes, differing in types of objects observed, structure, and interactions.

The dissertation also addresses questions of learning robustness in the presence of noise and incomplete data, provides critical comparison of similarity metrics for trajectories, the development of evaluation metrics to characterize activity analysis performance, and examines trajectory relevance for traffic monitoring, surveillance, and safety.

Chapter 2

Background

This chapter provides a definition of learning trajectory patterns for activity analysis and defines the scope of the dissertation. A summary of previous work is presented followed by a short explanation of the the trajectory learning approach taken in this dissertation.

2.1 The Trajectory Learning Problem

Automatically understanding activity is a very challenging problem. The space of possible activities is huge and defined differently for different scenes, for different objects, and by the specific monitoring requirements of the end user. With such a rich and diverse activity space it is difficult to imagine general procedures capable of working over a wide range of scenarios.

In order to be tractable, this dissertation limits itself to far-field applications, as would be typically encountered in surveillance or other monitoring situations. In these scenarios, it is quite difficult to extract elaborate descriptions of the objects or actions. But, the cue of interest, which can be reliably obtained, in these cases is motion. In addition, typical motion is not completely random but structured and repetitive. Since object motion is continuous, a trajectory (a sequence summarizing the spatio-temporal characteristics of a moving object) is assumed to have been generated as a realization from a set of hidden random processes that signify the typical activities that are present.

The diagram in Fig. 2.1 presents the basic steps for activity understanding from trajectory patterns. Objects are initially observed by a camera and the motion of each individual is tracked. The resulting trajectories are the only feature that is used to model the scene and infer the underlying activities. By learning a scene description low-level situational awareness is obtained automatically. The awareness goal is to understand and characterize the behavior of every object in the scene. In order to understand an object, the activity analysis engine must be able to characterize current behavior, predict future behavior, and detect abnormalities (things

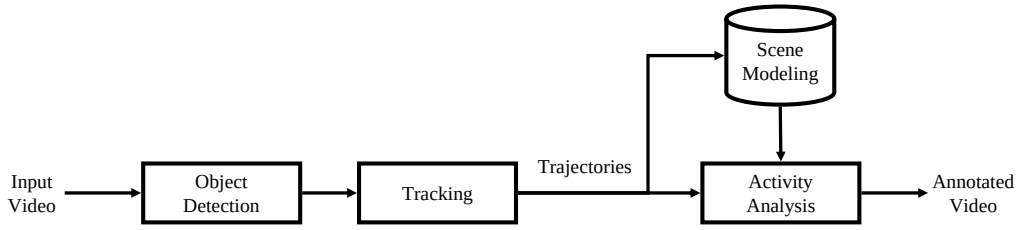


Figure 2.1: The basic block diagram for trajectory based surveillance. The front-end system consists of robust object detection and tracking to produce the trajectories. The tracks are used to build a scene model to describe the surveillance situation and perform activity analysis.

that are unusual) in an unsupervised fashion. For practical application, the analysis must happen in real-time to ensure prompt notification and adequate response and must be adaptable to changes for long term usage.

There are a number of components to a trajectory learning and activity analysis system and not all can be touched on in this dissertation. For further discussion, the review by Morris and Trivedi [90] presents a wide range of applications and highlights the challenges and common approaches to trajectory learning an modeling.

2.2 Existing Trajectory Learning Approaches

Automatic scene understanding got it's start in the mid 1990s when neural networks gained popularity in the learning community. Using neural networks to find complex relationships between inputs and outputs as well as pattern discovery gave birth to the field of trajectory learning. Unlike earlier work that used explicit models to describe a surveillance scene, trajectory learning avoids assigning any prior scene knowledge and instead builds activity models based on observable events in an unsupervised manner. Utilizing a data-driven modeling approach allows more flexible deployment because it does not require a priori scene knowledge or user intervention.

2.2.1 Flow Quantization

Flow quantization learning techniques sequentially fed in flow vectors to the learning machine. The sequence of vectors caused a unique activation pattern to explain an activity based on trajectory history.

The earliest work (summarized in Table 2.1) utilized neural networks to perform vector quantization and to learn the typical motion patterns in a surveillance scene. Pioneering work by Johnson and Hogg [55] described motion using a flow vector, summarizing position and velocity during tracking. The flow vector was feed to a large dual-layer self organizing map (SOM) connected by leaky neurons which jointly quantized the flow vectors and learned the typical

Table 2.1: Exemplary Flow Quantization Activity Modeling Techniques

Publication	Path Usage	Comments
Johnson 1995 [55]	Classification	Paths learned using a dual-layer SOM connected by leaky neurons. The leaky neurons contains the memory of a path while output nodes merely indicate a path cluster. The trajectories were linearly resampled for consistent point density during training. The flow vector input f_t consisted of both spatial and dynamic information.
Sumpter 2000 [111]	Prediction	The leaky SOM architecture was extended for prediction by incorporating feedback from the output nodes to the leaky neurons which enabled analysis of incomplete trajectories.
Owens 2000 [92]	Abnormality	A self-organizing feature map was used to quantize flow vectors. The two-dimensional structure created neighborhoods of similar prototype vectors where allowed for cluster visualization. Flow points were evaluated in real-time, without needed a complete trajectory, to distinguish abnormal behavior.

motion patterns. The motion patterns relied on the short term memory of flow activations which weakly implied the sequencing of trajectory points. Sumpter and Bulpitt [111] extended this work for motion prediction by incorporating a feedback loop between the output layer and the leaky neurons. The prediction capability enabled real-time analysis by handling incomplete trajectories. Owens and Hunter [92] used a self-organizing feature map (SOFM) to detect abnormalities. A two-dimensional output layer performed vector quantization where learning updates occurred in small neighborhoods around a winning neuron creating a topological map useful for cluster visualization. In this framework, learning proceeded in an online fashion on individual flow vectors rather than waiting for a full trajectory. Suspicious points were discovered if the match between a flow vector, which considered position, velocity, and acceleration, had a large Euclidean distance with the winning neuron exemplar vector.

The problem with these neural network techniques was the complicated training process. The networks required large amounts of data, a number of sensitive parameters and weights needed to be fine-tuned, and it took considerable amount of time [51].

2.2.2 Trajectory Clustering

Trajectory clustering techniques did not use individual flow vectors as learning inputs but instead used a full trajectory. This explicitly modeled the sequential nature of trajectory samples and improved learning speed.

In 2004, Hu *et al.* [51] greatly improved the learning speed by introducing a new SOFM where all the output neurons were arranged into a single layer which required only a single training phase rather than the two needed for the different SOM layers previously. This new structure used a full (fixed-length) trajectory as the network input with each input node connected to all the output nodes. The weights connecting the input to an output node defined a prototype trajectory (activity). Soon it was realized that this network architecture was performing clustering of trajectories and it paved the way for the trajectory clustering based activity modeling techniques currently in use (see Table 2.2). Work by Porikli [97] assumed a trajectory was generated by a hidden Markov model (HMM). A HMM was fit to every trajectory in the training set and a form of spectral clustering was performed in the HMM parameter space. This provided HMM models for the typical clusters that were used to classify new trajectories. The quality of the HMM models was quantified using a validity score. Also in 2004, Junejo *et al.* [56] developed a hierarchical activity representation to account for spatial, velocity, and curvature features. Abnormal trajectories were detected by testing each stage of the hierarchy which gave an explanation of why something was considered abnormal. This work used a min-cut graph algorithm, with trajectories as nodes and edges corresponding to the Hausdorff distance between nodes, to cluster trajectories. In 2005, Bashir [7] broke trajectories into smaller sub-tracks based on curvature. The sub-tracks were projected onto a PCA space and clustered using k-means. Each sub-track represented a state in a Markov model which allowed for prediction based on transitions. Hu *et al.* continued there trajectory learning work through 2007 [48, 50] by extending trajectory analysis to include classification, prediction, and anomaly detection. Activities were learned in a two-stage clustering process. In the first stage, only the spatial information was used while the second stage further refined the clusters by incorporating velocity. Each cluster was modeled as a chain of Gaussian distributions which enabled probabilistic explanations of activity. In addition, a tightness and separation criteria was used in an iterative learning framework to determine the number of clusters needed to describe the scene activity.

The previous approaches all utilized a batch learning approach. Before clustering, a database of trajectories needed to be collected by observing the scene over a sufficient amount of time. This collection process required a setup time before any analysis could be performed and limited activity models to only those that had been previously observed in training. In contrast, online learning methods allow for dynamic activity definitions which evolve as new data is observed. In 2005, Makris and Ellis [76] presented the first online trajectory learning method which was used to classify trajectories. Each trajectory pattern was described based on a spatial

Table 2.2: Exemplary Trajectory Clustering Activity Modeling Techniques

Publication	Path Usage	Comments
Hu 2004 [51]	Classification, Prediction, Abnormality	Neural network learning speed was greatly increased by using a full trajectory for input which required only a single training phase. A set linked output nodes corresponded to a path prototype which was characterized for prediction and abnormality detection.
Porikli 2004 [97]	Classification	A trajectory was modeled by an HMM whose parameters were used as a feature vector for clustering. Cluster centers were formed using a mutual fitness similarity measure. The number of clusters was estimated using a validity score.
Junejo 2004 [56]	Classification, Abnormality	Path clusters were formed using the min-cuts graph algorithm with trajectory nodes and associated Hausdorff distance as the edges. The paths consisted of the center plus a spatial envelope and was augmented with a velocity and curvature profile for abnormality characterization.
Makris 2005 [76]	Classification	Paths with a center and spatial envelope was learned in an online fashion for adaptability to changing conditions.
Bashir 2005 [7]	Classification, Prediction	Trajectories were broken into atomic sub-trajectories based on curvature. The sub-tracks were projected into a PCA space for k-means clustering. The learned sub-paths formed the states of a Markov model allowing prediction of behavior from sub-path transitions.
Piciarelli 2006 [95]	Classification, Prediction	Paths learned in an online fashion allowing paths to split into a tree-like structure where nodes were common among the children. Trajectories were compared without normalization using a novel time-windowed Euclidean distance. The tree representation allowed path prediction by estimating the transitions probabilities between nodes.
Hu 2007 [50]	Classification, Prediction, Abnormality	Paths are learned in a two-stage spectral clustering procedure. The first stage used only spatial information and verified clusters using the TSC. The second clustering stage included velocity and the resulting clusters were modeled by a chain of Gaussian probability distributions.

envelope. New trajectories that fell within the envelope modified the activity definition while new envelopes were spawned for ill-fitting trajectories. In a similar vein, Piciarelli and Foresti [95] developed an online trajectory clustering method in 2006. Clusters were represented in a tree-like structure with node sharing. Trajectories were compared with a novel time-windowed Euclidean distance designed to gauge the similarity between a trajectory and a cluster (which has a spatial extent). The tree representation allowed for prediction by counting the transitions between nodes.

The trajectory clustering approaches to activity analysis are popular and varied. A problem with these approaches is the lack of comparison to explain the relative strengths and weaknesses of the approaches.

2.2.3 Co-Occurrence

Although trajectory clustering has been the most popular automatic activity analysis approach, recently there has been a small push to avoid tracking altogether. Instead of explicitly tracking objects, motion units are examined to find structure. These techniques follow a bag-of-words approach and try to find words that co-occur with regularity. The set of co-occurring motion words represent an activity. Exemplary co-occurrence techniques are summarized in Table 2.3.

Stuaffer and Grimson [110] learned activities in a hierarchical fashion by building up a co-occurrence of codebook flows in 2000. They disregarded track ordering and instead viewed a trajectory as a set of points that tend to occur together. Given a newly tracked object, they could determine the activity just by vector quantization of the the flow. Zhong *et al.* created a codebook of visual words based on motion histograms to detect abnormal activity in 2004. The words describing an activity were learned through bipartite graph co-clustering. In 2008, Xiang and Gong [131] examined abnormality detection as well by providing incrementally adaptive models. Trajectory points were clustered using Gaussian mixture modeling into a codebook. Activities were described as a multiple observation HMM and found through spectral clustering. The resulting clusters provided a composite behavior model of a video clip. Current work by Wang [126, 125] adapts word-document clustering techniques for visual monitoring. First a codebook of typical flow vectors is defined from the training set to provide the vocabulary. The flow words are grouped into semantic regions corresponding to segments of activity. The generative models used for clustering provide an explanation for how activities are formed and abnormalities are probabilistically defined.

Co-occurrence techniques typically can handle lots of data, but they may require large amounts of data to accurately learn activity. A complaint is that the intuitive notion of temporal sequencing is lost when examining a bag-of-words representation which may ignore subtle differences in behavior with this aggregate view. Another shortcoming of the techniques that do not

Table 2.3: Exemplary Co-Occurrence Activity Modeling Techniques

Publication	Path Usage	Comments
Stauffer 2000 [110]	Classification	Each trajectory is quantized into a small set of acceptable flow vectors describing position and velocity and a codebook co-occurrence matrix is formed. The complete training co-occurrence matrix is hierarchically decomposed in binary tree-like fashion describing a pmf of codewords. The order of flow codewords was lost in the procedure.
Zhong 2004 [138]	Classification, Abnormality	A codebook of visual words is created from $m \times m$ motion histograms. The set of words in an activity is learned through bipartite graph co-clustering. The resulting eigen decomposition produces an embedding mapping into the activity space.
Xiang 2008 [132]	Classification, Abnormality	Trajectory points are clustered using GMM into a set of activities. The posterior probability of all activities is modeled with a multiple observation HMM (MOHMM) that allows computation of a similarity matrix for spectral clustering. The resulting clusters provides a composite behavior model for a video clip.
Wang 2008 [125]	Classification, Abnormality	Word-document clustering techniques are adapted for visual monitoring. A codebook of flow vectors is defined to describe the motion vocabulary and clustered into activity topics. The distribution of topics succinctly describes the scene activities.

use tracking is that they are not able to provide the activity of individuals but a more holistic scene description.

2.3 Proposed Approach

It has been shown that by analyzing trajectory patterns the activity structure of a scene can be automatically inferred. Further, these extracted patterns can be used to explain current behavior, predict future activity, and detect abnormalities. Unfortunately, rigorous performance characterization is lacking from the trajectory learning literature. Without this experimental validation, the quality and utility of these unsupervised activity analysis techniques in real-world deployment is unknown. This dissertation attacks this short coming by developing a practical trajectory learning framework that is able to model activities, provides a comparison of trajectory similarity measures, automatically determine the number of activities in a scene, and quantifies the real-time performance of activity analysis. The generality of the framework is highlighted by a number of experiments on very different scenes.

2.4 Acknowledgements

Chapter 2 is in part a reprint of material that appears in the IEEE Transactions on Circuits and Systems for Video Technology, 2008, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Collecting Trajectories

Before understanding activity, it is necessary to first determine how to describe an activity. In this work, motion is the important cue for activity. To fully describe the behaviors in a scene the motion of each object should be accurately recorded. In this preliminary chapter we describe how object motion is extracted by visual tracking. This low-level process results in the trajectory features necessary for inferring activity. This preliminary chapter presents a short overview of the visual tracker utilized throughout the dissertation.

3.1 Motivation

Object tracking is one of the fundamental tasks within the field of computer vision. The widespread use cameras has greatly increased the need for automated video analysis and interest in object tracking. As noted in the survey on object tracking by Yilmaz *et al.* [134], the use of object tracking is prevalent in a variety of tasks such as

- motion-based recognition - human identification based on gait or automatic object detection
- automated surveillance - monitoring a scene to detect suspicious activities or unlikely events
- video indexing - automatic annotation and retrieval of the videos in multimedia databases
- human-computer interaction - gesture recognition or eye gaze tracking for data input to computers
- traffic monitoring - real-time gathering of traffic statistics to direct traffic flow
- vehicle navigation - video-based path planning and obstacle avoidance capabilities.

Tracking is defined as the problem of estimating the trajectory of an object as it moves around a scene. To this end, a tracker must assign consistent labels to objects for the times they

are in view. The tracker generates a trajectory by detecting an object, locating its position in every video frame, and making associations between frames.

In the monitoring situation, there is a far-field view of objects which result in low resolution and discrimination. In these situations, it is difficult to accurately extract complex features. Only coarse motion attributes, *e.g.* location, can be robustly obtained. Since motion is the major cue for activity, activities can be defined by motion profiles which indicate how an object moves through a scene. By simultaneously tracking many objects, the sate of each individual object, *e.g.* position, velocity, acceleration, size, etc., can be continually maintained and updated to summarize activity.

Through the visual tracking front end, extracted trajectories are used as output features for higher level activity understanding and situational awareness.

3.1.1 Questions to Address

In order to analyze activity, it is necessary to simultaneously track many objects. Key issues to address are how to make the tracker flexible enough to work in a variety of monitoring environments and situations, be adaptable to changing conditions, and perform in real-time to enable live activity analysis.

3.1.2 Chapter Summary

This chapter presents a real-time background subtraction based visual tracker capable of generating the raw trajectories features for activity analysis. The tracker uses an adaptive background subtraction scheme to detect moving objects and tracks multiple targets. The tracker has been tested in a variety of locations and over long periods of time. But, the tracker performance is limited in dense scenes with high occlusion rates or in severe lighting conditions.

3.2 Background Modeling for Motion Detection

Foreground pixels belonging to moving objects are quickly determined by using an adaptive background subtraction scheme. Each background pixel is modeled as a single Gaussian process [129], composed of two parameters; μ, a time averaged pixel intensity, and σ, the standard deviation of pixel intensity. The Gaussian parameters at the current time t are adapted as

$$\mu_t = (1-\alpha)\mu_{t-1} + \alpha I_t \tag{3.1}$$

$$\sigma_t^2 = (1-\beta)\sigma_{t-1}^2 + \beta(I_t - \mu_t)^2. \tag{3.2}$$

The update parameters $\alpha, \beta \in [0,1]$ control how quickly the background distribution is able to change as each new video frame, I_t, is received. Foreground pixels, which indicate change, do

not fit the video statistics.

The foreground image is obtained by thresholding the background difference to find objects not part of the background

$$I_{foreground} = (I_t - \mu_t) > B(\sigma_t + \sigma_0). \tag{3.3}$$

Here σ_0 is a small constant to suppress noise associated with low variance scenes typically encountered during low light and shadowed situations and B is the deviation threshold. The threshold B determines how different the current intensity of a pixel needs to be from the background in order to be a likely object. Since threshold B is difficult to fix as a single value in outdoor applications with varying lighting conditions, it is adaptively defined for each pixel by its local neighborhood, $N \times N$, intensity,

$$B_{ij} = \min\left\{\left(\frac{B_{max} - B_{min}}{I_m + I_\sigma}\right) I_N + B_{min}, B_{max}\right\}. \tag{3.4}$$

Equation (3.4) sets up a threshold that adapts to local lighting intensity, I_N, by comparison to the mean image intensity and standard deviation, I_m and I_σ respectively. The values of B_{max} and B_{min} indicate the maximal and minimal deviation necessary for detection. The threshold B adjusts to local lighting conditions. When lighting is lower, a smaller threshold is used because object and background intensity are less differentiable. By defining the threshold for each individual pixel ij, the foreground extraction process is able to handle situations with varying lighting as would be the case if a large shadow from a building covered part of the camera view.

The foreground is further processed to fill in any holes using morphological operations. Each blob is then labeled by connected component analysis to form detections and simple morphological measurements are taken, $m_t = \{$area, breadth, compactness, elongation, perimeter, convex hull perimeter, bounding box, best fit ellipse parameters, roughness, centroid, $M_{10}, M_{01}, M_{20}, M_{02}\}$ [108], to compactly represent the object shape appearance.

3.3 Multi-Object Tracking

After foreground regions have been located, the detections can be tracked. By tracking objects, a sequence of object states

$$S_T = \{s_1, \ldots, s_t, \ldots, s_T\} \tag{3.5}$$

is used to produce a trajectory which summarizes an object's activity in the camera view. During tracking, each object is modeled as a feature augmented point. The state of an object

$$s_t = \begin{bmatrix} f_t \\ m_t \end{bmatrix}, \qquad f_t = \begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \end{bmatrix}, \qquad m_t = \begin{bmatrix} \eta_0 \\ \vdots \\ \eta_{15} \end{bmatrix}, \tag{3.6}$$

consists of the position xy, velocity uv, and shape appearance m at a given time t. It is assumed that objects are rigid bodies that move along a smooth path with constant velocity. These assumptions lead to the following state equations

$$s_{t+1} = \left[\begin{array}{c|c} \mathbf{F} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M} \end{array}\right] s_t. \tag{3.7}$$

Where $\mathbf{F}$ explains the dynamical changes of f_t and $\mathbf{M}$ characterizes the appearance change of m_t.

The identity of each individual object is maintained through the tracking procedure. New foreground detections are associated to existing tracks through nearest global neighbor matching. This track update occurs only when a detected object matches both a dynamics and appearance model which are treated separately in practice. Object dynamics are modeled by employing a Kalman filter [62] for optimal one step prediction and appearance is defined by a set of shape features.

3.3.1 Dynamics Modeling

An object's dynamics characterizes how it moves within the camera field of view

$$f_{t+1} = \mathbf{F} f_t. \tag{3.8}$$

It is assumed that an object (the centroid of a detection region) follows a regular motion pattern. Therefore, the future state of an object can be optimally predicted using a Kalman filter with a constant velocity model. The state update equation becomes

$$f_{t+1} = \mathbf{F} f_t + w_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \end{bmatrix} + w_t. \tag{3.9}$$

In this model, Δt corresponds to the update rate which is equal to the video frame rate and w_t is a noise term. The model uses the previous velocity to determine the new position. It is important that the model also allows for non-constant velocity because of the noise term as it is unlikely that object speed will remain completely constant.

The observation model directly relates the detected region measurement $\tilde{f}$ to object state

$$\tilde{f}_t = f_t + e_t = [\tilde{x}_t, \tilde{y}_t, \tilde{x}_t - x_t, \tilde{y}_t - y_t]^T. \tag{3.10}$$

The $\tilde{}$ notation denotes the dynamics of a detected foreground blob and e_t is a Gaussian noise term associated with measurement inaccuracy. New tracks are initialized by instantiating a Kalman filter with a velocity obtained by a nearest neighbor match $[\tilde{u}_1, \tilde{v}_1]^T = [\tilde{x}_2 - \tilde{x}_1, \tilde{y}_2 - \tilde{y}_1]^T$.

A Kalman filter for a track is updated using the matching blob detection. A blob matches if the Kalman track prediction $\hat{f}_{t+1}$ and a measurement are within a small distance

$$\hat{f}_{t+1} - \tilde{f}_t < \epsilon. \tag{3.11}$$

Detections that are deemed consistent with the dynamics model are fully matched after taking into account appearance.

3.3.2 Appearance Modeling

In addition to the dynamic model, a track has an associated appearance model. This appearance model is used to resolve matching ambiguities. These ambiguities mainly arise because of high object density or occlusion. This model guarantees appearance along a trajectory does not change drastically, by enforcing consistency between frames.

The similarity at time t between a detection, $\tilde{m}_t$, and a track, m_t is given by

$$S_m = [(m_t - \tilde{m}_t)^T \Sigma^{-1} (m_t - \tilde{m}_t)]^{-1} > T_m, \tag{3.12}$$

where Σ is a diagonal matrix with entries equal to the measurement variance learned during training and T_m is a match threshold. When in a crowded scene, the best match is the detection that fits the motion model and has the most similar appearance to the track. This constraint implicitly manages occlusion when objects either merge or split by instantiating new tracks. The track appearance is adaptively updated upon consistent match with a detection $\tilde{m}_t$

$$m_t = (1 - \gamma)\tilde{m}_t + \gamma m_{t-1}, \tag{3.13}$$

given the track and detection measurements at the current time. The $\gamma \in [0, 1]$ controls how quickly the objects appearance may change during tracking. By adjusting the shape appearance model, objects can be tracked through larger variations such as when a vehicle makes a turn at an intersection.

3.4 Tracking Difficulties

There are a number of common difficulties when tracking objects. Yilmaz *et al.* [134] summarizes them as

- loss of information caused by projection of the 3D world on a 2D image,
- noise in images,
- complex object motion,
- nonrigid or articulated nature of objects,

- partial and full object occlusions,
- complex object shapes,
- scene illumination changes, and
- real-time processing requirements.

The main difficulties encountered in this work stem from occlusion and illumination. Monitoring outdoor spaces over long periods of time present changing illumination which can cause strong cast shadows. These shadows can disrupt object detection by both making an object seem larger and causing two nearby objects to be detected as a single one as form of occlusion. These occlusion instances create broken and trajectories which ultimately results in an incomplete description of an activity.

3.4.1 Occlusion

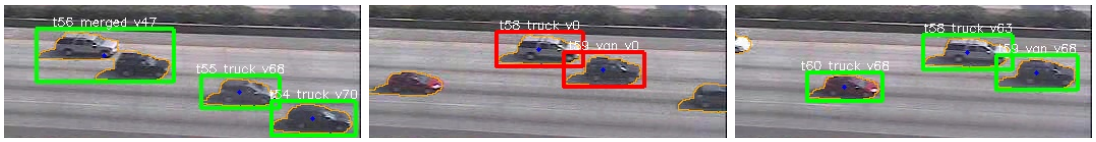
Vehicle and human tracking is a well understood problem and relatively easy to solve when there is only one object. Unfortunately as the number of objects increases the complexity increases as well. The major difficulty when tracking many objects is occlusion which causes incomplete or incorrect trajectories.

Inter object occlusion occurs with two different objects are in close proximity and one blocks the other from the camera view. Occlusions of short duration are gracefully handled by continuing to predict the object location until it reappears. In Fig. 3.1 two vehicles get close enough to be merged into a single detection. Since neither track has a matching detection because of the the differing appearance, a new track is instantiated. When the vehicles separate the tracker is able to reconnect the detections to the initial tracks. Longer occlusion can cause two types of tracking errors, merging and splitting of objects [57]. As shown in Fig. 3.2 a merge occurs when two separate objects are detected as one while splits arise when a two objects initially merged separate into individuals. A recovery pass over the completed tracks applying heuristic rules [79] or using spatio-temporal cues [58] can repair occluded tracks by accounting for motion consistency.

While track based occlusion recovery is possible in many situations, it will prove difficult when there is high object density which causes a high rate of occlusion and occlusion pairs. External cues are needed to indicate merging of multiple objects. The temporal constraints need to augmented by appearance modeling [59, 42, 93]. By improving object detection with a mean shift color clustering algorithm [25], shadow removal [98], and inclusion of temporal occlusion modeling will improve tracker performance.

Figure 3.1: Occlusion causes initialization of new trajectories. When the occlusion is of short duration the tracking module is able to recover the correct trajectory labels.

(a)



(b)

Figure 3.2: Merge and split occlusion types which can be recovered using temporal reasoning. (a) Merged detection (b) Split detection

3.4.2 Shadows

Shadows are a major concern for background subtraction based object detection. Outdoor monitoring over long periods of time guarantees cast shadows, either from surrounding environment structures such as buildings or from the moving objects themselves, will be present. While shadows do corrupt the quality of localization, object appearance, and cause object merges (shadows connect objects) this work does not specifically handle shadows. The assumption is that the trajectory patterns and inherent dynamics are only loosely correlated with shadow detection errors. A large literature dealing with shadow estimation and suppression techniques exists for the interested reader. [46, 98, 31]

3.5 Concluding Remarks

Although the visual tracker developed in this work is fairly simple, it has proven effective in a number of different environments and situations. Even while using a simple background model and limited appearance constraints, without explicit occlusion handling or shadow suppression, sufficient trajectories have been extracted for surveillance-type camera setups. The tracking operates in real-time which enables analysis of live video streams, detects objects accurately enough for recognition of many object types, and produces consistent trajectories suitable for activity analysis.

3.6 Acknowledgements

Chapter 3 is in part a reprint of material that appears in the IEEE Transactions on Intelligent Transportation Systems, 2008, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Learning Trajectory Patterns

Using background modeling and subtraction, moving objects are extracted and their identities maintained and tracked over time. Observable motion does not occur randomly but has structure. Repeated activity generate patterns of trajectories which can be examined over time for automatic learning of new scenes without manual supervision.

4.1 Motivation

The dramatic decrease in cost for quality video equipment coupled with the ease of transmitting and storing video data has led to widespread use of vision based analysis systems. Cameras are in continuous use all around, along highways to monitor traffic, for security of airports and other public places, and even in our homes. Methods to manage these huge volumes of video data is necessary. It becomes almost an impossible task to continually monitor these video sources manually forcing researchers to develop methods to recognize certain events and activities of interest. These methods provide a means to compress video data into a more manageable form as well as give annotations that can be used for search indexing. While understanding video in general seems a daunting task, it can be quite successful when analyzing a particular setting. The review of dynamic scenes by Buxton [16] presents a wide variety of techniques for visually understanding human and vehicle actions that are effective but usually require domain knowledge.

Researchers have long been interested in understanding human behavior. Early vision work examined activity sequences through template matching or state space approaches [2]. More recently, nine actions were examined with a HMM formulation based on a sequence of 3D voxel body configurations [53]. Volumetric space-time shapes induced by human actions have been constructed to compare with an activity database [40]. Current recognition techniques focus not just on full body gestures but might concentrate on specific body parts such as hand and arms or head and face [82]. These activity techniques rely on a local configuration of body parts which is

not well suited for far field applications because they require high resolution to distinguish parts.

In contrast, surveillance monitoring has focused on more coarse motion based activity with almost no prior object model. Rather than specifying individual parts, the center of mass motion is considered of either rigid or deformable objects (*e.g.* vehicles and humans). Early on, simple activity recognition schemes defined interesting zones and trip wires that could signal an alert when crossed. These alerts have been used to focus attention of high resolution PTZ image capture [115], accumulate traffic counts [67] and origin-destination information [80], and detect loitering individuals [12, 133]. As more video surveillance sites were erected, automatic methods to detect interesting activities increased with a major research thrust in the area of unusual or abnormal behavior detection. By observing and collecting trajectories, over time a model for scene is developed characterizing typical activity [27] from data rather than by manual specification. Trajectories that are not well explained by the typical models were considered abnormal [92, 52, 48, 56]. Others have tried to avoid difficulties with explicitly tracking individual objects, due to scene clutter and occlusion, and relied only on inter-frame motion. Drawing inspiration from document clustering research, activities (attributed to the entire frame or clip) are viewed as a collection of motion words [138, 130, 126, 125, 127].

Through motion analysis, video cameras are able to learn how and what to monitor. By adding intelligence to cameras, larger networks can be accurately managed and monitored over longer periods and wider areas with minimal supervision.

4.1.1 Questions to Address

Automatic scene understanding and activity analysis is a challenging problem which poses many difficult questions:

- How should activities be learned and represented?
- How can the number of activities in a novel scene be determined without a priori knowledge?
- How can the activity models be used over long periods of time when activities are dynamic?
- What types of analysis can be performed using unsupervised models?

4.1.2 Chapter Summary

In this chapter a practical approach toward understanding activities is presented. A multi-level learning framework is introduced to automatically learn the typical activities in a scene without requiring specific domain knowledge. Without supervision, trajectory patterns are extracted and used to form probabilistic models of activity. The models enable online analysis of live video and provide contextual awareness. The practical approach to learning trajectory patterns illuminates several key questions:

- In applications that provide low resolution information, motion is a reliable activity cue. An activity can be modeled as an ordered sequence of actions with each action denoted by location and dynamics.
- Activities can be robustly extracted by clustering observed trajectories.
- Although the number of activities in a scene is not known, it can be estimated by examining the quality of trajectory clusters. An accurate activity count is obtained by over-clustering the trajectories and merging similar clusters.
- When observing scenes over long time periods, the activities present will change. Activities can be adjusted online with new trajectories during live analysis to better match current observations. New activities can be modeled by collecting novel trajectories and periodically clustering.
- Using the action based activity models, it is possible to summarize what activities have occurred and determine which were atypical. More importantly, activity analysis can be performed in real-time to make predictions on future behavior and provide timely notification of unusual actions and events.

The analysis framework is evaluated on a number of diverse datasets to thoroughly characterize performance. This detailed evaluation is missing from the literature and provides a benchmark for future research.

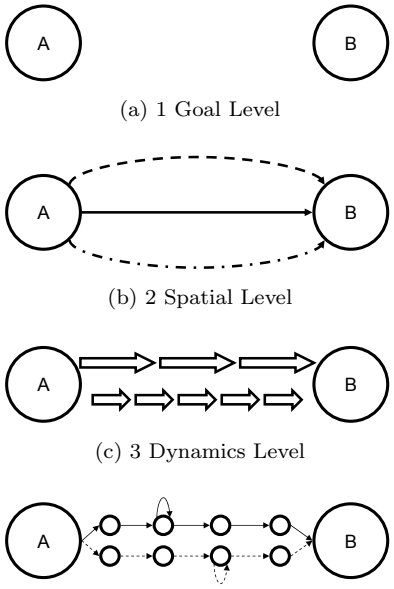
4.2 Hierarchical Activity Decomposition

A key to designing an effective learning framework is to examine and limit the types of behaviors to be analyzed. In the surveillance setting, rough body motion is a low level feature that can be reliably extracted through visual tracking. When using just this coarse feature, a behavior can be characterized by four main components (Fig. 4.1); its goal, the spatial extent, the dynamics, and the temporal duration. Together these characteristics are used to answer questions and describe an activity. They form an explanation hierarchy which allows more complete and refined description of an activity.

Level 1 (Goal)**.** *Describes the beginning and end of a behavior.*

Knowing the goal or objective of an object provides the simplest activity description by providing an origin and destination. This is the type of information is needed by the transportation community to understand where people are traveling and to adequately meet that demand.

Level 2 (Spatial)**.** *Describes where a behavior occurs in space.*



(a) 1 Goal Level

(b) 2 Spatial Level

(c) 3 Dynamics Level

(d) 4 Temporal Level

Figure 4.1: Hierarchy for activity understanding. (a) The goals indicate noteworthy destinations and provide a mechanism for high level reasoning of behaviors. (b) Spatially different routes to reach the same goal describe different behaviors. (c) Similar routes are further distinguished based on dynamic information such as speed. (d) A behavior is temporally decomposed into a set of sequential actions to provide the finest description granularity. Subtleties such as the duration of actions indicate differences between similar behaviors.

By recording the location an agent follows between goals (the route), it is possible to differentiate behaviors spatially. Different routes between locations can correspond to vastly different behaviors such as taking the straight route to a destination versus going on the "scenic" route.

Level 3 (Dynamic)**.** *Describes the manner in which a behavior is performed.*

Besides location, dynamic attributes provide further disambiguation between behaviors. This level allows activities to be separated from within the same route. A tentative driver will have dramatically different velocity and acceleration profile than a "speed demon" even when driving along the same highway.

Level 4 (Temporal)**.** *Describes when a behavior is performed and for what time duration.*

The last component for behavior characterization is the temporal description. By decomposing a behavior into smaller atomic parts, more detailed variations are allowed. This decomposition highlights the sequential nature of actions as well as their time duration with respect to an activity. Similar behaviors may be composed of the same core actions but are distinguished by the amount of time spent in each action or the order the actions are performed.

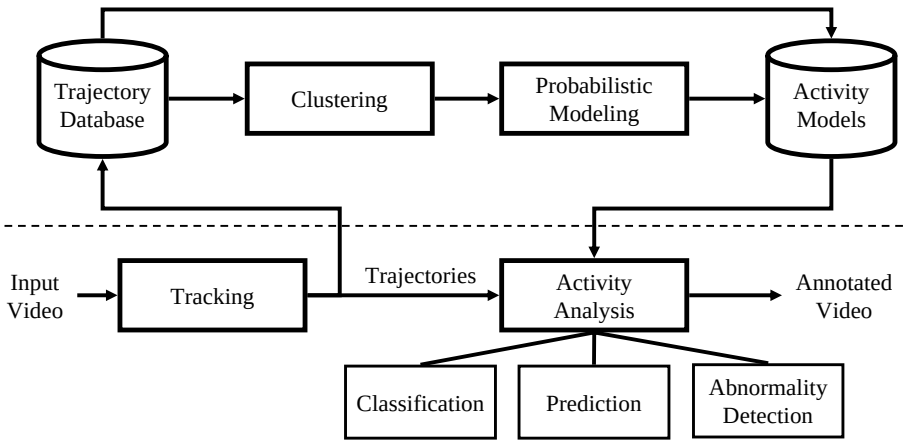


Figure 4.2: General framework for trajectory analysis. Trajectories are collected by visual tracking. They can be clustered into a set of typical patterns which are then probabilistically modeled and used for live activity analysis.

The above definitions describe a hierarchical behavior decomposition useful for description at varying levels of granularity. The multi-leveled interpretation allows disambiguation of behaviors given different criteria with each level adding to the completeness of activity description

4.3 Trajectory Learning Framework

This paper develops a probabilistic trajectory analysis framework, shown in Fig. 4.2, to learn and describe activity. The analysis occurs in two phases. The initial phase, observation, uses the behavior hierarchy to build activity models in the 3 staged approach shown in Fig. 4.3. In the second phase, testing, the activity models are used to describe and predict behavior and detect abnormalities in real-time on live video. The initial activity models are updated with new observations and augmented by consistent behaviors not observed during the training period to better reflect the surveillance scene.

The analysis framework must cope with a number of real-world implementation issues. In a new scene, the number of typical activities is not known a priori and must be estimated automatically. In addition, the learning and evaluation algorithms must be robust to noisy tracking because trackers will invariably fail due to environmental conditions and occlusion. The framework must gracefully handle these incomplete trajectories for real implementation. Finally, when monitoring a site over long periods, the initial training period may not reflect the current scene configuration necessitating approaches to modify the activity models.

During the learning phase, objects are observed and tracked using visual tracking software to compile a trajectory database. The tracker used in this work segments potential foreground objects using motion based background subtraction and tracks are maintained with the

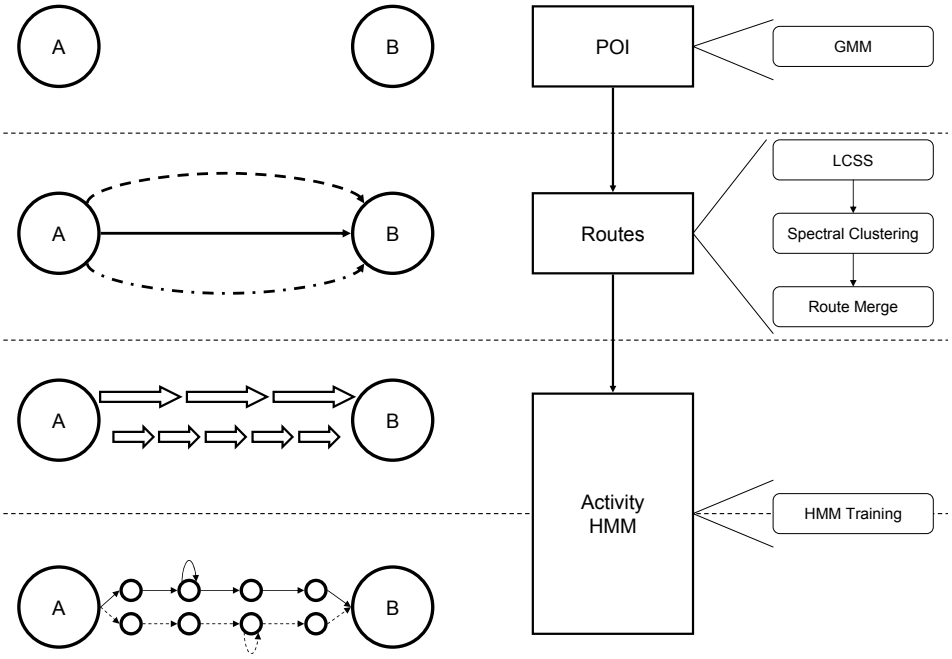


Figure 4.3: In order to match the multi-level behavior model, a three staged learning procedure is adopted to first learn goals, followed by spatial routes between goals, and finally probabilistically model spatio-temporal dyanmics.

use of a Kalman filter to predict motion and an appearance model for consistency [89].

A trajectory is a sequence $F = \{f_1, \ldots, f_t, \ldots, f_T\}$ of T flow vectors. A flow vector $f_t = [x_t, y_t, u_t, v_t]^{\mathrm{T}}$ compactly represents an object's motion at time t by the xy position and corresponding component velocities uv. After trajectories have been collected into a training database, activities are modeled by learning each of the hierarchy levels in the 3 stage approach of Fig. 4.3. Points of interest (POI) are learned through expectation maximization in the goal level. Spatial routes are formed by clustering trajectories. Finally, both the dynamic and temporal characteristics of an activity are encoded probabilistically using a hidden Markov model (HMM) in the third stage. The evaluation phase uses the learned activity models as a descriptive vocabulary for online analysis. As a new object is tracked, its activity is characterized, future behavior is predicted, and alarms can be raised if there is unusual behavior.

The following sections provide more of the framework details, providing explanation of the 3 staged learning procedure and evaluation methodology.

4.4 Goal Level Learning

The first activity level locates points of interest (POI) in the image plane, which are regions where objects enter and leave the scene or where objects tend to spend a significant

amount of time. By learning the graph nodes, a vocabulary to describe the path in terms of goals is formed. The POI are used to filter noise trajectories arising from incorrect tracking by retaining only tracks that begin and end at POI nodes. This filtering procedure provides robustness to tracking failures and improves later clustering results.

4.4.1 Points of Interest

There are three different types of zones in a scene, entry, exit, and stop. The entry and exit zones are the locations where objects either appear or disappear from the scene, such as the edge of a camera field of view, and correspond to the first, f_1, and last, f_T, tracking point respectively. The stop zones indicate scene landmarks where objects tend to idle or remain stationary, such as vehicles at a toll booth or a person sitting at a desk. The stop points arise from samples with very low speed that are also within a small radius R for more than τ seconds [13]. This ensures a stationary object remains in a particular location. Just relying on a low speed threshold can contaminate the set of stop points. For example, every sample of a vehicle tracked during congestion could be consistent with the low speed check even though it travel across the camera field of view.

The interest zones are learned through a 2D Gaussian mixture modeling (GMM) procedure [76]. A zone

$$\mathcal{Z} = \sum_{i=1}^{Z} z_i G([x, y]^{\mathrm{T}}, \mu_i, \Sigma_i) \tag{4.1}$$

is composed of Z Gaussians $G([x, y]^{\mathrm{T}}, \mu, \Sigma)$ of $[x, y]^{\mathrm{T}}$ image coordinates for a compact representation and learned using expectation maximization (EM) [28]. Each mixture component i denotes a different scene POI which is located on the image plane by its mean μ_i and its size is specified by Σ_i.

Since it is not known a priori how many POI belong in a zone, the number must be estimated. A zone is over-mixed by using a large Z to ensure all existing POI are modeled. The importance of a POI is defined by its density

$$d_i = \frac{z_i}{\pi\sqrt{|\Sigma_i|}}. \tag{4.2}$$

Zone components with low density do not have much support in the training data. They are considered of low importance and most likely arise from noise in the tracking process (incomplete trajectories). The noise components can be automatically discarded using a density criterion based on the threshold

$$L_d = \frac{\alpha_Z}{\pi\sqrt{|\Sigma_Z|}} \tag{4.3}$$

where $0 < \alpha_Z < 1$ is a user defined weight and Σ_Z is the covariance matrix of the entire zone dataset [76]. The threshold L_d indicates the density of an average mixture in the zone. In this

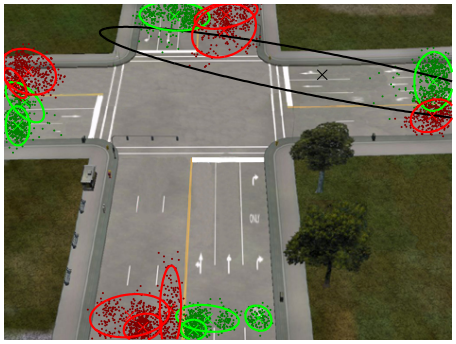


Figure 4.4: Entry/Exit interest are zones learned by Gaussian mixture modeling.

way, tight mixtures with high density indicate a true POI while low density mixtures, $d_i < L_d$, imply tracking noise since they are not well localized.

Fig. 4.4 shows the entry/exit zones learned for an intersection with green denoting entry and red exit POI. Noise mixtures from broken tracks, which are removed, are drawn in black in Figs. 4.14a and 4.15a. The yellow in Fig. 4.16a marks stop zones. Notice the desk in the upper right and smart board in the lower left are correctly discovered but stop zones were also found overlapping with entry/exit zones because objects further away from the omni camera have very little apparent motion due to the high degree of lens distortion.

4.4.2 POI-Based Filtering of Tracks

The POI modeling procedure indicates goals for the surveillance scene which can be used to help the activity training process. Trajectories that do not begin and end in a POI are considered a tracking error and removed from the training database and trajectories that travel through a stop zone are split into separate tracks leading into and out of the zone. This provides consistent trajectories for improved activity modeling. Noise tracks, generated by tracking failure, would make behavior learning more difficult because they are not adequately explained and effort would be wasted if the noise was modeled.

The choice of Z for a zone was not critical because when underspecified, EM will usually create a large mixture covering multiple POI. As long as Z is not so small as to group points from far away POI then noisy trajectories can be removed. Alternatively, instead of using classical EM, it might be possible to automatically select the number of mixture components in conjunction with estimation [35].

4.5 Spatial Level Learning

The second behavior level focuses on spatial support, differentiating tracks based on where they occur. Once the scene goals are explained by POI in the first level, the way from

node to node is described by spatial routes.

4.5.1 Trajectory Clustering

After POI-based filtering, a training database of clean trajectories is available for grouping into routes which separate different ways to get between goals. These routes can be learned in unsupervised fashion through clustering. The clustering extracts the most typical trajectory patterns and only relies on the definition of similarity between tracks. The main difficulty when trying to learn routes is the time-varying nature of activities which leads to unequal length trajectories. They must be compared using a procedure that is able to measure similarity between differing sized inputs.

In this work, spatial routes are learned by first finding the distance between training trajectory pairs using the longest common subsequence (LCSS) distance. LCSS was found to provide the best clustering performance when compared with a number of other popular distance measures [136, 88]. A similarity matrix is formed from the pairwise trajectory distances and spectrally decomposed to generate clusters. Since the number of clusters is not known a priori, the count is estimated through an agglomerative merge procedure to represent the scene routes.

LCSS Trajectory Distance

Since trajectories are realizations of an activity process, the length of trajectories can be of unequal length based on the properties of differing activities. Even worse, even trajectories sample from the same activity can be of unequal length due to sampling rate and speed of action. In order to compare trajectories, a distance measure must be able to handle variable sized inputs.

LCSS is an alignment tool for unequal length data that is robust to noise and outliers because not all points need to be matched. Instead of a one-to-one mapping between points, a point with no good match can be ignored to prevent unfair biasing. The LCSS distance for trajectories suggested by Vlachos *et al.* [123] is defined as

$$D_{LCSS}(F_i, F_j) = 1 - \frac{LCSS(F_i, F_j)}{\min(T_i, T_j)}, \tag{4.4}$$

where the $LCSS(F_i, F_j)$ value specifies the number of matching points between two trajectories of different length T_i and T_j.

$$LCSS(F_i, F_j) = \begin{cases} 0 & T_i = 0 \mid T_j = 0 \\ 1 + LCSS(F_i^{T_i-1}, F_j^{T_j-1}) & d_E(f_{i,T_i}, f_{j,T_j}) < \epsilon \\ & \quad \& \quad |T_i - T_j| < \delta \\ \max\left(LCSS(F_i^{T_i-1}, F_j^{T_j}), LCSS(F_i^{T_i}, F_j^{T_j-1})\right) & \text{otherwise} \end{cases} \tag{4.5}$$

The recursive definition of matches (4.5), which can be efficiently computed with dynamic programming, looks for points that are with small Euclidean distance ϵ and sampled within a δ time window. The term $F^t = \{f_1, \ldots, f_t\}$ denotes all the sample points in F up to time t.

Spectral Clustering of Trajectories

1. Construct similarity graph S,
   $s_{ij} = e^{-D^2_{LCSS}(F_i,F_j)/2\sigma^2}$ (4.6)
2. Compute the normalized Laplacian L,
   $L = I - D^{-1/2}SD^{-1/2}$ (4.7)
3. Compute the first K eigenvectors of L
4. Let $U \in \mathcal{R}^{NxK}$ be the matrix constructed with eigenvectors as columns
5. Cluster the rows of U using FCM

Figure 4.5: Basic steps for spectral clustering of trajectories as presented by Ng *et al.* [91].

Spectral Clustering

Spectral clustering has become popular technique recently because it can be efficiently computed and has improved performance over more traditional clustering algorithms. Spectral methods do not make any assumptions on the distribution of data points and instead relies on eigen-decomposition of a similarity matrix which approximates an optimal graph partition [91]. The basic steps for spectral clustering are outlined in Fig. 4.5

The similarity matrix $S = \{s_{ij}\}$, which represents a fully connected graph, is constructed from the LCSS trajectory distances using a Gaussian kernel function

$$s_{ij} = e^{-D^2_{LCSS}(F_i,F_j)/2\sigma^2} \in [0,1]. \tag{4.6}$$

where the parameter σ describes the trajectory neighborhood. Large values of σ cause trajectories to have a higher similarity score while small values lead to a more sparse similarity matrix (more entries will be very small).

Using the normalized spectral decomposition presented by Ng *et al.* [91] a Laplacian matrix is formed using the similarity of training trajectories

$$L = I - D^{-1/2}SD^{-1/2} \tag{4.7}$$

with D is a diagonal matrix with elements the sum of the same row in S. The first K eigenvectors of L are used as the columns of a new matrix $N \times K$, U . Finally, the rows of U, each corresponding to a trajectory, are clustered using fuzzy C means (FCM) [94]. The initial cluster centers are chosen based on the orthogonal initialization method proposed by Hu *et al.* [50].

The advantage of FCM clustering in the last spectral stage are soft class assignment that minimizes the effects of outliers and the resulting cluster membership values $u_{ik} \in [0,1]$ which indicate the quality of sample i. The membership variable u_{ik} tells how confidently trajectory i

is placed in cluster k. High membership means little route ambiguity, or a typical realization of an activity process.

Route Creation

The spectral clustering process partitions the trajectory training database into the groups of similar patterns. A route is defined as a prototype from each group. The route prototype is chosen as an average of trajectories in a cluster, which utilizes the membership values u_{ik} returned from FCM.

In order to average trajectories of different length, each trajectory F has its velocity information ignored and is spatially resampled to a fixed length L. Rather than simply subsampling the resampled track representation $\bar{F} = \{\bar{f}_1, \ldots, \bar{f}_L\}$ seeks to evenly distribute points along the trajectory (based on arc length). The sampling ensures the distance between consecutive points is equal

$$d(\bar{f}_l, \bar{f}_{l+1}) \sim \frac{1}{L-1} \sum_{t=1}^{T-1} d(f_t, f_{t+1}) \tag{4.8}$$

$$d(f_i, f_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{4.9}$$

to completely remove dynamic information (as hidden in the sampling rate). This prevents regions of higher sample density from contributing bunches of points in a single area as could occur if an object moved more slowly during a section of its trajectory. Finally, a vector $\bar{F} = [x_1, y_1, \ldots, x_L, y_L]$, representing a point in the $\mathbb{R}^{2L}$ route space, is constructed for each of the N training trajectories (the notation for $\bar{F}$ is slightly abused here).

Each route is determined by the weighted average of the training database

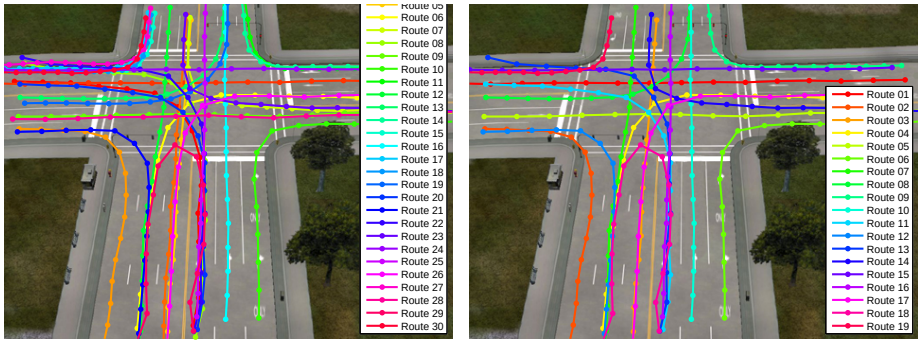
$$r_k = \frac{\sum_{i=1}^{N} u_{ik}^2 \bar{F}_i}{\sum_{i=1}^{N} u_{ik}^2} \tag{4.10}$$

where u_{ik} is the membership of training example i to cluster k and indicates the quality of assignment. High membership indicates little route ambiguity; a typical trajectory of route k. The set of typical routes in a scene are encoded by the prototypes $\{r_k\}$.

4.5.2 Cluster Validation

The number of routes, N_r, in a scene must be estimated because it is not known a priori. Initially, FCM clusters into a large number of prototypes, $K > N_r$. The prototypes are refined to a smaller number of routes (N_r) by merging similar clusters.

The merge procedure compares routes by finding an alignment between pairs of routes using dynamic time warping (DTW) [99]. DTW is used to equally value all matches and provide a one-to-one matching between points on a route. After alignment, two clusters, r_m and r_n, are



(a) (b)

Figure 4.6: (a) Spatial routes learned through spectral clustering with $K = 30$ clusters. (b) $N_r = 10$ remaining routes after merging similar routes.

considered similar based on the count of closely matched points or if the total distance between routes is small. The count of matches is

$$M(r_m, r_n) = \sum_{l=1}^{L} \mathcal{I}\left(d_l(r_m, r_n) < \epsilon_d\right) \tag{4.11}$$

$$d_l(r_m, r_n) = \sqrt{(x_m(l) - x_n(l))^2 + (y_m(l) - y_n(l))^2}, \tag{4.12}$$

where $\mathcal{I}(.)$ is the indicator function. The total distance between routes is

$$D_r = \sum_{l=1}^{L} d_l. \tag{4.13}$$

The threshold value ϵ_d pixels was chosen experimentally for good results and is based on how closely routes are allowed to exist in the image plane. When either $M(r_m, r_n) > T_M$ or $D_r < L\epsilon_d$, then routes r_m and r_n are considered similar. A cluster correspondence list is created from these pairwise similarities, forming similarity groups $\{V_s\}$. Each correspondence group is reduced to a single route. The membership weight of every training trajectory is pushed onto the reduced route, *e.g.*

$$\tilde{u}_{mi} = u_{mi} + u_{ni} \qquad \forall i. \tag{4.14}$$

where $\tilde{u}$ represents the membership after merging.

In practice we have found that FCM tends not to over fit the data but instead finds several very similar clusters, making the merge algorithm effective. Fig. 4.6 shows the paths learned by the clustering and merge procedure. The initial clustering used $K = 30$ but after merging only the $N_r = 19$ true lanes remain.

4.6 Dynamic-Temporal Level Learning

The clustering procedure locates paths spatially but this is insufficient for the highest level of behavior analysis. Not only do we need to know where objects are located but also the manner in which they travel. The dynamics are needed to completely characterize an activity. Using HMMs, the spatio-temporal properties of every activity is encoded probabilistically, completing the third and fourth levels of the behavior hierarchy.

The advantage of modeling activities by HMMs is the sequential nature matches the action sequence view of activity (and time-ordered progression of trajectories) and the simplicity of training and evaluation. HMMs define a natural procedure to compare different length tracks, as will generally occur, through optimal time normalization. Unlike with clustering, the full unsampled trajectories containing position and velocity are used to incorporate dynamics.

4.6.1 Activity HMM

The activity HMM is a standard hidden Markov model that leverages the structure of trajectories. Each HMM is compactly represented as $\lambda_k = (A_k, B_k, \pi_0)$ and is designed to have Q states, for simplicity, rather than to try and estimate an optimal number [113].

The HMM states reflect the underlying actions that make up an activity. An action is described by its position xy and velocity uv. Each of the Q states $\{q_j\}_{j=1}^{Q}$ are modeled by a Gaussian distribution

$$q_j = G(f, \mu_j, \Sigma_j) \tag{4.15}$$

of unknown mean μ_j and covariance Σ_j with flow $f = [x, y, u, v]^{\mathrm{T}}$. The HMM states correspond to B_k and explain the trajectory observations.

The vector π_0 specifies the initial state probabilities of the HMM and is defined based on the sequential structure of activity actions

$$\pi_0(j) = \frac{1}{C} e^{-\alpha_p j} \qquad j = 1, \ldots, Q. \tag{4.16}$$

with C a normalization constant to ensure valid probabilities. This definition allows tracks to begin in any action state which is important for online analysis when tracks are incomplete. The choice of α_p is not crucial as long as there is non-zero probability for each state.

The $Q \times Q$ matrix $A_k = \{a_{i,j}\}$ specifies the probability of transitioning from a state q_i to state q_j. Again, the nature of activity causes A_k to be highly structured. The matrix is block diagonal with highest probability on the main diagonal which indicates the action sequence taken to complete a behavior.

Both the transition probabilities A_k and the action states which define B_k must be estimated while π_0 is fixed to define an activity. The estimation was accomplished using the routes learned through clustering.

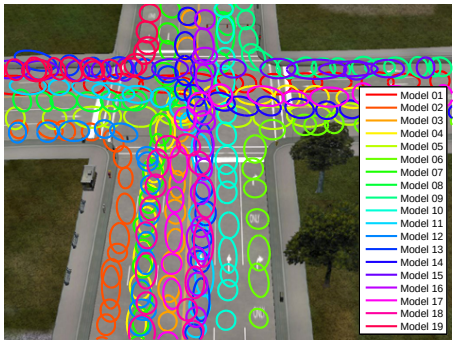


Figure 4.7: Spatio-temporal dynamics of paths encoded by HMMs.

4.6.2 Activity HMM Training

An HMM is trained for each activity by dividing the training set D into N_r disjoint sets, $D = \bigcup_{k=1}^{N_r} D_k$. The set D_k is the collection of trajectories belonging to route r_k based on membership

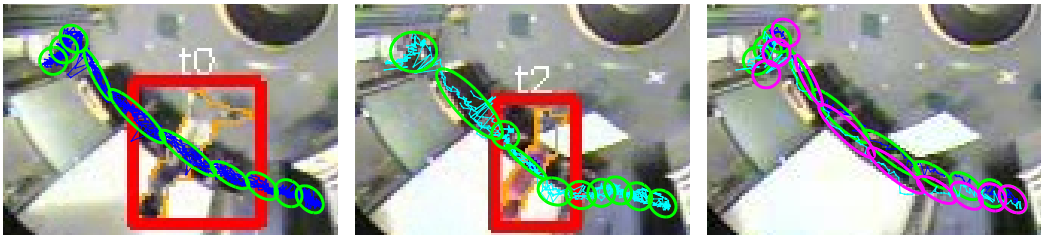
$$k^* = \underset{k}{\operatorname{argmax}}\, u_{ik} \quad \forall i. \tag{4.17}$$

Only those trajectories with membership $u_{ik^*} > 0.9$ are retained when creating training set D_k^* because they are the typical trajectories and can be confidently placed into route r_k^*. By using only highly confident tracks, HMM training is eased and made more precise because of outlier removal. Using path training sets D_k, the N_r HMMs can be efficiently learned using standard methods such as the Baum-Welch method [100]. The set of HMMs learned for the traffic intersection are shown in Fig. 4.7 and specify where lanes are located as well as how vehicles are expected to move in the lane.

There is no need to manually select "good" trajectories for the modeling step because they are automatically discovered through the POI-based filtering and route membership thresholding which makes is easy to deploy a new camera system.

4.6.3 Updating Activities

The activity HMMs learned above accurately depicts the scene at the time of training, but in a surveillance setting there is no guarantee that the activity processes are stationary. Activities will be dynamic and the models must reflect the changes over time. Two complimentary adaption methods are used to update the HMM database. The first is an online scheme which refined the existing activity HMMs based on newly observed trajectories while the second introduced new models by periodic learning rounds.



(a) (b) (c)

Figure 4.8: Online MLLR update of path HMM allows adjustments on the fly. (a) Initial path between doorways. (b) New path learned after a table is positioned to block the walkway. Notice the path must bend around the table. (c) The original path in green is adjusted to track the change with the MLLR incremental adaptation shown in magenta.

Online Incremental Update

After training, the activity models are optimal, for the training data, but might not reflect the current configuration of the scene. Small variations and perturbations could arise from various causes such as camera movement or path reconfiguration, *e.g.* people walking around a puddle. A trajectory deemed to have been generated by a particular activity can be used to update the HMM in an online fashion using maximum likelihood linear regression (MLLR). MLLR computes a set of linear transformations that reduce the mismatch between the initial model and new (adaption) data. The adapted HMM state mean is given by

$$\hat{\mu}_j = W_k \xi_j, \tag{4.18}$$

where W_k is the $4 \times (4+1)$ transformation matrix for activity k and ξ_j is the extended mean vector

$$\xi_j = [1, \mu_j^{\mathrm{T}}]^{\mathrm{T}} = [1, \mu_x, \mu_y, \mu_u, \mu_v]^{\mathrm{T}} \tag{4.19}$$

of state j. The state j represents an individual action that makes up an activity and $\mu_j = [\mu_x, \mu_y, \mu_u, \mu_v]^{\mathrm{T}}$ summarizes the location and dynamics of the action. $W_k = [b\ H]$ produces an affine transformation for each Gaussian HMM state with H is a transformation and b a bias term. The transformation matrix W_k can be found using EM [36] by solving the auxiliary equation

$$\sum_{t=1}^{T}\sum_{j=1}^{Q} L_j(t)\Sigma_j^{-1} f_t \xi_j^{\mathrm{T}} = \sum_{t=1}^{T}\sum_{j=1}^{Q} L_j(t)\Sigma_j^{-1} W_k \xi_j \xi_j^{\mathrm{T}} \tag{4.20}$$

where $L_j(t) = p(q_j(t)|F)$ is the likelihood of HMM state q_j at time t given the new trajectory F. The optimization is performed over all Q states of an HMM to provide an activity level regression.

Each time a new trajectory is classified into path λ_k (Section 4.7.1), a transformation is learned and applied to the mean of each of the HMM states for a sequential online update. The

update

$$\mu_j(t+1) = (1 - \alpha_{\text{MLLR}})\mu_j(t) + \alpha_{\text{MLLR}} W_k(t)\xi_j \quad \forall j \tag{4.21}$$

modifies the mean of existing path λ_k to better fit new observations at time t. The learning rate $\alpha_{\text{MLLR}} \in [0, 1]$ is a user defined parameter set to control the importance of a new trajectory. Using $\alpha_{MLLR} = 0.5$ there was almost complete adjustment after 10 trajectories when a path was blocked by a table and people were forced to walk around (Fig. 4.8).

Incorporating New Activities

The MLLR update allows modification of existing activities but in order to introduce new and unseen activities into the activity set, we adopt a periodic batch learning procedure [48]. Abnormal trajectories (defined in Section 4.7.1) that do not fit any of the HMMs well are collected into an auxiliary training database. Once the database has grown sufficiently large it can be passed through the learning machinery to periodically augment the activity set. Since the abnormality database is populated by trajectories which did not fit any of the existing models, any consistent patterns extracted indicate new activities. In this way, repetitive motions initially considered abnormal could be assimilated into the typical activity set, *e.g.* if construction of a new lane on a highway was completed and opened to traffic.

4.7 Activity Analysis

After the the offline learning process, the underlying activities present in a visual scene are compactly represented by the set of activity HMMs. Using these learned models, the activity of scene agents can be accurately characterized from live video. Activity analysis includes describing current actions, predicting future behavior, and finally, detection of abnormal and unusual events. The processing occurs either after an object exits the camera field of view, with a summary of its activity and whether it was expected, or it occurs online, while under observation and tracked.

4.7.1 Trajectory Summarization

As trajectories were completed, a summary of the object activity was generated. This summary explained what activity the object completed and determined if it was typical.

Trajectory Classification

A novel trajectory can be classified by the activity that best described the observation sequence. Using probabilistic Bayesian inference, the activity label is determined by maximum

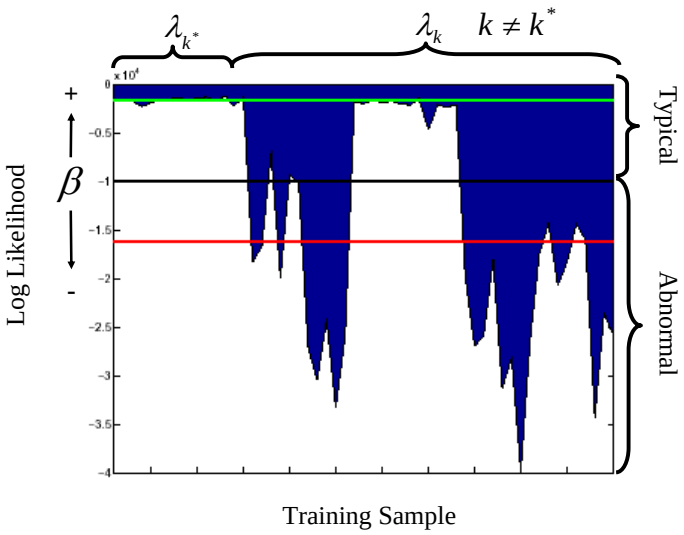


Figure 4.9: Automatic selection of abnormality threshold is based on the average likelihood of in class samples (green) to out of class samples (red). The sensitivity parameter β is a tunable parameter to control the sensitivity of abnormality detection.

likelihood estimation.

$$k^* = \underset{k}{\operatorname{argmax}}\, P(F|\lambda_k) \tag{4.22}$$

The likelihood of activity λ_k can be solved efficiently for the HMM using the forward-backward procedure [100]. The classification determines the activity which most likely generated the trajectory (which HMM best explains the observation sequence).

Abnormal Trajectories

While every track is classified into a path λ_{k^*}, the quality of this assignment will be low for abnormal trajectories. Since only typical trajectories are used to learn the HMMs, outliers are not well modeled. These abnormal trajectories can be recognized as those with low log-likelihood $\log P(F|\lambda_{k^*}) < LLT_{k^*}$. The decision threshold is learned during training by comparing the average likelihood of samples in training set D_k to those outside.

$$LL_k^{in} = \frac{1}{|D_k|} \sum_{i \in D_k} \log P(F_i|\lambda_k) \tag{4.23}$$

$$LL_k^{out} = \frac{1}{N - |D_k|} \sum_{i \notin D_k} \log P(F_i|\lambda_k) \tag{4.24}$$

$$LLT_k = \beta(LL_k^{in} - LL_k^{out}) + LL_{out} \tag{4.25}$$

The sensitivity factor $\beta \in [0, 1]$ controls the abnormality rate with larger β causing more trajectories to be considered anomalous. The automatic threshold selection process if visualized in

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

Figure 4.10: Abnormal trajectories (a) Wide u-turn (b) Illegal loop (c) Off-road driving (d) Travel in opposite direction

Fig. 4.9 where the green line signifies LL_k^{in}, red signifies LL_k^{out}, and black indicates the threshold LLT_k.

A few examples of abnormalities are displayed in Fig. 4.10. In (a), although u-turns were allowed, the wide arc was atypical. The illegal loop off-road driving in (b) and (c) respectively are clearly unusual. The trajectory in (d) looks acceptable but it actually came from a vehicle traveling in the wrong direction.

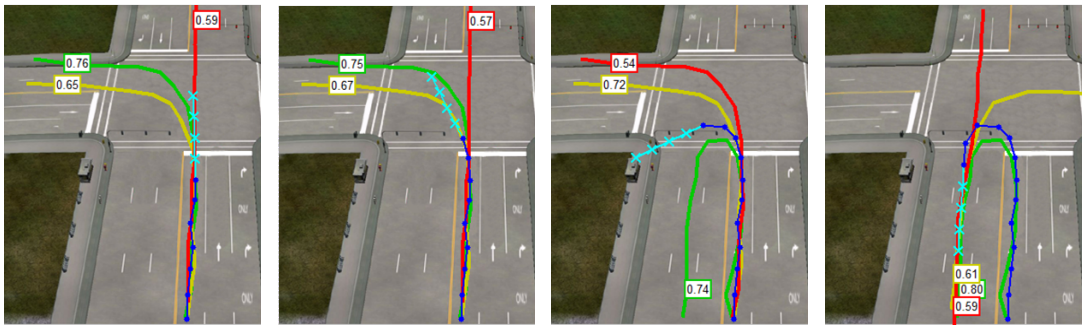
4.7.2 Online Tracking Analysis

Although it is interesting to provide a summary of complete tracks, it is often more important in surveillance to recognize activity as it occurs. In these situations, we want to know what actions are currently being performed in order to assess the surveillance situation. This allows timely response to critical events. It is important to note that within this framework each individual scene agent is considered separately to provide details for each rather than a more holistic frame level (or clip level) description. Rather than state something interesting is happening, the agent level analysis explains where and to whom it occurs.

During online analysis, a description of activity is generated based on the most current information. With each new video frame, a track is updated and the activity description is refined. The difficulty with online analysis is that at the time of evaluation an activity has not been completed (a full trajectory has not been obtained). The activity analysis must operate with incomplete data; the data up to the current time t, $F = \{f_1 \dots f_t\}$. Therefore, the online analysis infers activity based on the actions an agent has undertaken.

Activity Prediction

Online analysis grants the ability to react to situations as they arise. Real-time analysis reasoning provides the alerts necessary for timely response. This response time could be improved if the monitoring system were able to infer intentions and determine what will happen before it actually occurs. Accurate prediction can help reduce reaction time or even avoid undesirable



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

Figure 4.11: Activity prediction during a u-turn. The green signifies the most likely predicted maneuver, yellow the second best guess, and red the top-3 guess with the associated confidence in the colored box. (a) The motion model estimated points (light blue X) fit well during linear motion. (b) During turns, the motion model very poorly approximates the u-turn. (c) Halfway through the u-turn, the prediction can accurately gauge the maneuver while the motion model is very far off. (d) Realignment into a lane but activity memory still indicates the u-turn in green.

situations by providing a buffer to take counter measures and corrective actions.

Future object activity can be inferred from the current tracking information. But, instead of using all the tracking points accumulated up to time t, only a small window of data is retained

$$F_{w_c}^{w_p} = \{f_{t-w_c}, \ldots, f_{t-1}, f_t, \hat{f}_{t+1}, \ldots, \hat{f}_{t+w_p}\}. \tag{4.26}$$

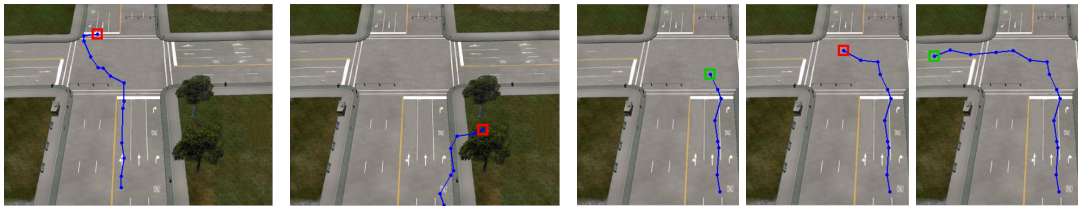
The windowed track consists of w_c past measurements, the current point f_t, as well as w_p future points. The future points $\hat{f}_{t+\tau}$ is estimated by applying the tracking motion model τ time steps ahead. By utilizing only the windowed trajectory, only the recent history is considered during online evaluation because old samples may not correlate well with the current activity. This allows an agent to be monitored over very long time periods by discarding stale data.

The activity prediction is made at the current time t by evaluating (4.22) with F replaced by its windowed version $F_{w_c}^{w_p}$.

$$\lambda_{k^*}(t) = \underset{k}{\operatorname{argmax}}\, P(F_{w_c}^{w_p} | \lambda_k) \tag{4.27}$$

This prediction has a further time horizon than standard one step prediction (Kalman prediction) because it leverages the acceptable activities rather than relying on a generic motion model. During complex maneuvers, motion models will fail when predicting more than a few time steps into the future.

An example of the superiority of the trajectory pattern prediction is shown in Fig. 4.11. The top 3 best activity predictions are color coded as green for best match, yellow as top-2, and red as top-3 and their associated confidence is presented in the colored box. During the straight



(a) (b) (c)

Figure 4.12: Online unusual event detection occurs at time of occurrence. Green bounding box indicates acceptable behavior while red is for something unusual. (a) Illegal loop. (b) Off-road driving (c) A left turn starts out as expected but a peculiarity is detected in the middle because of the sharp transition. A short time later the activity state returns to acceptable.

sections (a) the motion model estimates behavior well with the light blue X's. As the turn progresses, (b) and (c), the estimates drift while the activity prediction accurately determines the maneuver. Finally in (d), the estimates re-align in the straight section but the activity memory helps distinguish the u-turn in green. Over the life of a trajectory, the prediction encodes the object history $\{\lambda_{k^*}(1), \ldots, \lambda_{k^*}(t), \ldots, \lambda_{k^*}(T)\}$. Consistent activity is denoted by consecutive labels which are equal while lane changes occur at the transitions between labels.

Unusual Action Detection

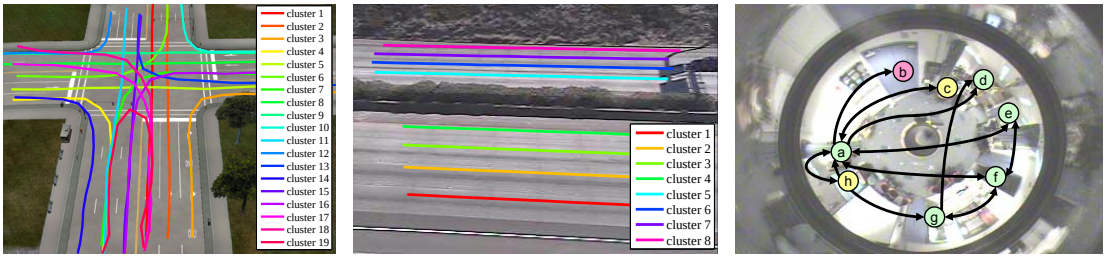
Similar to abnormal trajectories, unusual actions can be detected during tracking. These anomalies indicate deviations the instant they occur. Since only a windowed version of a track is used during tracking the the log-likelihood threshold (4.25) needs to be adjusted. The new threshold is

$$\begin{aligned} LLT_k^t &= \gamma_k^t \left[\beta_t(LL_k^{in} - LL_k^{out}) + LL_k^{out}\right], \\ \gamma_k^t &= \frac{E[\#\{q\}|w_c]}{Q}. \end{aligned} \tag{4.28}$$

The abnormality threshold is adjusted with γ_t to account for the reduced probability mass associated with a partial trajectory. The term γ_t corresponds to the fraction of states visited in the online evaluation window. This assumes uniform state duration for a full trajectory and averages the log-likelihood into each model state Q while the numerator $E[\#\{q\}|w_c]$ is the expected number of states that will be visited in an observation window. The adjustment term is estimated as

$$\gamma_k^t \sim \frac{Qw_c/\bar{T}_k}{Q} = \frac{w_c}{\bar{T}_k}, \tag{4.29}$$

with $\bar{T}_k$ the average length of training trajectories in D_k corresponding to path λ_k. $\bar{T}_k/Q$ the number of samples per state which results in $Qw_c/\bar{T}_k$ as the number of states in a window. Notice that the window only considers w_c and sets $w_p = 0$ to make assessments only on observed data.



(a) (b) (c)

Figure 4.13: (a) CROSS - Simulated traffic intersection with 19 maneuvers. (b) I5 - Highway traffic with 4 lanes in either direction. (c) OMNI - Human activity in a laboratory space viewed with omni-directional camera. The topographical map representation of a scene has POI nodes and graph edges depict the routes.

Table 4.1: Experimental Parameters

	N	D	L	Q	w_c	w_p	β	β_t
CROSS	1900	1764	15	15	10	1	0.88	0.9
I5	606	261	15	15	5	3	0.8	0.95
OMNI	117(131)	55(98)	15	30	30	10	0.9	0.65

Here $\beta_t \in [0, 1]$ is again chosen to ensure detection of most suspicious tracking points. We choose the β_t that results in approximately 10% false positive rate (FPR) on the training set with the assumption that there are no unusual events in the training set. (This is fair because any unusual actions in the training set would be discovered and provide a lower FPR).

As soon as an object strays from an activity model, an unusual action alarm is triggered for timely detection. Examples of unusual actions are given in Fig. 4.12. The illegal loop and off-road driving from Fig. 4.10 are seen in (a) and (b) respectively. The time evolution of a detection in shown in (c) where initially typical behavior is denoted with a green box but in the middle of a sharp left turn the box turns red indicating an unusual action. Finally, a short time later the action state stabilizes back at green.

4.8 Experimental Studies and Analysis

The following section presents performance evaluations for the proposed trajectory dynamics analysis system. We evaluate the accuracy of classification, prediction, and abnormality detection. The results are compiled from a set of experimental studies of different scenes. We consider a simulated traffic intersection (CROSS), Fig. 4.13a, a view of highway traffic on Interstate 5 (I5), Fig. 4.13b, and an indoor laboratory viewed by an omni-directional camera (OMNI),

Table 4.2: Trajectory Experimental Results

	N_p	lane assignment		abnormality detection	
CROSS	19	9191/9500	96.7%	168/200	84%
I5	8	879/923	95%	-	-
OMNI1	7	23/26	88.5%	10/14	71.4%
OMNI2	15	12/16	75%	15/18	83.3%

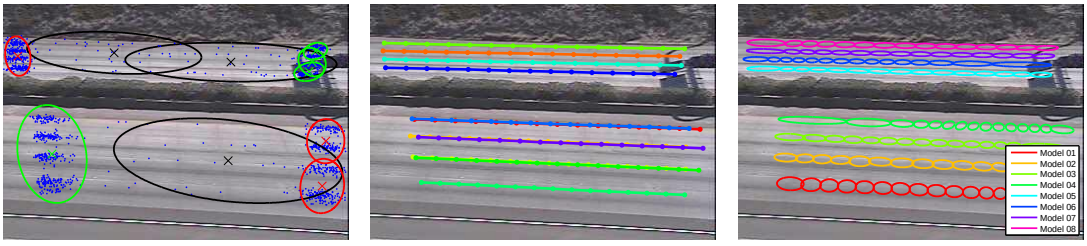
Fig. 4.13c. The parameters used in the experiments are compiled in Table 4.1 while Tables 4.2 and 4.3 summarize the study results.

4.8.1 Quality of Paths

Before evaluating the activity analysis performance, the quality of the automatically learned paths is addressed. The true number of lanes in the traffic scenes is known and the number of paths in the lab scene is manually defined based on examination of the training set.

The lanes of the traffic scenes were identified effectively. All 19 of the intersection maneuvers were discovered. The intersection contained the largest number of paths but they are most easily distinguished because of the favorable camera view. In the I5 experiment (Fig. 4.14), all 8 of the lanes were located but there were also 2 false lanes identified. These extra lanes appear in the southbound direction closest to the camera where perspective distortion causes more variance in lane localization. Since the trajectories logged the centroid of a vehicles' bounding box, different vehicles in the same lane generated slightly different tracks because of object height. In this case, the merge technique was not able to pinpoint a single lane but instead retained two routes. This compromise was necessary to resolve the more tightly spaced lanes in the northbound direction. Performing camera calibration to work in a regular coordinate system rather than the image plane would ameliorate the effects of perspective distortion.

In contrast, the omni camera observed lightly constrained motion through the lab space. Although there are no physical lanes, virtual lanes appear between doorways and desks. There were two separate omni experiments, denoted OMNI1 and OMNI2. The first experiment only contained 7 paths which were all correctly discovered (Fig. 4.15). The OMNI2 experiment shown in Fig. 4.16 was more complex, using the POI nodes shown in Fig. 4.13c there were 15 unique paths. Although the path learning procedure found 15 paths, 2 were unexpected noise paths around the table at node c (radius of stop zones was too small). The 2 missing paths had little training support after POI and membership filtering but could be learned with more data. These OMNI datasets are particularly difficult as paths do not have clear separation and contain significant overlap.



(a) (b) (c)

Figure 4.14: Interstate 5 (I5) experiments. (a) Learned Entry/Exit zones, enter in green, exit in red, and black indicates noise. (b) Routes after merging. (c) HMM path models.

4.8.2 Trajectory Classification

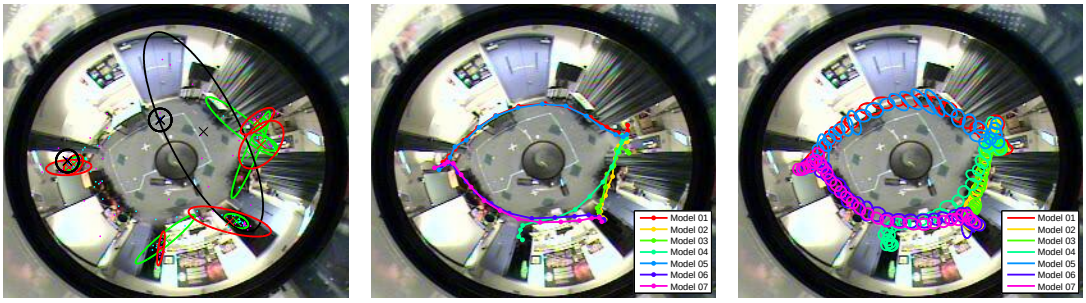
Using (4.22) each sample trajectory is placed into its most likely path. The intersection experiment had 9191 of 9500 trajectories correctly labeled. The lane number for 879 of 923 manually labeled tracks from I5 video were correctly labeled for 95% accuracy. Not surprisingly, most classification errors occur in the northbound lanes (93% vs. 98% southbound) where the lanes appear very close in the image plane. This proximity led to misclassification as large vehicles could occlude multiple lanes. The test set for the OMNI1 experiment was collected over 24 hours on a single Saturday without participant awareness for natural tracks. The 23 of 26 typical trajectories were correctly classified. The OMNI2 test set was collected by test subjects walking through the lab over a 30 minute period. In this set only 12 of the 16 (75%) modeled trajectories were correctly assigned to a path. But, 14/16 were in the top 2 best matches and 15/16 in the top 3.

4.8.3 Abnormal Trajectories

Using $\beta = 0.88$, 84% of the anomalous trajectories were correctly identified in the CROSS experiment. In the first omni experiment, 10/14 abnormal trajectories were detected and 15/18 were discovered in the second experiment. Fig. 4.17 gives examples of abnormal trajectories in the OMNI experiments.

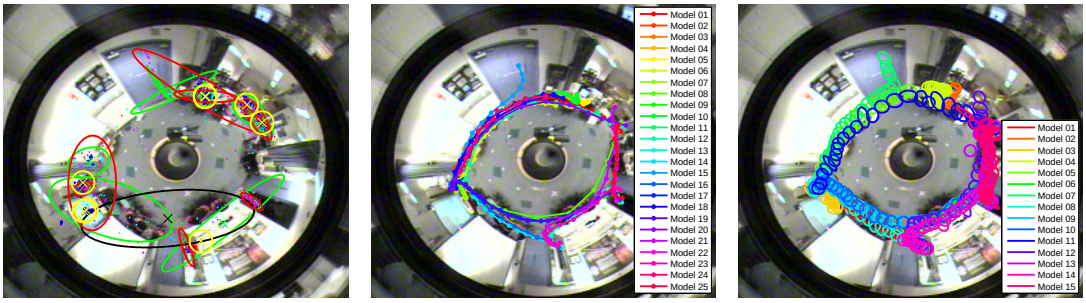
4.8.4 Tracking Classification

The tracking classification accuracy measures how many individual points during tracking had the same label as the true full track label. This assumes an object remains along the same path during its entire tracking life. The traffic scenes, with the simpler lanes, had high tracking classification results. The accuracy was 84.1% for the intersection and 94.4% for the highway. In contrast, the two omni experiments had significantly lower accuracy of 65.4% and 59.4% respectively. They suffered degraded performance due to the complexity of the scene such



(a) (b) (c)

Figure 4.15: OMNI1 experiments (a) Learned interesting zones, enter in green and exit in red. (b) Routes after merging. (c) HMM path models.



(a) (b) (c)

Figure 4.16: OMNI2 experiments, (a) Updated zones, notice there are stop zones (yellow) not previously present in OMNI2. (b) Re-clustering of trajectories forms new routes. (e) Larger set of HMM paths learned in experiment 2.

as partly overlapping paths. When viewing the windowed data, there is little distinction between going from node a to g or from a to f because the routes share significant space. Velocity profiles had the influence when trying distinguishing activities. Looking at the top-2 best matches provided 72.7% accuracy and the top-3 match had 84.2% detection for OMNI1. Using more historical track data would improve classification but add more detection delay to the system. It was not uncommon for the classification to make mistakes at the beginning and toward the end of tracking where data was unavailable and less salient. When ignoring the first $w_c/2 = 15$ samples at the beginning of trajectories improves the accuracy to 68% for the OMNI2 set, almost a 10% improvement.

Table 4.3: Live Experimental Results

	N_p	lane assignment		prediction		unusual event detection	
CROSS	19	35197/41871	84.1%	35077/41871	83.8%	830/999	83.1%
I5	8	14045/14876	94.4%	13859/14876	93.2 %	-	-
OMNI1	7	2054/3139	65.4%	2220/3139	70.7%	756/945	80.0%
OMNI2	15	1599/2693	59.4%	1656/2693	61.5%	-	-

(a) (b)

Figure 4.17: Abnormal trajectories. (a) Abnormal trajectory discovered in OMNI1 training set. Track follows along the border of the room. (b) Abnormal trajectory because of backtracking along path. Red X's indicate the beginning of a tracking abnormality when reversing path.

4.8.5 Tracking Prediction

Similar to classification, prediction tries to classify a window of tracking data, except future tracking points are inferred based on local motion description. Again, the accuracy is measured by the number of prediction labels that share the same label as the full trajectory. The prediction accuracy for the intersection was 83.8% and was 93.2% for I5. The results for the lab were 79.7% for OMNI1 and 61.5% for OMNI2. Clearly the prediction scheme works quite well for straight paths but is limited when analyzing more complex routes, as seen with classification as well. The estimated future tracking points do not always fit the path well when it is curved using just the local dynamics. In a hairpin turn, such as the u-turn in Fig. 4.11, the assumption is to continue straight rather than double back. As with tracking classification, the prediction accuracy for the omni scenes similarly suffers due to path overlap. Yet, surprisingly, by using $w_p > 0$ for prediction there is improvement in accuracy over classification for the OMNI experiments.

4.8.6 Abnormalities During Tracking

Tracking abnormalities indicate the exact time and location of unusual events. It is noted that these events occur in groups of successive points, where the number of points is proportional to the duration. The intersection contained 999 anomalous points and 839 were

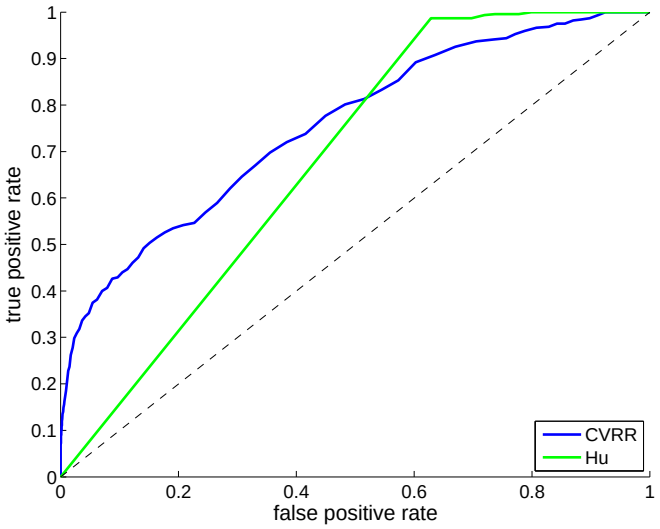


Figure 4.18: Comparison of live abnormality detection ROC characteristics for the CROSS intersection experiment. Using only the current tracking point as done by Hu produces a high false positive rate.

correctly detected. Using $\beta_t = 0.65$ in OMNI1, 756/945 abnormal points are detected. While these results seems promising, it comes at a steep price. The accompanying false positive rate is 55.1 and 63.7% respectively. Fig. 4.18 shows the receiver operating characteristic (ROC) curve for the CROSS experiment in blue. This curve diplays much room for improvement but may be misleading because it only accounts for direct correlation between detections and true abnormalities. If the detection delay is too long there will be no overlap between the detection and true abnormality groups. Even worse, each detection then contributes a false positive. A better measure of accuracy might be the detection of a single live abormality point within an appropriate delay window. Since the ground truth is determined by a human, this also highlights a major difficulty when evaluating an abnormality detection system. There may be time instances where abnormalities are too subtle for a human observer to detect but is very apparent to the computer and it is unknown which "expert" to trust. In Fig. 4.17b we show only the first point in a group of live tracking abnormalities. Though it looks to be an acceptable path, the person backtracks between node a and g before ending at node f.

4.8.7 Comparative Analysis

In order to validate our work, we compare or system against another trajectory analysis system. We use the system presented by Hu *et al*. in 2006 [48] with their improved spectral clustering technique developed in 2007 [50]. The comparison looks at the quality of route extraction from clustering and activity recognition results on the CROSS dataset.

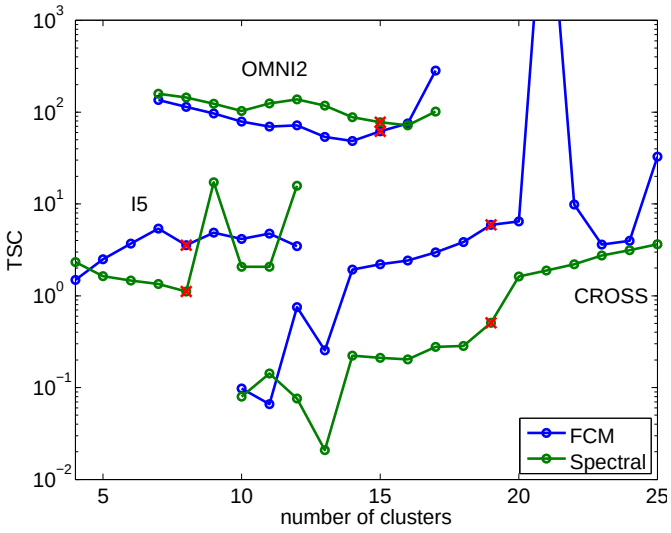


Figure 4.19: TSC measure of cluster quality used by Hu *et al*. to automatically select the number of routes. Spectral clustering had better separation between clusters than FCM but still had difficulty selecting N_r (denoted by the red X).

In order to learn the routes, Hu *et al*. [48] uses spectral clustering and automatically determines the number of routes using the tightness and separation criterion (TSC). The plot in Fig. 4.19 shows the TSC values for different number of clusters K for the test sets. Note: a smaller TSC value indicates better clusters (tighter more compact clusters with greater separation between different clusters). Although spectral clustering produces better separated routes than FCM it is not readily apparent what the optimal choice for K should be. The red X indicates the true number of paths and unfortunately choosing the K value that minimizes the TSC does not always produce the correct number because of local minima. In contrast, by our over-clustering method, all paths were found but with multiple routes in dense areas rather than providing an underestimate. The merge technique was effective when the separation between routes was similar but not as effective with large perspective distortion. The automatic choice of clusters is a formidable problem yet to be solved as both techniques presented have difficulties.

Next, the results of activity recognition between the two systems is compared. In [48], the routes are modeled as a chain of Gaussians very similar to an HMM. The main difference is time normalization is done probabilistically with the HMM while the chain does this based on track length (dividing a trajectory into Q parts with each part corresponding to a small set of points). Using the average distance between a trajectory and a chain, the likelihood of a path is estimated as an exponential random variable. Their abnormal trajectory threshold is chosen as the minimal value in a path set D_k. Finally, they estimate the probability of an unusual point during live analysis by seeing how well the current tracking point is modeled by the best fit Gaussian state. Table 4.4 gives quantitative comparison between the two systems on the intersection dataset.

Table 4.4: Comparison with Hu *et al.* [48] for CROSS dataset

	assignment	abnormality	prediction	live unusual
Hu [48]	98.01%	91.0%	69.48%	31.39%
CVRR	96.7%	84.0%	84.1%	53.84%

Notice there is no live classification column because activity classification and prediction are equivalent in Hu's implementation. Since a Gaussian chain and an HMM are very similar, the results for full trajectories are quite similar, but the HMM paths are much more effective during live analysis. The HMM time normalization procedure is more robust than the length based normalization used by Hu. The ROC curve in Fig. 4.18 shows significant improvement when using a small time window over just the current sample point. When accepting only 10% FP there is almost 30% better performance.

4.9 Concluding Remarks

This chapter proposes a general framework for live video activity analysis based on trajectory learning. A behavior vocabulary is learned in a 3 stage hierarchical process that locates goals with Gaussian mixture modeling, connects them spatially through clustering, and probabilistically models spatio-temporal dynamics by HMMs. In addition to learning activities, the models are able to adapt to changing conditions to better represent the scene's current configuration for long term analysis. This allows object trajectories to be classified into a typical activity, predict future behavior, and detect abnormal actions in real-time. The comprehensive experimental analysis of three varied scenes demonstrates the generality of the trajectory analysis framework.

4.10 Acknowledgements

Chapter 4 is in part a reprint of material in submission to the IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, and material that appears in the Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2009, the Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), 2008, and the Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), 2008, all by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of these papers.

Chapter 5

Monitoring Highway Traffic

The previous chapter explained how a scene can be automatically monitored just through careful observation of motion patterns. This chapter presents VECTOR (VEhicle Classifier and Traffic flOw analyzeR), a vision system for monitoring highway traffic as a concrete example of the utility of trajectory based analysis. In addition to replicating current highway management measurements, visual tracking and trajectory pattern analysis provides informative information not accessible through standard inductive loop sensors.

The learned motion patterns define the lane configuration of the highway link and the manner in which traffic flows. A more complete characterization of highway usage is presented by examining fleet composition (types of vehicles on the road), the safety of the link is characterized based on daily speed profiles, and spatial occurring events (*e.g.* lane changes and abnormal driving patterns).

5.1 Motivation

The last 100 years has brought great advancements and developments in personal transportation transforming the horse drawn world into one dominated by automobiles. The emergence of the automobile has opened up the world, providing almost unlimited access and mobility. In the United States alone, a staggering 240 million vehicles travel over 12 million miles annually on a network consisting of 4 million miles of road whose maintenance costs $40 million [120]. Such a large infrastructure has immediate social, economic, energy, and environmental impact. Motor vehicle taxes generate $30 billion annually. Americans use 175 billion gallons of fuel for highway travel releasing a number of emissions into the air. Between 1995 and 2001 there was 10% increase in average commute time as people experienced slower speeds and increased delay while stuck in congestion [119]. Perhaps most alarming were the 2.5 million injury accidents and 41 thousand motor vehicle related fatalities in 2008 [117]. These numbers represent just a

small portion of worldwide dependence on automobiles and emerging countries such as India and China will dramatically effect the future of transportation.

To manage such a vast transportation network, it is essential to invest in intelligent transportation systems (ITS) technologies. ITS solutions provide the means to extract and manage information necessary for continued advancement. Without their use it would be impossible to continually monitor our roadways, as no human could process such large amounts of data. The key requirements for successful ITS systems are the ability to extract and process data in real-time, provide robustness to a wide range of operating conditions, and technologies that can adapt to changes in the environment which is essential for long term deployment. The ITS community has the power to improve the quality of modern life by providing greener transportation solutions, greater satisfaction through smoother commutes, and ultimately a safer driving experience.

In order to assess a highway system, traffic engineers, planners, researchers, and users need data collection, processing, and analysis tools to provide performance measures. These performance metrics enable ITS solutions. They aid traffic engineering operational decisions and assist planners determine how to alleviate congestion bottlenecks or optimally place traffic control equipment.

The main sensor in use for traffic applications is the inductive-loop sensor which is able to sense passing metal objects. The loop is a mature technology that is resistant to inclement weather and other environmental factors and provides counts on the number of vehicles it sees (flow) and the amount of time it is active (occupancy). Although it is the industry standard, inductive loops must be installed into the road surface which requires a proper maintenance schedule for operation [65]. Unfortunately, even with with a regular service routine, many sensors will be down at a given time. California has an extensive network of loop sensors covering the entire state to monitor 30,573 miles of highway. Out of the 30,246 deployed detectors, it is estimated that 28.4% are not operable [19]. (This is just about the national average 25% [65]). A sensor is considered bad for a number of reasons with the most often reasons cited as no data transmitted, a controller that is down, or a sensor card just being off. There are a number of reasons for loop sensor failure but the most often cited is road wear-and-tear. It is costly, both in service time and associated delay for lane blockage, to repair these broken sensors.

In contrast to inductive loop sensors, video based traffic sensors are appealing because of generally lower maintenance and replacement costs [81]. Video cameras are an attractive infrastructure sensor because of an active research community, high informational content, and widespread deployment. Video sensors are also able to directly measure traffic flow parameters such as density and speed which must be inferred using loops from algorithms that analyze flow and occupancy. In addition, cameras provide a visual record (when saved) which humans can easily interpret, a wide field of view which enables coverage of multiple lanes and gives spatial information (origin-destination (OD) [80]), provides rich contextual data (vehicle appearance),

and can be utilized for multiple uses (red light violation, speed cameras). Red light and speeding cameras have been widely deployed by a number of agencies because they enable an accessible view of conditions with relative ease and minimal intrusion. These cameras provide valuable video signals that can be reused not just for enforcement but for alternative computer vision based traffic management such as highway congestion performance measurements [85], detection of stalled vehicles [57, 58], vehicle classification [84], as well as intersection safety [49, 106].

An active computer vision and ITS community ensure continual advancements in processing power and capabilities. To be useful, the visual traffic parameter data must be generated in real-time and stored in a database for highway model construction. This supply of historical information is required researchers for managing traffic congestion [101], is necessary for traffic prediction [124], and provides other application benefits by viewing data over time rather than just using current conditions [102].

5.1.1 Questions to Address

In this chapter we address the following questions:

- Can a single camera replace multiple inductive loops for highway monitoring?
- What added value can video supply to traffic management?
- How can historical measurements be utilized for activity analysis?

5.1.2 Chapter Summary

In this chapter, the VEhicle Classifier and Traffic flOw analyzeR (VECTOR) system is introduced which provides video based highway analysis. VECTOR provides wide coverage of a highway link using a single camera to classify passenger vehicles, generate loop-like measurements, provide real-time link efficiency estimates, and maintain day of the week history for vehicle speed profiles. This work proves the following:

- A single camera is able to generate traffic measurements similar to loops for all lanes and in both directions of travel. This makes video based highway monitoring an affordable alternative to inductive sensors.
- In addition to loop measurements, video provides a visual record of vehicles that can be used to determine the types of vehicles using a road segment based on appearance. The FOV of a camera provides spatial support which enables direct speed calculations as well as the study of the paths drivers take (for lane changes and unusual activity).
- Utilizing historical measurements provides context for behavior analysis. The record of speed and flow over the course of a day enables real-time link efficiency calculation and detection of potentially dangerous fast drivers.

The VECTOR system for highway monitoring provides a case study which proves the utility of trajectory learning techniques. Using the same observe and learn paradigm, more cameras can be used along the highway to provide greater coverage.

5.2 Related Work

Video has become a favorite traffic monitoring sensor because of flexible installation and high information content. With a single camera, a full highway segment can be monitored without disrupting traffic to install loops into every single lane. Initially, researchers tried to imitate loop detectors in video. Virtual loop detectors were arranged on the roadway in video to accumulate video counts [67]. The virtual loop acted exactly like an inductive loop placed in the ground, it detected when a vehicle crossed over it. Similarly, virtual lines have been constructed to count vehicles traveling in opposite directions [116]. The line stretches across the highway, perpendicular to the road direction, in order to count all vehicles with a single "sensor." These virtual sensor techniques did not take full advantage of the spatial information as observed in an image.

Visual tracking maintains a record of each vehicle while in the camera view. This record enables simple counting for flow as well as speed estimation by measuring the distance traveled in a time interval. Objects are detected for tracking with three popular methods; interest point detection, model fitting, and background subtraction. The main problem in video tracking is the extraction of objects in a wide range of conditions, enabling accurate association. Interest point based trackers [10] are robust to occlusion because they handle missing features at the cost of more complicated data association. Model based approaches deal with changing lighting and occlusion by fitting the object model, 3D models [49, 64] or contours [14], to the data. Unfortunately, it is computationally difficult to enumerate models for all potential objects. Background subtraction based trackers provide computationally simple solutions for foreground extraction but have diminished performance when occlusion or shadows are present. Recent work has looked to overcome shortcomings due to occlusion [57, 64, 122, 79, 139, 41], shadows [46, 98, 47], and from weather and lighting conditions [58, 139, 109, 54]. The computational simplicity coupled with the many techniques to combat lighting and occlusion issues has made background subtraction a favorite vehicle detection technique.

Unfortunately, the accuracy of the vehicle tracking framework for highway monitoring is dependent upon the quality of segmentation. The segmentation task becomes much more difficult as environmental factors change (lighting, shadows, occlusion, or camera motion), when objects are very small, or in very dense traffic situations (congetsion). To avoid these issues, some avoid tracking all together and instead rely on a holistic approach to examine group motion. The spatio-temporal properties of motion fields at different levels of congestion are learned to categorize traffic video [21]. This type of analysis only classifies the level of congestion but does

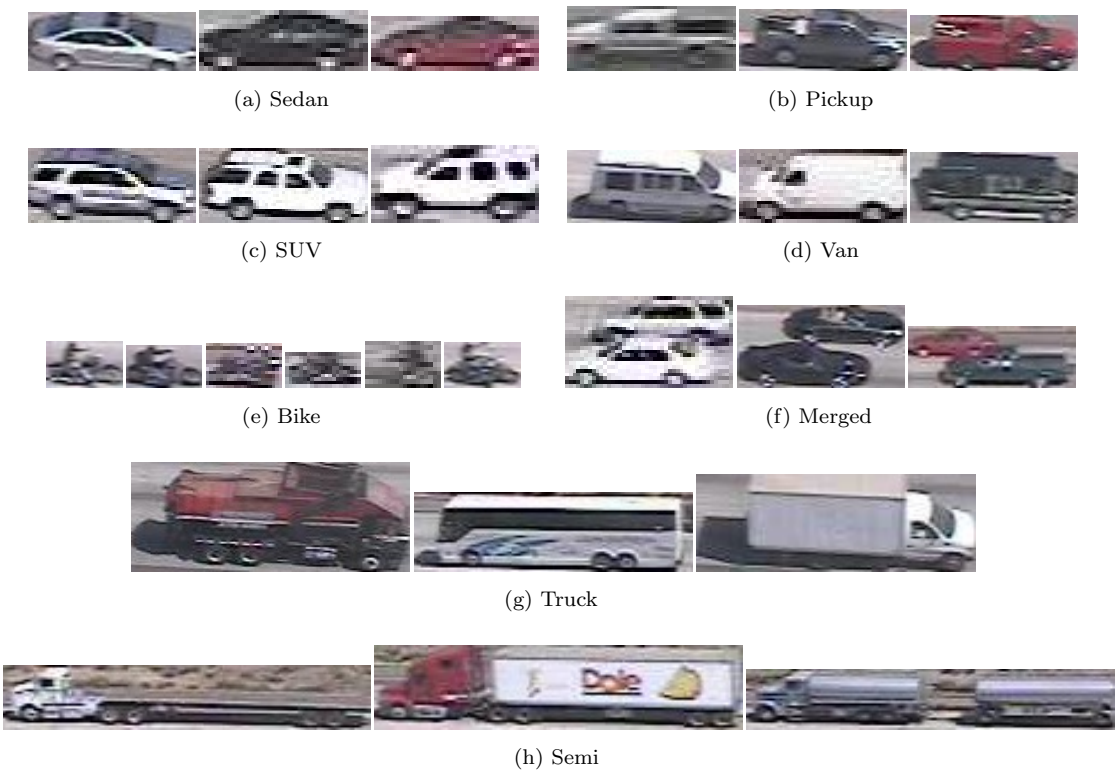
Table 5.1: Selected Studies in Video Based Vehicle Classification

paper	accuracy %	classes	speed	measurement
Lipton 98 [72]	83 (4 hrs)	3 - car, people, other	14 fps	dispersedness
Gupte 02 [43]	70 (20 mins)	2 - car, non-car	15 fps	width, height
Avery 04 [6]	92 (1 hr)	2 - truck, non-truck	-	length
Gepperth 05 [38]	90	2 - car, non-car	>30 fps	oriented energy
Hasegawa 05 [45]	91	6 - pedestrian, sedan, van, pickup truck, mule, other	2-3 fps	blob measurements
Zhang 06 [135]	70	4 - truck, sedan, van, pickup	5 fps	image based PCA
VECTOR	87 (1 hr) 78 (14 hrs)	8 - sedan, pickup, suv, van, semi, truck, bike, merged	30 fps	blob measurements

not quantify it.

A missing measurement need for transportation management is accurate fleet composition. This data is useful for understanding the differing effects of commercial or private vehicles on highway control and for the study of environmental impact from emissions [20] or infrastructure load assessment [118]. Current loop technology is able to separate large, multi-axle, vehicles from passenger vehicles but has difficulty make fine distinctions. Cameras offer a visual description of objects that can be leveraged to determine type. Table 5.1 provides a short summary of vehicle classification research. The aim is to distinguish between as many vehicles types as possible in real-time. Early work used a single dispersedness measure to separate vehicles and humans from other objects [72]. Vehicles in a parking lot were classified into 6 general types using linear discriminant analysis (LDA) of blob measurements [45]. A non-linear hierarchical classifier has been constructed by inputing a vehicle image into a multi-class kernel support vector machine (SVM) [135]. Good model type recognition results have been demonstrated using a hierarchical algorithm by extracting features from the front of a vehicle [137].

The rest of the chapter describes VECTOR's functionality. First, a method to accurately classify passenger vehicle types is presented followed by traffic statistic accumulation. The traffic measurements are used to characterize the highway segment usage and detect potentially dangerous driving. Finally, in depth experiments showed the effectiveness of vehicle classification and traffic measurement and explained how trajectory learning is used for further highway characterization.



(a) Sedan (b) Pickup

(c) SUV (d) Van

(e) Bike (f) Merged

(g) Truck

(h) Semi

Figure 5.1: Sample images from each vehicle class.

5.3 Vehicle Classification

VECTOR classifies vehicles into the 8 different types, {Sedan, Pickup, SUV, Van, Semi, Truck, Bike, Merged}, seen in Fig. 5.1. The block diagram depicting the VECTOR classification scheme is in Fig. 5.2. After object detection, the extracted blob features (3.13) are transformed into a space that better separates the vehicle types using Fisher's linear discriminant analysis (LDA) [9]. For each frame a vehicle is tracked, its transformed features are used to generate a single frame classification using a weighted K nearest neighbor (wkNN) technique. Finally, the frame classification results are compiled and incorporated into a single vehicle label for the entire track.

5.3.1 Vehicle Features

In order to distinguish each of the vehicle types, discriminating features had to be chosen to adequately express the inter class variation.

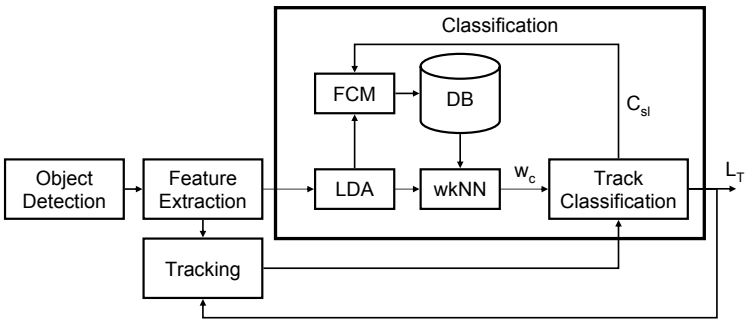


Figure 5.2: Block diagram for the proposed classification scheme.

Selection

Three different types of features were examined for vehicle type discrimination, image (im), blob (bm), and edge (em). The image features considered the raw pixel intensity of aligned vehicle detections. The blob measurements consisted of 16 features obtained using morphological operations, $m_t = [\eta_0, \ldots, \eta_{15}]^T = \{$area, breadth, compactness, elongation, perimeter, convex hull perimeter, length, long and short axis of fitted ellipse, roughness, centroid, the 4 first and second image moments$\}$ [108]. The edge based features were constructed using the popular SIFT [74] descriptor. Unfortunately, SIFT was not usable because of the low resolution of far away vehicles creating spare representations.

Processing

Rather than operate directly on raw features, the vehicle feature vector was transformed into a lower dimensional space to reduce computational complexity for real-time implementation. Features were projected either using principle component analysis (PCA) based on the covariance of all vehicles or using linear LDA which tries to separate classes using individual statistics.

Let $D_c = \{x_1, \ldots, x_{N_c}\}$ be a set of N_c training vectors for class c, each of dimension d, with mean $\mu_c = \frac{1}{N_c}\sum_{i=1}^{N_c} x_i$. The full training set, $D = \{D_1, \ldots, D_C\}$, is composed of the training samples from all classes and has mean $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, where $N = \sum_c N_c$. The LDA projection is given by the maximization problem

$$P_{LDA} = \underset{w}{\operatorname{argmax}} \frac{|w^T S_B w|}{|w^T S_W w|} \tag{5.1}$$

where S_B is the between class scatter matrix and S_W is the within class scatter matrix.

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{5.2}$$

$$S_W = \sum_{i=1}^{C} \sum_{x_k \in D_C} (x_k - \mu_i)(x_k - \mu_i)^T \tag{5.3}$$

$$\tag{5.4}$$

Table 5.2: Image and Measurement Based Classification Rates %

Classifier	Sedan	Pickup	SUV	Semi	Total
PCA-im	89.51	25.00	45.22	100	72.21
LDA-im	51.54	55.00	26.09	25.00	45.60
PCA-bm	90.43	20.00	23.48	91.67	67.12
LDA-bm	90.43	40.00	53.91	100	76.52

The solution to this maximization leads to the generalized eigen problem $S_B w = \lambda S_W w$. The top M eigenvectors are retained to obtain the LDA projection matrix,

$$x_{LDA} = P_{LDA} x = [w_1, ..., w_i, ..., w_M] x \tag{5.5}$$

Comparison

A comparison between the image and blob features is shown in Table 5.2 for a 4 class problem. The blob features were generally better than the image results because of difficulty accurately aligning vehicle samples. The combination of LDA on blob features was chosen for VECTOR because it was the best performing technique and because generating blob features and the projection were quite fast because of the small dimensionality.

5.3.2 Track Based Classification Refinement

Information redundancy obtained by the sequential images of a vehicle during visual tracking can be exploited to improve vehicle type classification. The uncertainty from a single video frame can be reduced through track based refinement to overcome noisy measurements.

Detection Classification

The wkNN rule [45] is a modification of the NN classifier to assign a sample to every class rather than a binary indication of class membership. This soft membership is encoded in the class weight w_c, which builds robustness to noise and outliers. A larger weight indicates a higher likelihood of class c being a match. The weight for a particular class w_c is determined by adding the similarity of the K closest training samples with label c. The label of an individual detection, L_D, is the class with highest weight.

$$w_c = \sum_{\substack{i=1 \\ m_i \in D_c}}^{K} \frac{1}{\| x_i - x_t \|} \tag{5.6}$$

$$L_D = \underset{c}{\text{argmax}}\ w_c \tag{5.7}$$

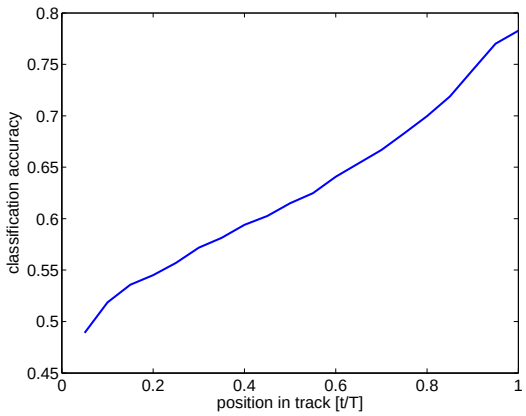


Figure 5.3: The classification accuracy improves as more tracking information is used with track based refinement. By using the complete track to do classification there is approximately 30% improvement over using just the first sample in a track. T is the total number of samples in a track and $t \leq T$ is the number of samples used for classification.

In (5.6), x_t is a new test sample to classify given the training, $D = \bigcup_{c=1}^{C} D_c$, set consisting of all x_i. To completely specify the weights of the C vehicle types, $K \times C$ comparisons must be made.

Track Classification

Tracking gives a record of an object while in the camera view. Each time instant along a track is an example of the object, giving us T examples over the course of a track. Given these T samples, the track classification is found by maximum likelihood estimation.

$$\begin{aligned} L_T &= \underset{c}{\operatorname{argmax}} \sum_{t=1}^{T} ln\, p(x_t|c) \\ &= \underset{c}{\operatorname{argmax}} \sum_{t=1}^{T} ln \frac{w_c^t}{\sum_c w_c^t}. \end{aligned} \tag{5.8}$$

The likelihood $p(x_t|c)$ of class c is approximated by normalizing (5.6) for each sample t in a track to be a valid probability. The track class is refined each frame as the track is updated. The track label takes into account all the evidence throughout the entire track to make a decision on class type rather than a single frame measurement that could potentially be corrupted by many sorts of noise. It is shown in Fig. 5.3 that as more information is utilized, the classification accuracy improves. The final track label is the last class assigned before the track ends. Fig. 5.14 gives examples of the track classifier overcoming incorrect detection classification results.

Classification Confidence

The confidence in classification label L_T can be measured by the track likelihood

$$C_{prob} = \frac{1}{Z} \max_c \prod_{t=1}^{T} p(x_t|c), \qquad Z = \sum_{c=1}^{C} \prod_{t=1}^{T} p(x_t|c) \tag{5.9}$$

$$C_{prob} = \max_c \sum_{t=1}^{T} ln\, p(x_t|c). \tag{5.10}$$

or by the sidelobe ratio

$$C_{sl} = \frac{p_1 - p_2}{p_1}, \tag{5.11}$$

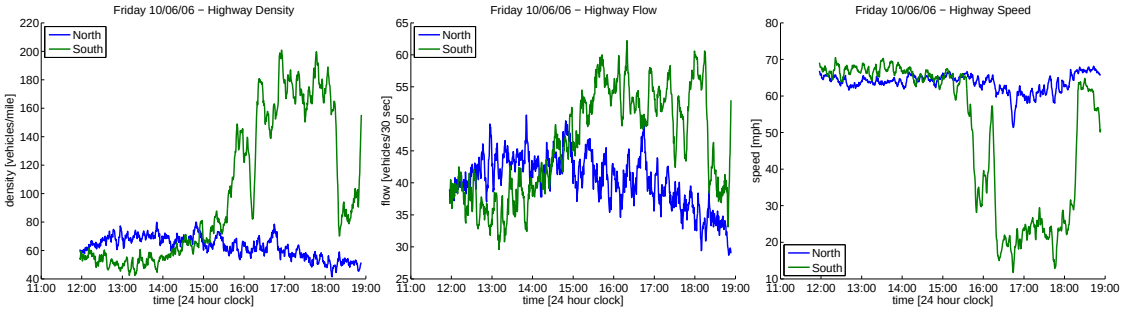
where p_1 and p_2 are the first and second highest track likelihoods. The sidelobe ratio is a more appropriate measure because it gives a measure of how much stronger class L_T is than the closest competing class (see Fig. 5.13 in Section 5.5).

5.4 Highway Flow Analysis

Flow analysis utilizes the trajectory learning models of activity to describe the road and lane configuration. As vehicles are tracked, their current lane number is determined using the techniques of Chapter 4 to mimic the output of inductive loop sensors. Using cameras has the advantage of providing appearance and spatial readings not possible with loops. Vehicle level analysis is accomplished through object classification and through understanding of trajectory patterns which enable detection of lane changes an abnormal activity.

5.4.1 Traffic Statistics

Using trajectory information, the time series of fundamental highway usage parameters, analogous to those obtained from conventional loop detectors, is collected in real-time. This system delivers flow ($\frac{\#vehicles}{time}$), density ($\frac{\#vehicles}{distance}$), and average speed (MPH) in 30 second intervals, averaged over a 5 minute window. The primary traffic measure of flow counts the number of vehicles every 30 seconds and indicates link usage. The VECTOR flow statistic is generated by counting the number of passing vehicles in the 30 second update interval. The vehicles are counted as they exit the camera field of view to simulate a spot sensor. Density is the average number of vehicles in the camera view normalized by the roadway length and measures highway crowding. The link density can be directly measured using the area coverage provided by cameras whereas it must be inferred based on occupancy (% of loop activity) and flow using loops. The speed is the average velocity of all tracked vehicles in the 30 second interval which is difficult to obtain using loops. Manual roadway calibration was necessary in order to obtain localized units and convert from pixels/sec into MPH. While it is possible to obtain speed measurements with special loop configurations (double loop detectors), research has shown that



(a) Density (b) Flow (c) Speed

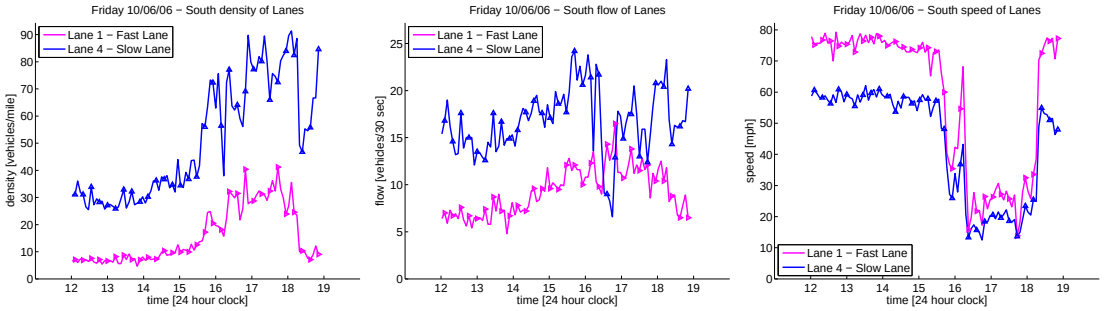
Figure 5.4: Density, flow, and speed for north and south bound directions of Interstate 5.

they are unreliable and that speed calculation algorithms based solely on flow and occupancy are superior [19].

Fig. 5.4 gives examples of the accumulated statistics in the north and south bound directions of US Interstate 5 (I5) on a Friday evening. Density greatly increases in the southbound direction between 15:00-16:00 with an accompanying increase in flow. But, the increased usage leads to a large reduction in link speed. Once the evening commute is in full swing, between 16:00-18:00, the speed is only 20 MPH, density is capped at approximately 175 vehicles per mile, and the flow follows a downward trend after reaching its limit of 60 vehicles per 30 seconds.

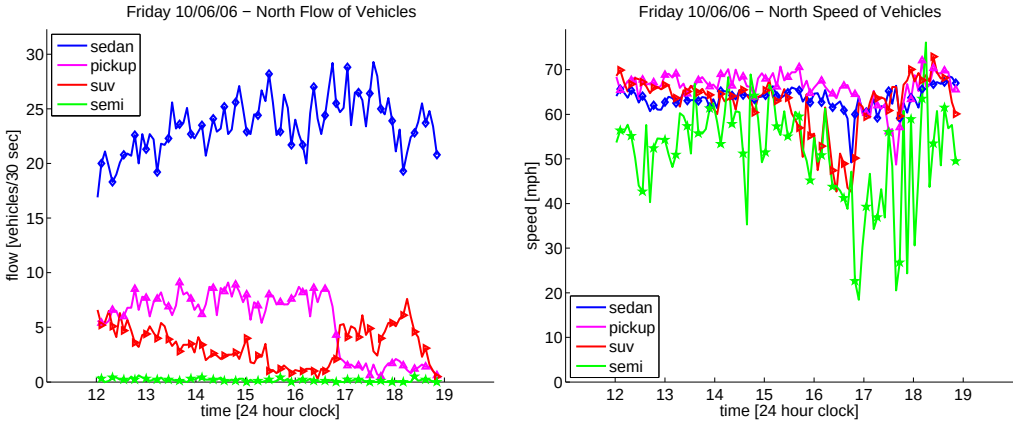
In order to match loop detector data, measurements must be taken for each highway lane. The lane is determined during tracking based on the road configuration. Each tracked vehicle is placed into its corresponding lane depending on its trajectory position to observe the effects of individual lane congestion. Fig. 5.5 shows the south bound statistics for each of the highway lanes. In 5.5a it is evident lane 4 (the slow lane) is occupied by more vehicles. During the commute hours this difference is greatly increased from 30 vehicles/mile to 80 vehicles/mile which causes congestion. This is revealed in Fig. 5.5b by noting the increased flow and density until a sudden flow drop shortly after 16:00. The congestion in the slow lane spills over into the adjacent lanes causing a comparable loss in speed over all the lanes as is evident in Fig 5.5c. This phenomenon demonstrates the need for on and off ramp management to control the slow lane as well as the entire highway link itself.

Besides reproducing loop detector data, video provides a means for extracting more rich contextual information. Traffic parameters are compiled for each type of vehicle based on the vehicle classifier. Fig. 5.6 plots the flow and speed of different vehicle types on a weekday. In Fig. 5.6a there are clearly many more sedans on the road than any other class of vehicles but during the evening commute the number of Pickups and SUVs on the road appear to switch; during the day there are more Pickups and during rush hour there are more SUVs. One may speculate this occurs because contractors and other workers (construction or landscaping) who need pickups start and end their work earlier than the more typical 9-5 day. In 5.6b it is noted



(a) Lane Density (b) Lane Flow (c) Lane Speed

Figure 5.5: Individual lane density, flow, and speed for south bound direction of Interstate 5.



(a) Flow: Distribution of highway usage by vehicle type.

(b) Speed: Semi trucks travel at noticeably lower speeds.

Figure 5.6: Traffic statistics separated by vehicle type.

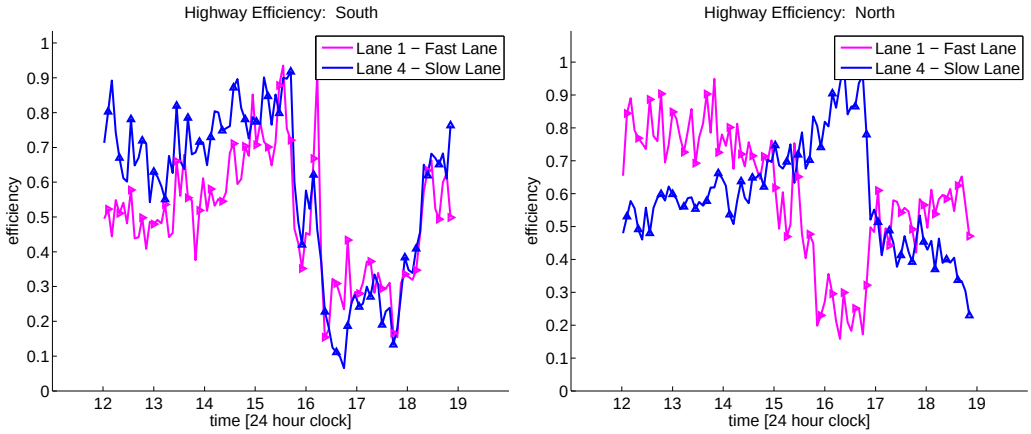
that most of the vehicles travel at approximately the same speed (the speed of traffic) but the larger Semi trucks tend to travel slower than passenger vehicles, matching intuition.

5.4.2 Utilization and Efficiency

Flow, speed, and occupancy are useful traffic parameters for congestion management [22]. Chen *et al.* showed that congestion was not caused by demand exceeding capacity but because of inefficient operation of highways during periods of peak demand. Their work considered a productivity argument to define link efficiency as

$$\eta = \frac{VMT/60}{VHT}. \tag{5.12}$$

The value $VMT = flow \times section\ length$ is the total number of vehicle miles traveled, $VHT = \frac{VMT}{speed}$ is the total number of vehicle hours, and 60 is the sustained speed at maximal flow in mph



(a) South: Low efficiency during evening commute because of congestion.

(b) North: Efficiency drop in fast lane because low flow caused under utilization.

Figure 5.7: Highway lane efficiency.

based on empirical distribution data. The efficiency, η, is a measure of congestion delay, or the extra time vehicles are forced to travel below 60 MPH.

Using the accumulated VECTOR usage statistics, the efficiency, at a given time t, can be better estimated taking into account changes in flow by

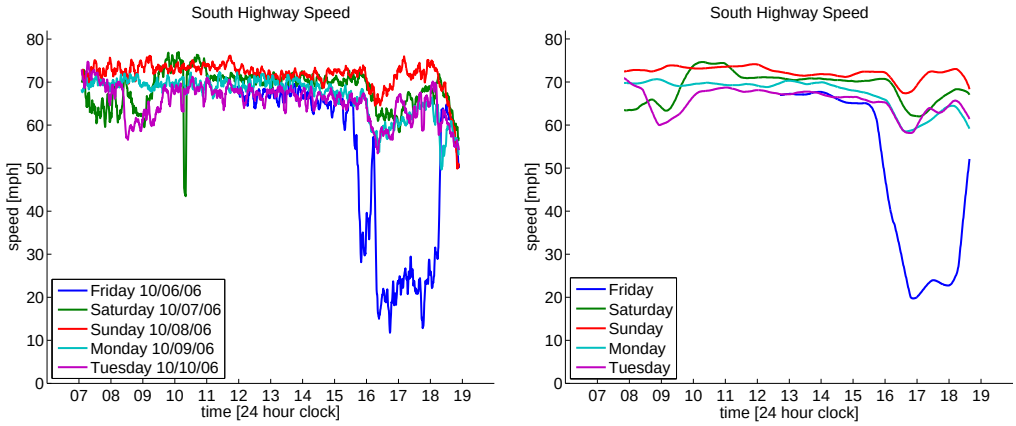
$$\hat{\eta}(t) = \frac{flow(t) \times speed(t)}{flow_{max} \times speed_{max}}. \tag{5.13}$$

This efficiency estimate $\hat{\eta}$ is the ratio of the actual to maximal throughput. The maximum values of flow and speed are maintained by averages over time (see Section 5.4.3) which means the efficiency rating can be greater than 1. Using VECTOR, the efficiency can be directly computed from raw measurements in real-time.

Fig. 5.7 shows lane efficiency over time of the north and south bound directions of the highway. Congestion is clearly evident during the evening commutes, where the efficiency drops significantly. It is interesting to note that while the efficiency of the south bound direction drops because of congestion the north bound highway does not suffer from congestion. The reduced efficiency in the fast lane is actually due to under utilization. Here the flow in lane 1 is much lower while utilization of lane 4 increased due to an exit ramp. These usage statistics can be used to develop control strategies to reduce congestion and save delay.

5.4.3 Daily Speed Profile

The large amounts of data collected by this system allows usage analysis not just over the course of a single day, but over many days. To build useful highway models, it is important to incorporate the differences in traffic behavior as a function of time. Fig. 5.8 demonstrates the differences in the speed profile for work and non-work days. The Friday congestion slow



(a) (b)

Figure 5.8: Highway speed profile comparing weekend and weekdays.

(a) Car slowing (yellow box) before stopping on the shoulder.

(b) Car coming to rest (red box) on the shoulder of the highway.

Figure 5.9: User defined speed profiling.

down between 16:00-19:00 is severe and significantly greater than the other weekdays. (The Monday and Tuesday commute is noticeable but it is a much more subtle speed disturbance). This demonstrates the need to model each day individually.

By using a database of historical speed measurements, a model of daily highway speed patterns can be constructed to incorporate the traffic speed fluctuations during the course of a week. Seven daily models are generated by averaging across each specific day (Fig. 5.8b). During live tracking, the VECTOR system indicates the motion state of each vehicle by the color of its' bounding box; {speeding, normal, slow, stopped} = {blue, green, yellow, red}. The motion state

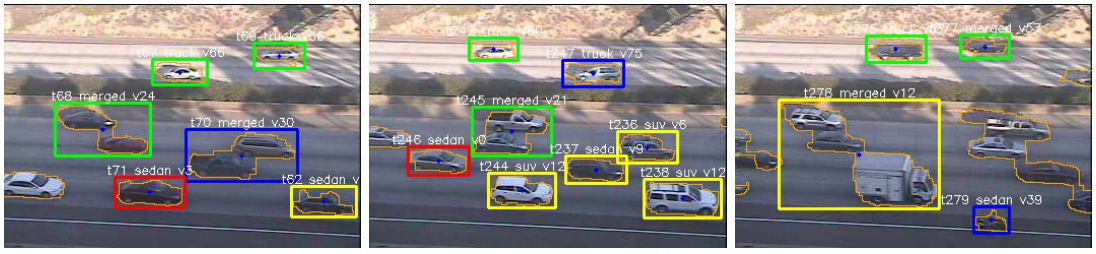


Figure 5.10: Friday evening high volume traffic showing merging of vehicles. The north and south bound directions have differing speed profiles because of southbound congestion.

is defined as

$$S_V(v) = \begin{cases} Stopped, & 0 \le v < 0.15V_D^d(t) \\ Slow, & 0.15V_D^d(t) \le v < 0.6V_D^d(t) \\ Normal, & 0.6V_D^d(t) \le v < 1.1V_D^d(t) \\ Speeding, & 1.1V_D^d(t) \le v \end{cases}, \tag{5.14}$$

where $V_D^d(t)$ is the average speed at time t for the day D in direction d ={North, South, East, West} and v is the vehicle speed. The average speed $V_D^d(t)$ is the average speed as observed over a number of training days. An example of the speed profiling is demonstrated in Fig. 5.9. A test vehicle slows (Fig. 5.9a) before coming to a complete stop on the shoulder of the road (Fig. 5.9b). Fig. 5.10 demonstrates the speed profiling at 18:30:00 during a Friday evening commute. Notice the north bound direction only contains freely moving vehicles while there are slow moving ones (red and yellow bounding box) in the south bound lanes. Different profiles are maintained in both the north and southbound directions to properly account for their own unique temporal variations. On Friday (D =F), congestion causes $V_F^S(18:30) \approx 25$ MPH going south ($d = S$outh) while North retains the faster $V_F^N(18:30) \approx 70$ MPH flow. The speed state can be used as an indicator of dangerous situations because it locates abnormal and potential unsafe driving patterns based on historical data.

5.5 Experimental Evaluations

The following experiments test the performance of the VECTOR system. Classification is tested over the course of a single day and flow analysis is compared both with hand counted vehicles and with inductive loop detector data from Berkeley's Freeway Performance Measurement Project (PeMS) [19]. The manual evaluation checks the raw 30 second output over a small interval while the PeMS loop comparison determines utility over longer time periods.

(a) 08:17:19 (b) 10:13:32

(c) 14:06:17 (d) 17:00:48

(e) 17:45:06 (f) 19:43:02

Figure 5.11: Examples of typical illumination and shadow variations in the highway scene over a day.

Table 5.3: Percentage Accuracy for Hourly Test Clips

time	sedan	pickup	suv	van	semi	truck	bike	merged	total	count
06:21	94.9	59.5	81.9	31.6	50.0	33.3	0	97.5	81.5	405
07:19	96.2	32.5	83.2	06.7	66.7	25.0	100	98.0	84.7	497
08:17	61.2	33.3	91.9	14.3	50.0	50.0	100	96.7	67.6	530
09:15	53.2	38.7	82.3	53.9	37.5	23.1	100	96.8	63.7	444
10:13	36.8	26.7	77.3	26.7	71.4	40.0	0	93.9	51.0	357
11:11	63.4	47.2	90.4	28.0	66.7	33.3	-	89.6	68.6	417
12:09	86.0	71.7	82.6	48.0	50.0	37.5	100	96.9	80.8	432
13:08	95.6	76.3	83.5	39.1	100	50.0	-	97.9	87.0	393
14:06	96.9	77.8	84.2	18.2	-	66.7	100	94.9	86.2	449
15:04	96.0	76.4	81.9	23.1	100	09.1	100	100	85.4	492
16:02	97.1	66.2	76.0	24.0	100	55.6	100	100	85.7	553
17:00	99.1	65.5	62.0	03.6	-	0	100	94.5	83.0	630
17:45	89.0	75.9	52.8	10.0	-	100	67.0	97.7	76.0	297
18:45	96.0	57.9	73.9	10.5	-	100	50.0	97.9	84.6	382
19:43	95.0	77.8	78.5	0	100	100	-	100	86.5	222

5.5.1 Vehicle Classification

The visual VECTOR system is in constant operation, continually generating tracking, classification, and traffic flow data. It is important for this system to work over long periods of time to be competitive with the loop detector network. This section presents classification results over a 24 hour test period. Example frames from the test are shown in Fig. 5.11 and demonstrate the various lighting conditions encountered over a day. Vehicles are classified into 8 different types {Sedan, Pickup, SUV, Van, Semi, Truck, Bike, Berged}, with example images of each shown in Fig. 5.1. The classes considered are the most often occurring vehicle types based on the 2001 National Household Travel Survey (NHTS) conducted by the U.S. Department of Transportation (DOT) [119]. This survey found non-commercial vehicle use of 56.5% wagon/auto/car, 18.4% pickup, 12.0% suv, 9.0% van, and 2.2% bike. The remaining classes of truck and semi model large commercial vehicles, while the merged type is used to indicate occlusion. The TSV class is a conglomerate of trucks, SUV, and Vans and comprise examples that were too difficult to be classified by human because of poor image resolution or vehicle ambiguity.

The classifier evaluation was conducted over a 24 hour period. The highway NTSC (352x240 resolution) video was streamed and processed at approximately 15 frames/sec. Five minute clips were annotated at each hour of the day for careful evaluation. A total of 6,500 vehicle tracks were labeled. The class distribution can be seen in Table 5.4.

Table 5.4: Confusion matrix for all hours of test. Total classification accuracy of 78.4% over 6500 test tracks

	sedan	pickup	suv	van	semi	truck	bike	merged
sedan	2726	127	202	55	0	0	1	0
pickup	40	374	52	24	0	14	0	4
suv	411	113	1147	172	0	3	0	4
van	15	11	54	83	0	6	0	7
semi	0	0	0	0	26	1	0	1
truck	1	5	1	2	11	36	0	0
bike	1	0	0	0	0	0	18	0
merged	7	7	6	10	3	31	2	677
total	3201	637	1462	346	40	91	21	702
% correct	85.2	58.7	78.5	24.0	65.0	39.6	85.7	96.4
% correct	85.2	58.7	80.1		56.5		85.7	96.4

Table 5.5: Confusion matrix for test clip at 14:06:17 on 06/12/06: Classification accuracy of 87.1% best performing time.

	sedan	pickup	suv	van	semi	truck	bike	merged
sedan	216	4	9	3	0	0	0	0
pickup	3	35	3	6	0	0	0	0
suv	2	3	65	17	0	0	0	1
van	0	0	1	6	0	0	0	0
semi	0	0	0	0	0	0	0	0
truck	1	2	0	0	0	4	0	1
bike	0	0	0	0	0	0	7	0
merged	0	1	0	1	0	2	0	57
totals	216/222	35/45	65/77	13/33	0/0	7/7	4/6	56/59
% correct	96.9	77.8	84.4	18.2	-	100	66.7	94.9

Vehicle Classifier Training

The classifier was trained by running the tracking software on video and manually labeling the vehicle type of the 4778 detected vehicles. A large amount of data is generated in a relatively short amount of time because each frame of a track gives another example of a particular vehicle. Unfortunately, the classifier is scene specific and any new location would have to be manually re-trained because the features are view dependent.

24 Hour Results

Using $k = 5$ for the wkNN classifier, a 78.4% classification rate was obtained over the 6500 tracks. The hourly classification accuracy can be seen in Table 5.3. Unfortunately, the classifier did not work well for all times of the day. At night, the low light conditions impeded

Table 5.6: Comparison of Nearest Neighbor Classifier Variants

	merged	sedan	pickup	suv	semi	van	truck	bike	total
NN	0.9302	0.7604	0.5513	0.6411	0.5750	0.2120	0.4286	0.7619	0.6963
kNN	0.9459	0.8391	0.5719	0.6815	0.5750	0.1920	0.4176	0.6667	0.7463
wKNN	0.9644	0.8516	0.5829	0.7813	0.6500	0.2321	0.3956	0.8571	0.7809

vehicle detection and instead focused on headlights. During the daylight hours, classification accuracy is usually in the mid 80% except between the morning hours of 08:00-12:00 where it drops significantly. Strong shadows caused distortion of vehicle shape which caused many sedans to be misclassified as an SUV. The vehicle $t2$ in Fig. 5.15 is a correctly detected sedan but $t3$ has a larger detection outline making it easily confusable with a SUV based on the blob measurements. The less severe performance hit between 17:00-18:00 is again attributed to shadows, but this time to environmental occlusion from trees on a hill (Figs. 5.11d and 5.11e). The utilization trajectory based classification helps recover from regions of poor detection due to cast shadows.

The performance of each vehicle type is presented in Table 5.4. Similar accuracy is obtained for Sedan, SUV, Bike, and Merged but Van and Truck had very low performance. Since Van appear quite similar to SUV and Truck with Semi, combining them resulted in a 3% improvement to 81.8%. A comparison with Table 5.5, which was the best performing hour, shows the most difficult class to distinguish is the Van. The lower rates associated with the Truck class mostly arise because the object detected would fragment these larger vehicles. Clearly, more robust detection schemes which account for shadows *e.g.* [98] would improve the results.

Nearest Neighbor Comparison

The performance of common NN variants are compared in Table 5.6 which shows that the wkNN technique has significant improvement over traditional NN and moderate gains over standard kNN, which only considers the k closest neighbors and ignores the labels. Interestingly, the performance of the wkNN classifier does not change much for different k values unlike kNN which needs large k to approach the wkNN rate (Fig. 5.12).

Track Based Refinement

Using the multiple views afforded by tracking, classification improves 8% using the full trajectory as opposed to the first frame. The performance improves as more information is included as can be seen in Table 5.7. The best match (BM) method [45] collects the classification weights at each frame and at the end of a trajectory makes a final class label decision based on

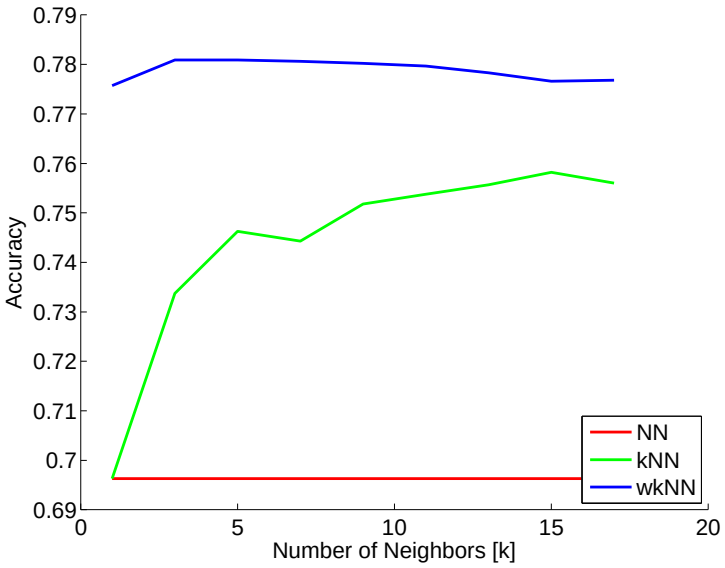


Figure 5.12: The performance of kNN is improved as the number of neighbors k is increased but it has little effect on the wkNN.

the highest observed weight.

$$L(t) = \underset{c}{\operatorname{argmax}}\ w_c(t) \tag{5.15}$$

$$w(t) = \max w_c(t) \tag{5.16}$$

$$t^* = \underset{t}{\operatorname{argmax}}\ w(t) \tag{5.17}$$

$$L^* = L(t^*). \tag{5.18}$$

The trajectory refinement method has only a small 2% improvement over BM. Prior work using a different set of vehicle classes and smaller training set displayed significant improvements [83]. In that work, using trajectory refinement produced a 14% improvement.

Confidence Weighted Classification

In Fig. 5.13, the classification accuracy is presented based on the classifier confidence. When considering vehicles with higher confidence, only examples greater than the threshold $C_{sl} > C_T$, the performance improves as expected. Accuracy increases from 77.5% to 94% while simultaneously using 6,500 to 1,336 tracks between 0% to 99.99% confidence. At low confidence there is a larger gap between the full 8 class and smaller 6 class problems. This indicates the trade-off between classification accuracy, number of vehicle types, and confidence. The plots were generated by utilizing a random subset, 60% split, of the training database and averaging the results of 20 different runs on the 24 hour test data.

Table 5.7: Tracking Refinement Compared with Best Match

	$t = 1$	$t = 0.5T$	$t = T$	BM
06:21	0.7630	0.8074	0.8049	0.7704
07:19	0.7425	0.8068	0.8471	0.8290
08:17	0.6075	0.6434	0.6755	0.6679
09:15	0.5518	0.6014	0.6374	0.6396
10:13	0.4790	0.5154	0.5070	0.5182
11:11	0.6307	0.6715	0.6835	0.6451
12:09	0.7176	0.7894	0.8056	0.7917
13:08	0.7964	0.8575	0.8651	0.8397
14:06	0.7996	0.8597	0.8619	0.8441
15:04	0.8110	0.8435	0.8537	0.8435
16:02	0.7920	0.8300	0.8571	0.8354
17:00	0.6857	0.8190	0.8286	0.8079
17:45	0.6128	0.7306	0.7306	0.6936
18:45	0.8063	0.8377	0.8429	0.8377
19:43	0.7928	0.8649	0.8649	0.8243
total	0.7071	0.7666	0.7809	0.7634

5.5.2 Flow Comparison

The feasibility of using a single camera to monitor all lanes in both direction of a highway link was examined in two ways. Raw sensor measurements, as accumulated in 30 second time intervals, were compared with hand counted flow over a small time period. A longer analysis examined how well VECTOR's measurement matched the loop data accumulated by PeMS.

Manual Flow Comparison

The accuracy of the VECTOR measurement system was tested by comparing the estimated flow to manual vehicle counts over a 30 minute period. The true 30 second vehicle counts are averaged in a 5 minute sliding window for a direct comparison with the VECTOR output. The comparison plots in Fig. 5.17 plots the 5 minute hand count average as a blue line, a green line represents the VECTOR output, and a red line shows the error. The ground truth flow error is usually less than 2 vehicles in every 30 second window demonstrating the estimate accuracy. The strong correlation between the truth data and the visual VECTOR output is clearly evident in lane 1 but there is inconsistency before 16:48 in lane 4 as seen in Fig. 5.17b. The flow is underestimated because of roadside shadowing similar to 5.11e. The two slowest lanes were covered by shadows and many vehicles were not detected. The detection parameters were adjusted (σ_0 in (3.3)) correcting the flow estimates for agreement with the truth data as shown in Fig. 5.17d.

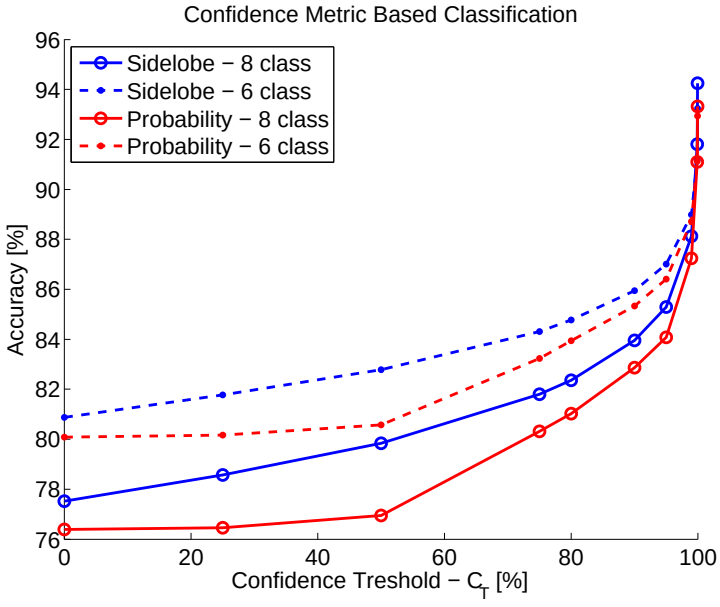


Figure 5.13: Classification accuracy using different number of classes and confidence metrics. The 6 class test combines SUV+Van and Semi+Truck. Only vehicles with $C_{sl} > C_T$ are considered when determining the classification accuracy.

PeMS Loop Comparison

The extracted traffic parameters were also evaluated over a longer time period by comparison with loop detector measurements available from the PeMS website. The PeMS website provides flow and average speed in 5 minute aggregate windows rather than the raw 30 second intervals output by the sensors. Figs. 5.18 and 5.19 plot the south bound PeMS data along with the 5 minute corrected VECTOR estimates.

The comparison plots in Figs. 5.18 and 5.19 show strong correlation between the video and loop based statistics. Flow spikes are accurately tracked close to 18:00 in Figs. 5.18b and (c). In the Monday plots, there seems to be a noticeable difference which is most apparent when examining speed (Fig. 5.19c). There is an early morning drop in PeMS speed and conversely a significant evening slowing of traffic in the VECTOR plot. Visual inspection of the video stream confirmed the slowing traffic described by VECTOR. The discrepancy in speed measurements come from the differing sensor configurations, Fig. 5.16. The PeMS loop detectors are on the opposite north side of the Busy Genesee Ave. ramp from the camera setup. The speed is mismatched because of vehicles entering and exiting this ramp. The higher level of congestion on Friday evening causes a backup north of the Genesee ramp to the southbound loop detector while on Monday it is not as severe and does not extend as far. The Monday morning slowdown observed by PeMS is from cars exiting the highway.

The added coverage area, lower cost of service, and multi-use possibility from video

(a) (b)

Figure 5.14: Examples of track classification correcting for misclassified detection. (a) Track 40: Van misclassified as SUV by Detection Classifier but correctly labeled by Track Classifier. (b) Track 56: SUV misclassified as Van by Detection Classifier but correctly labeled by Track Classifier.

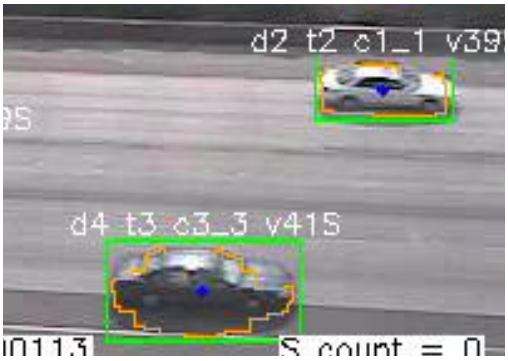


Figure 5.15: Object detection corrupted by shadows making a sedan appear as a SUV.

monitoring provides clear advantages over inductive loop sensors. The performance comparison of VECTOR with true flow and PeMS results show visual sensors can accurately estimate traffic parameters making it as useful as a traffic sensor.

Occlusion Handling

Care was necessary when comparing PeMS to VECTOR. The camera-roadway configuration was chosen for high classification but was not optimal for detection. Perspective distortion could cause multiple vehicles to merge into a single detection. The Merged vehicle class was defined in order to capture and compensate for this deficiency. The merged vehicles were statistically counted when determining flow aggregates. During free flowing traffic, occlusions occurred in 10.7% of tracks. (This includes anywhere during tracking not just at the counting region at the edge of the camera FOV). The merge situation significantly increases during high density as shown in Fig. 5.10. In congestion, 25% of all trajectories where merged with 20% of those containing 3 or more vehicles. When a merged track left the camera FOV, it was statistically counted as 2.17 which was the mean of the empirical distribution of vehicles in a merge situation.

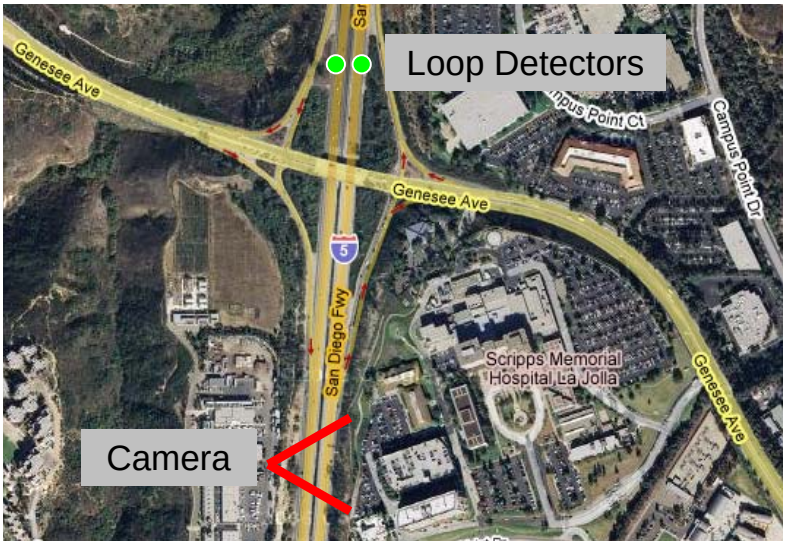


Figure 5.16: Camera and PeMS loop detector sensor configuration on opposite sides of the Genesee Ave. ramp.

Table 5.8: I5 Lane Classification Rate %

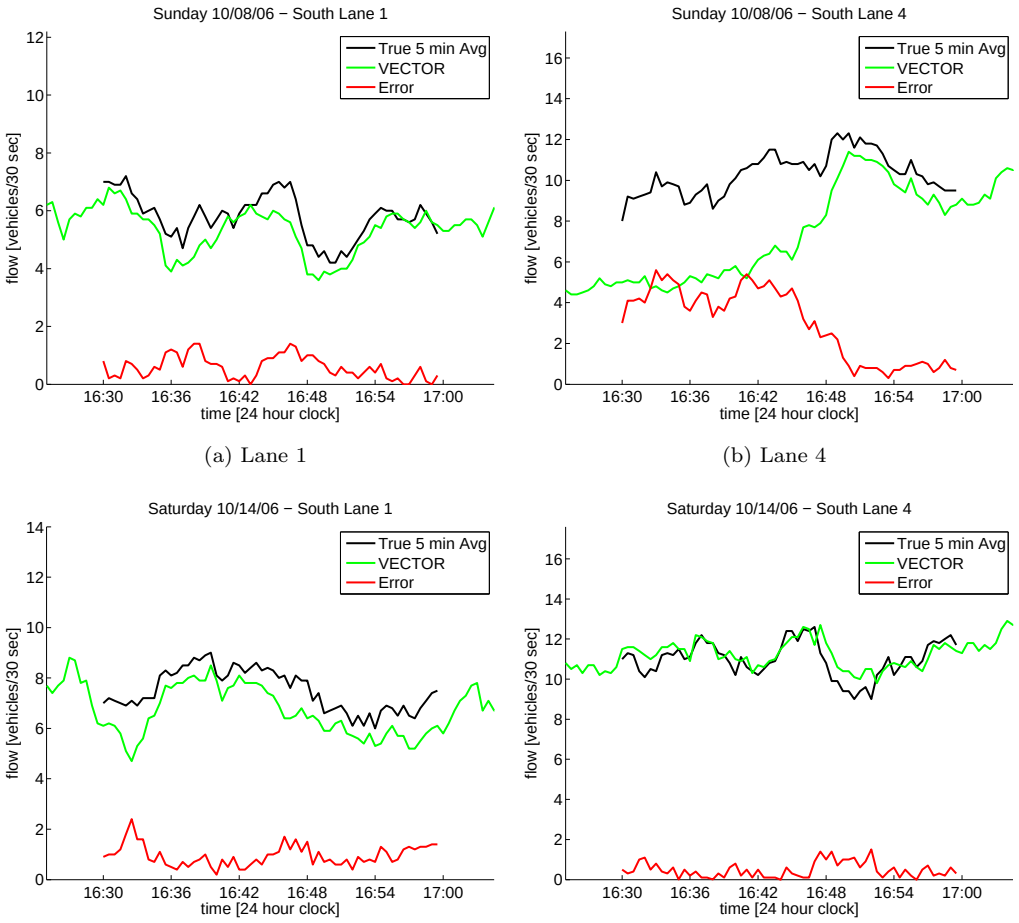
	Lane 1	Lane 2	Lane 3	Lane 4	Total
South	98.7	100	96.2	97.6	98.0
North	100	91.7	84.4	94.6	93.0

5.5.3 Trajectory Pattern Analysis

The following results can not be obtained using loop detectors since they are spot sensors and do not provide spatial coverage of a highway link. [REF why lane changes are good to know - emissions, congestion, etc... traffic microscopic models coifman itsc2006, lane changes for congestion kamijo, kang and chang itsc2004] Lane Changes, Abnormalities.

As indicated by Table 5.8, the trajectory learning technique works well for the straight lanes of the highway even at a long distance ($>$ 200 ft) under perspective distortion. In Fig. 5.20 the number of lane changes and abnormalities over the course of a day are recorded. A comparison with the Monday flow in Fig. 5.18c shows that there are increased lane changes during the higher flow times. This might be expected because more cars on the road would indicate a higher need or desire to maneuver into a more freely flowing lane. Fig. 5.21 highlights a few trajectories that were marked as abnormal.

24 hour test is a monday so make connection between lane change rate and increased flow on Fig. 5.18c.



(a) Lane 1

(b) Lane 4

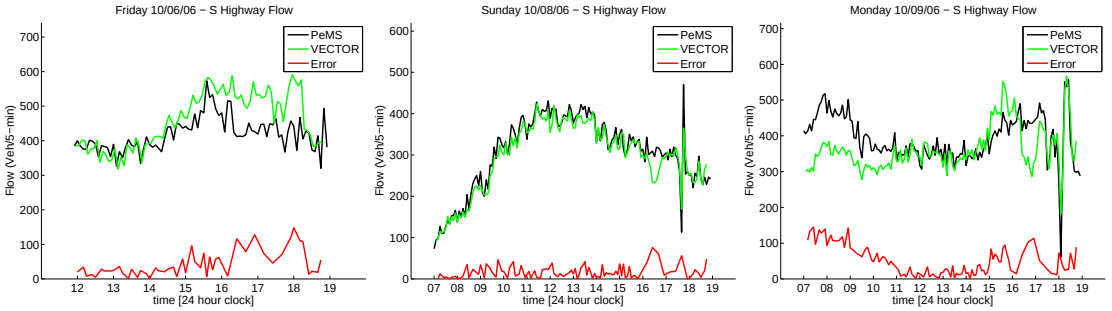
(c) Lane 1

(d) Lane 4

Figure 5.17: True lane flows, 5 minute average truth, and VECTOR module flow comparison.

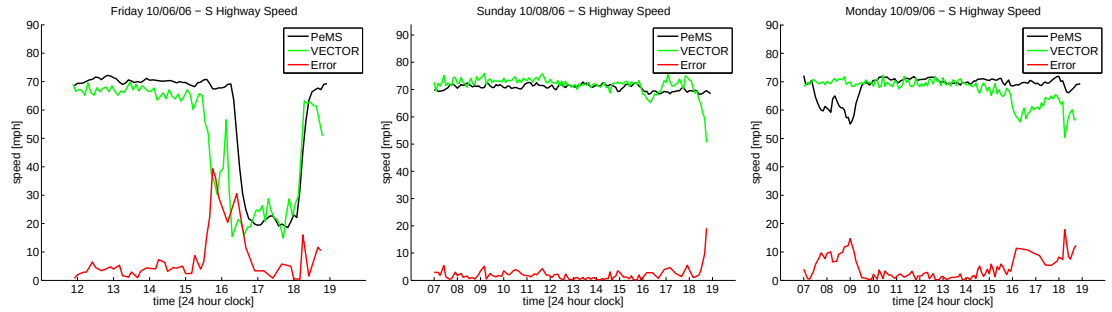
5.6 Concluding Remarks

A visual activity analysis scheme based on tracking was developed for live highway monitoring and activity analysis. The VECTOR system adds real-time situational awareness to visual monitoring by leveraging the recurrent patterns of highway motion to automatically build a lane description and high level activity models. VECTOR is able to use a single camera to extract highway performance measures crucial for traffic management in both directions of a busy link. It adds In addition to standard loop type measurements, the visual system is able to provide a more complete view of usage by categorizing the types of vehicles on the road, measuring link utilization and efficiency in real-time, and recognizing potentially dangerous driving based on historical speed profiles. Finally, by studying and understanding trajectory patterns on the highway, spatial analysis, not possible using current spot sensors, enable the detection and count of lane changes and abnormal driving behavior.



(a) Friday 10/06/06 Flow (b) Sunday 10/08/06 Flow (c) Monday 10/09/06 Flow

Figure 5.18: Comparison between PeMS data and the Flow Analysis module shows strong agreement in flow numbers.



(a) Friday 10/06/06 Speed (b) Sunday 10/08/06 Speed (c) Monday 10/09/06 Speed

Figure 5.19: Comparison between PeMS data and the Flow Analysis module for speed statistics.

Acknowledgements

Chapter 5 is in part a reprint of material that appears in the IEEE Transactions on Intelligent Transportation Systems, 2008, the Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2007, the Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), 2006, the Proceedings of ITSC, 2006, all by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

The support for this work came from the NSF-Bridge grant, ITR, and TSWG. Special thanks goes to UC Discovery for allowing the design, deployment, and continuous operation of the unique DIVA testbed at UCSD.

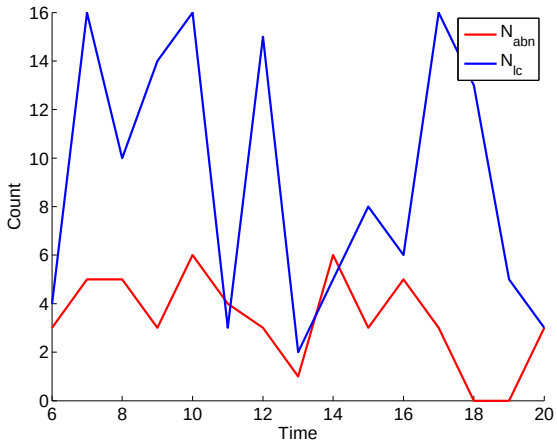


Figure 5.20: Highway trajectory pattern analysis shows increased lane changes during the hours of higher flow.

(a) (b) (c)

Figure 5.21: Highway I5 abnormalities. (a) Vehicle on shoulder during entrance. (b) Vehicle slows to exit in northbound direction on top of image. (c) Truck falsely identified as abnormal because detection centroid is high and appears to be on the side of the road.

Chapter 6

Wide-Area Contextual Awareness

The previous chapters presented methods to analyze a single infrastructure mounted camera in an unsupervised manner. This chapter develops a unifying framework that provides spatial context and awareness over large areas and multiple (different) sensors. The following framework provides the first steps toward wide area coverage and hints at safety applications that utilize communication between infrastructure and independent agents traveling around the coverage area.

6.1 Motivation

Intelligent monitoring of environments has progressed rapidly in the past 10 years [26]. Major technological advancements have pushed the field toward ever more complex environments. The decreased price of video cameras, a primary sensor for surveillance applications, along with improved quality has enabled the use of multiple cameras in more varied spaces. Vast amounts of generated data can be transmitted efficiently because of high quality video compression techniques and improved wireless communication which facilitates flexible setup and configuration. And most importantly, the research community has made great strides in providing intelligence to these spaces. Low level problems such as object detection and tracking are possible in real-time, making common surveillance tasks, such as monitoring a sensitive area for unauthorized entry, straightforward. Intelligent monitoring now seeks to provide situational awareness for semantically meaningful understanding of environment activity.

A key to accurately understanding an environment is to incorporate the needs of the user of the monitoring system. A human must be included in the analysis loop for critical decisions because these decisions must be based on a good understanding of the environment and the monitoring situation. Unfortunately, due to vasts amounts of streaming information, limited attention, and distributed awareness, a human operator can not accurately monitor large areas and

networks effectively. Automatic computational techniques are vital for the monitoring process in order to highlight and guide user attention to relevant areas to relieve tedious concentration on non-critical information. The challenge is to distill the volumes of monitoring information into a manageable quantity and present it to a user in such a way that appropriate decisions can be made promptly (in sufficient amount of time).

This chapter presents a surveillance and monitoring system called CANVAS. CANVAS is a Contextual Activity Notification Visualization Analysis System that spatially integrates distributed sensors. It is used to develop advanced monitoring techniques, integrate cameras and GPS enabled devices, and centralize information [115]. It provides a flexible backbone which allows improvements to vision algorithms while providing a seamless visualization interface. The visualization provides a user with environmental context for the distributed analysis modules in a customizable web interface.

6.1.1 Questions to Address

The following chapter tries to answer the following questions:

- How can multiple distributed sensors be combined to provide larger context?
- How can differing sensors and information sources be intuitively combined?
- How can monitoring analysis and results be presented in a usable fashion?

6.1.2 Chapter Summary

We developed the Contextual Activity Notification Visualization Analysis System (CANVAS) to provide large area environment coverage and broad situational awareness capabilities. The system architecture provides a general framework to collect data from a wide range of sensors, specifically infrastructure mounted cameras and the LISA vehicular testbeds, learn computational models of activity for live analysis, and provides an online user visualization interface. The development of CANVAS has led to the following observations:

- Spatially distributed sensors can be naturally combined based on their relative position. A larger environmental context is built through the combination of individual sensor scope.
- The use of the global positioning system (GPS) abstracts sensors type, *e.g.* infrastructure cameras, vehicles, or smart, in favor of measurements localization. By providing a transformation between the sensor space and GPS coordinates, arbitrary information sources can be fused for contextual understanding.
- With the use of GPS localization, a map based presentation of the monitoring environment along with contextual cues provides an intuitive interface for user interaction and higher level situational analysis.

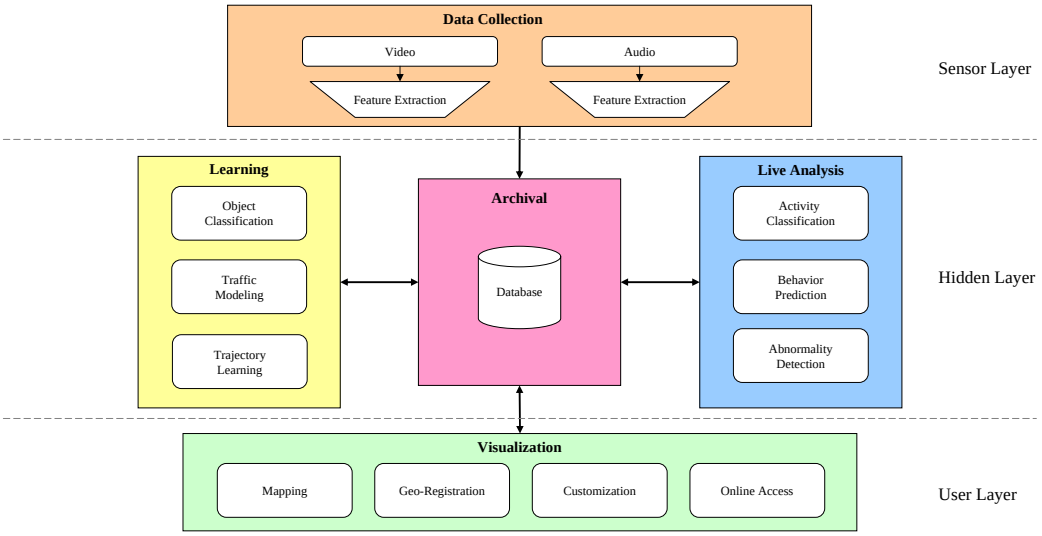


Figure 6.1: CANVAS Monitoring Diagram. The monitoring framework relies on a layer of physical infrastructure which includes cameras and other sensors. The raw sensory data is archived in a database for retrieval. The notification layer relies on a Visualization module to provide needed data in real-time whenever it is required. User inputs are able to customize and modify results. A hidden layer connects the physical devices to the visualization. This hidden layer incorporates the Analysis and associated Learning modules to provide contextual information to a user.

Initial CANVAS use has shown promise as a tool for developing advanced safety systems that utilize integrated sensing from infrastructure and GPS enabled devices.

6.2 System Description, Framework and Functionalities

The block diagram in Fig. 6.1 depicts the major components of CANVAS. The central goal of the system is to have ubiquitous access. This is reflected by the **Archival** block located in the center of the diagram. A database collects and stores data which is accessible through a standard internet connection for quick retrieval. Most of the database storage is devoted to **Data Collection** from connected sensors. Any number of sensors can be hooked into the database. Typical sensors are video and audio devices which each have specialized data extraction techniques, such as position estimates via tracking or object descriptors. The archive data is utilized to train computational modules in the **Learning** block. Example modules can distinguish different types of objects, *e.g.* distinguish pedestrians from vehicles, model highway traffic flow, and compactly represent activity through trajectory learning. The models are archived and used for **Live Analysis** where current sensor readings are utilized in conjunction with the trained models to describe the current state of the scene. The results of live analysis may be wired back into the database as added supplementary features, *e.g.* a trajectory has position and velocity

as well as a description of what type of object it is. Finally, the database contents are made available to a user through a **Visualization** module. A user is able to query the database to retrieve only relevant information and have the display updated in real-time.

Notice the modules (besides **Data Collection**) all pass informations in both directions from the Archival block. This gives the system the ability to dynamically change over time. Models developed through learning techniques can query more recent data in order to update and refine results which in turn modifies the output of live analysis. Modification is even possible through the **Visualization** module. In this case, the user is able to customize results in order to present the most relevant information for his task. These goals of the end user can help dictate the types of analysis that are necessary.

6.3 Information Archival

The heart of CANVAS is the database archival system. A MySQL relational database system is implemented to provide access to organized information tables to multiple users. The widespread use of MySQL has led to the development of many libraries to connect with the database from different programming languages and operating systems. This operational flexibility allows virtually any machine with a network connection to communicate with the database and access its data.

The main goal of the archival block is to timestamp and store sensor data which provides measurements on the state of the monitored world. As the database is updated, a historical context emerges which is necessary for accurate scene understanding. The centralized database allows for a fluid design because it can grow and adapt to new information types and requests as necessary. Training databases, used for learning, can be separated and maintained as subsets of the full database. New information and measurement types can be included with the addition of new sensors or computational modules. This adaptation is necessary for long-term use because monitoring needs can change over time.

The database is split into three main partitions, *sensors*, *models*, and *data*. The first partition holds information about all the connected *sensors*. Each camera sensor is denoted by its type (ptz or omni), location (latitude and longitude), and information for mapping (ptz setting and conversion from image to world coordinates). When new cameras are installed they are quickly integrated into the CANVAS system by including this sensor information. The *model* partition maintains the learning results utilized during live analysis. This partition denotes the model functionality and the parameters necessary for analysis. The last database partition deals with the raw sensor *data*.

A set of secondary databases are populated by video processing for use by the learning modules. The *measurement* database holds information describing the appearance of each detected object for type classification. Tracking information, including location, speed, and ac-

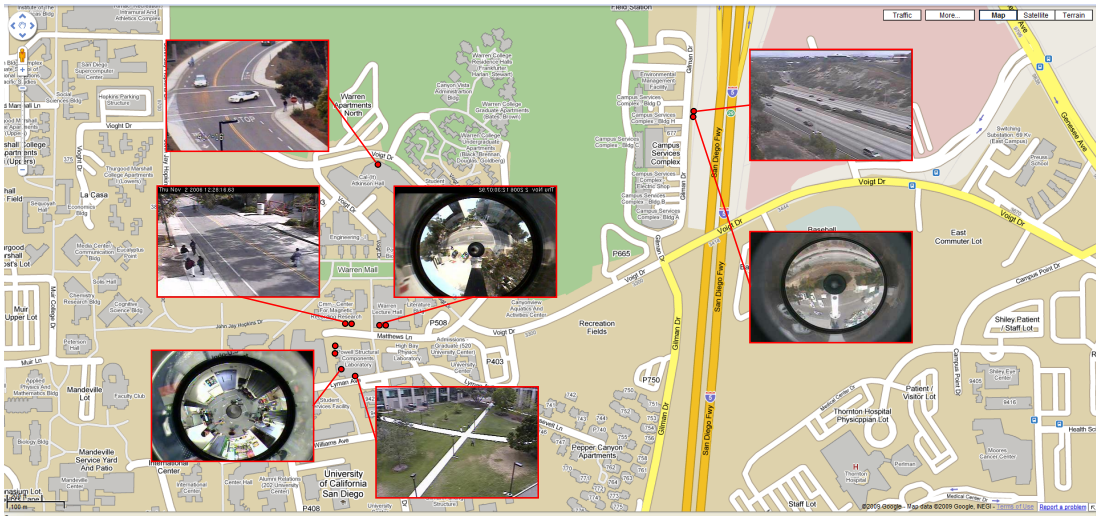


Figure 6.2: UCSD video network. A network of video cameras is situated around campus to provide coverage of different environments. Both rectilinear PTZ as well as omni-directional cameras are used to monitor highway vehicle traffic and the close interactions of people and vehicles on campus.

celeration, are stored in the *tracks* database for trajectory learning. The traffic modeling module relies on information stored in the *highway statistics* database which includes vehicle flow, density, and speed logged every 30 seconds. Finally, the *live* database is automatically updated using current data to provide information for visualization.

6.4 Data Collection and Sensors

The **Data Collection** front end provides all the meaningful and useful signals for CANVAS. All the low level data generation and extraction happens within this block. Sensor specific filters are designed to extract measurements or features from raw sensors. Some filters are quite simple and merely pass the raw measurement onto the database, *e.g.* inductive loop sensors, while more complicated filters require processing *e.g.* tracking for motion description and measurements of object size and shape.

Video cameras are the primary sensors in use for this work. Fig. 6.2 shows a map of UCSD along with a few of the many camera nodes situated around campus. A variety of environments, both indoor to outdoor, as well as different coverage and different objects of interest are present. Using the principle of distributed interactive video arrays (DIVA) [115], monitoring of highway traffic along Interstate 5, human/vehicle interactions on campus roads, and people indoors is performed using both pan-tilt-zoom (PTZ) controllable and wide area covering omni-directional cameras. The networked cameras stream video for remote processing

while non-streaming cameras require a local machine to capture and send analysis data along a network link.

Ancillary sensors CANVAS currently considered are GPS enabled devices. The popularity of smart phones provides location information of users. The LISA testbeds are equipped with GPS receivers and it provides tagged vehicle data. Driving parameters such as speed are logged according to road location and uploaded after a drive. With mobile internet connectivity, these measurements could be streamed in real-time.

Together the positions from infrastructure and mobile devices provide the raw data for environment monitoring and understanding. Logging the GPS coordinates provides trajectories that can be examined for activity awareness.

6.5 Learning and Analysis

Although the **Learning** module usually operates as an offline process and **Analysis** is needed in real-time, the two modules are very closely linked. Live analysis relies on the learned models to make sense of sensor data and understand the monitoring scene. This section describes a number of learning techniques and the questions that can be answered during live analysis using the *model* database.

For each Learning module, a training database is created by extracting the needed information from the **Archival** database. A training database is accumulated by collecting the appropriate signals over a sufficient time period. Analysis models can be created by applying learning algorithms on the compiled data. Database maintenance updates training data for adaptive models which more accurately represent the current configuration of the monitoring scene.

Analysis modules are essential for effective monitoring because it eases the cognitive load of a human observer. Multiple analysis tasks can be run in parallel on multiple video feeds which is something quite difficult for a human.

6.5.1 Object Classification

Classification, as described in Chapter 5, identifies the the type of detected object based on its visual signature. Each camera has an associated transformation matrix, P_{LDA} (5.1), a list containing the names of object types, and a nearest neighbor database for each type. Using the 2001 US DOT National Household Travel Survey for guidance, the 7 most often occurring vehicle types {Sedan, Pickup, SUV, Van, Semi, Truck, Bike} are identified in highway streams. This detailed real-time fleet composition is a missing management component essential for estimating emissions or infrastructure load assessment [118]. On campus, detected objects are marked as either {car, pedestrian, biker, skateboarder, or a group of people}. This classification helps with

criticality assessment of situations when vehicles and people interact in close proximity.

6.5.2 Traffic Modeling

Chapter 5 also demonstrated how effective traffic management relied on the knowledge of the location and number of vehicles in a transportation network. It showed how a single infrastructure camera could effectively monitor a highway link to extract the essential lane level measures of flow ($\frac{\#vehicles}{time}$), density ($\frac{\#vehicles}{distance}$), and speed (MPH). These traffic parameters are stored into the database where they can be aggregated over time to build the daily speed profiles, Section 5.4.3.

6.5.3 Trajectory Learning

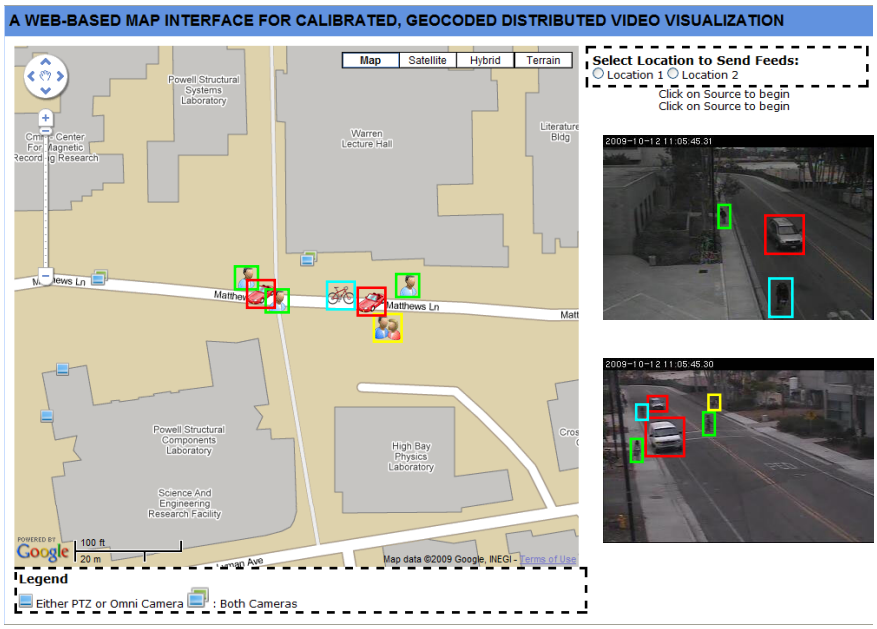
Recently, one of the most popular techniques for automated surveillance and monitoring is trajectory learning (see Chapter 4). This technique makes it easier to monitor larger video networks because activity models are learned automatically without need for manual specification. Trajectories, consisting of location and speed, are stored in the *model* database to build the activity models $\lambda = (\pi_0, A, B)$. Each of the Q Gaussian dynamics states are defined by its two parameters, mean μ_q and covariance Σ_q.

During live analysis, at each time instant an activity prediction is made to augment the trajectory description. At the end of tracking, a track level activity label is assigned along with the abnormality state. These features incorporate automatic trajectory learning into the contextual activity environment.

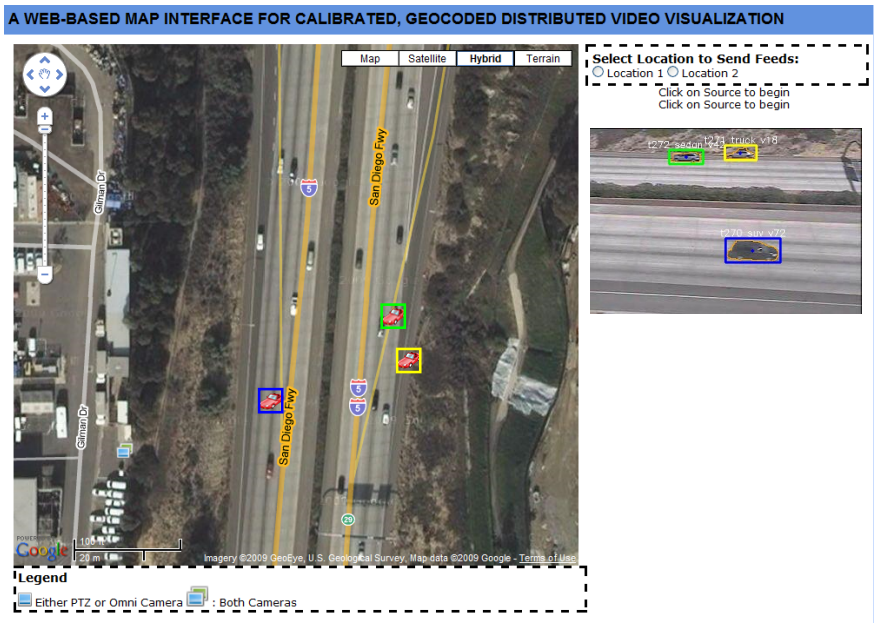
6.6 Visualization

The main goal of the **Visualization** block is to provide a common environment for the display of real-time information and live analysis. The visualization environment presents an immersive and interactive display that preserves the context of the information sources. Simultaneous access to different data sources, with control of the area, scale, and information of interest, while still preserving surrounding environmental context, enables a cohesive picture that provides the user a complete situational awareness [44]. Awareness is realized through functional display layers built for each **Analysis** module where each additional visualization layer provides a more detailed picture of the monitoring state.

While providing expansive environmental context, care is taken to avoid distractions that can detract from the principle monitoring task [68]. Instead of overloading the display with large amounts of annotations, information is distilled and visualized through the use of icons and avatars (examples in Fig. 6.3). The filtered view of information limits cognitive load and helps focus attention on the locations most likely to be interesting through automatic highlighting [39].



(a)



(b)

Figure 6.3: CANVAS Visualization Page. (a) A campus street is monitored using two slightly overlapping cameras. The output of object classification and tracking is marked on the map. Icons indicate the object type and are placed on the map based on camera geo-registration information to map image coordinates to GPS. (b) Environmental context is presented using an aerial image of the highway. The detected vehicles are marked with car icons which appear in the different road lanes.

The **Visualization** block indicates the location of sensors with respect to one another, gives access to raw video feeds, presents pertinent analysis results, and provides a user interface to navigate, query, and customize the display.

6.6.1 Mapping

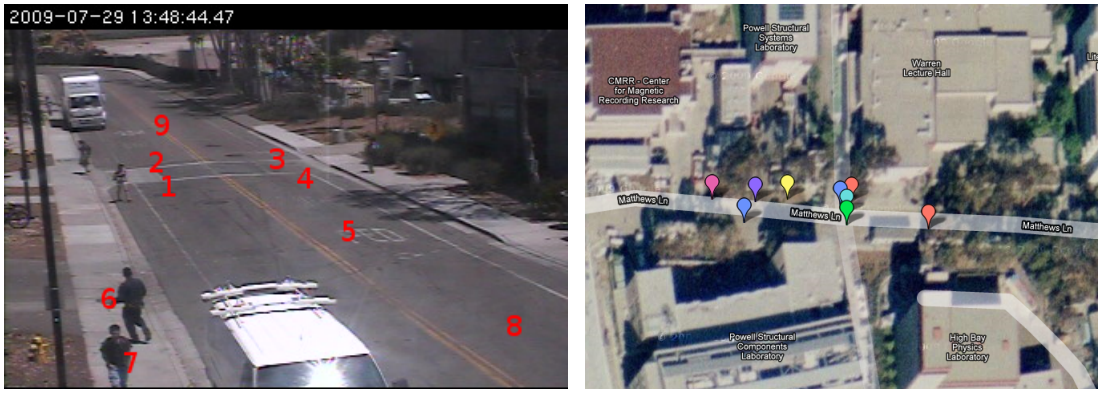
Although the world is three dimensional, information is not contextualized in a 3D environment because it limits usage to locations with complete 3D graphic models [18]. Instead, a 2D map representation of the environment is utilized. A map provides surrounding environmental context which assists comprehension of spatial relationships between objects, increasing situational awareness [33]. The user display is built using the Google Maps API because it is a familiar interface (often used for directions) and its wide coverage makes it applicable to most outdoor locations. The environment context is available through different modalities such as aerial imagery or through geographical information system (GIS) type layers depicting structures and areas of interest. The API also supports user interaction with the use of draggable markers and other line drawing tools.

6.6.2 Geo-Registration

Proper analysis visualization requires proper output alignment with the map coordinates. Sensor coordinates must be transformed into GPS latitude and longitude coordinates in a process called geo-registration. Geo-registration requires calibration between the sensor space and the map space. Simple spot sensor, such as inductive loops, only acquire measurements from a single location which makes the calibration straight forward. The sensor output can be overlayed on the GPS coordinate of the sensor location. It is more difficult to calibrate spatial sensors because of their coverage area. In this case, it is necessary to transform all points in the sensor FOV into a corresponding map location.

In order to geo-register a camera, the locations of objects in the image plane and the corresponding latitudes and longitudes on the map need to be known. This is a multi view registration problem. One view of the scene is generated by the camera and the second view is the map (satellite image). Typically, the epipolar constraint can be used to determine the relative pose between the two camera and solve for the transformation between views. But, since the map is only a 2D representation of the world, full three dimensional mappings are not required. The transformation between the map GPS coordinates and image coordinates reduces to a mapping between 2D planes. This calibration is learned a homography transformation, H, mapping the image pixel locations on the ground plane (*e.g.* the road) x_{im} to its corresponding latitude and longitude coordinates on the map X_{gps}.

$$X_{gps} = Hx_{im} \tag{6.1}$$



(a) (b)

Figure 6.4: Geo-registration calibration with GPS coordinates obtained using an iPhone. (a) Image locations of ground plane calibration points. (b) Google maps satellite image with GPS location of calibration points.

The homography can be found for a camera by using a GPS receiver to collect the GPS coordinates of specific image location. H can be estimated in a least squares sense using at least 4 corresponding coordinates by singular value decomposition using the four-point algorithm.[75].

Corresponding points between the map and video were obtained by walking on the street an iPhone as a GPS receiver while being recorded by the camera. GPS coordinates were extracted at specific points by remaining still until the GPS reading stabilized. The corresponding image point was manually marked at the point of contact between road and feet. Fig. 6.4a shows the camera view of Matthews Lane on campus and a satellite image map with pins for the corresponding latitude and longitude points is in Fig. 6.4b. The GPS coordinates obtained from the iPhone do not exactly match the Google road map and only cover a small strip of the map image. The coarse resolution of the GPS receiver coupled with this narrow view cause some numerical instability during the mapping from image to map coordinates. The raw GPS values, when streamed, provide another source of trajectory data.

6.6.3 Customization

Another design principle of the **Visualization** module is to present information to a user only when needed. Complex environments are filled with activities and events that may be outside of relevance for most users. User specific information is provided in order to answer the most relevant questions. An example of this type of design is personalized traffic reports to generate travel estimates given a user specific commute route [24].

The design paradigm called for a simple interface that would abstract the database connection and communication from a user. The display customization is available through

buttons which overlay results onto the map. In this way the appropriate SQL commands are generated by the webpage rather than by the operator, removing need for training.

The user interface presents clickable controls to select camera feeds, change environmental context (map layer in Fig. 6.3a or aerial imagery in Fig. 6.3b), and display analysis results.

There are 2 major user customization/selection modalities: video feed selection and map layers. Video feed selection is used to initialize raw video streams from up to 2 live feeds. The map layers provide the common map based visualization of results. Map scale and navigation is controlled through the Google Map API and computational layers are created for the analysis modules (traffic flow, classification results, and trajectory analysis). A layer is created for each analysis type and camera pair. Figure 6.3 shows two different classification layers. A classification layer denoting pedestrians and vehicles on campus is shown in Fig. 6.3a while Fig. 6.3b shows vehicle tracking.

Further customization is possible with advanced users who design specialized computational layers. Similar to GIS software, a user would define the queries necessary to extract pertinent information as well as define any visualization layers. An example is a zone alert to monitor a sensitive region. The advanced user would specify an polygon in the image and search the *tracks* database for objects within this region.

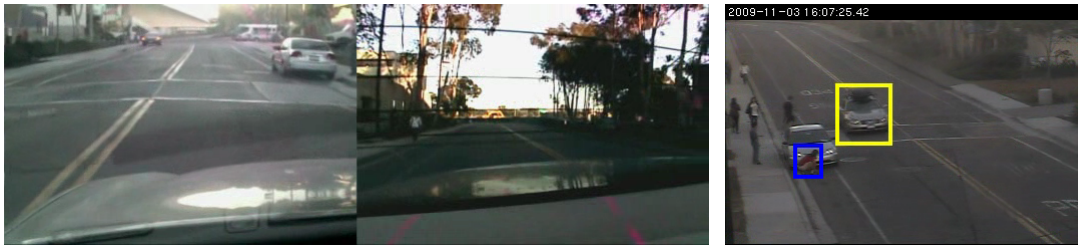
6.6.4 Online/Mobile Access

The final goal of the **Visualization** block is to provide access to information wherever it is needed through remote access. This allows more convenient monitoring because it does not have to occur on site. The visualization was built on web technology to be platform independent and portable relieving the need to design or compile different versions of the code for specific platforms.

Besides remote availability, design in web based technologies make it possible to realize mobile/portable access and help fulfill the promise of a a ubiquitous age where the rapid developments in mobile handset and network technologies can bring customized management services to all people [103]. The increasing popularity of mobile applications on cellular phones indicates the desire for instant connectivity and functionality.

6.7 Wide Area Activity Analysis

By exploring the environment with the map-based representation, activities can be understood within a larger spatial context. The relationships between cameras and monitored objects are contained in a single view to abstract the particulars of a specific location. CANVAS can be used to provide information within its sensor coverage. In Fig. 6.5a, a campus road is



(a) (b)

Figure 6.5: (a) A driver's awareness is limited to what can be seen in front and behind the vehicle. (b) Using infrastructure, situational awareness can be transferred to the driver. The car is warned of the occluded pedestrian tying his shoe on the left side of the road

Figure 6.6: A GPS enabled mobile device can be detected through visual occlusion for relaying appropriate safety warnings. The infrastructure knows of the impeding crosswalk situation between the car and pedestrian and messages can be sent to both the driver and mobile device.

shown as seen from within a vehicle. The driver view is limited to the front and rear windows but with help from infrastructure cameras, the pedestrian behind the vehicle is detected and a warning (yellow bounding box) can be relayed to the driver upon approach (Fig. 6.5b).

The integration of GPS into mobile devices provides a medium for understanding behavior. Using GPS enabled phones, a new stream of trajectory information can be acquired which supplements infrastructure. A pedestrian is tracked through occlusion in Fig. 6.6 and an alert is sent to the phone warning of the oncoming vehicle. The mobile devices provide a level coverage not feasible using infrastructure. Fig. 6.7 shows the route of a probe vehicle. The vehicle enables coverage well beyond the extent of the campus network, yet still can be seamlessly integrated through the map interface. The trajectories obtained from mobile devices and automobiles help complete the environment behavior and activity picture [3]. Combining visual cues along with GPS devices can improve tracking [70] and provide a better understanding of the monitoring situation.

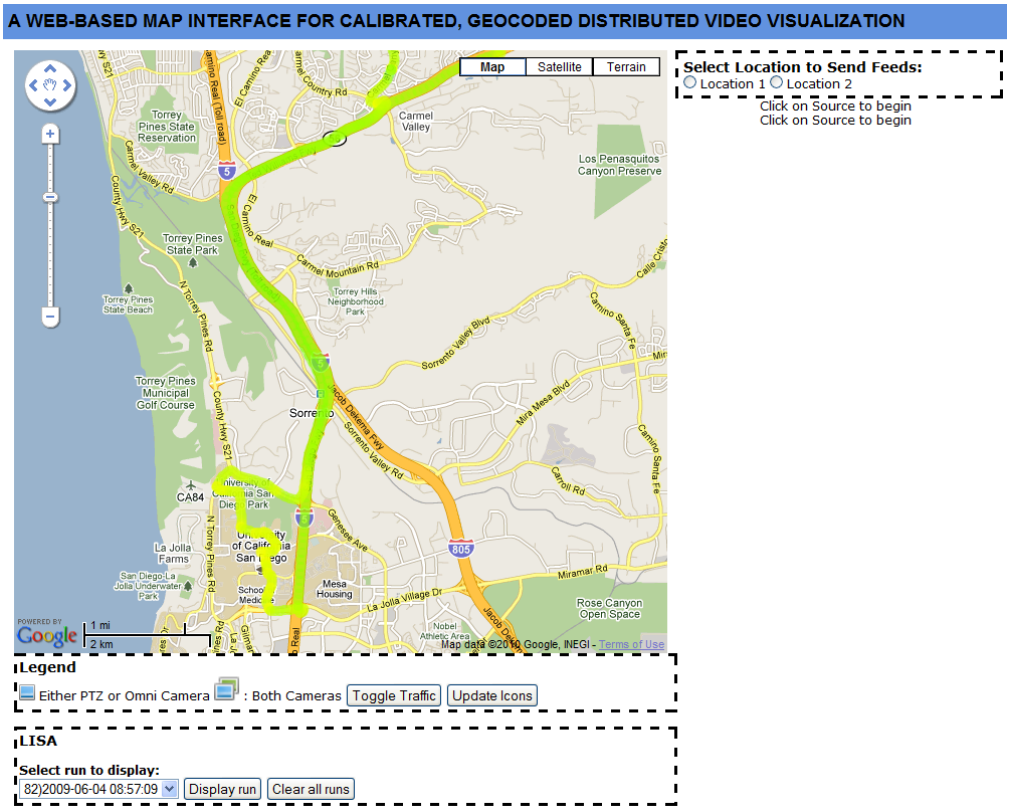


Figure 6.7: GPS enabled vehicles and devices are seamlessly integrated into the map. A recorded route taken by a GPS equipped vehicle is overlayed on the map. The route is color coded based on the speed of the automobile with respect to speed limits.

6.8 Future Directions

CANVAS provides a unified framework for improved site monitoring. It presents exciting opportunities to study large areas for re-identification and handoff between sensor coverage zones. Research into infrastructure-device communication for safety will greatly benefit from this type of platform since it will be possible to "see" what the infrastructure knowns and where the device resides. This will enable augmented surround awareness with traffic safety messages and information from infrastructure.

6.9 Concluding Remarks

This chapter presents a management system for monitoring complex environments. The focus is on building an upgradeable framework for simple user interaction through an accessible visualization. Rather than present just raw sensor data from the physical world, visualization layers are introduced to abstract the internals of monitoring algorithms and provide a clean

consumable computational output. Advanced algorithms for understanding scene activity can be designed and integrated into the visualization system to help highlight object or events of interest. This ultimately improves the effectiveness of the monitoring by focusing attention and presenting only the most relevant information. By building the visualization with web technology, the information is available to monitor the environmental situation anywhere at anytime.

Acknowledegments

Chapter 6 is in part a reprint of material that will appear in the IEEE Intelligent Systems Magazine, 2010, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of these papers.

This research is supported by research grants from the University of California Transportation Center (US Department of Transportation Center of Excellence) and the UC Discovery Program. The authors would like to acknowledge the contributions of Mr. Diego Villaseñor of UC Riverside under the UC LEADS research scholar program for the web visualization, Mr. Jeff Ploetner for the MySQL database setup, and other colleagues of from the CVRR Laboratory.

Chapter 7

Automobile Surround Maneuvers

In contrast with VECTOR and CANVAS, this chapter examines activity not from an infrastructure mounted camera but a completely different domain, a moving platform. A sensor equipped automobile, designed for intelligent driver support, observes the behaviors of surrounding drivers to learn cues that signal impending maneuvers. With early prediction of the maneuvers of surrounding vehicles, a driver is able to contextualize the driving environment and better prepare for dangerous situations.

7.1 Motivation

The automobile is ubiquitous in modern society but the access and mobility it grants comes at a staggering price. According to the 2009 World Health Organization's Global Status Report on Road Safety [128], road crashes cause over 1.27 million fatalities a year and between 20 and 50 million non-fatal injuries. It is estimated that this will increase to 2.4 million fatalities a year by 2030, ranking road traffic injuries as the fifth leading cause of death. While not as bad as less developed countries, there were still 2.5 million injury accidents and 41 thousand motor vehicle related fatalities in the United States in 2008 [117]. These alarming statistics highlight the critical need for advanced vehicle safety systems.

The US National Highway Traffic Safety Administration (NHTSA) report Traffic Safety Facts 2006 [121] found in passenger car crashes that the initial point of impact was the front 49.6%, either the right or left side 27.8% (left 14.2, right 13.6), or in the rear 21.2% of the time (see Fig. 7.1a). Therefore, heightened surround awareness can directly affect safe driving and maneuvering of an automobile. Successful systems currently in vehicles are active cruise control (ACC), which adapts vehicle speed to maintain a safe following distance, and lane assist, which can keep a vehicle in a lane or warn a driver when drifting. Unfortunately, such advanced systems exist only in the front of vehicles. The rest of the surround is usually covered by more basic

detectors such as back-up cameras, sonar parking assist, and radar/vision blind spot warnings which just give an indication or an impeding object. This work seeks to aid the development of more advanced surround based safety systems which can give advance warning of impending situations. By predicting the behavior of surrounding vehicles, a driver is able to prepare earlier to ensure safer conditions.

7.1.1 Questions to Address

Examining activity from a moving platform with real safety implications raises the following important questions:

- Is it possible to predict the driving maneuvers of surrounding vehicles without explicit motion models?
- Can meaningful driving behaviors be automatically extracted through trajectory pattern observation?
- How can the interactions between vehicles and the environment be incorporated into the learning procedure for better prediction?

7.1.2 Chapter Summary

Utilizing the same trajectory analysis framework initially designed for video monitoring with static infrastructure cameras, the driving patterns of vehicles surrounding an automobile were automatically extracted. This introductory exploration into unsupervised trajectory analysis for driver applications answered the following:

- It is possible to predict maneuvers even without considering complex motion models and parameter measurement by learning the typical patterns a driver would encounter during natural driving.
- Analysis of a vehicle trajectories demonstrated that meaningful patterns can be extracted from data in an unsupervised fashion. Actions corresponding to passing were found as well as an estimate of typical following distance. But, unlike for infrastructure monitoring, high variability between maneuvers created many uninterpretable patterns.
- Currently, no interactions between vehicles in considered when analyzing maneuvers. This limits predictive power because, in addition to driver intentions and style, it is clear that the road geometry and traffic configuration play a vital role in driver decision making. In this framework, maneuvers are tied to specific spatial regions. Instead, a technique to provide predictions for every surround vehicle regardless of location is preferred.

This chapter presents the first attempt to observe surrounding driving patterns to automatically build maneuver models rather than fit motion to complex dynamical models. The results indicate that the learned models correspond to real driver behavior and could be used for intent inferencing.

7.2 Related Work

Given the dangerous nature of driving, researchers have been searching for ways to make the driving experience safer, provide greater situational awareness, and promote overall enjoyment and comfort. On-board vehicular safety systems have fallen into two separate categories: passive and active. Passive safety systems act once collision is inevitable to mitigate severity. Successful examples of passive safety examples are seat belts and airbags which engage during impact to alleviate injuries. Current research has trended toward active safety, which aims to prevent accidents (*e.g.* anti-lock breaks). While accident prevention is obviously preferable to mere mitigation, the design of active safety systems is much more challenging because it requires accurate, reliable, and prompt identification of conditions that would lead to an accident in advance to allow time for corrective measures [114].

An active safety system has three major components. The first is a sensing subsystem that is able to extract information about the vehicle, the driver, and the surrounding environment to provide a description of the local dynamic state of the vehicle space. The second subsystem, the contextual processing module, uses the measured dynamic state to infer the criticality of the driving situation and safety state. When safety falls below an acceptable threshold, the final active-safety control subsystem is activated to perform corrective actions to prevent the impending accident.

Environment sensing is a mature area that relies on a number of sensing modalities. RADAR can detect reflections of electromagnetic waves off of surrounding vehicles in a variety of conditions making it a favorite sensor of car manufacturers. Other sensors such as LIDAR, LASER scanners, and ultrasonic sensors are commonly utilized in the research community for surround awareness. Video cameras have gained enormous popularity do to their rich contextual content delivered over a wide field of view, the decrease in high quality optics, the increase in computational power of modern processors which have enabled real-time application, and the active computer vision research community interested in automotive applications. The enormous interest is highlighted by a number of recent survey papers on video based vehicle detection [61, 112, 37] which demonstrate a number of differing methods to accurately sense the surrounding obstacles from moving vehicles. The survey by Kastrinaki *et al.* in 2003 describes video processing techniques for traffic applications and in 2006, Sun *et al.* presented a review of vision based detection [112] for front looking cameras and techniques for top mounted 360° omni-directional cameras was presented by Gandhi and Trivedi [37].

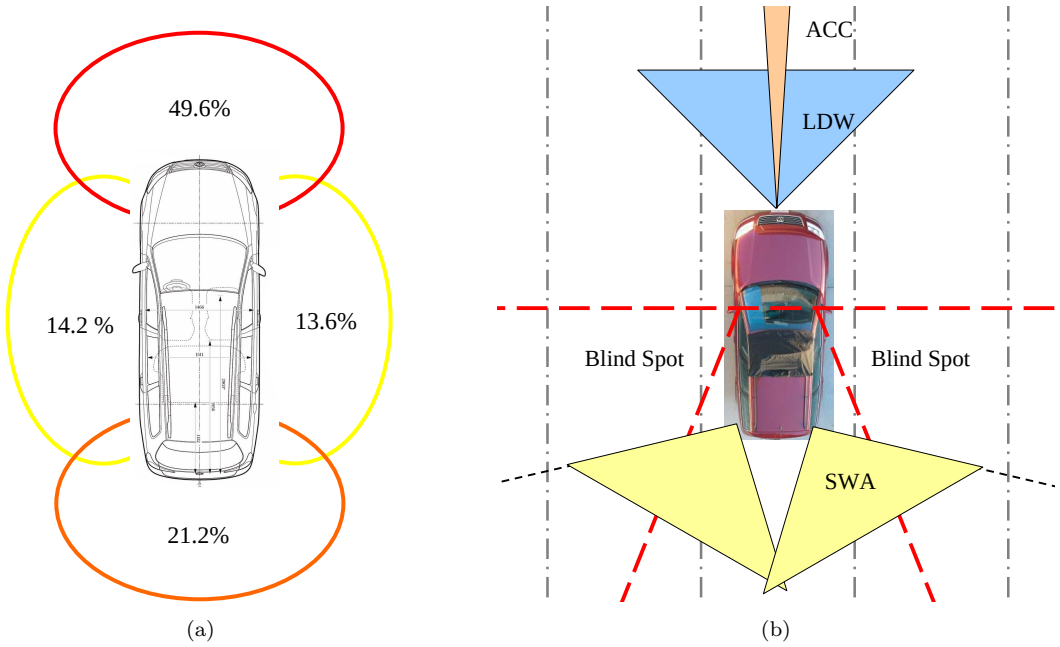


Figure 7.1: (a) Initial impact points of collision according to the 2006 Traffic Safety Facts report conducted by NHTSA. Half of all accidents occur in the surrounding regions that are not normally observed during driving (sides and rear). (b) Sensor configuration and coverage typically available in commercial vehicles today. The ACC system uses a narrow but long-range radar, LDW uses a camera, and SWA uses either cameras or radars. Notice the blind spot (in red) is significantly reduced with the SWA system.

Many current advanced driver assistance systems (ADAS) rely on understanding the intentions of the driver. By understanding what a driver wants to do, the vehicle can be better prepared to make corrective actions when there is deviation from intent. Systems in successful deployment today include active cruise control (ACC), which regulates the time gap (following distance) from a lead vehicle, and lane departure warning (LDW) systems, which use a camera to detect the lanes of the road and identify when the automobile is drifting out of the current lane. In 1997, Liu and Petland [73] proposed a method to infer driver maneuvers based on preparatory control actions including steering and acceleration in a simulator environment. Later researches used explicit modeling of surrounding vehicles [96] to provide a better picture of the driving situation and make threat assessments. Trivedi *et al.* [114] proposed a looking-in and looking-out (LILO) framework to investigate intent prediction. The framework provides a complete understanding of driving context by simultaneously capturing the outside surround environment of the vehicle, the vehicle control and dynamic state through on-board sensors, and finally the internal activities and state of the driver and other cockpit occupants. Integrating all the driving measurements and using machine learning techniques, a driver's intent to brake [77],

turn [23, 105], or change lanes [104, 78, 29, 32, 30] can be assessed seconds in advance.

In contrast to the ego-vehicle intent prediction problem, little surround intention work has been done outside of the autonomous driving field. This same type of prediction used for ego-manuevers must be done for surrounding vehicles to understand what actions they might perform in the near future as well. An active surround safety system could then use the surround predictions to warn a driver of possible dangerous situations for increased awareness. Unfortunately, the LILO framework is not applicable because a driver can only observe a surround vehicle from the outside and does not have any internal indicators for intent. In order to assess dangerous situations, the motion of surrounding vehicles is examined to determine whether a collision is imminent. Common to these surround systems is the use of an explicit motion model to predict future movement [96, 15, 71, 34, 4]. This work intends to learn surround patterns automatically in order to extract typical highway behaviors observed during natural driving. Rather than manually specifying expected behaviors of interest or motion models that govern behavior, a data-driven approach is utilized to learn patterns in an unsupervised fashion from the surround trajectories of detected objects.

7.3 Vehicle Detection

Intelligent driver support systems rely on a number of different sensors to to observe the vehicle surround. Typical sensors in use by manufacturers are RADAR (ACC, SWA), LASER (ACC), and video (SWA). Stereo cameras have recently become very popular because of active research by the vision community. Specialized research vehicles utilize more exotic, very high resolution, sensors such as LIDAR or the Velodyne's 360° scanning laser because price is not an issue. Ultrasonic sensors are used for parking assist technologies but are limited to very short range and low speeds and therefore not applicable during normal driving. While each particular sensor has its own strengths and unique set of requirements for effective use, the end goal is the same, to detect obstacles. Once vehicles are detected they can be tracked to maintain a history of motion.

7.3.1 LISA

The Laboratory for Intelligent and Safe Automobiles (LISA) is a mobile computing platform capable of monitoring the vehicle surround, inside the cockpit, and the state of the vehicle itself. The different test environments provide adaptable experimental testbeds for the evaluation of different sensor combinations and associated safety algorithms. A LISA vehicle is able to synchronously capture and process data from all sensor systems. Data from manufacturer installed onboard sensors is recorded by tapping into the controller area network (CAN) bus, cameras are mounted inside the vehicle to view driver behavior using face, hands, and feet

(a)

(b)

(c)

(d)

(e)

Figure 7.2: LISA video collection

cameras, and the surround is monitored with an array of sensors including monocular cameras, omnidirectional cameras, stereo camera pairs, and radars. Fig. 7.2 provides example video images obtained with the LISA testbeds in different configurations. The ability to reconfigure the sensor coverage enables focus on particular regions around the vehicle. This allows new detection algorithms to be developed which take advantage of the unique properties encountered when looking either in the front, sides, or rear of the vehicle.

7.3.2 Data Collection

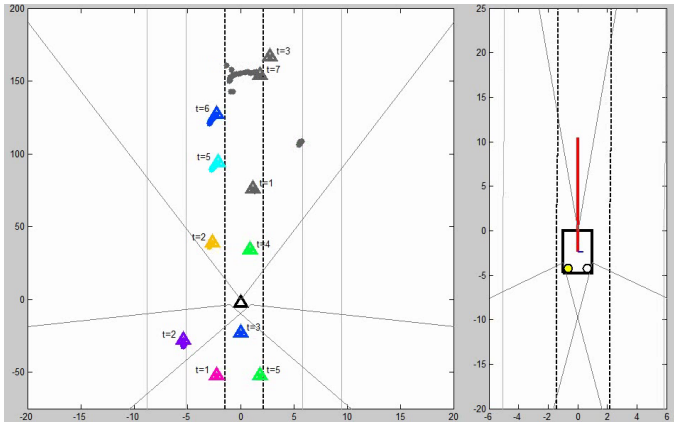
Rather than develop new vision based detection and tracking techniques, this work utilizes sensors that are currently available on production level vehicles. Given the threat level of the different regions of a vehicle (Fig. 7.1a), a LISA vehicle was equipped with a front looking RADAR for ACC, a front looking camera for LDW, and rear RADAR to protect the sides and rear of the vehicle for SWA. The sensor coverage is visualized in Fig. 7.1b. The camera system is used to detect lanes only while the RADAR systems track surrounding vehicles. The narrow field of view of the ACC system creates a large gap with the SWA sensors which leaves the front right and front left of the vehicle uncovered.

Vehicles are detected to generate position and velocity estimates relative to the ego-vehicle. Fig. 7.3 visualizes the trajectory data obtained from the production-level sensors. On the right side of the a subfigure is the obstacle map view which provides a top-down bird's-eye-view of the vehicle surround. At the center of the display, at coordinates (0,0), is an icon indicating the ego-vehicle. Obstacles are inserted around the vehicle in a wide field-of-view which extents laterally 25 meters on each side (ensuring at least 2 adjacent lanes are visible on either side) and longitudinally forward 200 meters and 100 meters in the rear. The locations recorded over last 2 seconds are displayed to show partial vehicle trajectories. The speed of the surround vehicles can be inferred based on the length of the trajectory tail. On the right side of the subfigure, the guide view provides the same top-down view of the surround but at higher resolution. Because of the closer view, the surrounding trajectories are much more detailed making it possible to resolve fine motion variations.

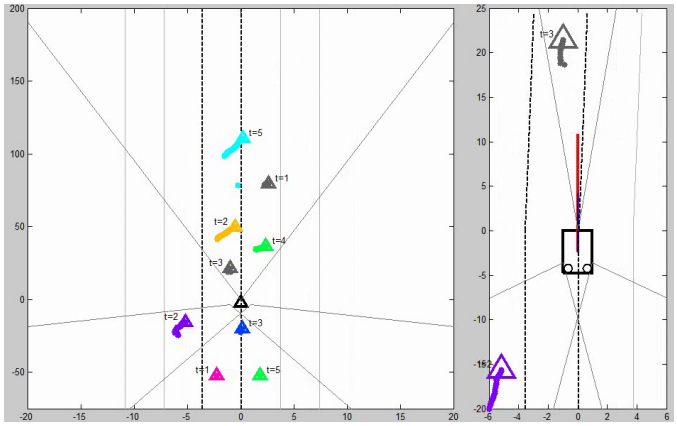
The example in Fig. 7.3 is of a lane change. It is possible to to see the relative motion between the surround vehicles and the ego-vehicle. The common lateral motion to the right of the surround tracks occurs because of the ego-vehicle moving left. Collecting the trajectories extracted through RADAR tracking results in the data shown in Fig. 7.4.

7.4 Trajectory Learning

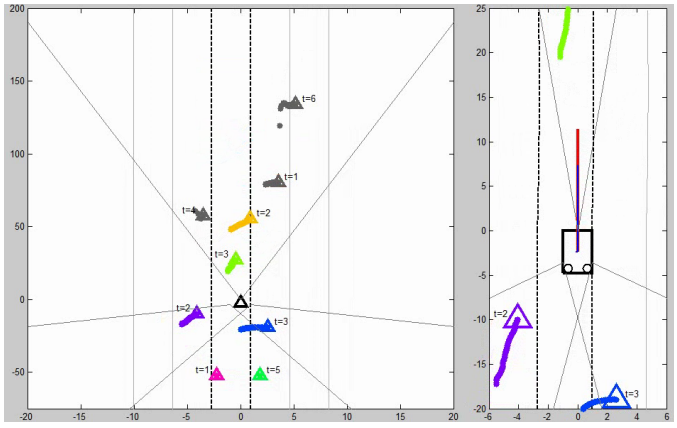
As described in Chapter 4, the main inputs for the learning engine are motion trajectories (Fig. 7.4). A trajectory $F = \{f_1, \ldots, f_T\}$ compactly represents object motion during the time



(a)



(b)



(c)

Figure 7.3: Lane change maneuver. Notice the turn indicator before the lane change and the lane line as the change occurs.

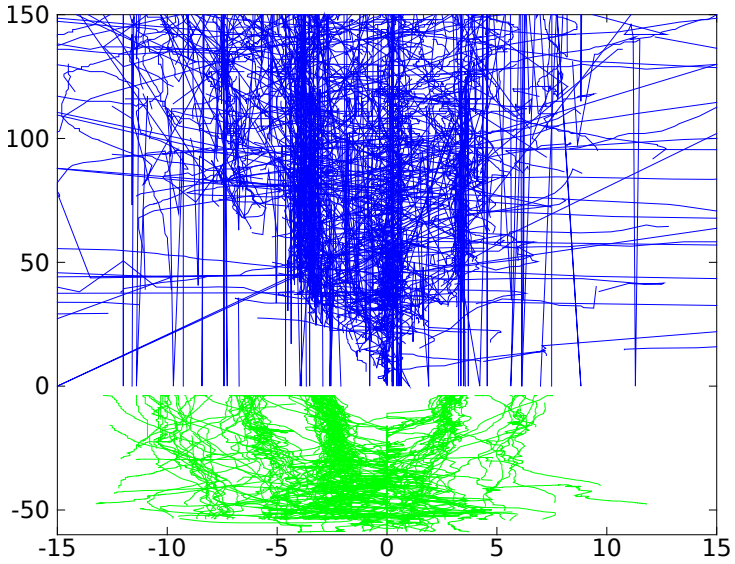


Figure 7.4: Trajectories collected using the front ACC system and the rear SWA system.

T it is observed. The flow vector f_t represents tracked parameters at time t. In this work, $f_t = [x, y, u, v]^T$ represents the xy position and uv velocity in either direction.

7.4.1 Special Considerations for Vehicle Surround

Although the trajectory learning framework was designed to be flexible and generalizable to a variety of scenes, there are a number of characteristics of vehicle surround analysis which differentiates it from infrastructure monitoring. Some of the key differences are expanded below:

- Non-stationary camera - motion is with respect to moving location.
- Potentially long tracks - detection can occur for long periods of time.
- Behavior in segments - may not spatially tied to specific locations.
- Limited surround coverage - do not see everything.
- Limited driver knowledge - need to infer intention without knowing the driver state.
- Time critical - safety concerns.

7.4.2 Learning Modifications

Before applying the trajectory learning machinery, the aforementioned differences between vehicle and infrastructure monitoring needs to be addressed.

- Non-stationary camera

Rather than develop camera based detection schemes, production level RADAR is used for detection. The trajectories use relative motion to relate to the ego-vehicle. Driving maneuvers are defined with respect the the ego-vehicle in a moving reference frame.

- Potentially long tracks

Trajectories are resampled to a fixed length for computational ease and to remove non-informative stationary points. Therefore, tracks correspond to the end-to-end maneuvers of vehicles while in view of the ego-automobile.

Each trajectory F has its velocity information ignored and is spatially resampled to a fixed length L. By resampling, each trajectory has the same number of points allowing straightforward Euclidean distance comparison. Rather than simply subsampling [48] or interpolating points, the resampled representation $\bar{F} = \{\bar{f}_1, \ldots, \bar{f}_L\}$ seeks to evenly distribute points along the trajectory, ensuring the distance between consecutive points is equal

$$d(\bar{f}_l, \bar{f}_{l+1}) \sim \frac{1}{L-1} \sum_{t=1}^{T-1} d(f_t, f_{t+1}) \tag{7.1}$$

$$d(f_i, f_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{7.2}$$

to completely remove dynamic information (as hidden in the sampling rate). This prevents regions of higher sample density from contributing bunches of points in a single area as could occur when there is car-following. Finally, a trajectory vector $F = [x_1, y_1, \ldots, x_L, y_L]$, representing a point in the $\mathbb{R}^{2L}$ route space, is constructed for each of the N training trajectories. Since trajectories are now fixed-length vectors, clustering is performed using a weighted FCM procedure where the weights incorporate the lateral importance of lanes.

- Behavior in segments

This is not addressed and is left for future work.

- Limited surround coverage

Trajectories are examined only within a sensor field of view. This is equivalent to learning for separate cameras.

- Limited driver knowledge

The learning framework infers activity from coarse motion only, no deeper understanding is needed.

- Time critical

By construction, the activity analysis is fast making it acceptable for time critical applications.

In the vehicle domain, trajectories are extracted using RADAR and post-processed by resampling to a fixed length suitable for input into the framework outlined in Chapter 4. The activity models that are eventually learned will correspond to typical driving maneuvers a driver would encounter while driving.

7.5 Experimental Studies

The activity learning scheme is tested on driving data collected from an instrumented vehicle. Vehicles are tracked with respect to the moving platform and represent differential motion. This work only considers natural driving patterns along a highway.

7.5.1 Examining Rear Trajectories

The behavior learning framework was tested on trajectories obtained from the rear of a car. Monitoring of this area is much less prevalent than the front or blind spots event though a high percentage of accidents have the rear as a point of contact. The trajectories do not correspond to a fixed world location but are measured with respect to the instrumented vehicle. Because both the surrounding vehicles and the instrumented vehicle can move independently, the trajectories are the sum of two motion components.

The 331 trajectories collected looking out the rear of the car over 100 minutes of natural driving are shown in Fig 7.5a. These trajectories correspond to vehicles that were tracked for a minimum of 5 seconds ($5 * 30$ fps $= 150$ samples) along highway segments. Notice some of the trajectories appear to move horizontally. This occurs when the ego-vehicle changes lanes. This added motion is not explicitly modeled or compensated for during the learning process but most of these trajectories are removed with POI filtering because they do not have much support.

The results from activity learning for the moving platform are shown in Fig. 7.5. The location of the lane markers is presented in the figures for scale and interpretation. The set of trajectories used for learning is presented in Fig. 7.5a. The tracks clearly follow the lane markings although there seems to be some bending as further from the ego-vehicle. The scene goals can be viewed in Fig. 7.5b. The regions where tracking begin (Enter) are marked with green, the locations when tracking ends (Exit) is marked with red, and locations where vehicles remain for an extended time (Stop) are marked with yellow. Although there seems to be a large number of randomly distributed points, the resulting zones are pleasing because they show surround vehicles appearing and disappearing either in the rear or on the passenger side of the ego-vehicle. In addition, there is a stop zone directly behind the ego-vehicle at approximately 18 meters. This following distance corresponds to approximately 0.6 second time gap at a typical highway speed of 70 mph. (This is closer than is generally considered safe).

Fig. 7.5c displays the 15 dominant routes automatically learned from the trajectory

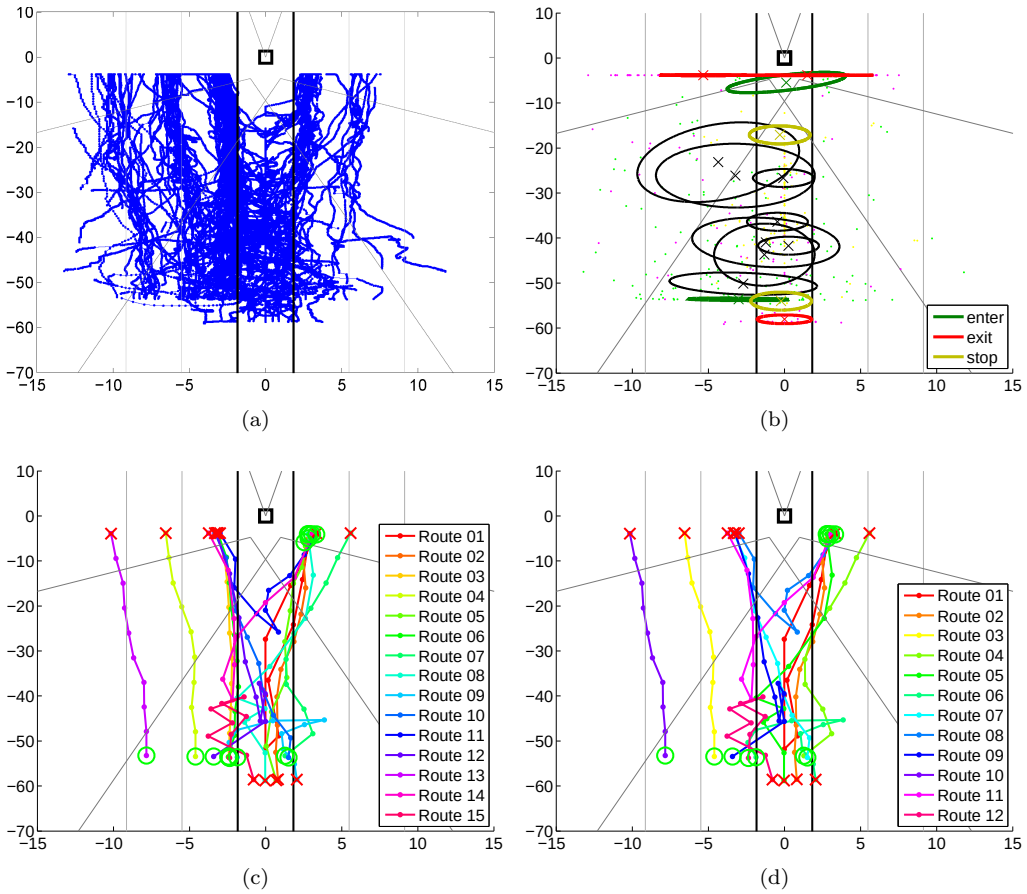


Figure 7.5: (a) Trajectories seen from behind a vehicle collected over 100 minutes of natural driving. (b) Entry/Exit/Stop zones: notice the stop zone behind the vehicle at -18 meters which is only a 0.6 second time gap. (c) Initial set of spatial routes. (d) Remaining routes after merge estimation. Notice the activities on the driver side ($-x$ axis) correspond to faster cars passing the ego-vehicle while the passenger side ($+x$ axis) shows the ego-vehicle passing a slower vehicle.

training set. The refined routes are shown in Fig. 7.5d after three similar activities are removed during the merge procedure. The routes seem to indicate mostly passing maneuvers *e.g.* Route 2, Route 3, or Route 11 in Fig. 7.5d. During an ego-pass (Route 3) vehicle tend to appear in the rear (green circle) and travel alongside the ego-vehicle in the adjacent lane before exiting sensor view on the driver side (red x). Conversely, during an pass (Route 2), vehicles appear on the passenger side and move further away until disappearing far behind the ego-vehicle indicating passing of a slow moving vehicle. This corresponds with typical driving conventions where the faster lanes are toward the inside of the highway on the driver side (left) while slower vehicles tend to travel on the passenger side. This generally results in a fast moving vehicle passing on the left side of a slow moving vehicle in the right-hand lane.

7.5.2 Examining Front Trajectories

The next experiment examined trajectories from the front of the vehicle as extracted from the ACC RADAR system. The ACC dataset consisted of 480 trajectories. As in the SWA experiment, the trajectories were obtained during natural highway driving and correspond to vehicles that were observed for a minimum of 5 seconds. In contrast with the SWA system, the front RADAR has a very narrow sensing cone because it is designed to detect only vehicles directly ahead. The narrow field of view results in trajectories that are typically much further away than in the rear case. Vehicles in the adjacent lanes are not detected until they further than almost 50 meters.

The ACC track learning results are summarized in Fig. 7.6. The trajectories, seen in Fig. 7.6a, mostly remain in the ego-lane or the directly adjacent lanes in contrast to SWA which seemed to detect 2 lanes over. In addition, there tends to be much more horizontal maneuvering between adjacent lanes. The zones in Fig. 7.6b suggests two different following distances for the driver. A close, and unsafe, distance of 25 meters which is approximately a 0.8 second time gap and another further gap of 2.5 seconds, or 80 meters. The final learned routes in Fig. 7.6d follow a similar pattern as for the rear. On the driver side, there are ego-passes (Route 4 and 11) and on the passenger side there are plain passes (Routes 1, 7, and 12). What is different than in the rear case is exemplified by Route 5. In this case, there is an ego-overtake. A vehicle on the driver side passes the ego-vehicle on the left and then merges to the right into the ego-lane to complete an overtake maneuver. This type of behavior could be potentially dangerous if it occurred close to the ego vehicle (here it occurs 65 meters ahead). Predicting this pattern early could greatly improve safety if a driver were alerted.

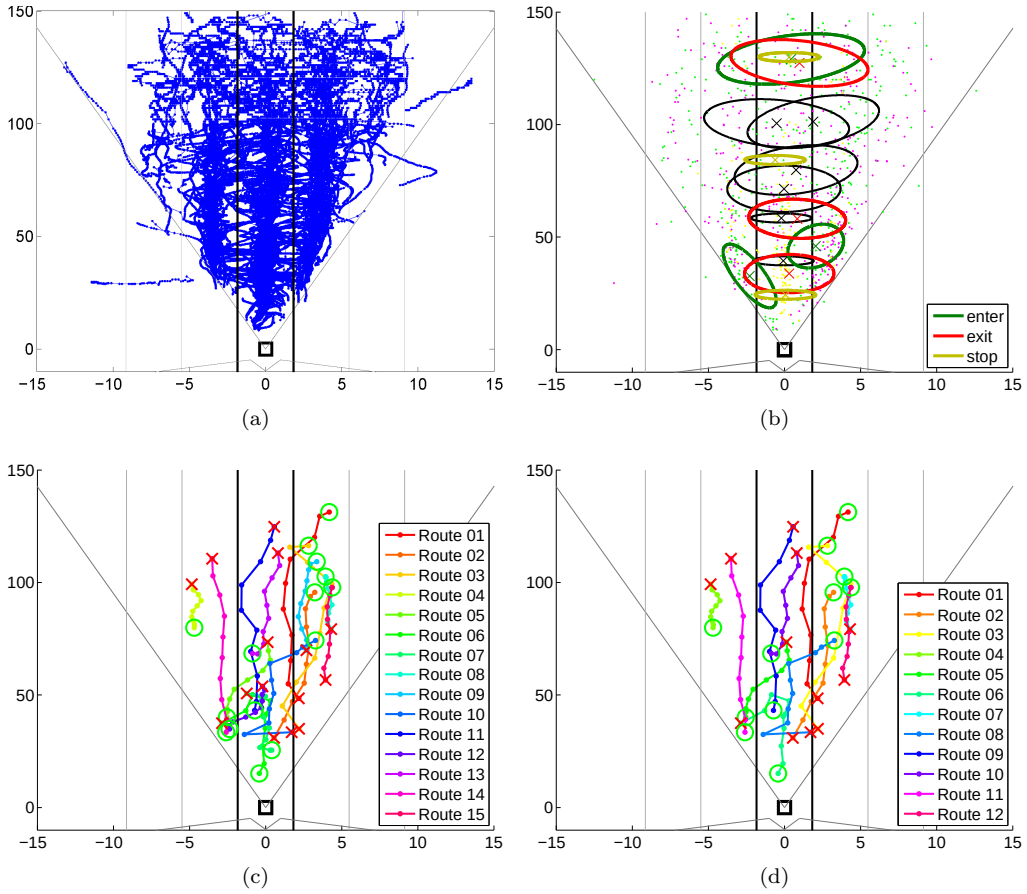


Figure 7.6: (a) Trajectories extracted by the ACC system in the front of the vehicle. (b) Entry/Exit/Stop zones: There are two typical following distances, 25 m and 85 m. (c) Initial set of spatial routes. (d) Remaining routes after merge estimation. The activities on the passenger side ($+x$ axis) correspond to the ego-vehicle passing slower vehicles.

7.6 Future Work

Preliminary analysis shows it is possible to use naturalistic driving and extract meaningful surround behaviors. Further research will need to account for the instrumented vehicle and environment when analyzing trajectories. The instrumented vehicle's ego-motion should be compensated when analyzing surrounding trajectories to model solely surround behavior. When the LISA vehicle changes lanes, the surround tracks all appear to change lanes when there is no ego-motion compensation. In addition, the shape of the road may play a key road in predicting behaviors as people will behave differently during curves and straight segments. The trajectories of similar actions may look quite different depending on road curvature. A true surround safety system will need to examine the sides in conjunction with the front and rear. A key research task will need to determine if each section of the vehicle should be considered separately or if tracks (and hence behaviors) need to be connected between the sensors in these regions.

More data needs to be collected with a front sensor with a wider FOV. The limited aperture of the ACC radar is designed only for looking straight ahead in the current lane. To provide real safety recommendations, the surround coverage needs to actively monitor close to the vehicle as this is where dangerous situations occur. It is also clear from the ACC experiment that maneuvers can occur anywhere. Learning needs to decouple the spatial location which will allow detection of the overtake maneuver closer to the vehicle where it counts for safety.

7.7 Concluding Remarks

This work presents the first attempt to automatically learn behaviors of surround vehicles based on natural observation during driving. The unsupervised learning framework introduced in Chapter 4 is applied after small modifications to the trajectory representation emphasizing its generality. Trajectories of surrounding obstacles obtained from both the rear and front of an instrumented vehicle are examined and key behaviors which correspond to observable phenomenon, passing and overtake, are automatically discovered demonstrating the value of this data driven approach.

Acknowledgements

Chapter 7 is in part a reprint of material that appears in the Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2009, by Brendan T. Morris and Mohan M. Trivedi. The dissertation author was the primary investigator and author of this paper.

Chapter 8

Concluding Remarks

In this dissertation, a critical exploration into automatic activity understanding through learning trajectory patterns is undertaken. A number of practical implementation issues were considered for real world implementation of such an unsupervised activity learning system. First, a representation for an activity was needed that was expressive enough to answer interesting monitoring questions while flexible enough to describe the activities encountered in a wide range of environments and applications. The activity definition had to be independent of the analysis domain for unsupervised learning. Second, activity analysis could not be an offline, post-mortum, analysis but must occur in real-time on live data for maximal utility. The analysis needed to describe the current activity state of the observed world in order to provide timely notification and alert for appropriate response.

In Chapter 4 a practical approach is presented for understanding activities. A multi-level learning framework is introduced which is able to automatically learn activity models, a sequence of actions, without a priori domain knowledge. Activities are inferred by the observation of trajectory patterns resulting from object motion. In an unsupervised manner, the trajectory patterns are extracted through clustering to form probabilistic activities models which enable analysis of live video. Key contributions to the trajectory learning field were the development of methods to learn activities in the presence of imperfect tracking providing robustness. A thorough comparative evaluation of trajectory specific similarity measures highlighted the importance of techniques to handle the variable lengths of trajectories with optimal alignment but also found that typical applications are insulated from the particular alignment algorithm choice. Finally, by providing a probabilistic definition of an activity, analysis results had an intuitive interpretation and activity models could be adapted to better reflect current conditions for long-term monitoring. Extensive evaluation of the framework, not seen in the trajectory learning literature, was performed to accurately characterize the quality of predication and abnormality detection in an online fashion which provides a benchmark for future research.

Chapter 5 introduces the VEhicle Classifier and Traffic flOw analyzer (VECTOR) system to provide video based analysis of highways. Utilizing the trajectory learning framework, wide coverage of a highway link in both directions of travel are covered by a single camera. The system provides, with a single sensor, the critical traffic management measures currently delivered by many intrusive inductive loops. In addition, VECTOR utilizes the rich informational content available in video to extend the traffic measurements to include a classification of the types of vehicles on the road. This real-time fleet composition is a currently missing component of advanced traffic management and modeling. Through trajectory analysis, the safety of the highway link is characterized based on daily speed profiles rather than posted speed limits, the location and rate of lane changing (needed for congestion analysis) is measured, and abnormal driving patterns which could signify an incident are detected.

The single camera methods of the previous chapters are extended in Chapter 6 with the development of the Contextual Activity Notification Visualization Analysis System (CANVAS). CANVAS provides wide-area coverage and contextual awareness to multiple different sensors in a unified interface. Activity analysis is provided over larger areas through a transformation of sensor coordinates into the latitude and longitude of a map for an intuitive monitoring interface.

Finally, in Chapter 7 the trajectory learning and analysis framework is pushed to its limits by examining the activities viewed from within a moving automobile. Rather than try to fit lavish dynamic motion models to the vehicles detected in the local surround, the maneuvers of detected surround vehicles are inferred based on experience. The trajectory learning framework was able to extract meaningful driving maneuvers, corresponding to passing and overtaking without explicitly defining these as patterns of interest. This type of trajectory analysis is of great importance to the intelligent vehicle community because early notification of the intents of surrounding vehicles gives a driver extra time to plan and prepare which in turn ensures safer driving conditions.

This dissertation highlights the importance of trajectory learning and analysis for unsupervised activity understanding. It provides rich descriptions of a wide range of environments and applications. The utility of these techniques will only grow more important as video cameras find there way into more parts of our daily lives and present the challenge of what to do with all that is being observed.

Appendix A

Dataset Description

A.1 Datasets

Experiments are conducted using six datasets with varying properties. They include several simulated scenes as well as surveillance scenes where trajectories are extracted using a background based object tracker. Figure A.1 illustrates each scene with true clusters and Table A.1 summarizes the datasets using the metrics defined in Appendix A. Each set is characterized by the number of trajectories N, the number of cluster labels K, speed deviation σ_V, length deviation σ_T, shape complexity ξ [136],

$$\xi = \frac{d_E(f_T, f_1)}{\sum_i d_E(f_{i+1}, f_i)} \tag{A.1}$$

and separability Δ_c/Δ and Δ_c/Δ_m. The average separability is

$$\Delta = \frac{1}{NK} \sum_{n=1}^{N} \sum_{c=1}^{K} d_{nc} \tag{A.2}$$

and the cluster tightness Δ_c and minimum cluster separability Δ_m are defined as

$$\Delta_c = \frac{1}{N} \sum_{\substack{n=1 \\ g_n=c}}^{N} d_{nc} \tag{A.3}$$

$$\Delta_m = \min_n \left[\min_c \; d_{nc} \right], \qquad c \neq g_n \tag{A.4}$$

where g_n is the ground truth label for track n and

$$d_{nc} = \frac{1}{N_c} \sum_{g_j=g_c} d_E(F_n, F_j) \tag{A.5}$$

with N_c the number of trajectories with label g_c.

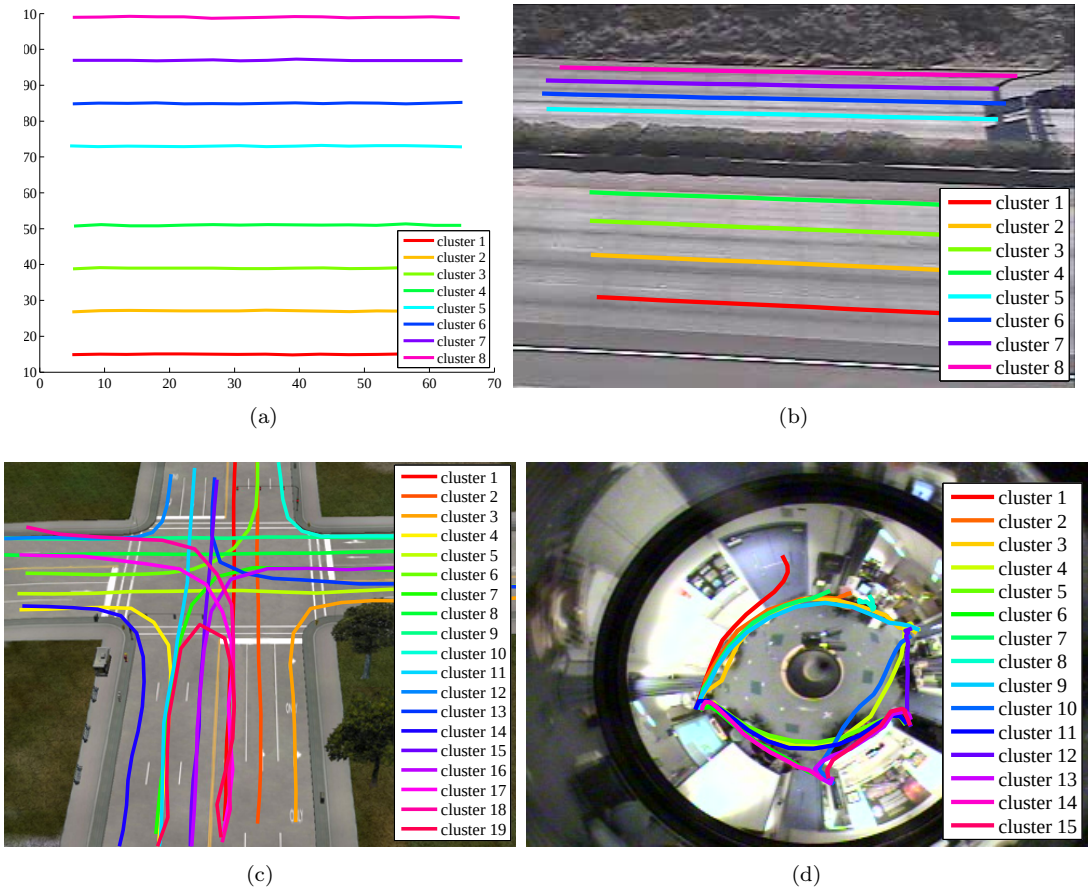


Figure A.1: Trajectory Clustering Datasets (a) I5SIM (b) I5 (c) CROSS (d) LABOMNI

A.1.1 I5SIM

The I5SIM datasets simulate trajectories obtained on a 4 lane highway with traffic in both directions (8 total lanes). The trajectory points are in real world coordinates. The first set contains only free flow traffic with a Gaussian speed distribution of 70 mph with a 5 mph standard deviation. The second and third sets contain the free flow traffic as well as trajectories during congestion (25 mph). These were designed to compare performance of different clustering goals. The trajectories in the I5SIM2 dataset are labeled by lane number (just spatial coordinates). Those in I5SIM3 were labeled by lane and flow type, differentiating trajectories in the same lane but traveling at different speeds, resulting in 16 labels (8 lanes at free flow and 8 during congestion).

Table A.1: Experimental Dataset Characterization

	N	K	σ_V	σ_T	ξ	Δ_c/Δ	Δ_c/Δ_m
i5sim	800	8	0.19	1.29	0.99	0.07	0.27
i5sim2	1600	8	0.81	18.78	0.95	0.08	0.31
i5sim3	1600	16	0.81	18.78	0.95	0.08	1.39
i5	806	8	2.38	4.10	1.00	0.16	1.30
cross	1900	19	5.07	4.27	0.85	0.07	0.97
labomni	209	15	0.31	142.26	0.71	0.22	1.44

A.1.2 I5

The I5 dataset contains trajectories obtained by visual tracking of vehicles from a highway mounted camera overlooking a busy interstate. The track labels correspond to one of 8 lane numbers as in the I5SIM sets. The raw tracker output was automatically filtered to remove clearly erroneous trajectories which occur because of occlusions.

A.1.3 CROSS

The CROSS dataset depicts a four way traffic intersection. These provide more complex trajectory shapes than in the highway datasets. The 19 acceptable intersection maneuvers include turns and even a u-turn.

A.1.4 LABOMNI

The final dataset examines humans rather than vehicles. An omni-directional camera was placed in the middle of a lab to observe trajectories from a less constrained environment than encountered by vehicle traffic. The participants were not aware of the data collection to ensure naturally occurring motion patterns. The trajectories have a long time duration and tend to have a large degree of overlap in the image plane.

Appendix B

Comparison of Trajectory Clustering Techniques

A major research area in computer vision is the study of activities and behavior. Recently there has been high interest in automatic activity and behavior understanding. Using unsupervised methods, researchers try to observe a scene, learn prototypical activities, and use the prototypes for analysis. This paradigm has been of particular interest for surveillance [110, 76] and traffic monitoring [95, 5, 89] where methods to categorize observed behavior, detect abnormal actions for quick response, and even predict future occurrences is desired. Because of the large number of cameras in use for these applications there is a constant stream of large amounts of data making it difficult to manually analyze each individually which necessitates the use of unsupervised methods. In these cases, activity is characterized by motion and can be succinctly represented with a trajectory. It is possible to collect trajectories over sufficient time and learn typical behaviors through clustering. Unfortunately, even with much work in the area it is unclear what are the best methods for clustering. There are a wide number of similarity functions for trajectories and researchers continue their design as well as little agreement of how clustering should be performed. A recent survey by Morris and Trivedi [90] presented the wide variety of procedures for trajectory learning and modeling. The following examines a number of popular trajectory clustering procedures to find their strengths and weakness with the intention of determining which might be the best for trajectory learning. The evaluation has three separate components which include comparison of trajectory distance measures (Table B.1), comparison of different clustering methods (Table B.2), and analysis on a variety of dataset with varying characteristics (Table A.1).

Table B.1: Trajectory Distance Measures

Technique	Publication
HU	Hu 2007 [50]
PCA	Bashir 2007[8]
DTW	Keogh 2000 [63]
LCSS	Buzan 2004 [17]
PF	Piciarelli 2006 [95]
MODH	Atev 2006 [5]

B.1 Distance Measures

Previous work by Zhang *et al.* [136] compared the use of a few popular distance measures at the time, the Hausdorff distance, a HMM-based distance, Euclidean distance, Euclidean distance in a PCA subspace, dynamic time warping (DTW), and longest common subsequence (LCSS). We expand the comparison by including new similarity measures that have been designed specifically for trajectories while ignoring both Hausdorff and HMM which were shown to have poor performance. Table B.1 lists the distance measures adopted in recent literature which are assessed in this work. The examination includes fixed length measures, Hu Euclidean and PCA, as well as time-normalized distances, DTW, LCSS, Piciarelli and Foresti (PF), and modified Hausdorff (MODH).

B.1.1 Notation

A trajectory

$$F = \{f_1, \ldots, f_t, \ldots, f_T\} \tag{B.1}$$

is a collection of flow vectors f_t representing the spatio-temporal characteristics of moving objects at each time t of the total track life T. A flow vector generally indicates location and dynamics, $f_t = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]$, but in this work we restrict ourselves to just spatial location, $f = [x, y]$. This is a common practice as it results in a natural interpretation of spatial proximity given the Euclidean the distance between flow points

$$d_E(f_t, f_\tau) = \sqrt{(x_t - x_\tau)^2 + (y_t - y_\tau)^2}. \tag{B.2}$$

Some of the following distance measures must use fixed length data and can not be used on raw trajectory data because they typically have varying length. Instead, a resampled version of a track is used and the trajectory notation is overloaded

$$F = \{f_1, \ldots, f_k, \ldots, f_L\}. \tag{B.3}$$

A resampled trajectory is referenced by an index k rather than time t and has a fixed length L. The resampling implementation used in this paper interpolates points to have equal distance between flow vectors [87].

B.1.2 HU

The HU distance is computed as the average Euclidean distance between points on two trajectories [50].

$$D_{HU}(F_i, F_j) = \frac{1}{L}\sum_{k=1}^{L} d_E(f_{i,k}, f_{j,k}), \tag{B.4}$$

This distance function relies on similar trajectories having the same point distribution with consecutive points in corresponding tracks in spatial proximity.

B.1.3 PCA

Instead of working in the trajectory coordinate space, PCA is used to transform the trajectories into a lower dimensionality subspace. The x and y coordinates of a trajectory are concatenated into a one dimensional vector and projected onto the subspace by PCA decomposition The PCA distance is computed as the Euclidean distance between PCA coefficients, a_l,

$$D_{PCA}(F_i, F_j) = \frac{1}{N_\lambda}\sum_{l=1}^{N_\lambda} d_E(a_{i,l}, a_{j,l}). \tag{B.5}$$

Only $N_\lambda << 2L$ coefficients are retained to limit the size of the space. N_λ is chosen by examining eigenvalues λ_k to retain 95% of the dataset variation [8]. The PCA distance is similar to Hu but works in a lower dimensional space for reduced computation and robustness through the PCA shape decomposition. Note trajectories must be of equal length for PCA decomposition.

B.1.4 DTW

The above distance measures require fixed length trajectories which do not normally occur because observation duration is variable. DTW is used to compare unequal length signals by finding a time warping that minimizes the total distance between matching points [99].

$$\begin{aligned} D_{DTW}(F_i, F_j) &= \frac{(d_{DTW}(F_i, F_j) + d_{DTW}(F_i, F_j))}{2} \\ d_{DTW}(F_i, F_j) &= \frac{1}{T_i}\sum_{t=1}^{T_i} d_E(\phi_{i,t}, \phi_{j,t})\; m_t/M_\phi \end{aligned} \tag{B.6}$$

where ϕ_i and ϕ_j are the time warping functions that minimize the distance between aligned points, m_t is a path weighting coefficient, and M_ϕ is a path normalization factor. The warping path ϕ is efficiently found using dynamic programming.

B.1.5 LCSS

LCSS is another alignment tool for unequal length data but is more robust to noise and outliers than DTW because not all points need to be matched. Instead of a one-to-one mapping between points, a point with no good match can be ignored to prevent unfair biasing. The LCSS distance suggested by [123] is defined as

$$D_{LCSS}(F_i, F_j) = 1 - \frac{LCSS(F_i, F_j)}{\min(T_i, T_j)}, \tag{B.7}$$

where the

$$LCSS(F_i, F_j) = \begin{cases} 0 & T_i = 0 \mid T_j = 0 \\ 1 + LCSS(F_i^{T_i-1}, F_j^{T_j-1}) & d_E(f_{i,T_i}, f_{j,T_j}) < \epsilon \\ & \& \quad |T_i - T_j| < \delta \\ \max\left(LCSS(F_i^{T_i-1}, F_j^{T_j}), LCSS(F_i^{T_i}, F_j^{T_j-1})\right) & \text{otherwise} \end{cases} \tag{B.8}$$

value specifies the number of matching points between two trajectories. $F^t = \{f_1, \ldots, f_t\}$ denotes all the flow vectors in trajectory F up to time t. The LCSS, like DTW, can also be efficiently computed using dynamic programming.

B.1.6 PF

In a similar spirit to DTW and LCSS, Piciarelli and Foresti [95] defined another distance measure to deal with time drift. They observed that matching tracks would generally agree early (consistent starting points) but over time matched points had a tendency to drift further away because of speed differences. Accordingly, their trajectory distance measure finds matching points within a time window that grows larger at each time

$$D_{PF}(F_i, F_j) = \frac{1}{T_i} \sum_{t=1}^{T_i} d_{PF}(f_{i,t}, F_j) \tag{B.9}$$

where

$$\begin{aligned} d_{PF}(f_{i,t}, F_j) &= \min_{\tau} \left(\frac{d_E(f_{i,t}, f_{j,\tau})}{Z_\tau} \right), \\ \tau &\in \{\lfloor (1-\delta)t \rfloor \ldots \lceil (1+\delta)t \rceil\}. \end{aligned} \tag{B.10}$$

Z_τ is a normalization constant that measures the variance of point τ. The definition is noteworthy because it allows comparison with incomplete trajectories (developing tracks) making it well suited for online clustering unlike DTW or LCSS.

In this work, $Z_\tau = 1$ in order to compare two trajectories rather than a trajectory and a cluster as originally designed. The temporal window is also slightly modified to grow logarithmically with trajectory length $\tau \in \{\lfloor t - \delta \log t \rfloor \ldots \lceil t + \delta \log t \rceil\}$ to prevent very large windows for long trajectories.

Table B.2: Clustering Techniques

Technique	Publication
Direct	Morris 2008 [86]
Divisive (rb,rbr)	Billotti 2005 [11]
Agglomerative	Buzan 2004 [17]
Hybrid (cham)	Karypris 1999 [60]
Graph	Li 2006 [69]
Spectral	Hu 2007 [50]

B.1.7 Modified Hausdorff

The Hausdorff distance has been commonly used to compare two unequal size sets but is not well suited for trajectories because it does not account for ordering [136]. The modified Hausdorff distance $D_{MODH}(F_i, F_j)$ [5] was designed to respect the time-ordering of points and reduce sensitivity to outliers by allowing slack when matching.

$$\begin{aligned} D_{MODH}(F_i, F_j) &= \overset{\alpha}{\underset{f_{i,t} \in F_i}{\text{ord}}} \, h(f_{i,t}, F_j) \\ h(f_{i,k}, F_j) &= \min_{f_{j,\tau} \in \mathcal{N}(C(f_{i,t}))} d_E(f_{i,k}, f_{j,\tau}) \end{aligned} \tag{B.11}$$

where $\mathcal{N}()$ is a neighborhood window, $C(f_{i,t})$ is the point in F_j that correspond to point $f_{i,t}$ in F_i, and ord $^{\alpha}_{f_{i,t} \in F_i} h(f_{i,t}, F_j)$ denotes the value of $h(f_{i,t}, F_j)$ that is larger than α percent of all other h values over F_i. The distance between trajectories is thus a prototypical distance chosen from among the best matching points. The trajectory alignment is controlled by the correspondence function $C(.)$ which assumes that corresponding points occur at the same fraction of total track length. This convention accounts for speed variation within similar spatial patterns.

B.2 Clustering Algorithms

Besides examining the effects of different trajectory distance measure, the quality of clusters returned by different types of clustering methods is explored to determine if certain techniques are better suited for trajectories. The classes of clustering algorithms we consider are direct methods, hierarchical agglomerative and divisive procedures, hybrid divisive-agglomerative techniques, graph cuts, as well as spectral methods. A summary of recent research utilizing these different clustering techniques is shown in Table B.2.

For ease of clustering, a similarity matrix $S = \{s_{ij}\}$, which represents a fully connected graph, is constructed from the trajectory distances using a Gaussian kernel function

$$s_{ij} = e^{-D^2(F_i, F_j)/2\sigma^2} \in [0, 1]. \tag{B.12}$$

where D represents one of the distance measure defined previously and the parameter σ describes the trajectory neighborhood. Large values of σ cause further apart trajectories to have a higher similarity score while small values lead to a more sparse similarity matrix (more entries will be very small). The S matrix along with the desired number of clusters are used as input into the differing clustering algorithms which are discussed below.

B.2.1 Direct

The direct clustering methods find the K clusters simultaneously. A initial guess of clusters is iteratively optimized by adjusting each cluster component in unison to find a globally satisfying solution

Popular direct optimization solvers in the Euclidean space are k-means and the soft assignment version fuzzy c means (FCM). These are used as the baseline clustering techniques for comparison.

B.2.2 Agglomerative

Agglomerative clustering is a bottom-up strategy that initially treats each trajectory as an individual cluster and merges similar clusters hierarchically in a tree-like structure, stopping when only K clusters remain. At each merge step, a hard decision on cluster membership is made limiting the algorithms ability to adjust at a higher tree level.

B.2.3 Divisive

Divisive clustering is the top-down dual to agglomerative clustering where the entire trajectory training set is considered a single cluster. The K clusters are obtained by performing $K-1$ repeated bisections where each bisecting cluster split results an optimal 2-way division of the similarity matrix. In addition to ensuring local optimality at the bisections, a global optimization step can be used to optimize the solution across all bisections.

B.2.4 Hybrid

Hybrid clustering solutions combine both divisive and agglomerative techniques. By using different criterion functions during the partitioning and agglomeration phases, more complex (non-globular) clusters can be discovered. The dataset is first clustered into $M > K$ clusters using one of the partition methods and the final K clusters are obtained by merging some of the M clusters.

Table B.3: Best CCR Performance (Average over 5 runs)

	kmeans	fcm	hu	pca	dtw	lcss	pf	modh
i5sim	0.8162	0.8900	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
i5sim2	0.7250	0.8154	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
i5sim3	0.4901	0.4984	0.5735	0.5574	0.9994	1.0000	0.9944	0.7725
i5	0.6397	0.8000	0.9107	0.7655	0.9722	0.9975	0.9613	0.9975
cross	0.6937	0.7163	0.9963	0.9947	0.9958	0.9869	0.9947	0.9916
labomni	0.7952	0.7923	0.8900	0.9091	0.8383	0.9091	0.8325	0.8325

B.2.5 Graph

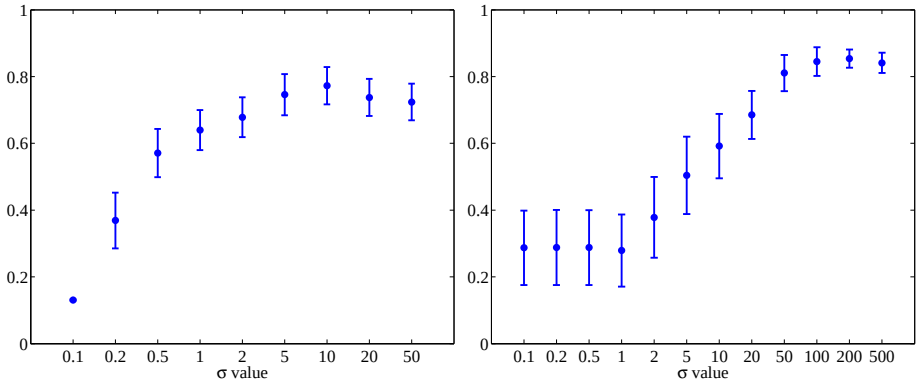
Similar to the divisive clustering method, graph methods seek to divide the full dataset into individual clusters [107]. Instead of operating directly on the similarity matrix, a nearest neighbor graph is constructed where a trajectory is a vertex. Each vertex is connected by a weighted edge to its most similar trajectories. The K clusters are found using a min-cut partitioning algorithm which finds a division of the graph with minimal loss of edge weights.

B.2.6 Spectral

Spectral clustering has become popular recently because it can be efficiently computed and improved performance over more traditional clustering algorithms such as k-means. Spectral methods do not make any assumptions on the distribution of data points and instead relies on eigen decomposition of the similarity matrix which approximates an optimal graph partition [91]. We compare compare 4 flavors of spectral algorithms by selecting to decompose either the Laplacian of Shi and Malik [107] or Ng *et al.* [91] followed by a final clustering of eigenvectors with either k-means or FCM.

B.3 Clustering Evaluation

The following section presents clustering results for a number of different experiments. The best classification results are displayed in Table B.3. These results are the average performance over 5 runs for each dataset and similarity type using 48 different clustering method variations. The average results versus cluster method, distance measure, and dataset are presented in Fig. B.2.



(a) (b)

Figure B.1: Clustering quality for different σ values for (a) HU and (b) DTW averaged across all datasets. As σ increases the performance improves resulting in higher CCR and lower variance.

B.3.1 Evaluation Criteria

Since the labels returned by a clustering run was arbitrary, the accuracy of a clustering was evaluated by finding the one-to-one mapping between the ground truth and clustering labels which maximized the number of matches. The assignment problem can be solved using the Hungarian algorithm [66] when recast to minimize the number of mismatched labels. Given the label mapping, the cluster quality is measured by the correct clustering rate (CCR) [136]

$$CCR = \frac{1}{N}\sum_{c=1}^{K} p_c \tag{B.13}$$

where N is the total number of trajectories and p_c denotes the total number of trajectories matched to the c-th cluster.

B.3.2 Procedure

The CLUTO [1] software package is used for agglomerative, divisive, hybrid (CHAMELEON [60]), and graph based clustering. The software provides a number of options and optimization criteria for each cluster method which results in a total of 44 different cluster variants. An additional 4 spectral cluster variants were implemented in Matlab for a total of 48 cluster variants applied for each similarity measure to each dataset. Every clustering combination was run 5 times with random initialization to better represent expected performance.

The experimental evaluation consisted of three main parts. The effect of the neighborhood parameter σ was investigated, a sweep was done through all distance measure parameters to ensure near optimal values, and finally the clustering evaluation was performed by varying the distance measure, cluster method, and dataset.

B.3.3 Gaussian Kernel Evaluation

The effect of trajectory neighborhood on clustering was examined by varying the parameter σ in (4.6). Fig. B.1 shows improved performance as σ increases. The average CCR not only increases but the variance decreases. Although the quality appears to saturate at a particular σ choice, larger values than this cause little performance degradation. An average similarity $\frac{1}{N^2}\sum_i \sum_j s_{ij} = 0.1$ was used to produce good results.

B.3.4 Clustering Method Evaluation

The plot in Fig. B.2a shows that on average the choice of clustering method has little effect on the quality of the results. It is noteworthy to mention that the soft membership of FCM improves performance 6% over k-means making it a clear winner between the baseline approaches. All cluster methods perform significantly better than the baseline except for the direct method which is the same category both k-mean and FCM fall into. Although the graph results were only 5% lower, graph based clustering was significantly more difficult as results were very sensitive to the graph neighborhood definition.

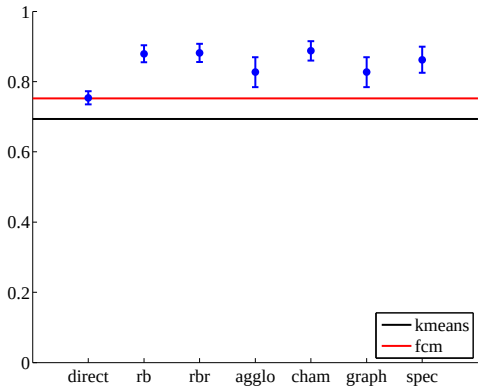
B.3.5 Distance Measure Evaluation

The average CCR results as a function of distance measure is shown in Fig. B.2b. This shows the effort in designing measures that allow the use of raw variable length trajectories is not wasted since the sampled measures, HU and PCA, perform almost 10% worse. Unfortunately, the newer trajectory specific distances, PF and MODH, have comparable performance with DTW and LCSS.

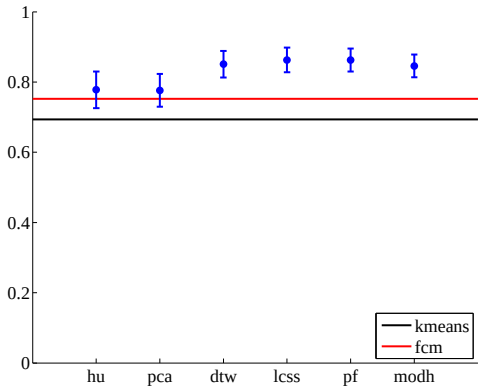
Further insight can be found by examining the columns of Table B.3. The performance of HU, PCA, and MODH all degrade for the I5SIM3 dataset where there are different speeds in the same lane. The speed information is thrown out during resampling when using HU and PCA and it is also ignored by the modified Hausdorff distance because the corresponding points are mapped based on total trajectory length. While LCSS performs uniformly well, it is surprising that both DTW and PF which do not have outlier robustness exhibit corresponding performance. The need for outlier suppression is lessened for trajectory data because of the smoothing inherent in the tracker (*e.g.* Kalman or particle filter). Unless the tracker makes a gross mistake it is unlikely for a trajectory to contain outlier points that would greatly influence warping match distance.

B.3.6 Dataset Evaluation

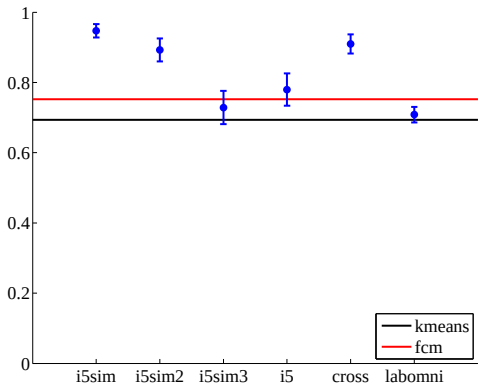
The preceding sections implied little difference in effectiveness when using different clustering methods or different time aligned distances but inspection of Fig. B.2c clearly differentiates performance between datasets. The CCR results are quite high for the more simple I5SIM,



(a)



(b)



(c)

Figure B.2: Average CCR performance plotted against experimental variables. (a) Clustering algorithm (b) Distance measure (c) Dataset

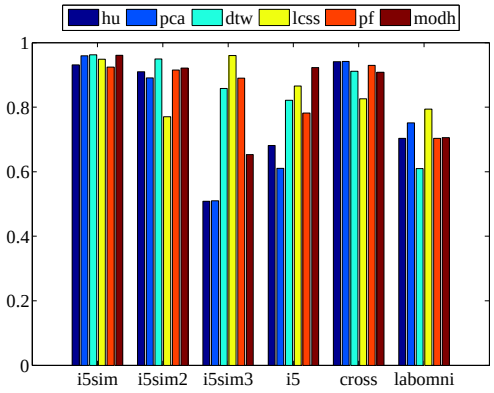


Figure B.3: Average performance for the different similarity measures for each dataset.

I5SIM2, and CROSS datasets. With $\Delta_c/\Delta_m < 1$, these sets have tight clusters well separated from one another.

The bar graph in Fig B.3 indicates a performance distribution across distance functions for each dataset which is lost in the averaged plot. Fig. B.4 shows a detailed view of I5SIM2 and I5SIM3. These sets had 8 highway lanes with trajectories collected from 2 different speed profiles, free flow and congestion, but only I5SIM3 required differentiation based on speed as well as lane number. All distance methods performed very well in the I5SIM2 set, Fig. B.4a, except LCSS which actually did worse than FCM. There was a large variation in performance given the clustering method and though perfect clustering was possible (see Table B.3) the direct and divisive solutions lowered the average performance. Fig. B.4b shows the dramatic improvements possible with the right distance choice. The DTW, LCSS, and PF distances were able to resolve both position and speed differences with a high degree of accuracy.

Viewing Table B.3 we see the LABOMNI dataset performance was similar across all the distance types, even the baseline k-means and FCM. HU and PCA which reduce dimensionality and focus on shape performed better than all the time alignment techniques, except LCSS, because the long length of trajectories which allow ample opportunity for misalignment..

Another interesting result from Table B.3 is the significant CCR loss for PCA in the I5 dataset. $\xi = 1$ and high Δ_c/Δ_m score indicates straight overlapping lanes due to camera perspective which causes the northbound lanes furthest from the camera to appear very close in the image plane. Unlike the LABOMNI set which has a lower ξ value, trajectories cannot be distinguished well by intermediate points and the PCA decomposition filters out the fine differences.

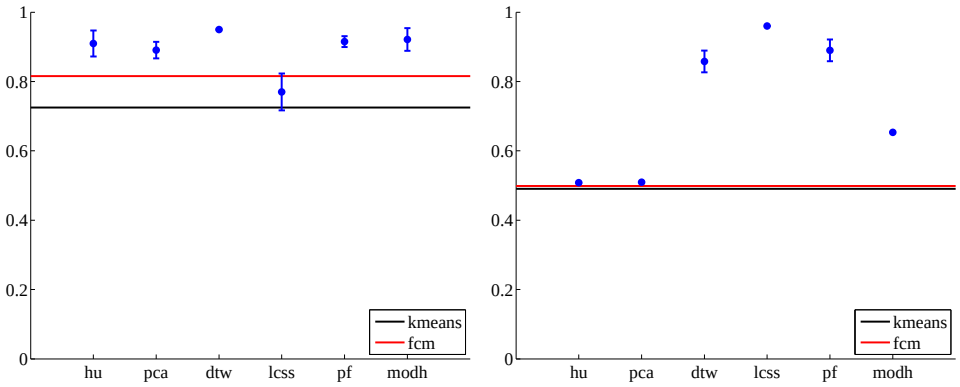


Figure B.4: (a) Average CCR for I5SIM2. The direct and divisive methods perform poorly for LCSS. (b) Average CCR for I5SIM3. The time alignment measures {DTW, LCSS, PF} perform significantly better.

B.4 Concluding Remarks

This work evaluated the performance of a number of clustering procedures for the trajectory clustering task. The evaluation consisted of a comparison of 6 trajectory distance measures, 7 clustering methods, and 6 varied datasets. Without prior knowledge, the choice of clustering method and distance measure was not important as long as it operated on full unsampled tracks, though LCSS was consistently a top performer. Performance was actually dictated by the trajectory properties encountered in a dataset. When trajectories were very long the data reduction techniques worked well by focusing on coarse shape and position and when dynamics were considered an important separating factor the time-normalized distances dominated.

B.5 Acknowledgements

Appendix B is a reprint of material that appears in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, by Brendan Morris and Mohan Trivedi. The dissertation author was the primary investigator and author of these papers.

We acknowledge the sponsorship of UC Digital Media Program for the experimental infrastructure and thank Mr. Minh Van Pham Ly of UC Berkeley who assisted as a UCLEADS program intern.

Bibliography

[1] (2008) CLUTO 2.1.1 - software for clustering high-dimensional datasets. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

[2] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding*, vol. 73, pp. 428–440, 1999.

[3] A. T. Ali and M. M. Venigalla, "Global positioning systems data for performance evaluation of hov and gp lanes on i-66 and i-395/i-95," in *Proc. IEEE Conf. Intell. Transport. Syst.*, 2006, pp. 915 –920.

[4] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 299–310, 2009.

[5] S. Atev, O. Masoud, and N. Papanikolopoulos, "Learning traffic patterns at intersections by spectral clustering of motion trajectories," in *IEEE Conf. Intell. Robots and Systems*, Bejing, China, Oct. 2006, pp. 4851–4856.

[6] R. P. Avery, Y. Wang, and G. S. Rutherford, "Length-based vehicle classification using images from uncalibrated video cameras," in *Proc. IEEE Int'l. Conf. Intell. Transport. Syst.*, Oct. 2004, pp. 737–742.

[7] F. Bashir, W. Qu, A. Khokhar, and D. Schonfeld, "HMM-based motion recognition system using segmented pca," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 3, sep 2005, pp. 1288–1291.

[8] F. I. Bashir, A. A. Khokhar, and D. Schonfeld, "Object trajectory-based activity classification and recognition using hidden markov models," *IEEE Trans. Image Process.*, vol. 16, no. 7, pp. 1912–1919, Jul. 2007.

[9] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[10] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, Jun. 1997, pp. 495–501.

[11] D. Biliotti, G. Anotonini, and J. P. Thiran, "Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences," in *IEEE Workshop on Application of Computer Vision.*, Brekenridge, CO,, Jan. 2005, pp. 50–57.

[12] N. D. Bird, O. Masoud, N. P. Papanikolopoulos, and A. Isaacs, "Detection of loitering individuals in public transportation areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 167–177, Jun. 2005.

[13] N. Brandle, D. Bauer, and S. Seer, "Track-based finding of stopping pedestrians - a practical approach for analyzing a public infrastructure," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Toronto, Canada, Sep. 2006, pp. 115–120.

[14] K. Branson and S. Belongie, "Tracking multiple mouse contours (without too many samples)," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2005, pp. 1039–1046.

[15] A. Broadhurst, S. Baker, and T. Kanade, "Monte carlo road safety reasoning," in *Proc. IEEE Conf. Intell. Veh.*, Jun. 2005, pp. 319–324.

[16] H. Buxton, "Learning and understanding dynamic scene activity: A review," *Image and Vision Computing*, vol. 21, pp. 125–136, Jan. 2003.

[17] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," in *Proc. IEEE Inter. Conf. on Pattern Recog.*, Aug. 2004, pp. 521–524.

[18] A. Calbi, C. Regazzoni, and L. Marcenaro, "Dynamic scene reconstruction for efficient remote surveillance," in *Proc. IEEE International Conference on Advanced Video and Signal based Surveillance*, Sydney, Australia, Nov. 2006, pp. 99–104.

[19] (2010) PeMS performance measurement system 10.3. Caltrans, The University of California, Berkeley, and PATH. [Online]. Available: http://pems.eecs.berkeley.edu

[20] C. Cardelino, "Daily variability of motor vehicle emissions derived from traffic counter data," *Journal of the Air and Waste Management Association*, vol. 48, no. 7, Jul. 1998.

[21] A. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *Proc. IEEE Conf. Intell. Veh.*, Jun. 2005, pp. 771–776.

[22] C. Chen, Z. Jia, and P. Varaiya, "Causes and cures of highway congestion," *IEEE Control Syst. Mag.*, vol. 21, no. 6, pp. 26–32, Dec. 2001.

[23] S. Y. Cheng and M. M. Trivedi, "Turn-intent analysis using body pose for intelligent driver assistance," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 28–37, Oct-Dec 2006.

[24] G. Chockalingam. (2009) California Wireless Traffic Report. [Online]. Available: http://traffic.calit2.net/

[25] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, Hilton Head Island, South Carolina, 2000, pp. 142–149.

[26] H. M. Dee and S. A. Velastin, "How close are we to solving the problem of automated visual surveillance?" *Machine Vision and Applications*, vol. 19, no. 5, pp. 329–343, Oct. 2008.

[27] D. Demirdjian, K. Tollmar, K. Koile, N. Checka, and T. Darrell, "Activity maps for location-aware computing," in *IEEE Workshop on Applications of Computer Vision*, Dec. 2002, pp. 70–75.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Stat. Soc.*, vol. 39, pp. 1–38, 1977.

[29] A. Doshi and M. Trivedi, "A comparative exploration of eye gaze and head motion cues for lane change intent prediction," in *Proc. IEEE Conf. Intell. Veh.*, Jun. 2008, pp. 49–54.

[30] ——, "Investigating the relationships between gaze patterns, dynamic vehicle surround analysis, and driver intentions," in *Proc. IEEE Conf. Intell. Veh.*, Jun. 2009, pp. 887–892.

[31] A. Doshi and M. M. Trivedi, "Satellite imagery based adaptive background models and shadow suppression," *Signal, Image and Video Processing*, vol. 1, no. 2, pp. 119–132, Apr. 2007.

[32] ——, "On the roles of eye gaze and head pose in predicting driver's intent to change lanes," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 453–462, Sep. 2009.

[33] J. L. Drury, J. Richer, N. Rackliffe, and M. A. Goodrich, "Comparing situation awareness for two unmanned aerial vehicle human interface approaches," The MITRE Corporation, Tech. Rep. 06-0692, Jun. 2006.

[34] A. Eidehall and L. Petterson, "Statistical threat assessment for general road scenes using monte carlo sampling," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 137–147, 2008.

[35] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, Mar. 2002.

[36] M. Gales, D. Pye, and P. Woodland, "Variance compensation within the MLLR framework for robust speech recognition and speaker adaptation," in *Proc. IEEE Inter. Conf. Spoken Language*, Oct. 1996, pp. 1832–1835.

[37] T. Gandhi and M. M. Trivedi, "Vehicle surround capture: Survey of techniques and a novel omni-video-based approach for dynamic panoramic surround maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 293–308, Sep. 2006.

[38] A. Gepperth, J. Edelbrunner, and T. Bücher, "Real-time detection and classification of cars in video sequences," in *Proc. IEEE Intell. Vehicles Symposium*, Jun. 2005, pp. 625–631.

[39] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini uav," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, Jan. 2008.

[40] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2247–2253, Dec. 2007.

[41] D. Greenhill, J. P. Renno, J. Orwell, and G. A. Jones, "Learning the semantic landscape: Embedding scene knowledge in object tracking," *Real Time Imaging*, vol. 11, no. 3, pp. 186–203, Jun. 2005.

[42] P. Guha, A. Biswas, A. Mukerjee, and K. S. Venkatesh, "Occlusion sequence mining for complex multi-agent activity discovery," in *Proc. IEEE Workshop on Visual Surveillance*, May 2006.

[43] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 37–47, Mar. 2002.

[44] T. B. Hall and M. M. Trivedi, "A novel interactivity environment for integrated intelligent transportation and telematic systems," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Singapore, Sep. 2002, pp. 396–401.

[45] O. Hasegawa and T. Kanade, "Type classification, color estimation, and specific target detection of moving targets on public streets," *Machine Vision and Applications*, vol. 16, pp. 116–121, Feb. 2005.

[46] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proc. IEEE Inter. Conf. Computer Vision Frame-Rate Workshop*, Kerkya, Greece, Sep. 1999.

[47] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W. Hu, "Automatic traffic surveillance system for vehicle tracking and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 175–187, Jun. 2006.

[48] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, Sep. 2006.

[49] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank, "Traffic accident prediction using 3-D model-based vehicle tracking," *IEEE Trans. Veh. Technol.*, vol. 53, no. 3, pp. 677–694, May 2004.

[50] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank, "Semantic-based surveillance video retrieval," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1168–1181, Apr. 2007.

[51] W. Hu, D. Xie, and T. Tan, "A hierarchical self-organizing approach for learning the patterns of motion trajectories," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 135–144, Jan. 2004.

[52] W. Hu, D. Xie, T. Tan, and S. Maybank, "Learning activity patterns using fuzzy self-organizing neural network," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 3, pp. 1618–1626, Jun. 2004.

[53] K. Huang and M. Trivedi, "3D shape context based gesture analysis integrated with tracking using omni video array," in *Proc. IEEE Workshop on Vision for Human Computer Interaction in conjunction with CVPR*, Jun. 2005.

[54] O. Javed, K. Shafique, and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," in *Workshop Motion and Video Computing*, Dec. 2002, pp. 22–27.

[55] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," in *Proc. British Conf. Machine Vision*, vol. 2, Sep. 1995, pp. 583–592.

[56] I. N. Junejo, O. Javed, and M. Shah, "Multi feature path modeling for video surveillance," in *Inter. Conf. Patt. Recog.*, Aug. 2004, pp. 716–719.

[57] Y.-K. Jung and Y.-S. Ho, "Traffic parameter extraction using video-based vehicle tracking," in *Proc. IEEE Int'l. Conf. Intell. Transport. Syst.*, Oct. 1999, pp. 764–769.

[58] S. Kamijo, H. Koo, X. Liu, K. Fujihira, and M. Sakauchi, "Development and evaluation of real-time video surveillance system on highway based on semantic hierarchy and decision surface," in *IEEE Int'l. Conf. Systems, Man, and Cybernetics*, Oct. 2005, pp. 840–846.

[59] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 2, pp. 108–118, Jun. 2000.

[60] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: A hierarchical clustering algorithm using dynamic modeling," *IEEE Computer*, vol. 22, no. 8, pp. 68–75, Aug. 1999.

[61] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, no. 4, pp. 359–381, Apr. 2003.

[62] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory.* Upper Saddle River, New Jersey: Prentice-Hall, 1993.

[63] E. Keogh and M. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Intl. Conf. on Knowledge Discovery and Data Mining*, sep 2000, pp. 285–289.

[64] Z. Kim and J. Malik, "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking," in *Proc. Ninth IEEE Int'l. Conf. Computer Vision*, Nice, France, Oct. 2003, pp. 524–531.

[65] L. A. Klein, M. K. Mills, and D. R. Gibson, "Traffic detector handbook: Third edition - volume ii," Federal Highway Administration, Turner-Fairbank Highway Research Center, FHWA-HRT-06-139, Oct. 2006.

[66] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[67] A. H. S. Lai and N. H. C. Yung, "Vehicle-type identification through automated virtual loop assignment and block-based direction-biased motion estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 1, no. 2, pp. 86–97, Jun. 2000.

[68] S. J. Landry, T. B. Sheridan, and Y. M. Yufik, "A methodology for studying cognitive groupings in a target-tracking task," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 92–100, Jun. 2001.

[69] X. Li, W. Hu, and W. Hu, "A coarse-to-fine strategy for vehicle motion trajectory clustering," in *Proc. IEEE Inter. Conf. on Pattern Recog.*, 2006, pp. 591–594.

[70] H.-C. Liao and H.-J. Wu, "An automatic camera calibration method using gps-enabled mobile device," in *International conference on Advanced Communication Technology*, 2009, pp. 760–763.

[71] K. Lidstrom and T. Larsson, "Model-based estimation of driver intentions using particle filtering," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Oct. 2008, pp. 1177–1182.

[72] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in *Proc. Fourth IEEE Workshop Applications of Computer Vision*, Princeton, New Jersey, Oct. 1998, pp. 8–14.

[73] A. Liu and A. Petland, "Towards real-time recognition of driver intentions," in *Proc. IEEE Conf. Intell. Transport. Syst.*, 1997, pp. 236–241.

[74] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Inter. Joural of Comp. Vision*, vol. 60, no. 2, pp. 91–110, Oct. 2004.

[75] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models.* Springer, 2005.

[76] D. Makris and T. Ellis, "Learning semantic scene models from observing activity in visual surveillance," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 3, pp. 397–408, Jun. 2005.

[77] J. C. McCall and M. M. Trivedi, "Driver behavior and situation aware brake assistance for intelligent vehicles," *Proc. IEEE*, vol. 95, no. 2, pp. 374–387, Feb. 2007.

[78] J. C. McCall, D. Wipf, M. M. Trivedi, and B. Rao, "Lane change intent analysis using robust operators and sparse bayesian learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 431–440, Sep. 2007.

[79] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor, "Viewpoint independent detection of vehicle trajectories and lane geometry from uncalibrated traffic surveillance cameras," in *Int'l. Conf. Image Anal. and Recog.*, Oct. 2004, pp. 454–462.

[80] S. Messelodi, C. M. Modena, and M. Zanin, "A computer vision system for the detection and classification of vehicles at urban road intersections," *Pattern Anal. and Applications*, vol. 8, no. 1-2, pp. 17–31, Sep. 2005.

[81] D. Middleton and R. Parker, "Initial evaluation of selected detectors to replace inductive loops on freeways," Texas Transportation Institute, College Station, TX, FHWA/TX-00/1439, Apr. 2000.

[82] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 3, pp. 311–324, May 2007.

[83] B. Morris and M. Trivedi, "Improved vehicle classification in long traffic video by cooperating tracker and classifier modules," in *Proc. IEEE International Conference on Advanced Video and Signal based Surveillance*, Sydney, Australia, Nov. 2006, pp. 9–14.

[84] ——, "Robust classification and tracking of vehicles in traffic video streams," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Toronto, Canada, Sep. 2006, pp. 1078–1083.

[85] ——, "Real-time video based highway traffic measurement and performance monitoring," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Seattle, Washington, Sep. 2007, pp. 59–64.

[86] ——, "An adaptive scene description for activity analysis in surveillance video," in *Proc. IEEE Inter. Conf. on Pattern Recog.*, Tampa, Florida, Dec. 2008.

[87] ——, "Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis," in *Proc. IEEE International Conference on Advanced Video and Signal based Surveillance*, Santa Fe, New Mexico, Sep. 2008, pp. 154–161.

[88] ——, "Learning trajectory patterns by clustering:experimental studies and comparative evaluation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami, Florida, jun 2009, pp. 312–319.

[89] B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 425–437, Sep. 2008.

[90] ——, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1114–1127, Aug. 2008, Special Issue on Video Surveillance.

[91] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, vol. 14, sep 2002, pp. 849–856.

[92] J. Owens and A. Hunter, "Application of the self-organising map to trajectory classification," in *Proc. IEEE Visual Surveillance*, Jul. 2000, pp. 77–83.

[93] J. Pan and B. Hu, "Robust occlusion handling in object tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8.

[94] W. Pedrycz, *Knowledge-Based Clustering: From Data to Information Granules.* Hoboken, New Jersey: John Wily, 2005.

[95] C. Piciarelli and G. L. Foresti, "On-line trajectory clustering for anomalous events detection," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1835–1842, Nov. 2006.

[96] A. Polychronopoulos, M. Tsogas, A. Amditis, U. Scheunert, L. Andreone, and F. Tango, "Dynamic situation and threat assessment for collision warning systems: the EUCLIDE approach," in *Proc. IEEE Conf. Intell. Veh.*, 2004, pp. 636–641.

[97] F. Porikli, "Learning object trajectory patterns by spectral clustering," in *Proc. IEEE Inter. Conf. Multimedia and Expo*, vol. 2, Jun. 2004, pp. 1171–1174.

[98] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: Algorithms and evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 918–923, Jul. 2003.

[99] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

[100] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[101] W. Recker, M. Zhang, L. Chu, A. Chen, and M. McNally, "Development of methods and tools for managing traffic congestion in freeway corridors," in *Proc. IEEE Intell. Transport. Syst. Conf.*, Toronto, Canada, Sep. 2006, pp. 30–37.

[102] K. A. Redmill and B. A. Coifman, "System management and monitoring - temporal evaluation of freeway management systems," in *Proc. IEEE Intell. Transport. Syst. Conf.*, Toronto, Canada, Sep. 2006, pp. 254–260.

[103] E.-S. Ryu and C. Yoo, "Towards building large scale live media streaming framework for a U-city," *Multimedia Tools and Applications*, vol. 37, no. 3, pp. 319–338, May 2008.

[104] D. D. Salvucci, H. M. Mandalia, N. Kuge, and T. Yamamura, "Lane-change detection using a computational driver model," *Human Factors*, vol. 49, no. 3, pp. 532–542, 2007.

[105] T. Sato and M. Akamatsu, "Modeling and prediction of driver preparations for making a right turn based on vehicle velocity and traffic conditions while approaching an intersection," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 11, no. 4, pp. 242–258, 2008.

[106] N. Saunier, T. Sayed, and C. Lim, "Probabilistic collision prediction for vision-based automated road safety analysis," in *Proc. IEEE Conf. Intell. Transport. Syst.*, Seattle, Washington, Sep. 2007, pp. 872–878.

[107] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[108] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd ed. Pacific Grove, CA: Brooks/Cole, 1999.

[109] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 1999, pp. 246–252.

[110] ——, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, Aug. 2000.

[111] N. Sumpter and A. J. Bulpitt, "Learning spatio-temporal patterns for predicting object behavior," *Image and Vision Computing*, vol. 18, pp. 697–704, Jun. 2000.

[112] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 28, no. 5, pp. 694–711, May 2006.

[113] E. Swears, A. Hoogs, and A. Perera, "Learning motion patterns in surveillance video using hmm clustering," in *IEEE Workshop on Motion and video Computing*, Jan. 2008, pp. 1–8.

[114] M. M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 1, pp. 108–120, Mar. 2007.

[115] M. M. Trivedi, T. L. Gandhi, and K. S. Huang, "Distributed interactive video arrays for event capture and enhanced situational awareness," *IEEE Intell. Syst.*, vol. 20, no. 5, pp. 58–66, Sep. 2005.

[116] B. Tseng, C.-Y. Lin, and J. Smith, "Real-time video surveillance for traffic monitoring using virtual line analysis," in *IEEE Inter. Conf. on Multimedia and Expo*, vol. 2, 2002, pp. 541–544.

[117] (2009) FARS encyclopedia. U.S. Department of Transportation - National Highway Traffic Safety Administration. [Online]. Available: http://www-fars.nhtsa.dot.gov/Main/index.aspx

[118] (2008) Traffic monitoring guide. U.S. Department of Transportation - Office of Highway Policy Information. [Online]. Available: http://www.fhwa.dot.gov/ohim/tmguide/

[119] (2001) 2001 national household travel survey. U.S. Department of Transportation (DOT). [Online]. Available: http://nhts.ornl.gov/2001

[120] (2009) Highway statistics 2006 - FHWA. US Department of Transportation, Federal Highway Administration. [Online]. Available: http://www.fhwa.dot.gov/policy/ohim/hs06/index.htm

[121] (2006) Traffic safety facts 2006. U.S. National Highway Traffic Safety Administration. [Online]. Available: http://www-nrd.nhtsa.dot.gov/Pubs/TSF2006FE.PDF

[122] H. Veeraraghavan, O. Maoud, and N. Papanikolopulos, "Computer vision algorithms for intersection monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 2, pp. 78–89, Jun. 2003.

[123] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. IEEE Conf. on Data Engineering*, Feb. 2002, pp. 673–684.

[124] E. I. Vlahogianni, M. G. Karlaftis, J. C. Golias, and N. D. Kourbelis, "Pattern-based short-term urban traffic predictor," in *Proc. IEEE Intell. Transport. Syst. Conf.*, Toronto, Canada, Sep. 2006, pp. 389–393.

[125] X. Wang, K. T. Ma, G.-W. Ng, and W. Grimson, "Trajectory analysis and semantic region modeling using a nonparametric bayesian model," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8.

[126] X. Wang, X. Ma, and E. Grimson, "Unsupervised activity perception by hierarchical bayesian models," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8.

[127] X. Wang, X. Ma, and W. Grimson, "Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 539–555, Mar. 2009.

[128] (2009) Global status report on road safety. World Health Organization. [Online]. Available: http://www.who.int/violence_injury_prevention/road_safety_status/2009/en/index.html

[129] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

[130] T. Xiang and S. Gong, "Video behaviour profiling and abnormality detection without manual labeling," in *IEEE Inter. Conf. Computer Vision*, Beijing, China, Oct. 2005, pp. 1238–1245.

[131] ——, "Incremental and adaptive abnormal behaviour detection," *Computer Vision and Image Understanding*, 2008.

[132] ——, "Video behaviour profiling for anomaly detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 893–908, May 2008.

[133] W. Yan and D. A. Forsyth, "Learning the behavior of users in a public space through video tracking," in *IEEE Workshop on Application of Computer Vision.*, Brekenridge, CO,, Jan. 2005, pp. 370–377.

[134] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Journal of Computing Surveys*, vol. 38, no. 4, Dec. 2006.

[135] C. Zhang, X. Chen, and W. bang Chen, "A PCA-based vehicle classification framework," in *Proc. IEEE Data Engineering Workshops*, Apr. 2006.

[136] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Proc. IEEE Inter. Conf. on Pattern Recog.*, 2006, pp. 1135–1138.

[137] M. Zheng, T. Gotoh, and M. Shiohara, "A hierarchical algorithm for vehicle model type recognition on time-sequence road images," in *Proc. IEEE Intell. Transport. Syst. Conf.*, Toronto, Canada, Sep. 2006, pp. 542–547.

[138] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activities in video," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, jun 2004, pp. 819–826.

[139] L. Zhu, J. Song, Q. Huang, M. Zhang, and H. Liu, "A novel module of tracking vehicles with occlusion," in *Proc. IEEE Intell. Vehicles Symposium*, Jun. 2005, pp. 894–899.