VISUAL ANALYTICS TECHNIQUES FOR EXPLORATION OF SPATIOTEMPORAL DATA

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
POLYTECHNIC SCHOOL OF ENGINEERING

by

Nivan Ferreira

September 2015

VISUAL ANALYTICS TECHNIQUES FOR EXPLORATION OF SPATIOTEMPORAL DATA

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
POLYTECHNIC SCHOOL OF ENGINEERING

by

Nivan Ferreira

September 2015

Approved:

_____

Department Head Signature

_____

Date

Copy No. _____

University ID#: N17069336

Approved by the Guidance Committee:

Major: Computer Science

Claudio T. Silva
Professor of
Computer Science and Engineering

Juliana Freire
Professor of
Computer Science and Engineering

Enrico Bertini
Assistant Professor of
Computer Science and Engineering

Jean-Daniel Fekete
Senior Research Scientist

Vita

Nivan Ferreira was born in Recife, Brazil in July 1988. He holds a Bachelor's degree in Computer Science and a Master's degree in Mathematics both from the Universidade Federal de Pernambuco in Brazil. In September 2010 he started his graduate studies towards his Ph.D. under the suppervision of professor Claudio Silva, first at the University of Utah and later at the Polytechnic Institute of New York University. During this time he has done research on data visualization focusing on interactive analysis of large datasets and the use of machine learning techniques in visualization systems. He is the recipient of 2013 Deborah Rosenthal, MD Award for outstanding performance on the PhD qualifying examination and of the 2015 Pearl Brownstein Doctoral Research Award, given to PhD students in the Department of Computer Science and Engineering whose doctoral research shows the greatest promise. His paper on trajectory clustering was awarded an Honorable Mention at EuroVis 2013.

Acknowledgements

First and foremost, I would like to thank my parents Nivan and Fátima. Without their endless encouragement and support this dissertation would not be possible. I also would like to thank professors Ramón Mendoza, Francisco Brito, and Sóstenes Lins from Universidade Federal de Pernambuco and professor Antonio Sá Barreto from Purdue University for their guidance throughout my first steps in my academic life and also for providing me with encouragement to come to United States to pursue my PhD.

Very special thanks go to my advisor, Cláudio Silva, for the opportunity to join the VGC team and also for his guidance over the course of my PhD studies. I also would like to thank the other members of my committee, Juliana Freire, Enrico Bertini, and Jean-Daniel Fekete, for their valuable feedback. During this time, I have had the opportunity to work with a number of talented collaborators, to whom I am grateful for their contributions that have aided this work and in my training as a researcher. Among my collaborators, I'd like to especially thank Juliana Freire, Lauro Lins, Carlos Scheidegger, Huy Vo, Jorge Poco, Harish Doraiswamy, and Marcos Lage.

Finally, I also would like to thank all the (former and current) members of the VGC team that made the our lab a stimulating research environment and the moments outside the lab full of joy.

Nivan Ferreira
September 2015

To my parents, with affection.

ABSTRACT

VISUAL ANALYTICS TECHNIQUES FOR EXPLORATION OF SPATIOTEMPORAL DATA

by

Nivan Ferreira

Advisor: Prof. Claudio Silva, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science)

September 2015

Spatial and temporal interactions are central and fundamental in pretty much all activities in our world and society. Everyday, people and goods travel around the world at different speeds and scales; migratory animals engage in long-distance travels that demonstrate the biological integration around the globe; weather phenomena, like typhoons and hurricanes, form and move around the Earth and may have large social-economic impact. In all these examples, proper understanding of the underlying phenomena can produce insights with the potential to shape the future development in those domains.

The rapid development of acquisition technology and the popularization of GPS enabled mobile devices has resulted in spatiotemporal data being produced at massive rates. These create opportunities for data-driven analysis that can highly influence decision making in a diverse set of domains. In order to take advantage of

all these data and realize their potential, it is crucial to be able to extract knowledge from them.

Interactive visualization systems are acknowledged to be important tools in this scenario: they leverage the human cognitive system and the power of interactive graphics to enable quick hypothesis testing and exploration. However, the volume and inherent complexity of spatiotemporal data makes designing such systems a difficult problem. In fact, such complex data collections pose challenges in both managing the data for interactive exploration as well as in designing visual metaphors that enable effective data exploration. Also, such visual metaphors are limited by constraints imposed by the display and data dimensions, often resulting in extremely cluttered visualizations that are hard to interpret. While, filtering and aggregation strategies are often applied to eliminate clutter, they might hide interesting patterns. Therefore, purely visual/interaction methods need to be complemented with techniques that help in the process of pattern discovery.

This dissertation presents novel visual analytics contributions for the analysis of spatiotemporal data to attack the challenges aforementioned. Visual analytics combine interactive visualization with efficient pattern mining techniques to enable analysts to explore large amounts of complex data.

The first contribution is the design of the TaxiVis visual exploration system. This system couples together a novel visual query model with an efficient custom-built data layer. These two components enable easy query composition via visual methods as well as interactive query response times. TaxiVis also makes use of coordinated views and rendering strategies to generate informative visual summaries for query results even when those are large.

The remaining of the contributions in this thesis consists of two pattern mining techniques that help in the navigation through the data via pattern discovery. These two techniques have the goal of enhancing the analytical power of tools such as TaxiVis. Furthermore, these techniques have in common the of use concepts and techniques widely applied in scientific visualization and computer graphics. This approach allows us to have novel perspectives on the problems of finding patterns in spatiotemporal data that, to the best of our knowledge, have not been considered in the machine learning and data mining fields. The first technique consists of a topology-based technique whose main objective is to help users to find the “needle

in the hay stack", i.e., guide users towards *interesting* slices (spatiotemporal regions) of the data. We call this process *event guided exploration*. The overall idea behind this technique is to treat topological features of time-varying scalar functions derived from spatiotemporal data as treated as *events*. Via visual exploration of the collection of extreme points extracted over time, important events of the data can be found with relatively a small amount of work by the user.

The second pattern mining technique consists of a novel model based clustering technique designed for *trajectory datasets*. This technique, called *Vector field k-means*, models trajectories as streamlines of vector fields. One important feature of this modeling strategy is that it tries to avoid overlapping trajectories to have discrepant directions at their intersections. Clustering is achieved by using the spatial component of trajectories to fit a collection of vector fields to the given trajectories. This technique achieves richness and expressivity of features, simplicity of implementation and analysis, and computational efficiency. Furthermore, the obtained vector fields serve as a visual summary of the movement patterns in each cluster. Finally, Vector field k-means can be naturally generalized to also consider trajectories with attributes. This is achieved by using a different modeling strategy based on *scalar fields*, which we call *Attribute field k-means.*

Contents

Vita v
Acknowledgements vi
Abstract viii
List of Figures xxi
List of Tables xxii

1 Introduction **1**
1.1 Motivation 1
1.2 Spatiotemporal Data Types 2
1.3 Visual Analysis of Spatiotemporal Data 3
1.4 Thesis Contributions 6
1.5 Thesis Outline 8

2 The TaxiVis System for Exploration of Taxi Trip Records **9**
2.1 Related Work 13
2.2 Data and Design Requirements 15
2.3 Visual Query Model 18
2.3.1 Defining and Composing Queries 18
2.3.2 Exploring Query Results 22
2.3.3 Query Expressiveness 22
2.4 The System 23
2.4.1 User Interface Components 24
2.4.2 Visual Query Specification 25
2.4.3 Query Result Visualization 26
2.4.4 Storage Manager 28
2.4.5 Rendering Considerations 29

2.5 Case Studies . . . 31
2.5.1 Investigating Taxi Activity in Different Regions . . . 31
2.5.2 Exploring Movement: Transportation Hubs . . . 32
2.5.3 Studying Behavior over Time . . . 33
2.6 Discussion . . . 35
2.7 Summary . . . 36

3 Event Guided Exploration of Spatiotemporal Data 37
3.1 Related Work . . . 39
3.2 Background . . . 42
3.3 Spatiotemporal Data and Scalar Functions . . . 44
3.3.1 NYC Taxi Trips Data . . . 45
3.3.2 MTA Subway Data . . . 45
3.4 Identifying and Managing Events . . . 47
3.4.1 Computing Events . . . 47
3.4.2 Event Group Index . . . 48
3.4.3 Querying Events . . . 52
3.5 Visual Exploration Interface . . . 53
3.5.1 Map View and the Query Interface . . . 53
3.5.2 Event Group Distribution View and Timeline View . . . 53
3.5.3 Filtering Interface . . . 55
3.6 Case Studies . . . 57
3.6.1 NYC Taxi data . . . 57
3.6.2 MTA data . . . 60
3.7 Discussion . . . 63
3.8 Summary . . . 65

4 Clustering Trajectory Data for Visual Presentation and Pattern Discovery 66
4.1 Related Work . . . 68
4.2 Background and Notation . . . 70
4.3 Vector Fields as a Generative Model of Trajectories . . . 71
4.4 Vector Field K-Means Fitting Process . . . 72
4.4.1 Fitting Vector Fields . . . 73

4.4.2 Assigning trajectories to vector fields . . . 75
4.4.3 Algorithm Initialization . . . 75
4.4.4 Computational Complexity and Convergence . . . 76
4.5 Experiments and Results . . . 76
4.5.1 Synthetic Data . . . 77
4.5.2 Atlantic Hurricanes . . . 78
4.5.3 GeoLife GPS Trajectory Dataset . . . 78
4.5.4 Call Detail Record Dataset . . . 80
4.5.5 Comparison . . . 81
4.6 Attribute field k-means: Clustering Trajectories with Attributes . . . 83
4.6.1 Extended Notation and Terminology . . . 84
4.6.2 Attribute field k-means Overview . . . 84
4.6.3 Attribute field k-means Overview Fitting Process . . . 85
4.6.4 Comments on Attribute field k-means . . . 86
4.6.5 Sample Experiment . . . 87
4.7 Discussion . . . 87
4.7.1 Parameter selection . . . 88
4.7.2 Advantages . . . 89
4.7.3 Limitations . . . 89
4.8 Summary . . . 90

5 Conclusions and Future Work **91**

A Detailed Derivation of Matrix Form for Vector Field K-Means Fitting Step **95**

List of Figures

1.1 Spatiotemporal data classification by Kisilevich et al. [74]. 3

1.2 Examples of the use of visualization of spatiotemporal data from the 19th century. **(a)** Minard's visualization of Napoleon's campaign, which demonstrates the use of visualization for communication or story telling. **(b)** Map produced by Dr. John Snow in his investigation on the outbreak of Cholera that happened in London in 1854. This demonstrates the use of visualization for data analysis. 4

1.3 Examples spatiotemporal data visualization of different types. **(a)** Show the visualization of ST events (Figure 1.1), while **(b)** and **(c)** show visualizations of trajectory datasets. Notice that due to the display limitations and positioning constraints imposed by the space and time components of the data, these displays get cluttered. In all these examples, interactive filtering and aggregation is used to overcome/reduce the clutter. 5

2.1 Taxis as sensors of city life. The plot on the top shows how the number of trips varies over 2011 and 2012. While some patterns are regular and appear on both years, some anomalies are clear, e.g., the drops in August 2011 (Hurricane Irene) and in October 2012 (Hurricane Sandy). In the bottom, we show pickups (blue) and dropoffs (orange) in Manhattan on May 1st from 7am to 11am. Notice that from 8-10am, there are virtually no trips along 6th Avenue, indicating the traffic was blocked. 11

2.2 Comparison of taxi trips from Lower Manhattan to JFK and LGA airports in May 2011. The query on the left selects trips that occurred on Sundays, while the one on the right selects trips that occurred on Mondays. Users specify these queries by visually selecting regions on the map and connecting them. In addition to inspecting the results depicted on the map, i.e., the dots corresponding to pickups (blue) and dropoffs (orange) of the selected trips, they can also explore the results through other visual representations. The scatter plots below the maps show the relationship between hour of the day and trip duration. Points in the plots are colored according to the spatial constraint represented by the arrows between the regions: trips to JFK in blue, and trips to LGA in red. The plots show that many of the trips on Monday between 3PM and 5PM take much longer than trips on Sundays. 12
2.3 High-level architecture of TaxiVis. 18
2.4 TaxiVis user interface components. (A) Time selection widget, (B) Map, (C) Tool bar, and (D) Data summary. We can also see an example of three distinct queries specified by colors. The orange query (orange border polygon) represents an atomic query with a spatial single region start constraint. The red query (red border polygon) represents an atomic query with a spatial single region end constraint. The blue query is a complex query which is the union of two atomic queries: the polygon with blue border (atomic single region start constraint) and a directional query (blue arrow). 19
2.5 Time selection widgets. The regular time selection widget (top) allows the user to define an atomic temporal constraint. The recurrent time selection widget (bottom) allows the user to easily define complex temporal constraints. In this figure, the widget selects for the years of 2011 and 2012, the period between 9AM and noon in all Tuesdays and Fridays in the month of September. 23

2.6 Attribute constraints are specified (in the middle) to refine the query on the left. They are shown as dark gray strips on the histogram. The result of the refined query is shown on the right and consists of a single trip. 24
2.7 Different spatial visualizations of taxi trips for the week 05/01/2011 to 05/07/2011. In (a) all the trips are rendered To reduce the clutter, TaxiVis uses both LOD (b) and density heat maps (c). The system also supports heat maps based on well-defined regions such as neighborhoods (d). For the latter, summary information about the region can be overlaid on the map, for example, hovering the mouse over a neighborhood highlights it and shows the number of trips. 27
2.8 Comparison of taxi pickups (left) and dropoff (right) in different neighborhoods over the first week of May 2011. The plots show that Midtown and the Upper East side are the most active areas. But over the weekend, there is an increased number of dropoffs in Downtown. The figure also highlights the fact the Harlem is underserved by taxis. 28
2.9 Comparing tips per trip and fare per miles for trips originating in different neighborhoods in the period of 05/01/2011 - 05/07/2011. . 30
2.10 Comparing movement across NYC transportation hubs. On the top, we examine trips starting at the two major airports in NYC: JFK and La Guardia. In the bottom, we refine the query to compare trips starting at the airports with trips starting at the train stations, Penn Station and Grand Central. 32
2.11 Time exploration. (a) shows activity for all Mondays in May 2011 and May 2012. Two such Mondays stand out: 05/30/11 and 05/28/12. Examining the summary plot in (b), we see that the number of trips in these two days is significantly lower than the on other Mondays. 34
2.12 Taxi activity in Manhattan during the week of Hurricane Sandy. On the day the hurricane hit, there are very few taxis on the streets throughout Manhattan. On the next four days, activity goes back to normal in all parts of the city, except in lower Manhattan, where there was a power outage which lasted for five days. 35

2.13 Taxi activity in Manhattan during the week of Hurricane Irene. . . 36

3.1 It is difficult to identify short irregular patterns when the data is aggregated over either space or time. **(a)** A heatmap of taxi locations in Manhattan on 1 May 2011 between 8 am and 9 am. Note that the path of the bike tour contains no taxis. **(b)** The time series plots compare the number of trips that occurred in Manhattan on three Sundays in 2011: 24 April, 1 May, 8 May. It is difficult to distinguish between the three Sundays using just the number of trips, even though an entire stretch of streets are blocked to traffic on May 1st. **(c)** The trips are aggregated over time and displayed as a heat map for the three Sundays. Note that the path of the bike tour (highlighted) looks similar in all the heat maps. 40

3.2 Topology of scalar functions. **(a)** The height function f defined on a graph. The super-level set at f_1 is the set of all points above the highlighted plane and consists of *two* components (colored red). **(b)** Join tree of f. **(c)** The features of the input are defined based on the edges of the join tree. The labeled peaks denote the set of maxima. The features are colored the same as the corresponding edges in (b). π_i denotes the persistence of maximum v_i. Intuitively, the persistence of each maximum is equal to the height the corresponding peak. **(d)** The simplified join tree obtained from removing the maximum v_3. **(e)** The resulting smoothed function. 41

3.3 Overview of the event guided exploration technique. First, (1) the input data is transformed into a time-varying scalar function. (2) Topological features are computed from the scalar functions to identify the set of events. (3) An event group index is then created from the identified events to support efficient querying over a large number of events. (4) A visual interface guides the user towards interesting events (5) in the data, allowing them to select an event and (6) interactively search for similar events. 46

3.4 Identifying minima events. *Left:* The scalar function corresponding to the time step 10 am-11 am on 24 November 2011 is shown using a heat map. *Right:* The set of minima events identified using the split tree. Each connected component of the colored subgraph corresponds to an event. The event corresponding to Macy's Parade is highlighted. 48

3.5 Event group index. The event group key of an event group corresponding to Macy's Parade. The black nodes and edges correspond to the maximum common subgraph of regions of all the events in the event group, and is used to represent the region of the event group. 50

3.6 The visual exploration interface. **(a)** The time line view showing 4 event groups with different densities and ranges. In this view, event groups are sorted according to their topological importance τ. **(b)** The event group distribution view where the time resolution is an hour. **(c)** The event group distribution view when the time resolution is changed to a day. The daily event (purple) moves to Region I for this time resolution. **(d)** The event group distribution view when the time resolution is changed to a week. The weekly event (orange) moves to Region II for this time resolution. 51

3.7 The event group distribution view for November 2011 after filtering. Note that the event group corresponding to Macy's Parade stands out in this view, thus helping the user identify this pattern. 55

3.8 Minima events in NYC. **(a)** Selecting the lone high density event group from Region I of October's event distribution view identifies the event corresponding to the Halloween Parade in 2011. **(b)** Changing the time unit to daily, and selecting high density event groups with range = 2 identifies events corresponding to Hispanic Day Parade and Columbus Day Parade that occurred on consecutive days. **(c)** Changing the time unit to weekly for the month of August, and selecting high density event groups with range = 3 identifies events corresponding to the NYC Summer streets that occurred on three consecutive Saturdays. 56

3.9 Using the timeline view to isolate events. **(a)** Identifying daily events. The most important event group (the topmost event group) in the timeline view corresponds to the daily event corresponding to the Hispanic Day Parade and Columbus Day Parade. **(b)** Identifying weekly events. The timeline view is used to select a periodic event group, which corresponds to the NYC summer streets that happened on Park avenue for 3 consecutive Saturdays. 57

3.10 Identifying trends. **(a)** Taxi hot spots correspond to areas of high activity in NYC. These include transit locations and tourist spots. **(b)** Taxi hot spots during the night corresponds to areas of active nightlife. 61

3.11 Frequent minima events for August 2013 for the south bound 3 trains. Two of the event groups in Region II of the event group distribution view (daily events) corresponds to the Wall Street and the 14^{th} Street stations. Frequency distribution of events in the minimum event group corresponding to Wall street station indicates that this event predominantly occurs during rush hours, i.e., between 8 and 10am and between 4 and 8pm. Also note that the frequency of this event is higher on weekdays than on weekends. 62

3.12 Due to the impact of the scalar function on the shape of the events found, the path taken by the bike tour is split into multiple regions. 64

4.1 Left: 1415 Atlantic tropical storms (from the HURDAT dataset) used as input. The orientation of each trajectory is represented by linearly interpolating from blue (start) to orange (end). This color scheme is used throughout this chapter. Right: four trajectory clusters and their corresponding vector fields showing relative speed. Clusters (b) and (c) contain the Cape Verde-type cyclones, and separate them according to whether they dissipate in North America (b) or turn back to the Atlantic Ocean (c). Clusters (a) and (d) show storms developing in the Gulf of Mexico. We observed faster-moving storms to be in cluster (a), and slower-moving ones in (d). 68

4.2 Illustration of the trajectory tessellation and Laplacian matrix computation. The trajectories are tessellated so each segment is contained on a face of the grid. Each segment s_j determines a constraint in the form of a matrix C_{s_j}. The Laplacian matrix enforces our smoothness penalty. 71

4.3 An illustration of vector field k-means as it partitions 2000 synthetic trajectories into two clusters. The algorithm alternates between fitting the best possible vector fields from the current assignment ("optimize") and matching trajectories to the vector field which fits them best ("assign"). Although no trajectories form a complete circle, vector field k-means still recovers the two separate circular patterns. 77

4.4 Large-scale movement patterns around Beijing, from the GeoLife Trajectories dataset. Clusters (a), (b) and (d) appear to depict travel in and out of the city through the surrounding highways. Cluster (c) has much slower speeds and its trajectories are tightly packed around a small region. Upon inspection, we found this to area to contain the Microsoft Research Asia campus. 78

4.5 The GeoLife Trajectories dataset clustered using vector field k-means. The original trajectories were cropped to a 10 block area in downtown Beijing. We partition the data into four clusters ($k = 4$), and then subdivide each cluster resulting in 16 subclusters. Images (a) and (b) show two clusters from the first-level subdivision, and images (c) and (d) show two clusters from the second level. The first two vector fields show trajectories in patterns of faster vehicular traffic, while the latter appears to show pedestrian traffic moving to and from a lunch spot near the Microsoft Research Asia campus. . . . 79

4.6 **Left.** The anonymized call detail records for over 370,000 cell phone calls produced noisy trajectories around a suburban city. Three of the four clusters computed are shown. **Right.** Despite the noisy trajectories, we recovered clear movement patterns related to highway (bold black lines) traffic. The three vector fields correspond to the clusters on the left. 80

4.7 TraClus experiments using the synthetic dataset. In (a) 268 clusters are found with $\epsilon = 0.03$ and $MinLns = 2$. In (b), with $\epsilon = 0.23$ and $MinLns = 140$ TraClus detects two clusters (drawn here separated for clarity), but clearly merges portions of the two circular patterns. Slight variations on the parameters ($\epsilon = 0.25$ and $MinLns = 160$) causes TraClus to merge the two cluster into one as seen in (c). These results were obtained in 0.8, 1.6, and 6.5 seconds respectively. 81

4.8 TraClus computed nine clusters for the HURDAT dataset. Cluster representative trajectories are in black. Notice that because TraClus starts by subdividing trajectories, no cluster from TraClus captures the pattern of trajectories found by vector field k-means in Figure 4.1 (c). 82

4.9 Illustration of basic concepts used. An 1-attribute f is represented as a height field. A trajectory $\overline{\gamma}$ consistent with f is also shown. . 83

4.10 **(a)** The synthetic dataset described in Section 4.5.1 with colors representing groups of trajectories randomly generated within each circular pattern. **(b)** For each trajectory, a new attribute is generated by sampling the corresponding field. **(c)** By considering these attributes, attribute field k-means is able to separate the generated groups. 87

5.1 Illustration of the interface of the 3D *Urbane* visualization system. Urbane provides architects, developers, and planners with a new, data and analysis rich way of reading the city ultimately facilitating better decision making. Using the visual interface together with the map view, they can simulate the affect of such development. For example, the views to the freedom tower (highlighted in green) of the buildings highlighted in red would be adversely impacted if the new constructions (colored purple) are built. 92

List of Tables

2.1 Summary of experiments with data storage strategies. 29

3.1 Event groups similar to Hispanic Day Parade. 59

4.1 Experimental results: For each dataset, we report the number of trajectories, the grid resolution, the number of clusters (k), and the total running times (in seconds) for the vector field fitting (optimize) and trajectory assignments. 76

Chapter 1

Introduction

This dissertation concerns the design of visual analytics techniques for the exploration of spatiotemporal data.

1.1 Motivation

Spatial and temporal interactions are central and fundamental in pretty much all activities in our world and society. Everyday, people and goods travel around the world at different speeds and scales; migratory animals engage in long-distance travels that demonstrate the biological integration around the globe; weather phenomena, like typhoons and hurricanes, form and move around the earth and may have large social-economic impact. In all these examples, proper understanding of the underlying phenomena can produce insights with the potential to shape the future development in those domains.

The rapid development of acquisition technology and the popularization of GPS enabled mobile devices has resulted in the production of data with spatial and temporal components (also known as spatiotemporal data) at massive rates and at low cost. A famous assertion in the geoscience and geo-business communities claims that "80% of all information contains some geospatial reference" [66]. While it is hard to be precise on the exact proportion, recent studies claim that the amount of spatial/spatiotemporal data is in the order of 50%-80% of all the produced data [20, 66]. The availability of such large collections of data has given rise to new opportunities for data-driven analysis that can 1) generate new insights; 2)

new value creation; and 3) highly influence decision making in a diverse set of domains. For example, it was estimated that the amount of personal location data produced in the year of 2009 was 1 petabyte (with expected growth of 20 percent a year), and that over the following decade, new applications in sectors such as telecommunications, retail, and media would bring revenues of the order of hundreds of billions of dollars [88].

In order to take advantage of all these data and realize their potential for impact in decision making and business, it is crucial to be able to extract knowledge from them.

1.2 Spatiotemporal Data Types

Formally, spatiotemporal data are defined as high-dimensional data records for which the different dimensions can be classified in three components: spatial (e.g., geographical coordinates), temporal (e.g., timestamps), and attribute (e.g., temperature, pressure, etc.), Given the wide range of applications, it is not surprising that there exists a number of distinct types of spatiotemporal data. Kisilevich et al. [74] proposed the classification of spatiotemporal data summarized in Figure 1.1. This classification is based on two main aspects namely the *spatial dimension* and the *temporal dimension*. With respect to the spatial dimension, the data is classified according to whether or not the spatial component of the data records is fixed (such as information collected by sensors with fixed location) or not (GPS tracks of moving vehicles). The temporal dimension is classified with respect of how much of temporal evolution is recorded. In the simplest case, the data recorded does not evolve, in which case only a *single snapshot* is available. A more complex case corresponds to data that evolves with time, but only the most recent snapshot is available, which is called *updated snapshot*. Finally, the most complex case, called *time series*, corresponds to the case in which the full temporal history is kept. As displayed in Figure 1.1, these different alternatives generate multiple types of datasets. Furthermore, as also shown in Figure 1.1, the classification above can be extended by considering the *spatial extension* or geometry of the objects under consideration. In fact, in some scenarios, it is important to properly model the geometry of the objects. For example, in applications that deal with road segments

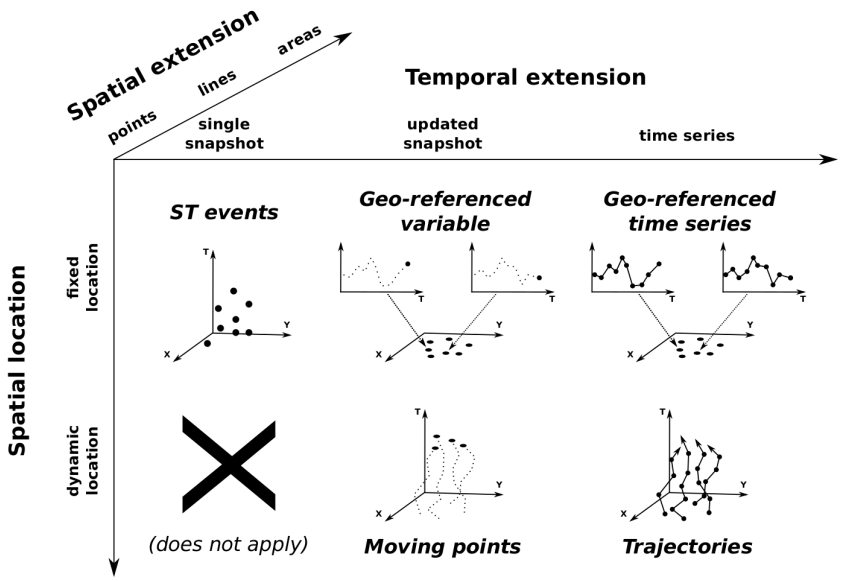
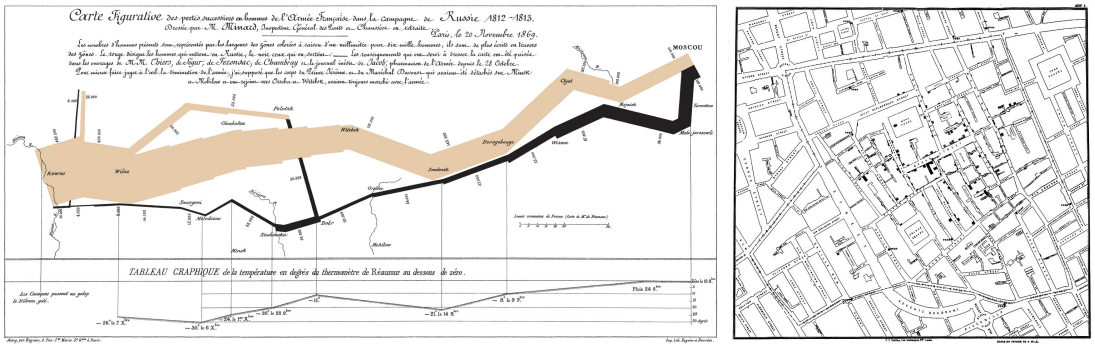


Figure 1.1: Spatiotemporal data classification by Kisilevich et al. [74].

or pieces of land one might consider line and area objects. However, this is rare and the most common case corresponds to point objects, which do not consider the spatial extension. Therefore, our focus is on point objects in the classification.

1.3 Visual Analysis of Spatiotemporal Data

Data visualization is a fundamental tool in the presentation and analysis of spatiotemporal data. By definition, the goal in understanding this type of data is to put the phenomena being studied in context with location and temporal aspects. Visualization techniques attack this problem by leveraging the power of the human visual system to interpret visual representations of the data to be analyzed. The related field of cartography is a great example of the effectiveness of the use of visualizations for spatiotemporal data. In fact, for centuries, cartography techniques have been developed to produce maps to effectively present spatial and spatiotemporal data, with the earliest maps dating from 4^{th} millennium BC[114]. Figure 1.2 shows two famous and influential cases of the use of such techniques, namely Minard's visualization of Napoleon's campaign (Figure 1.2(a)) and the London cholera cases map by Dr. John Snow (Figure 1.2(b)). Minard's visualization



(a) (b)

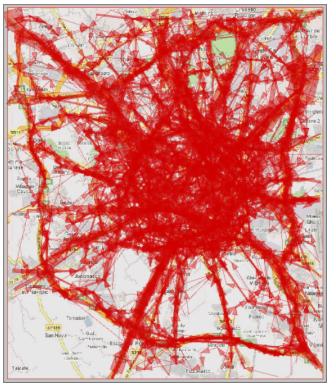
Figure 1.2: Examples of the use of visualization of spatiotemporal data from the 19th century. **(a)** Minard's visualization of Napoleon's campaign, which demonstrates the use of visualization for communication or story telling. **(b)** Map produced by Dr. John Snow in his investigation on the outbreak of Cholera that happened in London in 1854. This demonstrates the use of visualization for data analysis.

is a typical case of the use of visualization for presentation, which was the concern of most of the early works in cartography. On the other hand, John Snow's map is an example of the use of maps for data exploration and decision-making, which has been the concern of most modern map design techniques [86] and is the focus of this dissertation.

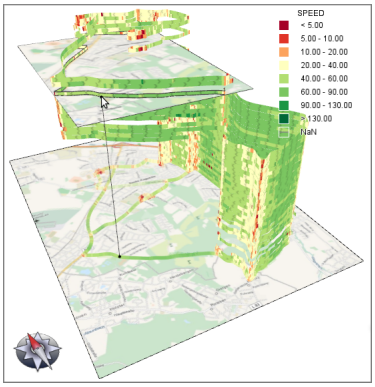
The popularization of powerful computers and interactive computer graphics has widely changed the way visual exploration of spatiotemporal data is done. It induced a major shift from manually produced visualizations (such as the ones in Figure 1.2) to highly interactive visualization systems. In general, such systems provide interactive tools to select portions of the data and browse through them. This enables quick hypothesis making and confirmation, important stages of the data exploration process. All these features make interactive systems essential in the exploration of large collection of complex data such as spatiotemporal data [56].

However, there are many challenges in designing systems for the exploration of spatiotemporal data, and these have been the focus of several recent research efforts [6, 31, 53, 54, 83, 117, 127**?**]. One of these challenges is providing interactivity while handling large datasets. This involves being able to process (e.g., query, filter, and aggregate) large amounts of data at interactive rates in real time in order to visualize them. This is crucial since interactivity plays a major role

(a) Map showing location types by Kruger et al. [75]



(b) Movement visualization by Rinzivillo et al. [108]



(c) Multivariate trajectory visualization by Tominski et al. [118]

Figure 1.3: Examples spatiotemporal data visualization of different types. **(a)** Show the visualization of ST events (Figure 1.1), while **(b)** and **(c)** show visualizations of trajectory datasets. Notice that due to the display limitations and positioning constraints imposed by the space and time components of the data, these displays get cluttered. In all these examples, interactive filtering and aggregation is used to overcome/reduce the clutter.

in the data analysis process [56] and even small latency can drastically impact the outcome of the analysis [84]. A second major challenge is how to effectively provide visual representations and interactions of spatiotemporal [62, 109]. In general lines, this is due to the fact that spatiotemporal data is by its nature high-dimensional and that, different from abstract high-dimensional data, space and time have contextual information which constrains how the data are placed in a visualization. The consequence of this is that visualizations of spatiotemporal data get cluttered even with small amounts of data. Figure 1.3 exemplifies this fact for different types of spatiotemporal data. Filtering and aggregation strategies are often applied to solve this problem [4, 54]. While this approach eliminates clutter,

it might hide interesting patterns. Furthermore, deciding the right way to slice the data through filtering and also the right level of aggregation in order to find patterns are hard problems. In fact, interesting patterns in spatiotemporal data might happen in different scales over different spatiotemporal regions. In order to find these patterns one would need to examine a prohibitively large number of spatiotemporal slices even when using sophisticated interactive tools. For this reason it is widely acknowledged that purely visual/interaction methods are not enough to allow an effective exploration and pattern discovery [14]. Therefore there is a need for techniques that will enhance the analytical capabilities of interactive visualization.

The study of such techniques is the goal *visual analytics*. As defined by Thomas and Cook [34], visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces. The main overall premise of this research area is that while techniques of the interactive visualization and pattern mining domains are important and powerful on their own, they are limited if considered in isolation. Therefore, in order to build more effective tools that support analytical reasoning for data analysis it is necessary to combine the capabilities of these two domains.

1.4 Thesis Contributions

The goal of this thesis is to develop visual analytics techniques to attack the challenges in the visual exploration of spatiotemporal data.

The first contribution of this thesis is the design of the TaxiVis visual exploration system. TaxiVis brings together two main components: an expressive visual query model for spatiotemporal data and a custom-built data layer. In spatiotemporal data, querying is complex due to the many dimensions of the data. Being able to easily express complex queries is key to an effective exploration process. The proposed visual query model is able to express a wide range of spatiotemporal queries. These queries can be easily composed and refined, which makes the process of testing new hypothesis very simple. In TaxiVis, the query model is coupled with an efficient data layer that enables query response at interactive rates even with larges amount of data. In addition, TaxiVis makes use of coordinated views and strategies based on aggregation and adaptive level-of-detail to generate informative

visual summaries for query results even when these results are large. While a particular example, TaxiVis can be seen as a representative of the class of systems for visualization of spatiotemporal data in the sense that it employs many of the visualization/interaction techniques generally used in this type of system.

As briefly discussed earlier (and exemplified later in this dissertation), while systems like TaxiVis are very powerful and enable interactive exploration of large spatiotemporal datasets, many patterns might remain undiscovered. Motivated by this fact, in this thesis we propose two pattern mining techniques that help in the navigation through the data via pattern discovery. The two techniques presented have in common the of use concepts and techniques widely applied in scientific visualization and computer graphics such as computational topology and vector fields. As described later, this approach allows us to have novel perspectives on the problems of finding patterns in spatiotemporal data that, to the best of our knowledge, have not been considered in the machine learning and data mining fields. Furthermore, in doing so we can leverage the extensive body of algorithmic and visualization research on those concepts to propose mining techniques that are mathematically well founded, computationally efficient, and the results of which can be understood via previously proposed visual representations. On the other hand, the two techniques differ in the type of spatiotemporal data they handle as well as the type of pattern that they can unveil.

The first technique consists of a topology-based technique whose main objective is to help users to find the "needle in the hay stack", i.e., guide users towards *interesting* slices (spatiotemporal regions) of the data. We call this process *event guided exploration*. The overall idea behind this technique is to convert the data to a time-varying scalar function (or *geo-referenced time series*, as described in Figure 1.1). This scalar function models the phenomena under study. Extreme points of this time-varying scalar functions treated as *events* and these are discovered through the use of computational topology, which is not only computational efficient, but also allows a diverse set spatiotemporal events to be mined. The discovered events are then organized in an index data structure which allows both fast retrieval and querying. As shown later, via visual exploration of the collection of extreme points extracted over time, important events of the data can be found with relatively a small amount of work by the user. By definition, extreme points are related to

static patterns in the data, i.e., they do not explicitly model the notion of movement.

Unveiling movement patterns is the goal of the second pattern technique proposed in this thesis. It consists of a clustering technique designed for pattern finding in *trajectory data*. As described previously, this type of data is characterized by the temporal evolution of spatial and attribute components in a continuous manner, which causes of clutter in visual representations. Our technique attacks this problem by, instead of visualizing the entire data, partitioning the data in a way that each group in the partition avoid clutter and therefore reduce the complexity of the data that is shown. Since clearer visualizations are produced, it becomes easier to understand the visualizations and identify patterns. In order to achieve this, we propose a novel clustering approach for trajectory data, called *Vector field k-means*. In this approach, trajectories are modeled as streamlines of vector fields. One important feature of this modeling strategy is that it tries to avoid overlapping trajectories to have discrepant directions at their intersections. Clustering is achieved by using the spatial component of trajectories to fit a collection of vector fields to the given trajectories. This technique achieves richness and expressivity of features, simplicity of implementation and analysis, and computational efficiency. Furthermore, the obtained vector fields serve as a visual summary of the movement patterns in each cluster. Therefore, techniques for vector field visualization can be leveraged to display these movement patterns. Finally, Vector field k-means can be naturally generalized to consider not only the spatial component of the trajectories, but also their attribute components. This is achieved by using a different modeling strategy based on *scalar fields*, which we call *Attribute field k-means*.

1.5 Thesis Outline

Chapter 2 describes the design of the TaxiVis system for the exploration of large collections of taxi trip records. Chapter 3 presents the use computational topology based technique for event guided exploration. Chapter 4 describes the Vector field k-means technique and its Attribute field k-means generalization for trajectory clustering. Finally, Chapter 5 concludes this thesis and presents directions for future work. The work presented in this dissertation has been published in peer-reviewed journals [38, 51, 54, 60].

Chapter 2

The TaxiVis System for Exploration of Taxi Trip Records

In this chapter we present the design of the interactive visualization system TaxiVis. This system allows the exploration of large collections of spatiotemporal data. The main motivation for the development of TaxiVis was the exploration of one such dataset namely New York City's taxi trip records.

In New York City, each day 13,000 taxis carry over one million passengers and make, on average, 485,000 trips—totaling over 175 million trips a year. Studying taxi trip records can reveal social and economic aspects of the city. For example, consider the plot in Figure 2.1, which shows how the number of trips per day varies over 2011 and 2012. There is a lot of regularity: the plot lines are very similar for the two years. For example, on Thanksgiving, Christmas and New Year's eve, there is a substantial drop in the number of trips. But the plot also shows some anomalies. There are big drops in August 2011 and October 2012, which correspond to hurricanes Irene and Sandy, respectively. Looking at the data at a finer grain, other interesting patterns emerge. The maps in Figure 2.1 show the density of taxis across Manhattan from 7am to 11am, on May 1st, 2011. From 8am to 10am, taxis disappear along 6th avenue, from Midtown to Downtown; and then, at 10am they reappear. As it turns out, during this period, the streets were closed to traffic for the NYC Five Boro Bike Tour.[1] This example shows that taxi cabs can be seen as valuable sensors of city life. Furthermore, this dataset (described in Section 2.2)

[1] http://www.nycbikemaps.com/spokes/five-boro-bike-tour-sunday-may-1st-2011

presents multiple challenges not only to the current tools used by domain experts interested in the data, but also to the most used visual data exploration system such as Tableau and Spotfire. For this reason, while TaxiVis is general and its components can be used in a variety of datasets, we base the discussion of this chapter on the New York City's taxi trip dataset.

We highlight two main challenging aspects in the exploration of this dataset that are part of the general challenges discussed in Chapter 1.3. First, its size limits the use of tools with in-memory storage such as R, MatLab, Stata, ArcGIS and Excel. These tools are commonly used by traffic engineers, economists and urban planners in their data analysis tasks. While some of these tools allow the use of external database management systems, they do not provide the desired latency for interactive analysis [49]. Second, the multiple attributes of the dataset along with its spatiotemporal components makes data selection for exploration difficult. The consequence of this is that domain experts often rely on intuition when selecting portions of the data, which limits them to only perform confirmatory analysis. These challenges prevent the domain experts from analyzing the *whole* data. In fact, the usual analysis workflow when dealing with this dataset is to first select small slices and then load them into these tools for analysis. This process is both tedious and time consuming.

In order to overcome those challenges, an important goal in the TaxiVis design was to unify the two phases of exploration: data selection and visual analysis. We posit that by doing so, domain experts will be able to carry out both confirmatory and exploratory analyses over the complete taxi data set. Since domain experts often do not have computer science training, another desired feature of our system was that it should be easily usable and not require knowledge of specialized programming or query languages. Simplicity must be balanced with expressiveness: the exploratory system has to be expressive enough to support spatiotemporal selection and analysis at different scales both in space and time, selections over the different trip parameters, and the ability to refine and generalize queries. Furthermore, due to the scale of the data, the system should be able to provide interactive selection response times as well as support result summarization. Besides giving insights into the data, summaries can help guide the exploration, including hints about potentially useful query refinements. Also, exploration must be flexible,

Figure 2.1: Taxis as sensors of city life. The plot on the top shows how the number of trips varies over 2011 and 2012. While some patterns are regular and appear on both years, some anomalies are clear, e.g., the drops in August 2011 (Hurricane Irene) and in October 2012 (Hurricane Sandy). In the bottom, we show pickups (blue) and dropoffs (orange) in Manhattan on May 1st from 7am to 11am. Notice that from 8-10am, there are virtually no trips along 6th Avenue, indicating the traffic was blocked.

allowing users to go back and forth from aggregated summaries to individual objects.

To address these needs we designed TaxiVis which is shown Figure 2.2. The main features of this system are a custom-built data layer and a new visual query model. This data layer is able to provide interactive query response times for the queries supported by our query model. The proposed query model enables complex spatiotemporal queries over all the components in the taxi dataset (Section 2.3). The use of visual querying eleminates the need of the knowledge of a textual query language: the data can be directly queried using visual operations. We show that this model is able to express a wide range of queries, and in particular, the three classes of queries in Peuquet's typology [106]: identify a set of objects at a given location and time; given a time and a set of objects, describe the locations occupied by the objects; and describe the times a set of objects occupied a given set of locations.

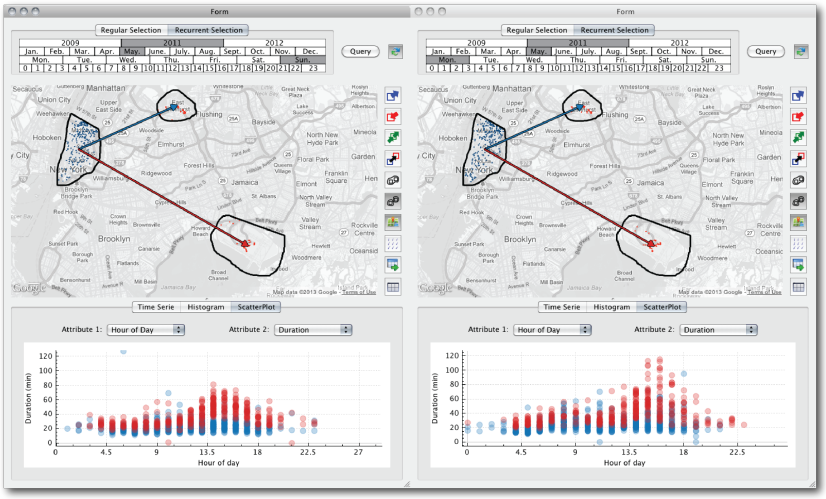


Figure 2.2: Comparison of taxi trips from Lower Manhattan to JFK and LGA airports in May 2011. The query on the left selects trips that occurred on Sundays, while the one on the right selects trips that occurred on Mondays. Users specify these queries by visually selecting regions on the map and connecting them. In addition to inspecting the results depicted on the map, i.e., the dots corresponding to pickups (blue) and dropoffs (orange) of the selected trips, they can also explore the results through other visual representations. The scatter plots below the maps show the relationship between hour of the day and trip duration. Points in the plots are colored according to the spatial constraint represented by the arrows between the regions: trips to JFK in blue, and trips to LGA in red. The plots show that many of the trips on Monday between 3PM and 5PM take much longer than trips on Sundays.

The ability to specify queries using graphical widgets and visualize their results was in part inspired by the seminal work by Ahlberg et al. [3] on dynamic queries. Our focus, however, is different: we aim to support the exploration of large, spatiotemporal OD data, and provide visualization services that are both usable and efficient. Another important feature of our model is that each query is associated with a set of trips. As a result, not only can queries be composed and refined, but also query results can be aggregated and different visual representations can be applied while still maintaining the spatial and temporal contexts. Query composition also enables the use of *cross-filtered views* [128], which is key in our model to support the query classes in Peuquet's typology.

TaxiVis combines a number of interaction capabilities that enable users to

pose queries over all the dimensions of the data and flexibly explore the attributes associated with the taxi trips. Another important feature of the system is the ability to compare spatiotemporal slices through multiple coordinated views. Users can interactively compose and refine queries as well as generalize them by performing parameter sweeps. To avoid clutter, the system implements a number of strategies to render a large number of graphical primitives on a map. As discussed in Section 2.4, these include the use of aggregation and adaptive level-of-detail rendering.

TaxiVis has been deployed to a number of agencies in NYC such as the Taxi and Limousine Commission (TLC) and Department of Transportation (DoT) and has been in use since its release. In Section 2.5, we demonstrate the usefulness of TaxiVis through a series of case studies motivated by traffic engineers and economists whose needs have driven our design. These case studies show how our model (and system) enable domain experts to perform tasks that were previously unattainable for them.

2.1 Related Work

Analyzing Taxi Data. Recognizing the wealth of information that can be obtained from taxi data, recently, several efforts have focused on their analysis. Ge et al. [59] and Yuan et al. [132] proposed recommendation systems for taxi drivers to optimize the cost of finding passengers (and therefore reduce gas emission). Pan et al. [99] used pickup and dropoff information to classify location according to land use. Veloso et al. [122, 123] used taxi data to study human mobility in Lisbon. They explored patterns in distribution of pickup and dropoff locations and the association with the city context. Liang et al. [81] studied the taxi data from Beijing and reported that the displacement distribution of taxis follows a different distribution than the majority of human mobility data. Peng et al. [105] analyzed 1.58 million taxi trips in Shanghai, and through the use of modeling algorithms, they concluded that people travel on workdays mainly for three purposes: commuting between home and workplace, traveling from workplace to workplace, and others such as leisure activities. Liao et al. [82] designed a visual analytics system to detect anomalies in GPS streams produced by taxis. They produce a visualization of the stream, apply an anomaly detection model, and leverage the user interaction

to incrementally improve their model. These works share an important requirement: the need to perform exploratory analyses. The visual query model and scalable system we propose have the potential to benefit such efforts by making it easier to carry out these analyses.

Visualizing Movement. Movement data has recently received substantial attention in the visualization community (see [16] for a survey). Much of the work has focused on trajectory data, where the complete trace of the moving entities is recorded. In contrast, the data we consider (Section 2.2) is multi-variate OD data [131]: only the start and end positions are recorded, together with attributes associated with the movement. Techniques have also been devised to visualize OD data. Phan et al. [107] proposed a method to automatically generate flow maps that show the movement between two locations. Wood et al. [131] proposed OD maps, which encode trips as a set of spatially ordered small multiples to avoid occlusion effects that occur when flow maps are applied to a large number of trips. While these two techniques consider only space, Boyandin et al. [23] proposed Flowstrates, which takes both space and time into account. These techniques are orthogonal and could be combined with our model. For example, in addition to the plots currently supported by the data summary view of our system (Section 2.4), we could also display a Flowstrate visualization for users to explore the results of origin-destination queries.

Visual Data Selection and Queries. Data selection is a fundamental task for querying and visualization. A number of approaches have been proposed to simplify this task. Ahlberg et al. [3] introduced the concept of dynamic queries, where queries are specified through graphical widgets. VIQING [98] allowed users to create selection queries by directly manipulating visualizations. Heer et al. [68] went a step further and proposed the use of query relaxation to interactively generalize selections. Our approach follows in the tradition of these systems: users can specify selections visually as well as explore the results through direct manipulation of visualizations; and they can also generalize queries. Shrinivasan and Van Wijk [113] proposed the creation of a Select & Slice table during an exploration process to help cross-tabulate semantic zones (i.e., user-defined data regions) and data subsets. Using these tables, users can manipulate zones and explore the relationship between zones and data slices. Another notable effort in this direction is the Polaris

system [116] and its successor, Tableau. Polaris pioneered a visual model for users to specify queries and visualizations by dragging database column names onto shelves for visual variables. A distinguishing feature of our model is that the visual operations were designed to support spatiotemporal queries over OD data. Note that systems like Tableau were developed to manipulate tabular data, and thus, they can also support spatiotemporal queries over data stored in tables. However, specifying spatiotemporal selections and comparing them in these general systems can be challenging. Another difference between our model and Tableau's is that the latter provides a visual interface for users to construct queries—it does not support direct querying over the visualized data.

Visual query models have been proposed for spatiotemporal data [40, 47, 55]. They infer queries from the topology of query sketches specified by the users. Like our model, these works aim to achieve expressiveness. However, they have important limitations that make them unsuitable for our problem. First, they were designed for continuous spatiotemporal data, i.e., when full trajectories are recorded. Second, their "dictionary" of visual representations is very complex—users need to master a number of logic operators and visual syntax, which negatively impacts usability.

2.2 Data and Design Requirements

The New York City Taxi Data. The data used in our study was provided by Taxi and Limousine Commission of New York City and contains information about all medallion taxi trips in 2009, 2011, and 2012. The raw data came in CSV files with a total size of approximately 120 GB and contained over 540 million trips. Each trip record consists of: trip id, taxi id, driver id, pickup location, dropoff location, pickup date and time, dropoff date and time, traveled distance, fare amount, tip amount and toll amount.

Taxi and driver ids were anonymized so as to avoid the linking of records to the actual taxi medallion and taxi driver's license. Note that the trajectory of a trip is not recorded.

As just described, due to technological limitations, at the time of our collaboration only the start and end positions for each trip are recorded. This type of spatiotemporal dataset is called *origin-destination* or *OD* dataset. This can be

seen as a subtype of trajectory datasets (Chapter 1.2). The design of TaxiVis takes this in consideration and focus on OD data. This simplifies the type of queries that need to be performed and therefore the data layer used to process them. However, as discussed in Section 2.6, it is easy to extended our system to support queries on general trajectory datasets.

Desiderata for Visual Exploration. We have conducted interviews with a total of five researchers from Economics and Traffic Engineering departments (2 engineers and 3 economists) from New York University to better understand the questions they want to investigate. We have identified different kinds of queries the experts want to perform. They are interested in understanding the dynamics of the city, how different aspects of the data vary over space and time. For example, "What is the average trip time from Midtown to the airports during weekdays?" or "How does the taxi fleet activity vary during weekdays?". They also want to explore particular events at a given time such as "How was the taxi activity in Midtown affected during a presidential visit" or "How did the movement patterns change during hurricane Irene?"; and how patterns differ across different neighborhoods, for example "What is the frequency of taxis in Midtown and Harlem?". As they explore the data, they need to freely mix these queries and drill down from a high-level summary to an individual event.

As in any longitudinal analysis, comparing different data slices is essential: how patterns vary across space and time, and at different scales. For example, "How the movement changes between Midtown and JFK throughout the day, and over different days of the week". In addition, they need the ability to quickly test hypotheses. For example, starting with a query about a specific neighborhood ("What are the movement patterns between the Midtown and JFK?") and then generalizing it to all neighborhoods in Manhattan.

Currently, the analyses carried out by the experts are mostly confirmatory. They use general-purpose analysis tools such as R, MatLab, Stata, Excel, and use basic filters and visual tools to verify their hypotheses. Since these tools have limited scalability, the experts must select a (small) subset of the data to analyze. As they identify patterns and explore hypotheses that need to be tested on other data slices, they need to go back to the raw data. Not only is this process tedious and error-prone, but it also prevents them from performing analyses over the whole data

set, or even over a single year. Furthermore, it is hard to replicate the analyses, to apply an analysis over different data subsets, and to compare different data slices.

To address these limitations, an important goal of our work is to unify the two phases of exploration: data selection and visual analysis. We posit that by doing so, domain experts will be able to carry out both confirmatory and exploratory analyses over the complete taxi data set. Since these experts do not have computer science training, the system should be usable and not require knowledge of specialized programming or query languages. Simplicity must be balanced with expressiveness: the exploratory system has to be expressive enough to support spatiotemporal selection and analysis at different scales both in space and time, selections over the different trip parameters, and the ability to refine and generalize queries.

Due to the large-scale of the data, the system should also support result summarization. Besides giving insights into the data, summaries can help guide the exploration, including hints about potentially useful query refinements. Furthermore, exploration must be flexible and allowing users to go back and forth from aggregated summaries to individual objects.

The TaxiVis System. With these requirements in mind, we have built TaxiVis, a system for exploring large OD and spatiotemporal data. A key component of the system is a visual query model that is easy to use, yet expressive. As we discuss in Section 2.3, the system supports the query types defined in the *Triad Framework* [106]. The functional modules of the system are shown in Figure 2.3. Users formulate queries visually, by interacting with maps and other visual representations. Internally, a textual query is generated which is then evaluated by the storage manager. To support spatiotemporal queries at interactive rates, we have built a specialized index based on k-d trees [37] (Section 2.4.4). Once the results are derived, the system renders them on the map and users can iteratively refine their queries through visual interactions. Since result sets can be large, we make use of adaptive level of detail and density heat maps. (Section 2.4.3). In order to create these visualizations, additional information such as trip frequencies, must be computed for each spatial region. TaxiVis makes use of data filters to generalize this process. In Section 2.4, we describe the system in more detail.

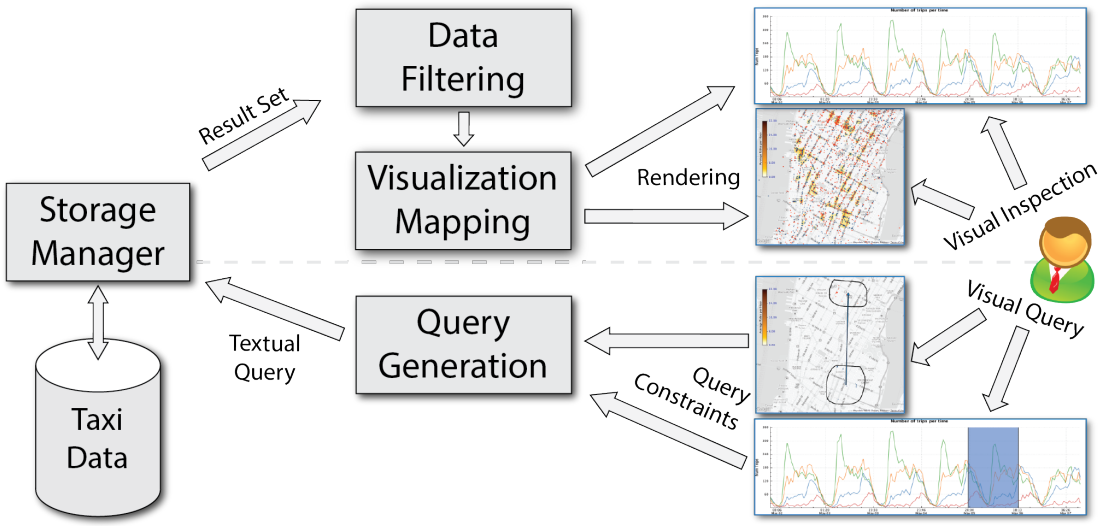


Figure 2.3: High-level architecture of TaxiVis.

2.3 Visual Query Model

Based on the requirements set forth in Section 2.2, we have designed a visual query model that aims to achieve a balance between simplicity and expressiveness. Users specify queries visually and they can iteratively refine their queries through direct manipulation of the results. Below, we present the model and describe how it simplifies the selection of spatiotemporal slices, and enables both query composition and result exploration. We also discuss the different classes of queries it supports. Note that, while this model was designed for taxi trips, it can be applied to other similar OD and spatiotemporal data.

2.3.1 Defining and Composing Queries

A key challenge in formulating spatiotemporal queries is selecting (and refining) the data slices through query constraints. In our model, queries follow the following template:

```
SELECT * FROM trips
WHERE <constraints>
```

Instead of requiring users to write the constraints in the WHERE clause, they do so through visual operations. In our model, there are three types of constraints:

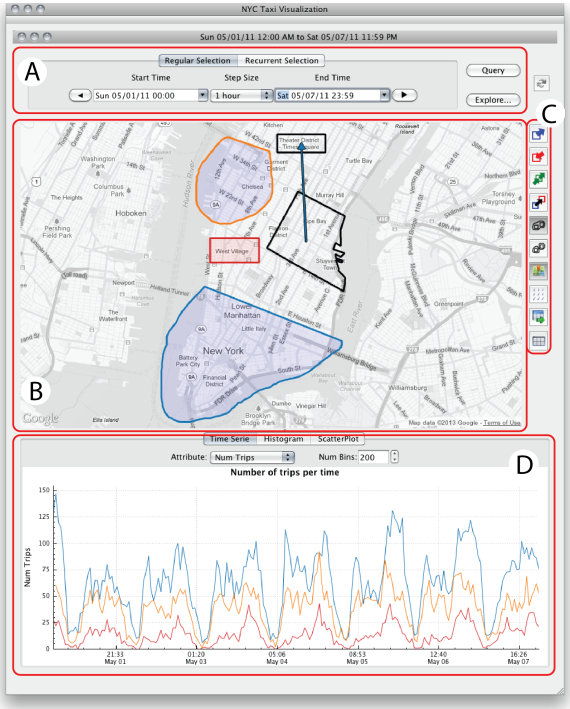


Figure 2.4: TaxiVis user interface components. (A) Time selection widget, (B) Map, (C) Tool bar, and (D) Data summary. We can also see an example of three distinct queries specified by colors. The orange query (orange border polygon) represents an atomic query with a spatial single region start constraint. The red query (red border polygon) represents an atomic query with a spatial single region end constraint. The blue query is a complex query which is the union of two atomic queries: the polygon with blue border (atomic single region start constraint) and a directional query (blue arrow).

spatial, temporal, and attribute constraints. These constraints span all the variables in the taxi dataset (in fact, any OD or spatiotemporal dataset). Furthermore, each query is associated with the set of trips contained in its results. Since each trip is uniquely identified by the trip id, *queries can be composed*: users can iteratively refine queries and further explore results. This has two important implications: it allows the creation of summaries and visualizations while maintaining the spatial and temporal contexts, and enables queries to be applied directly to the derived visualizations. To formalize the process of query composition and properly define query semantics, we use two types of queries: atomic and complex queries, which use atomic queries as building blocks.

Atomic Queries. An atomic query consists of a set of temporal, attribute and spatial constrains. *Temporal constraints* define intervals that bound the values of the time range of the query. A temporal constraint is specified by an interval $[t_{Min}, t_{Max}]$. A trip satisfies the constraint if *trip.pickup_time*,*trip.dropoff_time* $\in [t_{Min}, t_{Max}]$. It is also possible to have constraints that bound just the pickup or just the dropoff time.

An *attribute constraint* can be expressed using equality conditions (for categorical attributes) or interval conditions (for numerical attributes). A trip satisfies an attribute equality constraint associated with a categorical attribute A if for the given value a, $trip.A = a$. If the constraint is associated with a numerical attribute, the trip satisfies the constraint for the interval $[l_A, r_A]$ if $trip.A \in [l_A, r_A]$.

Spatial constraints come in two flavors: single-region and directional constraints. A single-region constraint is defined by a connected spatial region and is associated either with the pickup location (start constraint) or the dropoff location (destination constraint). A trip satisfies the constraint for region r if either $trip.pickup\_region \in r$ (for start constraints) or $trip.dropoff\_region \in r$ (for destination constraints). Directional constraints are used to construct queries about origins and destinations. A directional constraint bounds the regions associated with both pickup and dropoff locations. Given source and destination regions, r_{source} and r_{dest}, respectively, a trip satisfies the constraint if $trip.pickup\_location \in r_{source}$ and $trip.dropoff\_location \in r_{dest}$.

We define a function called *result* which takes as input an atomic query and returns the set of all *trip* records that satisfy the query constraints. The *result* function determines how queries are evaluated. Atomic queries can be combined to construct new atomic queries. Given two atomic queries Q_1, Q_2, a new query Q_3 can be constructed such that $result(Q_3) = result(Q_1) \cap result(Q_2)$. This is possible due to a basic property of the query constraints: they are *closed under intersection*. This can be easily verified for interval and equality constraints, since both are closed under intersection; of course, intersections can be empty.

For spatial constraints, if they are of the same type (start and destination single regions, or directional constraints) or if one is a single region constraint and the other is a directional constraint, they can be combined into a single constraint by reducing (intersecting) the corresponding regions. Otherwise, one must be a single

region start constraint and the other must be a single region destination constraint in which case both can be combined in a directional constraint. As we describe in Section 2.4, this forms the underpinning of the *grouping* operation in TaxiVis.

Complex Queries. A complex query is constructed by combining a set of atomic queries through disjunction. We give meaning to these queries by extending the *result* function inductively. Note that an atomic query is a special case of a complex query, where the query set has a single element. Then, given two complex queries, Q_1 and Q_2, $result(Q_1 \cup Q_2) = result(Q_1) \cup result(Q_2)$. In general, given an atomic query Q it is not possible to find an atomic query Q' such that $result(Q') = result(Q)^C$ (the complement of $result(Q)$). However, it is always possible to define a complex query Q' that satisfies this condition. Thus, set theoretic operations can be performed on the result of complex queries to build new complex queries.

Visual Representation. Figure 2.4 illustrates how atomic and complex queries are visually represented in our system. Temporal constraints are specified using time-selection widgets (A), and attribute constraints are defined in a separate view (see Figure 2.6). We describe both of these, as well as constraints defined within the tool bar (C), in Section 2.4. Here, to illustrate the semantics of the query model, we focus on spatial views which are defined on the map view (B). Single-region constraints are defined by polygons and directional constraints are defined by arrows. The transparent color in the interior of the polygons define the type of the constraint: blue means start constraint, red means destination constraint (see Figure 2.4). The colors on polygon borders and arrows identify distinct queries (there are 3 queries orange, red, and blue). The orange and red queries are atomic queries, consisting of only atomic temporal and spatial constraints. The blue query Q is a complex query, composed by the union of two atomic queries: a single-region start query Q_1 and a directional query Q_2. In SQL-like textual notation, Q_1 can be represented as:

SELECT * **FROM** trips
WHERE $trip.pickup\_time \in [05/01/2011, 05/07/2011]$
AND $trip.pickup\_location \in R_1$

where R_1 denotes the blue region selected in the map. And Q_2:

SELECT * **FROM** trips
WHERE *trip.pickup_time,trip.dropoff_time* $\in [05/01/2011, 05/07/2011]$
AND *trip.pickup_location* $\in$ NYCNeighborhood('*Gramercy*')
AND *trip.dropoff_location* $\in$ NYCRegion('*Times Square*')

where NYCNeighborhood and NYCRegion are functions that given a neighborhood name or region name, respectively, returns the corresponding spatial region.

2.3.2 Exploring Query Results

As described above, atomic and complex queries return a set of trips. Thus, given a set of queries, such as the ones shown in Figure 2.4, other queries can be applied to their results and different visual representation can be used to explore them. For example, in this figure, the plot below the map shows the number of trips returned by each query—the lines in plot are linked to the queries by their color. In Figure 2.2, a scatter plot is used to examine the duration of trips to the airports at different times of the day. Other types of visual representations can be used, including, for example, representations that are specific for OD data [23, 107, 131]. As we discuss later, these visualizations can be directly manipulated to visually define attribute constraints and construct refined queries (see Figure 2.6). Last but not least, through the use of multiple coordinated views, query results can be compared side by side.

2.3.3 Query Expressiveness

The proposed query model is able to express a rich class of queries. In particular, it supports queries types in Peuquet's Triad Framework [106]. Peuquet considered the different components of spatiotemporal data—space (where), time (when) and objects (what), and classified the set of questions that are possible over these components. Below, we describe these questions and how they are supported in our model.

when + where → what. These queries describe objects that are present at a given location or set of locations at a given time or set of times. In our model, they can be constructed in a straightforward fashion, through the definition of spatial and time constraints.

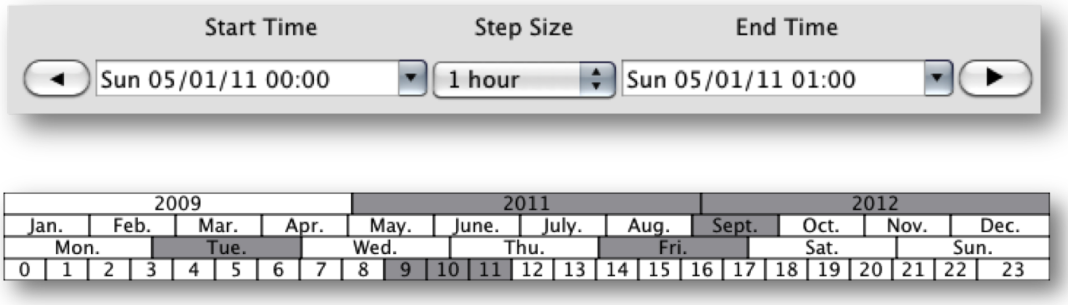


Figure 2.5: Time selection widgets. The regular time selection widget (top) allows the user to define an atomic temporal constraint. The recurrent time selection widget (bottom) allows the user to easily define complex temporal constraints. In this figure, the widget selects for the years of 2011 and 2012, the period between 9AM and noon in all Tuesdays and Fridays in the month of September.

when + what → where. Given a set of objects and a time or set of times, these queries return the location of the objects. This can be achieved by combining time and attribute constraints.

where + what → when. These queries return a time or a set of times when given object or set of objects occupied a given location or set of locations. They can be constructed by combining spatial and attribute constraints.

Although our initial goal was to support these three classes of queries, by having separate constraints, our model is able to express other types of queries, including *when → what + where*, *where → when + what*, and *what → where + when*, by simply defining a single type of constraint.

2.4 The System

In this section, we describe the system we built to support the interactive analysis of the taxi data. It combines the visual query model described above with other visual operations and representations to cater to the requirements set forth in Section 2.2.

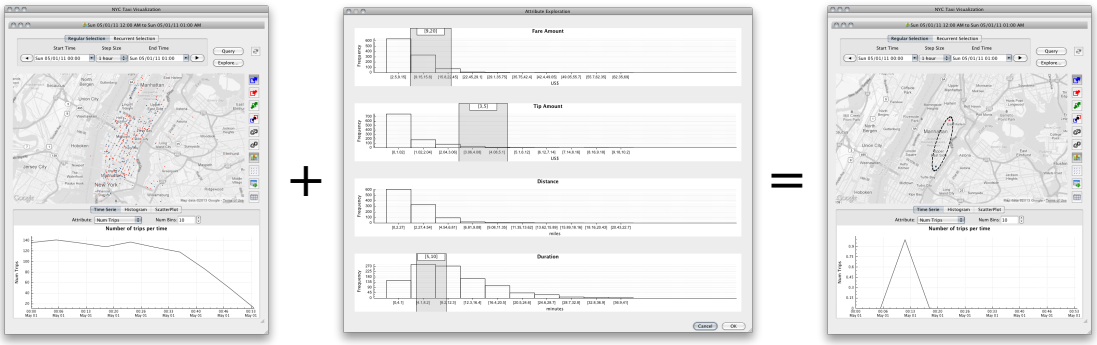


Figure 2.6: Attribute constraints are specified (in the middle) to refine the query on the left. They are shown as dark gray strips on the histogram. The result of the refined query is shown on the right and consists of a single trip.

2.4.1 User Interface Components

The main view of our system is shown in Figure 2.4. Its components and their role in the system are described below.

Map. Maps serve different purposes in our system. They provide a canvas for displaying query results, for users to specify spatial constraints and compose/refine queries.

Time selection widget. This widget allows users to specify temporal constraints. As shown in Figure 2.5, two widgets are available which we describe in Section 2.4.2.

Data summary view. The information associated with the results of a query can be visualized using different representations within the data summary view. For example, this view can display the selected trips as time series, histograms, and scatter plots over different attributes. Since our query model supports multiple sub-queries in a view (represented by different colors), visualization filters can distinguish their results. For example, plots can be generated where each line corresponds to one sub-query.

Tool bar. Several operations are supported through the tool bar. The first 3 buttons (from top to bottom) allow users to specify whether their queries should consider pickups, dropoff, or both. The fourth button supports the creation of directional queries. The group and ungroup (fifth and sixth) buttons provide a simple mechanism for users to combine (and split) both region-based and directional queries. The system can also export query results as a CSV file, which could then be analyzed using other tools. Lastly, the attribute exploration button provides a

visual mechanism for users to define attribute constraints.

Multiple Coordinated Views. The view in Figure 2.4 is used to specify a set of queries that share both time constraints and attribute constraints, but with different spatial constraints. By using multiple views the user can specify other queries with distinct time and attribute constraints (see Figure 2.2). To enable comparison, these views can be synchronized to show the same spatial region and to synchronize the scale of the attribute summaries.

2.4.2 Visual Query Specification

Spatial constraints are specified by polygons and arrows on the map view. These are created either by brushing or by selecting predefined polygons corresponding to NYC's neighborhoods, zip codes, and boroughs. In order to create a single-region spatial constraint, the user first chooses which parameter is associated with the selection to be created by selecting either start/end constraint button (via the toolbar) and through brushing, the user then creates an atomic spatial constraint. Selected regions can be moved, edited, and deleted. The user can also link two regions to form a directional constraint (Figure 2.4). Atomic spatial constraints can be grouped to form complex ones. This is achieved by first selecting the regions and arrows to be grouped and then pressing on the merge button in the toolbar (see the blue query in Figure 2.4).

Temporal constraints are specified using the time selection widgets: *Regular Selection* and *Recurrent Selection* (Figure 2.5). In the Regular Selection widget, the user defines an atomic temporal constraint by assigning the values of the start time and end time fields. Using the Recurrent Selection widget, the user can specify complex constraints that cover different temporal scales, by selecting an arbitrary combination of years, months, days of the week, and hours. This widget is similar to the *time wheel* in the TEMPEST system [45].

Attribute constraints are defined through the attribute selection view, which is accessed through the exploration button in the toolbar. As illustrated in Figure 2.6, this view shows histograms that summarize the attribute values for the trips in the result set of the query shown on the left. By brushing the desired values or value ranges (the dark gray rectangles), attribute constraints are derived and added to the original query. The result of the refined query is shown on the right.

2.4.3 Query Result Visualization

Rendering Trips on a Map. The spatial component of the query results is visualized in the map view. Figure 2.7 illustrates alternative visualizations. A basic visual representation for this kind of data is a *point cloud*, where each trip is represented by a pair of points denoting their pickup and dropoff locations. The two points of a pair are differentiated by color: blue is for pickup and red is for dropoff. For a small number of trips, this visualization can quickly give us a sense of how the taxi activity is distributed throughout the city. However, as the number of trips increases, it gets cluttered very quickly, as shown in Figure 2.4.3(a). This figure contains the points corresponding to all taxi trips in one week. The point cloud covers almost the entire map, making it hard for users to discern what is happening.

To address this issue, our we applied a set of techniques to provide alternative visualizations to the user. First, as shown in Figure 2.4.3(b), we employ an adaptive *level-of-detail (LOD)* strategy to reduce the number of points rendered. Our LOD strategy works by first sorting all the points based on their spatial coordinates, i.e.their distance from the equator and prime meridian. We then build a binary tree on top of the sorted points and perform an in-order traversal to sort them again based on the visiting order. This is equivalent to building hierarchical indexing for regular grids on a Z-order curves [101]. In the end, all the points are arranged linearly in such a way that the first n elements are also a hierarchical subsampling of size n of the original point cloud. During user interaction, n will scale proportionally to the map zoom level with $n = 1e6$ at the finest level. This is also the maximum number of points that our application would display even if the actual number of matched records is higher.

Second, our system supports density summary visualizations or *heat maps* (see Figure 2.4.3(c)) that can be used to show the distribution of pickups and/or dropoffs in an area. The tool buttons on the right of the user interface may also be used for selecting which location attribute (pickup, dropoff or both) is being used for constructing the heat maps. For example, if both pickup and dropoff locations are selected, both pickup and dropoff locations would be used for aggregation on each pixel of the heat map. Such heat map can help answer questions such as “How often do taxis travel to a particular neighborhood?”. Darker locations on a heat

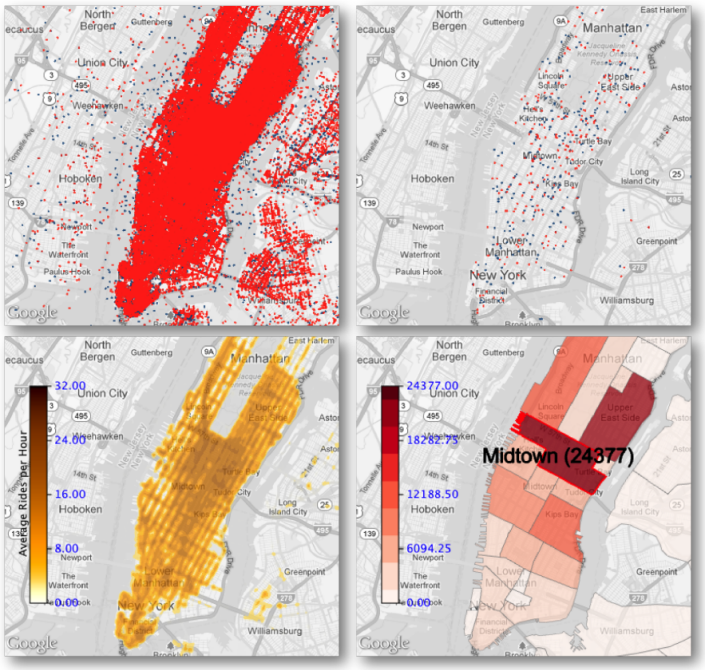


Figure 2.7: Different spatial visualizations of taxi trips for the week 05/01/2011 to 05/07/2011. In (a) all the trips are rendered To reduce the clutter, TaxiVis uses both LOD (b) and density heat maps (c). The system also supports heat maps based on well-defined regions such as neighborhoods (d). For the latter, summary information about the region can be overlaid on the map, for example, hovering the mouse over a neighborhood highlights it and shows the number of trips.

map indicate a higher level of activity in an area. Combined with the point cloud LOD, this is a powerful tool to quickly summarize the data.

Finally, we have also generalized the concepts of heat maps to apply to grid maps in our system. A grid map is a set of cells where their geometries and visual representations can be customized by the users. An example of this is a grid map of zip codes or neighborhoods in NYC showing the number of taxi pickups (see Figure 2.4.3(d)). A heat map may also be considered as a grid map where its cells are points on a regular grid and its visual representations are just spherical gradient textures.

Visualizing and Interacting with Trip Data. Besides displaying query results on a map, filters can be applied to the results to derive different visual representa-

Figure 2.8: Comparison of taxi pickups (left) and dropoff (right) in different neighborhoods over the first week of May 2011. The plots show that Midtown and the Upper East side are the most active areas. But over the weekend, there is an increased number of dropoffs in Downtown. The figure also highlights the fact the Harlem is underserved by taxis.

tions. In our current implementation, we provide support for visual representations that are suitable for the types of attributes associated with the trips. For example, time series, histograms, and scatter plots (see Figure 2.4, Figure 2.2, and Figure 2.6). In addition, as we discussed in Section 2.4.2, these visualizations can be *active* and serve as a means to further refine queries.

2.4.4 Storage Manager

Supporting interactivity is an important requirement of our approach. Thus, performance was a key factor in the design of our system. We have experimented with several designs for data storage that can be run interactively on a single machine. In particular, we have evaluated two traditional database management systems, PostgreSQL and SQLite, with the latter being used for in-memory storage. Though both systems provide extensions for spatial queries, their query performance is not suitable for interactivity, not to mention the fact that both take a considerable amount of time to build the spatial indices. For instance, SQLite took 52 hours just to build the indices for a single year of data. Moreover, a single atomic spatiotemporal query could take from seconds to tens of seconds to complete, while complex ones such as those specified by the recurrent time selection widget, can take minutes. Another issue with these database systems is their large memory footprint. In our experiments, PostgreSQL and SQLite used more than 200GB and 100GB of RAM (in memory setup for SQLite), respectively. We deemed them

Table 2.1: Summary of experiments with data storage strategies.

	SQLite	PostgreSQL	Ours
Storage Space	100GB	200GB	30GB
Building Indices Time	52h	13h	28m
1k-query	8s	3s	0.2s
100k-query	85s	24s	2s

unsuitable for our interactive system because high memory usage would lead to more paging to disk.

In order to address these issues, we have built a light-weight database variant that allows fast queries on all attributes including spatiotemporal constraints. Our implementation is based on a space-partitioning data structure, k-d tree [37], that treats each taxi trip as a point in a k-dimensional space. In our implementation, points are only stored in leaves. Our code takes only 30 minutes to build the indices for the full 3 years of data and uses only 30GB of disk space. At run-time, the whole data structure, including the data points, are mapped to the system virtual memory, thus, it may operate in-core or out-of-core adaptively, depending on the available resources. In our tests, compared to the database systems mentioned above, our system memory usage is considerably smaller, mostly staying at hundreds of megabytes relatively to the amount of data being explored. This design has performed within the bounds of the needs of our interactive system, and queries are significantly faster. In Table 2.1, we summarize the results obtained in our our experiments where 1k-query and 100k-query refer to queries returning approximately 1000 and 100,000 trips respectively.

2.4.5 Rendering Considerations

The performance of the map view is also extremely important in delivering a good user experience. Thus, selecting a map rendering system with both flexibility and efficiency was a top priority in our design. Different options are available: (1) web-based engine provided by online map services such as Google Maps, Bing Maps or OpenStreetMap; (2) 2D desktop-based engines for rendering map tiles from OpenStreetMap such as KDE's Marble. Since web-based rendering engines do not guarantee a consistent graphics acceleration across web browsers and hardware, it

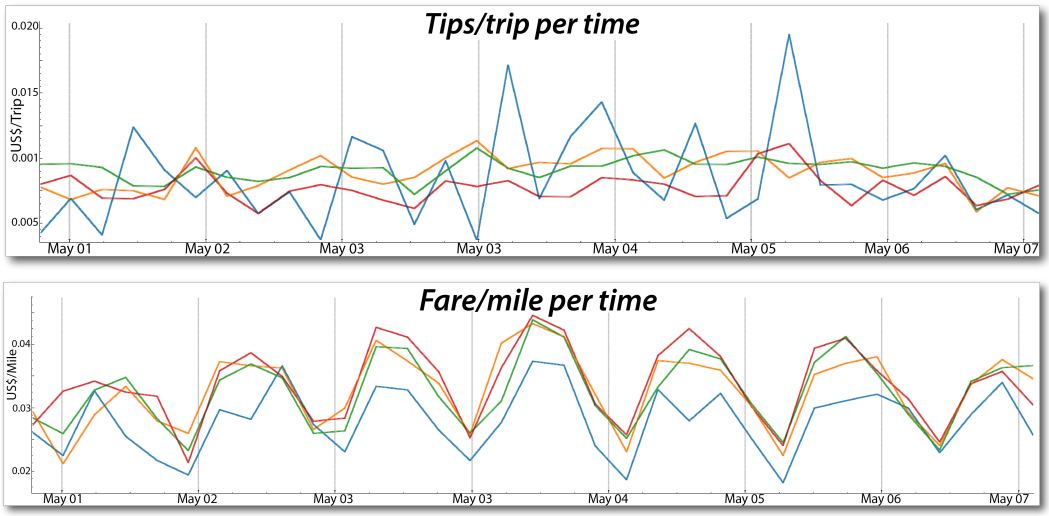


Figure 2.9: Comparing tips per trip and fare per miles for trips originating in different neighborhoods in the period of 05/01/2011 - 05/07/2011.

would hinder some of our visualizations such as the ability to build heat maps on the fly using Frame Buffer Objects or performing trip animations with OpenGL shaders. Moreover, effectively displaying a large amount of data with the web-based map API is still a major challenge including where to host the data and how to render them efficiently. On the other hand, though available systems in the option (2) solve the compatibility issue and data transfer, they only support a specific 2D rendering engine. For example, in the current KDE's Marble, rendering has to be done through the `QPainter` object of Qt; OpenGL is not yet supported. Unfortunately, the use of OpenGL is required for many of our rendering layers. Our solution was a combination of (1) and (2): we embed a web browser as our bottom layer for rendering maps and place other native visualizations on top of it. In our application, we use Qt and promote the `QGraphicsWebView` as our embedded layer. This widget is placed inside an OpenGL canvas of a `QGraphicsView` widget, thus, making it possible for other layers to be compatible with both Qt's `QPainter` and OpenGL native graphics. All geospatial transformations are done in a thin layer above the map view. It should be noted that the web-based component is only used for displaying maps, all other rendering is done in OpenGL to maximize the system performance.

2.5 Case Studies

In this section, we present case studies that illustrate both the power and simplicity of our model and system.

2.5.1 Investigating Taxi Activity in Different Regions

While analyzing taxi service in a city, it is useful to compare different geographical areas. In TaxiVis, users can select regions at different levels of granularity: through free selection, by zip code, and neighborhoods. Figure 2.8 shows the how pickups and dropoffs vary over a period of one week in four different areas. Here, we made use of grouping to analyze the behavior of combined neighborhoods. For example, we grouped East, West and Greenwich Village (shown in green) as well as Harlem and East Harlem (blue). By far, Midtown (orange) is where most activity happens during the week days, followed by the Upper East Side (red). During the weekends, the situation changes and we see greater activity in Downtown. Note the increase in the number of trips that starts to happen on Thursday (May 5), with big peak for pickups on Friday (May 6) in the evening—this indicates that the nightlife on weekends is very lively in Downtown.

This one-week overview provides an accurate overview of city life, where people go and when. It also highlights social inequalities. People who live in Harlem have long complained about the lack of taxi service in their neighborhood. The plot clearly shows that their discontent is well justified. There is over one order of magnitude difference in the number of trips to/from Harlem compared to other more affluent neighborhoods. The heat map also shows that while people take taxis to Harlem, there are barely any pickups there. Exploring other parameters associated with the trips we found one surprising fact: the tips per trip originating in Harlem are higher than for the other neighborhoods (see Figure 2.9). Further analysis also showed that the fare per mile is lower for Harlem, and thus, there is less economic incentive for taxis to be in that area. The higher tips may be a means to reward drivers that go to Harlem.

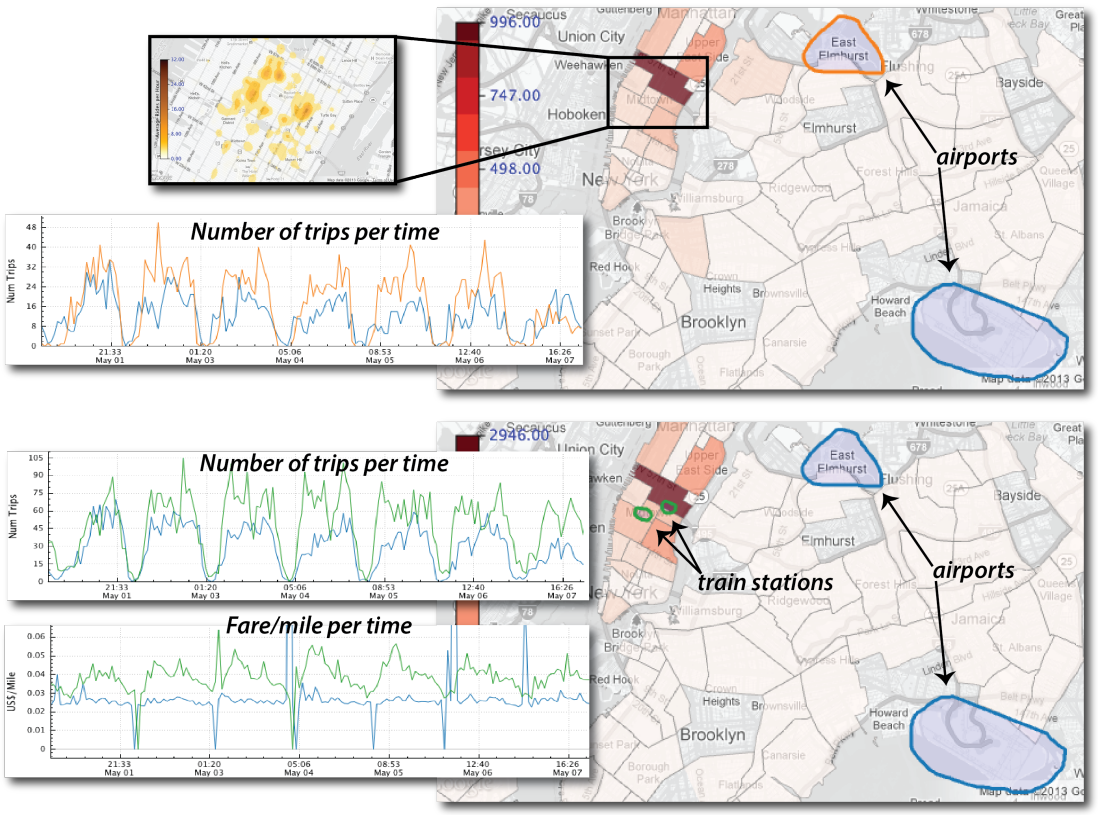


Figure 2.10: Comparing movement across NYC transportation hubs. On the top, we examine trips starting at the two major airports in NYC: JFK and La Guardia. In the bottom, we refine the query to compare trips starting at the airports with trips starting at the train stations, Penn Station and Grand Central.

2.5.2 Exploring Movement: Transportation Hubs

Airports and major train stations (i.e. Penn Station and Grand Central) are key transportation hubs in NYC. By analyzing taxi movement to and from these locations, we can obtain insights into how people move into and out of the city. To compare the number of trips originating at JFK and La Guardia, we select the regions in their vicinity and examine a 1-week period (05/01/2011 through 05/07/2011). As the plot in the top of Figure 2.10 shows, there are more pickups at La Guardia than at JFK on most days. Another interesting question is where passengers go. The choropleth (Figure 2.10 top) that highlights NYC neighborhoods, shows that most people go to Midtown (the darkest region), followed by the Upper West Side.

By hovering the mouse over a neighborhood, the system displays the exact number of trips ending in that neighborhood. We can also obtain more fine-grained information about the exact dropoff locations—the popular destinations, using a heat map.

In order to study the movement patterns for airports and train stations, we can group them (Figure 2.10 bottom) . We select the regions around Penn Station and Grand Central, and group them using the Group/Ungroup button (note the two green outlines); we also group the trips that start at the airports (blue outline). Immediately, the plot is updated to show the number of pickups in the two regions. Note that there are many more pickups around the train stations. Another interesting observation is that the number of trips originating at the train stations remains roughly constant from Monday through Thursday, and starts to decrease on Friday, hitting a low on Saturday. This reflects the behavior of many commuters who go to the City during the week, but not on weekends. Note that, while in this example we have focused on pickups, i.e., people arriving, it is easy to also study dropoffs. Starting from the map view shown in Figure 2.10, we can simply select the airport and train regions (by double-clicking on them), and then click on the "Dropoff" button.

Using the summary view, we can further explore features of the selected trips. For example, by examining the average cost of trip per mile, we can see that it is higher within Manhattan. This provides an incentive for taxi companies to stay within Manhattan and avoid trips to the airport. Note that while it is illegal for taxis to reject rides, this is a common practice when the destination is JFK.[2] This problem is accentuated during rush hour on weekdays, when trips take much longer (see Figure 2.2) and lead to a potential reduction in revenue.

2.5.3 Studying Behavior over Time

Taxi Demand Patterns. Studying how taxi demand varies over time can be useful to understand city dynamics. For taxi companies, this information can help in decision making, both to schedule driver shifts and maximize profits. To simplify the process of comparing multiple times slices, TaxiVis provides a *time*

[2]http://cityroom.blogs.nytimes.com/2011/02/24/taxi-panel-focuses-on-destination-discrimination.

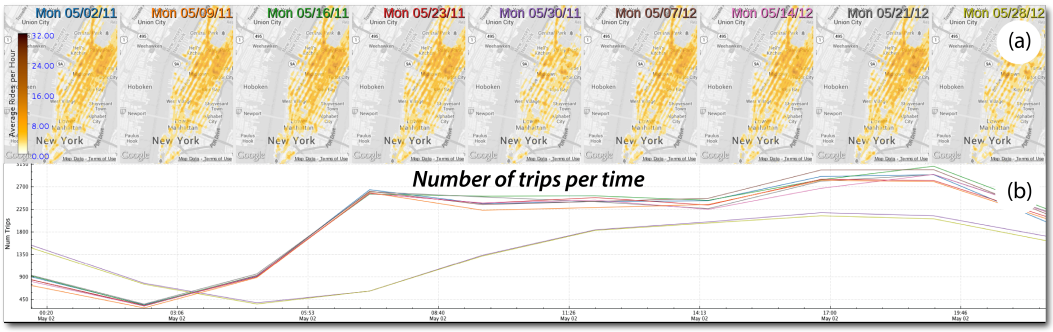


Figure 2.11: Time exploration. (a) shows activity for all Mondays in May 2011 and May 2012. Two such Mondays stand out: 05/30/11 and 05/28/12. Examining the summary plot in (b), we see that the number of trips in these two days is significantly lower than the on other Mondays.

space exploration mechanism. The user first selects the time slices of interest. This can be done using the time selection widgets (Figure 2.5). In the regular selection mode, the slices are selected by specifying a time range, a step size (e.g.an hour, a day, a week), and the number of steps. In the recurrent selection mode, the list of time ranges is already expressed and generated by the widget. For example, by selecting 2011, May and Sunday, 5 times ranges are returned–each corresponding to a Sunday in the month of May, 2011. Given a list of time ranges, the result of a time space exploration is a multi-view visualization displaying one map per time interval, and a data summary view that aggregates the results for the time intervals. Each map view and plot line is associated with a color assigned to its time range. This is illustrated in Figure 2.11. Here, we examined all Mondays in May 2011 and May 2012. The number of trips for the two years is very similar, including the significant drop on Memorial Day. The latter indicates that the number of taxis on the streets could potentially be reduced on the holiday.

Hurricanes Sandy and Irene. The taxi data can also give insights into the effects of major events. We used the time space exploration to study taxi activity during the the week of Hurricane Sandy. Figure 2.12 shows the taxi trips starting on Sunday, the day before the hurricane, through Saturday. The heat maps closely reflect the extent of the disruption caused by this event. On Monday, the day the hurricane landed, there was a big drop in the number of trips throughout Manhattan. On Tuesday, life starts to get back to normal in most regions, except

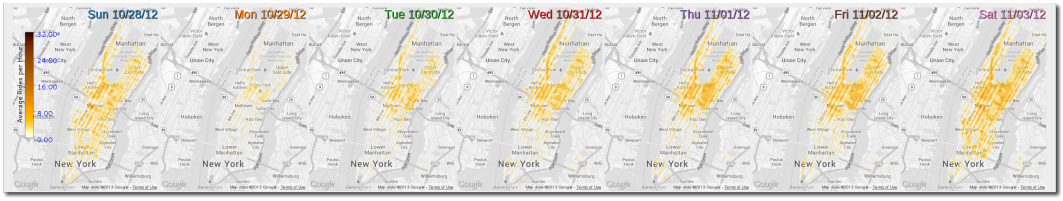


Figure 2.12: Taxi activity in Manhattan during the week of Hurricane Sandy. On the day the hurricane hit, there are very few taxis on the streets throughout Manhattan. On the next four days, activity goes back to normal in all parts of the city, except in lower Manhattan, where there was a power outage which lasted for five days.

for Lower Manhattan, where there are virtually no taxis for five days. This area suffered a major power outage which was only restored on Saturday. We have also studied the period around Hurricane Irene (see Figure 2.13). Note that although activity came back to normal sooner, on the day of the hurricane, there were virtually no cabs: there were only 1076 trips, while on average, there are 500,000. This seems to indicate that although shorter, Irene caused a bigger disruption in Manhattan.

2.6 Discussion

TaxiVis was designed with the intent to explore the taxi trips dataset. However the overall framework is general and can support the exploration of other datasets from different applications such as bike sharing [112], population migration [65], and property owenership [70] to name a few. Also, while our query model and data layer focused on OD data, these can be adapted to support queries on general trajectory data. In fact, similar problems were pursued by the recent works by Krueger et al. [76] and Wang et al. [126].

There are number of future directions for improvement of TaxiVis. One of them is realted to do with the fact that while our visual query model is flexible, the current user interface has some limitations. For instance, there are useful temporal constraints that cannot be expressed with the current time widgets. For example, the recurrent time selection widget (Figure 2.5) can be used to select all Tuesdays in a particular month. However, it is not possible in the current interface to exclude

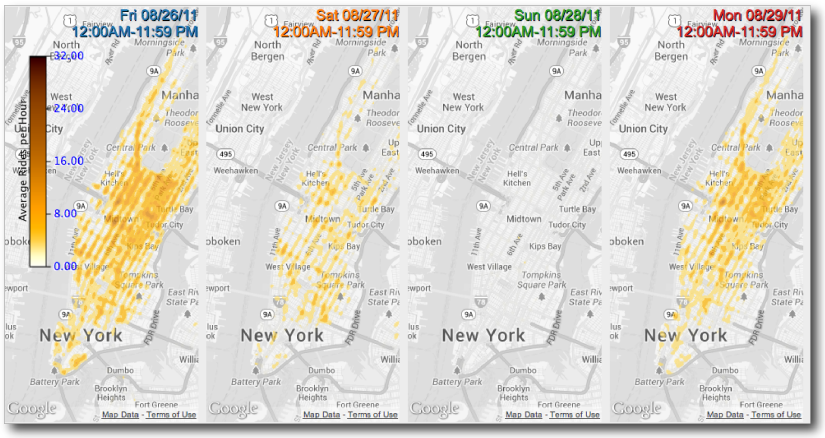


Figure 2.13: Taxi activity in Manhattan during the week of Hurricane Irene.

one of these Tuesdays from the set of temporal constraints. We plan to experiment with alternative and more flexible widgets both for time and attribute constraint specification. We also plan to add support for multiple data sources to help in the analysis. In fact, our current system relies on user knowledge of the city to make inferences about the patterns on taxi data. We intend to add more context information about the city so the user is able to correlate the taxi data with, for example, information about the main business activities in the regions.

2.7 Summary

In this chapter, we presented a new system that supports visual exploration of big origin-destination and spatiotemporal data. A key component of this system is a visual query model that allows users to quickly select data slices and explore them. We have shown that this model attains a good balance between simplicity and expressiveness. Another important contribution of this work is the system design, which not only combines the visual query model with other visualization primitives, but also addresses performance challenges that arise due to the scale of the data. In particular, to support interactivity, we designed an efficient storage manager as well as a rendering subsystem. We have present a series of case studies, using a large data set consisting of over 520 million taxi trips in NYC, which illustrates the capabilities and effectiveness of our system and design decisions.

Chapter 3

Event Guided Exploration of Spatiotemporal Data

Patterns in spatiotemporal data usually occur at multiple spatiotemporal scales. For example, traffic accidents usually have a local influence on traffic (block a road), conventions can happen over many consecutive weekends, and severe weather events may last for a few days and completely change the behavior of a particular location. For example, as can be seen in Figure 2.1, patterns corresponding to high impact weather events like hurricanes Sandy and Irene affect a large extension of space for a long period of time. On the other hand, as shown on the bottom of Figure 2.1, patterns corresponding to the NYC Five Boro Bike Tour only affect a small region in space and during short period of time.

As described in Chapter 2, interactive visualization systems are very powerful tools to perform exploratory data analysis of spatiotemporal data. On the other hand such systems are constrained by the display and data dimensions. This limits the exploration workflow to a few options. In fact, the most common approach is to explore the data in the spatial sense, in which case time is usually fixed, or it can be explored over time, in which case space is usually fixed for comparison purpose. In such setting, the usual approach to analyze this kind of data is to use select portions of the data and use different types of aggregation and produce visual summaries [4, 83]. Due to the large number of possibilities for occurrence of patterns, finding patterns like the Five Boro Bike Tour is difficult.

This leads to a trade-off between the level of aggregation and the number of

data slices to be explored. The use of a coarse (spatial or temporal) aggregation reduces the number of data slices, but it may result in loss of information. Consider the example illustrated in Figure 3.1(a), which shows a heatmap of pickups and drop-offs of NYC taxi trips dataset (Chapter 2.2) on Sunday 1 May 2011 between 8 and 9 am, which corresponds to the NYC Five Boro Bike Tour. Note that there are no taxi pickups and drop-offs along 6^{th} avenue. However, as shown in Figures 3.1(b) and (c), a coarse level of aggregation makes it difficult to identify small or local events. While a finer level of aggregation would avoid these problems, it requires the exploration of a large number of data slices. In order to find patterns such as Figure 3.1(a), current state of the art systems require their users to perform manual (exhaustive) exploration. This is not only time consuming, but for large datasets, it becomes impractical. For example, temporal aggregation of a year's worth of data into a discrete set of hourly intervals results in over 8000 data slices to be explored per year.

As a step towards addressing this challenge, in this chapter, we propose an efficient and scalable technique that automatically discovers events and guides users towards potentially interesting data slices. Event detection is accomplished through the application of topological analysis on a time-varying scalar function derived from the spatiotemporal data (Section 3.4). We use the minima and maxima of a given function to represent the events in the data. Intuitively, a minimum (maximum) captures a feature corresponding to a valley (peak) of the data. For example, the lack of taxis along 6^{th} avenue during the bike tour event forms a local minimum and is therefore captured using our technique. The use of topology also allows for events having an arbitrary spatial structure. In order to support a potentially large number of events, we design an indexing scheme that groups similar patterns across time slices, thereby allowing for identification of not only periodic events (hourly, daily, and weekly events), but also of events with varying frequency (regular and irregular). Thus, unlike previous approaches that impose a rigid definition of what constitutes an event [8], our technique is flexible and able to capture a wide range of spatiotemporal events. The index further allows users to efficiently search for occurrences of similar patterns. Compared to techniques based on statistical analysis that support different kinds of events, our approach is computationally efficient and scales to large datasets. We also describe a visual

interface (Section 3.5) designed to aid in event guided exploration of spatiotemporal data that integrates the event detection and indexing techniques. The interface allows users to interactively query and visualize interesting patterns in the data. Finally, we show the effectiveness of our approach on two spatiotemporal datasets: the NYC taxi trip records described in Chapter 2.2, and subway service, published by the Metropolitan Transit Authority (MTA). We presented results of preliminary analyses to experts at both the TLC and MTA. While they offered insights for some of the events we found, they were surprised by others which indicated potential problems they had to investigate (see Section 3.6). This initial feedback suggests our technique is effective and has the potential to help in exploratory analyses of large, spatiotemporal data.

3.1 Related Work

The problem of event detection in the context of spatiotemporal data has been recognized and addressed in previous works. We discuss approaches to event detection in three categories: those that use computational topology for feature tracking, those related to visual analytics, and techniques for event detection from statistics and machine learning. Note that there is no universal definition of an event [8]. It is thus difficult to quantitatively compare different techniques. Here, we present a qualitative comparison where we consider flexibility to support different event types, efficiency, and scalability.

Computational topology. Computational topology has been used to identify and track features of spatiotemporal data. Laney et al. [79] and Bremer et al. [24] used the Morse decomposition of a scalar field to identify features of the input and track these features across time using geometric properties of the features. Pascucci et al. [103] identified features, which correspond to burning cells during turbulent combustion, using merge trees and tracked them by computing the overlap of the features. Widanagamaachchi et al. [129] extended this technique and designed a framework to explore time-varying data. Kasten et al. [73] mapped critical points of the input scalar function across time steps and created a merge graph that is used to track unsteady flow fields. Doraiswamy et al. [39] identified cloud systems in each time step using the join and split tree, and tracked them

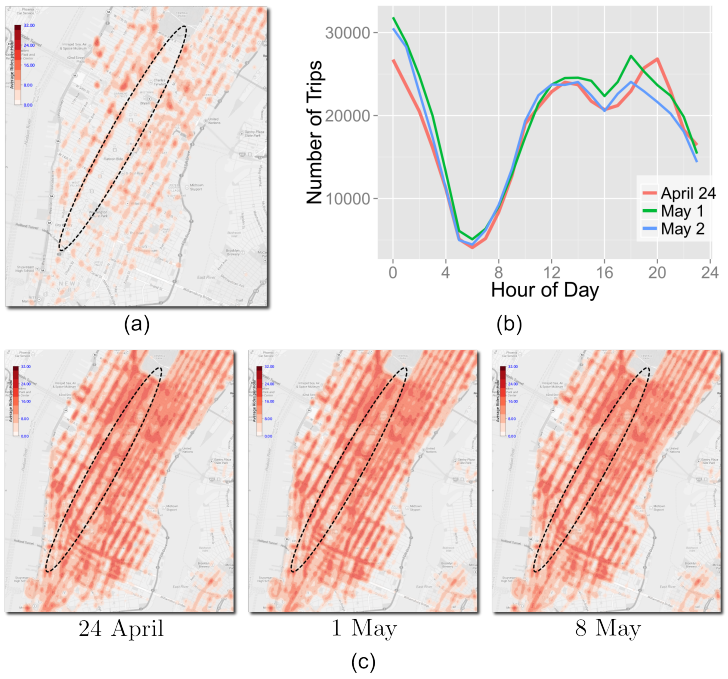
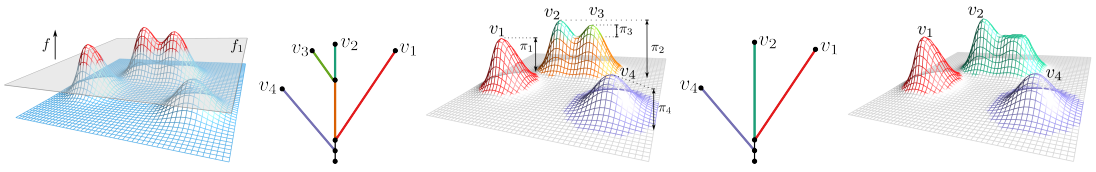


Figure 3.1: It is difficult to identify short irregular patterns when the data is aggregated over either space or time. **(a)** A heatmap of taxi locations in Manhattan on 1 May 2011 between 8 am and 9 am. Note that the path of the bike tour contains no taxis. **(b)** The time series plots compare the number of trips that occurred in Manhattan on three Sundays in 2011: 24 April, 1 May, 8 May. It is difficult to distinguish between the three Sundays using just the number of trips, even though an entire stretch of streets are blocked to traffic on May 1st. **(c)** The trips are aggregated over time and displayed as a heat map for the three Sundays. Note that the path of the bike tour (highlighted) looks similar in all the heat maps.

across time using optical flow. These methods are only interested in movement of features across consecutive time steps, which is accomplished by looking at adjacent time-slices. Such tracking cannot be applied to our problem, since we need to identify features that have similar behavior but are spread across non-adjacent time slices.

Visual Analytics for Spatiotemporal Data. Scholz et al. [111] proposed a technique to analyze hotspots using taxi data in San Francisco. They pre-defined regions of interest, modeled the taxi activity in each census tract in these regions, and used the model to predict the life cycle of hotspots. By pre-defining regions of interest and using artificial boundaries such as census tracts, this approach



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

Figure 3.2: Topology of scalar functions. **(a)** The height function f defined on a graph. The super-level set at f_1 is the set of all points above the highlighted plane and consists of *two* components (colored red). **(b)** Join tree of f. **(c)** The features of the input are defined based on the edges of the join tree. The labeled peaks denote the set of maxima. The features are colored the same as the corresponding edges in (b). π_i denotes the persistence of maximum v_i. Intuitively, the persistence of each maximum is equal to the height the corresponding peak. **(d)** The simplified join tree obtained from removing the maximum v_3. **(e)** The resulting smoothed function.

is likely to miss events which are of arbitrary shapes and happen at different granularities (see Figure 3.1). Maciejewski et al. [87] used kernel density estimation to model hotspots in spatial distributions along with time series analysis to detect anomalous hotspots. Andrienko et al. [6, 7, 8, 10] proposed visual analytics procedures to determine places of interest based on high-frequency events that also have high density of occurrence. The user first applies a set of filters to define features of potentially interesting events. Then, the points with those features are clustered to find interesting recurring locations. Unlike these techniques, which have a rigid definition of events (e.g., high density and recurrent occurrence), our technique is able to capture a wide spectrum of events, both based on density (low and high) and frequency of occurrence at different time scales.

Statistical Event Detection in Spatiotemporal Data. The problem of event detection has also been studied by the statistics and machine learning communities [69, 78, 89, 94, 95, 125]. The area is closely related to spatial scan statistics [77] and anomaly detection [30], albeit exploiting the spatiotemporal nature of the domain and focusing on the discovery of "interesting" contiguous regions in space and time. Previous work examined multiple overlapping spatiotemporal subsets of data and identified significant deviations from a *baseline*, e.g., an expectation over time, via a frequentist likelihood ratio test or a Bayesian posterior probability

distribution over events [95]. However, the majority of the literature has focused on either purely spatial data or has accounted for temporal variations and effects via simplistic approaches such as exponentially weighted linear regression or data partitioning based on day-of-week or season. Furthermore, the time complexity for these approaches is exponential $O(2^N)$ in the number of *pre-defined* space-time partitions, with polynomial approximations (non-exhaustive search) available only for the frequentist hypothesis tests that require extensive randomization [89] for p-value estimation. In contrast, our technique allows for detection of events that can have arbitrary spatial geometry, different time intervals, scales up with polynomial time complexity of $O(n^2)$ in the number of events, and enables user exploration of urban datasets via efficient event querying. The latter ensures flexibility of the technique across applications and domains, where users can define and query interesting events based on prior knowledge and different spatiotemporal properties of the data.

3.2 Background

The topological representation of large datasets provides an abstract and compact global view that captures different features and leads to enhanced and easier analysis across applications [57, 102]. In this section, we briefly introduce concepts from computational topology that serve as the basis of the proposed technique. Comprehensive discussions on this topic can be found in [42, 67, 91].

Scalar functions. A *scalar function* maps points in a spatial domain to real values. The spatial domain of interest for the techniques developed in this chapter is a graph G representing a particular aspect of an urban environment like the road network. The scalar function is represented using the graph G, together with a piecewise linear (PL) function $f : G \to \mathbb{R}$. The function is defined on the vertices of the graph and linearly interpolated within each edge. Figure 3.2(a) shows an example of a scalar function defined on a graph representing a terrain. The function value at each point on this graph is equal to the point's y-coordinate. A *super-level set* of a real value a is defined as the pre-image of the interval $[a, +\infty)$. Similarly, the *sub-level set* of a is the pre-image of the interval $(-\infty, a]$. Figure 3.2(a) highlights the super-level set at function value f_1.

Critical points. The *critical points* of a smooth real-valued function are exactly where the gradient becomes zero. Points that are not critical are *regular*. We are interested in the evolution of super-level sets (sub-level sets) against decreasing (increasing) function value. Topological changes occur at critical points, whereas topology of the super-level set (sub-level set) is preserved across regular points [91].

The critical points of a PL function are always located at vertices of the mesh [19, 43]. Consider a sweep of the function f in decreasing order of function value. The nature of topological change to the super-level sets of f when the sweep passes a vertex determines the type of that vertex. A new super-level set component is created at a *maximum*, while two super-level set components merge into one at a *join saddle*. Similarly, during the sweep of the input in increasing order of function value, a new sub-level set component is created at a *minimum*, while two sub-level set components merge into one at a *split saddle*. The scalar function shown in Figure 3.2(a) has 4 maxima (see Figure 3.2(c)).

Different types of critical points of a scalar function capture different types of features. In particular, a maximum captures a peak of the function, where the function value is higher than its neighborhood. Similarly, a minimum captures a valley of the function. The set of peaks and valleys are the natural features of a given function, and are therefore of interest in this work. We use the set of minima and maxima to represent features (events) of the given data.

Topological persistence. Consider the sweep of the input function f in decreasing order of function value. As mentioned above, the topology of the super-level sets change when this sweep encounters a critical point. A critical point is called a creator if a new component is created, and a destroyer otherwise. It turns out that one can pair up each creator v_c uniquely with a destroyer v_d that destroys the component created at v_c. The persistence value of v_c is defined as $\pi_c = f(v_c) - f(v_d)$, which is intuitively the lifetime of the feature created at v_c, and is thus a measure of the importance of v_c. The traditional persistence of the global maximum is equal to ∞ since there is no pairing destroyer for that maximum. In our technique, we use the notion of extended persistence [2] which pairs the global maximum with the global minimum. For the height function shown in Figure 3.2(a), the persistence of each feature corresponds to the height of the corresponding peak, highlighted in Figure 3.2(c). Given an input domain of size n, the persistence of the set of

minima and maxima can be computed efficiently in $O(n \log n)$ time [41, 44].

Join tree and split tree. The join tree and split tree abstracts the topology of a scalar function f, and are useful for extracting and representing features of f (the regions corresponding to maxima and minima). The *join tree* tracks the changes in the connectivity of super-level sets of an input function f with decreasing function value. Figure 3.2(b) shows the join tree corresponding to the function shown in Figure 3.2(a). The *split tree* of f is defined similarly, and tracks the connectivity of the sub-level sets of f with increasing function value. Nodes of the join tree and split tree correspond to the set of critical points of f.

Regular points are often inserted into the join/split tree as degree-2 nodes to obtain an *augmented join tree/augmented split tree*. We use the subgraph of the input mesh induced by the regular vertices that are part of an edge in the augmented join/split tree to represent the feature corresponding to the maximum/minimum. The colors of the different features of the function in Figure 3.2(c) correspond to the colors of the edges of the join tree shown in Figure 3.2(b). Optimal algorithms exists to compute join and split trees of a PL function [28, 33, 85, 100].

Simplification. The input is often simplified to remove noise. This is accomplished by removing low persistent features. The join and split trees provide an efficient mechanism to perform this simplification [29]. Removing an edge in the join/split tree corresponds to smoothing the corresponding region of the function. For example, consider the feature represented by v_3 in Figure 3.2 which has low persistence. Simplifying this feature corresponds to smoothing the function in order to remove the maximum v_3. The simplified join tree is shown in Figure 3.2(d), while function resulting from this simplification is illustrated in Figure 3.2(e). Features can also be simplified based on geometric measures like hyper-volume [29].

3.3 Spatiotemporal Data and Scalar Functions

We model urban data as a time-varying scalar function f defined on a graph G, where the temporal dimension is represented as a set of discrete time steps. In this section, we describe two datasets that we use throughout the chapter and the scalar functions derived from them.

3.3.1 NYC Taxi Trips Data

We use the NYC taxi trips described in Chapter 2.2. In our experiments, we use data of taxi trips that took place in Manhattan during 2011 and 2012. The data set is first divided into a set of hourly intervals. Note that this time interval is not fixed, and can be changed depending on the application. Manhattan is represented using the graph corresponding to its road network. Each node of this graph represents an intersection of two or more streets, and each edge corresponds to a street segment.

Analysts at the TLC and at the Department of Transportation (DoT) are interested in identifying traffic-related events that have led to road closures as well as taxi hot spots (see Section 3.6.1). To capture these events, we define the scalar function for an hourly interval at each node of this graph as the density of taxis within a small circular region surrounding the corresponding location. The radius of the circular region is approximately equal to half the distance between two avenues in Manhattan. The density is then computed as the Gaussian weighted sum of the trips within this neighborhood, where the weights correspond to the trip's distance from the node. This ensures that trips closer to a node have a higher contribution to the density compared to a trip that is farther away. Recall that the set of minima and maxima are used to represent events in the data. Given a single time step, a minimum of the above function represents a region where the density of taxis is lower than its local neighborhood, implying a relative scarcity of taxis in that region. Similarly, a maximum represents a region where the density of taxis is higher than that of its local neighborhood, implying a relatively high concentration of taxis.

Such a density function can also be used on many other urban datasets such as twitter feeds [119] and GPS traces from mobile devices [63] which have a representation similar to that of the taxi data.

3.3.2 MTA Subway Data

The MTA provides real-time information for the numbered lines of the NYC subway system [92]. These data consist of the time stamps of all the stops for all the trips that happen each day. Engineers at MTA are interested in analyzing data from these feeds to improve operations and scheduling of the subway system. In

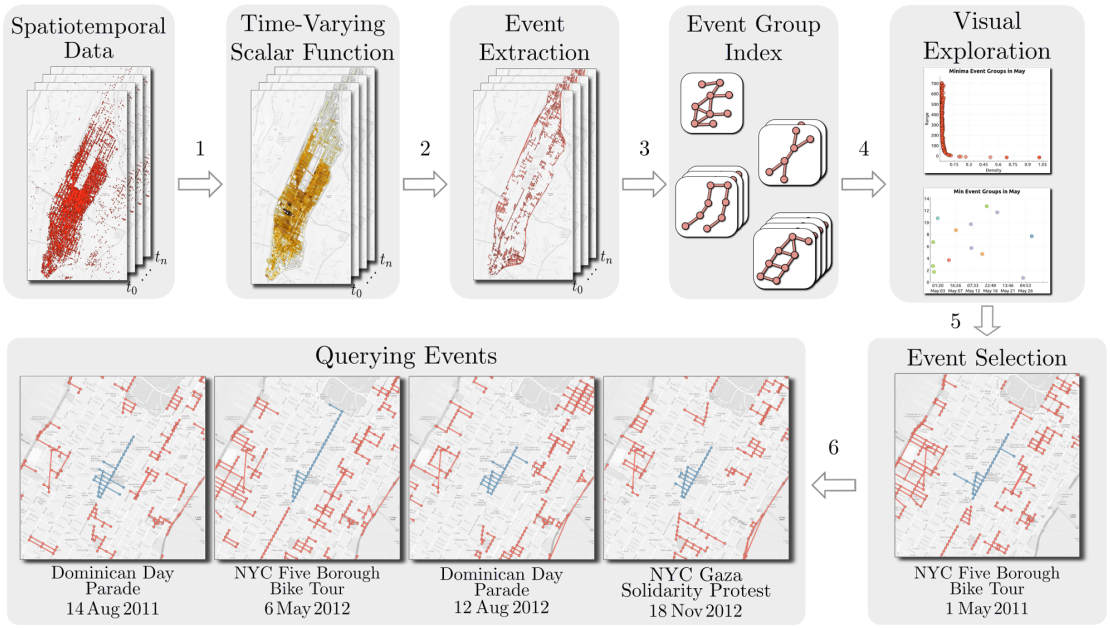


Figure 3.3: Overview of the event guided exploration technique. First, (1) the input data is transformed into a time-varying scalar function. (2) Topological features are computed from the scalar functions to identify the set of events. (3) An event group index is then created from the identified events to support efficient querying over a large number of events. (4) A visual interface guides the user towards interesting events (5) in the data, allowing them to select an event and (6) interactively search for similar events.

particular, they are interested in delays in the schedule of the different trains. In order to capture this for a given train, we compute the scalar function at each station as the average delay of the train at that station. The delay at a station for a given train is the difference between the time the train was scheduled to arrive and the actual time at which it arrived. The underlying graph used to define the scalar function is essentially a simple path representing the route of the train. The nodes of this path corresponds to the different stations along its route. As with the previous data set, the temporal dimension is divided into a set of hourly intervals. The above scalar function is computed for each of the train lines.

3.4 Identifying and Managing Events

Our framework consists of two main steps. First, a set of potential events is computed from the input scalar function – these constitute all features from each time step of f. Then, similar single time-step events are grouped and an index is built that supports efficient queries over a possibly large number of events. The process is illustrated in Figure 3.3 and the details are presented below. We use the NYC taxi data as a running example to illustrate our technique.

3.4.1 Computing Events

First, the split tree of the scalar function f is computed. The set of minima and the regions corresponding to them constitute the set of *minima events*. Since we are mainly interested in the set of "significant" events, simplification of the split tree is performed to prune uninteresting (noisy) minima. We use a small threshold (close to zero) during this simplification process. While persistence captures the importance of a feature only in terms of the scalar function, for the taxi data, we are also interested in capturing the geometric size of a feature. We therefore use hyper-volume as the importance measure. The hyper-volume [29] of a topological feature is defined as the integral of the input scalar function over the corresponding region. This allows features that occupy a large area but have low persistence (depth) to also be considered important. Thus, this simplification retains "deep valleys" as well as "shallow, but large" valleys. Note that using persistence instead of hyper-volume could potentially remove the large shallow valleys during the simplification process. The top-k from the set of minima that remain after simplification and their corresponding regions constitute the set of minima events. Here, k is a user defined parameter. In our experiments, we found that setting $k = 50$ provides a good threshold that is large enough to ensure no significant event is lost. Figure 3.4 shows the scalar function and the associated set of minima events identified for the time step 10-11am on 24 November 2011. This was one of the time steps during which the Macy's Thanksgiving Parade (highlighted in the figure) occurred. The set of *maxima events* are computed similarly using the join tree of the scalar function.

Figure 3.4: Identifying minima events. *Left:* The scalar function corresponding to the time step 10 am-11 am on 24 November 2011 is shown using a heat map. *Right:* The set of minima events identified using the split tree. Each connected component of the colored subgraph corresponds to an event. The event corresponding to Macy's Parade is highlighted.

3.4.2 Event Group Index

Each of the events computed in the previous stage corresponds to a single time step. Multiple such events can be part of a larger *macro event* that spans multiple time steps. For example, the Macy's Parade consisted of a set of events that spanned several hours when the roads were blocked for the parade. To group such events, we first define a notion of similarity between events based on their geometric and topological properties. Similar events within a user-defined time interval are then grouped together to obtain the set of *event groups*. There can potentially be a large number of event groups across different time intervals. To support efficient search over event groups, we define a *key* that is used to index these groups.

3.4.2.1 Similarity Between Events

An event E is formally represented as a pair (R, τ), where R is a subgraph of G denoting the spatial region of E, and τ is a real number that represents the

topological importance of E. τ is the same measure that is used to simplify the join and split trees in the previous step, which for the taxi data is the hyper-volume of E. Consider two events $E_1(R_1, \tau_1)$ and $E_2(R_2, \tau_2)$. We use the *graph distance metric* [26], δ, to measure the *geometric similarity* between R_1 and R_2:

$$\delta(E_1, E_2) = 1 - \frac{|R_1 \bigcap R_2|}{\max(|R_1|, |R_2|)},$$

where $R_1 \bigcap R_2$ denotes the maximum common subgraph between R_1 and R_2, and $|R|$ denotes the number of nodes in R. The *topological similarity* between two events is defined as:

$$T(E_1, E_2) = |\tau_1 - \tau_2|$$

The geometric similarity measures the amount of overlap between two regions, ensuring that similar regions have a significant overlap. The topological similarity on the other hand ensures that the two events are topologically close with respect to the topological importance measure used. Two events E_1 and E_2 are *similar* if $\delta(E_1, E_2) \leq \epsilon_\delta$ and $T(E_1, E_2) \leq \epsilon_\tau$, where, ϵ_δ and ϵ_τ are user-defined thresholds.

3.4.2.2 Event Group and Event Group Key

An *event group* comprises a set of similar events that occur within a given time interval. A brute-force approach to compute event groups from a set of n events would require the computation of similarity between all pairs of events ($\binom{n}{2}$ comparisons). For example, even considering 50 events per hourly time step results in a total of 1200 events per day. Since urban datasets typically contain data for multiple years, computing similarity between all event pairs is not practical.

To avoid a combinatorial explosion, we propose to group events for fixed time intervals. In the experiments described in this chapter, we use a time period equal to one month. We choose this time interval since it provides a good trade-off between efficiency and number of events: there is a sufficient number of events so as to not to miss periodic events, but the number of events in this interval is small enough and does not create a computational bottleneck. Moreover, a time frame of a month provides a natural and easily understandable abstraction for the user to explore event groups.

Given an event group $\Sigma = \{E_1, E_2, \ldots, E_k\}$, we define the *event group key* of Σ

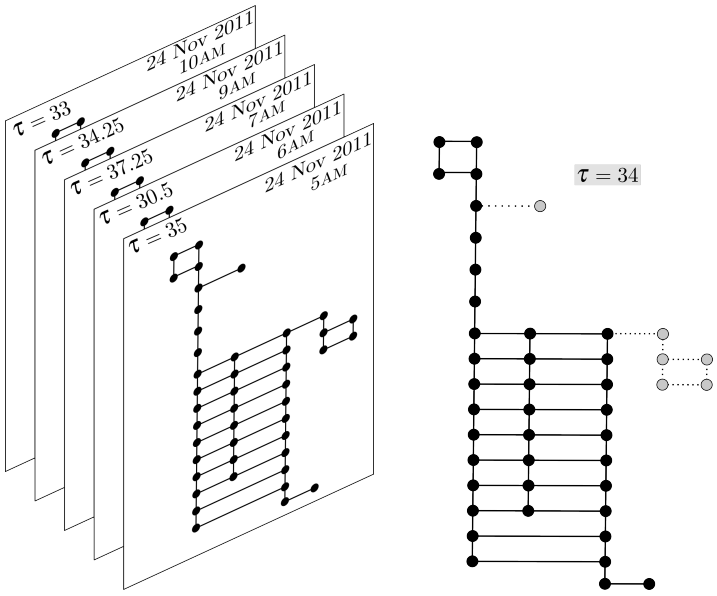
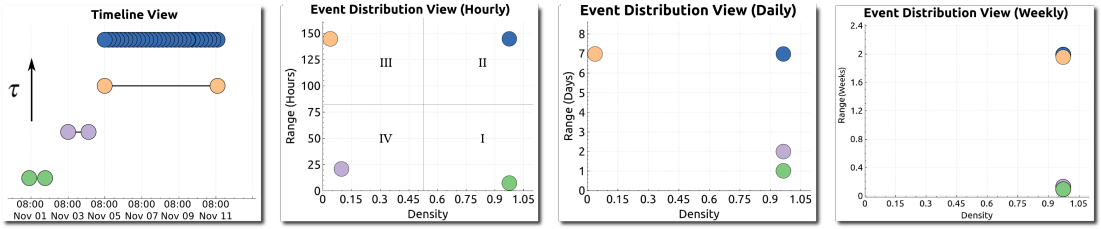


Figure 3.5: Event group index. The event group key of an event group corresponding to Macy's Parade. The black nodes and edges correspond to the maximum common subgraph of regions of all the events in the event group, and is used to represent the region of the event group.

as (R_Σ, τ_Σ), where

$$R_\Sigma = \bigcap_{i \in [1,k]} R_i \quad \text{and} \quad \tau_\Sigma = \sum_{i=1}^{k} \tau_i / k$$

The above definition of the event group key follows directly from the definition of geometric and topological similarity measures. R_Σ is the maximum common subgraph of the geometric regions of all the events in Σ. Since the events in Σ are similar, we can conclude that there is considerable overlap among them due to the similarity condition. Thus, R_Σ provides a good representation for the region where events in Σ occur. τ_Σ captures the topological importance of the Σ as the average of the topological importance of the events in Σ. The definition of event group key also helps in using a consistent definition for the similarity between event groups. Two event groups are similar if their keys satisfy the similarity constraints described earlier.



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

Figure 3.6: The visual exploration interface. **(a)** The time line view showing 4 event groups with different densities and ranges. In this view, event groups are sorted according to their topological importance τ. **(b)** The event group distribution view where the time resolution is an hour. **(c)** The event group distribution view when the time resolution is changed to a day. The daily event (purple) moves to Region I for this time resolution. **(d)** The event group distribution view when the time resolution is changed to a week. The weekly event (orange) moves to Region II for this time resolution.

3.4.2.3 Computing Event Groups

Even when restricting the events to be within a time interval, comparison between all pairs of events is a costly operation. To decrease the number of such comparisons, we relax the condition of similarity between two events: two events are similar if their event groups are similar. Using the relaxed condition, event groups for each time interval are computed as follows. Initially, each event is its own group. The algorithm iterates through the set of event groups to identify similar group pairs. Events are processed in increasing order of their time step. When two similar event groups are found, they are merged into a single group. The algorithm continues until no event groups are similar. Figure 3.5 shows an event group corresponding to the Macy's Thanksgiving Parade along with its event group key.

The quadratic number of comparisons to be performed among the events present in a given time interval is still a computationally expensive operation (approximately 650 million comparisons in the worst case are required per month assuming 50 events per hour). However, utilizing the spatial information of the events, it is possible to drastically reduce the number of comparisons. The spatial region of the input graph is first divided into a set of smaller subregions. An event intersecting a subregion is assigned to that subregion. It is then sufficient to group only events present in

each subregion. Note that an event can be assigned to multiple subregions. These multi-region events can be efficiently handled by maintaining the event groups using the union-find data structure [35].

3.4.2.4 Analysis

Time complexity. Let the graph G have N nodes. Computing the join and split trees of a scalar function defined on G takes $O(N \log N + N\alpha(N))$ time [28], where α is the inverse Ackermann function. Given n events per time interval, computing event groups for that period requires $O(n^2)$ time in the worst case. However, we note that in practice the constant associated with the above bound is small, thus allowing for fast computation. For example, when using 50 events per hour on the taxi data, the average number of events assigned to a subregion of Manhattan was about 1000. Manhattan was divided into a total of 50 subregions, thus amounting to a maximum of 25 million comparisons as opposed to 650 million that is required by a brute-force technique.

Scalability. The set of event groups along with the corresponding keys are stored separately on disk for each time interval. As and when data is obtained for newer time steps, it is easy to update the event group index. Computing the scalar function followed by identifying the events is independent for each time step. If the newer events correspond to an already existing time interval, then they are grouped into the event groups of that interval. If the time interval does not exist, then a new set of event groups are formed for this time interval.

3.4.3 Querying Events

Once an event is discovered, it is often useful to identify *similar* events that may have occurred at different times. Manual search is impractical due to the large number of data slices. For example, consider a case where a particular road block had unintended effects on the traffic in Manhattan. Experts at the DoT are interested in identifying how frequently such road blocks occur, if they are periodic, and if they have the same effects. This will help them design preventive measures and improve decision making.

The event group key enables the efficient evaluation of such similarity queries.

Given a query consisting of a geometric pattern (a sub-graph) together with a real value, we perform a linear search over the set of event groups to identify the set of events similar to the given query. Since the set of event groups for each time interval are stored separately on disk, queries can be executed in an out-of-core manner by sequentially loading event groups from each interval.

3.5 Visual Exploration Interface

In this section we describe an interactive visual exploration interface that uses coordinated views and allows users to browse through the data based on the detected events. Given the input data set, the event group index is first created in a pre-processing step. The data together with the created index is then loaded by the interface and used to support interactive exploration and querying of events in the data. The rest of this section describes the different views and options available in this interface for data exploration.

3.5.1 Map View and the Query Interface

The map view, as the name suggests, provides geographical context to the user. For a given time step, the geometry of the different events in that time step is visualized in the context of a map of the city of interest (see Figure 3.4). This view also allows users to select events of interest to search for similar events.

3.5.2 Event Group Distribution View and Timeline View

The event detection technique can generate a large number of event groups, many of which may be uninteresting. It is thus important to allow users to explore the set of event groups and *guide* them towards potentially interesting events. Events can be broadly classified into two categories – recurring events and sporadic (or one-off) events. To capture these categories, we define two attributes – range and density. The *range* of an event group is defined as the amount of time between the first and the last event in that group based on their time steps. Its *density* is defined as the number of events of that group that happen per time unit. It

measures the time frequency of the events within the group. The resolution of time determines the time unit. Our tool supports three units – hours, days, and weeks.

A combination of these two attributes enables the classification of an event group. We provide an event group distribution view that uses a scatter plot to visualize the event groups, where the axes correspond to the two attributes. Figure 3.6(b) illustrates an event group distribution view consisting of 4 event groups. As illustrated in the figure, the different combinations of the two attribute values roughly divides the event group distribution space into four regions:

Region I Event groups in this region have a low range, but high density. This indicates rare occurrence of such events, and can be used to identify irregular patterns of the data. The green event group shows one such example – it consists of events that happen over two consecutive hours.

Region II Event groups in this region have high range and high density, thus implying that such events occur over frequent periods throughout the given time interval. Event groups in this region can be used to identify trends in the data. The blue event group consists of similar events that occur every hour during a 7 day period, and therefore has both high range as well as density.

Region III Event groups in this region consists of a small number of events that span a large range. Such events could move to Region II at a lower time resolution, and could potentially represent patterns that are regular over a large time interval, but irregular with respect to the range of the input data. The orange event group contains events that occur over two weeks, but only for one hour per week. Note that this event group moves to Region II (Figure 3.6(d)) when the time resolution is changed to a week.

Region IV Event groups in this region have low range as well as low density. However, such events could move to Region I at a lower time resolution. Figure 3.6(c) shows an example where the purple event group, having events over two days, moves from Region IV to Region I when the time resolution is changed to a day.

As we show later in this chapter, this view acts as a powerful device to help identify many interesting patterns. Using it in conjunction with the query interface simplifies the exploration of large datasets.

While the distribution view gives an overview of an event group, its exact periodicity cannot be inferred accurately. This is instead accomplished using the

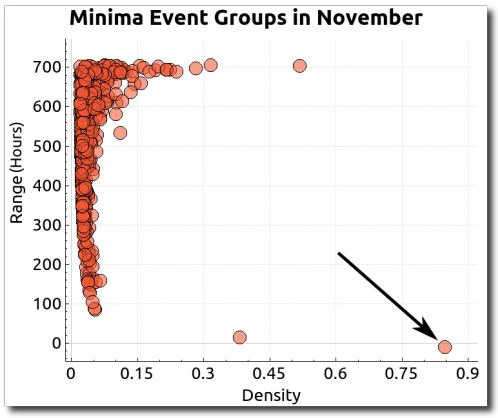


Figure 3.7: The event group distribution view for November 2011 after filtering. Note that the event group corresponding to Macy's Parade stands out in this view, thus helping the user identify this pattern.

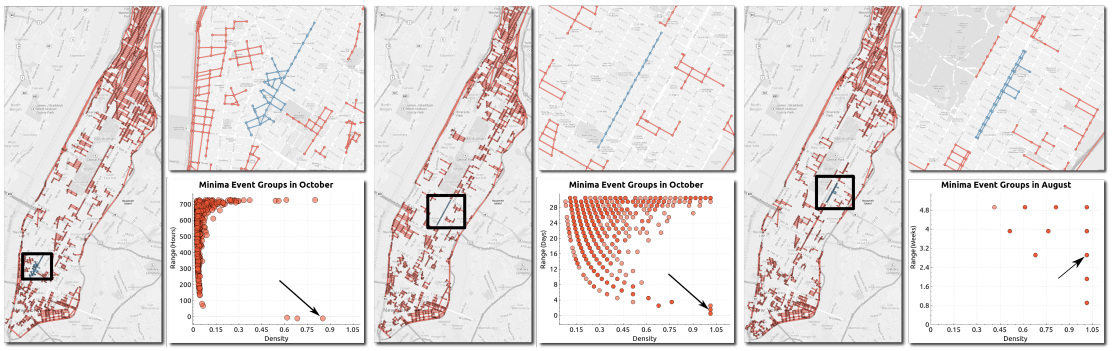
timeline view which visualizes the individual events in an event group over time (see Figure 3.6(a)). This view is inspired by the Gantt chart visualization [58] which is commonly used to represent activities (events) over time. Event groups are represented by a horizontal sequence of points, each point representing individual events. The x-axis in this view represents time. In the timeline view, the event groups are sorted based on their topological importance in order to help the user in identifying significant event groups (the most important event group is on the top). Figure 3.6(a) shows the timeline view containing the four event groups described above.

3.5.3 Filtering Interface

Given the possibly large number of event groups, the ability to filter them not only helps to remove spurious events, but it also allows users to focus on specific types of events. Our visual exploration interface supports multiple property-based filters:

Event group size allows users to filter event groups based on the number of events constituting that group. The user can specify both the maximum and minimum size using this filter. For example, if the user is interested in patterns that happen for at least 4 hours, then minimum size should be set to 4.

Event size allows the users to filter event groups based on the geometric size of



(a) (b) (c)

Figure 3.8: Minima events in NYC. **(a)** Selecting the lone high density event group from Region I of October's event distribution view identifies the event corresponding to the Halloween Parade in 2011. **(b)** Changing the time unit to daily, and selecting high density event groups with range = 2 identifies events corresponding to Hispanic Day Parade and Columbus Day Parade that occurred on consecutive days. **(c)** Changing the time unit to weekly for the month of August, and selecting high density event groups with range = 3 identifies events corresponding to the NYC Summer streets that occurred on three consecutive Saturdays.

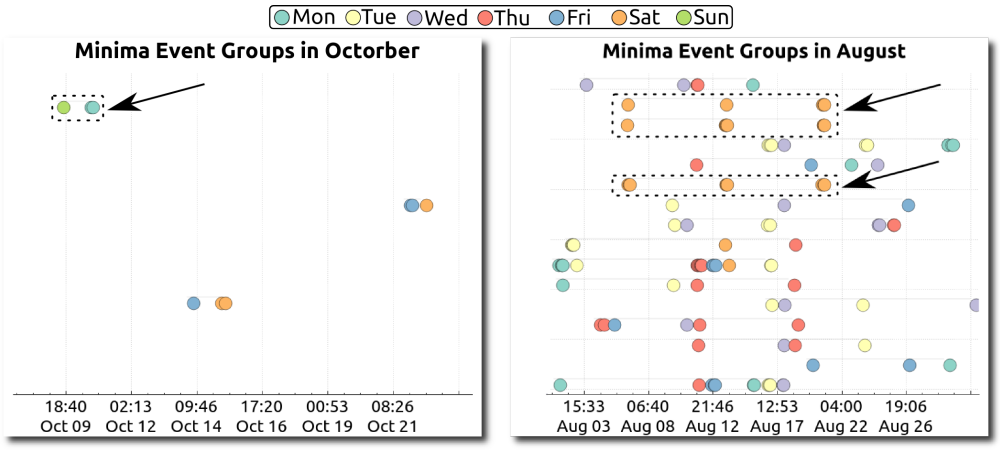
the events in the group. The geometric size of an event is defined as the size of the subgraph representing that event. For example, the user might be interested in viewing only events that span at least 10 nodes.

Event time allows the user to filter events for a particular time period. For example, the user can search for events that occur only at night.

Spatial region allows the user to select regions on the map and filter out events that occur outside this region.

Figure 3.7 shows the event group distribution view for November 2011 after the application of the first two filters. The minimum event group size was set to 4, and the minimum event size was set to 10. The lone highlighted event group in Region I corresponds to Macy's Parade.

Multiple event groups can overlap in the distribution view. To help identify event groups that are occluded in the distribution view, we also allow users to filter event groups using the distribution view, and visualize the selected events in the time line view.



(a) (b)

Figure 3.9: Using the timeline view to isolate events. **(a)** Identifying daily events. The most important event group (the topmost event group) in the timeline view corresponds to the daily event corresponding to the Hispanic Day Parade and Columbus Day Parade. **(b)** Identifying weekly events. The timeline view is used to select a periodic event group, which corresponds to the NYC summer streets that happened on Park avenue for 3 consecutive Saturdays.

3.6 Case Studies

In this section, we present scenarios that illustrate the features and benefits of event-guided exploration for two datasets: NYC taxi trips and subway data (Section 3.3).

3.6.1 NYC Taxi data

We applied our technique on the density scalar function derived from the NYC taxi data. In what follows, we describe a use case where we explored events at different time granularities and queried for similar events. Motivated by a problem posed by the TLC, we also looked for trends in the data to help them identify areas with high concentrations of taxis. We have been collaborating with experts from the DoT and the TLC who are currently using TaxiVis [54] to analyze the taxi data. We demonstrated the event-guided exploration framework to them. Their feedback was very positive and they expressed interest in using the framework together with TaxiVis to improve policy decisions in their respective organizations.

3.6.1.1 Minima events in NYC

Minima events are of interest for the taxi data, since they provide information about regions where there are comparatively fewer taxis. If such minima events occur in places where there is usually a high density of taxis, then this implies blockage of streets. Policy makers in the DoT are particularly interested in identifying such road blocks. By analyzing its spatial location and frequency of occurrence, they are interested in putting in place policies to help them handle such situations. Since such phenomena occur rarely, we focus on Region I of the distribution view to identify such events. We categorize the events into hourly, daily, and weekly respectively, denoting the periodicity of the event. In this section, we briefly discuss a few examples of events that we found while exploring different months.

Hourly events. Figure 3.8(a) shows the event group distribution view for the month of October in 2011. Filtering out event groups having less than 4 events and event size greater than 10, and selecting the highlighted event group reveals an event that occurred along Sixth avenue in Greenwich Village on October 31st. This corresponds to the annual NYC Halloween Parade. As shown in the running example, we also find the Macy's Thanksgiving Parade in November. Figure 3.3 illustrates the process used to identify the NYC Five Boro Bike Tour that happened on 1 May 2011. Using similar a similar process, we were able to find many other events such as the New year's eve ball drop, St. Patrick's Day Parade in March, etc.

Daily events. Figure 3.8(b) shows the event group distribution view for October when the time unit is changed to days. The highlighted high density point with range = 2 consists of multiple event groups. However, using the timeline view to choose the most important event group identifies the one that happened on Fifth avenue on October 9th and 10th, 2011 (see Figure 3.9(a)). This corresponds to the Hispanic Day Parade on 9th October and the Columbus Day Parade on October 10th.

This change in the resolution of the time unit essentially helps in boosting the density of periodic events that happen on consecutive days, but for only a few hours per day. Note that it will be difficult to isolate this event group in the distribution view when the time unit is an hour since it is part of the dense cluster of points in Region IV (Figure 3.8(a)). Exploring the month of May, we also found events

Date	Event
17 March 2011	St. Patrick's Day Parade
7 October 2011	Pulaski Day Parade
10 September 2011	Labor Day Parade
8 September 2012	Labor Day Parade
8 October 2012	Columbus Day Parade
14 October 2012	Hispanic Day Parade
11 November 2012	Veterans Day Parade

Table 3.1: Event groups similar to Hispanic Day Parade.

corresponding to the 9th avenue Food Festival that happened on May 14th and 15th, 2011.

Weekly events. Changing the time unit to "week", we were able to isolate the event corresponding to the NYC Summer streets that happens on Park avenue as shown in Figure 3.8(c). The Summer streets for the Year 2011 occurred on three consecutive Saturdays, 6th, 13th, and 20th August respectively. Note that selecting high density event groups with range equal to 3 weeks results in multiple event groups as shown in the timeline view in Figure 3.9(b). This includes event groups in which the events are not periodic. The events in the timeline view are colored based on the day of the week. This helps in visualizing the periodicity of the events, and one can immediately locate the periodic 3 week event group (highlighted in Figure 3.9).

3.6.1.2 Querying events

Querying for events similar to a given pattern is essential for the analyses performed by experts in DoT. Using the interface described in Section 3.5, we can search for events similar to a selected event that occurs in other months. We now discuss a few results obtained when querying for events similar to the ones automatically detected.

In Figure 3.3, searching for events similar to 2011's Five Boro Bike Tour, we find the Five Boro Bike Tour that happened in 2012 together with the Dominican Day Parades that happened in 2011 and 2012. Additionally, we also find that the Gaza solidarity protest was held at the same location on November 18th, 2012.

When querying for patterns similar to Hispanic day parade, we were able to find other parades that also occurred in the same location. Table 3.1 lists the set of events similar to the Hispanic day parade. Similarly, the query with the New

year's ball drop event on December 31st, 2011 returns the same event from January 1st, 2012 and 2011, and 31 December 2012.

3.6.1.3 Identifying trends

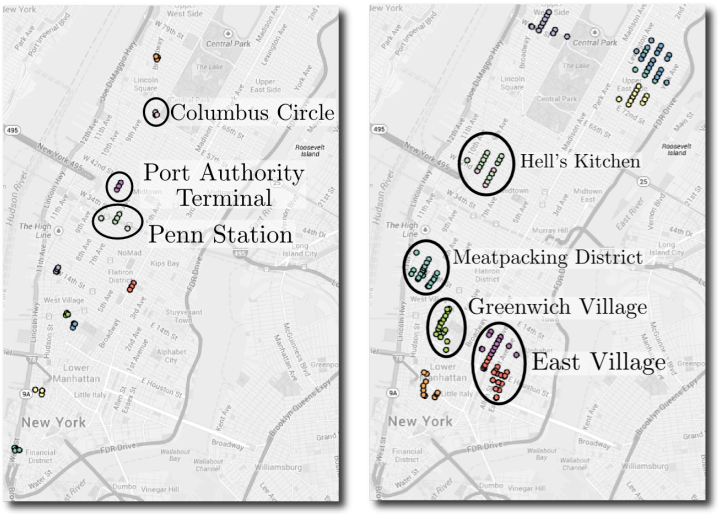
Maxima events show high concentration of taxis. If such concentrations are frequent, then it could imply taxi hot spots. Experts from TLC are particularly interested in identifying such locations. They intend to install data receivers throughout Manhattan to collect data at regular intervals from all the taxis. By placing the receivers at strategic locations such as hotspots they hope to optimize the amount of hardware used. The location of these hot spots can also identify interesting regions in Manhattan.

Selecting highly frequent maxima event groups from Region II of the event distribution view locates the different taxi hot spots in NYC. Figure 3.10(a) shows the top 10 hot spots for the month of November 2011. Note that the frequent hot spots include transit locations such as the New York Penn station and the Port Authority Bus terminal, in addition to tourist locations such as Central Park (Columbus circle).

By switching to lower time resolution (weekly) and viewing event groups that consist of only events from 9 pm to 6 am, helps us identify various places of nightlife in Manhattan, as shown in Figure 3.10(b). This includes areas popular for restaurants and night clubs in Lower Manhattan such as Greenwich Village, East Village, and Meatpacking district, in addition to the Hell's Kitchen region in Midtown. Note that there is also a high concentration of taxis during this time on both Upper East Side and Upper West Side. Experts from TLC pointed out that this was because a lot of people in that area use taxis to return home late night. It is interesting to note that the frequency of these events was more prominent during weekends compared to weekdays.

3.6.2 MTA data

To identify events related to delays, we used the average delay of trains at a given station as the scalar function. Since we are only interested in the amount of delay, topological persistence is used as the importance measure for this data



(a) (b)

Figure 3.10: Identifying trends. **(a)** Taxi hot spots correspond to areas of high activity in NYC. These include transit locations and tourist spots. **(b)** Taxi hot spots during the night corresponds to areas of active nightlife.

set. As we discuss below, we have presented the automatically identified events to engineers at the MTA, who not only provided interesting insights, but were also piqued by some of them.

3.6.2.1 Identifying trends

Minimum event groups. A minimum event for this function corresponds to a station at which the delay is lower than that of its neighbors. This also signals the station where trains start to get delayed. Therefore, a frequent presence of such events could indicate a problematic situation at a station that needs to be investigated. Such frequent patterns are represented as event groups in Region II of the event group distribution view. We now discuss a few such interesting event groups along with their implications.

Figure 3.11 shows the event group distribution view in August 2013 for the southbound *Line 3* trains, which run from Harlem in Manhattan to New Lots Avenue in Brooklyn. A frequent daily minimum event group for this month corresponded to the Wall Street station. Plotting the frequency distribution of events in this group across both hours of the day, as well as days of the week

Figure 3.11: Frequent minima events for August 2013 for the south bound 3 trains. Two of the event groups in Region II of the event group distribution view (daily events) corresponds to the Wall Street and the 14^{th} Street stations. Frequency distribution of events in the minimum event group corresponding to Wall street station indicates that this event predominantly occurs during rush hours, i.e., between 8 and 10am and between 4 and 8pm. Also note that the frequency of this event is higher on weekdays than on weekends.

indicates that such an event occurs at this station predominantly during the rush hour period on weekdays. Note that a large number of people use Wall street station, with it being in the middle of the financial district that houses a lot of offices. We noticed this pattern even for other months.

According to MTA engineers, the delay at this station is due to two reasons. Because the station is small, it can be difficult for passengers to board on or off the trains due to the crowds. More importantly, passengers tend to hold doors in order to allow other passengers to board, thus delaying the train from leaving the station. A similar daily minimum event group was also found on the 14^{th} street station. Here the delay is because the 3 train sometimes waits for the 1 train to arrive in order to allow passenger transfer between them.

When considering northbound Line 3 trains, in addition to event groups similar to that of the southbound line, we also found a frequent event group at the Borough Hall station. The MTA engineers did not expect delays to occur at this station, and they were interested in investigating further as to why it occurs. Another unexpected delay was found at the Dyckman Street station along the northbound Line 1 trains. This was interesting for the engineers because this delay occurred predominantly during late nights and early hours of the day.

Maximum event groups. A maximum event for the computed delay function corresponds to a station from which the train makes up on time. One such frequent maximum event group was found on 34^{th} street Penn station for the Line 3 southbound train. This is because the stop after Penn station, which is 14^{th} street, is far from Penn station. Additionally, this stretch of the route is straight allowing the train to speedup and make up on time. Other frequent maximum event groups we found were also along such straight stretches of the route.

3.7 Discussion

Event structure. The structure of topological features depends on the input scalar function. Small changes in the scalar function can change the geometric size of the events found, causing the geometry of the actual event to be split across multiple topological features. For example, the event group found for the NYC 5 Boro bike tour consists of 2 events corresponding to time steps 8 am and 9 am respectively. However when the map view for one of those time steps is viewed, one can find that there exists another event that also corresponds to the bike tour. Figure 3.12 illustrates this phenomena, where the blue region is part of the event group identified by exploring the visualization interface. However, by *guiding* the user to this time step, the user can also find the red region. The two regions represents a huge portion of the path taken by bike tour in Manhattan that starts at Battery Park and passes through 6^{th} avenue into Central Park.

As a side effect, this also causes some of the events not to be part of the event group. For example, while the roads are blocked for the Macy's Parade from early hours of the day until 11 am, the event group corresponding to it consists of events from only a subset of these time steps. Again, once the users are assisted towards

Figure 3.12: Due to the impact of the scalar function on the shape of the events found, the path taken by the bike tour is split into multiple regions.

one of these time steps, they can find the others by exploring close by time steps. It will, however, be interesting to explore techniques to allow combining such split events into a single event.

Similarity computation. The grouping of events depends on the thresholds ϵ_δ and ϵ_τ. For the results reported in this paper, we used $\epsilon_\delta = 0.3$ and $\epsilon_\tau = 0.2$. While these values gave good results for both the taxi and subway datasets, it would be interesting to explore methods that help identify a good threshold based on the data. The ordering of events when computing event groups can also affect the number of groups found. We plan to explore the effects of event order in the quality of the event groups found.

Currently, we restrict events in a group to be present in the same spatial location. Users might also be interested in the occurrence of events that have a certain shape. For example, using the taxi data, user might be interested to find events where a long stretch of an avenue is blocked irrespective of the location. This is a difficult problem which we intend to explore in future work.

Scalar function computation. While we show two possible transformations of the raw data to scalar functions, we expect many other scalar functions to be useful. The design of the scalar function is dependent on the application. For example, another possible scalar function that can be computed using the MTA data set

is the average waiting time of a passenger at each station. MTA engineers are interested in maintaining high frequency between trains especially during peak hours. Hence a frequent maxima event group of such a scalar function would help in identifying stations which have significant waiting times.

Another parameter in designing the time-varying scalar function is the time interval used to convert the temporal component into a set of discrete time steps. We chose an hourly interval since it was small enough to not conceal short events, but was large enough to provide sufficient data to avoid spurious events.

3.8 Summary

In this chapter, we introduced a topology-based technique for event-guided exploration of spatiotemporal data, which aims to extend the capabilities of current visual analysis systems by guiding users towards interesting portions of the data. Our technique uses efficient algorithms to capture topological features of spatiotemporal data and enables flexible exploration of events by grouping and indexing events. As demonstrated in a variety of case studies, the proposed approach is efficient, scales to large datasets, and is able to handle a wide range of event types. As discussed in Section 3.7 our work opens many directions for future work, both in exploring possibilities to improve the event detection mechanism and also in user interaction with this novel exploration tool.

Chapter 4

Clustering Trajectory Data for Visual Presentation and Pattern Discovery

In this chapter, we continue to pursue the problem of designing techniques for pattern mining in spatiotemporal data. However, while in Chapter 3 we were concerned with static patterns present spatiotemporal data, in the present chapter, we are concerned with dynamic or movement patterns in trajectory data.

Trajectory data is often captured to study movement in different domains. For example, ecologists study animal movements to learn about population growth, social interactions, feeding and migratory patterns, etc. [25, 61]. Meteorologists use trajectory data to help predict storm paths [27, 46], and researchers from a wide variety of fields study human mobility to perform targeted advertising, predict traffic and commuting patterns, as well as data-driven urban planning [21]. In all such cases, due to the vast amount of data being collected, there is a great need for scalable and efficient techniques for analyzing this data and discovering the underlying patterns [63].

As briefly discussed in Chapter 1, the analysis of this kind of data is challenging not only because of its size, but also due to its complexity [108]. Trajectories are spatiotemporal in nature, involving dynamic geometric positions, directions, velocities, durations, and other characteristics specific to the entities being tracked. Hurricane tracks may include overall storm strength, wind speeds, or seasonality.

Animal tracks may be influenced by their size, age, or gender. Incorporating these characteristics, when available, can help direct the trajectory analysis, but adds complexity. In particular, all of these features poses challenges to visual analysis of trajectory data. In fact, effective representation of large collections of trajectory data is difficult and common solutions often produce extremely cluttered displays (see Figure 4.1).

In this chapter, we propose a technique to attack the problems of movement pattern mining and visual representation at the same time. This technique consists on a novel approach for model-based trajectory clustering based on vector field fitting. Vector field is a mathematical concept widely used in the fields of computer graphics, scientific computing and scientific visualization [32, 121]. Intuitively, the idea behind our method is to use vector fields as a generative model for the spatial component of the trajectory data. This can be seen as the inverse operation of producing characteristic lines, such as *streamlines*, of a vector field. This process induces a similarity notion on the dataset by considering a set of trajectories *similar* if they can be approximated well by streamlines of a single vector field (see). This modeling approach solves the two main drawbacks of previous methods, namely that we can define a similarity that naturally encodes features of the trajectories together with their geometries. Clustering is achieved through parameter estimation algorithm based on alternating optimization called *Vector field k-means*. One important consequence of our modeling approach that is not present in previous work is the ability to capture *global patterns* in the data that are not evident when considering only local information (Section 4.5). Furthermore, the obtained vector fields are a good summary of the movement patterns in the clustered data. Finally, we can leverage a vast set of techniques for vector field visualization to display such movement patterns (see Figure 4.1 and Section 4.5).

Section 4.6 describes how to modify the method above to not only consider the spatial component of the trajectories, but also their attribute components. This is achieved by using a different generative model based on *scalar fields*.A parameter estimation algorithm similar to vector field k-means can be used obtain the clustering based on the attribute components. We call this algorithm *Attribute field k-means*. As we discuss in Section 4.6, the scalar field modeling approach provides a generalization of the vector field modeling previously described.

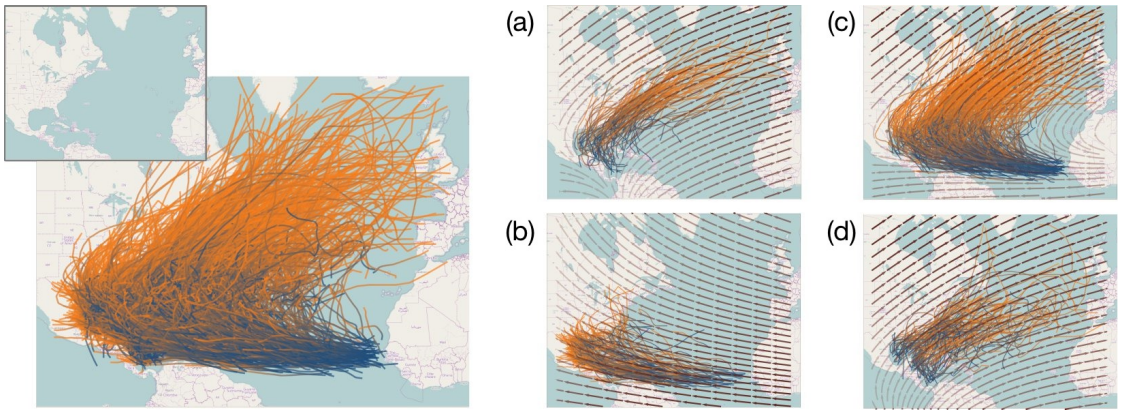


Figure 4.1: Left: 1415 Atlantic tropical storms (from the HURDAT dataset) used as input. The orientation of each trajectory is represented by linearly interpolating from blue (start) to orange (end). This color scheme is used throughout this chapter. Right: four trajectory clusters and their corresponding vector fields showing relative speed. Clusters (b) and (c) contain the Cape Verde-type cyclones, and separate them according to whether they dissipate in North America (b) or turn back to the Atlantic Ocean (c). Clusters (a) and (d) show storms developing in the Gulf of Mexico. We observed faster-moving storms to be in cluster (a), and slower-moving ones in (d).

4.1 Related Work

Due to the growing rate at which mobility data is being collected, *computational movement analysis* is a very active research field, combining techniques and expertise from many fields, including GIS, information visualization, computational geometry, databases, and data mining [63]. In this work, we focus on just one of the problems in movement analysis, that of extracting arbitrary movement patterns from trajectory data. In other words, given a large number of trajectories of moving objects, e.g. animals, people, or vehicles, we want to quickly identify underlying patterns that exist and that shed light on the global movement trends of the moving objects. Our approach for identifying these patterns is to perform trajectory clustering. As Kisilevich et al. [74] provide a thorough examination of many trajectory clustering techniques, we briefly review the most relevant methods here.

Rinzivillo et al. [108] designed a density-based progressive clustering technique that can utilize different distance functions at each step of their clustering. This allows analysis of objects with heterogeneous properties to be handled differently

during the cluster refinement stages. Lee et al. [80] also use density-based clustering, but believe that clustering whole trajectories may miss common sub-trajectories. They partition the trajectories into line segments based on a simplification algorithm and cluster these segments using the notions of neighborhood and density. The techniques of both Rinzivillo and Lee rely on the definition of a distance measure between trajectories. This is known to be a difficult problem, in the sense that no proposed distance measure captures well all the attributes of trajectories [63]. For example, both of these methods only use the geometry of the trajectory and therefore they cannot encode speed information, which might be relevant in cases like storm track analysis. Pelekis et al. [104] exploit local similarities of subtrajectories too, but they also study the effect of uncertainty (in measurement) in the original trajectory data.

Like Rinzivillo et al., our overall approach falls within the broader category of *visual and exploratory movement analysis*, which exploits humans' ability to visually detect patterns, and then steer the visualization and analysis to those regions of greatest interest. The research of Andrienko and Andrienko [11, 13, 108] has focused on human-in-the-loop analysis systems, but has also included more general aggregation and visualization of movement data [5], and more recently the identification of important locations and events by analyzing movement data [9, 15] and the visualization of trajectory attributes [118]. All these works could benefit from more efficient clustering methods since they rely on clustering for display purposes.

An important problem related to trajectory clustering is how to ultimately visualize trajectory data. Traditionally flow maps [15, 107, 124] have been used to convey the amount of people and goods that moved between locations but without necessarily reporting the exact routes that were taken. More recently, there have been several compelling techniques based on density maps [110, 130], heat maps [97], kernel density estimation [36], and 3D based visualizations [12, 118]. Vector fields have been widely used in scientific visualization and even by some researchers doing trajectory clustering analysis to show speed and direction of animal movements [25] and wind [27]. In these cases, they have only been used to visualize the results, rather than as an integral part of the underlying clustering technique.

The related problem of deriving vector fields from trajectory datasets has been

studied in different contexts. Jänicke et al. [72] defined a measure to evaluate vector field visualizations based on how well a computer vision algorithm could derive a vector field from an image containing visual representations (e.g., streamlines) of the vector field. This problem was independently investigated in the image processing community by Nascimento et al. [93]. Although the model proposed by Nascimento et al. has some similarities with ours, we point out that it is not a clustering algorithm, but a way to model movement as vector fields. Furthermore, their model has a set of complex parameters to be estimated which brings higher computational costs, and more importantly increases the complexity for the human analyst. Thus their method is not suitable for our purposes. We propose a different (and simpler) modeling approach which has a smaller number of variables and outputs not only the vector fields (that in our case do represent distinct mobility patterns) but also a meaningful clustering of the input dataset, which makes our algorithm suitable for visual data analysis.

4.2 Background and Notation

Trajectories are modeled as paths of the form $\alpha : [t_0, t_1] \rightarrow \mathbb{R}^2$. We assume we are given a set of n trajectories $\mathcal{T} = \{\alpha_1, ..., \alpha_n\}$. Each trajectory is given as a sample, i.e., for each $i = 1, ..., n$, we are given a sequence of space-time points $\widehat{\alpha_i} = \{(\alpha_i(t_1^i), t_1^i), (\alpha_i(t_2^i), t_2^i), \ldots, (\alpha_i(t_{p_i}^i), t_{p_i}^i)\}$. We approximate each trajectory α_i with piecewise linear curves (constant velocity between two consecutive samples). This results in a polygonal line representation for each trajectory. For each α_i we denote the interval $[t_1^i, t_{p_i}^i]$ by I_i and by $|I_i|$ the time span for α_i, i.e., $|I_i| = t_{p_i}^i - t_1^i$. We call each portion of a trajectory between two samples a segment of the trajectory α_i. For each segment $s_j = [\alpha_i(t_j), \alpha_i(t_{j+1})]$ of α_i we define $\omega_{s_j} = \frac{t_{j+1} - t_j}{T}$, where $T = \sum_{\alpha_i \in \mathcal{T}} |I_i|$ is the total time span in the dataset.

In this paper, we consider vector fields as functions defined over a domain $\Omega \subset \mathbb{R}^2$ with values in $\mathbb{R}^2$. We discretize Ω as a regular triangle grid G with resolution R (R^2 vertices) and assume linear interpolation within each face of the grid for the reconstruction of the vector field. We assume all trajectories are contained in this grid and are tessellated so that each trajectory is comprised of segments that do not cross the boundaries of the domain triangles as in Figure 4.2.

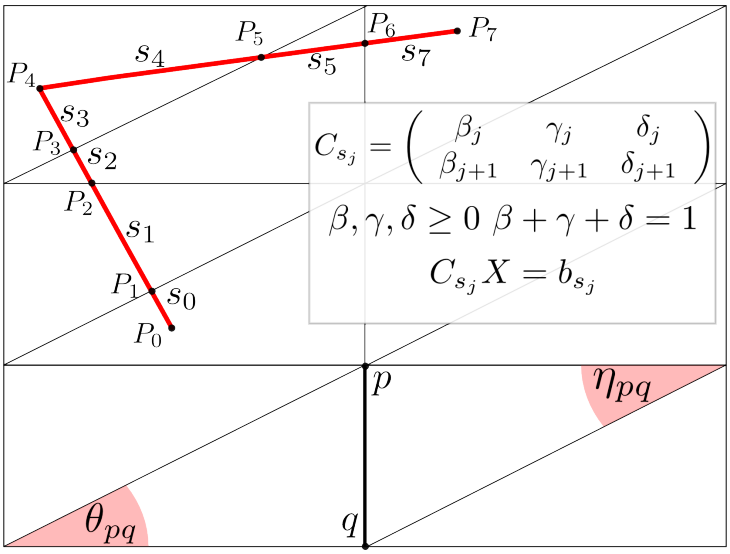


Figure 4.2: Illustration of the trajectory tessellation and Laplacian matrix computation. The trajectories are tessellated so each segment is contained on a face of the grid. Each segment s_j determines a constraint in the form of a matrix C_{s_j}. The Laplacian matrix enforces our smoothness penalty.

For the optimization problems described in this paper, the components of the vector field can be treated separately. Therefore, the presented formulas are written as if X were a scalar field. For each segment s_j of a trajectory α_i we denote by C_{s_j} the $2 \times R^2$ matrix that contains in the first and second rows respectively the barycentric coordinates of the first and second vertices of segment s_j (with 0 entries everywhere else); Figure 4.2 illustrates the setting. We also define the vector b_{s_j} as the vector whose entries equal the value of the velocity vector of the segment s_j, i.e., $b_{s_j} = \frac{\alpha_i(t_{j+1}) - \alpha_i(t_j)}{|I_s|}$.

4.3 Vector Fields as a Generative Model of Trajectories

As briefly described before, our approach consists of capturing movement patterns by defining a vector field for which the trajectories are approximately streamlines, i.e., we attempt to separate the trajectories into a small number of clusters according to the best vector field that approximates them. Streamlines are

characteristic lines that are tangent to the vector field at each point. More formally, given a vector field X, a trajectory α is a streamline of X if $X(\alpha(t)) = \alpha'(t)$, where $\alpha'(t)$ denotes the velocity vector of α at time t.

Our assumption is that for every set of trajectories $\mathcal{T}$, there exists a set of *smooth vector fields* $X_j \in F, |F| = k$ that explains most of the mobility in the data, in the sense that each trajectory would be approximately tangent to one of the X_j. More formally, we define this as the problem of finding the vector fields X_j and the cluster assignment function $\Phi : \mathcal{T} \to \{1, \ldots, k\}$ such that for each $j = 1, \ldots, k$

$$\begin{cases} \Delta X_j = 0 \\ \alpha_i'(t) = X_j(\alpha_i(t)), \forall \alpha_i \in \Phi^{-1}(j) \text{ and } \forall t \in I_i \end{cases}$$

where Δ denotes the vector Laplace operator. The first equation directs the solution towards smooth vector fields; the second equation ensures that vector fields represent the trajectories well. It is clear that with this formulation the problem may have no solution. We propose instead to solve it in the least squares sense, by minimizing the following energy

$$E(X_1, \ldots, X_k, \Phi) = \sum_{j=1}^{k} \lambda_L \, ||\Delta X_j||^2 + \sum_{\alpha_i \in \Phi^{-1}(j)} \frac{(1-\lambda_L)}{T} \int_{t_1^i}^{t_{p_i}^i} ||X_j(\alpha_i(t)) - \alpha_i'(t)||^2 \, dt, \tag{4.1}$$

where T is the normalization factor (defined previously) and $\lambda_L > 0$ plays the role of a weighting factor: for small values of λ_L less weight is given to the smoothness of vector fields and therefore we get vector fields that match the given trajectories but are less smooth. For large values of λ_L we are giving relatively high priority to the smoothness of the vector field.

4.4 Vector Field K-Means Fitting Process

The model presented in Section 4.3 has two main elements: the vector fields and the assignment of trajectories to clusters. Therefore the model fitting problem consists of defining the vector fields $(X_1, ..., X_k)$ and assigning each trajectory to a vector field (Φ), guided by Equation 4.1. We propose a 2-level alternate optimization

Algorithm 1 Vector Field K-Means Outline

Require: k: # of clusters, $\mathcal{T} = \{\alpha_1, \ldots, \alpha_n\}$: Array of curves
Ensure: $V = \{X_1, \ldots, X_k\}$, $\Phi : \mathcal{T} \to \{1, \ldots, k\}$
  $\Phi \leftarrow$ Initialize($\mathcal{T}$,k)
  repeat
    for $i = 1$ to k **do**
      $X_i \leftarrow \mathit{fitVectorField}(\Phi^{-1}(i))$
    end for
    for $i = 1$ to n **do**
      $j_0 \leftarrow \underset{j \in \{1,2,\ldots,k\}}{\operatorname{argmin}} E''(X_j, \alpha_i)$
      $\Phi(\alpha_i) \leftarrow j_0$
    end for
  until converge

algorithm as the vector field k-means fitting process. It consists basically of the following steps:

(*) Given a candidate assignment of trajectories Φ, for each set $\Phi^{-1}(i)$, $i = 1, \ldots, k$, we find the best-fitting vector fields *over those particular trajectories*, and

(**) Given a set of vector fields $\tilde{V} = \{X_1, \ldots, X_k\}$, we compute the best assignment function *for those particular vector fields* by picking, for each trajectory, the vector field that best approximates it.

Algorithm 1 contains the outline of vector field k-means. In this pseudo-code, the step $(*)$ corresponds to the *fitVectorField* routine. As we show below, we can formulate this step as a sparse linear system (in the general form of [115]) whose solution can be computed essentially in linear time, and which gives us the smoothest, best-fitting vector field for a set of trajectories. The step $(**)$ corresponds to finding the vector field with smallest error with respect to a given trajectory. We further describe these components below.

4.4.1 Fitting Vector Fields

The *fitVectorField* routine is the central step of vector field k-means. It consists of an optimization problem with two types of constraints: value and smoothness,

defined below. As depicted in Algorithm 1, in this step we are given as input a *fixed subset* $\mathcal{T}'$ of $\mathcal{T}$ and we want to minimize

$$E'(X, \mathcal{T}') = \lambda_L \, ||\Delta X||^2 + \sum_{\alpha_i \in \mathcal{T}'} \frac{(1-\lambda_L)}{T} \int_{t_1^i}^{t_{p_i}^i} ||X(\alpha_i(t)) - \alpha_i'(t)||^2 \, dt \qquad (4.2)$$

We now use the grid G to write E' in matrix form. First, we use the well-known Laplacian matrix cotangent weights [120], denoted by L, to get a finite representation of the Laplace-Beltrami operator. In general, if we let p, q be a pair of adjacent vertices of the grid (see Figure 4.2), then the entries of the cotangent Laplacian matrix L are given by

$$\bar{L}_{pq} = \frac{1}{2}(\cot \theta_{pq} + \cot \eta_{pq}),$$
$$\bar{L}_p = - \sum_{t \in Neighborhood(p)} \Delta_{pt}.$$

This formula allows us to compute the Laplacian of scalar functions defined over the grid by simply representing the values of the function on the vertices of the grid as a vector and multiplying the vector field by the Laplacian matrix. More concretely, by using the Laplacian matrix of G we can represent the first term of Equation 4.2 in matrix form as $||\sqrt{\lambda_L} LX||^2$, where X is the finite-dimensional representation of the vector field. Before representing the second term of E' in matrix form we define for each segment s_j the matrix $\widetilde{C}_{s_j} = \sqrt{(1-\lambda_L)\omega_{s_j}} \Lambda C_{s_j}$ and the vector $\widetilde{b}_{s_j} = \sqrt{(1-\lambda_L)\omega_{s_j}} \Lambda b_{s_j}$, where Λ is a 2 by 2 matrix given by

$$\Lambda = \frac{1}{2\sqrt{6}} \begin{pmatrix} \sqrt{3}+1 & \sqrt{3}-1 \\ \sqrt{3}-1 & \sqrt{3}+1 \end{pmatrix}.$$

We denote by $\widetilde{C}$ the matrix that consists of stacking C_{s_j} for all segments s_j and denote by $\widetilde{b}_{s_j}$ the vector obtained by stacking all the $\widetilde{b}_{s_j}$ again for all s_j. A simple calculation shows that the second summand in Equation 4.2 is given by $||\widetilde{C}X - \widetilde{b}||^2$ (see Appendix A). Therefore

$$E'(X,\mathcal{T}') = ||LX||^2 + \left|\left|\widetilde{C}X - \widetilde{b}\right|\right|^2$$

The minimization of this energy is a least-squares problem and can be solved by solving the corresponding system of normal equations: $(L^TL + \widetilde{C}^T\widetilde{C})X = \widetilde{C}^T\widetilde{b}$.

4.4.2 Assigning trajectories to vector fields

In the second phase of Algorithm 1, we assume we have the vector fields $\mathcal{V} = \{X_1, \ldots, X_k\}$ fixed. The goal is to build the next function Φ that assigns each trajectory to one of the k cluster centers, i.e. the vector fields $X_1, \ldots, X_k$. The assignment algorithm is trivial: for each trajectory α_i, we simply evaluate $E''(X_j, \alpha_i) = \int_{t_1^i}^{t_{p_i}^i} ||X_j(\alpha_i(t)) - \alpha_i'(t)||^2 \, dt$, for each vector field X_j, and define the new assignment to be the minimizer for all the k possible choices.

4.4.3 Algorithm Initialization

As it is common to alternate optimization methods, initialization is critical to performance [90]. In spite of being crucial it is difficult, in general, to have guarantees on the optimization performance based on the optimization. This was achieved for particular cases such as $K-$Means clustering [17] and low-rank matrix completion [71]. The usual solution is to use heuristics for initialization [90].

We implemented a simple method to choose the initial vector fields and trajectory partitions that was effective in our experiments. The main idea is to try to have as diverse initial clusters as possible. The algorithm takes as inputs an array of curves and a number k of clusters to be created, and starts by choosing a trajectory α at random to be part of the first cluster. It uses the *fitVectorField* routine previously described to fit a single trajectory to the first vector field. The algorithm proceeds by fitting to the i-th vector field the trajectory that has the worst error among all previously fit vector fields. After computing k vector fields, we compute the assignment Φ by picking the best vector fields for each trajectory.

Dataset	Trajectories	Resolution	k	Optimize	Assign
Synthetic	2000	3	2	1.730s	0.074s
Atlantic	1415	5	7	8.076s	0.335s
Beijing Wide	45563	5	4	110.99s	4.265s
Beijing Campus	12883	10	16	124.35s	2.195s
CDR	37435	4	4	201.24s	7.597s
CDR Large	372601	4	4	2497s	75.24s

Table 4.1: Experimental results: For each dataset, we report the number of trajectories, the grid resolution, the number of clusters (k), and the total running times (in seconds) for the vector field fitting (optimize) and trajectory assignments.

4.4.4 Computational Complexity and Convergence

As discussed in Section 4.4.2, the assignment step consists of a linear pass over the trajectory data, and for each of these, we need to find the vector field that minimizes the error. This can be implemented in $O(k|S(\mathcal{T})|)$, where $S(\mathcal{T})$ denotes the set of line segments that compose the trajectories in $\mathcal{T}$. For the fitting step, we have used a simple Unconstrained Conjugate Gradient algorithm as a linear system solver [96]. Therefore, the complexity of this step is given by $O(kN(R^2 + |S(\mathcal{T})|))$, where N denotes the maximum number of iterations of the Conjugate Gradient Method, R denotes the grid resolution corresponding to the multiplication by the Laplacian matrix, and $|S(\mathcal{T})|$ corresponds to the multiplication by the constraint matrix C. As we see in the experiments, good results can be obtained with relatively low values of R and hence the complexity is dominated by $kN|S(\mathcal{T})|$. The choice of the Conjugate Gradients solver was convenience; we could further optimize our implementation by using more sophisticated methods to solve systems of linear equations [96]. We note that each iteration of the main loop in Algorithm 1 is guaranteed to decrease the energy E and thus no assignment can be repeated, from which we conclude that our algorithm converges in a finite number of steps.

4.5 Experiments and Results

We now report the results of running vector field k-means. In our experiments, the algorithm was able to efficiently extract significant movement patterns across diverse datasets. We start with a small synthetic dataset and progressively increase the input sizes until we reach an example with over 370,000 very noisy trajectories. All running times (see Figure 4.1) are from our prototype implementation: a single-

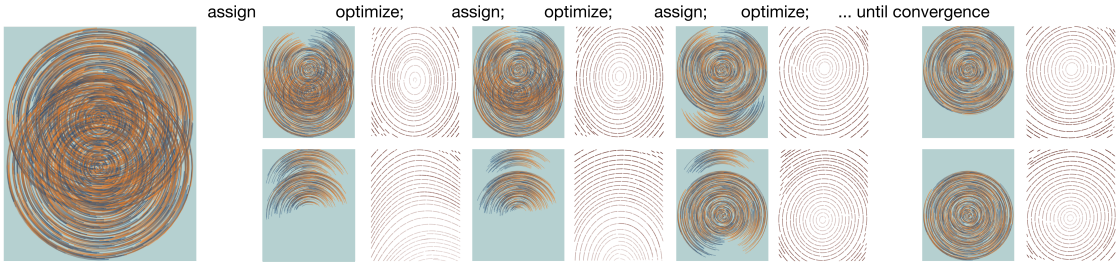


Figure 4.3: An illustration of vector field k-means as it partitions 2000 synthetic trajectories into two clusters. The algorithm alternates between fitting the best possible vector fields from the current assignment ("optimize") and matching trajectories to the vector field which fits them best ("assign"). Although no trajectories form a complete circle, vector field k-means still recovers the two separate circular patterns.

threaded, single-process C++ application running on an Intel Core i7-960 desktop with 6GB of RAM. The total memory required by our application remained under 1GB for all reported experiments.

As there are no standard methods for evaluating trajectory clustering methods, we evaluate our results based on visual inspection and on whether patterns in other data available about the trajectories (besides position and time information), can be found based on our clusters. The parameters used were selected iterativelybasically trial and error and, in some cases, background knowledge. In Section 4.7, we discuss parameter dependence and selection in more detail. We ran the algorithm until convergence, although the optimization could be stopped earlier when the number of trajectories that change from one cluster to another is small.

4.5.1 Synthetic Data

In this dataset there exists two overlapping circulatory movement patterns. Each trajectory covers a partial, randomly selected section of the circle at a random distance from the center. We sampled 1000 trajectories from each overlapping pattern. As we show in Figure 4.3, vector field k-means recovers the two overlapping patterns perfectly, which illustrates that our method does not create clusters by selecting representative trajectories at all: its vector fields fit *all* circular trajectories equally well. Our implementation converged in 30 iterations, in less than 2 seconds, using a 3x3 grid.

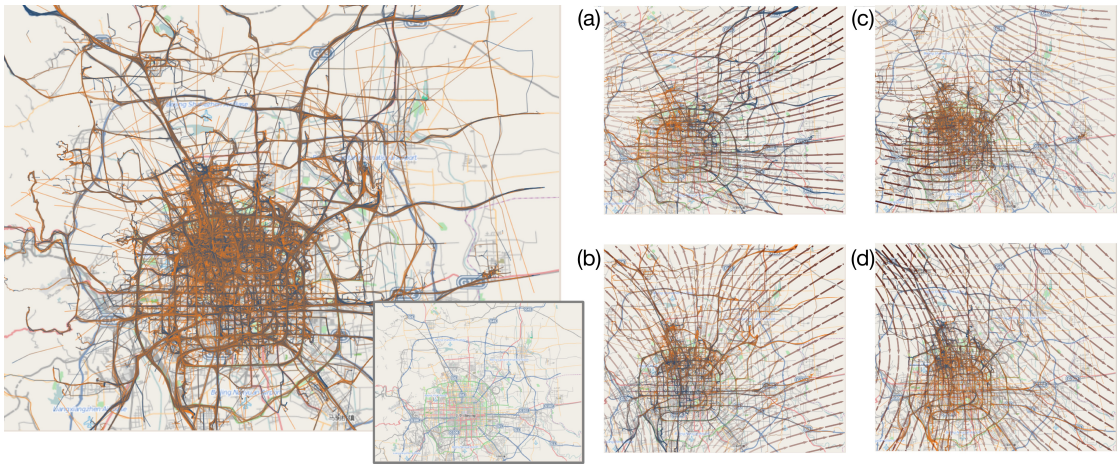


Figure 4.4: Large-scale movement patterns around Beijing, from the GeoLife Trajectories dataset. Clusters (a), (b) and (d) appear to depict travel in and out of the city through the surrounding highways. Cluster (c) has much slower speeds and its trajectories are tightly packed around a small region. Upon inspection, we found this to area to contain the Microsoft Research Asia campus.

4.5.2 Atlantic Hurricanes

HURDAT is a hurricane tracking dataset maintained by the National Hurricane Center (NHC) [1]. The dataset contains trajectories of the 1415 Atlantic tropical storms between 1861 and 2011. It contains not only position and time information, but also sustained surface wind speeds and sea-level pressure. The data are recorded with a resolution of 6 hours.

In Figure 4.1, vector field k-means separates what looks like a fairly uniform set of trajectories. One of the clusters neatly captures Cape Verde hurricanes which tend to make landfall in North America, while two other clusters show storms that originate in the Caribbean and Gulf of Mexico. Upon closer inspection, this separation of two similar looking clusters is due to the more chaotic trajectories of one of the clusters, which result in a generally lower-velocity vector field.

4.5.3 GeoLife GPS Trajectory Dataset

The GeoLife GPS dataset consists of 17,621 trajectories recorded by Microsoft Research at Beijing. The trajectories are GPS tracks of 178 users from April 2007 to October 2011 [133]. Although the dataset encompasses trajectories across the globe,

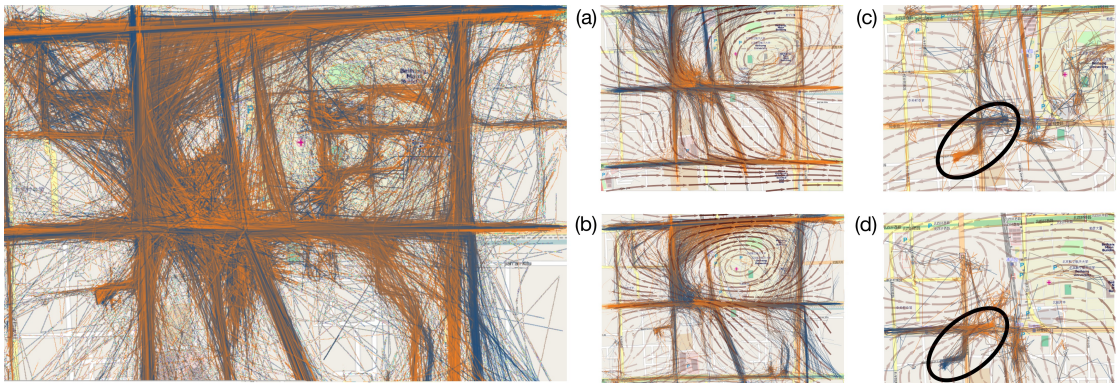


Figure 4.5: The GeoLife Trajectories dataset clustered using vector field k-means. The original trajectories were cropped to a 10 block area in downtown Beijing. We partition the data into four clusters ($k = 4$), and then subdivide each cluster resulting in 16 subclusters. Images (a) and (b) show two clusters from the first-level subdivision, and images (c) and (d) show two clusters from the second level. The first two vector fields show trajectories in patterns of faster vehicular traffic, while the latter appears to show pedestrian traffic moving to and from a lunch spot near the Microsoft Research Asia campus.

we focus on two regions around Beijing. As the raw trajectories are unsegmented, some lasting for days, we split trajectories whenever the time between two samples is larger than 2.5 times the median time between samples. We then reduced the sampling rate by only keeping measurements at least 2 minutes apart.

Figure 4.4 shows a first run of vector field k-means on the GeoLife dataset in which the algorithm was able to find general movement trends within the trajectories. Three of these are clear directional patterns heading west, north, and south. We speculate these to be mainly commuting patterns, since the remaining cluster (subfigure c) consists essentially of trajectories inside the city's road network. Although we believe that with the chosen grid resolution (5x5) vector field k-means cannot reliably resolve the patterns in that cluster (and hence the vector field is not very informative), the large density of trajectories around a relatively small area in the cluster suggested further analysis centered in that region.

Figure 4.5 shows an exploration we performed on that narrower region of Beijing. In this case we used a grid of resolution 15. By looking at the darkness of the arrows in Figure 4.5 we see that regions of different speeds are found. For example, traffic over the streets (Figure 4.5 (a) and (b)) is much faster than the speed of

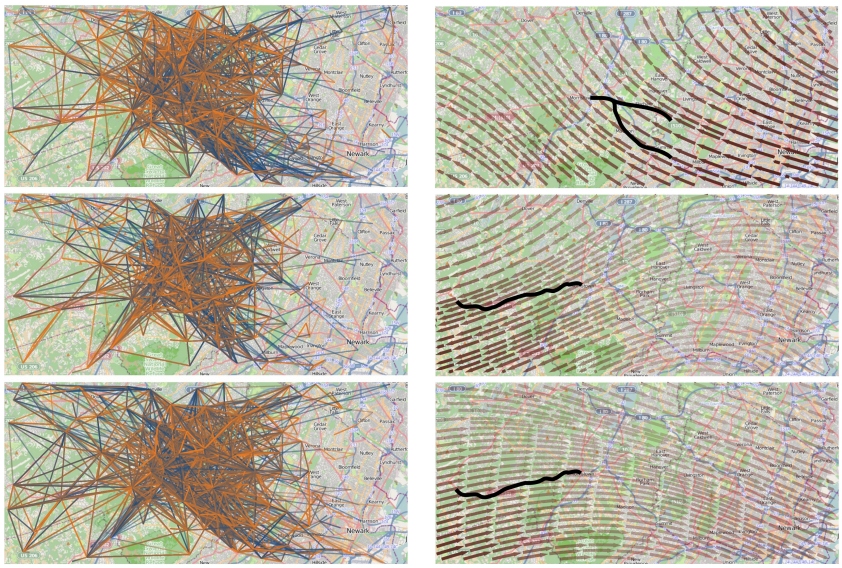
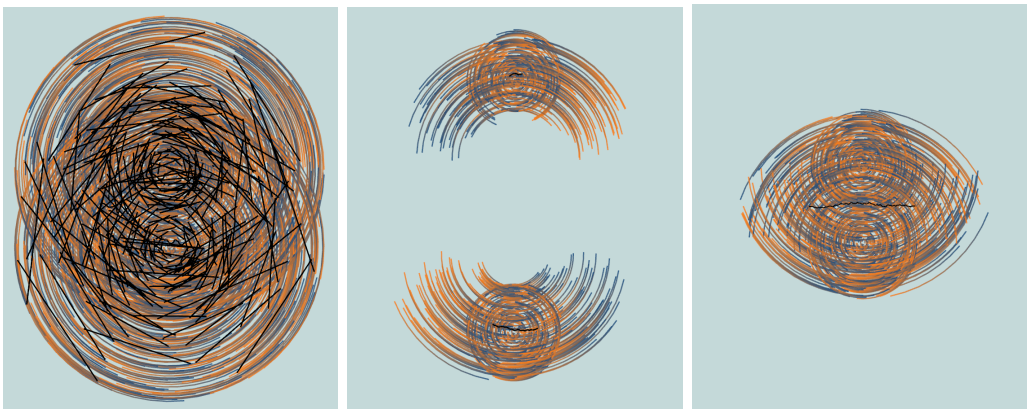


Figure 4.6: **Left.** The anonymized call detail records for over 370,000 cell phone calls produced noisy trajectories around a suburban city. Three of the four clusters computed are shown. **Right.** Despite the noisy trajectories, we recovered clear movement patterns related to highway (bold black lines) traffic. The three vector fields correspond to the clusters on the left.

people going to and from the speculative lunch spot location (Figure 4.5 (c) and Figure 4.5 (d)), which suggests that people walk to that destination.

4.5.4 Call Detail Record Dataset

We collected anonymized Call Detail Records (CDR) from the cellular network of a large U.S. communications service provider. For each phone call, CDRs provide us with the sequence of cellular antenna locations, known as handoffs, used during that call. We collected over 370,000 calls during one week in 2011 for approximately 300 antenna located near a suburban city with 20,000 residents. Our goal was to capture handoffs related to vehicular traffic.Given the sensitivity of CDRs, we took several steps to ensure privacy. The data was collected and anonymized by a third party not involved in the data analysis. Unlike other studies [22] our dataset cannot associate multiple calls made by the same individuals: each call is independent. As a result, all of our input trajectories (i.e.handoff patterns), are only partial trajectories for any individual. As we show in Figure 4.6, although the handoffs are



(a) (b) (c)

Figure 4.7: TraClus experiments using the synthetic dataset. In (a) 268 clusters are found with $\epsilon = 0.03$ and $MinLns = 2$. In (b), with $\epsilon = 0.23$ and $MinLns = 140$ TraClus detects two clusters (drawn here separated for clarity), but clearly merges portions of the two circular patterns. Slight variations on the parameters ($\epsilon = 0.25$ and $MinLns = 160$) causes TraClus to merge the two cluster into one as seen in (c). These results were obtained in 0.8, 1.6, and 6.5 seconds respectively.

quite noisy, we can still recover movement patterns clearly related to the highway traffic around the city.

4.5.5 Comparison

We compare vector field k-means with TraClus, by Lee et al. [80], a density-based algorithm that is one of the main references in trajectory clustering. Roughly speaking, the algorithm consists of two steps: trajectory simplification into line segments and segment clustering. The algorithm also has two main parameters: a distance threshold ϵ used to define neighborhoods for each segment, and a density lower bound $MinLns$ that is used to find neighborhoods that define clusters. Unlike vector field k-means, TraClus does not incorporate the time information, thus trajectory speed information is lost.

We used the C++ implementation of their algorithm that is available on the author's webpage (`http://dm.kaist.ac.kr/jaegil/#Publications`). In the following experiments, we use the heuristic proposed in their paper (and also part of the author's implementation) to select the parameter values. Whenever this

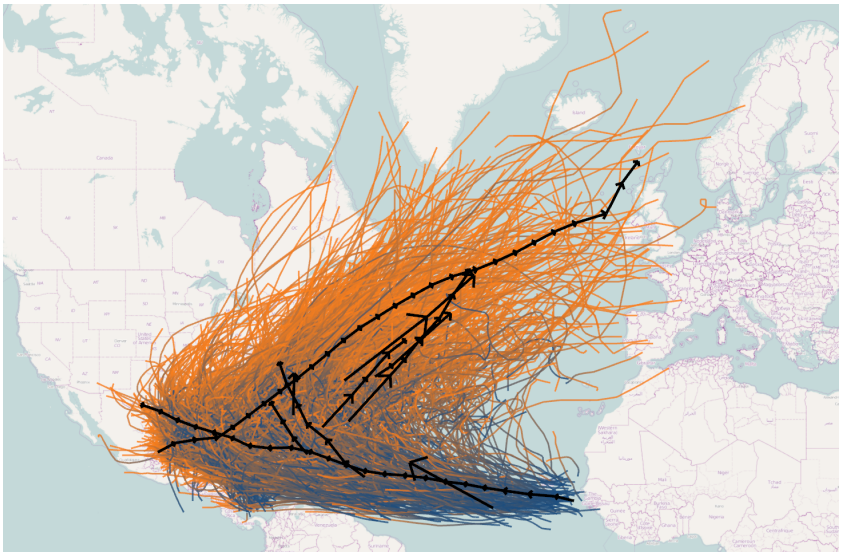


Figure 4.8: TraClus computed nine clusters for the HURDAT dataset. Cluster representative trajectories are in black. Notice that because TraClus starts by subdividing trajectories, no cluster from TraClus captures the pattern of trajectories found by vector field k-means in Figure 4.1 (c).

heuristic fails to provide the parameter estimates, we adjusted the parameters by manually searching over a range of values. We reiterate that as vector field k-means and TraClus mine different patterns, it is difficult to say that one method is always better than the other, but we do investigate what kind of patterns TraClus is not able to find that vector field k-means is, and vice versa.

We first present the performance of TraClus on the synthetic discussed in Section 4.5.1. The results are shown in Figure 4.7. Note that because TraClus utilizes local features while clustering, it is unable to capture the overall global structure of this dataset. More specifically, when the neighborhood size ϵ is very small, TraClus considers each segment as its own cluster, while making ϵ larger groups the segments of the two centers of the circular patterns together, as shown in Figure 4.7.

Our results of running TraClus on the HURDAT dataset, shown in Figure 4.8, are very similar to the ones obtained by the authors [80] though not exact because our input contained more trajectories. In this case, TraClus detects nine clusters in four seconds. The representative trajectory for each of these clusters is highlighted in black, but are not as informative as the vector fields of Figure 4.1. In addition, their

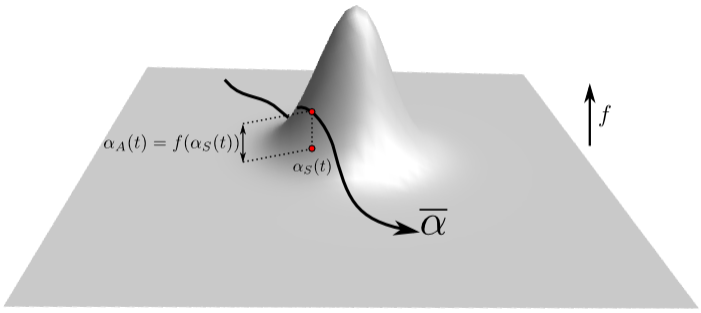


Figure 4.9: Illustration of basic concepts used. An 1-attribute f is represented as a height field. A trajectory $\overline{\gamma}$ consistent with f is also shown.

partitioning step precludes TraClus from generating clusters of Atlantic crossing storms that then proceed up the U.S. east coast. That information has been lost by their approach.

We also performed experiments using the TraClus algorithm and the GeoLife dataset (used in Figure 4.5). In this case, TraClus failed to produce any meaningful clusters. Using the parameters reported by the TraClus heuristics ($\epsilon = 20$ and $MinLns = 12454$), the algorithm finds no clusters, after 54 seconds of computation. Changing the parameters to $\epsilon = 20$ and $MinLns = 1000$, the algorithm reports a single cluster which does not represent any meaningful pattern, after a little more than 3.6 hours of computation.

4.6 Attribute field k-means: Clustering Trajectories with Attributes

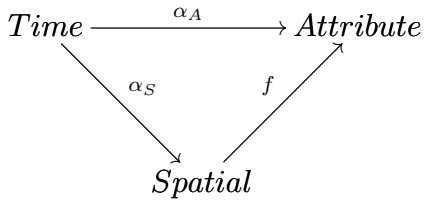
In this section, we describe how to generalize the vector field k-means approach to obtain a clustering method that handles trajectories with attributes. To do so, we need to generalize the idea of streamline (i.e., trajectories that *match* a vector field) to the scalar case to deal the attribute values. In order to achieve this we first extend the notation to be able to express trajectories with attributes.

4.6.1 Extended Notation and Terminology

We define a trajectory with M-attributes, or just *M-trajectory* for simplicity, as functions of the form $\overline{\alpha} : [t_0, t_1] \to \mathbb{R}^2 \times \mathbb{R}^M$, such that $\overline{\alpha}(t) = (\alpha_S(t), \alpha_A(t))$, where $\alpha_S(t) \in \mathbb{R}^2$ denotes the spatial component of the trajectory and $\alpha_A(t) \in \mathbb{R}^M$ denotes the attributes of the trajectory. The concept of trajectory used in the previous sections of this chapter can be seen as 0-trajectories, i.e., only the spatial component is considered. We now define the counterpart of vector fields in our current setting. An M-attribute field, or simply M-field, is defined as a function $f : \mathbb{R}^2 \to \mathbb{R}^M$. Finally, we say that a trajectory $\overline{\alpha}$ is *consistent* with an attribute field f, if $f(\alpha_S(t)) = \alpha_A(t)$ for all $t \in [t_0, t_1]$. Notice that this notion corresponds to the idea of streamlines in the previous approach. These concepts are illustrated in Figure 4.9 for the case where $M = 1$. Similarly, given a set of M-trajectories $\mathcal{T} = \{\overline{\alpha}_1, \overline{\alpha}_2, ..., \overline{\alpha}_n\}$ we say that an attribute field f is consistent with the set T, if each α_i is consistent with f for $i = 1, 2, ..., n$. Given a vector $x = (x_1, , ..., x_M)$ we denote the d^{th} entry of x by $x^{(d)}$. Thus, given an M-trajectory $\overline{\alpha}$ its d^{th} attribute is denoted by $\alpha_A^(d)$. Similarly, the d^{th} attribute of an M-field, f, if denoted by $f^{(d)}$.

4.6.2 Attribute field k-means Overview

The general goal of attribute field k-means is to capture patterns in trajectory attribute values. The main idea behind it is to do so by exploring the attributes's spatial distribution via attribute fields, as illustrated in the following diagram



As in the case of vector fields (Section 4.3), we use the notion of consistency defined above to model trajectories with attributes. However, as in Section 4.3, for a non-trivial set of trajectories there is no attribute field f consistent with it. Therefore, in order to handle real world scenarios we need to define consistency in an approximate sense.

We do this by defining a measure of how consistent an attribute field f is to a

set of M-trajectories $T = \{\overline{\gamma}_1, \overline{\gamma}_2, ..., \overline{\gamma}_n\}$ as follows

$$Cons_W(f, T) = \sum_{i=1}^{n} \sum_{d=1}^{M} w_d \int_{t_0^{\gamma_i}}^{t_1^{\gamma_i}} \left|\left| f^{(d)}(\gamma_S(t)) - \gamma_A^{(d)}(t) \right|\right|^2 dt, \tag{4.3}$$

where $W = (w_1, ..., w_M)$ denotes a weight vector for the different attributes. By changing these weights, one can express how the consistency of one attribute is preferred compared to the others. For simplicity of notation, we will omit the vector W from formulas in the rest of this section and will assume that weights are fixed for the application in mind.

Given a set of M-trajectories $T = \{\overline{\alpha}_1, \overline{\alpha}_2, ..., \overline{\alpha}_n\}$ and the number of clusters K to be found, we use the notion of approximate consistency (Equation 4.3) to Equation 4.1 as follow:

$$E(f_1, ..., f_K, \Phi) = \sum_{j=1}^{K} \lambda_L \left|\left|\Delta f_j\right|\right|^2 + \frac{(1 - \lambda_L)}{T} \sum_{\overline{\alpha_i} \in \Phi^{-1}(j)} Cons(f_j, \overline{\alpha_i}), \tag{4.4}$$

where $f_1, ..., f_K$ are $M-$attribute fields. As before,$\Phi : \{1, ..., n\} \to \{1, ..., K\}$ is an assignment function and λ_L and T are defined as in Section 4.3. The intuition behind Equation 4.4 is to derive the attribute fields f_j and partition the set of trajectories according the how consistent they are with each of the fields.

4.6.3 Attribute field k-means Overview Fitting Process

To perform the clustering, we perform an optimization process to minimize the function defined in Equation 4.4. In order to do so, similar to the vector field k-means setting we tessellate the M-trajectories so that each segment is contained in a face of the grid G. Similar to the vector field case, attribute fields are defined on the vertices of the grid and linear interpolation is used to compute the values inside the each face of the grid.

We perform an alternating optimization process similar to the one described in Section 4.4. In more details, we also use a two phase optimization process with a *fitting* phase in which the attribute fields are derived from a given partition of

the trajectories and an *assignment* phase where trajectories are assigned to the different clusters given a collection of attribute fields.

In the fitting phase, for each $j = 1, ..., k$, we are given as input a *fixed subset* $\mathcal{T}' = \Phi^{-1}(j)$ of $\mathcal{T}$ and we want to minimize

$$E'(f, \mathcal{T}') = \lambda_L \, ||\Delta f||^2 + \frac{(1 - \lambda_L)}{T} \sum_{\alpha_i \in \mathcal{T}'} Cons(f_j, \overline{\alpha_i}) \tag{4.5}$$

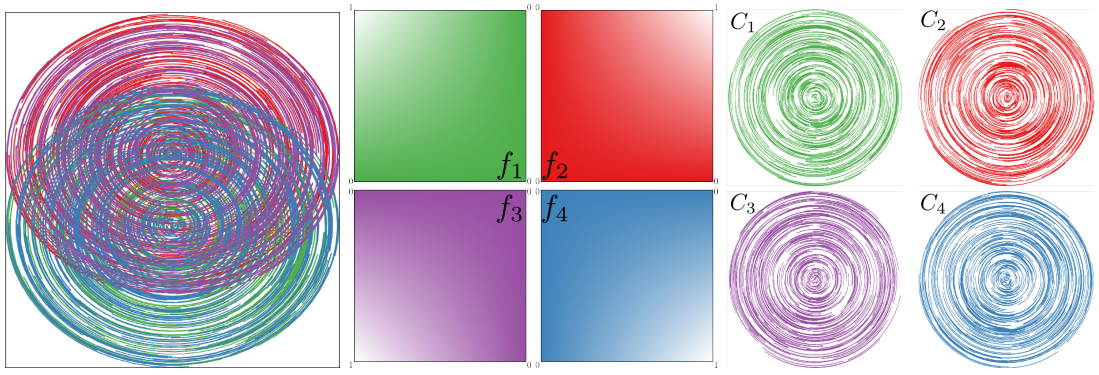
Following the same derivation to the one described in Section 4.4.1, we obtain a least-squares problem for each component f^d of the attribute field f that is similar to the least-squares problem in Section 4.4.

In the assignment phase, we derive the next function Φ that assigns each trajectory to one of the k clusters, i.e. the attribute fields $f_1, \ldots, f_k$. As before, the assignment algorithm is trivial: for each trajectory $\overline{\alpha_i}$, we simply evaluate $Cons(f_j, \overline{\alpha_i})$, for each attribute field f_j, and define the new assignment to be the minimizer for all the k possible choices.

4.6.4 Comments on Attribute field k-means

One important observation is that given a 0-trajectory α it is possible to define a canonical 2-trajectory from it by defining the attribute component the tangent vector of α. More precisely, we can define $\overline{\alpha}$ as $\overline{\alpha}(t) = (\alpha(t), \alpha'(t))$. Given a set of 0-trajectories then we can generate a set of 2-trajectories as described above. It is easy to see that in setting, performing Attribute field k-means corresponds to the performing Vector field k-means in the initial set of trajectories. In this sense, Attribute field k-means can be seen as a strict generalization of Vector field k-means.

For this reason, Attribute field k-means inherits most of the features of Vector field k-means. In particular, it is very dependent on the initialization and we use the same initialization method proposed in Section 4.4.3. Also, the algorithm is guaranteed to converge (Section 4.4.4).



(a) (b) (c)

Figure 4.10: **(a)** The synthetic dataset described in Section 4.5.1 with colors representing groups of trajectories randomly generated within each circular pattern. **(b)** For each trajectory, a new attribute is generated by sampling the corresponding field. **(c)** By considering these attributes, attribute field k-means is able to separate the generated groups.

4.6.5 Sample Experiment

In order to demonstrate how attribute field k-means can recover complex patterns we modify the synthetic dataset described in Section 4.5.1. The modified dataset was obtained by randomly splitting each circular pattern in two groups of equal size Figure 4.10(a). For each of these groups, a scalar attribute is generated by using the corresponding attribute field in Figure 4.10(b). Each of these attribute fields has value equal 1 in one of the corners of the domain and 0 in the remaining ones. Notice that in this setting, vector field k-means is not able to split the different groups. On the other hand, using the same parameters as in Section 4.5.1 and by considering the dataset of 3-trajectories obtained by including the generated attribute together with the trajectories tangent vectors, Attribute field k-means is able to separate the for groups Figure 4.10(c).

4.7 Discussion

In this section, we discuss issues related to parameter selection, advantages and limitations of vector field k-means, as well as possible extensions of our algorithm.

We base the discussion on vector field k-means. As discussed before, attribute field k-means inherits the same features.

4.7.1 Parameter selection

Although we can select both grid resolution R and the weight given to the Laplacian regularization λ_L, the two parameters are not independent. The eigenvectors of the Laplacian are naturally interpreted as equivalent to the fundamental frequencies on the mesh, exactly like sines and cosines are the fundamental frequencies on a circle [120]. The corresponding eigenvalues are the squares of the frequencies themselves, which means that as we increase λ_L, we give larger weights to the eigenvectors corresponding to high-frequency signals, and the system tends towards lower-frequency results. A similar effect is achieved by reducing R, which directly band-limits the signal on the vector field. Using this dependency between these parameters, we can virtually eliminate one of them. As a result, we fix λ_L to be 0.05 and interactively select R in our experiments. The resolution parameter R thus controls the amount of detail in the derived vector fields: by increasing R more complex behavior can be modeled and smaller features have more influence in the clustering. Decreasing R enables the algorithm to filter smaller, possibly insignificant, features and capture global trends. As with most parameters, a clear trade-off exists.

Picking an appropriate number of clusters remains an open problem even in the case of traditional k-means, and we offer no substantive contributions on that matter. Many proposed methods try to attack this problem (see [48] and references therein), however no definitive algorithm solves this problem optimally for all applications in general settings. Still, we stress that as far as performance is concerned, vector field k-means compares quite favorably to results reported in the literature, and thus it is much easier to keep a human analyst in the loop and make cluster count an interactive procedure with vector field k-means than with previous methods.

4.7.2 Advantages

Our model aims at a balance between richness and expressivity of features, and simplicity of implementation and analysis. We believe this is a significant advantage over the current methods for trajectory clustering. As mentioned earlier, by representing the cluster centers as vector fields and using those as a means to define similarity between trajectories, we can eliminate expensive computations of metrics for trajectories and the computations of the centroid trajectory as well [63]. Another advantage is our ability to capture global patterns that are not apparent at the local level in highly noisy trajectories or even partial trajectories.

Vector field k-means is also highly parallelizable. Our prototype does not take advantage of this and includes no significant optimizations, but it is obvious that separate components of vector fields can be computed in parallel, and that many of the intermediate matrices in the linear solvers can be reused from one iteration to the next. We expect these to further increase the performance, and allow vector field k-means to handle even larger datasets.

4.7.3 Limitations

The choice of the initial clusters impact the results achieved by vector field k-means, as it converges to a local minimum and we currently do not offer any guarantees that our result is close to the global optimum. Many techniques have been proposed to choose good initial centers for the clusters in the regular k-means case, one of the most important being the k-means++ approach by Arthur et al. [18], which proves a $\log k$ approximation factor. However it is not yet obvious how to extend this technique to vector field k-means results.

Although vector field k-means is able to naturally encode some attributes such as direction and speed, which are not well handled by previous methods, the notion of spatial distance is not directly represented, which may result in clusters with different behaviors in different spatial regions.

The vector fields derived by our method are steady, and in general, trajectories with self-intersections can lead to artifacts in steady vector fields near these areas. However, our results show that vector field k-means is able to mine interesting patterns even in the presence of self-intersections.

4.8 Summary

We have introduced a novel trajectory clustering technique that brings together ideas from visualization, data clustering, and scalar field design. Vector field k-means can find global patterns, handle partial trajectories, scale to large datasets, and is simple to implement. We also showed an extension of vector field k-means to consider trajectories with attributes, named attribute field k-means. The algorithm opens many possibilities for modeling and user interfaces design, notably for querying and exploring trajectory datasets.

Chapter 5

Conclusions and Future Work

This dissertation presented three main contributions to the area of visual analytics of spatiotemporal data. The first was the design of the TaxiVis visualization system which provides interaction mechanisms for querying large collections of spatiotemporal data (Chapter 2). Furthermore, we presented two pattern mining techniques that were designed to solve problems inspired by limitations of interactive visualization systems for the exploration of spatiotemporal data (Chapter 3 and Chapter 4). Both of these pattern mining techniques apply ideas from scientific visualization and computer graphics which were not used before in the analysis of spatiotemporal data. As described in this thesis, this approach leads to novel techniques that are mathematically well founded, computationally efficient, and the results of which can be understood via previously proposed visual representations.

While many advances have been done in the field of visual analysis of spatiotemporal data, there are still many open problems. Also, new and more sophisticated data sources challenge current technology for data analysis. We highlight some important directions for future research in the following.

Data Management for interactive data analysis. One of the hot topics in visualization research is how to support interactive analysis when datasets are very large. Specialized data layers such as the one presented in Chapter 2 are able to provide interactive query response times for a class of queries even when the dataset in consideration is large. However, as more data become available and more complex queries are needed, more sophisticated data layers will be necessary. In a recent

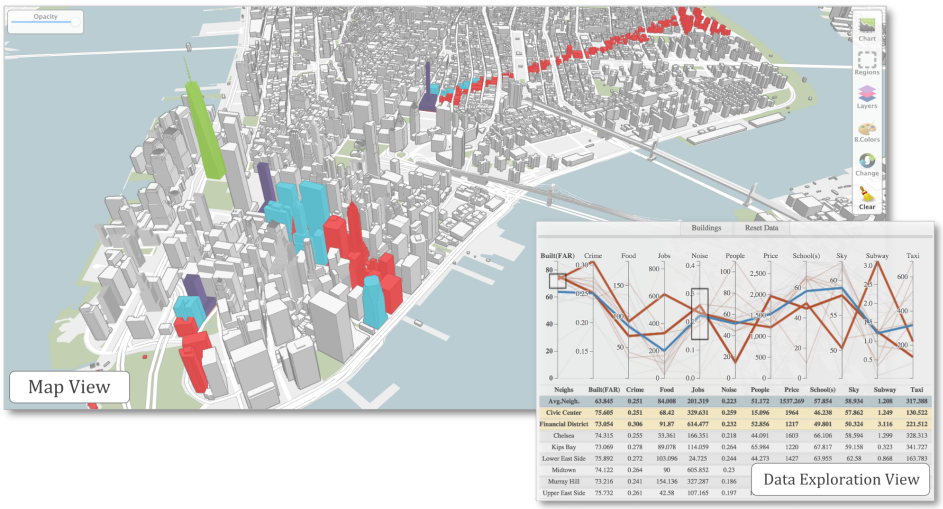


Figure 5.1: Illustration of the interface of the 3D *Urbane* visualization system. Urbane provides architects, developers, and planners with a new, data and analysis rich way of reading the city ultimately facilitating better decision making. Using the visual interface together with the map view, they can simulate the affect of such development. For example, the views to the freedom tower (highlighted in green) of the buildings highlighted in red would be adversely impacted if the new constructions (colored purple) are built.

work Lins et al. [83] proposed the *Nanocubes* data structure. Nanocubes enable interactive responsive times for aggregate queries on extremely large spatiotemporal datasets. This is achieved by an expensive precomputation phase in which all possible queries are performed and their results stored. The main drawbacks of this approach are that they often result in large and static (can not handle updates) data structures. The major competitor to precomputation based approaches in this context is the use of incremental query results to provide interactivity in handling large datasets. Only portions of the dataset are used at each time to produce the visual summaries that are explored by the users. This approach presents some advantages compared to the precomputation based techniques. For example, it can process general queries (not only aggregate ones) and the resulting data collections can, in general, be easily updated. However, by the nature of the process, aspects such as animation and uncertainty are involved in the visualization which are challenging for decision making [50]. One interesting direction for research is how to design data layers that combine the good aspects of both approaches.

Visualization of Spatial/Spatiotemporal 3D data. For a long time, 3D visualizations have been avoided in information visualization and visual analytics due the problems such as occlusion and navigation. However, new applications require actual representation of 3D spatial data such as real world geometries. This feature which used to be only of concern in scientific visualization, is now of great importance in visual analytics. For example, spatial data in urban datasets such as price of properties and noise complaints in large cities can be mapped not only with respect to latitude and longitude, but also with respect to the buildings' geometries. In this scenario, the usual problems for visualization in 3D need to be faced. Therefore, the main challenge is how to create effective representations and interactions that can be easily used by users, while keeping the 3D nature of the data evident. Furthermore, these visualizations should be effective not only for data exploration (when they are used by the experts), but for communication, since many of them are going to be used by experts to present to their clients and collaborators. Figure 5.1 shows our first research effort [52], the Urbane system. This type of application has not received much attention from the research community. Therefore, there is a need for more research efforts that explore the design space of such visualizations and how users will interact with those. The final objective is to understand in which cases should 3D visualizations should/can be used.

The role of pattern mining techniques in real world analysis scenarios. Understanding how pattern mining techniques could be integrated and used in real world visualization systems is a very important research question. The main challenge here is in integrating these techniques in real systems in a transparent way so that users do not need to understand their details and also are not deviated from their main analysis tasks. This is a difficult problem since, for example, techniques such as *event guided exploration* and vector field k-means are based on complex concepts that might not be known for many potential users of those techniques. However, in such scenarios it is desirable that users can not only understand the results of the computation, but also leverage domain knowledge to provide feed back. This can help, for example, in defining parameters and improve results of those techniques. More traditional techniques such as *K-means clustering* [108]

and *self-organizing maps* [64] have been applied in real world system with success. However, more research is required to have the same success with recent techniques such as the ones presented in this thesis.

Appendix A

Detailed Derivation of Matrix Form for Vector Field K-Means Fitting Step

In the following we include a detailed explanation of the derivation of the matricial form of Equation 4.2 of the paper and in particular explain the need for the matrix Λ. The idea is to convert the problem in a finite dimensional least-squares problem, *i.e.*, we want to write the energy E' in the form $||Dx - c||^2$, for some matrix D and some vector c.

Assume we have a vector field X defined over a grid G as before and also we are given a trajectory $\alpha : [a, b] \to \mathbb{R}^2$ and let $a = t_0 < t_1 < ... < t_{n-1} < t_N = b$, be the partition of the interval $[a, b]$ where each point of the partition is either a vertex of α or a point where this trajectory touches the grid (all the vertices and grid intersections are included). The first summand in E' (Equation 4.2) corresponds to the Laplacian matrix and its matricial form is simply given by $||LX||^2$, where L is the Laplacian matrix of the grid, defined in Section 4.4.1. The second summand in E' is given by:

$$||X \circ \alpha - \alpha'||^2_{L^2} = \int_a^b ||X(\alpha(t)) - \alpha'(t)||dt =$$
$$\sum_{i=1}^{n-1} \int_{t_i}^{t_{i+1}} ||X(\alpha(t)) - \alpha'(t)||^2_{L^2} = \sum_{i=1}^{n-1} \int_{t_i}^{t_{i+1}} ||X(\alpha(t)) - \alpha_i'||^2_{L^2},$$

where α_i' denotes the velocity vector of the i^{th} segment of α. Given this fact, we can proceed with the following derivation:

$$||X \circ \alpha - \alpha'||^2_{L^2} = \sum_{i=1}^{n-1} \int_{t_i}^{t_{i+1}} ||X(\alpha(t)) - \alpha_i'||^2_{L^2} =$$
$$\sum_{i=1}^{n-1} (t_{i+1} - t_i) \int_0^1 ||(1-\sigma)X(\alpha(t_i)) + \sigma X(\alpha(t_{i+1})) - \alpha_i'||^2 d\sigma =$$
$$\sum_{i=1}^{n-1} (t_{i+1} - t_i) \int_0^1 ||(1-\sigma)(X(\alpha(t_i)) - \alpha_i') + \sigma(X(\alpha(t_{i+1})) - \alpha_i')||^2 d\sigma =$$
$$\sum_{i=1}^{n-1} (t_{i+1} - t_i)[\tfrac{1}{3} ||X(\alpha(t_i)) - \alpha_i'||^2 + \tfrac{1}{3} ||X(\alpha(t_{i+1})) - \alpha_i'||^2 +$$
$$\tfrac{1}{3} \langle X(\alpha(t_i)) - \alpha_i'), X(\alpha(t_{i+1})) - \alpha_i' \rangle].$$

Therefore, we can see that the 3 terms appear with the coefficients $\frac{1}{3}$, and the matrix Λ makes this coefficient appear in $||\widetilde{C}X - \widetilde{b}||^2$, since

$$\Lambda^T \Lambda = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}.$$

Bibliography

[1] HURDAT: The national hurricane center's north atlantic hurricane database, Feb 2012.

[2] P. K. Agarwal, H. Edelsbrunner, J. Harer, and Y. Wang. Extreme Elevation on a 2-manifold. *Disc. Comput. Geom.*, 36(4):553–572, 2006.

[3] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: An implementation and evaluation. In *CHI*, pages 619–626, 1992.

[4] G. Andrienko and N. Andrienko. Spatio-temporal Aggregation for Visual Analysis of Movements. In *Proc. of IEEE VAST*, pages 51–58, 2008.

[5] G. Andrienko and N. Andrienko. Spatio-temporal aggregation for visual analysis of movements. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 51–58, 2008.

[6] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel. Visual Analytics Focusing on Spatial Events. In *Visual Analytics of Movement*, pages 209–251. Springer Berlin Heidelberg, 2013.

[7] G. Andrienko, N. Andrienko, G. Fuchs, A.-M. O. Raimond, J. Symanzik, and C. Ziemlicki. Extracting Semantics of Individual Places from Movement Data by Analyzing Temporal Patterns of Visits. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place*, COMP '13, pages 9:9–9:16. ACM, 2013.

[8] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From movement tracks through events to places: Extracting and characterizing

significant places from mobility data. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 161–170, 2011.

[9] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 159–168, 2011.

[10] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. Scalable Analysis of Movement Data for Extracting and Exploring Significant Places. *IEEE TVCG*, 19(7):1078–1094, July 2013.

[11] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti. Interactive visual clustering of large collections of trajectories. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 3–10, 2009.

[12] G. Andrienko, N. Andrienko, H. Schumann, and C. Tominski. Visualization of trajectory attributes in space-time cube and trajectory wall. In *Cartography from Pole to Pole*, pages 157–163. Springer, 2014.

[13] G. Andrienko, N. Andrienko, and S. Wrobel. Visual analytics tools for analysis of movement data. *SIGKDD Explorations*, 9(2):38–46, 2007.

[14] N. Andrienko and G. Andrienko. Designing Visual Analytics Methods for Massive Collections of Movement Data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 42(2):117–138, 2007.

[15] N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Trans. Visualization and Computer Graphics*, 17(2):205–219, 2011.

[16] N. Andrienko and G. Andrienko. Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization*, 12(1):3–24, 2013.

[17] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[18] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[19] T. F. Banchoff. Critical Points and Curvature for Embedded Polyhedral Surfaces. *Am. Math. Monthly*, 77:475–485, 1970.

[20] L. Barbosa, K. Pham, C. Silva, M. R. Vieira, and J. Freire. Structured Open Urban Data: Understanding the Landscape. *Big Data*, 2(3):144–154, 2014.

[21] M. Bayir, M. Demirbas, and N. Eagle. Discovering spatiotemporal mobility profiles of cellphone users. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops*, pages 1–9, 2009.

[22] R. A. Becker, R. Caceres, K. Hanson, J. M. Loh, S. Urbanek, A. Varshavsky, and C. Volinsky. Route classification using cellular handoff patterns. In *13th International Conference on Ubiquitous Computing*, pages 123–132, 2011.

[23] I. Boyandin, E. Bertini, P. Bak, and D. Lalanne. Flowstrates: An approach for visual exploration of temporal origin-destination data. *Computer Graphics Forum*, 30(3):971–980, 2011.

[24] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. *IEEE TVCG*, 16(2):248–260, Mar. 2010.

[25] D. Brillinger, H. Preisler, A. Ager, and J. Kie. An exploratory data analysis (EDA) of the paths of moving animals. *Statistical Planning and Inference*, 122(2):43–63, 2004.

[26] H. Bunke and K. Shearer. A Graph Distance Metric Based on the Maximal Common Subgraph. *Pattern Recogn. Lett.*, 19(3):255–259, 1998.

[27] S. J. Camargo, A. W. Robertson, S. J. Gaffney, P. Smyth, and M. Ghil. Cluster analysis of typhoon tracks. *Climate*, 20(14):3635–3676, 2007.

[28] H. Carr, J. Snoeyink, and U. Axen. Computing Contour Trees in All Dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.

[29] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *Proc. IEEE Visualization*, pages 497–504, 2004.

[30] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[31] R. Chang, G. Wessel, R. Kosara, E. Sauda, and W. Ribarsky. Legible Cities: Focus-Dependent Multi-Resolution Visualization of Urban Relationships. *IEEE TVCG*, 13(6):1169–1175, 2007.

[32] G. Chen, V. Kwatra, L. Wei, C. Hansen, and E. Zhang. Design of 2d time-varying vector fields. *IEEE Trans. Visualization and Computer Graphics*, 18(10):1717–1730, 2012.

[33] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and Optimal Output-Sensitive Construction of Contour Trees Using Monotone Paths. *Comput. Geom. Theory Appl.*, 30(2):165–195, 2005.

[34] K. A. Cook and J. J. Thomas. Illuminating the Path: The Research and Development Agenda for Visual Analytics. May 2005.

[35] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[36] O. Daae Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *IEEE Pacific Visualization Symposium*, pages 171–178, 2011.

[37] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer, 1997.

[38] H. Doraiswamy, N. Ferreira, T. Damoulas, J. Freire, and C. Silva. Using Topological Analysis to Support Event-Guided Exploration in Urban Data. *IEEE TVCG*, 20(12):2634–2643, 2014.

[39] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An Exploration Framework to Identify and Track Movement of Cloud Systems. *IEEE TVCG*, 19(12):2896–2905, 2013.

[40] A. DUlizia, F. Ferri, and P. Grifoni. Moving geopql: a pictorial language towards spatio-temporal queries. *GeoInformatica*, 16(2):357–389, 2012.

[41] H. Edelsbrunner and J. Harer. Persistent Homology — A Survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry. Twenty Years Later*, pages 257–282. Amer. Math. Soc., Providence, Rhode Island, 2008. Contemporary Mathematics 453.

[42] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2009.

[43] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale Complexes for Piecewise Linear 3-Manifolds. In *Proc. Symp. Comput. Geom.*, pages 361–370, 2003.

[44] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. *Disc. Comput. Geom.*, 28(4):511–533, 2002.

[45] R. Edsall and D. Peuquet. A graphical user interface for the integration of time into GIS. In *Proceedings of the American Cong. of Surveying and Mapping Annual Convention and Exhibition, Seattle, WA*, pages 182–189, 1997.

[46] J. Elsner. Tracking hurricanes. *Bulletin of the American Meteorological Society*, 84(3):353–356, 2003.

[47] M. Erwig and M. Schneider. Query-by-trace: Visual predicate specification in spatio-temporal databases. *Advances in Visual Information Management–Visual Database Systems*, pages 199–218, 2000.

[48] Y. Fang and J. Wang. Selection of the number of clusters via the bootstrap method. *Comput. Stat. Data Anal.*, 56(3):468–477, Mar. 2012.

[49] J.-D. Fekete and C. Silva. Managing data for visual analytics: Opportunities and challenges. *IEEE Data Engineering Bulletin*, 35(3):27–36, 2012.

[50] N. Ferreira, D. Fisher, and A. C. König. Sample-oriented task-driven visualizations: Allowing users to make better, more confident decisions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 571–580. ACM, 2014.

[51] N. Ferreira, J. Klosowski, C. Scheidegger, and C. Silva. Vector Field K-Means: Clustering Trajectories by Fitting Multiple Vector Fields. *Computer Graphics Forum*, 32(3pt2):201–210, 2013.

[52] N. Ferreira, M. Lage, H. Doraiswamy, H. Vo, L. Wilson, H. Werner, M. C. Park, and C. Silva. Urbane: A 3D Framework to Support Data Driven Decision Making in Urban Development. In *Submitted to Vast 2015*.

[53] N. Ferreira, L. Lins, D. Fink, S. Kelling, C. Wood, J. Freire, and C. Silva. BirdVis: Visualizing and understanding bird populations. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2374–2383, 2011.

[54] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual Exploration of Big Spatio-temporal Urban Data: A Study of New York City Taxi Trips. *IEEE TVCG*, 19(12):2149–2158, 2013.

[55] F. Ferri and M. Rafanelli. Geopql: a geographical pictorial query language that resolves ambiguities in query interpretation. In *Journal on Data Semantics III*, pages 50–80. Springer, 2005.

[56] D. Fisher, I. Popov, S. M. Drucker, and mc schraefel. Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster. In *Proceedings of the 2012 Conference on Human Factors in Computing Systems (CHI 2012)*. ACM Conference on Human Factors in Computing Systems, May 2012.

[57] A. T. Fomenko and T. L. Kunii, editors. *Topological Modeling for Visualization*. Springer Verlag, 1997.

[58] H. L. Gantt. *Work, Wages, and Profits*. Engineering Magazine Co., 1913.

[59] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *SIGKDD*, pages 899–908. ACM, 2010.

[60] G. Groth, N. Ferreira, C. Scheidegger, J. Comba, L. G. Nonato, and C. Silva. Attribute Field K-Means: Clustering Trajectories with Attributes by Fitting Multiple Fields. In *Submitted to InfoVis 2015*.

[61] E. Grundy, M. W. Jones, R. S. Laramee, R. P. Wilson, and E. L. Shepard. Visualisation of sensor data from animal movement. In *Computer Graphics Forum*, volume 28, pages 815–822. Wiley Online Library, 2009.

[62] Y. Gu and C. Wang. itree: Exploring Time-Varying Data Using Indexable Tree. In *IEEE PacificVis*, pages 137–144, 2013.

[63] J. Gudmundsson, P. Laube, and T. Wolle. Computational Movement Analysis. In *Springer Handbook of Geographic Information*, pages 423–438. Springer, 2012.

[64] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (vis-stamp). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, Nov. 2006.

[65] D. Guo and X. Zhu. Origin-destination flow data smoothing and mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2043–2052, Dec 2014.

[66] S. Hahmann and D. Burghardt. How much information is geospatially referenced? Networks and cognition. *International Journal of Geographical Information Science*, 27(6):1171–1189, 2013.

[67] A. Hatcher. *Algebraic Topology*. Cambridge U. Press, New York, 2002.

[68] J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 959–968. ACM, 2008.

[69] M. Hoai and F. De la Torre. Max-Margin Early Event Detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2863–2870, 2012.

[70] T.-A. Hoang-Vu, V. Been, I. G. Ellen, M. Weselcouch, and J. Freire. Towards Understanding Real-Estate Ownership in New York City: Opportunities and Challenges. In *Proceedings of the International Workshop on Data Science for Macro-Modeling*, DSMM'14, pages 15:1–15:2. ACM, 2014.

[71] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 665–674, New York, NY, USA, 2013. ACM.

[72] H. Jänicke, T. Weidner, D. Chung, R. S. Laramee, P. Townsend, and M. Chen. Visual reconstructability as a quality metric for flow visualization. In *Computer Graphics Forum*, volume 30, pages 781–790. Wiley Online Library, 2011.

[73] J. Kasten, I. Hotz, B. Noack, and H.-C. Hege. Vortex merge graphs in two-dimensional unsteady flow fields. In *EuroVis - Short Papers*, pages 1–5, 2012.

[74] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo. Spatio-temporal clustering. In *Data Mining and Knowledge Discovery Handbook*, pages 855–874. Springer US, 2009.

[75] R. Kruger, D. Thom, and T. Ertl. Semantic Enrichment of Movement Behavior with Foursquare - A Visual Analytics Approach. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1–1, 2014.

[76] R. Krger, D. Thom, M. Wrner, H. Bosch, and T. Ertl. Trajectorylenses a set-based filtering and exploration technique for long-term trajectory data. *Computer Graphics Forum*, 32(3pt4):451–460, 2013.

[77] M. Kulldorff. A Spatial Scan Statistic. *Communications in Statistics-Theory and methods*, 26(6):1481–1496, 1997.

[78] M. Kulldorff, F. Mostashari, L. Duczmal, W. Katherine Yih, K. Kleinman, and R. Platt. Multivariate Scan Statistics for Disease Surveillance. *Statistics in Medicine*, 26(8):1824–1833, 2007.

[79] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities. *IEEE TVCG*, 12(5):1053–1060, Sept. 2006.

[80] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007.

[81] X. Liang, X. Zheng, W. Lv, T. Zhu, and K. Xu. The scaling of human mobility by taxis is exponential. *Physica A: Statistical Mechanics and its Applications*, 391(5):2135–2144, 2012.

[82] Z. Liao, Y. Yu, and B. Chen. Anomaly detection in gps data based on visual analytics. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 51–58, 2010.

[83] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE TVCG*, 19(12):2456–2465, 2013.

[84] Z. Liu and J. Heer. The Effects of Interactive Latency on Exploratory Visual Analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2122–2131, Dec 2014.

[85] S. Maadasamy, H. Doraiswamy, and V. Natarajan. A Hybrid Parallel Algorithm for Computing and Tracking Level Set Topology. In *Proc. Intl. Conf. High Performance Computing*, pages 12.1–12.10, 2012.

[86] A. M. MacEachren and M.-J. Kraak. Exploratory Cartographic Visualization: Advancing the Agenda. *Computers & Geosciences*, 23(4):335–343, May 1997.

[87] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, M. Wade, and D. S. Ebert. Understanding Syndromic Hotspots-A Visual Analytics Approach. In *Proc. of IEEE VAST 2008*, pages 35–42. IEEE, 2008.

[88] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers, and M. G. Institute. Big data: The next frontier for innovation, competition, and productivity. 2011.

[89] E. McFowland III, S. Speakman, and D. B. Neill. Fast Generalized Subset Scan for Anomalous Pattern Detection. *Journal of Machine Learning Research*, 14:1533–1561, 2013.

[90] V. Melnykov and I. Melnykov. Initializing the {EM} algorithm in gaussian mixture models with an unknown number of components. *Computational Statistics & Data Analysis*, 56(6):1381 – 1395, 2012.

[91] J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

[92] NYC MTA API. http://web.mta.info/developers/.

[93] J. C. Nascimento, M. A. T. Figueiredo, and J. S. Marques. Trajectory analysis in natural images using mixtures of vector fields. In *IEEE International Conference on Image Processing*, pages 4353–4356, 2009.

[94] D. B. Neill and G. F. Cooper. A Multivariate Bayesian Scan Statistic for Early Event Detection and Characterization. *Machine learning*, 79(3):261–282, 2010.

[95] D. B. Neill, A. W. Moore, and G. F. Cooper. A Bayesian Spatial Scan Statistic. *Adv. Neur. In.*, 18:1003, 2006.

[96] J. Nocedal and S. Wright. *Numerical optimization*. Springer Verlag, 1999.

[97] G. Oliveira, J. Comba, R. Torchelsen, M. Padilha, and C. Silva. Visualizing running races through the multivariate time-series of multiple runners. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on*, pages 99–106. IEEE, 2013.

[98] C. Olston, M. Stonebraker, A. Aiken, and J. Hellerstein. Viqing: visual interactive querying. In *Visual Languages*, pages 162–169, 1998.

[99] G. Pan, G. Qi, Z. Wu, D. Zhang, and S. Li. Land-use classification using taxi gps traces. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):113–123, 2013.

[100] V. Pascucci and K. Cole-McLaughlin. Parallel Computation of the Topology of Level Sets. *Algorithmica*, 38(1):249–268, 2003.

[101] V. Pascucci and R. J. Frank. Global static indexing for real-time exploration of very large regular grids. In *Supercomputing, ACM/IEEE 2001 Conference*, pages 45–45. IEEE, 2001.

[102] V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, editors. *Topological Methods in Data Analysis and Visualization*. Springer, 2010.

[103] V. Pascucci, G. Weber, J. Tierny, P.-T. Bremer, M. Day, and J. Bell. Interactive Exploration and Analysis of Large-Scale Simulations Using Topology-Based Data Segmentation. *IEEE TVCG*, 17(9):1307–1324, 2011.

[104] N. Pelekis, I. Kopanakis, E. Kotsifakos, E. Frentzos, and Y. Theodoridis. Clustering trajectories of moving objects in an uncertain world. In *Ninth IEEE International Conference on Data Mining*, pages 417–427, 2009.

[105] C. Peng, X. Jin, K.-C. Wong, M. Shi, and P. Liò. Collective human mobility pattern from taxi trips in urban area. *PloS one*, 7(4):e34487, 2012.

[106] D. Peuquet. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. *Annals of the Association of american Geographers*, 84(3):441–461, 1994.

[107] D. Phan, L. Xiao, R. Yeh, and P. Hanrahan. Flow map layout. In *Proceedings of InfoVis 2005*, pages 219–224, 2005.

[108] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko. Visually driven analysis of movement data by progressive clustering. *Information Visualization*, 7(3-4):225, 2008.

[109] R. E. Roth. An Empirically-Derived Taxonomy of Interaction Primitives for Interactive Cartography and Geovisualization. *IEEE TVCG*, 19(12):2356–2365, 2013.

[110] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk. Composite density maps for multivariate trajectories. *IEEE Trans. Visual. and Computer Graphics*, 17(12):2518–2527, 2011.

[111] R. W. Scholz and Y. Lu. Detection of Dynamic Activity Patterns at a Collective Level from Large-Volume Trajectory Data. *International Journal of Geographical Information Science*, (ahead-of-print):1–18, 2014.

[112] S. A. Shaheen, S. Guzman, and H. Zhang. Bikesharing in europe, the americas, and asia. *Transportation Research Record: Journal of the Transportation Research Board*, 2143(1):159–167, 2010.

[113] Y. B. Shrinivasan, J. Van Wijk, et al. Supporting exploratory analysis with the select & slice table. *Computer Graphics Forum*, 29(3):803–812, 2010.

[114] C. D. Smith. Cartography in the prehistoric period in the Old World: Europe, the Middle East, and North Africa. *History of Cartography*, 1:54–107, 1987.

[115] O. Sorkine, D. Cohen-Or, D. Irony, and S. Toledo. Geometry-aware bases for shape approximation. *IEEE Trans. Visualization and Computer Graphics*, 11(2):171–180, 2005.

[116] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[117] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A Survey of Visual Analytics Techniques and Applications: State-of-the-Art Research and Future Challenges. *J. of Comp. Sci. and Tech.*, 28(5):852–867, 2013.

[118] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2565–2574, 2012.

[119] Twitter API. https://dev.twitter.com/.

[120] B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Proceedings Eurographics)*, 2008.

[121] J. J. van Wijk. Image based flow visualization. *ACM Trans. Graph.*, 21(3):745–754, July 2002.

[122] M. Veloso, S. Phithakkitnukoon, and C. Bento. Urban mobility study using taxi traces. In *Proceedings of the 2011 international workshop on Trajectory data mining and analysis*, TDMA '11, pages 23–30. ACM, 2011.

[123] M. Veloso, S. Phithakkitnukoon, C. Bento, P. Olivier, and N. Fonseca. Exploratory study of urban flow using taxi traces. In *First Workshop on Pervasive Urban Applications (PURBA)*, 2011.

[124] K. Verbeek, K. Buchin, and B. Speckmann. Flow map layout via spiral trees. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2536–2544, 2011.

[125] J. Wakefield and A. Kim. A Bayesian Model for Cluster Detection. *Biostatistics*, 14(4):752–765, 2013.

[126] F. Wang, W. Chen, F. Wu, Y. Zhao, H. Hong, T. Gu, L. Wang, R. Liang, and H. Bao. A visual reasoning approach for data-driven transport assessment on urban roads. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pages 103–112, Oct 2014.

[127] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. v. d. Wetering. Visual Traffic Jam Analysis Based on Trajectory Data. *IEEE TVCG*, 19(12):2159–2168, 2013.

[128] C. Weaver. Multidimensional visual analysis using cross-filtered views. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 163–170, 2008.

[129] W. Widanagamaachchi, C. Christensen, P.-T. Bremer, and V. Pascucci. Interactive Exploration of Large-Scale Time-Varying Data Using Dynamic Tracking Graphs. In *Proc. of IEEE LDAV*, pages 9–17, 2012.

[130] N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of vessel movements. *Computer Graphics Forum (Proceedings of EuroVis)*, 28(3):959–966, 2009.

[131] J. Wood, J. Dykes, and A. Slingsby. Visualisation of origins, destinations and flows with od maps. *Cartographic Journal, The*, 47(2):117–129, 2010.

[132] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *Proceedings of the 13th international conference on Ubiquitous computing.*, pages 109–118. ACM, 2011.

[133] Y. Zheng, X. Xie, and W. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–40, 2010.