Using Asymmetry in the Spectral Clustering of Trajectories

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Stefan Emilov Atev

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Nikolaos P. Papanikolopoulos

July 2011

Acknowledgements

I'd like to acknowledge my wife Elena, who always encouraged me and believed in this enterprise even when I had doubts. This thesis would not have happened without her constant support and unwavering belief in my capabilities. My parents Emil and Evgenia have always allowed me to follow my passions and instilled in me forever a love of learning.

I am indebted to my advisor Nikolaos Papanikolopoulos, for accepting me as a student and for indulging my frequent flirtation with topics that strayed from the projects I was working on. The journey may have been long, but thanks to his support I have never felt it to be any other than my own. Osama Masoud, the voice of reason in any academic endeavor I have undertaken, has contributed greatly to my understanding of many topics. Discussions with Gilad Lerman and his students have also helped me refine my ideas about numerous issues.

I would be remiss not to acknowledge my labmates, with whom I am lucky to have had only great relationships and fun times. I am especially indebted to Evan Ribnick, Ajay Joshi, and Nathaniel Bird, who have, in one way or another, contributed to my thinking on the problems discussed in this thesis, as well as everything else.

The computer science and mathematics faculty at Luther College has left an indelible mark on me, as has Veliko Kolev, my inimitable high school math teacher. I also thank Boyko Banchev for his teaching and for never treating me like the teenager I was...

To Konstantin and Elena.

Abstract

Spatial trajectories, for example those of vehicles passing through a traffic intersection, are of interest in many data collection applications since they capture a lot of semantic information in a fairly compact representation. Data mining, or unsupervised learning from sets of trajectories can be challenging since intuitive notions of trajectory similarity are hard to encode rigorously. Many similarity measures for trajectories that are needed for tasks such as clustering fail to satisfy basic metric properties like the triangle inequality, or even symmetry. While in some simple practical applications such measures have been quite successful, the violation of basic properties poses unique challenges for more advanced methods such as spectral clustering. We show how the asymmetry of a trajectory similarity measure can be exploited when clustering a set of trajectories. The asymmetry is used both indirectly, in a traditional spectral clustering method, and directly, by developing a spectral clustering method that can handle asymmetric affinity matrices natively without requiring an artificial symmetrization step to be performed, thus avoiding the attendant loss of information entailed by that process. We propose a modification of the Hausdorff distance for comparing trajectories, which we first symmetrize in a non-standard fashion inspired by a local scaling approach, and then further show that the distance can be used directly without prior symmetrization. We perform a variety of experiments, with a focus on vehicle trajectories. A novel automated tracking method is developed to provide experimental data.

Table of Contents

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

2 Related Work **4**

2.1 Automated Tracking 4

2.2 Trajectory Comparison and Clustering 5

2.3 Spectral Clustering and Kernel Methods 7

2.4 Kernels for Time Series, Sequences, and Shapes 10

3 Background **12**

3.1 Time Series Similarity 12

3.1.1 Dynamic Time Warping 12

3.1.2 Longest Common Sub-Sequence 16

3.1.3 The Hausdorff distance 19

3.2 Spectral Clustering 20

3.2.1 Overview 20

3.2.2 Summary of Multiway Spectral Clustering 21

3.3 Evaluating Clustering Algorithms 24

3.3.1 Notation 24

3.3.2 Comparing Clusterings 24

3.3.3 Confusion Matrix 25

3.3.4 Variation of Information 25

3.3.5 Clustering Error 26

4 Vehicle Tracking **27**
4.1 Introduction . . . 27
4.2 Low-Level Vision . . . 29
4.2.1 Illumination Filter and Background Model Maintenance . . . 29
4.2.2 Camera Shaking Compensation . . . 31
4.2.3 Noise Removal . . . 32
4.3 Image-Level Tracking . . . 34
4.3.1 Image-Space Tracker . . . 34
4.4 Ground-Plane Tracking . . . 35
4.4.1 Scene Calibration . . . 37
4.4.2 Vehicle Model . . . 39
4.4.3 Measurement Model . . . 43
4.4.4 Constrained State Estimation . . . 46
4.4.5 Tracking . . . 47
4.5 Results . . . 48
4.5.1 Single-View Results . . . 48
4.5.2 Multi-View Results . . . 49
4.5.3 Data for Clustering Experiments . . . 50

5 Modified Hausdorff Distance for Trajectory Clustering **57**
5.1 Introduction . . . 57
5.2 A Modified Hausdorff Distance . . . 58
5.2.1 Connections with DTW and LCSS . . . 63
5.3 Clustering Methods . . . 64
5.3.1 Agglomerative Clustering . . . 64
5.3.2 Spectral Clustering Modifications for Trajectory Clustering . . . 64
5.4 Experiments . . . 66
5.4.1 Results . . . 66
5.4.2 Discussion . . . 67
5.4.3 Time Cost of the Methods . . . 69
5.4.4 Sensitivity of the Agglomerative Clustering . . . 69
5.5 Conclusions . . . 70

6 Clustering with Asymmetric Similarity Measures **74**
6.1 Introduction 74
6.2 Related Work 76
6.2.1 Spectral Decomposition with Asymmetric Affinities 77
6.2.2 Eigenpair Selection and Embedding 78
6.2.3 Cluster Assignment 78
6.3 The k-Axes Algorithm 79
6.3.1 Minimization with Orthogonality Constraints 81
6.3.2 Periodicity Order of the Objective 83
6.3.3 Convergence Speed of Algorithm 1 83
6.4 Experiments 84
6.4.1 Local Scaling and Model Selection 85
6.4.2 Preserving Asymmetry in Clustering 86
6.4.3 Using Locally Scaled, Non-Symmetrically Normalized Affinity . 87
6.4.4 Pen Trajectories 89
6.4.5 Vehicle Trajectories 91
6.4.6 Comparison with Global Alignment Kernel 93
6.5 Discussion and Conclusions 95

7 Conclusions **99**
7.1 Contributions 99
7.2 Future Work 101

Bibliography **102**

Appendix: Matlab Implementation of the k-Axes Algorithm **113**

List of Tables

3.1 Common spectral decomposition steps 22
3.2 Common spectral embedding steps 23
3.3 Example confusion matrices and error measures 26

4.1 Summary of vehicle state constraints 47
4.2 Position and dimension measurement accuracy 48

5.1 Comparisons with ground-truth for vehicle trajectories 72
5.2 Summary of methods evaluated in Table 5.1 73
5.3 Running time of the different clustering methods 73

6.1 Iteration counts for convergence on selected data sets 87
6.2 Methods evaluated with synthetic affinity matrices 87
6.3 Results from experiment with synthetic affinity matrices 89
6.4 Comparison of clustering with Fischer and Poland [39] 97
6.5 Comparison of clustering methods on traffic data 98

List of Figures

3.1 Illustration of a dynamic time warp between two sequences 17
3.2 The Sakoe-Chiba and Itakura DTW constraints 18
3.3 Illustration of a longest common subsequence match 19
3.4 Illustration of a directed Hausdorff distance match 20
3.5 Illustration of spectral embedding in spectral clustering 23

4.1 Low-level vision system diagram 29
4.2 Video stabilization results 33
4.3 Noise removal examples 34
4.4 Noise removal structuring elements 35
4.5 State and input character label diagram with example 36
4.6 Connected region interactions 37
4.7 Typical traffic camera views 38
4.8 State variables in vehicle model 40
4.9 Illustration of measurement model 44
4.10 Single-view tracking results 49
4.11 Multi-view tracking results 52
4.12 Additional multi-view tracking results 53
4.13 Vehicle trajectory data sets used in experiments 54
4.14 Histogram of trajectory length distribution 55
4.15 Top-down view of ground-truth data 55
4.16 Camera-angle view of ground-truth data 56

5.1 Counterintuitive results with the Hausdorff distance 59
5.2 Illustration of the correspondence function and neighborhood structure 61
5.3 Distortion measure plot for data set 1 65
5.4 Results for automatic spectral clustering with the Hausdorff distance . 67

5.5 Sensitivity of agglomerative methods to the cluster distance threshold 70

6.1 Sample asymmetric affinity for handwritten characters 75
6.2 Search direction for conjugate gradient 85
6.3 Detain on convergence speed on the Iris data set 86
6.4 Results on datasets from Zelnik-Manor and Perona [123] 88
6.5 Sample synthetic affinity matrices 89
6.6 Selected data sets from Fischer and Poland [39] 90
6.7 Handwritten characters data set 90
6.8 Affinities for handwritten characters 91
6.9 Comparison of k-Axes and normalized cuts on character trajectories . . 92
6.10 Ground truth for vehicle trajectory experiment with k-Axes 93
6.11 Comparison of k-Axes and normalized cuts on vehicle trajectories . . . 94
6.12 Comparison with the time-alignment kernel of Cuturi et al. [34] 95

Chapter 1

Introduction

The overall problem addressed by this thesis is how to extract some structure from a set of spatial trajectories, in particular vehicle trajectories automatically computed from video. The immense challenges faced by any practical trajectory extraction method are often caused by ambiguities that cannot be resolved without prior information. Briefly, the challenges in tracking are due to target initialization problems, static and dynamic occlusions, the inability of simple dynamic models to predict driver behavior at traffic intersections, and due to the unpredictable nature of environmental change in outdoor scenes over extended periods.

That large set of previously extracted vehicle trajectories can be a powerful prior in aid of tracking is demonstrated by Magee [75] and Hsieh et al. [53]. Anomaly detection requires that a background, or "normal" model is learned, and Hu et al. [54] and Fu et al. [41] show the utility of using trajectories as the basic feature of such models in the context of traffic monitoring. The influential work of Stauffer and Grimson [105] provides yet another example that extraction and clustering of object trajectories can support event detection in a surveillance scenario.

The tracking method detailed in Chapter 4 does not use prior information in the form of learned trajectories, but is rather intended to allow us to bootstrap the process by providing reasonably reliable trajectories from which we may learn a model for the trajectories in a scene. For such purposes, tracking failures are acceptable as long as the successful tracking remains representative of the true distribution of trajectories in the scene. A distinguishing feature of the method we present is that it can estimate the real-world dimensions of vehicles, which makes outlier filtering easier. The method also uses a dynamic model fully specified in real-world units so parameter tuning

need not be performed anew for novel scenes. The measurement model used by the proposed tracker allows us to fuse measurements from overlapping camera views, and to partially update the state estimates of tracked vehicles while they undergo occlusions, even in the single-view case.

The output of any tracking method is a set of trajectories. This is the starting point for Chapter 5, where we present a modified Hausdorff distance for comparing trajectories. We show that the modifications are quite successful in addressing problems with the use of the unmodified Hausdorff distance, identified in works such as those by Zhang et al. [126] and Antonini and Thiran [3]. We show that the distance is quite effective when used in a spectral clustering framework, and demonstrate how to exploit the directed Hausdorff distance to improve the affinity matrix used in the clustering.

The experiments in Chapter 5 compare the proposed modifications to the Hausdorff distance with the two most commonly proposed non-parametric measures for comparing time series and trajectories of unequal lengths — the Dynamic Time Warping distance and the Longest Common Subsequence distance. Since we evaluate these in a spectral clustering framework as well, we effectively cover the majority of kernels for trajectories proposed in the literature for use with support vector machines and other kernel methods.

The results of Chapter 5 are satisfactory, but the observation that a lot of classical trajectory distance measures are either directed, or have directed variants led to the speculation that using the asymmetry directly can aid the clustering of trajectories. This intuition is supported by the observation that asymmetric distances are used to great success in speech applications, for example. We find this to be the case indeed, and Chapter 6 shows how any asymmetric similarity measure (asymmetric kernel) can be used in a spectral clustering framework. We construct a Hermitian kernel from the underlying asymmetric similarity measure which preserves all information (the map between the asymmetric kernel and the Hermitian one is bijective). We then show how to cluster with complex spectral embedding coordinates using a method that is inspired both by the "rotation" method in Zelnik-Manor and Perona [123], and the k-lines method proposed by Fischer and Poland [39] as an improvement to the final clustering step in spectral methods.

The resulting algorithm for clustering presented in Chapter 6 can work with asymmetric similarity measures, but is also shown to be competitive with other common

spectral clustering methods even in the symmetric case, which is just a special case for the method. The use of a Hermitian affinity has been proposed in the context of social network analysis by Hoser and Geyer-Schulz [51], but we believe that this idea can be applied more generally in any kernel method. Since asymmetric similarity measures are quite frequently used in speech recognition and in biological sequence analysis, the k-Axes algorithm has potentially wide applicability.

The contributions of the thesis encompass three broad areas: vehicle tracking, spectral clustering of vehicle trajectories using a modified Hausdorff distance, and generalization of spectral clustering to support non-symmetric similarity measures. In the area of vehicle tracking, the contributions are in the geometric measurement model, the motion model in real world coordinates, and the application of a constrained Extended Kalman Filter. A modified Hausdorff distance adapted for trajectory comparison is the first contribution in the area of trajectory clustering, followed by the idea of directed local scaling for spectral clustering and the extensive comparison with other commonly used distance measures. On the topic of spectral clustering with asymmetric similarities, the contributions are in the transformation of the asymmetric similarity into a Hermitian similarity, the k-Axes algorithm for clustering of complex-valued spectral images, as well as the application of advanced optimization methods in the implementation of k-Axes.

Chapter 2

Related Work

In this section, we discuss a number of references relevant to the subsequent chapters. This survey is not exhaustive, and a number of important references will be discussed very briefly here. Note in particular, that our discussion of trajectories is mostly concerned with vehicle trajectories in the context of intelligent transportation systems.

2.1 Automated Tracking

Tracking of objects in video is addressed by many researchers, but the most relevant part for the experiments in this thesis are fully automated methods that do not require manual target initialization. This rules out many active-contour and appearance-based trackers that normally require manual user interaction — in such situations the benefits of unsupervised methods are less pronounced since the additional effort necessary in labeling a trajectory is similar to the effort in manually initializing a target in the first place. Tracking is relevant to this work as it provides both an application that can benefit from learning scene structure, motion models and other specifics, but also because tracking data is a good source of noisy, multivariate time series of various lengths. Automatically extracted motion data typically has many outliers, and a challenging set of trajectories is important to ensure that the developed methods are robust. For a survey with extensive bibliography on the subject of video processing methods for ITS applications, one can refer to Kastrinaki et al. [64].

An early example of a complete vehicle tracking system is Koller et al. [69, 70], where contours extracted from the optical flow field represent tracked vehicles. An

Extended Kalman Filter (Welch and Bishop [118]) is used to estimate the state of the vehicles under an affine motion model, and a dynamic belief-network method is used to detect traffic events. Such region-based trackers using the Kalman filter and its variants are not uncommon for traffic monitoring. In Badenas et al. [11], the tracker estimates region statistics such as center position, grayscale image intensity and velocity, and in Lou et al. [74], a sophisticated vehicle motion model that accounts for steering geometry is considered. Model-based methods are also employed for vehicle tracking, as by Kim and Malik [68], who consider a parametric model suitable for the tracking of light vehicles (i.e., cars, but not vans and trucks). When camera calibration is available, trackers may provide trajectory information in real world coordinates, as in Coifman et al. [28].

A multiple-hypothesis region based tracker is presented by Buzan et al. [20], who also consider the problem of clustering the obtained motion trajectories using a non-metric similarity measure. Tracking plays a role in the work of Morris and Trivedi [81], where the trajectory information aids vehicle classification. A relevant discussion of how trajectory information can improve tracking performance is provided in Magee [75], where previously observed trajectories provide information that disambiguates subsequent measurements. In similar vein, Hsieh et al. [53] uses trajectory clustering to determine lane boundaries in a traffic video. The boundaries aid shadow detection and elimination, which improves subsequent tracking and target detection.

Applications that can benefit from unsupervised learning of trajectory patterns are demonstrated by Hu et al. [54], for use in an accident detection and prediction system. The extraction and clustering of trajectories is used for event detection by Stauffer and Grimson [105]. A trajectory prior learned through clustering also enables anomaly detection, as exemplified in Fu et al. [41], who use the spectral clustering method of Ng et al. [83] without modifications.

2.2 Trajectory Comparison and Clustering

Works in the data mining and data management fields are chiefly concerned with deriving dissimilarity measures on time series with specific invariance properties, but use relatively simple clustering and indexing schemes. Examples of this approach are Buzan et al. [20], which uses the non-metric Longest Common Sub-Sequence (LCSS) dissimilarity measure to agglomeratively cluster vehicle trajectories. Another

non-metric measure, the Edit Distance on Real Sequences (EDRS) is presented in Chen et al. [23, 24], who also discuss strategies to reduce search effort using the measure in large trajectory databases.

Another popular distance measure for trajectories and general time series is Dynamic Time Warping (DTW), exemplified in Vlachos et al. [115]. In earlier work, Vlachos et al. [113] argued in favor of LCSS over DTW, a view shared by Duchene et al. [38], but in the specific case of [115] DTW is applied to data transformed explicitly to a rotation-invariant representation where the advantages of LCSS are not as pronounced. A modification of DTW that addresses some of its perceived deficiencies is presented in Keogh and Pazzani [65], where DTW is applied to the derivative of the time series in order to avoid "pathological results". Grid-based methods for trajectory comparison and clustering are discussed in Meratnia and de By [80], who also point out issues arising from partial trajectory matching, such as violations of the triangle inequality for distance measures. In Antonini and Thiran [3], trajectory clustering is used for pedestrian counting, and LCSS is compared to the Hausdorff distance. When outliers are accounted for, the authors find the two measures to perform similarly.

Hidden Markov Models are sometimes used to represent sequences, as this converts uneven-length sequences to a representation with a fixed number of parameters. An example of clustering using the HMM representation are Oates et al. [84] (who use DTW to bootstrap their HMM-based algorithm), Alon et al. [2], and Porikli [91]. While [2] uses Expectation Maximization (EM) with soft-assignment for clustering and the Minimum Description Length principle for model selection, [91] uses a spectral clustering method based on thresholding a derived correlation matrix. The EM algorithm (and the special case of k-Means) are considered by Lin et al. [72] for multi-scale time-series clustering using wavelets, and by Chudova et al. [27] for clustering of a more generic class of time series. The use of multi-scale approaches in this field has been motivated by the desire to avoid local extrema when using local optimization methods such as EM, a problem that is not as pronounced in spectral methods which tend to be convex relaxations to k-means-like cost functions for which the computation of a globally optimal solution is tractable. Another EM-based algorithm is used by Gaffney and Smyth [42] to cluster collections of time series modeled as a mixture of regression models, with some proposed extensions to handle the multivariate case and non-parametric regression. The modeling of univariate time series by auto-regressive models is exemplified by Xiong and Yeung [121], but apart from treating

each dimension separately, the multivariate case is not considered. Finally, Kalpakis et al. [63] considers the Discrete Wavelet Transform (DWT), Principal Component Analysis (PCA), the Discrete Fourier Transform (DFT), and linear predictive coding coefficients as normalized (i.e., fixed-length) representations of trajectories for the clustering of Auto-Regressive, Integrated Moving Average (ARIMA) time series. The need for compact representation and fast retrieval of trajectory data has resulted in dimensionality reduction approaches such as the segmented PCA in Bashir et al. [12].

2.3 Spectral Clustering and Kernel Methods

A wide variety of non-linear dimensionality reduction, classification and clustering methods can be formulated as linear algorithms in a high-dimensional feature space. Kernel methods refer to algorithms that represent the input-to-feature space maps implicitly through a kernel function satisfying a number of constraints. A very approachable introduction to kernel methods can be found in Schölkopf and Smola [97], which covers both supervised and unsupervised methods. Another introduction is by Shawe-Taylor and Christiani [100], which includes code fragments and pseudo-code descriptions of many common basic operations that serve as building blocks for kernel methods. A third of the book is devoted to various domain-specific and general kernels. A discussion of kernel methods from the point of view of Gaussian Processes can be found in Rasmussen and Williams [93].

As we are mostly concerned with unsupervised learning, the focus of this review is mainly on the application of kernel methods to clustering, where many competing methods have demonstrated good performance (Verma and Meilă [108]). A number of early spectral clustering methods are analyzed in Weiss [117], and their performance explained using perturbation bounds of the pairwise similarity matrices used in the respective methods (another justification of these algorithms' performance is given by Brand and Huang [19]). The early motivation for spectral clustering comes from results in spectral graph theory, summarized for example in van den Heuvel and Pejić [107], which give information about the component structure of a weighted graph based on the eigen-spectrum of its edge-weight matrix. We will discuss the now-classic algorithm of Ng et al. [83], the locally-scaled algorithm of Zelnik-Manor and Perona [123] and other related methods in more detail in Section 3.2 background. The relationship between spectral clustering and kernel PCA is well known, and "spectral

clustering" kernels have been proposed (Bengio et al. [15], Jenssen et al. [61]).

The clustering problem can be considered as one of finding graph cuts with special properties, a fact recognized by many researchers (e.g., Meilă and Shi [79], Shi and Malik [101]). In Dhillon et al. [35], spectral approximations to various normalized, average, and ratio cuts are discussed, and the connection between spectral methods and kernel k-Means is made explicit.

The connection to weighted kernel k-Means is important as several trajectory clustering methods discussed in Section 2.2 are based on the Expectation Maximization (EM) algorithm applied to non-metric distances. The work of [27, 42, 72] can be related to spectral and kernel methods through the results in [35], and the early work on kernelized k-means in Girolami [45]. Results on using EM for mixture estimation also relate to spectral methods, and some connections are explored in Narayanan et al. [82].

The view of kernel methods as a non-linear extension of traditional linear methods such as Principal Component Analysis is the main focus of Schölkopf et al. [98], where kernel PCA is introduced. As Ham et al. [48] show, a number of successful non-linear dimensionality reduction and clustering methods such as Locally Linear Embedding (LLE, Roweis and Saul [94]), Metric Multidimensional Scaling (MDS, Cox and Cox [31]), Isomap (Tenenbaum et al. [106]), and Laplacian Eigenmaps (Belkin and Niyogi [13]), are essentially variants of kernel PCA with different approaches to the normalization and construction of the kernel Gram matrix.

As traditional kernel PCA requires positive definite kernels, a number of approaches have been developed to handle the case of non-positive definite and semi-metric kernels. One approach, exemplified by Pękalska et al. [88], is to extend the feature map to a pseudo-Euclidean space. Another approach is to modify the kernel matrix so that it becomes positive definite — a few of those methods are discussed in Wu et al. [119]. In Fowlkes et al. [40], which we discuss later, non-positive-definite kernels are handled by adding an additional orthogonalization step in the diagonalization of the Gramian. Negative eigenvalues in the spectrum of the kernel Gram matrix pose both interpretation and implementation challenges for kernel methods. In supervised methods, negative eigenvalues may complicate or prevent the use of efficient optimization techniques, a problem that is less common for kernel-based clustering methods. The interpretation of negative eigenvalues, especially ones of large absolute magnitude is problematic for both supervised and unsupervised methods.

Since the eigenvalues and eigenvectors of a Gram matrix are sought, the computational complexity of kernel PCA can grow cubically in the number of data points (and since those matrices are often diagonally dominant, methods such as Lanczos (Golub and Loan [46]) can fail to converge fast). What is worse, to find the low-dimensional coordinates of a novel data point using a kernel method, one may be forced to perform such a costly decomposition repeatedly, with a growing Gramian. A solution to both problems is to find a suitable approximation that lets one find the embedding of data points that are not in the input collection used to compute the Gramian. As Belongie et al. [14], Fowlkes et al. [40] propose, only a subset of the data points can be used to form a Gramian, and an out-of-sample extension motivated by the Nyström approximation method can be used to extend the embedding to the remaining data points. Out-of-sample extensions based on the Nyström method for LLE, Isomap, Laplacian Eigenmaps, MDS and spectral clustering are detailed in [15]. As Platt [90] shows, several approximate MDS methods such as Landmark MDS and MetricMap are in fact also based on the Nyström approximation.

Greedy methods are sometimes used to speed up kernel methods. One example is the Incomplete Cholesky Decomposition of the kernel matrix, discussed by Bach and Jordan [10], who also show how to use side information improve the low-rank Gramian approximation. Many ideas for sparse and greedy implementations are discussed in [97] and references therein. Since k-Means is a greedy algorithm, the kernelized k-Means of [35, 45] fall in the same category.

Approximate incremental methods can address the computational cost of kernel methods as well. The efficient incremental SVD method of Brand [17, 18] can be "kernelized", as proposed in Chin et al. [26]. Incremental versions of kernel PCA based on the Generalized Hebbian Algorithm (GHA) have been recently proposed by Kim et al. [66, 67], while Chin and Suter [25] modify an incremental linear PCA algorithm.

The problem of selecting a subset of the data points that would adequately represent the variability of the data is becoming more and more important as kernel methods are applied to increasingly larger problems. The sampling approach, based on either side information [40], uniform sampling (Pękalska et al. [89]), or optimal sampling (Drineas and Mahoney [36]), is one avenue open for exploration in addition to the consideration of deterministic subset-selection approaches.

Kernels for time-series can be constructed in a generic fashion when a generative model for the trajectories is available though constructions such as the as the Fisher

Kernel of Jaakkola and Haussler [59, 60]. Kernels exist for text classification (Lodhi et al. [73]), HMMs (Shawe-Taylor and Christiani [100], Vert et al. [110]), trees and graphs (Gärtner et al. [43], Vert [109], Vishwanathan and Smola [111]). Few kernels apart those mentioned in Section 2.4 have been proposed for general time series. A statistics perspective on various kernels is given in Genton [44], which also summarizes the various methods in which kernels can be combined.

Many methods have been proposed to reduce the cost of spectral clustering and related kernel methods, either through smart sampling (Drineas and Mahoney [36], Zhang et al. [125]), low-rank approximations (Ouimet and Bengio [85], Zhang and Kwok [124]), or numerical approximation methods such as in Schraudolph et al. [99].

2.4 Kernels for Time Series, Sequences, and Shapes

The use of kernel methods for time series comparison has been investigated in the speech-recognition and bioinformatics communities, where dynamic time warping and various edit distances are well-known. It must be highlighted that most methods in this section have been experimentally validated on strings over finite alphabets or univariate time series. Several shape-statistics and shape learning methods have made use of kernel methods as well, and the objects of interest there are very similar to trajectories (the biggest difference being that shapes are typically closed contours, not open, which precludes the use of level-set embedding methods for the representation and manipulation of trajectories).

A DTW-inspired kernel for use in speech-recognition is proposed by Shimodaira et al. [102]. Rather than minimizing a distance between two aligned sequences, [102] maximize an inner product between the sequences over all possible alignments. This construction leads to a kernel that is symmetric and which satisfies the Cauchy-Schwartz inequality, but is not positive definite. As with other kernels based directly on an underlying time sequence distance, the end result is rarely positive definite, as [110] and Cortes et al. [29, 30] establish. The time-alignment kernel of Cuturi et al. [34] is positive definite, but that is achieved at the cost of replacing a maximum with a soft-max operator. The time-alignment kernel is composed by aggregation from a basic kernel, which must satisfy somewhat awkward constraints that do not hold for common kernels such as the Gaussian RBF. We will investigate some of the properties of this time-alignment kernel, even though its actual application in [34] is through the

non-positive logarithm of the kernel. Some strategies for the kernelization of LCSS are presented in Gruber et al. [47], and as identified by the authors, none of these strategies produces a positive definite kernel. The global kernel proposed in [47] is very similar to the use of LCSS with spectral clustering in Chapter 5.

Kernels on structured data, of which strings are one specific instance, have been investigated actively. The convolution kernel of Haussler [49] is an example of a family of kernels defined whenever a class of objects can be decomposed into a fixed number of parts (even if there are multiple decompositions). The time-alignment kernel of [34] bears a lot of similarity with convolution kernels since it is built up from all possible alignments between two sequences. Other structured data kernels that provide intuition on kernel construction are graph kernels, exemplified in Vishwanathan et al. [112]. The semi-group kernels of Cuturi et al. [33] and the rational kernels of [30] both extend the methods of [49] and describe families of kernel functions with desirable properties (though not all positive).

More model-oriented kernels between series can also be constructed from probabilistic models through the Fisher kernel of Jaakkola et al. [58], also explained in [60]. We will not consider this family of kernels since a probabilistic model of the underlying series is assumed to be available.

A kernel method for shape priors is proposed by Cremers et al. [32], who use shape statistics for variational image segmentation. The methods discussed in the work discuss a Mahalanobis distance in feature space. Another approach to incorporating second-order statistics in shape density estimation is proposed by Charpiat et al. [22] who compute an empirical mean and variance on a shape manifold using partial differential equations. This necessitates that the manifold and its geometry be defined by differentiable metrics, and in earlier work the authors propose differentiable approximations to the Hausdorff distance that are relevant for trajectory kernels as well Charpiat et al. [21].

Chapter 3

Background

3.1 Time Series Similarity

This section provides the necessary background on various dissimilarity measures between time series and point sets. Some of the measures were originally proposed in the univariate case, or only considered the natural (Euclidean) norm, so the discussion below will generalize them whenever possible. Finally, several of the measures were originally intended for comparing strings (sequences with elements from a finite set), and were later extended to alphabets over metric spaces.

3.1.1 Dynamic Time Warping

Many methods for comparison of uneven-length sequences fall in the category of Dynamic Time Warping methods (DTW). Since there is not one standard DTW method, we will focus on one particular variant, and point out the design choices that can be modified to derive variants of DTW. The description of DTW in Kruskal and Liberman [71] is particularly appealing since the method is presented in the continuous case, and then discretized. The same reference discusses both symmetric and directed variants of DTW, which will be relevant in Chapter 6. Finally, the discussion in [71] provides some points of reference with respect to the Edit distances discussed later on.

Continuous Case

We will initially consider two continuous-time trajectories, $\mathbf{x}(t_x)$ and $\mathbf{y}(t_y)$, traversing the same region of the ambient space $\mathscr{T}$, but with different parameterizations. The time parameters t_x and t_y vary from 0 to T_x and T_y respectively. The trajectories $\mathbf{x}$ and $\mathbf{y}$ are connected by a time-warping (u, v) if $u(t)$ and $v(t)$ are two continuous, monotonically increasing real-valued functions such that:

$$\mathbf{x}(u(t)) = \mathbf{y}(v(t)) \qquad 0 \leq t \leq T \tag{3.1.1}$$

$$u(0) = 0 \tag{3.1.2}$$

$$v(0) = 0 \tag{3.1.3}$$

$$u(T) = T_x \tag{3.1.4}$$

$$v(T) = T_y, \tag{3.1.5}$$

where the last four constraints are one way to ensure that no degenerate warpings are considered.

A time warping induces a correspondence between the elements of the series and $\mathbf{x}$ and $\mathbf{y}$, and only the correspondence has an intrinsic meaning. That is to say, the time warping (u, v) induces the same correspondence as $(u \circ f, v \circ f)$, for an arbitrary monotonically increasing function f such that $f(0) = 0$ and $f(s) = T$ for an arbitrary positive time endpoint s.

In practice, we are concerned with *approximate* time warpings, where the condition $\mathbf{x}(u(t)) = \mathbf{y}(v(t))$ holds only in some approximate sense (i.e., $d_{\mathscr{T}}(\mathbf{x}(u(t)), \mathbf{y}(v(t)))$ should be small). With that consideration, an optimal approximate time warp is a pair of functions (u^*, v^*) that minimize the functional:

$$J_{\text{DTW}} = \int_0^T g\left(d_{\mathscr{T}}(\mathbf{x}(u(t)), \mathbf{y}(v(t)))\right) \mathrm{d}\, t. \tag{3.1.6}$$

The transformation g applied to the metric $d_{\mathscr{T}}$ must be such that J_{DTW} is invariant to differences in warps that induce the same correspondence. As [71] points out, the choice $g(x(t)) = x(t)M(u'(t), v'(t))$ is invariant and results in a symmetric distance for a wide variety of generalized means M, such as the arithmetic, geometric, or weighted (with weights $(u^{-1}(T), v^{-1}(T)$, for example). A directed distance can also be constructed, by using only $u'(t)$ or $v'(t)$ as a scale normalization factor in g.

It is unclear if the freedom in choosing a generalized mean M in (3.1.6) leads to many functionally different DTW variants. We follow [71] in defining only two variants, one symmetric, and one directed. The symmetric variant of DTW minimizes:

$$J_{\text{sym}}(u,v) = \int_0^T d_{\mathscr{T}}\left(\mathbf{x}(u(t)), \mathbf{y}(v(t))\right) \frac{u'(t)+v'(t)}{2}\,\mathrm{d}\,t, \tag{3.1.7}$$

and the directed version is:

$$J_{\text{dir}}(u,v) = \int_0^T d_{\mathscr{T}}\left(\mathbf{x}(u(t)), \mathbf{y}(v(t))\right) u'(t)\,\mathrm{d}\,t. \tag{3.1.8}$$

Discrete Case

In practical applications, trajectories are observed at discrete time points, so the definitions we have developed so far do not apply directly. In particular, (3.1.6) must be discretized, and a discrete analog of the continuity constraints on the warping functions must be developed.

Let $t_{x,1}, t_{x,2}, \ldots, t_{x,n}$ denote the times at which $\mathbf{x}$ is sampled, and similarly, let $t_{y,1}, \ldots, t_{y,m}$ denote the sampling times of $\mathbf{y}$. The assumptions about the times are:

$$i < j \iff t_{x,i} < t_{x,j} \qquad 1 \le i, j \le n \tag{3.1.9}$$

$$i < j \iff t_{y,i} < t_{y,j} \qquad 1 \le i, j \le m, \tag{3.1.10}$$

that is, the times are distinct and increasing with respect to their index.

The discrete analog of the time warp (u,v) is a pair of integer-valued non-decreasing functions p and q whose ranges are the index sets $[1,n]$ and $[1,m]$ respectively. Both p and q are defined on the index set $[1,k]$. The continuity constraint is most commonly expressed as:

$$(p(i+1), q(i+1)) = \begin{cases} (p(i)+1, q(i)) \\ (p(i), q(i)+1) \\ (p(i)+1, q(i)+1) \end{cases} \qquad 1 \le i < k. \tag{3.1.11}$$

As [71] points out, (3.1.11) has several variants, some of which impose constraints on more than two consecutive steps of the time warp. We will focus our attention

only on the variant described above, which is the most common outside of the speech recognition community. The continuous case constraints (3.1.2)–(3.1.5) are translated as:

$$p(1) = 1 \qquad p(k) = n \tag{3.1.12}$$

$$q(1) = 1 \qquad q(k) = m. \tag{3.1.13}$$

The final choice we have to make to produce a practical description of DTW is to specify how (3.1.6) must be discretized. Since both the symmetric and directed variants require evaluation of derivatives, we have a number of possible choices to make for the discretization of the differentials irrespective of how we evaluate the integral. The choices that lead to the most common DTW discrete variants are to use backward differences for the derivatives, and the right rectangle rule for the integral. Specifically:

$$f'(t_i) \triangleq \frac{f(t_i) - f(t_{i-1})}{t_i - t_{i-1}} \tag{3.1.14}$$

$$\int_a^b f(t)\,\mathrm{d}\,t \triangleq f(b)(b-a), \tag{3.1.15}$$

with which definitions we can write the discrete symmetric distance functional as:

$$\sum_{i=2}^{k} d_{\mathscr{T}}\left(\mathbf{x}(t_{x,p(i)}), \mathbf{y}(t_{y,q(i)})\right) \frac{1}{2}\left(\frac{p(i) - p(i-1)}{t_{x,p(i)} - t_{x,p(i-1)}} + \frac{q(i) - q(i-1)}{t_{y,q(i)} - t_{y,q(i-1)}}\right), \tag{3.1.16}$$

and the discrete equivalent of (3.1.8) as:

$$\sum_{i=2}^{k} d_{\mathscr{T}}\left(\mathbf{x}(t_{x,p(i)}), \mathbf{y}(t_{y,q(i)})\right) \frac{p(i) - p(i-1)}{t_{x,p(i)} - t_{x,p(i-1)}}, \tag{3.1.17}$$

The optimization of (3.1.16) over all possible time warps (p, q) is computed using dynamic programming. If $D_{i,j}$ denotes the optimal time warp cost between the partial sequences formed by $\mathbf{x}$'s first i elements and $\mathbf{y}$'s first j elements, we can use the

following recurrence to solve the minimization problem:

$$D_{i,j} = \min \begin{cases} D_{i-1,j} & + \frac{1}{2} d_{\mathscr{T}} \left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j}) \right) \frac{1}{t_{x,i} - t_{x,i-1}}, \\ D_{i,j-1} & + \frac{1}{2} d_{\mathscr{T}} \left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j}) \right) \frac{1}{t_{y,j} - t_{y,j-1}}, \\ D_{i-1,j-1} & + \frac{1}{2} d_{\mathscr{T}} \left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j}) \right) \left(\frac{1}{t_{x,i} - t_{x,i-1}} + \frac{1}{t_{y,j} - t_{y,j-1}} \right), \end{cases} \tag{3.1.18}$$

while the minimization of (3.1.17) is achieved with the recurrence:

$$D_{i,j} = \min \begin{cases} D_{i-1,j} & + d_{\mathscr{T}} \left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j}) \right) \frac{1}{t_{x,i} - t_{x,i-1}}, \\ D_{i,j-1} & , \\ D_{i-1,j-1} & + d_{\mathscr{T}} \left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j}) \right) \frac{1}{t_{x,i} - t_{x,i-1}}. \end{cases} \tag{3.1.19}$$

Note that the value $D_{1,1}$ does not affect the form of the solution and can be set to 0. It is important to remark that some descriptions of DTW do not weigh the metric $d_{\mathscr{T}}$ differently in the three cases of the recurrences (3.1.18) and (3.1.19). This "slope-weighting" (as the practice of scaling the off-diagonal transitions is called), is not an *ad hoc* attempt to bias the solutions towards the diagonal (as [65] imply), but an artifact of the discretization of the continuous DTW.

Variants and Properties of DTW

Most variants of DTW are not metrics as it is usually possible to construct examples violating the triangle inequality. However, in domains such as dictionary matching, the inequality is rarely, if ever, violated ([95]). The most common modifications to the DTW distance are the introduction of matching constraints in (3.1.19) that prohibit certain elements of the $D_{i,j}$ space to be visited during a match. The Sakoe-Chiba Band (Sakoe and Chiba [96]) and the Itakura Parallelogram (Itakura [57]) illustrated in Figure 3.2 are the two most common bandwidth constraints used with DTW.

3.1.2 Longest Common Sub-Sequence

The Longest Common Sub-Sequence (LCSS) is a specific string editing problem relevant to approximate string matching applications, including protein sequence comparison. Several different solution methods are summarized in Apostolico [4], who consid-

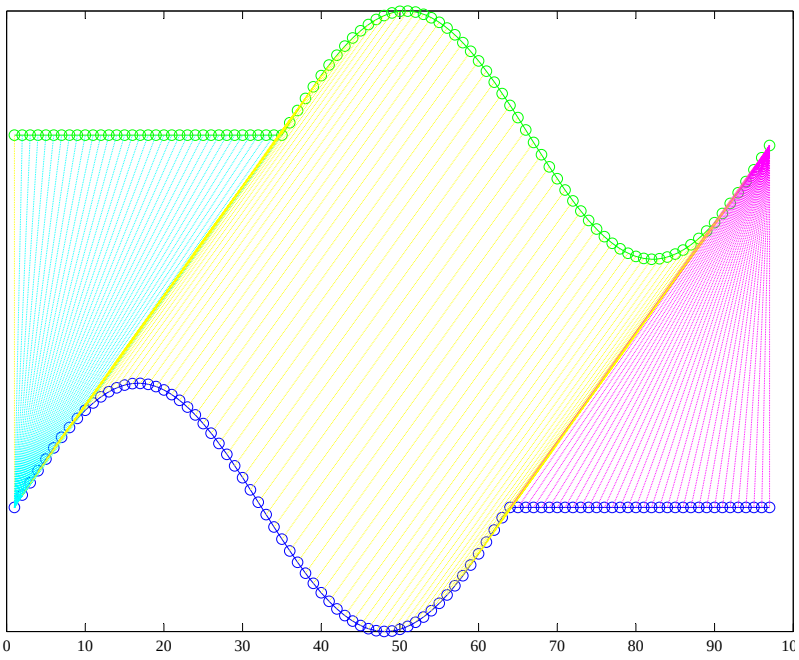
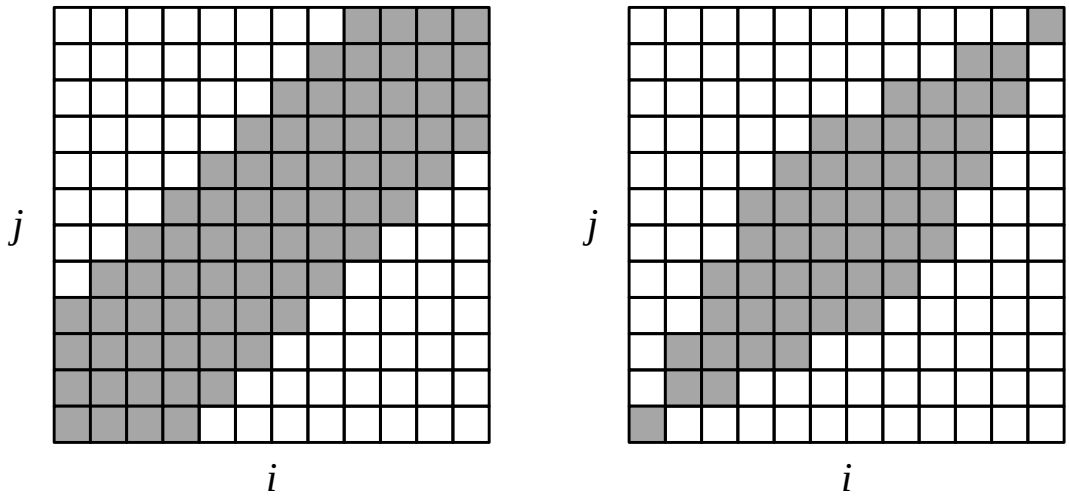


Figure 3.1. Illustration of a dynamic time warp between two sequences. Yellow lines indicate transition from $D_{i-1,j-1}$ to $D_{i,j}$ in (3.1.19), while cyan shows correspondence between $D_{i,j-1}$ and $D_{i,j}$, with magenta indicating transition from $D_{i-1,j}$ to $D_{i,j}$ on the least-cost path.

ers the two main computational approaches due to Hunt and Szymanski [55] and Hirschberg [50].

LCSS is normally defined over discrete (but possibly infinite) alphabets, and is a measure of overlap that counts elements of the sequence. The notion of equality/inequality of symbols for more general sequences (i.e., real-valued, multivariate), is not very useful as a match criterion since in practice the sequences under comparison are likely to be observed with noise.

In Buzan et al. [20], object trajectories in image coordinates are automatically recorded as two-dimensional time series. The series are split into x-coordinate series and y-coordinate series, and a similarity measure is defined that combines the one-dimensional LCSS match results for both coordinates. The match criterion considers two elements $a \in \mathbb{R}$ and $b \in \mathbb{R}$ identical whenever $|a - b| < \epsilon$ for some pre-specified positive real ϵ. Another approach, represented by Vlachos et al. [113], generalizes LCSS without splitting the coordinates of the trajectories. Again, LCSS is used to cluster two-dimensional trajectories, and the match criterion used is the infinity norm



(a) Sakoe-Chiba constraints **(b)** Itakura constraints

Figure 3.2. The two most common bandwidth constraints for DTW. The shaded regions indicate the only i, j pairs for which $D_{i,j}$ in (3.1.18) and (3.1.19) is computed and valid.

(i.e., $a \in \mathbb{R}^2$ and $b \in \mathbb{R}^2$ match if $\|a - b\|_\infty < \epsilon$). A justification for the use of the infinity norm is not given, and presumably any other norm would do.

We use the definition from [113, 114], where LCSS (with the same notation as (3.1.18)) is computed as:

$$D_{i,j} = \min \begin{cases} 0, & i = 0 \vee j = 0 \\ D_{i-1,j-1} + 1, & d_{\mathscr{T}}\left(\mathbf{x}(t_{x,i}), \mathbf{y}(t_{y,j})\right) < \epsilon \wedge \left|i - j\right| < \delta \\ \max(D_{i-1,j}, D_{i,j-1}), & \text{otherwise.} \end{cases} \tag{3.1.20}$$

The condition $\left|i - j\right| < \delta$ enforces a "stiffness" of the matching and prevents degenerate cases. This constraint is similar to the Sakoe-Chiba DTW constraints. Note that for irregularly sampled sequences, the constraint $\left|t_{x,i} - t_{y,j}\right| < \delta$ is more appropriate, and in that case the bandwidth parameter δ is in time units. The value of ϵ can be thought of as blunt scale parameter.

The arguments advanced in favor of LCSS over DTW is that the former is more resilient to outliers as the method is not forced to match every point from all curves. However, DTW with support for deletion and insertions, described in [71], allows

for similar robustness by imposing fixed costs on deletions (that skip a point of one sequence without matching it).

The LCSS computation in (3.1.20) results in a similarity measure. It can be turned into a metric using the definition:

$$d_{\text{LCSS}} = 1 - \frac{D_{n,m}}{\min(n,m)}, \tag{3.1.21}$$

which measures fractional overlap. Note, however, that this distance is unitless (i.e., unlike DTW, it does not have the same units as $d_{\mathscr{T}}$). We discuss the implications of this in Chapter 5. Finally, Figure 3.3 visually illustrates the matching process of LCSS.

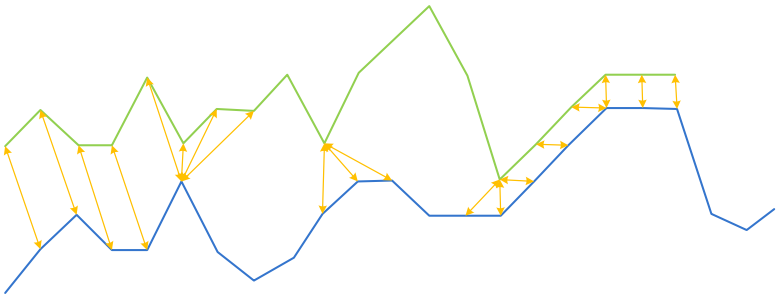


Figure 3.3. Illustration of a longest common subsequence match. Note that not all points on the green and blue curves are matched – these are effectively considered outliers due to the choice of ϵ employed in this example. Figure adapted from [47].

3.1.3 The Hausdorff distance

The Hausdorff distance between two closed subsets P and Q of a compact metric space $\mathscr{Z}$ is:

$$d_{\mathscr{H}}^{\mathscr{Z}}(P,Q) = \max\left(\sup_{p\in P}\inf_{q\in Q} d_{\mathscr{Z}}(p,q), \sup_{q\in Q}\inf_{p\in P} d_{\mathscr{Z}}(p,q)\right), \tag{3.1.22}$$

where $d_{\mathscr{Z}}$ is the metric of $\mathscr{Z}$.

It will be convenient to also consider the directed Hausdorff distance:

$$h_{\mathscr{Z}}(P,Q) = \sup_{p\in P}\inf_{q\in Q} d_{\mathscr{Z}}(p,q), \tag{3.1.23}$$

which is simply one of the terms in (3.1.22). The symmetrization of h using max is the

only combination that results in a distance measure that satisfies all metric properties. For a discussion of some modifications, see Dubuisson and Jain [37].

Note that the Hausdorff distance is not specifically defined for time series; unless the time aspect of the data is considered, the results can be sub-optimal. This is in contrast with most applications of the Hausdorff distance to (closed) shape comparison, where this poses no such problem.

The Hausdorff distance has been used for edge template matching by Huttenlocher et al. [56]. It has also been used without modification for trajectory comparison, for example in Zhang et al. [126], who also compare the unmodified Hausdorff distance against LCSS and DTW on a clustering task. A differentiable approximation to the Hausdorff distance has been used to construct shape priors by Charpiat et al. [22].

The Hausdorff distance and possible modifications are further discussed in Chapter 5, where comparisons with LCSS and DTW are performed.

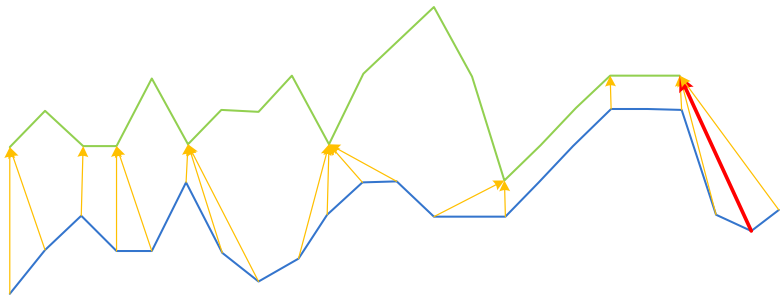


Figure 3.4. Illustration of a the directed Hausdorff distance between the blue and green trajectory. The arrows indicate the shortest distances from each point on the blue curve to a point on the green curve. The red arrow indicates the longest such distance and its length yields the directed Hausdorff distance.

3.2 Spectral Clustering

3.2.1 Overview

This section summarizes the most common multi-way spectral clustering methods. Different parts of this thesis modify or generalize specific steps in the spectral clustering "pipeline", hence the modular exposition here. In particular, Chapter 5 proposes a

slightly non-standard approach for the construction of a symmetric affinity matrix from a directed distance, while Chapter 6 does away with symmetry altogether and instead proposes modifications to the final clustering step.

Throughout this section we assume the number of clusters in the data to be known. Both Chapter 5 and Chapter 6 propose ways to automatically find the number of clusters in specific circumstances, but the problem of identification of the number of clusters in a general setting, even a spectral one, is not addressed

3.2.2 Summary of Multiway Spectral Clustering

Affinity and Degree Matrices

We summarize the relevant steps in multiway spectral clustering methods in order to establish a base for discussion. The starting point of the methods is the $n \times n$ *affinity matrix*, which stores the pairwise similarities between the objects to be clustered. We will denote the affinity matrix $\mathbf{A}$, and will also refer to its in-degree and out-degree matrices $\mathbf{D}_{\text{in}} = \text{diag}(\mathbf{1}^\top \mathbf{A})$ and $\mathbf{D}_{\text{out}} = \text{diag}(\mathbf{A}\mathbf{1})$, where $\mathbf{1}$ is a vector of all ones of appropriate size. Whenever $\mathbf{A}$ is symmetric, will omit the subscripts and denote the degree matrix as simply $\mathbf{D}$. For a comprehensive overview of spectral clustering, one can refer to von Luxburg [116] or [108].

Standard Affinity and Scale Selection

By far the most common approach to creating an affinity matrix between objects $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathscr{X}$ is to use the Gaussian kernel and define:

$$a_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right), \tag{3.2.1}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is a distance measure between the objects. When d is symmetric, $\mathbf{A}$ will also be symmetric. In Chapter 6 we will show how to extend spectral clustering to the case when d is not symmetric. The performance of spectral clustering can be strongly dependent on the choice of σ, and for certain data sets a global σ is inappropriate.

The local scaling strategy of [123], defined as:

$$a_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma_i\sigma_j}\right), \tag{3.2.2}$$

where σ_ℓ is the distance from $\mathbf{x}_\ell$ to its N-th nearest neighbor. We specify N in our experiments and in certain cases vary it. Since we consider d that can be asymmetric, we use the slight modification:

$$a_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma_i^{\rightarrow}\sigma_j^{\leftarrow}}\right), \tag{3.2.3}$$

where $\sigma_\ell^{\rightarrow}$ is the N-th shortest distance from $\mathbf{x}_\ell$ to any other point, and $\sigma_\ell^{\leftarrow}$ is the N-th shortest distance from any point to $\mathbf{x}_\ell$. Note that if d is symmetric (3.2.3) and (3.2.2) are equivalent.

Spectral Decomposition

The *spectral decomposition step* requires an eigensystem of the form $S(\mathbf{A})\mathbf{x} = \lambda\mathbf{x}$ to be solved, where the transformation S depends on the particular method. We will denote the solution as $\mathbf{V}\mathbf{L}\mathbf{V}^\mathsf{T} = S(\mathbf{A})$, and will assume that the diagonal matrix $\mathbf{L}$ has diagonal elements sorted in descending order. We further assume that $\mathbf{V}\mathbf{V}^\mathsf{T} = \mathbf{I}$. Table 3.1 summarizes the approaches.

Table 3.1. Common spectral decomposition steps and representative references for each method.

$S(\mathbf{A})$	Symmetric $\mathbf{A}$?	References
$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$	yes	[83, 123],[122]
$\mathbf{D}^{-1}\mathbf{A}$	yes	[79, 101]
$\mathbf{D}_{\text{out}}^{-1/2}(1/2(\mathbf{A}+\mathbf{A}^\mathsf{T}))\mathbf{D}_{\text{out}}^{-1/2}$	no	[78]

Embedding

Once the eigen-decomposition of $S(\mathbf{A})$ is available, in the *eigenpair selection step*, a number of corresponding eigenvalues and eigenvectors are chosen for the embedding. The most common approach is to select the first k columns of $\mathbf{V}$ (which we denote simply $\mathbf{V}_k$ and the corresponding $k \times k$ principal subblock of $\mathbf{L}$, denoted $\mathbf{L}_k$. Alternative approaches to simply picking the eigenvectors corresponding to the largest eigenvalues are proposed in Jenssen et al. [62], Xiang and Gong [120].

Finally, in the *spectral embedding step*, a set of new coordinates for the data is generated based on the chosen eigenpairs. We will denote these new coordinates

Table 3.2. Commonly used spectral embedding steps and relevant references.

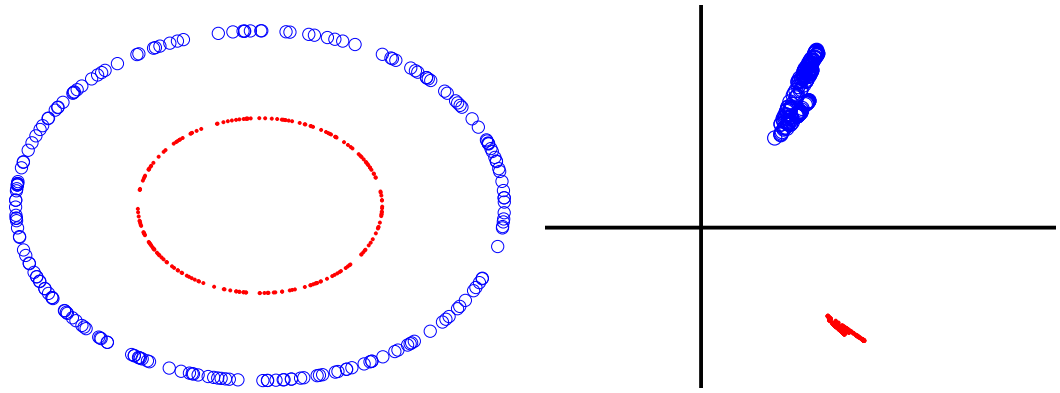
$\mathbf{Y}$	References
$\mathbf{V}_k^\mathsf{T}\,\mathrm{diag}^{-1}(\mathrm{diag}(\mathbf{V}_k\mathbf{V}_k^\mathsf{T}))$	[83]
$\mathbf{V}_k^\mathsf{T}$	[39, 123]
$\mathbf{V}_k^\mathsf{T}\mathbf{D}_{\text{out}}^{-1/2}$	[78]
$\mathbf{L}_k^{-1/2}\mathbf{V}_k^\mathsf{T}$	[98], [52]

$\mathbf{y}_1, \ldots, \mathbf{y}_n$, which will be stored as columns of the matrix $\mathbf{Y}$. We summarize the common embeddings in Table 3.2.

We illustrate the embedding $\mathbf{Y} = \mathbf{V}_k^\mathsf{T}$ for $S(\mathbf{A}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ in Figure 3.5.

Clustering of Embedded Data

The *cluster assignment step* dictates how the spectral embedding of the input is used to produce the final clustering. In [83], k-means is applied, while in [39] the assignment is based on the k-lines algorithm. The approach in [123] is to find a rotation of the spectral coordinates that maximally aligns the embedded data with the axes of the coordinate system. Each point is then assigned to the cluster corresponding to the largest coordinate after the rotation. All of the discussed methods work perfectly with the example of Figure 3.5 — the data falls very close to two lines through the origin so k-lines works. The lines are orthogonal, so the rotation method of [123] works as well.



(a) Input data **(b)** Embedding with $\mathbf{V}_k^\mathsf{T}$ for $S(\mathbf{A}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$

Figure 3.5. Illustration of the embedding from [39, 123]. The coordinate axes in the spectral space are shown as thick lines.

Finally, after a projection onto the unit circle, the data forms two very tight clusters so the k-means step of [83] also successfully splits the clusters.

3.3 Evaluating Clustering Algorithms

3.3.1 Notation

Let $[k]$ denote the set of positive integers from 1 to k. The input to a clustering algorithm is a discrete set of objects $X \equiv \{\mathbf{x}_i \in \mathscr{X}\}_{i\in[n]}$. The objective of a clustering algorithm is to partition this set into disjoint subsets, or clusters, of objects that are somehow similar. We will denote the output of a clustering algorithm as a mapping $C : [n] \to [k]$, where k is the number of desired clusters, which may be pre-specified, or an output of the algorithm. We will also use the shortcut c_i to denote $C(i)$, the cluster to which the object $\mathbf{x}_i$ is assigned.

Given a clustering C, we will also make use of the so-called cluster-indicator matrix $\mathbf{W}_C \in \mathbb{R}^{n\times k}$, with elements w_{ij}:

$$w_{ij} = \begin{cases} 1 & \text{if } C(i) = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3.1}$$

3.3.2 Comparing Clusterings

The task of comparing two different clusterings of the same data set is important in evaluating the performance of clustering algorithms. When a reference clustering is known, the evaluation task is quite similar to that of evaluating the accuracy of a classifier, with the only difference arising from the fact that any labels assigned by a clustering method are arbitrary, so clusterings that differ up to a permutation of labels are considered identical.

To compare the different clustering methods we use the Variation of Information distance by Meilă [77], who discusses several desirable properties for clustering comparison metrics. It is proved that only unbounded distances such as the Variation of Information metric satisfy all desirable properties for comparison of clustering methods. We will also make use a clustering error measure derived from the classification error

metric, which can be interpreted more intuitively and is quite common. We follow [77] in the description of both measures.

3.3.3 Confusion Matrix

To compare two clusterings $C_1 : [n] \to [k_1]$, and $C_2 : [n] \to [k_2]$, we first introduce the concept of the normalized confusion matrix $\mathbf{C} \in \mathbb{R}^{k_1 \times k_2}$ with elements c_{ij}:

$$c_{ij} = \frac{1}{n} \left| \{ m \,|\, C_1(m) = i \wedge C_2(m) = j \} \right|. \tag{3.3.2}$$

If $\mathbf{W}_{C_1}$ and $\mathbf{W}_{C_2}$ are the respective cluster indicator matrices, we can alternatively use the relationship:

$$\mathbf{C} = \frac{1}{n} \mathbf{W}_{C_1}^{\mathsf{T}} \mathbf{W}_{C_2}. \tag{3.3.3}$$

The element c_{ij} of $\mathbf{C}$ gives the joint probability that an object is assigned to cluster i by C_1 and to cluster j by C_2. The vectors $\mathbf{m}_1 = \mathbf{C}\mathbf{1}$ and $\mathbf{m}_2 = \mathbf{C}^{\mathsf{T}}\mathbf{1}$ give the unconditional probablities that C_1 and C_2 assign a point to a given cluster.

3.3.4 Variation of Information

The Variation of Information distance $d_{\mathrm{VI}}(\mathbf{C})$ measures the information lost and gained when switching from clustering C_1 to C_2, and is defined as:

$$\begin{aligned} d_{\mathrm{VI}}(\mathbf{C}) &= H(C_1|C_2) + H(C_2|C_1) \\ &= H(C_1) + H(C_2) - 2I(C_1, C_2), \end{aligned} \tag{3.3.4}$$

where $I(C_1, C_2)$ is the mutual information between clusterings C_1 and C_2, and $H(C_1)$ and $H(C_2)$ are the respective entropies of the clusterings. These quantities can be computed from $\mathbf{C}$ as follows:

$$I(C_1, C_2) = \mathbf{1}^{\mathsf{T}} \left(\mathbf{C} \cdot \left(\log \mathbf{C} - \log(\mathbf{m}_1 \mathbf{m}_2^{\mathsf{T}}) \right) \right) \mathbf{1} \tag{3.3.5}$$

$$H(C_1) = -\mathbf{m}_1^{\mathsf{T}} \log \mathbf{m}_1 \tag{3.3.6}$$

$$H(C_2) = -\mathbf{m}_2^{\mathsf{T}} \log \mathbf{m}_2 \tag{3.3.7}$$

$$\tag{3.3.8}$$

Table 3.3. Example confusion matrices for the data set from Figure 4.13d for two different clustering methods, against a reference clustering. The corresponding values of the discussed clustering comparison measures are shown.

Method	$\mathbf{C}$	$d_{\mathrm{VI}}(\mathbf{C})$	$d_{\mathrm{CE}}(\mathbf{C})$	$d_{\mathrm{CE}^*}(\mathbf{C})$
1	$\frac{1}{227}\begin{pmatrix} 10 & 0 & 0 \\ 4 & 2 & 56 \\ 0 & 145 & 10 \end{pmatrix}$	0.474	0.903	0.070
2	$\frac{1}{227}\begin{pmatrix} 0 & 10 & 0 \\ 0 & 0 & 62 \\ 155 & 0 & 0 \end{pmatrix}$	0.000	1.000	0.000

where the log operator is taken componentwise, $\cdot$ is the Hadamard product, and we use the covention that $0\log(0) = 0$ where necessary. Note that unlike [77], we use the natural logarithm rather than a base-2 logarithm.

3.3.5 Clustering Error

The Classification Error distance $d_{\mathrm{CE}}(\mathbf{C})$ is normally used to compare classification methods (where the class labels must agree). We use it only when $k_1 = k_2 = k$, and use the standard definition:

$$d_{\mathrm{CE}}(\mathbf{C}) = \operatorname{tr}\mathbf{C}. \tag{3.3.9}$$

Since we want an error measure that is invariant to permutations of the labels, we use the derived Clustering Error distance $d_{\mathrm{CE}^*}(\mathbf{c})$ defined as follows:

$$d_{\mathrm{CE}^*}(\mathbf{C}) = \min_{P\in\Pi_k} d_{\mathrm{CE}}(\mathbf{PC}), \tag{3.3.10}$$

where Π_k is the set of all $k \times k$ permutation matrices. It is usually not necessary to do an exhaustive search over all $k!$ permutation matrices to compute this distance since it is possible to directly compute some elements of the permutation matrix when an element c_{ij} is the largest element in both its corresponding column and row. More sophisticated methods based on graph-matching algorithms are also available for the computation of $d_{\mathrm{CE}^*}(\mathbf{C})$ when k is large.

Chapter 4

Vehicle Tracking

The purpose of this chapter is twofold: to present the methods used to collect data for the experiments in Chapter 5 and Chapter 6 in full, and to highlight some contributions to the state-of-the art in vehicle tracking. The good tracking results obtained by the method presented here are encouraging, but the challenges encountered serve to strengthen the case for scene-specific knowledge in support of tracking methods — the resolution of occlusion, partial visibility, and initialization problems can in principle greatly benefit from prior knowledge of the observed scene. Such knowledge can be extracted automatically from the scenes using the clustering methods of Chapter 5 and Chapter 6. The interest in learning with trajectories that led to the present work was born out of frustration with the many challenges faced by a "uninformed" tracking method such as presented here.

The most important contribution described in this chapter is the virtual vanishing point "caliper" used to provide partial measurements of vehicles under occlusion (see Figure 4.9). The fact that vehicle dimensions can be measured as well is a distinguishing feature of the method, as is the use of the constrained Kalman filter.

4.1 Introduction

Our goal is to automatically track vehicles at common traffic scenes, such as highways and intersections. Providing geometric models for all possible types of vehicles encountered on roads is a daunting task, so we settle for a simpler, box model of vehicles. Since vehicle outlines can be quite useful, we use connected regions of

foreground pixels as the basic feature to track. Connected regions (blobs) extracted from a foreground mask provide us with good object outlines, allow for automatic target identification, and are well-suited for real-time systems.

In order to classify foreground pixels, we need a background model of the observed scene. Due to the gradual changes in scene appearance over extended periods of time, we cannot use a static background model. Instead, we use an adaptive background model based on the mixtures of Gaussians segmentation method in Stauffer and Grimson [104]. The resulting background/foreground classifier adapts well to gradual changes in the monitored outdoor environment and allows for the detection of targets even if they are not moving — a common occurrence at traffic intersections. Occasionally, the background extraction can fail, either because of sudden illumination changes in the scene caused by passing clouds or a camera's gain control circuitry, or due to minor camera shaking resulting from road vibrations or wind load. We present efficient methods for compensating for sudden illumination changes and camera shaking. We also describe a fast implementation of the method presented in Bevilacqua [16] for cleaning up the foreground masks.

Vehicles that move throughout the scene will sometimes occlude other moving objects, or be themselves occluded by static objects such as road-sign poles, traffic lights, etc. Such occlusions cause blobs to merge, split into smaller regions, or to disappear completely. These interactions between connected regions can either cause a single target to be visible as more than one blob in the foreground, or cause several targets to be represented as a single blob. The problem can be alleviated by making use of multiple cameras to observe the same intersection; proper camera placement can ensure that vehicles are rarely occluded in the views of all cameras. However, even when multiple cameras are used, it is advantageous to have some means of handling occlusions in a single view. For that reason, we represent targets as sets of regions and introduce a second-level tracker that is capable of handling blob merges and splits. The second-level tracker also makes use of camera calibration data in order to estimate the position and velocity of vehicles in world-coordinates. The calibration method we use is presented in Masoud and Papanikolopoulos [76] and allows us to accurately map points from the image-planes of all cameras to positions on a single world-coordinate ground-plane.

It is common practice to use the centroids of connected regions to represent a target's position. Such an approach is sub-optimal, because the position of a centroid

relative to the ground-plane depends on the size and orientation of vehicles and also on the particular camera placement. The last fact complicates multi-camera tracking since the centroids tracked in different camera views do not correspond to the same real-world point. We introduce a method that can identify the centers of vehicular bases on the ground plane given the outlines of the vehicles and the camera calibration data. The base centers identified by our method correspond to the same real-world point, which allows for sequential incorporation of the measurements in a target's state vector. The method also produces estimates of the width, length, and height of vehicles.

4.2 Low-Level Vision

In this section we elaborate on the methods used for background model maintenance, illumination filtering, camera shaking compensation, and noise removal in the foreground mask. Figure 4.1 shows the components of the low-level vision system and the data interactions between them.

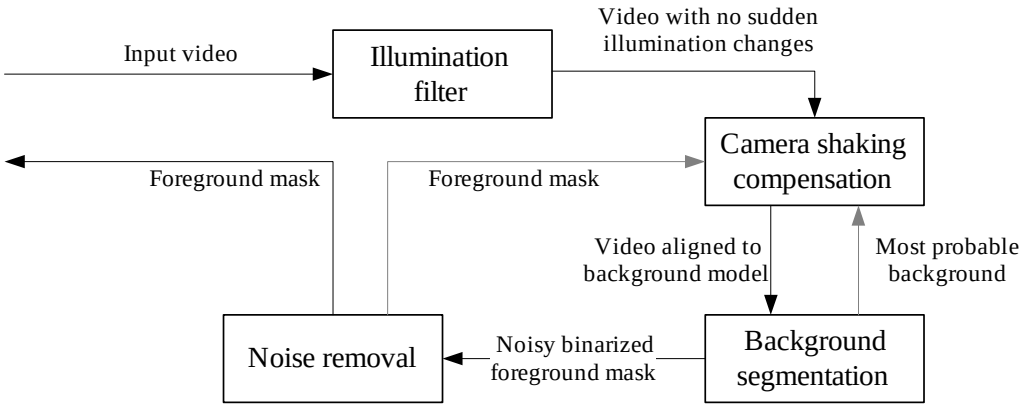


Figure 4.1. Low-level vision system components and data flow for a single frame. Lighter lines indicate the use of data from the processing of the previous frame.

4.2.1 Illumination Filter and Background Model Maintenance

The illumination filter runs before any other processing takes place. Its purpose is to prevent sudden brightness or contrast changes in its output, while preserving the color

information in its input. The filter scales and adds an offset to all pixels in the image. The multiplicative and additive factors used are the same for each pixel component (red, green, and blue). The mean m_{in} and root-mean-square (rms) r_{in} of all pixel components (regardless of channel) are used as measures of brightness and contrast, respectively. The scale factors are chosen so that the mean m_{out} and rms r_{out} of the output image match some specified target values m_{tgt} and r_{tgt}. Finally, we obtain the target values for the next video frame according to the equations:

$$m_{\text{tgt}} \leftarrow (1-\omega)m_{\text{tgt}} + \omega m_{\text{in}} \tag{4.2.1}$$

$$r_{\text{tgt}} \leftarrow (1-\omega)r_{\text{tgt}} + \omega r_{\text{in}}. \tag{4.2.2}$$

The factor ω is chosen to be lower than the learning rate used in the background model update equations in order to ensure that the output of the filter never changes faster than the background model can accommodate the changes.

The background model represents the probability of observing a 3-component background pixel as a mixture of Gaussians:

$$P(\mathbf{c}) = \sum_{i=1}^{n} \pi_i P(\mathbf{c}|M_i) = \sum_{i=1}^{n} \pi_i \frac{e^{-\frac{1}{2}(\mathbf{c}-\mu_i)^T \Sigma_i^{-1}(\mathbf{c}-\mu_i)}}{\sqrt{(2\pi)^3|\Sigma|}}. \tag{4.2.3}$$

The mixture weights π_i, the mixture means μ_i, and the mixture covariance Σ_i are stored for each pixel and updated at every frame. An input pixel matches a mixture component if its Mahalanobis distance to the component's mean is less than a threshold. The mixture components are arranged in descending order of their $\pi_i/\sqrt{|\Sigma_i|}$ ratios as suggested by [104]. At every new frame, the index j of the first component (if any) to match the input pixel is recorded. The pixel is labeled as background if $\sum_{i=1}^{j-1} \pi_i \leq T$ for some threshold T. Unlike [104], we do not make simplifying assumptions about the structure of Σ_i.

If an input pixel $\mathbf{c}$ does not match any mixture component, the least probable one is replaced by a component with mean $\mathbf{c}$ and suitable initial weight and covariance. Mixture components are updated using the following transformations applied

sequentially:

$$\pi_i \leftarrow (1-\alpha)\pi_i + \delta_{ij}\alpha \tag{4.2.4}$$

$$\mu_i \leftarrow (1-\delta_{ij}\beta)\mu_i + \delta_{ij}\beta\mathbf{c} \tag{4.2.5}$$

$$\Sigma_i \leftarrow (1-\delta_{ij}\beta)\Sigma_i + \delta_{ij}\beta(\mathbf{c}-\mu_i)(\mathbf{c}-\mu_i)^T. \tag{4.2.6}$$

The factor δ_{ij} denotes the Kronecker delta. An account of our choices for the learning rates α and β, as well as all other parameters for the illumination filter and background segmentation can be found in Atev et al. [6]. Theoretical justification of the method is available in Power and Schoonees [92].

4.2.2 Camera Shaking Compensation

Camera shaking is problematic for the vision system since the static camera assumption is violated. We cannot afford to match input pixels against a neighborhood of background pixel models since each match involves the calculation of Mahalanobis distances to a number of mixture components. Even a modestly-sized neighborhood increases the computational burden considerably.

The approach we take allows us to compensate for whole-pixel translations of the input relative to the background. Due to the need for good coverage, the cameras observing the traffic are at a significant distance from the road. At such distances, the effects of minute camera pose changes can be approximated by a single translation.

To find the translation that describes a camera pose change most accurately, we match the input video frame against the most probable background image consisting of all the μ_1 mixture component means at each pixel location. The matches are performed at a fixed number of possible horizontal and vertical offsets and the one with the least sum of absolute differences yields the desired translation. The sum of absolute differences measure is used because it is not affected significantly by a small number of outliers. If the background image at a pixel location $\mathbf{p}$ is denoted as $B(\mathbf{p})$, and the input frame is denoted by $I(\mathbf{p})$, we find the optimal translation $\mathbf{q}$ as follows:

$$\mathbf{q} = \underset{\|\mathbf{q}\|_\infty \le w}{\operatorname{argmin}} \frac{\sum_{\mathbf{p}} V(\mathbf{p})|B(\mathbf{p}) - I(\mathbf{p}+\mathbf{q})|}{\sum_{\mathbf{p}} V(\mathbf{p})}. \tag{4.2.7}$$

The parameter w limits the search to a small square window. The mask $V(\mathbf{p})$ is

zero-valued if the pixel at location $\mathbf{p}$ was a foreground pixel in the previous frame and 1 otherwise. The mask $V(\mathbf{p})$ prevents the motion of objects in the scene from being mistaken for a camera displacement. In practice, when only a small portion of the scene is classified as foreground, the mask $V(\mathbf{p})$ can be ignored since the sum of absolute differences distance measure is robust against the small number of outliers present.

Equation (4.2.7) does not lead to a very efficient implementation since it requires a total of $(2w+1)^2$ full-image matches to be performed. In order to improve the performance of the camera shaking compensation, we use a hierarchical approach. We build an ℓ-level pyramid of the input image $I(\mathbf{p})$ and the most probable background image $B(\mathbf{p})$, denoted as $I_1(\mathbf{p})\ldots I_\ell(\mathbf{p})$ and $B_1(\mathbf{p})\ldots B_\ell(\mathbf{p})$ in the subsequent discussion. The first level is at full resolution (i.e. $B_1(\mathbf{p}) = B(\mathbf{p})$ and $I_1(\mathbf{p}) = I(\mathbf{p})$), while each subsequent level is obtained by down-sampling the previous one by averaging 2×2 pixel regions. The optimal translation $\mathbf{q}_j$ at pyramid level j is found using the following recurrence:

$$\mathbf{q}_j = \underset{\|\mathbf{q}_j - 2\mathbf{q}_{j+1}\|_\infty \le 1}{\operatorname{argmin}} \frac{\sum_{\mathbf{p}} V_j(\mathbf{p}) |B_j(\mathbf{p}) - I_j(\mathbf{p}+\mathbf{q}_j)|}{\sum_{\mathbf{p}} V_j(\mathbf{p})}. \tag{4.2.8}$$

The search starts at the highest pyramid level ℓ (using the definition $\mathbf{q}_{\ell+1} = \mathbf{0}$), and proceeds downwards until we find the full-resolution optimal displacement $\mathbf{q}_1$. At each level, a total of 9 image-to-background matches are performed. The effective range of (4.2.8) with ℓ levels is the same as that of (4.2.7) using a window size of $2^\ell - 1$. If the images under consideration have N pixels, the number of absolute differences computed using the hierarchical method is at most $9\sum_{i=1}^{\ell} \frac{N}{4^{i1}} \le 12N$. In contrast, the non-hierarchical method requires a number of absolute differences computed proportional to $(2w+1)^2 N$, which is much larger even for modest values of w. In our implementation we use 5 pyramid levels, which enable us to find translations with horizontal and vertical offsets in the range ± 31 pixels.

Once the appropriate image-to-background translation $\mathbf{q}_1$ is found, we simply shift the input image by $-\mathbf{q}_1$ to align it with the background model.

4.2.3 Noise Removal

The foreground mask obtained using the described background extraction method usually contains a small amount of noise, which affects the connectedness and outlines

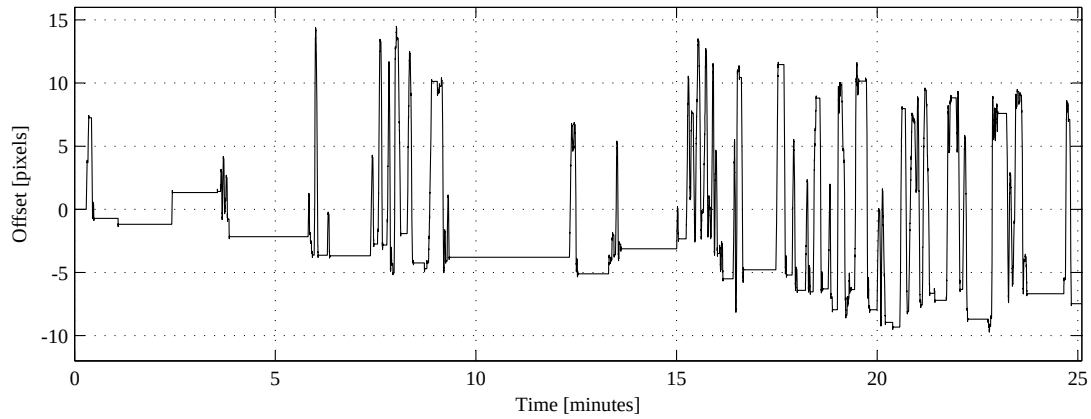


Figure 4.2. Video stabilization results on video from Scene 1. The lateral translation is shown for the duration of the tracking sequence.

of the identified blobs. It is common to post-process such binarized background masks using a sequence of simple morphological operations, such as erosions and dilations. However, determining the appropriate number and size of such operations is hard, and a wrong choice can severely affect the quality of the resulting mask (Figure 4.3). The blob segmentation method in [16] addresses these issues.

The method evaluates the degree to which pixels in the binary mask belong to a connected region. The basic structuring element used for determining the fitness of pixels is shown in Figure 4.4(a). Arrows in the diagram represent logical *AND* operations between the indicated pixels, and the target of the arrows counts the number of operations that yield a true value. The basic elements are combined into a compound structuring element as illustrated in Figure 4.4(b) and the sum of their fitness values is the fitness value for the central pixel. The sum of fitness values over a small neighborhood of each pixel is thresholded to obtain the final binary mask.

A direct implementation of the method would require 9 memory accesses, 12 logical *AND* operations and 11 additions per pixel. To reduce the run-time overhead of the method, we based our implementation on a finite-state transducer with 64 states, an 8 character input alphabet and 12 character output alphabet. Figure 4.5(a) shows how the state and input character depend on the values of the pixels in a compound region. At each pixel, we need 3 memory accesses, 2 shifts, and 2 additions to compute the input character. A table lookup given the current state and the input character yields the desired output and next state. We combine the output and next state information in a single integer value (lower 4 bits for the output, next 6 bits for

a) b) c) d) e)

Figure 4.3. a) Foreground mask before noise removal; b) Noise removal using 1 erosion followed by 1 dilation; c) Using 2 erosions and 2 dilations; and d) Using 1 erosion and 3 dilations; e) Structural denoising using a 3×3 neighborhood and fitness threshold 30.

the next state). One iteration through the transducer therefore requires 1 memory lookup, 1 bitwise *AND* operation to extract the output character, and 1 shift to obtain the next state. Using this approach, we achieved a 44% reduction[1] in the per-frame processing time with respect to an optimized direct implementation of the method.

4.3 Image-Level Tracking

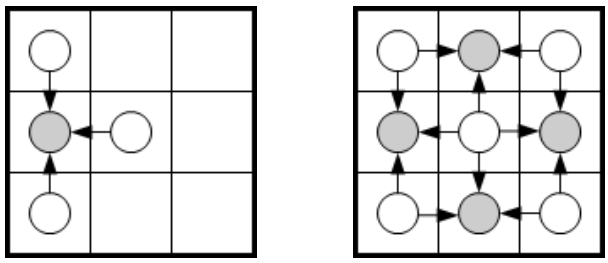
The output of the low-level vision system is a foreground mask on which we perform connected region extraction. The target tracking proceeds in two stages — image-space region tracking and ground-plane tracking of sets of connected regions.

4.3.1 Image-Space Tracker

The image-space region tracker computes the fractional overlap between regions in the current frame, denoted $Q = \{(\mathbf{q}_j, \mathbf{v}_j)\}_{j=1}^{n_q}$, and the predicted position of blobs from the previous frame, denoted as $P = \{(\mathbf{p}_i, \mathbf{u}_i)\}_{i=1}^{n_p}$. The predicted position of a blob $P_i \in P$ is obtained by adding its velocity $\mathbf{u}_i$ to its position $\mathbf{p}_i$.

If the overlap is above a threshold (50%), the respective blobs are associated and the velocity $\mathbf{v}_j$ of the current frame blob $Q_j \in Q$ located at $\mathbf{q}_j$ is estimated for use in the

[1] 2.3 ms vs. 4.2 ms on a 1.2GHz Pentium III for 320×240 pixel masks.



(a) A single structural element

(b) The compound structural element composed of 4 basic elements

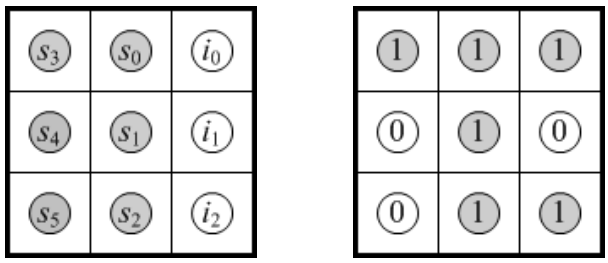
Figure 4.4. Noise removal structuring elements, adapted from [16].

next frame. Figure 4.6 shows the possible overlap configurations that can occur from one frame to the next one; a particular configuration is indicative of an occlusion event that must be handled as a separate case when estimating the image-plane velocity of connected regions.

Since we only need to predict the future positions of blobs in the image for one frame into the future, we make use of a simple method for estimating the image-space velocities: If a blob P_i is only related to one blob Q_j (Figure 4.6(a)), $\mathbf{v}_j$ is estimated as $0.5\mathbf{u}_i + 0.5(\mathbf{q}_j - \mathbf{p}_i)$. In the case of a split of the blob P_i into the blobs $Q_{j_1}, Q_{j_2}, \ldots, Q_{j_m}$ (Figure 4.6(b)), we set each of the velocities $\mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \ldots, \mathbf{v}_{j_m}$ to $\mathbf{u}_i$. Newly appearing blobs are assumed to have zero velocity, while the velocity of a blob which has more than one predecessor (Figures 4.6(c) and 4.6(d)) is computed by taking the average velocity of all predecessors, weighted by their areas.

4.4 Ground-Plane Tracking

In this section we present an automated vehicle tracking method that supplies trajectory, orientation, and dimensions data about the identified vehicles in real-world units, in contrast with most of the cited tracking methods which provide such state information in image coordinates only. Recently, [86] have used a similar 3D approach to solve the related problem of vehicle counting during multiple occlusions, and their approach may be applicable to the initial vehicle detection and target initialization stages of our tracker.



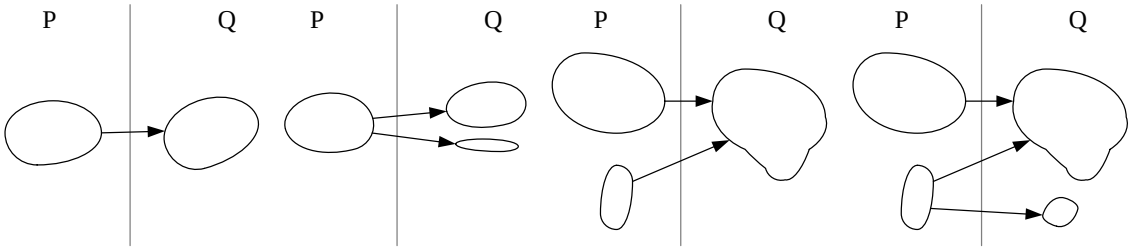
(a) Pixels contributing to the state and input character

(b) A sample input with central pixel fitness 5

Figure 4.5. a) Illustration of the pixels contributing to the six-bit state $\overline{s_5 s_4 s_3 s_2 s_1 s_0}$ (obtained from the first two columns) and the three-bit input character $\overline{i_2 i_1 i_0}$ (third column); b) Example computation: before processing the central pixel, the transducer is in state 15. The input character is 5, and the transition from state 15 on input 5 produces the desired output value 5 and the next state 30.

The benefits of tracking vehicles in a world frame are several: we do not need to make assumptions about camera orientation that are often implicit in traffic monitoring applications (e.g. camera has no roll, camera looks along the direction of motion, camera looks sideways at vehicles, etc.); we can impose meaningful constraints on the estimated state of vehicles in real-world units (e.g. street-legal vehicles are not wider than 14ft, vehicles are at least 3ft tall, etc.), which improves the state estimates and reduces their uncertainty; we can design the noise covariance matrices of the tracking stochastic filter from first principles as each element of the state vector has a physical meaning. Finally, the fusion of measurements from multiple views into a target's state comes naturally, as all estimated state variables are view-independent.

Our vehicle model accounts for the 3D structure of vehicles, but is simpler than the models used by [54, 68]. As such, it generalizes better to vans, trucks, and other vehicles with non-sedan shapes. One motivation for the specific vehicle and measurement models used is the ability to make partial measurements of a vehicle's state during periods of static or dynamic occlusion. In this context, a partial measurement is one which does not necessarily constrain all of the estimated state variables. Such measurements provide useful information to the tracker and drastically improve our ability to track vehicles during transient occlusions. In addition, when our method is applied to multiple views, a set of partial measurements from different views may be



(a) one-to-one **(b)** simple split **(c)** simple merge **(d)** split-merge conflict

Figure 4.6. Possible blob interactions between the current frame and the previous frame.

fused so as to make the state fully observable.

We will briefly discuss the camera calibration data available to the ground-plane tracker in the following section and then describe the two tracking stages in more detail.

4.4.1 Scene Calibration

Camera calibration is performed for a given traffic intersection using the approach outlined by [76]. Geometric primitives on the scene such as sets of parallel lines, vertical lines, and perpendicular lines are identified on a still-image view from each camera. Correspondences between points in the views of different cameras and some real-world distances between points in the same camera view are also specified. Figure 4.7 shows a typical intersection view from two angles. The calibration procedure in [76] jointly optimizes the camera calibration parameters for the two views to ensure consistency of the coordinate systems.

The camera calibration routine then generates the following data for each camera: a 3×3 intrinsic matrix $\mathbf{A}$, a world–to–camera coordinate transformation represented by the 4×4 matrix $\mathbf{T}_{\text{wc}} = [t_{ij}]$, an image-plane–to–ground-plane perspective mapping represented by the 3×3 matrix $\mathbf{G} = [g_{ij}]$, and the camera's location $\mathbf{c} = (c_1, c_2, c_3)^{\mathsf{T}}$ in real-world coordinates. The calibration data allows us to perform a number of computations that are relevant to the ground-plane tracker and the size estimation component of the collision prediction module.

A point (u, v) in the image-plane of a camera can be mapped to a point (x, y) on

(a) First camera view **(b)** Second camera view

Figure 4.7. Typical traffic camera views. The highlighted region indicates the region of overlap between the two views.

the ground plane using the following relation:

$$z \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{G} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \tag{4.4.1}$$

The scalar z in (4.4.1) represents the perspective division. The calibration data also allows us to find the image-plane coordinates of a vanishing point $\mathbf{w} \in R^2$ in the direction of any world-coordinate unit-vector $\mathbf{d} \in R^3$ as follows:

$$\mathbf{F} = \mathbf{A}\mathbf{I}_{3\times 4}\mathbf{T}_{\text{wc}} \tag{4.4.2}$$

$$\mathbf{r} = \mathbf{F}\mathbf{d} \tag{4.4.3}$$

$$\mathbf{w} = \begin{pmatrix} r_1/r_3 \\ r_2/r_3 \end{pmatrix}. \tag{4.4.4}$$

To find the length of a line segment perpendicular to the ground-plane, we need to determine the real-world height of its floating end-point $\mathbf{x}$ given the image coordinates of its projection $\mathbf{y} = (y_1, y_2)^{\mathsf{T}}$ on the ground-plane (the other end-point). The height h

is given by:

$$z = \frac{(\mathbf{x}-\mathbf{y})^{\mathsf{T}}((f_{13}, f_{23})^{\mathsf{T}} - f_{33}\mathbf{y})}{f_{13}^2 + f_{23}^2 + f_{33}^2\mathbf{x}^{\mathsf{T}}\mathbf{y} - f_{33}(f_{11}, f_{23})(\mathbf{x}+\mathbf{y})}$$
$$h = \frac{z}{g_{31}y_1 + g_{32}y_2 + g_{33}}. \tag{4.4.5}$$

The scalars f_{ij} are the components of the matrix $\mathbf{F}$ in (4.4.2).

To compute the ground-plane projection $\mathbf{y}$ of a point whose height h and image coordinates $\mathbf{x}$ are known, we first find the ground-plane point $\mathbf{z}$ which corresponds to $\mathbf{x}$ in the image-plane (using (4.4.1) with $\mathbf{x}$ and $\mathbf{z}$). The projection of $\mathbf{x}$ on the ground-plane is then computed as:

$$\mathbf{y} = \mathbf{c} + \left(1 - \frac{h}{c_3}\right)(\mathbf{z} - \mathbf{c}). \tag{4.4.6}$$

Let $\mathbf{P}$ denote the 3×4 world-to-image projection matrix obtained during camera calibration [76]. A point $\mathbf{p} = (x_w, y_w, z_w)^{\mathsf{T}}$ in the world is related to an image-coordinate point $\mathbf{q} = (u, v)^{\mathsf{T}}$ by the equation:

$$(u \cdot w, v \cdot w, w)^{\mathsf{T}} = \mathbf{P}(x_w, y_w, z_w, 1)^{\mathsf{T}}. \tag{4.4.7}$$

4.4.2 Vehicle Model

The complexity of our vehicle representation is lower than that of a model-based tracker, as we model vehicles as 3D boxes and do not make use of model features such as edges and planar patches. We estimate the real-world dimensions of the boxes that represent vehicles, unlike trackers that merely estimate position. Figure 4.8 introduces the state variables estimated for each tracked vehicle — the position in world coordinates $(x(t), y(t))$, the speed $s(t)$ and orientation $\phi(t)$, the half-width $w(t)$ and half-length $\ell(t)$, and the height $h(t)$. We will omit the time argument in the sequel, and all derivatives are with respect to time unless specified otherwise.

State Equations

We estimate the state of the vehicles using a constrained Extended Kalman filter. We present the non-linear motion model and provide details on the linearization used in

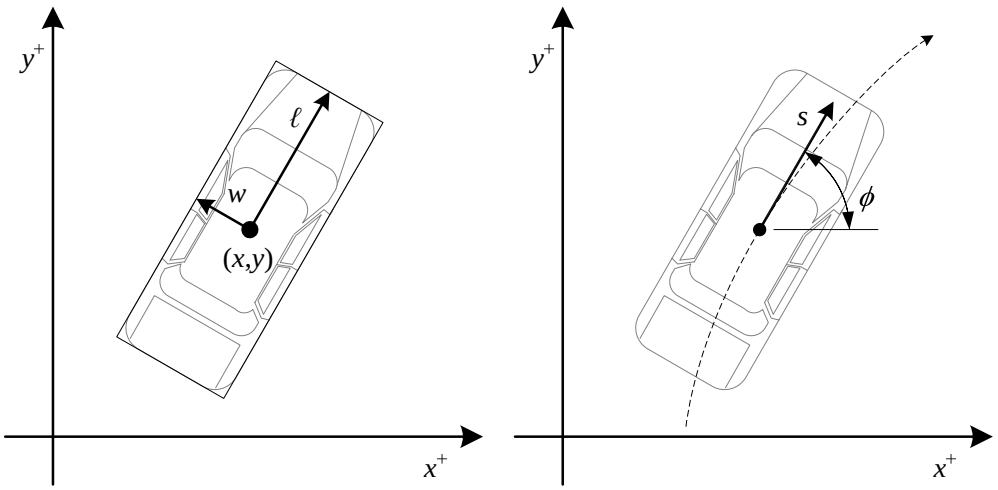


Figure 4.8. State variables of interest, excluding height $h(t)$ and curvature $\kappa(t)$.

the filter. Constraints are discussed in Section 4.4.4.

The vehicle dynamics model we consider is given by the system:

$$\begin{aligned}
\dot{x} &= s\cos\phi \\
\dot{y} &= s\sin\phi \\
\dot{s} &= 0 + \omega_s \\
\dot{\phi} &= \kappa s \\
\dot{\kappa} &= 0 + \omega_\kappa \\
\dot{w} &= 0 + \omega_w \\
\dot{\ell} &= 0 + \omega_\ell \\
\dot{h} &= 0 + \omega_h,
\end{aligned} \tag{4.4.8}$$

where the white-noise ω variables drive the evolution of the system. The coupling of ϕ and s through the curvature term κ allows us to better handle stop-and-go traffic and allows us to model the turning motion of vehicles at intersections. Highway-only tracking usually does not require the curvature term, which is then assumed to be zero.

System Linearization

For convenience, let $\mathbf{x}$ denote the state of the system and $\boldsymbol{\omega}$ denote the noise:

$$\mathbf{x} = (x, y, s, \phi, \kappa, w, \ell, h)^{\mathsf{T}} \tag{4.4.9}$$

$$\boldsymbol{\omega} = (\omega_s, \omega_\kappa, \omega_w, \omega_\ell, \omega_h)^{\mathsf{T}}, \tag{4.4.10}$$

so the vehicle motion equations become $\mathbf{x} = \mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$, yielding the discrete-time system $\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n, \boldsymbol{\omega}_n)$, with $\mathbf{x}_n = \mathbf{x}(n\Delta t)$ and $\boldsymbol{\omega}_n = \int_{n\Delta t}^{(n+1)\Delta t} \boldsymbol{\omega}(\tau)\,\mathrm{d}\,\tau$. The time step Δt is determined by the frame rate of the incoming video.

For the purpose of the filter, we need to be able to evaluate $\mathbf{F}(\mathbf{x}_n, \mathbf{0})$, the 8×8 Jacobian $\mathbf{A}_n = [\partial \mathbf{F}_i / \partial \mathbf{x}_{n,j}]_{1 \le i,j \le 8}$, and the 8×5 Jacobian $\mathbf{W}_n = [\partial \mathbf{F}_i / \partial \boldsymbol{\omega}_{n,j}]_{1 \le i \le 8,\ 1 \le j \le 5}$. We denote the i-th component of $\mathbf{F}(\mathbf{x}_n, \mathbf{0})$ by $\mathbf{F}_i$, $\mathbf{x}_{n,j}$ is the j-th component of $\mathbf{x}_n$, and similarly $\boldsymbol{\omega}_{n,j}$ is the j-th component of the noise $\boldsymbol{\omega}_n$. The integration $\int_{n\Delta t}^{(n+1)\Delta t} \mathbf{f}(\mathbf{x}(\tau), \mathbf{0})\,\mathrm{d}\,\tau$ yields:

$$\mathbf{F}_1(\mathbf{x}_n, \mathbf{0}) = x_n + (\sin(\phi_n + \kappa_n s_n \Delta t) - \sin\phi_n)\kappa_n^{-1} \tag{4.4.11}$$

$$\mathbf{F}_2(\mathbf{x}_n, \mathbf{0}) = y_n + (\cos\phi_n - \cos(\phi_n + \kappa_n s_n \Delta t))\kappa_n^{-1} \tag{4.4.12}$$

$$\mathbf{F}_4(\mathbf{x}_n, \mathbf{0}) = \phi_n + \kappa_n s_n \Delta t \tag{4.4.13}$$

$$\mathbf{F}_i(\mathbf{x}_n, \mathbf{0}) = \mathbf{x}_i \quad \text{for} \quad i = 3, 5, 6, 7, 8. \tag{4.4.14}$$

If $\kappa_n = 0$, we need to calculate $\lim_{\kappa_n \to 0} \mathbf{F}(\mathbf{x}_n, \mathbf{0})$, which is well-defined. We omit the expression for this special case here, but it should be used for small values of κ to avoid numerical problems. The same holds for the expressions defining $\mathbf{A}_n$ and $\mathbf{W}_n$.

The non-trivial (neither 0, nor 1) elements of the Jacobian $\mathbf{A}_n$ are specified below

(with the n-indices omitted for brevity, and $\theta = \phi + \kappa s \Delta t$):

$$\partial \mathbf{F}_1 / \partial \mathbf{x}_{n,3} = (\cos\theta - \cos\phi)\kappa^{-1} \tag{4.4.15}$$
$$\partial \mathbf{F}_1 / \partial \mathbf{x}_{n,4} = \Delta t \cos\theta \tag{4.4.16}$$
$$\partial \mathbf{F}_1 / \partial \mathbf{x}_{n,5} = (\kappa s \Delta t \cos\theta + \sin\phi - \sin\theta)\kappa^{-2} \tag{4.4.17}$$
$$\partial \mathbf{F}_2 / \partial \mathbf{x}_{n,3} = (\sin\phi - \sin\theta)\kappa^{-1} \tag{4.4.18}$$
$$\partial \mathbf{F}_2 / \partial \mathbf{x}_{n,4} = \Delta t \sin\theta \tag{4.4.19}$$
$$\partial \mathbf{F}_2 / \partial \mathbf{x}_{n,5} = (\kappa s \Delta t \sin\theta + \cos\theta - \cos\phi)\kappa^{-2} \tag{4.4.20}$$
$$\partial \mathbf{F}_3 / \partial \mathbf{x}_{n,4} = \kappa \Delta t \tag{4.4.21}$$
$$\partial \mathbf{F}_3 / \partial \mathbf{x}_{n,5} = s \Delta t, \tag{4.4.22}$$

and similarly, the non-trivial components of $\mathbf{W}_n$ are:

$$\partial \mathbf{F}_1 / \partial \boldsymbol{\omega}_{n,1} = (\cos\theta - \cos\phi)\kappa^{-1} \tag{4.4.23}$$
$$\partial \mathbf{F}_1 / \partial \boldsymbol{\omega}_{n,2} = \Delta t \cos\theta \tag{4.4.24}$$
$$\partial \mathbf{F}_2 / \partial \boldsymbol{\omega}_{n,1} = (\kappa s \Delta t \cos\theta + \sin\phi - \sin\theta)\kappa^{-2} \tag{4.4.25}$$
$$\partial \mathbf{F}_2 / \partial \boldsymbol{\omega}_{n,2} = (\sin\phi - \sin\theta)\kappa^{-1} \tag{4.4.26}$$
$$\partial \mathbf{F}_3 / \partial \boldsymbol{\omega}_{n,1} = \Delta t \sin\theta \tag{4.4.27}$$
$$\partial \mathbf{F}_3 / \partial \boldsymbol{\omega}_{n,2} = (\kappa s \Delta t \sin\theta + \cos\theta - \cos\phi)\kappa^{-2}. \tag{4.4.28}$$

Noise Covariance

The process noise of the system representing tracked vehicles is assumed to be Gaussian, as required by the Extended Kalman filter. If $\mathbf{Q}$ denotes the covariance of the system noise $\boldsymbol{\omega}$, the covariance of the system is approximated as $\mathbf{W}\mathbf{Q}\mathbf{W}^\mathsf{T}$. It is important to point out that $\mathbf{Q}$ can be designed quite easily, as each of its elements has a meaningful physical interpretation. Further, once tuned, these parameters do not need to change for use in different locations or camera views, which is a significant advantage over image-space trackers, and which is an argument in favor of calibration. The term "typical" in the discussion below refers to values that are one standard deviation away from their respective means.

The covariances of the noise elements ω_w, ω_ℓ, and ω_h are technically zero, as vehicles are rigid bodies. Our tracker can work with these quantities set to zero, but

in practice, to avoid some numerical instabilities and avoid overcommitment to early measurements, we assign a very small value to those three variances (equivalent to a typical drift in dimensions of $1^{mm}/_{s}$).

The velocity noise ω_s must account for the absence of an acceleration term in our vehicle model; as such, its covariance should be just big enough to allow the model to adapt to the typical acceleration/deceleration of vehicles. Typical default values in vehicle modeling allow for maximum accelerations of about $4.5^{m}/_{s^2}$, which translates into a covariance equivalent to typical accelerations of $1.5^{m}/_{s^2}$.

The only system parameter that needed empirical adjustment was the curvature noise term. When its covariance was assumed to be too large, curvature constraints in our filter were violated too often for slow-moving and stopped vehicles (note that the curvature constraint is related to the minimum turning radius of vehicles and the aggressiveness of the steering, which can in principle be determined using knowledge about typical vehicles on the road). When the covariance of the curvature noise was assumed to be too low, turning vehicles were not tracked adequately as the filter's estimates lagged measurements too much. After experimentation, we assumed a value of the covariance equivalent to a typical change in curvature of about $0.08^{m^{-1}}/_{s}$.

4.4.3 Measurement Model

The idea behind our measurement model is that we can measure some of the corners of the boxes that represent vehicles (typically 6 of the 8 corners are visible), and that even during static and dynamic occlusions, at least a few of those corners are visible. While the dimensions of the vehicle box cannot be estimated unless most of the corners are visible, once the dimension estimates converge they change very slowly, so it becomes possible to track the vehicles even if only one corner is visible (two corners would be required for a turning vehicle). We measure the corners by finding tangents from vanishing points related to a vehicle's orientation and the outline of the image regions associated with a given vehicle, as illustrated in Figure 4.9. Each of the 8 different vehicle box corners has a different measurement function associated with it (that maps the vehicle state to the image-coordinates of the respective corner), so once we have determined which of the corners are visible, we can compose an overall measurement function for a target by simple augmentation. In the subsequent section we will derive a single corner's measurement function.

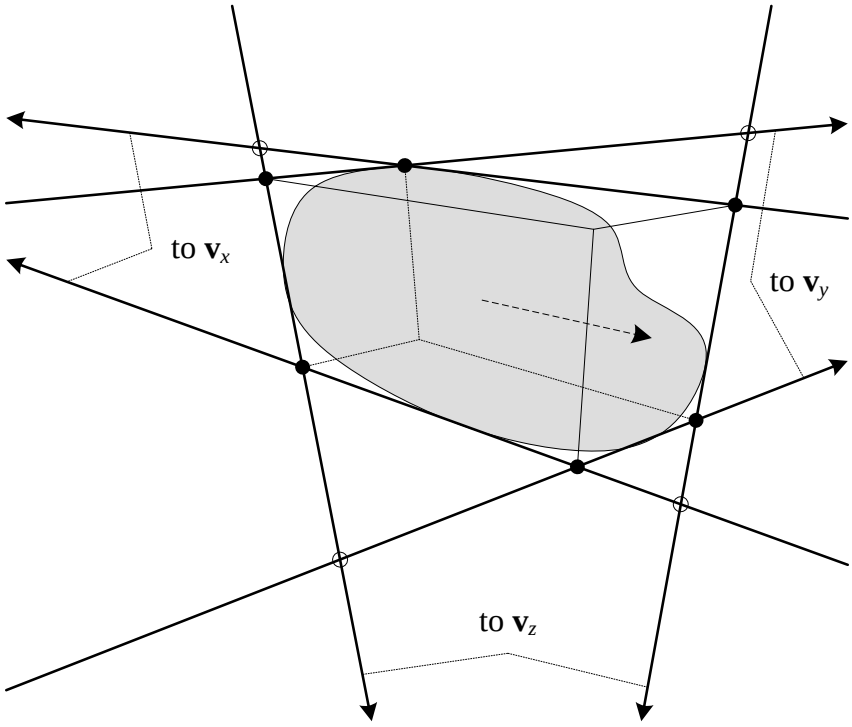


Figure 4.9. A connected region with tangents from the vanishing points $\mathbf{v}_x$ (in the direction of motion), $\mathbf{v}_y$ (perpendicular to the direction of motion), and $\mathbf{v}_z$ (vertical). In general, there are 6 tangents, and 6 of the pairwise tangent intersections are corners of the bounding box of a vehicle used in the measurements (filled circles).

Per-Corner Measurement Function

The corner with coordinates $(\hat{x}, \hat{y}, \hat{z})$ in the frame of a vehicle corresponds to a world-coordinate point $\mathbf{c}_w$, with homogenous coordinates given by:

$$\mathbf{c}_w = (x + \hat{x}\cos\phi + \hat{y}\sin\phi, y - \hat{x}\sin\phi + \hat{y}\cos\phi, \hat{z}, 1)^\mathsf{T}, \tag{4.4.29}$$

and the 8 corners we care about are given by the possible combinations of $\hat{x} = \pm\ell$, $\hat{y} = \pm w$, and $\hat{z} = 0, h$.

If $\mathbf{g}$ denotes the image point corresponding to $\mathbf{c}_w$ according to (4.4.7), the measurement function for the particular corner is simply $\mathbf{h}(\mathbf{x}_n, \boldsymbol{\nu}_n) = \mathbf{g} + \boldsymbol{\nu}_n$, where $\boldsymbol{\nu}_n = (\nu_u, \nu_v)^\mathsf{T}$ is a Gaussian random variable representing the image-space measurement noise. We denote the covariance of $\boldsymbol{\nu}_n$ as $\mathbf{R}_n$.

For the purpose of the Extended Kalman filter, we need to be able to evaluate $\mathbf{h}(\mathbf{x}_n, \mathbf{0})$ as already described, and the Jacobians $\mathbf{H}_n = \{\partial \mathbf{h}_i / \partial \mathbf{x}_{n,j}\}_{1 \le i \le 2,\ 1 \le j \le 8}$ and $\mathbf{V}_n = \{\partial \mathbf{h}_i / \partial \boldsymbol{\nu}_{n,j}\}_{1 \le i,j \le 2}$. We omit the expression for $\mathbf{H}_n$ since it is derived in much the same way as the expressions for $\mathbf{A}_n$. Since the measurement noise is additive, $\mathbf{V}_n$ is the identity matrix.

Target Measurement Function

A target's measurement function is simply a composite of the individual corners' measurement functions. However, the task of assigning the tangent line intersections in each view to a subset of the vehicle box corners is not trivial. In our early work (Atev et al. [7]), this problem was solved by a laborious process of geometric reasoning that accounted for the many special cases arising from different viewing angles and vehicle orientations. A simpler solution, adopted in this work, is to find the Maximum Likelihood (ML) assignment of the N_t tangent intersections to a subset of the 8 vehicle corners, subject to the constraint that corners that are endpoints of a vehicle box edge must lie on the same tangent line. The expectation of an integer assignment $a : \{1, 2, \ldots, N_t\} \to \{1, 2, \ldots, 8\}$ is proportional to:

$$\prod_i \exp\left(-\frac{1}{2} (\mathbf{t}_i - \mathbf{g}_{a(i)})^{\mathsf{T}} (\mathbf{H}_{a(i)} \mathbf{P} \mathbf{H}_{a(i)}^{\mathsf{T}})^{-1} (\mathbf{t}_i - \mathbf{g}_{a(i)}) \right), \tag{4.4.30}$$

where $\{\mathbf{t}_i\}_{i=1}^{N_t}$ is the set of N_t tangent intersections, $\{\mathbf{g}_j\}_{j=1}^{8}$ is the set of vehicle corners projected onto the image plane, $\mathbf{H}_j$ denotes the Jacobian of the j-th vehicle corner's measurement function, and $\mathbf{P}$ is the current estimate of the state covariance. Using a branch-and-bound search strategy and the constraint, only about 3 to 6 (of $8!/(8-N_t)!$) assignments are evaluated in most cases to find the ML assignment.

After the best assignment in each view is determined, we further prune the corner measurements with a threshold on their likelihood (given by the individual terms in (4.4.30)), in addition to removing corners that lie on the video frame boundary. This form of measurement gating eliminates outliers which are generated by the tangent-fitting procedure during occlusions and is essentially a test of whether a corner is observable from the given object outline. This gating is critical and is the reason that sometimes we only have partial measurements available.

4.4.4 Constrained State Estimation

We utilize the Extended Kalman filter to estimate the state of each target. In the subsequent discussion, $\mathbf{h}_n(\mathbf{x}, \boldsymbol{\nu})$ refers to a target's composite measurement function (a concatenation of the 2 outputs of each corner's measurement function), $\mathbf{H}_n$ refers to the composite Jacobian (a concatenation of each corner's 2×8 Jacobian), and $\mathbf{R}_n$ refers to the composite measurement noise covariance (a block-diagonal matrix composed of each corner's 2×2 noise covariance). The current estimate of the state covariance is denoted by $\mathbf{P}_n$, and the composite measurement by $\mathbf{z}_n$ (a vector of the same dimension as $\mathbf{h}_n(\mathbf{x}, \boldsymbol{\nu})$ holding tangent intersection coordinates). Quantities with a minus sign in subscript refer to predicted values, a plus sign indicates an unconstrained estimate, and values without either refer to the final, corrected estimates.

The unconstrained estimates of the state are obtained in the standard Extended Kalman filter fashion:

$$\mathbf{P}_n^- = \mathbf{A}_n \mathbf{P}_{n-1} \mathbf{A}_n^\mathsf{T} + \mathbf{W}_n \mathbf{Q}_{n-1} \mathbf{W}_n^\mathsf{T} \tag{4.4.31}$$

$$\mathbf{K}_n = \mathbf{P}_n^- \mathbf{H}_n^\mathsf{T} \left(\mathbf{H}_n \mathbf{P}_n^- \mathbf{H}_n^\mathsf{T} + \mathbf{R}_n \right)^{-1} \tag{4.4.32}$$

$$\mathbf{x}_n^+ = \mathbf{x}_n^- + \mathbf{K}_n \left(\mathbf{z}_n - \mathbf{h}_n(\mathbf{F}(\mathbf{x}_{n-1}, \mathbf{0}), \mathbf{0}) \right) \tag{4.4.33}$$

$$\mathbf{P}_n = (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n) \mathbf{P}_n^-. \tag{4.4.34}$$

Since the elements of the state vector are meaningful physical quantities, we can impose some constraints on the estimates. This improves tracking performance, prevents the filter from diverging, and helps us identify targets such as pedestrians, which consistently violate the constraints. The state covariance estimate $\mathbf{P}_n$ of targets that often violate the constraints fails to converge, which allows us to identify false targets. In contrast, the unconstrained model eventually converges on non-vehicular targets, albeit more slowly.

We impose a number of constraints on a vehicle, summarized in Table 4.1. The constraints are purposefully lax to account for unexpected deviations such as wide-load trucks, mini-vehicles, and motorcycles. The constraints are incorporated into the estimation procedure using the projection method of Simon and Simon [103]. The method, briefly summarized, consists of two steps – identification of the active constraints (currently violated), and the projection of the unconstrained state estimate onto the feasible region of the state space. The k inequality constraints active at time

Table 4.1. Summary of vehicle model constraints used for tracking.

State Variable	Allowed Range	Notes
s	$-15\,^{\mathrm{km}}/_{\mathrm{h}}$ to $160\,^{\mathrm{km}}/_{\mathrm{h}}$	
κ	$-0.5\,\mathrm{m}^{-1}$ to $0.5\,\mathrm{m}^{-1}$	Empirically determined
w	0.6 m to 2.0 m	Must be less than ℓ
ℓ	1.0 m to 30.0 m	Must exceed w
h	0.8 m to 4.9 m	

n must be in the form $\mathbf{D}_n\mathbf{x} \leq \mathbf{d}_n$, where $\mathbf{D}_n$ is a $k \times n$ matrix, $\mathbf{d}_n$ is a k-dimensional vector, and the inequality is componentwise.

The constrained estimate is then obtained as:

$$\mathbf{x}_n = \mathbf{x}_n^+ - \mathbf{G}_n\mathbf{D}_n^\mathsf{T}(\mathbf{D}_n\mathbf{G}_n^{-1}\mathbf{D}_n^\mathsf{T})^{-1}(\mathbf{D}_n\mathbf{x}_n^+ - \mathbf{d}_n), \tag{4.4.35}$$

where $\mathbf{G}_n$ is a positive definite weight matrix. The estimate is unbiased, and in the case of linear systems, [103] proves that the choice $\mathbf{G}_n = \mathbf{P}_n^{-1}$ minimizes the covariance of the estimation error. We adopt the same choice of weighting matrix even though its theoretical optimality does not extend to a linearized system.

4.4.5 Tracking

The initial speed and orientation of vehicles is determined by projecting the image-space velocity vector of their associated region onto the ground plane. The initial dimensions are estimated using the measurement method in [7], which also provides the initial location estimates. The method is similar to the assignment in (4.4.30), but may fail to fail to determine the width and length under some viewing angles, in which case reasonable defaults are used. The initial curvature is assumed to be zero, which is true often enough as new targets at intersections are usually acquired in their approach to the intersection, prior to turning motions, and as vehicles in free-flowing highway traffic usually move straight. Finally, the initial estimate of the state error covariance $\mathbf{P}_0$ is initialized using the variance of the uniform distribution encompassing the full range of allowed values for each parameter as given by Table 4.1.

We keep track of the set of connected regions that overlap each target by making use of two tests: whether the regions (or regions from which they split off) were associated with the vehicle in previous frames, and whether the polygon obtained by

projecting the vehicle model box onto a view has significant fractional overlap with a region from that view. The second test prevents multiple instantiation of targets when a target enters an overlapping view. The convex hull of the set of regions associated with a target provides the input to the methods in Section 4.4.3, which obtain measurements about each target. The measurements are incorporated into the targets' state estimates using equations (4.4.31) through (4.4.35). Targets that do not receive any measurements for several consecutive time steps are removed from further consideration, as are targets whose estimated error covariance **P** fails to decrease significantly from its initial value.

4.5 Results

4.5.1 Single-View Results

Table 4.2. Position and dimension measurement accuracy of tracker. Root-mean-squared errors are reported in centimeters, while average relative errors are reported as percentages. a) Results from all 15 vehicles; b) Excluding a turning vehicle whose long shadow caused severe errors in the width estimate (18 of 292 samples removed).

	Position	Length	Width	Height
a)	72 cm	32 cm / 6%	82 cm / 33%	22 cm / 14%
b)	70 cm	34 cm / 7%	33 cm / 14%	23 cm / 16%

We manually counted the number of cars in 86 regularly spaced video frames. Of the 273 vehicles found, 231 (85%) were correctly identified by our system, 19 (7%) were not identified at all, and 23 (8%) were identified, but merged with other targets in the scene. We then manually fit bounding boxes around 15 different vehicles for several consecutive frames, which gave us 292 individual boxes. The root-mean squared error in the estimated quantities, as well as the average relative error for the three reported vehicle dimensions are indicated in Table 4.2.

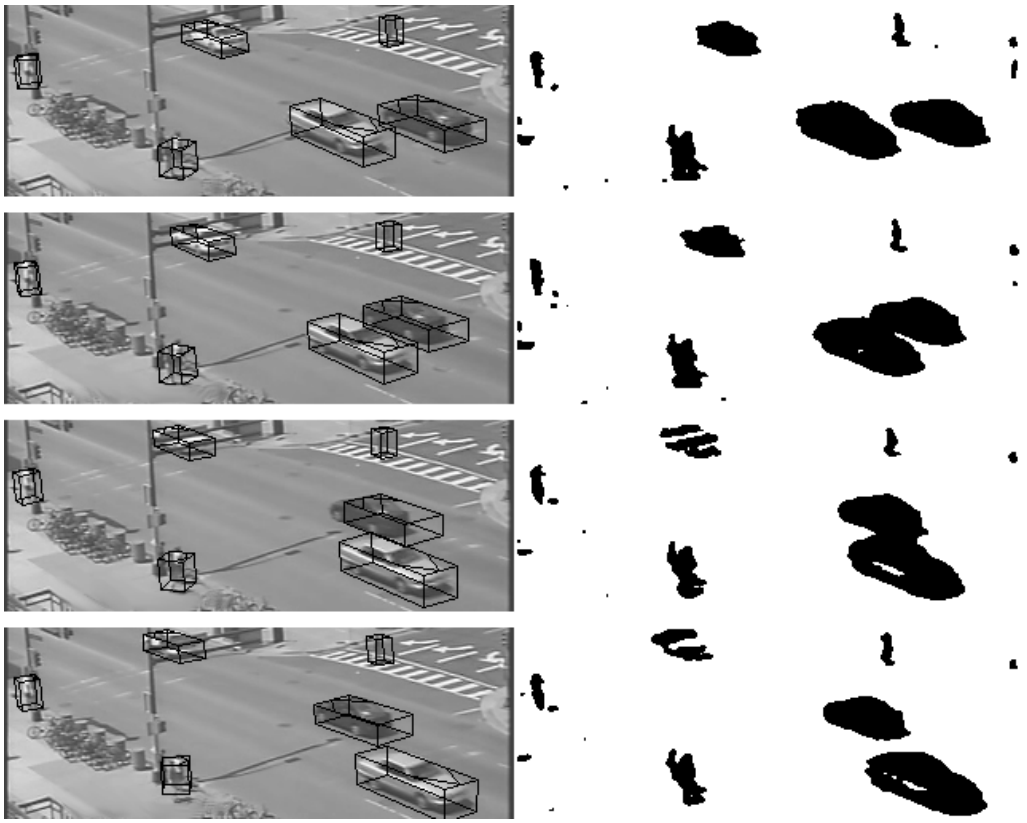


Figure 4.10. A tracking sequence illustrating tracker behavior. Two regions are merged in the second and third frame (dynamic occlusion). The output of the background subtraction after noise removal is shown for reference.

4.5.2 Multi-View Results

Figure 4.11 shows the tracker's state immediately prior (row 1), during (rows 2–4), and immediately after (row 5) a dynamic occlusion. The leftmost column shows the behavior of the tracker when only a single view is used to track. Prior to the occlusion, both vehicles involved receive 6 corner measurements; As the vehicles' associated regions occlude (row 2), each target receives 3 measurements (at this point the width is unobservable, but its estimate has already converged). As the occlusion continues, partial measurements allow the tracking to continue. Immediately after the occlusion has passed, the state estimates have diverged a bit (as the overlap between the bounding boxes and the vehicles indicates), especially for the darker car.

The middle and rightmost columns of Figure 4.11 illustrate the benefits of using a second camera view to track during the same sequence. The targets are occluded

in the second view in row 1, but due to the viewing angle of the second camera, this occlusion is resolved early (in row 3). Note that in row 2, two additional partial measurements supplement the tracking result, which improves the state estimate as compared to the single-view case, where the estimate lags. In the single-view case, the orientation of the darker car has diverged in row 3, whereas the 5 additional corners provided by the second view ensure no such problems occur in the multi-view case.

Figure 4.12 presents an occlusion that lasts longer than the one in Figure 4.11, and where the targets are occluded in both views for about 20 consecutive frames (row 5 comes half a second later than row 4, while rows 1-4 are spaced 1/15 seconds apart). The right column illustrates how the set of available partial measurements changes during the occlusion. While the single-view tracker manages to track the vehicles throughout this occlusion, its state estimates gradually deteriorate. In contrast, the multiple-view tracker maintains a tight fit between estimated and actual vehicle positions, despite having only partial measurements available to it from both views.

4.5.3 Data for Clustering Experiments

We use data from the 4 traffic scenes shown in Figure 4.13 to test the clustering algorithms in Chapter 5 and Chapter 6. Two of the data sets, 1 and 2, are from traffic intersections, and the rest are from free-flowing highway traffic. The input to the algorithms is the set of trajectories after outlier pre-filtering. The ground-truth partitions of trajectories for the four scenes is indicated on each panel. A histogram of trajectory lengths combined over all data sets is shown in Figure 4.14 in order to illustrate the great variability in trajectory lengths (the x-axis is logarithmic).

A top-down view of the ground-truth data (manual labeling) for the four scenes in Figure 4.13 is shown in Figure 4.15. The same data is shown superimposed on the scene views in Figure 4.16 as well.

The trajectories in Figure 4.15 were obtained from the automated tracking method. A trajectory of a tracked object that violates any of the following rules is considered an outlier and not used in our experiments:

1. At least 90 samples (30 for highways) must be available.

2. The object length must exceed the width.

3. The object length must be between 1.5 and 6 meters.

4. The width must be between 1 and 4 meters.

5. The maximum instantaneous velocity over the trajectory must be between 10 and 100 kilometers per hour.

6. The objects' turning rate must be below 45° per frame.

A certain amount of trajectories that conform to all criteria listed above were considered outliers in the ground-truth (the exact count is indicated in Figure 4.15). We retain these outliers in the input to the algorithms in the subsequent chapters in order to test the methods' robustness. This is in contrast with [126], who eliminated all outliers manually in their trajectory clustering experiments. The outliers present in the ground truth are not used in the computation of the Clustering Error and variation of Information metrics of Section 3.3 since none of the methods identify outliers explicitly.

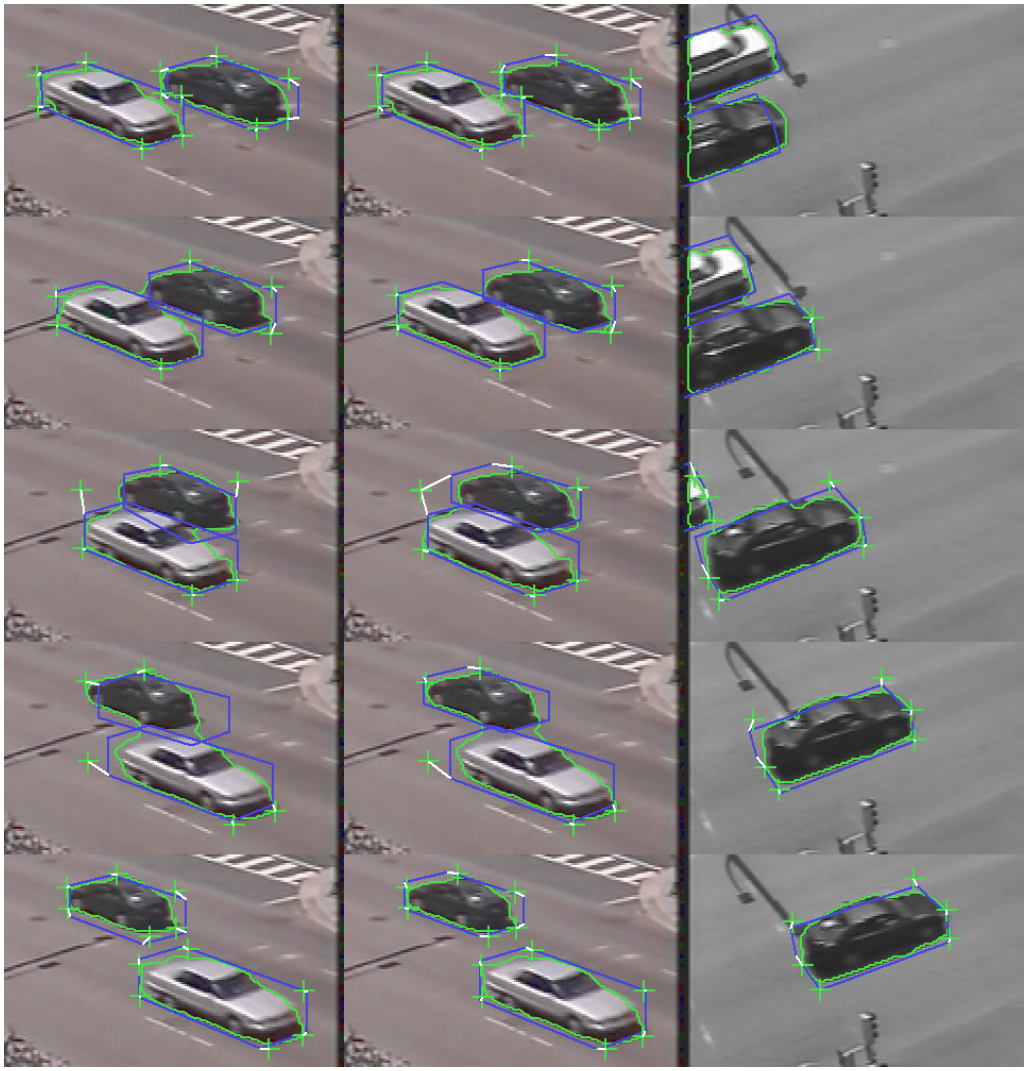


Figure 4.11. A sequence showing the behavior of tracking during a dynamic occlusion. The left column shows the tracker's operation in single-view mode (see Figure 4.10 for a more detailed view of the same interaction). The middle and right columns show multi-view tracking. The bounding boxes depict the estimated state of the targets, the crosses indicate the locations of tangent intersections connected to the vehicle corner assigned to them by the ML process.

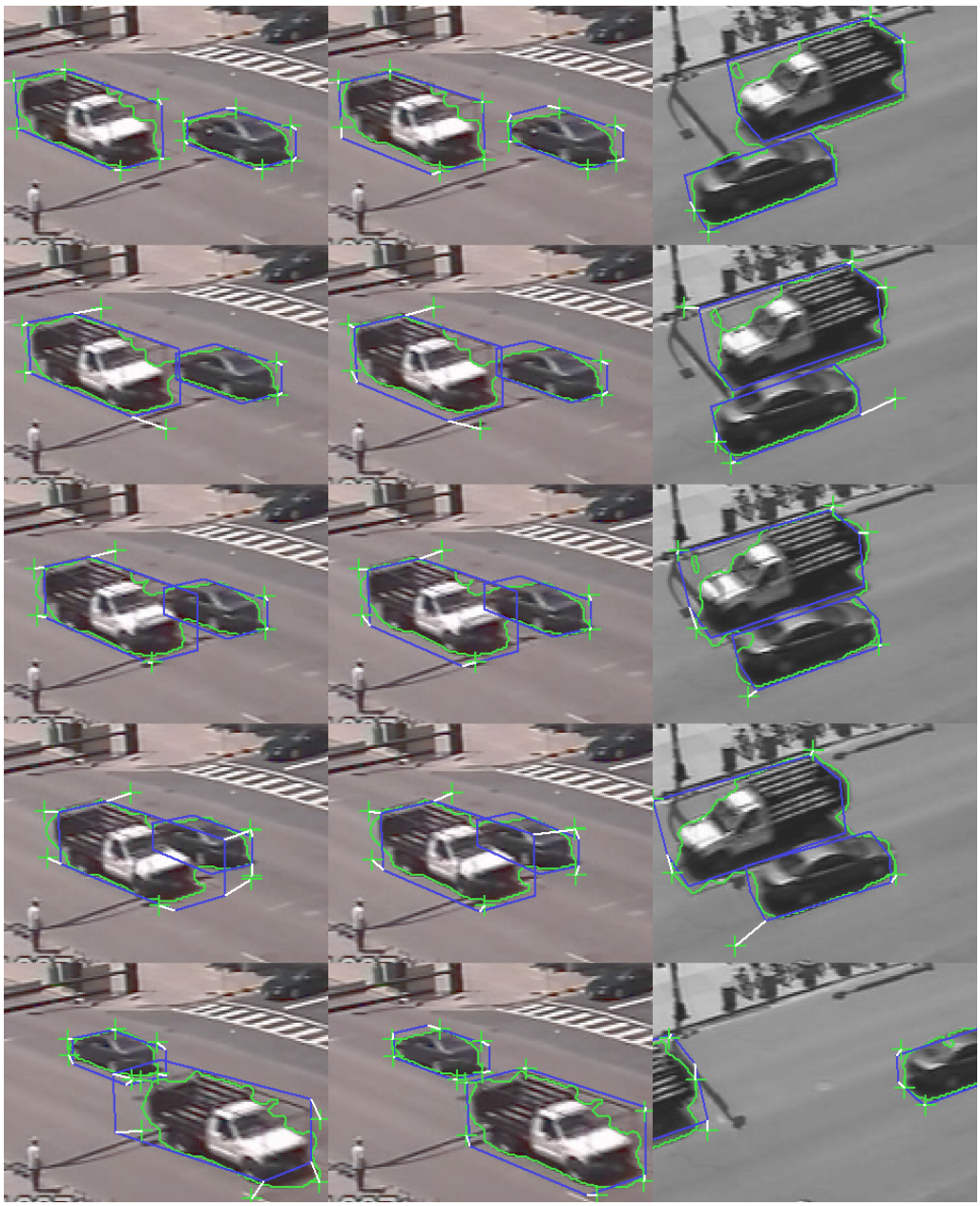


Figure 4.12. A second, longer dynamic occlusion situation. The layout and indications on this figure are the same as in Figure 4.11.

(a) Data set 1 – busy urban traffic scene

(b) Data set 2 – urban intersection with many occlusions

(c) Data set 3 – highway traffic

(d) Data set 4 – busy highway

Figure 4.13. The four scenes from which the trajectories were collected. The first two data sets are from urban traffic intersections and the rest show results from highway traffic.

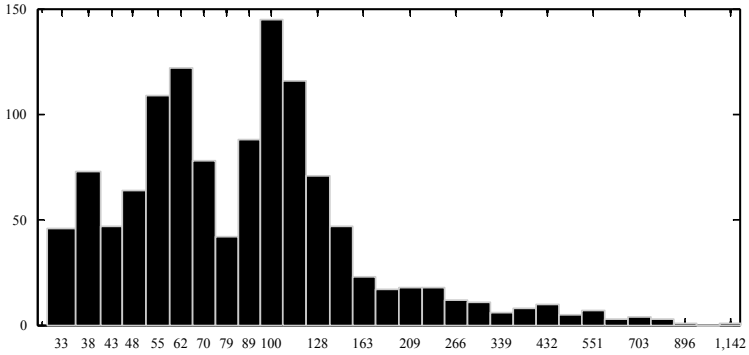
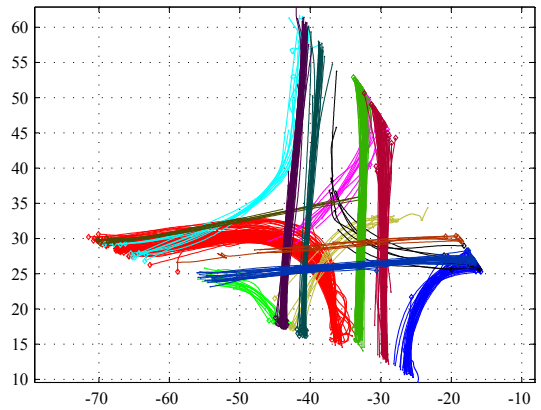
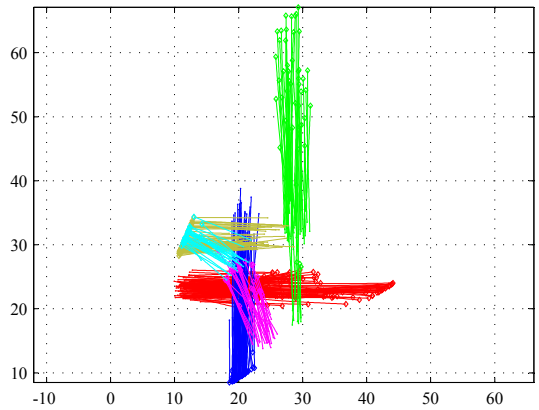


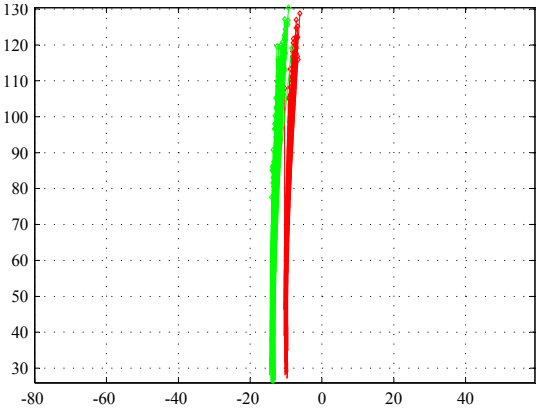
Figure 4.14. Distribution of trajectory lengths (number of points) over all data sets from Figure 4.16. Note that the x-axis scale is logarithmic.



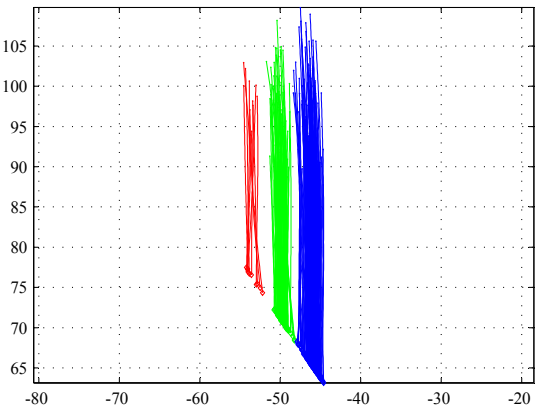
(a) Ground-truth for Figure 4.13a: 403 trajectories, 38(9.4%) outliers



(b) Ground-truth for Figure 4.13b: 391 trajectories, 6(1.5%) outliers

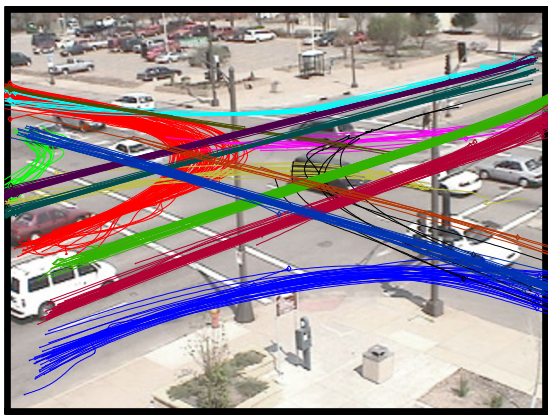


(c) Ground-truth for Figure 4.13c: 161 trajectories, 1(0.6%) outliers

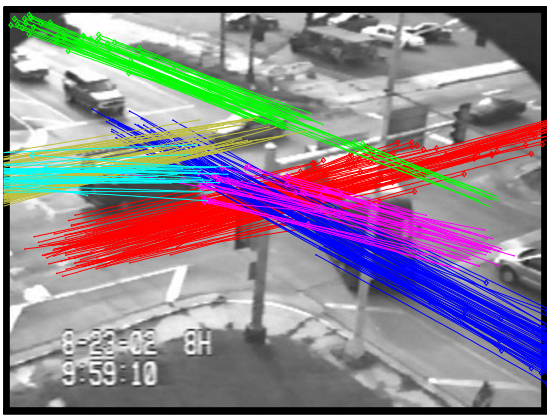


(d) Ground-truth for Figure 4.13d: 240 trajectories, 13(5.4%) outliers

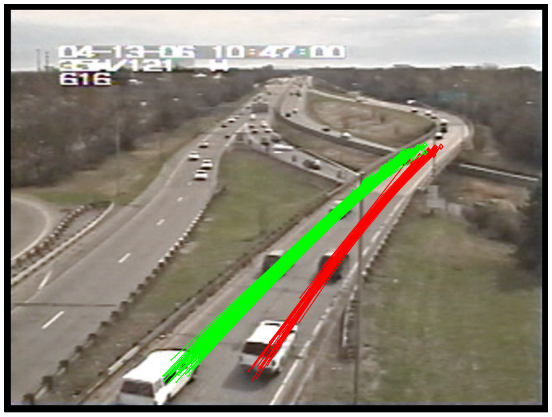
Figure 4.15. Ground truth data for the data scenes in Figure 4.13.



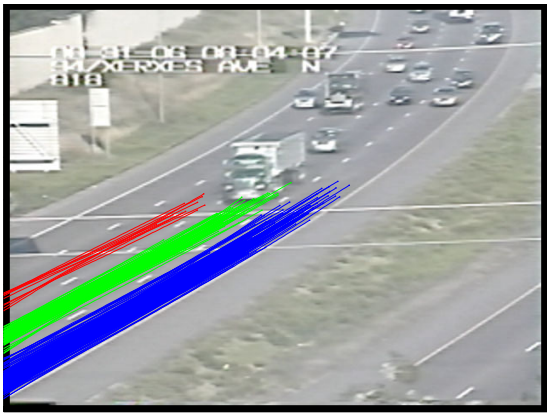
(a) Ground-truth for Figure 4.13a



(b) Ground-truth for Figure 4.13b



(c) Ground-truth for Figure 4.13c



(d) Ground-truth for Figure 4.13d

Figure 4.16. Alternative view of ground truth data for the data scenes in Figure 4.13.

Chapter 5

Modified Hausdorff Distance for Trajectory Clustering

This chapter presents a modified Hausdorff distance for the matching of trajectories. The contributions are in the distance itself, in the particular approach to kernelizing it without first symmetrizing the distance, and in the comparison of the proposed distance to both DTW and LCSS. In addition to spectral clustering, standard agglomerative clustering is also used to provide a baseline. The results of this chapter appear in a somewhat abridged version in Atev et al. [9].

5.1 Introduction

The trajectories of vehicles through a traffic scene are an information-rich feature that reveals the structure of the scene, provides clues to the events taking place in the scene, and allows inference about the interactions between traffic objects. Many vision algorithms are designed to collect trajectory data at various traffic scenes, either as their primary output, or as a by-product of their operation.

The focus of this chapter is to present a method for unsupervised clustering of vehicle trajectories of widely varying lengths. We augment the preliminary results presented in Atev et al. [8] with a comparison of the proposed method with two well-established approaches for trajectory clustering on several sets of trajectories for which ground-truth was manually obtained. An early version of the method has been used in the traffic data collection system described in Atev et al. [7].

Our objective is to compare DTW, LCSS, and a modified Hausdorff distance when applied to the task of clustering trajectories of vehicles at traffic scenes. Since the focus is not on indexing and retrieval, but rather on unsupervised learning, the distances are not approximated; similarly, the fact that the triangle inequality does not hold for DTW and the modified Hausdorff distance is not as relevant as in exact indexing schemes. As our results show, there is no undue computational burden when working with spectral clustering and a moderately sized data sets. The clustering method we use is a modification of the method in Ng et al. [83], which uses a symmetrically normalized graph Laplacian as the affinity matrix. Rather than use a fixed global scale, as [83], we have adopted the scale selection strategy suggested by [123], where local scaling was introduced to the same spectral clustering variant. The discussion in Section 3.2 provides an overview of the relevant methods.

The work of Zhang et al. [126] compares the unmodified Hausdorff distance, LCSS and DTW on a very similar task to the one considered here. Some of the distances tested require resampling of trajectories, which we specifically wish to avoid. The work only uses the uses spectral clustering method of Meilă and Shi [79] but does not provide sufficient detail on how the method is used to allow a direct comparison. Some of the deficiencies of the standard Hausdorff distance identified by [126] are specifically addressed, and as we discover, propel the Hausdorff variant to the top of the results table.

5.2 A Modified Hausdorff Distance

In this section, we propose a trajectory similarity measure based on a modification of the Hausdorff distance. Recall from (3.1.22) that the directed Hausdorff distance between the sets $P \subseteq \mathscr{Z}$ and $Q \subseteq \mathscr{Z}$ is defined as:

$$h(P,Q) = \max_{\mathbf{p}\in P} \left\{ \min_{\mathbf{q}\in Q} d_{\mathscr{Z}}(\mathbf{p},\mathbf{q}) \right\} \tag{5.2.1}$$

and it should be noted that in general, $h(P,Q) \neq h(Q,P)$. As Section 3.1.3 mentions, the standard extension of h to a metric is to define $H(P,Q) = \max\{h(P,Q), h(Q,P)\}$. We do not consider the case when P or Q may be uncountable sets as we deal with finitely-sampled time series. As will become apparent in our discussion of local scaling, we will not use the symmetric distance H, but a variant of the semi-metric h.

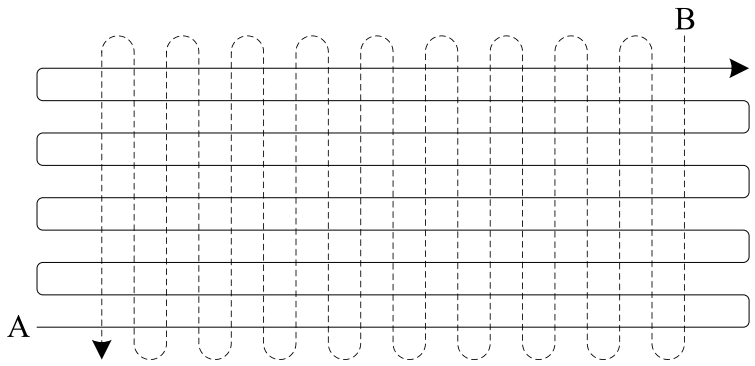


Figure 5.1. An example of two trajectories A and B, which are close to each other if they are treated as unordered point sets, as each point in A has a point from B nearby.

One deficiency of the Hausdorff distance is the fact that even a single outlier point in either P or Q can result in an arbitrarily large distance between the sets, even if they are otherwise identical. Also, time series are ordered collections of points, and the Hausdorff distance is defined on unordered collections; as Figure 5.1 illustrates, two trajectories that are quite different may be close together under the Hausdorff distance if treated as unordered point sets.

We address the task-specific deficiencies of the Hausdorff distance by modifying it in two ways: we reject a certain proportion of the worst matches under the maximum in (5.2.1), and restrict the set under which the minimum is taken to a subset of the second set:

$$h_{\alpha,N,C}(P,Q) = \operatorname*{ord}_{\mathbf{p}\in P}^{\alpha} \left\{ \min_{\mathbf{q}\in N_Q(C_{P,Q}(\mathbf{p}))} d(\mathbf{p},\mathbf{q}) \right\}, \tag{5.2.2}$$

where $C_{P,Q} : P \to Q$ is a function that assigns to each point $\mathbf{p} \in P$ a corresponding point $\mathbf{q} \in Q$, and $N_Q : Q \to \mathscr{P}(Q)$ is a function that specifies a subset of Q as the neighborhood of the point $q \in Q$. Together, N_Q and $C_{P,Q}$ impose a structure over which the matching is performed. The operator $\operatorname{ord}_{s\in S}^{\alpha} f(s)$ denotes the value among the image $f(S)$ of the set S that is greater than $\alpha \left| f(S) \right|$ of all the values. Note that the directed Hausdorff distance is a special case of (5.2.2) since $h(P,Q) = h_{1,N,C}(P,Q)$ for any combination of maps N_Q and $C_{P,Q}$ that satisfy $N_Q(C_{P,Q}(\mathbf{p})) = Q$ for all $\mathbf{p} \in P$ (i.e., an unrestricted correspondence structure). The meaning of α can be intuitively

understood using the following facts:

$$\operatorname*{ord}_{s\in S}^{1} f(s) = \max_{s\in S} f(s) \tag{5.2.3}$$

$$\operatorname*{ord}_{s\in S}^{0} f(s) = \min_{s\in S} f(s) \tag{5.2.4}$$

$$\operatorname*{ord}_{s\in S}^{1/2} f(s) = \operatorname*{median}_{s\in S} f(s). \tag{5.2.5}$$

The rejection of a certain proportion of large distances in the Hausdorff distance has been suggested for edge template matching by [37]. The properties of the distance that make it a good candidate for shape matching are identified in [56]. The imposition of a correspondence structure on the two sets was not considered in these early works as they treated objects and shapes as unordered collections of points.

A choice of $\alpha < 1$ makes the distance robust to a fraction of outliers $1-\alpha$, but also allows for partial trajectories to match and can lead to violations of the triangle inequality. As [80] points out, such violations can result in inconsistent clustering. As our early results in [8] indicated, the overall clustering results are sensitive to the choice of α – in runs of the clustering algorithm with different values of α, increasing differences in α led to increasingly different clustering results as more and more partial matches are allowed. These two considerations prompted us to fix the value of α to 0.88 in subsequent experiments. The value was determined experimentally in our earlier work.

Our choices for a correspondence function C and a neighborhood structure N both rely on a parameterization of the trajectories. Since our goal is to discover spatial structures of the trajectories, we chose the arc-length parameterization. Given a trajectory $P = \{\mathbf{p}_i\}_{i=1}^{n}$ with $\mathbf{p}_i \in \mathbb{R}^2$, we first define the length of P as:

$$|P| = \sum_{i=2}^{n} \left\| \mathbf{p}_i - \mathbf{p}_{i-1} \right\|_2, \tag{5.2.6}$$

which can be defined for any kind of series using the same norm as the one used in the Hausdorff distance for matching. The extension of this length definition to a more general $d_{\mathscr{Z}}$ is trivial.

For a trajectory P, we define the subsequence formed by the first k points as $P_k = \{\mathbf{p}_i\}_{i=1}^{k}$ (so $P = P_n$), and define the relative position of the j^{th} point $\mathbf{p}_j$ in P as

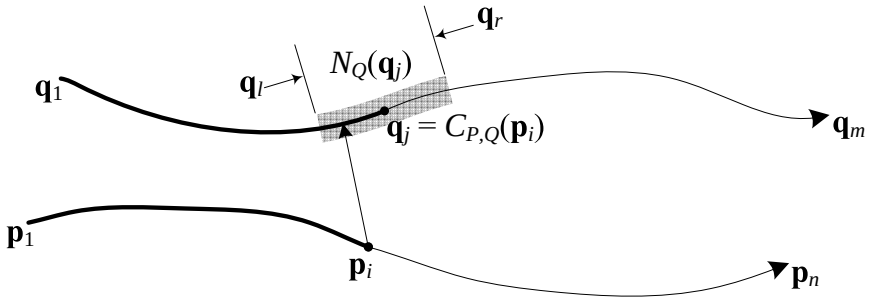


Figure 5.2. An illustration of the correspondence function $C_{P,Q}$ and neighborhood N_Q as used in the definition of $h_{\alpha,N,C}$. The following relationships hold for points indicated on the diagram: $\pi_P(\mathbf{p}_i) = \pi_Q(\mathbf{q}_j)$: the relative positions of $\mathbf{p}_i$ in P and its corresponding point $\mathbf{q}_j$ in Q are the same; $\pi_Q(\mathbf{q}_j) - \pi_Q(\mathbf{q}_l) = \pi_Q(\mathbf{q}_r) - \pi_Q(\mathbf{q}_j) = w/2$: the arc-length of the neighborhood $N_Q(\mathbf{q}_j)$ is w, and $\mathbf{q}_j$ is centered in it.

$\pi_P(\mathbf{p}_j)$:

$$\pi_P(\mathbf{p}_j) = \frac{|P_j|}{|P|}. \tag{5.2.7}$$

Using the notation $P(j) = \mathbf{p}_j$, we also define the reverse transformation $r_P : [0,1] \to P$ as:

$$r_P(a) = P\left(\underset{1\le j\le n}{\operatorname{argmin}} \left| a - \frac{|P_j|}{|P|} \right|\right). \tag{5.2.8}$$

The correspondence structure $C_{P,Q}$ is then defined as:

$$C_{P,Q}(\mathbf{p}) = Q(r_Q(\pi_P(\mathbf{p}))), \tag{5.2.9}$$

while the neighborhood structure N_Q is defined as:

$$N_Q(\mathbf{q}_0) = \{\mathbf{q} \in Q \mid \left|\pi_Q(\mathbf{q}_0) - \pi_Q(\mathbf{q})\right| < w/2\}, \tag{5.2.10}$$

where the parameter w controls the relative size of the neighborhood. Note that w has a similar function to the parameter δ of the LCSS-based distance $L_{\epsilon,\delta,\rho}$, but that the underlying parameterization is not time-based as it is in LCSS.

An illustration of $C_{P,Q}$ and N_Q is given in Figure 5.2, which should make the somewhat abstract definitions more clear.

Now that the parameters α and w are specified, we will denote the Hausdorff distance as $h_{\alpha,w}$, where it is understood that N_Q and $C_{P,Q}$ use the arc-length parameterization for correspondence and w for neighborhood size. The standard spectral clustering method of [83] uses affinities (similarities) using the Gaussian kernel with a global scale parameter σ:

$$k_{\mathrm{G}}(\mathbf{p},\mathbf{q}) = \exp\left(-\frac{d^2(\mathbf{p},\mathbf{q})}{2\sigma^2}\right), \tag{5.2.11}$$

where $d^2(\mathbf{p},\mathbf{q})$ is the squared distance between $\mathbf{p}$ and $\mathbf{q}$. A problem with using a fixed global scale parameter σ is that this assumes that clusters are equally dense and have similar scales. Because spectral clustering is very sensitive to the choice of scale, and since the selection of a proper value for σ is non-trivial, [123] proposed a local scaling method that estimates the scale of the data in the neighborhood of each data point:

$$k_{\mathrm{L}}(\mathbf{p},\mathbf{q}) = \exp\left(-\frac{1}{2}\frac{d(\mathbf{p},\mathbf{q})^2}{\sigma(\mathbf{p})\sigma(\mathbf{q})}\right), \tag{5.2.12}$$

where the scale σ is now dependent on the data points. The standard way to apply the locally scaled kernel calls for the use of a symmetric distance. Instead, we split the squared distance in (5.2.12) into a product of two directed distances:

$$k_{\mathrm{D}}(\mathbf{p},\mathbf{q}) = \exp\left(-\frac{1}{2}\frac{d(\mathbf{p},\mathbf{q})}{\sigma(\mathbf{p})}\frac{d(\mathbf{q},\mathbf{p})}{\sigma(\mathbf{q})}\right). \tag{5.2.13}$$

The splitting of the squared distance into the product of $d(\mathbf{p},\mathbf{q})$ and $d(\mathbf{q},\mathbf{p})$ suggests that there is no need for d to be symmetric (the kernel will be symmetric regardless), which means we can use the directed Hausdorff distance $h_{\alpha,w}$ directly. Intuitively, $d(\mathbf{p},\mathbf{q})/\sigma(\mathbf{p})$ is the distance from $\mathbf{p}$ to $\mathbf{q}$ as seen on a scale appropriate for the neighborhood of $\mathbf{p}$, and $d(\mathbf{q},\mathbf{p})/\sigma(\mathbf{q})$ is the distance from $\mathbf{q}$ to $\mathbf{p}$ as seen on a scale appropriate for $\mathbf{q}$'s neighborhood. If the underlying distance is symmetric, k_{D} is equivalent to k_{L}. As Chapter 6 demonstrates, even the symmetry of the kernel can be relaxed with appropriate modifications to the underlying spectral clustering.

We choose the scale $\sigma(\mathbf{p})$ based on the distance of $\mathbf{p}$ to its 9^{th} nearest neighbor, using similar reasoning to that of [123]. The algorithm produces practically identical results with the 5^{th} and 7^{th} nearest neighbors, so we kept a value consistent with our

earlier work. If the σ value is outside of the range 0.4 and 2 meters, we clip the value to the appropriate boundary. With that definition, we can write the final similarity measure we use as:

$$k(\mathbf{p},\mathbf{q}) = \exp\left(-\frac{h_{\alpha,w}(\mathbf{p},\mathbf{q})h_{\alpha,w}(\mathbf{q},\mathbf{p})}{2\sigma(\mathbf{p})\sigma(\mathbf{q})}\right), \tag{5.2.14}$$

which is used whenever we need to use the modified Hausdorff distance in a spectral clustering framework.

5.2.1 Connections with DTW and LCSS

At this point we would like to summarize the differences and similarities between the proposed distance and the LCSS and DTW methods described in Section 3.1.2 and Section 3.1.1 respectively. The actual distances computed by both DTW and the modified Hausdorff distance depend on an underlying metric directly, and have the same units as the point distance itself; in contrast, LCSS always returns a unitless distance that is based on a count. The one-dimensional LCSS distance between two series of length N can take on only $N+1$ distinct values as it ultimately depends on an integer count that can range between 0 and N. The normalization of the DTW distance is somewhat ambiguous when two sequences of widely differing lengths are compared as the normalization factor depends on the number of points of only one of the series, while the modified Hausdorff distance is based on a robust norm that needs no normalization by the number of points at all. The parameterization of LCSS used to control the "slack" in the matching is time-based (i.e., the parameter δ limits a correspondence neighborhood based on time), while the proposed Hausdorff distance controls the matching based on the arc-length, which is important when the series under comparison have longer periods without change (e.g., a vehicle stopped to wait for a traffic light does not change its position). While both DTW and the modified Hausdorff distance allow for a meaningful computation of relative distances, the LCSS distance is less discriminating because of the early application of the threshold ϵ.

The time complexity of all distances in general is $O(mn)$, while the space requirements are linear. A benefit of the Hausdorff distance is that it allows for an embarrassingly parallel implementation on systems with multiple processors, unlike DTW and LCSS which depend on dynamic programming approaches that are non-

trivial to parallelize.

5.3 Clustering Methods

5.3.1 Agglomerative Clustering

The agglomerative clustering method we use is identical to the one in [20] — each trajectory is initially assigned a singleton cluster. The two closest clusters are iteratively merged until all clusters are separated by a distance greater than a pre-specified threshold τ. If I and J denote the sets of indices of the trajectories of two disjoint clusters, the distance between the clusters is:

$$d_{\text{avg}}(I,J) = \frac{1}{|I|\,|J|}\sum_{i\in I}\sum_{j\in J} d_{ij}. \tag{5.3.1}$$

In our tests, we specify the desired number of clusters rather than choosing a value of τ since the number of clusters is known from the manually labeled ground-truth data. The range of values for τ that produce the same number of clusters is indicative of the sensitivity of the underlying distance measure and will be part of our experiments.

5.3.2 Spectral Clustering Modifications for Trajectory Clustering

We refer to Section 3.2 for an overview and notation for spectral clustering. We use the scale selection strategy of [123] described by (3.2.2) and (5.2.12), with the modification explained in (5.2.14). We do not use their proposed clustering method for the spectrally embedded data (which we generalize in Chapter 6).

In an ideal, noise-free case, the matrix $S(\mathbf{A}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is block diagonal and has g eigenvalues with value 1, and $n-g$ zero eigenvalues, where g is the actual number of clusters in the data. Unfortunately, counting the number of eigenvalues of $S(\mathbf{A})$ that are equal to 1 is not a practical way of figuring out the number of clusters as pointed out in [123]. However, we can at least find a suitable range for the number of clusters using the spectrum of $S(\mathbf{A})$. Let $1 \geq \lambda_1 \geq \lambda_2 \geq \cdots \lambda_n \geq 0$ be the eigenvalues of $S(\mathbf{A})$. We estimate the minimum number of clusters in the data, $g_{\min}$, by counting how many eigenvalues are greater than 0.99, and the maximum number of clusters

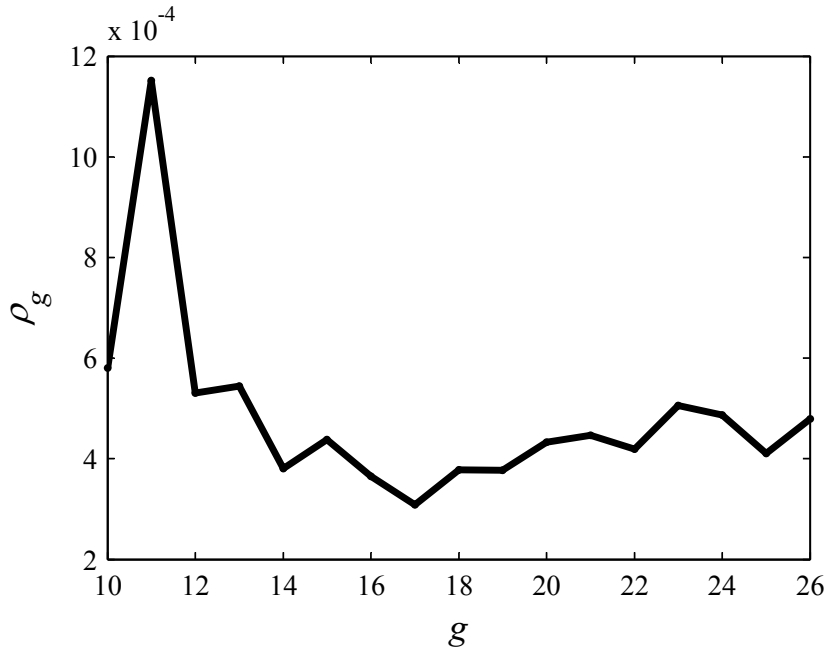


Figure 5.3. Distortion ρ_g for data set 1 of Figure 4.13 using the Hausdorff distance.

$g_{\max}$ by counting how many eigenvalues are greater than 0.8. These thresholds were chosen experimentally. For some of our tests, the possible range was very narrow, and for others quite large.

Once we choose the search range for the number of clusters, we compute the spectral embedding $\mathbf{Y} = \mathbf{V}_k^{\mathsf{T}} \operatorname{diag}^{-1}(\operatorname{diag}(\mathbf{V}_k \mathbf{V}_k^{\mathsf{T}}))$ as described in Section 3.2. As [83] propose, we define the distortion score $\rho_g = W/(T-W)$ as the ratio of the within-class to between-class scatter. The within-class scatter W is simply the objective function minimized by k-means, and T is the total sum of squared distances between all spectrally embedded points and the class centers identified by k-Means.

The value of g that produces the least distortion ρ_g is the automatically determined number of clusters, and the $c(i)$ obtained with that number of clusters is the class indicator function for the input trajectories. We show a plot of ρ_g versus g in Figure 5.3. Note that the actual cost ρ_g is not monotonic or well-behaved, so a search over all reasonable values of g is necessary.

5.4 Experiments

The experiments in this section are on the data sets introduced in Section 4.5.3, and evaluated against the ground truth segmentation shown in Figure 4.15.

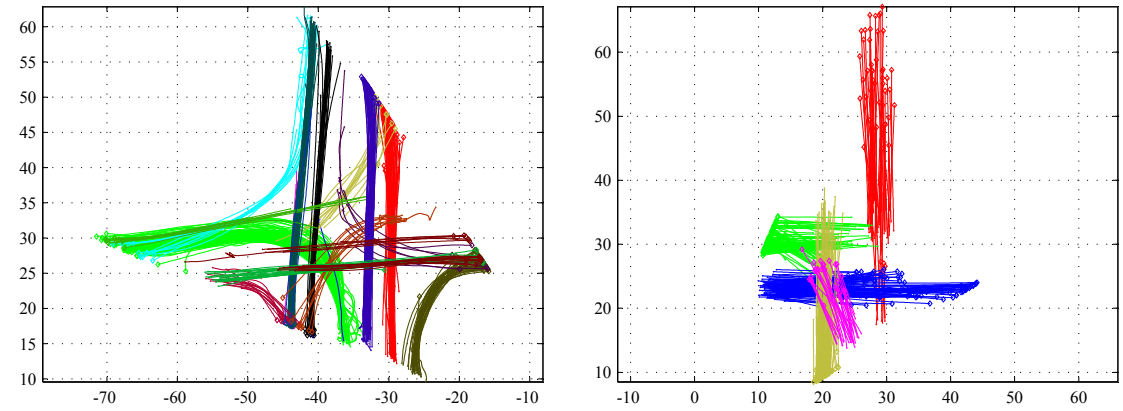
To the extent possible, we use equivalent parameters in all similarity measures. A particular problem with LCSS was the fact that for some choices of ϵ, the algorithm was unable to produce the desired number of clusters since too many of the trajectories were at infinite distance from each other (the range for the LCSS distance in $\mathbb{R}^2$ is $[\sqrt{2}/2, \infty)$, but any two trajectories separated by a distance exceeding ϵ everywhere result in a distance of ∞). In such situations, we show the results of the algorithm using two different settings of ϵ – the first one is 12 feet (the lane-width in the scenes), and the second one is the smallest value of ϵ that was large enough so that the desired number of clusters was produced.

In cases where the automatically detected number of clusters differs from the true number of clusters, we present two scores for the proposed algorithm. To give an idea of the suitability of the proposed trajectory distance, we also used agglomerative clustering with the modified Hausdorff distance.

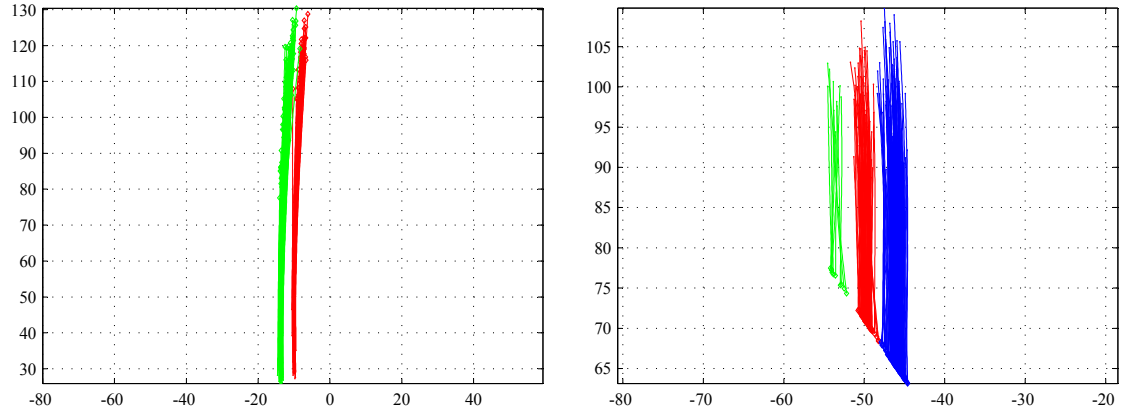
For LCSS and the modified Hausdorff distance, we used a slack parameter of one-half the trajectory lengths, that is, we set $w = 0.5$ for the modified distance and $\delta = 0.5$ for LCSS (very similar to the value in [20]). Finally, the parameter α for the Hausdorff method was set to 0.88, which was determined to be optimal on a subset of data set 1 in [8]. Note that the optimal α in [8] was chosen without recourse to the ground-truth data, it was the value that minimized the distortion score discussed in Section 5.3.2. The same value was used on all other data sets with good results and was not optimized further for each of the remaining data sets.

5.4.1 Results

The Variation of Information between the various clustering methods and the ground-truth is shown in Table 5.1. The Clustering Error is also shown for clusterings that produce the same number of clusters as the ground truth. The actual number of clusters in the input is known to all algorithms, except for HS, DS, and LS-ϵ, which determine it automatically. The LCSS-based methods indicate the value of ϵ used as well, since it varied in one of the experiments. The best result for each distance measure (as measured by d_{VI}) is highlighted in *italics*, and the best overall result is



(a) Data set 1 **(b)** Data set 2



(c) Data set 3 **(d)** Data set 4

Figure 5.4. A top-down view of the output of the proposed method on the four data sets. The grid units are in meters.

highlighted in **bold**. The method names in Table 5.1 are given in table Table 5.2.

A top-down view of the output of the proposed HS method is shown in Figure 5.4.

5.4.2 Discussion

The first very clear result from Table 5.1 is that the modified Hausdorff distance better reflects the notion of trajectory similarity; even when used in an agglomerative fashion, the distance outperforms both DTW and LCSS. The modified Hausdorff distance also outperforms the other distances when used with the spectral clustering algorithm. The second observation is that when given the correct number of clusters, the spectral methods outperform the equivalent agglomerative method with the LCSS

and Hausdorff distances. The fact that the LCSS distance used with larger thresholds does not benefit from the spectral clustering is a result of the unreasonably high thresholds used (which were necessary so that the agglomerative method produced the required number of clusters). The advantages of the spectral clustering are especially clear in data set 3, where the clusters contain many partially overlapping trajectories due to the widely varying starting point of successful tracking. While data set 3 appears very simple, the great variability in trajectory length and trajectory endpoints challenges all agglomerative clustering variants since the inter-cluster and intra-cluster distances both have large variances. The freeway data sets did not challenge the spectral method at all – both the number of clusters and class memberships were recovered perfectly. Surprisingly, HS outperformed HM on the second data set, even though it underestimated the true number of clusters. This can be explained by the fact that HS merged two clusters that had significant overlap, but otherwise produced results identical to the ground-truth for all other trajectories. Due to the overlap, HM also merged the same clusters, but it split another sparse cluster into two sub-clusters in order to obtain the specified 7 clusters.

The performance of LCSS for this problem, when used in the agglomerative fashion, is discouraging. When given a very reasonable ϵ threshold, equal to the actual inter-lane width, LA-12 failed to merge enough clusters to reach the specified correct number of clusters. The problem was that LCSS produced only two types of distances – very close to the minimum possible, and ∞, and any outliers or well-separated clusters produced many infinite distances. Increasing ϵ improved the performance of the method for the freeway data sets, but had the opposite effect on the intersection data sets, and affected negatively the spectral clustering results.

While the performance of DTW was not very good, we believe this is mostly due to the very strict constraints of (3.1.2) – (3.1.5) that require the starting points of the trajectories to be matched. Due to the unpredictability of target initialization in the tracker and the effects of foreshortening, trajectories in the same cluster could have varying starting and ending positions. Since the Hausdorff distance does not impose such a constraint, it has a clear advantage over DTW on data set 3, and to a lesser extent on data set 4. We were surprised by the poor results when using DTW in a spectral framework, and have no definitive explanation for this observation, except to note that on the data set with the largest intra-cluster variation in trajectory lengths, data set 1, DS and DM did not perform well, while the same methods outperformed

DA for data set 2, which exhibits the least variation in intra-cluster trajectory lengths.

5.4.3 Time Cost of the Methods

There are two significant factors affecting the running time of the compared methods – the computation of a pairwise distance matrix, and the actual clustering. The asymptotic complexity of the first step is very similar for all of the compared methods; in practice, as Table 5.3 shows, the modified Hausdorff distance and LCSS have very similar running times, and DTW is slower by a constant factor. The reason for the longer run-time of DTW is that we do not use a band-limited version of the distance (the slack parameters δ for LCSS and w for the proposed distance reduce the running time). All implementations of this component are in C++ but have not been heavily optimized.

The running times of spectral clustering and agglomerative clustering are comparable. Both times are overstated for different reasons. In the case of spectral clustering, we compute all eigenvalues and eigenvectors of $S(\mathbf{A})$, even though in general we only need a few of the top ones. This is a limitation of the dense eigensolver in Matlab. The agglomerative clustering is also implemented in Matlab, but makes less use of built-in functions (e.g., `eig` and `kmeans`) and as such is running at a performance disadvantage.

5.4.4 Sensitivity of the Agglomerative Clustering

Figure 5.5 illustrates the behavior of the agglomerative clustering with the various distance measures. The width of the steps indicates the range over which the distance threshold can be varied without changing the number of clusters. Both DTW and the modified Hausdorff distance behave as expected – as the number of clusters gets smaller, the threshold range increases. The modified Hausdorff distance is very insensitive to the threshold when a few clusters are desired, but is more sensitive than DTW for smaller thresholds. The erratic behavior of the LCSS distance is due to the fact that the distances returned are either very small, or infinite. As the plot shows, LCSS is unable to merge clusters past a given point as all distances become infinite. Note that the steep portions of the plots for the modified Hausdorff distance occur prior to the correct number of clusters is reached for each of the four data sets. This means that the correct number of clusters may be easier to detect automatically

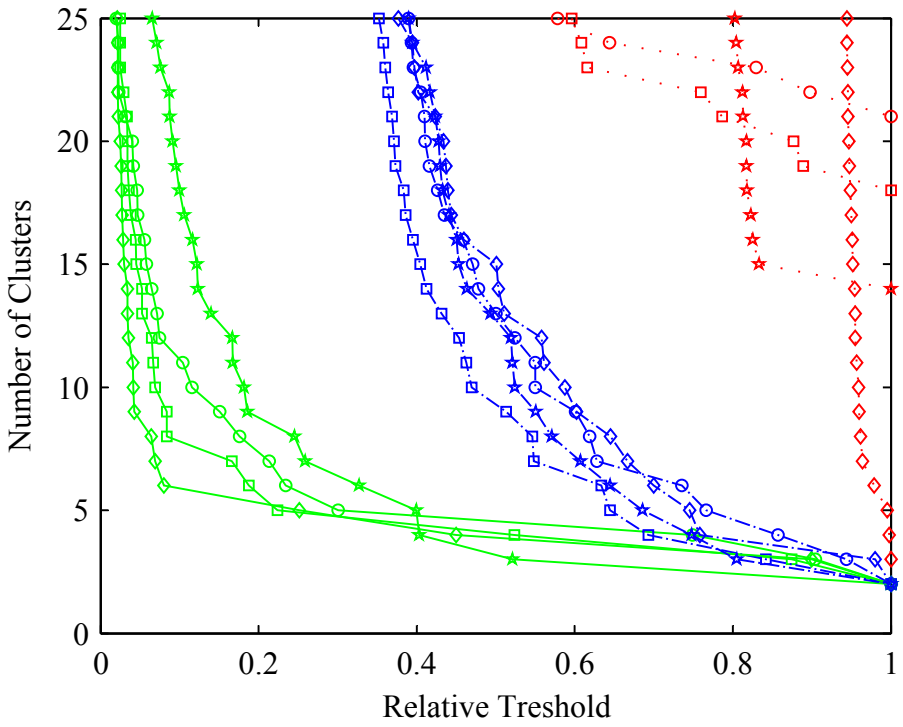


Figure 5.5. A plot of the threshold in the agglomerative clustering versus the number of generated clusters. The solid green lines are for the modified Hausdorff distance, the dash-dot blue lines are for DTW, and the short-dash red lines are for the LCSS distance with $\epsilon = 12$. The threshold is relative to the largest finite value obtained (i.e., the ∞ of LCSS is ignored). The lines for the data sets are marked with circles, squares, diamonds, and five-pointed stars respectively.

using the Hausdorff distance by searching for the knee point on the cluster size vs. the threshold graph.

5.5 Conclusions

We presented a trajectory similarity measure based on the directed Hausdorff distance and applied it with both a spectral and an agglomerative clustering method. The imposition of some correspondence structure between the trajectories is a modification necessary to make the Hausdorff distance applicable for the comparison of ordered point sets, while the rejection of a certain amount of worst matches in the distance

improves its robustness to noise and increases the tolerance to partial matches. Rather than use a distortion metric to select a globally optimal scale parameter, as [83] suggest, we used the metric to determine the optimal number of clusters from a set of possibilities obtained by examination of the spectrum of the affinity matrix. The local scale selection from [123] performed very well for the clustering of trajectories, hence there was no need to perform a search for any scale parameters.

In general, the average distance over corresponding pairs of points used by DTW seemed more likely to be influenced by outliers than the robust trimmed maximum we utilized in the modified Hausdorff distance. The relaxation of the starting/ending point constraints in DTW can also improve the suitability of the distance for matching trajectories with differing lengths and allow for some partial matches. However, it is not clear that both the monotonicity constraint of (3.1.11) and the trimmed maximum of (5.2.2) can be combined in a method of complexity $O(mn)$ for comparing an n-element and an m-element trajectory.

The LCSS based distance measure performed disappointingly even on simpler data sets. Mostly, trajectories were considered very similar or very dissimilar, with no apparent middle case. This would make the LCSS distance unsuitable for use in spectral clustering with local scaling as the relative distances would be similarly clustered at a few extreme values. Using a robust loss in the DTW distance should allow for some of the desirable qualities of LCSS to be emulated.

Table 5.1. Comparison to ground-truth clustering for the four scenes in Figure 4.16.

Method	Data set 1			Data set 2			Data set 3			Data set 4		
	k	d_{VI}	d_{CE^*}	k	d_{VI}	d_{CE^*}	k	d_{VI}	d_{CE^*}	k	d_{VI}	d_{CE^*}
	Dynamic Time Warping											
DA	*14*	*0.671*	*0.255*	6	1.048	0.486	2	0.913	0.438	*3*	*0.474*	*0.070*
DS	24	1.321	–	*6*	*0.535*	*0.229*	*3*	*0.852*	–	4	0.700	–
DM	14	1.114	0.375	*6*	*0.535*	*0.229*	2	0.961	0.537	3	0.647	0.370
	Modified Hausdorff Distance											
HA	14	0.124	0.044	6	0.373	0.151	2	0.895	0.419	**3**	**0.000**	**0.000**
HS	17	0.191	–	**5**	**0.152**	–	**2**	**0.000**	**0.000**	**3**	**0.000**	**0.000**
HM	**14**	**0.102**	**0.030**	6	0.213	0.101	**2**	**0.000**	**0.000**	**3**	**0.000**	**0.000**
	LCSS With $\epsilon = 12$											
LA-12	21	1.477	–	18	1.509	–	3	1.107	–	14	2.229	–
LS-12	26	1.477	–	*5*	*0.558*	–	3	0.664	–	*3*	*0.743*	*0.176*
LM-12	*14*	*1.314*	*0.397*	6	0.611	0.239	*2*	*0.610*	*0.119*	*3*	*0.743*	*0.176*
	LCSS With Varying ϵ											
	$\epsilon = 22$			*$\epsilon = 43$*			*$\epsilon = 15$*			*$\epsilon = 30$*		
LA-ϵ	*14*	*1.600*	*0.468*	6	1.943	0.561	*2*	*0.895*	*0.419*	*3*	*1.751*	*0.595*
LS-ϵ	26	1.731	–	15	2.148	–	3	1.491	–	4	1.927	–
LM-ϵ	14	1.637	0.490	*6*	*1.887*	*0.517*	2	1.185	0.331	3	1.770	0.551

Table 5.2. Method name abbreviations used in table Table 5.1 along with a summary of the method type and behavior with respect to the choice of the number of clusters.

Name	Distance	Type	Setting of k
HA	Hausdorff	Agglomerative	Manual
HS	Hausdorff	Spectral	Automatic
HM	Hausdorff	Spectral	Manual
DA	DTW	Agglomerative	Manual
DS	DTW	Spectral	Automatic
DM	DTW	Spectral	Manual
LA-ϵ	LCSS	Agglomerative	Manual
LS-ϵ	LCSS	Spectral	Automatic
LM-ϵ	LCSS	Spectral	Manual

Table 5.3. Running time in seconds of the selected methods from Table 5.1 on the data sets from Figure 4.15.

Data Set (Size)	Method	Similarity	Clustering	Total
1 (403)	HS	13.8	1.19	14.99
	HA	13.8	1.03	14.83
	LA-12	19.7	2.26	21.96
	DA	269.9	1.03	270.93
2 (391)	HS	1.44	0.62	2.06
	HA	1.44	0.91	2.35
	LA-12	1.56	2.08	3.64
	DA	20.6	0.92	21.52
3 (161)	HS	0.89	0.075	0.965
	HA	0.89	0.039	0.929
	LA-12	0.97	0.048	1.018
	DA	13.8	0.038	13.838
4 (240)	HS	0.81	0.18	0.99
	HA	0.81	0.10	0.91
	LA-12	0.88	0.27	1.15
	DA	11.6	0.10	11.7

Chapter 6

Clustering with Asymmetric Similarity Measures

6.1 Introduction

In this chapter we address the problem of clustering data based on pairwise similarity measurements that are not necessarily symmetric. The problem of symmetry showed up in Chapter 5, where we postponed the symmetrization of the modified Hausdorff distance until after local scaling. Since results improved by postponing symmetrization until the very last step prior to spectral clustering, the question arose on whether it was possible to avoid this step altogether. The rest of this chapter answers this affirmatively by presenting concrete ideas and methods to avoid symmetrization.

Asymmetry can arise for various reasons — due to the use of asymmetric distance measures such as the directed Hausdorff distance, Levenstein distances, and certain formulations of Dynamic Time Warping, or due to useful transformations of symmetric measures such as local scaling. For example, Figure 6.1 shows an asymmetric similarity arising from the use of directed Dynamic Time Warping to compare handwritten characters based on pen velocity.

The kernel view of spectral clustering (i.e., spectral clustering is kernel PCA followed by traditional clustering) is not very informative in such situations, since work in kernel methods rarely handles asymmetric kernels, even if the study of indefinite kernels has attracted sufficient attention. The graph view of spectral clustering (i.e., spectral clustering as an approximate solution to a graph cut problem) offers hints

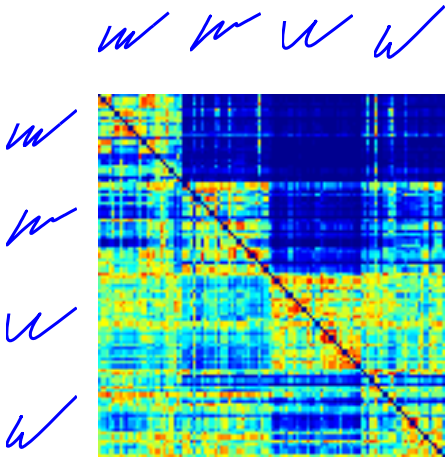


Figure 6.1. Sample asymmetric affinity between handwritten characters from the M, N, U, and W classes.

as to the utility of clustering in directed graphs, but the complex eigenvectors and eigenvalues that arise in such problems are usually treated in a somewhat *ad hoc* manner. Our approach is twofold — we encode an asymmetric similarity measure in a derived Hermitian similarity measure to which the kernel view applies (most results on real-valued kernels extend to complex-valued ones), and we also provide a method to recover a clustering (extract the underlying indicator matrix) from the spectral embedding resulting from the use of the Hermitian measure. The method handles both real-valued and complex-valued eigenvectors and is an efficient alternative to other methods that try to "undo" a rotation applied to an idealized cluster membership indicator matrix. In a number of experiments we demonstrate that using the original asymmetric similarity is superior to using a symmetrized version of the measure, and that the proposed method for recovering the cluster assignments is competitive with other methods both on symmetric and asymmetric similarities.

The rest of the chapter is organized as follows: In Section 6.2 we discuss related work in spectral clustering, focusing on methods that support asymmetric similarities. We also discuss our approach for handling asymmetric affinities. The rotation-based clustering algorithm designed to work with complex eigenvectors is discussed in Section 6.3. Our experiments are then presented in Section 6.4.

6.2 Related Work

Asymmetric distances are common when dealing with biological sequences and speech, and in analyzing social, citation, or other types of networks. The focus of Meilă and Pentney [78], for example, is in constructing a symmetric matrix that in some sense best represents the asymmetric similarities obtained from biological sequence data. The method is applied to analyzing an Internet hyperlink network as well. Similar work by the same authors demonstrates the importance of preserving asymmetry when clustering biological sequence data [87]. While [87] simply adds the imaginary and real parts of complex eigenvectors whenever they arise, we use a method that directly makes use of the complex-valued vectors. The idea of using a Hermitian adjacency matrix appears in the context of social network analysis in Hoser and Geyer-Schulz [51], Hoser and Schröder [52], but the clustering method used after spectral embedding is not analyzed.

There is a clear relationship between the method proposed here and the k-lines method of [39], which fits lines through the origin to the spectral embedding of the data. Our contribution is to show how the optimization of the k-lines objective function can be performed under orthogonality constraints. The theoretical motivation for imposing the orthogonality constraint is provided in the analysis of spectral clustering in Ng et al. [83]. The "rotation" method presented in Zelnik-Manor and Perona [123] also optimizes a clustering objective under orthogonality constraints. The major difference in our approach is the use of a sophisticated and efficient optimization procedure, and in using an objective function that has a clear geometric interpretation. In contrast with all of the aforementioned methods, our algorithm can be used on asymmetric affinity problems without modification. Note that if our method is used with a symmetric similarity measure, all intermediate operations are performed on real-valued objects so the symmetric case simply falls out as a special case of the more general, asymmetric case.

We want to emphasize the kernel view of spectral clustering, hence the desire to use a Hermitian matrix — while kernels are typically real, most of the results apply to complex-valued kernels. The kernel view allows us to apply results from Fowlkes et al. [40] or Bengio et al. [15] to do out-of-sample extension to the results, and is potentially useful in semi-supervised learning and classification problems.

6.2.1 Spectral Decomposition with Asymmetric Affinities

Asymmetric affinities can arise in various applications, and a typical approach for users of spectral clustering is to symmetrize them since most well-known methods require symmetric affinities. The most common approach is to use either $1/2\left(\mathbf{A}+\mathbf{A}^{\mathsf{T}}\right)$ or $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ in lieu of $\mathbf{A}$ in the spectral decomposition step. Of those approaches, the additive approach has the benefit that it does not modify the affinity when it happens to be symmetric, while the second approach has the benefit of always producing a positive semi-definite similarity measure. A more sophisticated approach that still leads to a symmetric eigenproblem is presented in [78], which differs subtly from the additive symmetrization: the latter leads to

$$S(\mathbf{A}) = \mathbf{D}_{\text{sym}}^{-1/2}(1/2(\mathbf{A}+\mathbf{A}^{\mathsf{T}}))\mathbf{D}_{\text{sym}}^{-1/2}, \tag{6.2.1}$$

with $\mathbf{D}_{\text{sym}} = 1/2(\mathbf{D}_{\text{in}} + \mathbf{D}_{\text{out}})$, while [78] uses only $\mathbf{D}_{\text{out}}$ to normalize (see Table 3.1).

In all cases, it is important to remember that the affinity matrix is the information bottleneck — no other information is available to the spectral clustering algorithm. One feature of the symmetrization procedures is that they throw away information. As [87] shows, the information that is thrown out may be highly relevant in certain problems. The strength of the approach in [78] is that it is shown that in some sense, the information most relevant to the clustering objective is preserved better, a result quantified in [78, Proposition 5.1].

Our approach is to construct a Hermitian matrix from the affinity $\mathbf{A}$, namely:

$$H(\mathbf{A}) = \frac{1}{2}\left(\mathbf{A}+\mathbf{A}^{\mathsf{T}}\right) + i\frac{1}{2}\left(\mathbf{A}-\mathbf{A}^{\mathsf{T}}\right), \tag{6.2.2}$$

which presents the subsequent clustering algorithm with the exact same data as the asymmetric $\mathbf{A}$ since the mapping $H(\mathbf{A})$ is a bijection:

$$\mathbf{A} = \Re H(\mathbf{A}) + \Im H(\mathbf{A}). \tag{6.2.3}$$

We solve $H(\mathbf{A})\mathbf{x} = \lambda\mathbf{x}$, and denote the solution with $\mathbf{V}\mathbf{L}\mathbf{V}^{\mathsf{H}} = H(\mathbf{A})$. Our notation uses $\mathbf{X}^{\mathsf{H}}$ for the conjugate transpose of $\mathbf{X}$ and $\mathbf{X}^{*}$ to denote the conjugate of $\mathbf{X}$.

We use a Hermitian matrix for several reasons. The spectrum of $H(A)$ is real, so the eigenvector selection approach needs no modification. A Hermitian matrix can still

be interpreted as a kernel matrix, and most results in the kernel methods literature will continue to apply. The particular choice of $H(\mathbf{A})$ from (6.2.2) is appealing since a symmetric affinity would lead to a real problem. The benefits of this choice are also discussed in [52]. Note that we do not specify specific transformations of the affinity prior to the construction of the Hermitian matrix — that choice is left unspecified. In our experiments, we have used $H(\mathbf{D}_{\text{out}}^{-1/2}\mathbf{A}\mathbf{D}_{\text{in}}^{-1/2})$, $H(\mathbf{D}_{\text{out}}^{-1}\mathbf{A})$ or simply $H(\mathbf{A})$.

6.2.2 Eigenpair Selection and Embedding

In all experiments, we simply chose the k eigenvectors corresponding to the k largest eigenvalues. We also tried the approach to eigenpair selection proposed in Jenssen et al. [62], but did not observe an improvement in performance over using the graph Laplacians and choosing the largest k eigenvalues. When using $H(\mathbf{A})$ directly in the spectral decomposition step, the selection method may be preferable, but we did not conduct further experiments with the method.

We use the same spectral embedding as Kernel PCA and MDS, namely $\mathbf{Y} = \mathbf{L}_k^{-1/2}\mathbf{V}_k^{\mathsf{T}}$. Since we work with complex numbers, the scaling is well-defined even in the case that an eigenvalue is negative. Section 3.2 lists alternative approaches that can be used.

6.2.3 Cluster Assignment

The starting point for the cluster assignment are the spectral embedding coordinates. While we make no reference to how these coordinates were obtained, it must be emphasized that the orthogonality constraint we impose is motivated by the analysis in [83] and [123], so in some sense, we assume that the affinities we work with are block-diagonal (possibly perturbed).

When the transformation in (6.2.2) is used, $\mathbf{Y} \in \mathbb{C}^{k\times n}$ in general, with $\mathbf{Y} \in \mathbb{R}^{k\times n}$ if $\mathbf{A}$ is symmetric. In the ideal case, all of the points $\mathbf{y}_1, \ldots, \mathbf{y}_n$ lie on a set of k mutually perpendicular lines through the origin. In the general case, when the embedding is produced by a perturbed ideal affinity matrix, we will seek to find a set of k mutually perpendicular lines such that the sum of squared distances from each point to the nearest line is minimized. We call our algorithm the k-Axes algorithm in analogy with the k-lines algorithm of [39], which shares the same objective function, but differs in the imposition of orthogonality constraints to the lines. Unlike [123], the cost function we optimize under orthogonality constraints is smooth, and lends itself to

optimization by the efficient Riemannian Conjugate Gradient method proposed by Abrudan et al. [1]. The optimization approach in [123] uses gradient descent over a principal angles parametrization of the rotation matrix. Small steps have to be taken so that the constraints are not violated, and the parametrization suffers from singularities as an angle approaches $-\pi$ or π. A final difference is that we can use soft-assignment with our cost function, while in [123], the objective function requires a hard cluster membership decision at each outer iteration of the algorithm, a problem shared with an earlier rotation-based method presented in Yu and Shi [122].

We continue the discussion of the k-Axes algorithm in the next section, where we present the cost function and optimization strategy.

6.3 The k-Axes Algorithm

The clustering problem we are trying to solve can be written as the discrete minimization problem:

$$\begin{aligned} \text{minimize} \quad & J(\mathbf{Y}, \mathbf{U}, \mathbf{W}) \\ \text{subject to} \quad & \mathbf{U}^{\mathsf{H}}\mathbf{U} = \mathbf{I} \\ & w_{ij} \in \{0, 1\} \\ & \sum_{j=1}^{k} w_{ij} = 1, \end{aligned} \tag{6.3.1}$$

where $J(\mathbf{Y}, \mathbf{U}, \mathbf{W})$ is the k-Axes cost function:

$$J(\mathbf{Y}, \mathbf{U}, \mathbf{W}) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} \left(d^{\perp}(\mathbf{y}_i, \mathbf{u}_j) \right)^2. \tag{6.3.2}$$

The function $d^{\perp}(\mathbf{y}, \mathbf{u})$ measures the orthogonal distance from the point $\mathbf{y}$ to the line through the origin along the direction specified by the unit vector $\mathbf{u}$. The notation $\mathbf{u}_j$ simply denotes the j-th column of $\mathbf{U}$, and the $n \times k$ matrix $\mathbf{W}$ encodes the cluster membership of the columns of $\mathbf{Y}$. The function $d^{\perp}(\mathbf{y}, \mathbf{u})$ can be expressed as:

$$d^{\perp}(\mathbf{y}, \mathbf{u}) = \sqrt{\mathbf{y}^{\mathsf{H}}(\mathbf{I} - \mathbf{u}\mathbf{u}^{\mathsf{H}})\mathbf{y}}. \tag{6.3.3}$$

Algorithm 1 The k-Axes algorithm for clustering of spectrally embedded data.

Initialization

Let $\mathbf{U}_0 = \mathbf{I}$ and $\sigma_0 = \max_{i=1}^{n} \min_{j=1}^{k} \left|y_{ij}\right|$. Compute $\mathbf{W}_0$ using (6.3.4). Let $J_0 = J(\mathbf{Y}, \mathbf{U}_0, \mathbf{W}_0)$.

For $t = 1, 2, \ldots$

Compute the softmax scale $\sigma_t = (J_{t-1}/n)^{t/2}$.

Use $\mathbf{U}_{t-1}$ as an initial solution and solve the following using the methods of Section 6.3.1:

$$\mathbf{U}_t = \arg\min_{\mathbf{U}^{\mathsf{H}}\mathbf{U}=\mathbf{I}} J(\mathbf{Y}, \mathbf{U}, \mathbf{W}_{t-1}). \tag{6.3.5}$$

Update the weights $\mathbf{W}_t$ using (6.3.4) and σ_t, and let $J_t = J(\mathbf{Y}, \mathbf{U}_t, \mathbf{W}_t)$. If J_t is not appreciably smaller than J_{t-1}, stop. Otherwise, proceed with the next iteration.

Result Generation

Assign each point to the cluster with the largest weight in $\mathbf{W}_t$. The weights can be used as a confidence measure.

We solve (6.3.1) by alternating between steps that modify $\mathbf{U}$ and $\mathbf{W}$ and reduce the value of J. Keeping $\mathbf{W}$ fixed, we need to minimize J under the orthogonality constraints on $\mathbf{U}$. Then, keeping $\mathbf{U}$ fixed, we modify $\mathbf{W}$ as follows:

$$w_{ij} = \frac{\exp(-d^{\perp}(\mathbf{y}_i, \mathbf{u}_j)^2/\sigma^2)}{\sum_{\ell=1}^{k} \exp(-d^{\perp}(\mathbf{y}_i, \mathbf{u}_\ell)^2/\sigma^2)}, \tag{6.3.4}$$

where σ determines the "softness" of the softmax function defined by (6.3.4). While this means that the intermediate values for $\mathbf{W}$ do not satisfy all constraints of (6.3.1), there is a benefit in the early iterations of the process since $\mathbf{U}$ changes less erratically, so the number of Conjugate Gradient iterations is lower overall. In practice, we decrease σ extremely fast after the first two iterations, so $\mathbf{W}$ quickly approaches a binarized cluster membership matrix.

Algorithm 1 summarizes our approach. The algorithm converges, because each of the steps reduces the objective: $J(\mathbf{Y}, \mathbf{U}_t, \mathbf{W}_{t-1}) \leq J(\mathbf{Y}, \mathbf{U}_{t-1}, \mathbf{W}_{t-1})$ by the definition of $\mathbf{U}_t$ in (6.3.5), and $J(\mathbf{Y}, \mathbf{U}_t, \mathbf{W}_t) \leq J(\mathbf{Y}, \mathbf{U}_t, \mathbf{W}_{t-1})$ since $\sigma_t \leq \sigma_{t-1}$ leads to a "sharpening" of the softmax function defined by (**??**). We omit the details of this proof, but it can be

shown that the gradient of $J(\mathbf{Y}, \mathbf{U}, \mathbf{W}(\sigma, \mathbf{Y}, \mathbf{U}))$ with respect to σ is negative (or zero) when $\mathbf{W}(\sigma, \mathbf{Y}, \mathbf{U})$ is computed according to (**??**).

A note on the initial choice $\mathbf{U}_0 = \mathbf{I}$ is in order. While this can be a very poor initialization for a k-means procedure or the k-lines algorithm, that is not the case with the *k*-Axes method. The orthogonality constraints prove useful in this regard as they impose a certain rigidity on $\mathbf{U}$. Intuitively, all k axes are coupled through the constraints so all data points, even those far away from a given axis, contribute information for the optimization, which is especially relevant in the early stages of optimization when the process may still be far away from the eventual minimum. In k-means and k-lines, data points only contribute to the estimation of the closest center or line.

6.3.1 Minimization with Orthogonality Constraints

The description of Algorithm 1 is incomplete until we demonstrate how the minimization of (6.3.5) works. We use the method of [1] to perform the minimization, which requires us to specify an appropriate gradient of $J(\mathbf{Y}, \mathbf{U}, \mathbf{W})$ with respect to $\mathbf{U}$. Since $\mathbf{U} \in \mathbb{C}^{k \times k}$, the appropriate gradient is given by the complex conjugate derivative operator $\frac{\partial}{\partial \mathbf{U}^*} = 1/2 \left(\frac{\partial}{\partial \Re \mathbf{U}} + i \frac{\partial}{\partial \Im \mathbf{U}} \right)$.

$$
\begin{aligned}
\frac{\partial J}{\partial \mathbf{U}^*} &= \frac{\partial}{\partial \mathbf{U}^*} \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} d^{\perp}(\mathbf{y}_i, \mathbf{u}_j)^2 \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{U}^*} \sum_{j=1}^{k} w_{ij} \mathbf{y}_i^{\mathsf{H}} (\mathbf{I} - \mathbf{u}_j \mathbf{u}_j^{\mathsf{H}}) \mathbf{y}_i \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{U}^*} \left(\mathbf{y}_i^{\mathsf{H}} \mathbf{y}_i - \sum_{j=1}^{k} w_{ij} \mathbf{y}_i^{\mathsf{H}} \mathbf{u}_j \mathbf{u}_j^{\mathsf{H}} \mathbf{y}_i \right) \\
&= -\sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{U}^*} \left(\sum_{j=1}^{k} w_{ij} \mathbf{y}_i^{\mathsf{H}} \mathbf{u}_j \mathbf{u}_j^{\mathsf{H}} \mathbf{y}_i \right). \qquad (6.3.6)
\end{aligned}
$$

Code Listing 1 MATLAB® code for the gradient of $J(\mathbf{Y},\mathbf{U},\mathbf{W})$ with respect to $\mathbf{U}^*$ given a data matrix $\mathbf{Y}=(\mathbf{y}_1,\ldots,\mathbf{y}_n)$ and a weight matrix $\mathbf{W}$.

```
function dU = gradient(U,Y,W)
    n = size(U,1);
    for i = 1:n
        Z = (Y .* repmat(W(:,i)',[n,1]));
        dU(:,i) = -Z * (Y' * U(:,i));
    end
```

We will evaluate (6.3.6) term-by-term for a given row r and column c, at a fixed data point i (so we omit the index i for convenience):

$$\begin{aligned}
\left[\frac{\partial J}{\partial \mathbf{U}^*}\right]_{rc} &= -\frac{\partial}{\partial u^*_{rc}}\left(\sum_{j=1}^{k} w_j \mathbf{y}^{\mathsf{H}}\mathbf{u}_j\mathbf{u}_j^{\mathsf{H}}\mathbf{y}\right)\\
&= -\sum_{j=1}^{k} w_j \frac{\partial}{\partial u^*_{rc}}\left((\mathbf{y}^{\mathsf{H}}\mathbf{u}_j)(\mathbf{u}_j^{\mathsf{H}}\mathbf{y})\right)\\
&= -\sum_{j=1}^{k} w_j \frac{\partial}{\partial u^*_{rc}}\left(\left(\sum_{p=1}^{k} y_p^* u_{pj}\right)\left(\sum_{q=1}^{k} y_q u_{qj}^*\right)\right)\\
&= -\sum_{j=1}^{k} w_j \frac{\partial}{\partial u^*_{rc}}\left(\sum_{p=1}^{k}\sum_{q=1}^{k} y_p^* u_{pj} y_q u_{qj}^*\right)\\
&= -w_c \frac{\partial}{\partial u^*_{rc}}\left(\sum_{p=1}^{k} y_p^* u_{pc} y_r u_{rc}^*\right)\\
&= -w_c \sum_{p=1}^{k} y_p^* u_{pc} y_r\\
&= -w_c y_r \mathbf{y}^{\mathsf{H}}\mathbf{u}_c, \qquad (6.3.7)
\end{aligned}$$

which can be computed easily as shown in the code fragment 1.

It is important to note that $\frac{\partial J}{\partial \mathbf{U}^*}$ will not vanish at the optimal solution, but its projection onto the manifold of unitary matrices will be zero.

6.3.2 Periodicity Order of the Objective

The Riemannian Conjugate Gradient method uses the almost-periodicity of smooth functions along geodesics in order to do line search efficiently. In this section we compute the periodicity order of the k-Axes cost function since that information is needed by the method.

Using a polynomial approximation, the first zero-crossing of $\frac{\partial J}{\partial \mathbf{U}^*}$ along a search direction $\mathbf{H} \in \mathbb{C}^{k \times k}$ is sought. The search is over the interval $\left[0, \frac{2\pi}{q|\omega_{\max}|}\right)$, where $\omega_{\max}$ is the eigenvalue of $\mathbf{H}$ with largest magnitude, and q is the order of h in the Taylor expansion of $T(h) = J(\mathbf{Y}, \mathbf{U} + h\mathbf{Z}, \mathbf{W})$ around $h = 0$.

Expanding $T(h)$ leads to:

$$\begin{aligned}
T(h) &= \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} d^{\perp}(\mathbf{y}_i, \mathbf{u}_j + h\mathbf{z}_j)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} \mathbf{y}_i^{\mathsf{H}} (\mathbf{I} - (\mathbf{u}_j + h\mathbf{z}_j)(\mathbf{u}_j^{\mathsf{H}} + h\mathbf{z}_j^{\mathsf{H}}))\mathbf{y}_i \\
&= J_{\mathrm{L}}(\mathbf{Y}, \mathbf{U}, \mathbf{W}) - \cdots \\
&\cdots \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} \mathbf{y}_i^{\mathsf{H}} (h\mathbf{z}_j\mathbf{u}_j^{\mathsf{H}} + h\mathbf{u}_j\mathbf{z}_j^{\mathsf{H}} + h^2\mathbf{z}_j\mathbf{z}_j^{\mathsf{H}})\mathbf{y}_i \\
&= T(0) - hf_1(\mathbf{Y}, \mathbf{U}, \mathbf{Z}) - h^2 f_2(\mathbf{Y}, \mathbf{U}, \mathbf{Z}),
\end{aligned} \tag{6.3.8}$$

where $f_1(\mathbf{Y}, \mathbf{U}, \mathbf{Z})$ and $f_2(\mathbf{Y}, \mathbf{U}, \mathbf{Z})$ do not depend on h. But that means that $T(h)$ is a second-degree polynomial in h, hence the highest power of h in its expansion is 2. With that, we determine that the periodicity order of the function is $q = 2$, and that is the value used in the polynomial approximation for the line search.

The minimization procedure under orthogonality constraints is outlined in Algorithm 2. Figure 6.2 illustrates some of the variables introduced in the algorithm description and shows the difference between a conjugate gradient approach and a gradient descent approach.

6.3.3 Convergence Speed of Algorithm 1

We conclude this discussion with an example of the convergence behavior of the algorithm. Figure 6.3 shows the value of the objective and the number of CG iterations

Algorithm 2 Conjugate Gradient with orthogonality constraints [1, Table 3].

Initialization

Let $\mathbf{U}_0$ be an initial guess with $\mathbf{U}_0^{\mathsf{H}}\mathbf{U}_0 = \mathbf{I}$. Compute $\mathbf{\Gamma}_0 = \frac{\partial}{\partial \mathbf{U}^*} J(\mathbf{U}_0)$, and let $\mathbf{G}_0 = \mathbf{\Gamma}_0\mathbf{U}_0^{\mathsf{H}} - \mathbf{U}_0\mathbf{\Gamma}_0^{\mathsf{H}}$. Finally, set $\mathbf{H}_0 = \mathbf{G}_0$.

For $k = 0, 1, \ldots$

If $\operatorname{tr}(\mathbf{G}_k^{\mathsf{H}}\mathbf{G}_k)$ is small, return $\mathbf{U}_k$ and stop.

Let $\omega_{\max}$ be the eigenvalue of $\mathbf{H}_k$ of largest magnitude. Set $T_{\max} = \frac{2\pi}{2|\omega_{\max}|}$.

Perform the line search:

$$\mu = \underset{\mu \in [0, T_{\max})}{\arg\min} J(\mathbf{Y}, \exp(-\mu\mathbf{H}_k)\mathbf{U}_k, \mathbf{W}), \tag{6.3.9}$$

using the polynomial approximation for the first zero crossing of $\frac{\partial J}{\partial \mathbf{U}^*}$ along $-\mathbf{H}_k$ [1, Table 1]. Perform the updates:

$$\begin{aligned}
\mathbf{U}_{k+1} &= \exp(-\mu\mathbf{H}_k)\mathbf{U}_k \\
\mathbf{\Gamma}_{k+1} &= \frac{\partial}{\partial \mathbf{U}^*} J(\mathbf{U}_{k+1}) \\
\mathbf{G}_{k+1} &= \mathbf{\Gamma}_{k+1}\mathbf{U}_{k+1}^{\mathsf{H}} - \mathbf{U}_{k+1}\mathbf{\Gamma}_{k+1}^{\mathsf{H}} \\
\mathbf{H}_{k+1} &= \mathbf{G}_{k+1} + \frac{\Re \operatorname{tr}\left((\mathbf{G}_{k+1} - \mathbf{G}_k)^{\mathsf{H}}\mathbf{G}_k\right)}{\operatorname{tr}(\mathbf{G}_k^{\mathsf{H}}\mathbf{G}_k)}\mathbf{H}_k
\end{aligned}$$

If $\Re \operatorname{tr}(\mathbf{H}_{k+1}^{\mathsf{H}}\mathbf{G}_{k+1}) < 0$, reset $\mathbf{H}_{k+1} = \mathbf{G}_{k+1}$.

---

at each iteration of Algorithm 1 on the Iris data set, while Table 6.1 shows the number of outer and CG iterations during the processing of the data sets discussed in Section 6.4.3. Note that we run the CG process to full convergence, which is likely unnecessary, especially in the initial k-Axes iterations.

6.4 Experiments

We replicate or repeat experiments in a number of previous spectral clustering methods in order to demonstrate the applicability of k-Axes. Our first set of experiments replicates the synthetic problems in [123] and demonstrates that when using the same (symmetric) affinity matrix, k-Axes performs equally well to the "rotation" method

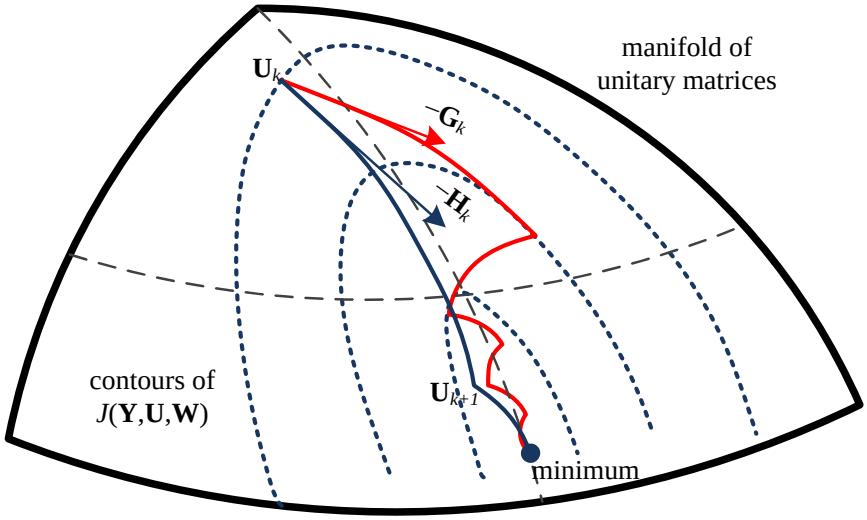


Figure 6.2. Illustration of the search direction $\mathbf{H}_k$ and gradient $\mathbf{G}_k$ from Algorithm 2. Adapted from [1].

proposed by the authors both in clustering and model selection. We then replicate a synthetic experiment similar to those in [78] in order to show that a direct use of an asymmetric affinity can be superior to the use of normalized cuts with a symmetrized similarity. The experiments in [39] are replicated in order to evaluate the impact of the orthogonality constraint added in the *k*-Axes algorithm, and to show situations in which the symmetrizing of the affinity matrix is unnecessary.

We evaluate the performance of the various algorithms using the Variation of Information distance discussed in detail in Section 3.3.

6.4.1 Local Scaling and Model Selection

We retrieved the data sets used in [123] and ran the *k*-Axes algorithm on the data using locally scaled affinity matrices. The search for the optimal number of clusters was performed over the same range (2 to 5 clusters) using minimum mean squared error as the criterion for selecting the number of clusters. Our results are identical to those of [123] in both the clustering output and the determination of the number of clusters. Since the *k*-Axes method converges in very few iterations, our unoptimized Matlab implementation takes a similar amount of time as the rotation method implemented in C. Figure 6.4 shows our results on the 6 datasets we have data for.

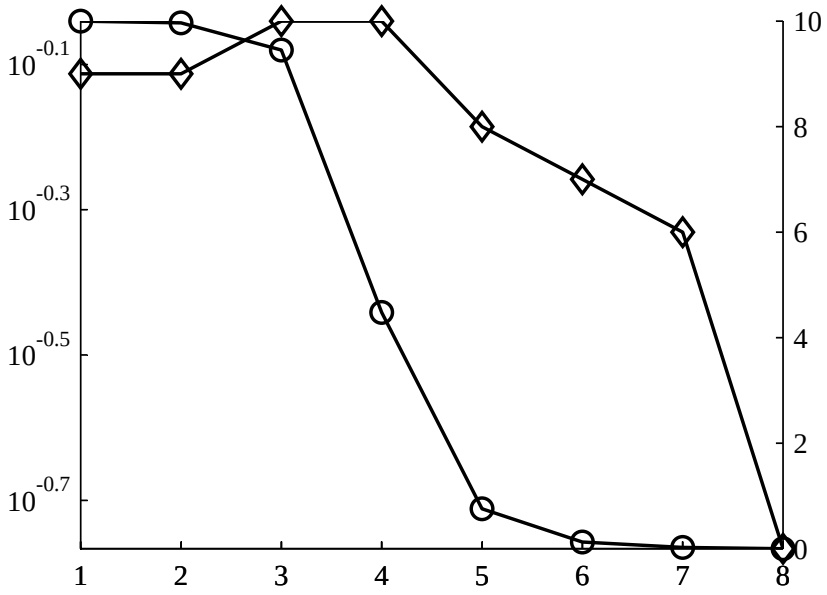


Figure 6.3. Detail about convergence on the Iris data set from Table 6.1. Circles mark the value of the cost function (on a logarithmic scale), while diamonds mark the number of CG iterations. The x axis is the outer iteration (t in Algorithm 1).

6.4.2 Preserving Asymmetry in Clustering

The experiments in this section mirror those in [78]. We construct random affinity matrices for 400 points with 6 clusters each and evaluate the performance of k-Axes with $H(\mathbf{A})$, k-Axes with $H(\mathbf{D}_{\text{out}}^{-1}\mathbf{A})$, normalized cut on $\mathbf{A}\mathbf{A}^{\mathsf{T}}$, and normalized cut on $1/2(\mathbf{A}+\mathbf{A}^{\mathsf{T}})$. Figure 6.5 shows a typical asymmetric affinity matrix used in this experiment. Table 6.2 summarizes the results of this experiment. The k-Axes algorithm uses a deterministic optimization strategy. For all the other algorithms, we set the number of random restarts in k-means to 10 so that the running time of the algorithms is similar and so that the quality of the spectral embedding can be evaluated (i.e., how easy is it to cluster the data after the embedding). We show results for three different levels of noise – 60%, 70%, and 80%, where the percentage indicates the probability that a random directed edge connects a point with any other point. A noise level of 100% would mean that all points are connected with random edges, and 0% means that no random edges are added at all. All edges in the graph receive a weight in the range $(0, 1)$ with uniform probability. Eventually, all methods break down, but K1,

Table 6.1. Inner (CG) and Outer iteration counts for k-Axes on data sets from Section 6.4.3.

Data Set		Iterations	
Name	Size	Outer	CG (Total)
5G	250	5	38
4G	200	2	11
2RG	290	4	27
2R	420	6	26
2S	220	4	7
BC	569	6	28
Iris	150	8	59
Wine	178	5	30

K2, and W1, which are all designed to handle asymmetric affinities, fare better than the two normalized cut methods. Both K2 and W1 perform admirably and only break down at levels of noise exceeding 85%. On several occasions, the k-means step of W1 failed to achieve reasonable clustering with 10 random restarts, hence the small non-zero variation of information at the 70% noise level (2 k-mean failures out of 100 tests).

Table 6.2. Method names and normalizations used in the synthetic affinity experiments.

Method	Matrix	Type
K1	$\mathbf{D}_{\text{out}}^{-1/2}\mathbf{A}\mathbf{D}_{\text{in}}^{-1/2}$	k-Axes
K2	$\mathbf{D}_{\text{out}}^{-1}\mathbf{A}$	k-Axes
N1	$\mathbf{D}_{\text{mul}}^{-1/2}\mathbf{A}\mathbf{A}^{\mathsf{T}}\mathbf{D}_{\text{mul}}^{-1/2}$	n-cut
N2	$1/2\mathbf{D}_{\text{sym}}^{-1/2}(\mathbf{A}+\mathbf{A}^{\mathsf{T}})\mathbf{D}_{\text{sym}}^{-1/2}$	n-cut
W1	$1/2\mathbf{D}_{\text{out}}^{-1/2}(\mathbf{A}+\mathbf{A}^{\mathsf{T}})\mathbf{D}_{\text{out}}^{-1/2}$	w-cut

6.4.3 Using Locally Scaled, Non-Symmetrically Normalized Affinity

While both [83] and [123] use a symmetrically normalized affinity matrix, it is also possible to normalize rows of the affinity only as recommended in von Luxburg [116]. The resulting normalized affinity is asymmetric in general . While this asymmetry is

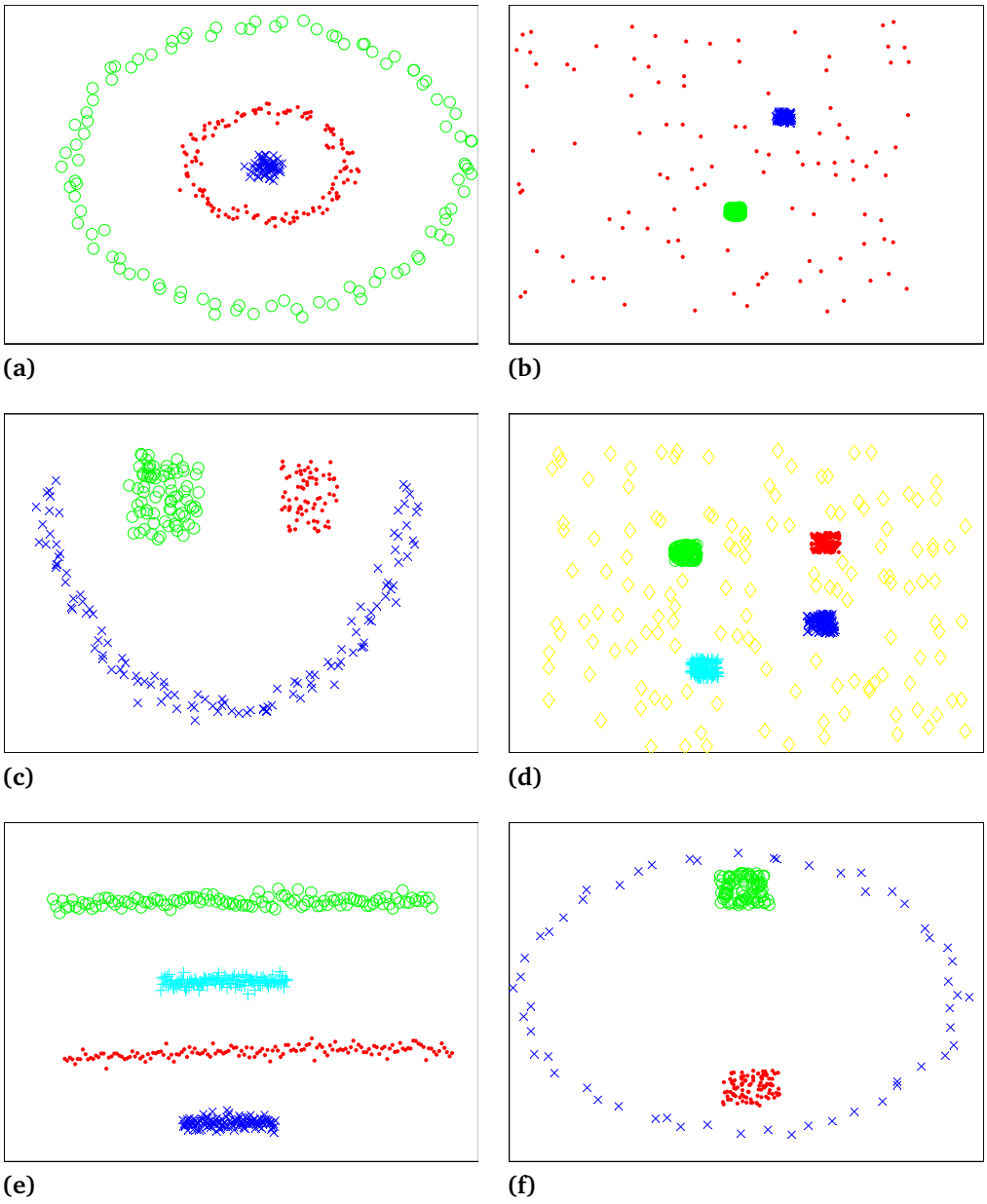
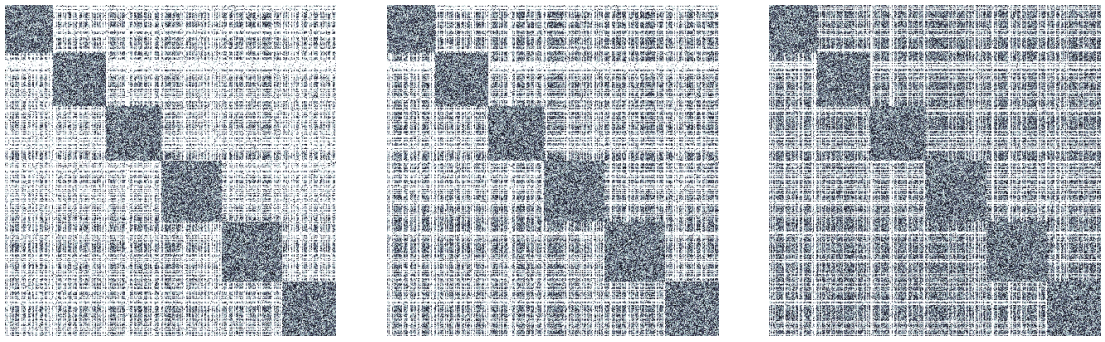


Figure 6.4. Results on data from [123] with automatic cluster number detection.

induced rather than intrinsic to the similarity measure used, it can still be used with our approach. In this section we use data sets from [39]. We use local scaling with the 3rd nearest neighbor for all data sets, and normalize the affinity matrix so that all rows sum to one. Table 6.4 is from [39], and we have added our method as the last column. The "Ng" method refers to the method in [83], "Girolami" refers to the

Table 6.3. Performance of the methods from Table 6.2 on synthetic affinity matrices of varying noise levels. The Variation of Information metric is used.

	60%		70%		80%	
Method	Mean	Std	Mean	Std	Mean	Std
K1	0.000	0.00	0.015	0.15	1.366	0.75
K2	0.000	0.00	0.000	0.00	0.000	0.00
N1	1.317	0.31	1.350	0.09	1.552	0.24
N2	0.106	0.31	0.669	0.65	1.041	0.33
W1	0.000	0.00	0.013	0.13	0.060	0.23



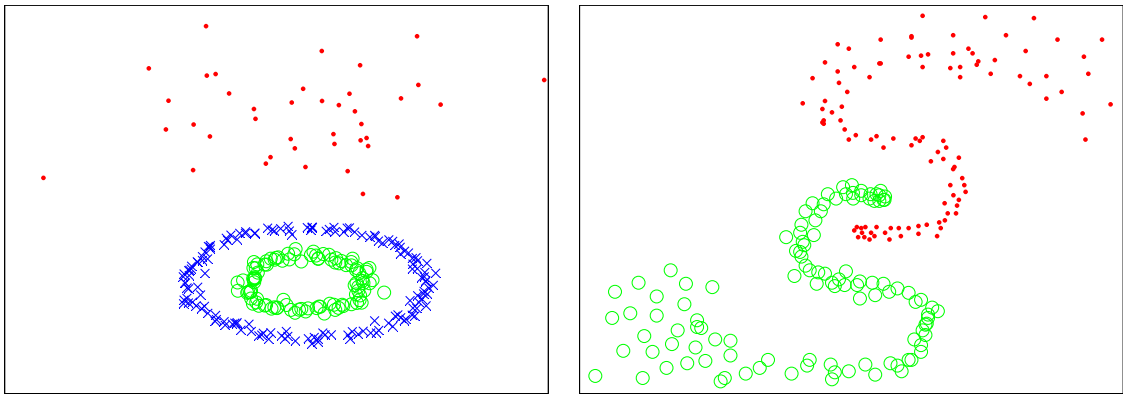
(a) 60% **(b)** 70% **(c)** 80%

Figure 6.5. Sample 400×400 affinity matrices with varying noise levels.

method in [45], while "Fischer-σ" and "Fischer-τ" refer to the two methods discussed in [39]. We label the proposed method "LS(3)–HA", since we use local scaling by the 3rd neighbor and the Hermitian Adjacency with k-Axes.

6.4.4 Pen Trajectories

We show results on the Character Trajectories Data Set from the UCI repository [5]. Figure 6.7 shows representative handwritten characters from each class in the data set. The data consists of the velocity and pen tip force information collected using a graphics tablet from handwritten characters. Twenty single-stroke character classes are represented, and ground-truth data is available. We use the directed DTW distance to compare the pen trajectories. The directed dynamic time warp between two trajectories



(a) Data set 2RG **(b)** Data set 2S

Figure 6.6. The 2RG and 2S data sets from Table 6.4.

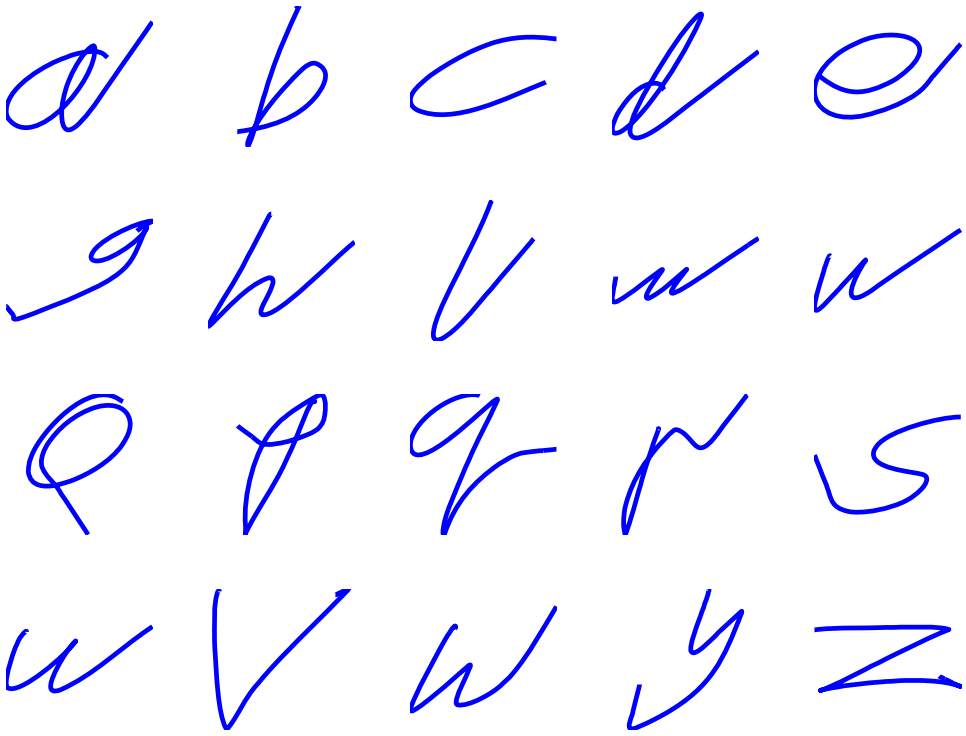


Figure 6.7. The character trajectories data set, with prototypical trajectories for each class. Only single-stroke characters are used.

$\mathbf{x} : [0, T_x] \to \mathbb{R}^d$ and $\mathbf{y} : [0, T_y] \to \mathbb{R}^d$ is defined by:

$$W(\mathbf{x}, \mathbf{y}) = \arg\min_{u,v} \frac{1}{T} \int_0^T \|\mathbf{x}(u(t)) - \mathbf{y}(v(t))\|_2^2 u'(t) \, \mathrm{d}\, t, \tag{6.4.1}$$

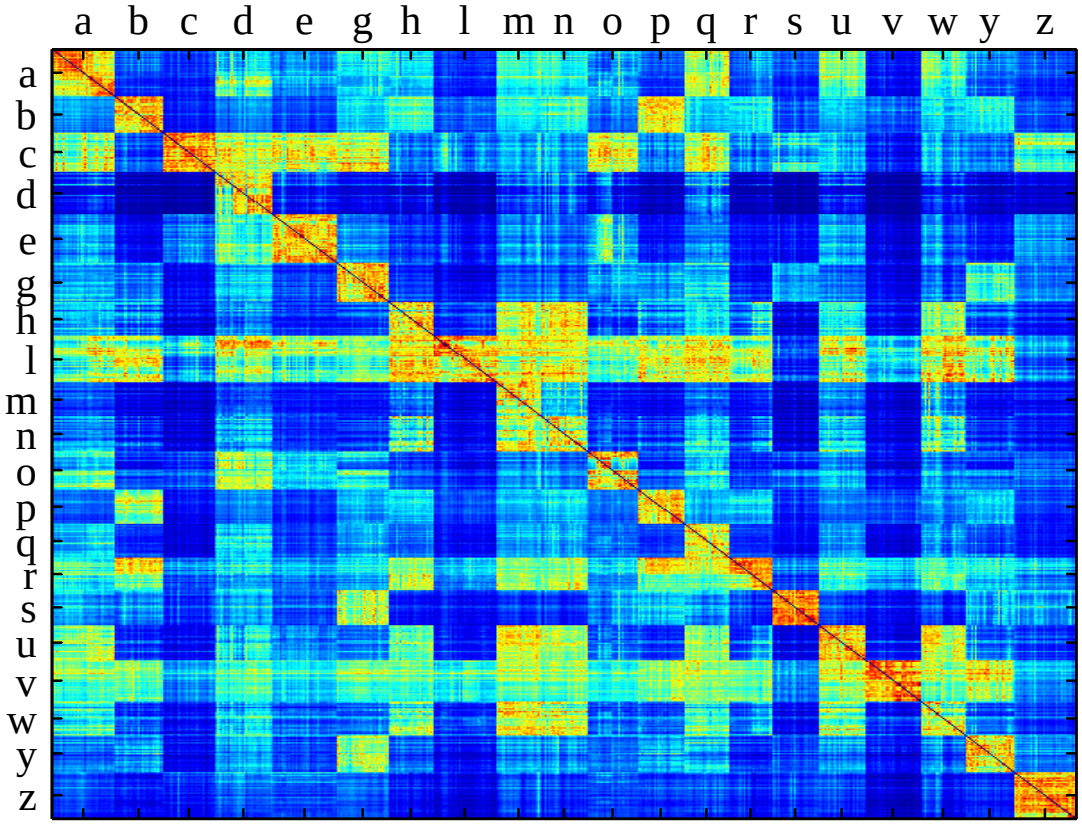


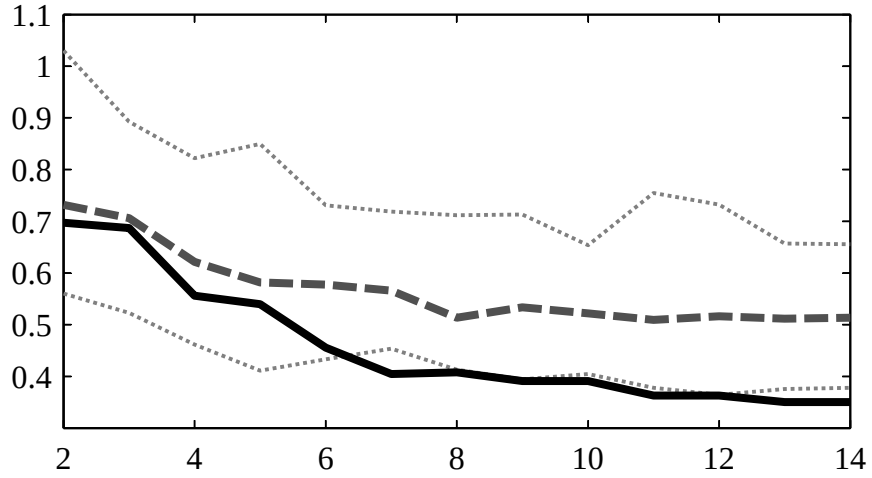
Figure 6.8. Affinities between the character trajectories using the non-symmetric DTW.

where $u : [0, T] \to [0, T_x]$ and $v : [0, T] \to [0, T_y]$ are non-decreasing bijective maps.

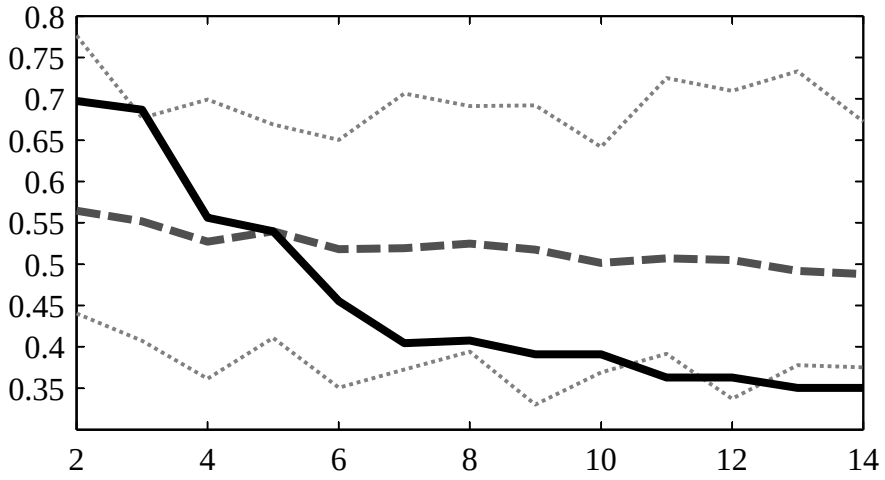
The discretization of (6.4.1) using forward differences for the derivative and the right Riemann sum for the integration leads to the standard discrete version of the distance presented in Section 3.1.1.

6.4.5 Vehicle Trajectories

The unsupervised clustering of vehicle trajectories has many applications. In this section we compare the performance of k-Axes and normalized cuts using trajectory data obtained by automatic tracking of vehicles at an urban traffic intersection. A manual segmentation of the 403 trajectories was performed, and 365 were classified into one of 14 clusters. The remaining 38 trajectories were labeled as outliers. While the outliers are present in the input to the clustering algorithms, they are ignored when computing the performance measures.



(a) Comparison with normalized cuts on $1/2(\mathbf{A}+\mathbf{A}^{\mathsf{T}})$



(b) Comparison with normalized cuts on $\mathbf{A}\mathbf{A}^{\mathsf{T}}$

Figure 6.9. Comparison of *k*-Axes (solid black line) and normalized cuts (dashed lines) on symmetrized affinities. For normalized cuts, we show the best, worst, and average result over 50 trials. The horizontal axis indicates the nearest neighbor used in the local scaling and the vertical is the variation of information.

We use a modified Hausdorff distance in the experiment. The modifications are described in Chapter 5, where we measure is used successfully to cluster vehicle trajectories. However, the method in Chapter 5 required the symmetrization of the kernel (using the less standard componentwise multiplicative approach $\mathbf{A}\cdot\mathbf{A}^{\mathsf{T}}$). We show that the *k*-Axes algorithm can outperform normalized cuts on this task. As

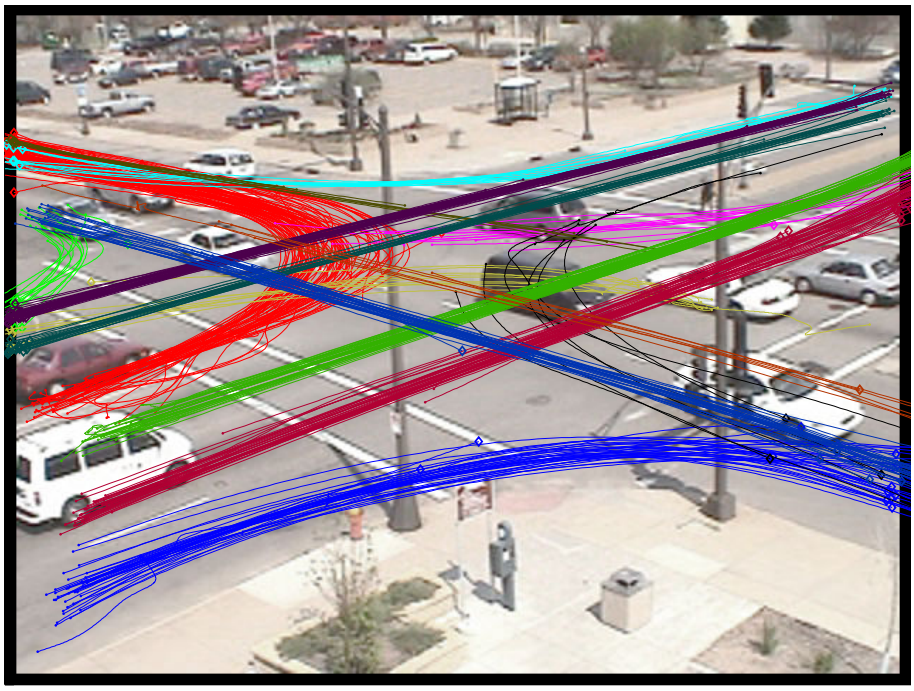


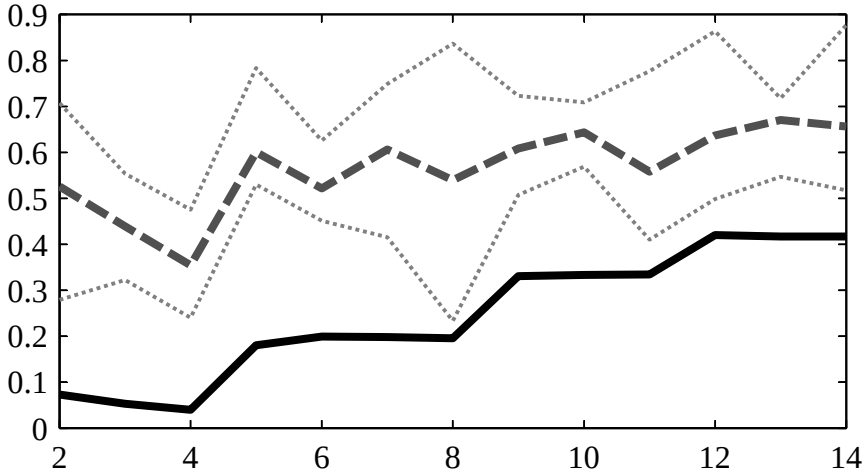
Figure 6.10. The traffic intersection from which the trajectories were obtained, overlaid with the ground-truth data. See Figure 4.16a.

Table 6.5 shows, the direct use of the asymmetric similarity outperforms even the specialized symmetrization approach. Note that we did not restrict the local scale in the asymmetric clustering method to the range 0.4 to 12 feet, which was a piece of prior knowledge that helped the results of Chapter 5.

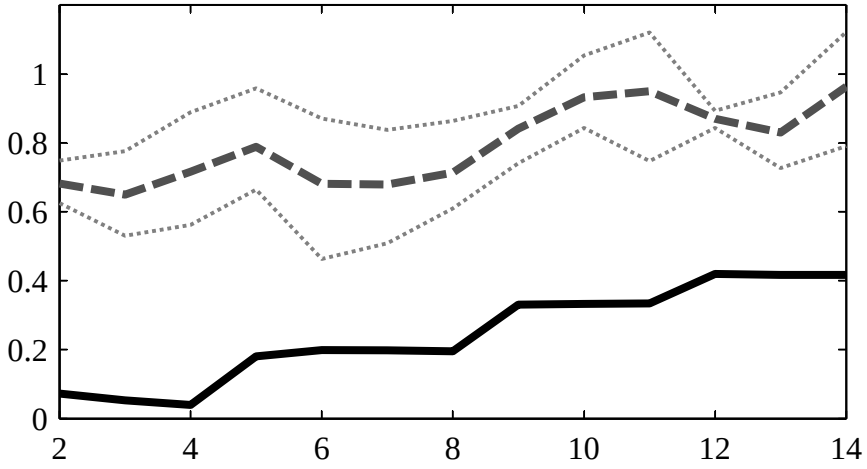
6.4.6 Comparison with Global Alignment Kernel

The global time alignment kernel of Cuturi et al. [34] is the only time alignment kernel with proven positive definiteness. All of the experiments in Chapter 5 use non-metric distances with the Gaussian kernel and local scaling, so the result is predictably not positive definite. We decided to try out the kernel of [34], which can be understood as a variant of DTW where the similarity value is a weighted average over all possible warps between the two sequences.

We faced two challenges in evaluating the kernel — even the C version gracefully provided by the author of [34] is far slower than than even the slowest (DTW) similarity measure identified in Chapter 5. On an identical machine, the DTW kernel in Chapter 5 took 270 seconds, while the global alignment kernel took about 1600 sec-



(a) Comparison with normalized cuts on $1/2(\mathbf{A}+\mathbf{A}^{\mathsf{T}})$



(b) Comparison with normalized cuts on $\mathbf{A}\mathbf{A}^{\mathsf{T}}$

Figure 6.11. Comparison of k-Axes (solid black line) and normalized cuts (dashed lines) on symmetrized affinities. For normalized cuts, we show the best, worst, and average result over 50 trials. The horizontal axis indicates the nearest neighbor used in the local scaling and the vertical is the variation of information.

onds. Since the global alignment kernel has a scale parameter that needs optimization, that running time is greatly magnified.

The second challenge we faced is in fact recognized by [34] — the kernel is extremely diagonally dominant. The proposed solution in [34] is to use the logarithm of the kernel, which results in loss of positive definiteness. To address this, [34] shift

the spectrum of the kernel to make all eignevalues of the empirical kernel matrix positive is not really necessary for clustering (which depends on the eigenvectors). Note that any interpretation advantage provided by the P.D. guarantee is lost by this transformation. We show the performance of the kernel in Figure 6.12. While the kernel outperforms the DTW distance used with the Gaussian kernel on the same data set as listed in Table 6.5 (compare also with Table 5.1), the method is not competitive with any variant of the modified Hausdorff distance but the additively symmetrized one. We used the k-Axes algorithm with the kernel, which produced effectively the same results as the regular normalized cuts.

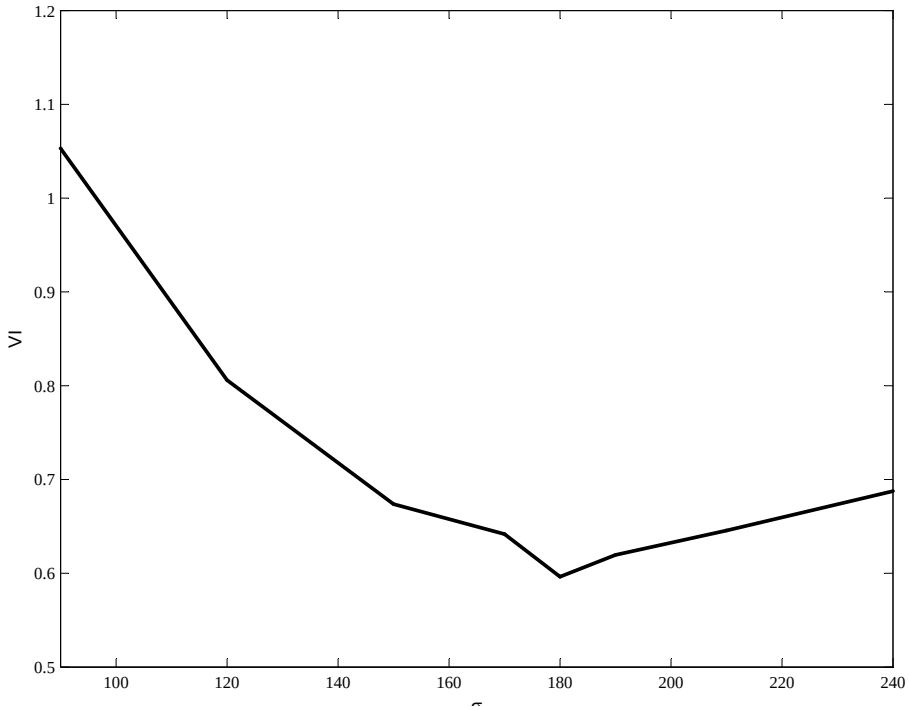


Figure 6.12. Comparison with the time-alignment kernel of Cuturi et al. [34]. The x-axis indicates the scale σ used, and the y-axis shows the Variation of Information distance to the ground truth.

6.5 Discussion and Conclusions

The experiments in this chapter demonstrate that the k-Axes algorithm is competitive with other spectral clustering algorithms when used with a symmetric kernel, and that utilizing the asymmetry of a similarity measure can be quite beneficial.

One interesting observation is that with k-Axes, the scale parameters of the kernels that led to the best performance were always lower than the scales used by the normalized cut algorithms. This can be attributed to the orthogonality constraints of k-Axes, which favor precisely the type of spectral embedding generated by lower scales — where the angle in feature space between any two compact clusters tends to be large. In Brand and Huang [19], it is shown that the angle between points after spectral embedding (the polarization) increases as the rank of the embedding is decreased. We observe a similar pattern for a fixed rank and decreasing scale. Intuitively, at lower scales, the affinity matrix becomes increasingly sparse and more points become orthogonal in the feature space (i.e., their similarity approaches zero).

As the experiments show, it is hard to claim that any specific normalization or symmetrization of the affinity matrix is superior. For normalized cuts, we observe that the additive symmetrization appears slightly better for our experiments. Both the weighted cut of [78] and the Hermitian construction for k-Axes involve the additively symmetrized affinity, which can serve as additional evidence that this symmetrization is to be preferred if no other information is available.

Table 6.4. Comparison of results from [39] with the proposed method.

		Method				
Data Set	Clusters	Girolami	Ng	Fischer-σ	Fischer-τ	LS(3)–HA
2R3D.2	2	68 (202.9 ± 54.6)	4 (7.5 ± 28.9)	6 (8.2 ± 1.9)	93	11 (0.21)
2RG	3	66 (121.7 ± 26.5)	101 (102.2 ± 3.8)	2 (16.8 ± 38.0)	0	0 (0.00)
4G	4	2 (2.6 ± 5.8)	18 (35.1 ± 22.2)	2 (3.0 ± 7.0)	1	1 (0.05)
5G	5	13 (21.2 ± 14.1)	33 (40.1 ± 14.6)	13 (34.5 ± 14.6)	11	8 (0.24)
2S	2	25 (49.9 ± 26.6)	0 (9.3 ± 10.8)	97 (101.8 ± 4.8)	0	0 (0.00)
BC	2	20 (21.5 ± 1.0)	22 (22.8 ± 0.6)	21 (25.1 ± 2.0)	20	95 (0.78)
Iris	3	8 (10.3 ± 2.2)	14 (14.8 ± 3.6)	10 (12.3 ± 5.3)	7	9 (0.39)
Wine	3	3 (3.9 ± 0.9)	3 (3.3 ± 0.5)	12 (18.9 ± 6.6)	65	4 (0.20)

Table 6.5. Summary of results on traffic data clustering

		VI	
Method	Best N	Best	Mean(±Std.)
N1	3	0.531	0.650(±0.110)
N2	4	0.240	0.354(±0.103)
K1	4	0.040	—
Chapter 5	9	0.102	—

Chapter 7

Conclusions

This thesis explored the problem of extracting and clustering trajectories. The work progressed from the immediate concern with tracking vehicles from video cameras, to the desire to collect and summarize statistics about vehicle behavior. The initial work with the Hausdorff distance and spectral clustering was application focused, and only through encountering numerous challenges in using existing methods did we arrive at the modified Hausdorff distance, and ultimately to the problem of asymmetric clustering.

In the end, trajectories have turned out to be a very interesting object of study not for the specific application originally intended, but due to the challenges presented when attempting to use them in increasingly popular kernel methods. While the problems posed by indefinite kernels in the supervised and semisupervised setting have been attacked frequently enough, the idea that a kernel can somehow be asymmetric has not be investigated as deeply in the context of spectral methods.

7.1 Contributions

The contributions of this thesis are spread over Chapter 4, Chapter 5, and Chapter 6. We briefly summarize them here:

1. A novel geometric measurement model for vehicles state estimation and tracking in calibrated traffic scenes was developed. The model provides three desirable capabilities: measurement of vehicle dimensions in real-world units, fusion of

multi-view measurements, and support for partial measurements during static and dynamic occlusions.

2. A vehicle dynamics model in real-world coordinates. The chief advantage of this approach is that the stochastic state estimation parameters such as process noise and measurement noise do not change for different camera views.

3. A constrained Extended Kalman Filter was used to improve the state estimation and to aid the early detection of outliers.

4. The Hausdorff distance was modified to address problems when it is applied directly for trajectory comparison. A correspondence structure similar in spirit to the slack parameter in LCSS and the Sakoe-Chiba band for DTW was introduced.

5. A symmetric affinity matrix was constructed from the directed modified Hausdorff distance using the idea of directed local scaling.

6. An extensive comparison between LCSS, DTW, and the modified Hausdorff distance was performed, both with an agglomerative clustering method and an automated spectral clustering method.

7. An information-preserving spectral embedding for asymmetric similarity measures was proposed based on the construction of a Hermitian kernel.

8. The k-Axes algorithm was proposed to perform clustering with complex embedded coordinates. The algorithm generalizes the rotate-threshold approach for clustering with eigenvectors to complex numbers through an intuitive geometric cost function.

9. The k-Axes algorithm was validated on both symmetric and asymmetric clustering tasks and found to be competitive in both results and computation time.

10. A full Riemannian conjugate gradient optimization was developed for the inner iteration of the k-Axes algorithm. The gradient and periodicity order of the k-Axes cost function were derived.

7.2 Future Work

The tracking method presented in Chapter 4 can only be improved incrementally, by using better motion segmentation methods, shadow detection, and other low-level vision improvements. The incorporation of prior knowledge about a traffic scene obtained using the methods in this thesis would be the one non-trivial improvement for the tracking, which can reduce or eliminate problems with target initialization, target association and aid anomaly detection.

The comparison of clustering methods in Chapter 5 focused on the most common methods, as used for trajectory clustering in the literature to this point. The modified Hausdorff distance borrowed ideas from LCSS and DTW to improve performance, and it is likely that some of the lessons learned in these experiments can be applied to modifications of DTW and LCSS.

The Hermitian construction of Chapter 6 should apply in supervised and semi-supervised methods alike, and one possibility for future work is to see which kernel methods can work with complex kernels easily — while common definitions for kernels often work in $\mathbb{C}$, most uses of kernels to date have only used real-valued kernels.

Bibliography

[1] T. Abrudan, J. Eriksson, and V. Koivunen. Conjugate gradient algorithm for optimization under unitary matrix constraint. *Signal Processing*, 89(9):1704–1714, Sept. 2009.

[2] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 375–381, June 2003.

[3] G. Antonini and J. P. Thiran. Trajectories clustering in ICA space: an application to automatic counting of pedestrians in video sequences. In *Proc. Advanced Concepts for Intelligent Vision Systems*, Aug. 2004.

[4] A. Apostolico. String editing and longest common subsequences. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, pages 361–398. Springer-Verlag, New Your, NY, 1997.

[5] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`.

[6] S. Atev, O. Masoud, and N. P. Papanikolopoulos. Practical mixtures of gaussians with brightness monitoring. In *Proc. IEEE Conf. Intelligent Transportation Systems*, pages 423–428, Oct. 2004.

[7] S. Atev, H. Arumugam, O. Masoud, R. Janardan, and N. P. Papanikolopoulos. A vision-based approach to collision prediction at traffic intersections. *IEEE Trans. Intelligent Transportation Systems*, 6(4):416–423, Dec. 2005.

[8] S. Atev, O. Masoud, and N. Papanikolopoulos. Learning traffic patterns at intersections by spectral clustering of motion trajectories. In *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, pages 4851–4856, Oct. 2006.

[9] S. Atev, G. Miller, and N. Papanikolopoulos. Clustering of vehicle trajectories. *IEEE Trans. Intelligent Transportation Systems*, to appear, 2010.

[10] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proc. Int'l Conf. Machine Learning*, pages 33–40, Aug. 2005.

[11] J. Badenas, J. M. Sanchiz, and F. Pla. Using temporal integration for tracking regions in traffic monitoring sequences. In *Proc. Int'l Conf. Pattern Recognition*, volume 3, pages 1125–1128, Sept. 2000.

[12] F. I. Bashir, A. A. Khokhar, and D. Schoenfeld. Segmented trajectory based indexing and retrieval of video data. In *Proc. IEEE Int'l Conf. Image Processing*, volume 2, pages 623–626, Sept. 2003.

[13] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[14] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the Nyström extension. In *Proc. European Conf. Computer Vision*, pages 531–542, May 2002.

[15] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. Technical Report 1238, Dept. Informatics and Operational Research, University of Montreal, Montreal, Canada, July 2003.

[16] A. Bevilacqua. Effective object segmentation in a traffic monitoring application. In *3rd IAPR Indian Conf. Computer Vision, Graphics and Image Processing*, Dec. 2002.

[17] M. Brand. Incremental singular value decomposition of uncertain data with missing values. Technical Report 2002–24, Mitsubishi Electric Research Laboratory, Cambridge, MA, May 2002.

[18] M. Brand. Fast online SVD revisions for lightweight recommender systems. In *Proc. SIAM Int'l Conf. Data Mining*, pages 37–46, May 2003.

[19] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, Jan. 2003.

[20] D. Buzan, S. Sclaroff, and G. Kollios. Extraction and clustering of motion trajectories in video. In *Proc. Int'l Conf. Pattern Recognition*, volume 2, pages 521–524, Aug. 2004.

[21] G. Charpiat, O. D. Faugeras, and R. Keriven. Approximations of shape metrics and application to shape warping and empirical shape statistics. *Foundations of Computational Mathematics*, 5(1):1–58, Feb. 2005.

[22] G. Charpiat, O. D. Faugeras, and R. Keriven. Shape statistics for image segmentation with prior. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–6, June 2007.

[23] L. Chen, M. T. Özsu, and V. Oria. Symbolic representation and retrieval of moving object trajectories. In *ACM SIGMM Int'l. Wkshp. Multimedia Information Retrieval*, pages 227–234, Oct. 2004.

[24] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pages 491–502, 2005.

[25] T.-J. Chin and D. Suter. Incremental kernel PCA for efficient non-linear feature extraction. In *Proc. British Machine Vision Conference*, volume III, pages 939–948, Sept. 2006.

[26] T.-J. Chin, K. Schindler, and D. Suter. Incremental kernel SVD for face recognition with image sets. In *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pages 461–466, Apr. 2006.

[27] D. Chudova, S. Gaffney, E. Mjolsness, and P. Smyth. Translation-invariant mixture models for curve clustering. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 79–88, Aug. 2003.

[28] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *J. Transportation Research C: Emerging Technologies*, 6(4):271–288, Aug. 1998.

[29] C. Cortes, P. Haffner, and M. Mohri. Positive definite rational kernels. In *Proc. Conf. Learning Theory*, pages 41–56, Aug. 2003.

[30] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *J. Machine Learning Research*, 5:1035–1062, Jan. 2004.

[31] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall / CRC, Boca Raton, FL, second edition, 2001.

[32] D. Cremers, T. Kohlberger, and C. Schnörr. Shape statistics in kernel space for variational image segmentation. *Pattern Recognition*, 36(9):1929–1943, Sept. 2003.

[33] M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *J. Machine Learning Research*, 6:1169–1198, 2005.

[34] M. Cuturi, J.-P. Vert, Ø. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *IEEE Int'l Conf. Acoustics, Speech and Signal Processing*, volume 2, pages 413–416, Apr. 2007.

[35] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report 04–25, Dept. Computer Science, University of Texas at Austin, Austin, TX, Feb. 2005.

[36] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. Technical Report 1319, Dept. Computer Science, Yale University, New Haven, CT, Apr. 2005.

[37] M.-P. Dubuisson and A. K. Jain. A modified Hausdorff distance for object matching. In *Proc. Int'l Conf. Pattern Recognition*, volume 1, pages 566–568, Oct. 1994.

[38] F. Duchene, C. Garbay, and V. Rialle. Similarity measure for heterogeneous multivariate time-series. In *Proc. European Signal Processing Conference*, pages 1605–1608, Sept. 2004.

[39] I. Fischer and J. Poland. Amplifying the block matrix structure for spectral clustering. Technical Report 03–05, IDSIA, Manno-Lugano, Switzerland, Jan. 2005.

[40] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2): 214–225, Feb. 2004.

[41] Z. Fu, W. Hu, and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Proc. IEEE Int'l Conf. Image Processing*, volume 2, pages 602–605, Sept. 2005.

[42] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 63–72, Aug. 1999.

[43] T. Gärtner, K. Driessens, and J. Ramon. Graph kernels and Gaussian processes for relational reinforcement learning. In *Proc. Int'l Conf. Inductive Logic Programming*, pages 146–163, Sept. 2003.

[44] M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *J. Machine Learning Research*, 2:299–312, Dec. 2001.

[45] M. Girolami. Mercer kernel-based clustering in feature space. *IEEE Trans. Neural Networks*, 13(3):780–784, May 2002.

[46] G. H. Golub and C. F. V. Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, third edition, 1996.

[47] C. Gruber, T. Gruber, and B. Sick. Online signature verification with new time series kernels for support vector machines. In *Proc. Int'l Conf. Advances in Biometrics*, pages 500–508, Jan. 2006.

[48] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. Technical Report 110, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, July 2003.

[49] D. Haussler. Convolution kernels on discrete structures. Technical Report CRL–99–10, University of California at Santa Cruz, Santa Cruz, CA 95064, July 1999.

[50] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. Association for Computing Machinery*, 24(4):664–675, Oct. 1977.

[51] B. Hoser and A. Geyer-Schulz. Eigenspectral analysis of Hermitian adjacency matrices for the analysis of group substructures. *J. Mathematical Sociology*, 29 (4):265–294, 2005.

[52] B. Hoser and J. Schröder. Automatic determination of clusters. In K.-H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006*, pages 439–444. Springer, Berlin, Germany, 2007.

[53] J. W. Hsieh, S. H. Yu, Y. S. Chen, and W. F. Hu. Automatic traffic surveillance system for vehicle tracking and classification. *IEEE Trans. Intelligent Transportation Systems*, 7(2), June 2006.

[54] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank. Traffic accident prediction using 3D model-based vehicle tracking. *IEEE Trans. Vehicular technology*, 53(3): 677–694, May 2004.

[55] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350–353, May 1977.

[56] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sept. 1993.

[57] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 23:67–72, Feb. 1975.

[58] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Int'l Conf. Intelligent Systems for Molecular Biology*, pages 149–158, Aug. 1999.

[59] T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, Jan. 1999.

[60] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 487–493, Cambridge, MA, 1999. MIT Press.

[61] R. Jenssen, D. Erdogmus, J. Principe, and T. Eltoft. The Laplacian PDF distance: A cost function for clustering in a kernel feature space. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 625–632, Cambridge, MA, 2005. MIT Press.

[62] R. Jenssen, T. Eltoft, M. Girolami, and D. Erdogmus. Kernel maximum entropy data transformation and an enhanced spectral clustering algorithm. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 633–640, Cambridge, MA, 2007. MIT Press.

[63] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proc. IEEE Int'l Conf. Data Mining*, pages 273–280, Nov. 2001.

[64] V. Kastrinaki, M. E. Zervakis, and K. Kalaitzakis. A survey of video processing techniques for traffic applications. *Image and Vision Computing*, 21(4):359–381, Apr. 2003.

[65] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proc. SIAM Int'l Conf. Data Mining*, Apr. 2001.

[66] K. I. Kim, M. O. Franz, and B. Schölkopf. Kernel Hebbian algorithm for iterative kernel principal component analysis. Technical Report 109, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, June 2003.

[67] K. I. Kim, M. O. Franz, and B. Schölkopf. Kernel Hebbian algorithm for single-frame super-resolution. In *Proc. ECCV Wkshp. Statistical Learning in Computer Vision*, pages 135–149, May 2004.

[68] Z. Kim and J. Malik. Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In *Proc. IEEE Int'l Conf. Computer Vision*, volume 1, pages 524–531, Oct. 2003.

[69] D. Koller, J. W. Weber, T. T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel. Towards robust automatic traffic scene analysis in real-time. In *Proc. Int'l Conf. Pattern Recognition*, volume 1, pages 126–131, Oct. 1994.

[70] D. Koller, J. W. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. California PATH Working Paper 94–1, Dept. Electrical Engineering and Cmputer Science, University of California at Berkeley, Berkeley, CA, 1994.

[71] J. B. Kruskal and M. Liberman. The symmetric time-warping problem: From continuous to discrete. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 125–161. Addison Wesley, Reading, MA, 1983.

[72] J. Lin, M. Vlachos, E. J. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *Int'l Conf. Extending Database Technology*, pages 106–122, Mar. 2004.

[73] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Machine Learning Research*, 2:419–444, Feb. 2002.

[74] J. Lou, H. Yang, W. Hu, and T. Tan. Visual vehicle tracking using an improved EKF. In *Proc. Asian Conf. Computer Vision*, pages 296–301, Jan. 2002.

[75] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22(2):143–155, Feb. 2004.

[76] O. Masoud and N. Papanikolopoulos. Calibrating traffic camera systems based on geometric primitives. *J. Transportation Research C: Emerging Technologies*, 15(6):361–379, Dec. 2007.

[77] M. Meilă. Comparing clusterings – an information based distance. *J. Multivariate Analysis*, 98(5):837–895, May 2005.

[78] M. Meilă and W. Pentney. Clustering by weighted cuts in directed graphs. In *Proc. SIAM Int'l Conf. Data Mining*, Apr. 2007.

[79] M. Meilă and J. Shi. A random walks view of spectral segmentation. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, Jan. 2001.

[80] N. Meratnia and R. A. de By. Aggregation and comparison of trajectories. In *Proc. ACM Int'l Symp. Advances in Geographic Information Systems*, pages 49–54, Nov. 2002.

[81] B. Morris and M. Trivedi. Robust classification and tracking of vehicles in traffic video streams. In *Proc. IEEE Conf. Intelligent Transportation Systems*, pages 1078–1083, Sept. 2006.

[82] H. Narayanan, M. Belkin, and P. Niyogi. On the relation between low density separation, spectral clustering and graph cuts. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1417–1424, Cambridge, MA, 2007. MIT Press.

[83] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856, Cambridge, MA, 2002. MIT Press.

[84] T. Oates, L. Firoiu, and P. R. Cohen. Clustering time series with hidden Markov models and dynamic time warping. In *Proc. IJCAI–99 Wksp. Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pages 17–21, Aug. 1999.

[85] M. Ouimet and Y. Bengio. Greedy spectral clustering. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, pages 253–260, Jan. 2005.

[86] C. C. Pang, W. L. Lam, and H. C. Yung. A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images. *IEEE Trans. Intelligent Transportation Systems*, 8(3):441–459, Sept. 2007.

[87] W. Pentney and M. Meilă. Spectral clustering of biological sequence data. In *Proc. Nat'l Conf. Artificial Intelligence (AAAI'05)*, volume 2, pages 845–850, July 2005.

[88] E. Pękalska, P. Packlík, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *J. Machine Learning Research*, 2:175–211, Mar. 2002.

[89] E. Pękalska, R. P. W. Duin, and P. Packlík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, Feb. 2006.

[90] J. C. Platt. FastMap, MetricMap, and Landmark MDS are all Nyström algorithms. In *Int'l. Wksp. Artificial Intelligence and Statistics*, pages 261–268, Jan. 2005.

[91] F. M. Porikli. Learning object trajectory patterns by spectral clustering. In *Proc. IEEE Int'l Conf. Multimedia and Expo*, volume 2, pages 1171–1174, June 2004.

[92] W. Power and J. Schoonees. Understanding background mixture models for foreground segmentation. In *Proc. Imaging and Vision Computing New Zealand*, pages 267–271, Nov. 2002.

[93] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[94] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, Dec. 2000.

[95] E. V. Ruiz, F. C. Nolla, and H. R. Segovia. Is the DTW "distance" really a metric? an algorithm reducing the number of DTW comparisons in isolated word recognition. *Speech Communication*, 4(4):333–334, Dec. 1985.

[96] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 26:43–49, Feb. 1978.

[97] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyound*. MIT Press, Cambridge, MA, 2002.

[98] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, Dec. 1996.

[99] N. N. Schraudolph, S. Günter, and S. Vishwanathan. Fast iterative kernel PCA. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1225–1232, Cambridge, MA, 2007. MIT Press.

[100] J. Shawe-Taylor and N. Christiani. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, United Kingdom, 2004.

[101] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug. 2000.

[102] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 921–928, Cambridge, MA, 2002. MIT Press.

[103] D. Simon and D. L. Simon. Kalman filtering with inequality constraints for turbofan engine health estimation. *IEE Proceedings – Control Theory and Applications*, 153(3):371–378, May 2006.

[104] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 246–252, June 1999.

[105] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):747–757, Aug. 2000.

[106] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, Dec. 2000.

[107] J. van den Heuvel and S. Pejić. Using Laplacian eigenvalues and eigenvectors in the analysis of frequency assignment problems. CDAM Research Report Series LSE–CDAM–2000–20, Dept. Mathematics, London School of Economics, London, U.K., Dec. 2000.

[108] D. Verma and M. Meilă. A comparison of spectral clustering algorithms. Technical Report UW–CSE–03–05–01, Dept. Computer Sceience and Engineering, University of Washington, Seattle, WA, 2003.

[109] J.-P. Vert. A tree kernel to analyze phylogenetic profiles. *Bioinformatics*, 18 (Supplement 1):S276 – S284, July 2002.

[110] J.-P. Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel methods in Computational Biology*, pages 131–154. MIT Press, Cambridge, MA, 2004.

[111] S. Vishwanathan and A. J. Smola. Fast kernels for string and tree matching. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 569–576, Cambridge, MA, 2003. MIT Press.

[112] S. V. N. Vishwanathan, K. M. Borgwardt, R. I. Kondor, and N. N. Schraudolph. Graph kernels. *CoRR*, abs/0807.0093, 2008.

[113] M. Vlachos, G. Kollios, and D. Gunopoulos. Discovering similar multidimensional trajectories. In *Proc. 18th Int'l Conf. Data Engineering*, pages 673–684, Feb. 2002.

[114] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series with support for multiple distance measures. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 216–225, Aug. 2003.

[115] M. Vlachos, D. Gunopulos, and G. Das. Rotation invariant distance measures for trajectories. In *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pages 707–712, Aug. 2004.

[116] U. von Luxburg. A tutorial on spectral clustering. Technical Report 149, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, Aug. 2006.

[117] Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 975–982, Sept. 1999.

[118] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report 95–041, Univ. North Carolina, Chapel Hill, NC, May 2006.

[119] G. Wu, E. Y. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. Technical report, Dept. Electrical & Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, June 2005.

[120] T. Xiang and S. Gong. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41(3):1012–1029, Mar. 2008.

[121] Y. Xiong and D.-Y. Yeung. Mixtures of ARMA models for model-based time series clustering. In *Proc. IEEE Int'l Conf. Data Mining*, pages 717–720, Dec. 2002.

[122] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proc. IEEE Int'l Conf. Computer Vision*, volume 1, pages 313–319, Oct. 2003.

[123] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 1601–1608, Cambridge, MA, 2005. MIT Press.

[124] K. Zhang and J. T. Kwok. Block-quantized kernel matrix for fast spectral embedding. In W. W. Cohen and A. Moore, editors, *Proc. Int'l Conf. Machine Learning*, volume 148 of *ACM International Conference Proceeding Series*, pages 1097–1104. ACM, June 2006.

[125] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nyström low-rank approximation and error analysis. In A. McCallum and S. Roweis, editors, *Proc. Int'l Conf. Machine Learning*, pages 1232–1239. Omnipress, 2008.

[126] Z. Zhang, K. Huang, and T. Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proc. Int'l Conf. Pattern Recognition*, pages 1135–1138, Aug. 2006.

Appendix: Matlab Implementation of the k-Axes Algorithm

This appendix includes the neccessary code to implement the k-Axes algorithm in Matlab. The code is included since unlike normalized cuts, it is not widely available. We would like to highlight that despite the conceptual complexity in optimization with unitary constraints, the actual implementation is quite reasonable.

The two functions in Code Listing 2 implement the outer iteration of the k-Axes algorithm. The gradient of the k-Axes cost function presented in Chapter 6 is used without modification. Code Listing 3 implements the inner iteration of the method by doing applying the conjugate gradient method on the manifold of unitary matrices. The line search routine in Code Listing 4 exploits the Lie structure of unitary matrices.

Code Listing 2 Simplified MATLAB® code for the k-Axes algorithm.

```
function idx= kaxes(A, C)
    maxOuterIters= 50;
    minImprovement= 1e-6;
    n= size(A, 1);
    K= 0.5* ((A+ A')+ sqrt(-1)* (A- A'));
    [V, L]= eig(K);
    [L, perm]= sort(diag(L), 'descend');
    V= V(:, perm(1:C));
    Y= diag(sqrt(L(1:C)))* V.';
    err= Inf;
    U= eye(C);
    psigma= max(min(abs(Y),[],1));
    [W, psigma]= kaxes_weights(U, Y, psigma);
    for iter= 1:maxOuterIters
        [U, iters]= unitary_optim(U, @gradient, Y, W);
        [W, sigma]= kaxes_weights(U, Y, psigma);
        if sigma> (1- minImprovement)* psigma && iter> 1
            break;
        end
        psigma= sigma;
    end
    [dummy, idx]= max(W, [], 1);

function [W,sigma]= kaxes_weights(U,Y,sigma)
    [k, n]= size(Y);
    D= zeros(k, n);
    for i= 1:k
        P= eye(k)- U(:, i)* U(:, i)';
        Q= conj(Y).* (P* Y);
        D(i, :)= abs(real(sum(Q, 1)));
    end
    M= repmat(min(D, [], 1), [k, 1])- D;
    W= exp(M/ sigma);
    W= W./ repmat(sum(W, 1), [k, 1]);
    sigma= sum(sum(D.* W))/ n;
```

Code Listing 3 Simplified MATLAB® code for optimization under unitary constraits.

```
function [U, iters]= unitary_optim(U, gradf, varargin)
    n= size(U, 1);
    tol= 1e-7;
    maxiter= min(100, ceil(2* log(n)* n^2));
    EG= gradf(U, varargin{:});
    G= EG* U'- U* EG';
    H= G;
    gg= 0.5* real(trace(G'* G));
    iters= 0;
    while iters< maxiter && gg> tol
        iters= iters+ 1;
        mu= line_search(U, H, gradf, varargin{:});
        if mu< 0 && H== G
            return;
        end
        U= expm(-mu* H)* U;
        EG= gradf(U, varargin{:});
        Gn= EG* U'- U* EG';
        gk= real(trace((Gn- G)'* Gn))/ real(trace(G'* G));
        G= Gn;
        gg= 0.5* real(trace(G'* G));
        H= G+ gk* H;
        if mod(iters, n)== 0 || real(trace(H'* G))< 0
            H= G;
        end
    end
```

Code Listing 4 Simplified MATLAB® code for the line search with polynomial approximation for a function of periodicity order 2.

```
function step= line_search(W, H, grad, varargin)
    q= 2;
    P= 3;
    wmax= max(abs(eig(H)));
    Tmu= 2* pi()/ (q* wmax);
    R1= expm(-Tmu/ P* H);
    gmu= zeros(P, 1);
    R= eye(size(R1));
    for i= 1:P+1
        tg= grad(R* W, varargin{:});
        gmu(i)= 2* real(trace(tG* W'* R'* H'));
        R= R1* R;
    end
    J= gmu(2:end)- gmu(1);
    M= zeros(P);
    for i= 1:P
        M(i, :)= (i* Tmu/ P).^ [1:P];
    end
    a= M\ J;
    z= roots([a(P:-1:1)', gmu(1)]);
    zreal= z(abs(imag(z))< P* eps());
    if isempty(zreal)
        step= -1;
        return
    end
    zpos= zreal(zreal> 0);
    if isempty(zpos)
        step= -1;
        return;
    end
    step= min(zpos);
```