

Trajectory Analysis: Bridging Algorithms and Visualization

Maximilian Konzack

Trajectory Analysis: Bridging Algorithms and Visualization

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op woensdag 9 mei 2018 om 16:00 uur

door

Maximilian Peter Konzack

geboren te Schwabach, Duitsland

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter: prof.dr. J.J. Lukkien

promotor: prof.dr. M.T. de Berg

copromotor: dr. K.A. Buchin

copromotor: dr. M.A. Westenberg

leden:
prof.dr. A. Kerren (Linnaeus University)
dr.ir. E.E. van Loon (University of Amsterdam)
prof.dr. R. Weibel (University of Zurich)
prof.dr.ir. J.J. van Wijk

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.



The work in this thesis is supported by the Netherlands' Organization for Scientific Research (NWO) under project no. 612.001.207.



The work in the thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Cover design: H.N. Adams © 2018. All rights reserved.
Used with permission.

Printing: Ipkamp Printing

ISBN: 978-90-386-4493-6

© 2018 by Maximilian Peter Konzack. All rights are reserved. Reproduction in whole or in part is prohibited without the written consent of the copyright owner.

A catalogue record is available from the
Eindhoven University of Technology Library

TU/e Technische Universiteit
Eindhoven
University of Technology

Black
then
white are
all I see
in my infancy
Red and yellow then came to be
reaching out to me
lets me see
there is
so
much
more and
beckons me
to look through to these
infinite possibilities
As below, so above and beyond, I imagine
drawn outside the lines of reason
Push the envelope
Watch it bend

Tool - Lateralus

Acknowledgments

Thank you all
Cannot put in words
Gratitude

At this point, I am looking back on this memorable phase of my life. It has been blissful, challenging, rewarding, difficult at times, and most importantly satisfying and foremost fulfilling. Becoming a researcher has been a journey that influenced me in ways that I would have never imagined. To me, a PhD is not merely a degree as an intellectual exercise; it is much more than that. I have had the honor to practice a variety of skills: reading, programming, writing drafts, defining research questions, discarding drafts, reviewing papers, incorporating feedback, giving presentations, writing emails, editing – yes, a lot of editing – and, most importantly, talking to my supervisors.

First and foremost, I full-heartedly thank my supervisors Kevin Buchin and Michel Westenberg. Kevin and Michel were much more than academic mentors to me. They had an open ear and an open door whenever it was necessary: when our first paper was accepted, when one of our papers was rejected, when that paper was finally accepted, and also on personal matters. I learned from you how to walk in academia and how practical and theoretical computer science enhance each other. I owe you a wholehearted thank you, Kevin and Michel.

I am also very grateful that Mark de Berg was my promoter. Even though Mark was not involved directly in the research of this thesis, he supported me by discussing ideas and by handling the organizational matters around my PhD project. I cannot imagine a more experienced promoter who always listened to the concern at hand very carefully. Thank you, Mark.

By the interdisciplinary nature of my PhD project, I have had the honor of collaborating with a wide variety of researchers. I would thank all of my collaborators for their help and feedback: Maike Buchin, Panos Giannopoulos, Pieter Gijsbers, Luca Giuggioli, Joachim Gudmundsson, Jed Long, Emiel van Loon, Thomas McKetterick, Wolfgang Mulzer, Trisalyn Nelson, Tim Ophelders, Wim Reddingius, André Schulz, Ferry Timmers, and Georgina Wilcox.

Furthermore, I want to thank all reviewers of my papers for their time, effort, and feedback which greatly improved the content of this thesis.

I would like to thank my committee for thoroughly reading my thesis and for their time serving in my defense: Andreas Kerren, Emiel van Loon, Johan Lukkien, Robert Weibel, and Jack van Wijk.

In my time at the TU Eindhoven, I have had the pleasure to be surrounded by many smart and excellent scientists. I thank all of my colleagues and guest researchers: Hans Bodlaender, Quirijn Bouts, Riet van Bull, Rafael Cano, Bram Cappers, Thom Castermans, Meivan Cheng, Paul van der Corput, Alberto Corvo, Kasper Dinkla, Huib Donkers, Anne Driemel, Stef van Elzen, Arthur van Goethem, Herman Haverkort, Michael Horton, Andrei Jalba Bart Jansen, Sándor Kisfaludi-Bak, Sudeshna Kolay, Robert van Liere, Aleksandar Markovic, Mehran Mehr, Ali Mehrabi, Wouter Meulemans, Dániel Oláh, Tim Ophelders, Astrid Pieterse, Jorn van der Pol, Marcel Roelofzen, Ignaz Rutter, Roeland Scheepens, Max Sondag, Willem Sonke, Bettina Speckmann, Mehmodoud Talebi, Kevin Verbeek, Mickael Verschoor, Huub van de Wetering, Tim Willemse, Jack van Wijk, Gehard Woeginger, and Jules Wulms. Especially, I want to thank my office mates, Q, Ali, Irina, Rafael, Thom, Max, and Huib, for all the fun times and for all the hours that we spent with each other¹.

According to an African proverb, “it takes a village to raise a child”. I feel lucky and happy for all fellow human beings who have accompanied me through different stages in my life: Heather Adams, Jakob Albert, Dorothea Eisenmann, Sebastian Faubel, Elfriede Fehr, Maximilian Freidl, Simone Geißelsöder, Stefan Geißelsöder, Ruud Hagen,

¹No office mates were locked in our office intentionally.

Markus Harrer, Vera Harrer, Peter Hofmann, Johannes Klick, Irena Kpogbezan, Aljoscha Krettek, Christian Kuschel, Michelle Meekes, Bianca Mikolajewski, Francisco Montellano, Peter Neubauer, Johanna Neubauer, Joana Pedro, Niels Peekstok, Rena Peters, Florian Policnik, Marcus Raab, Tobias Reichard, Sonja Schlusche, Fabian Schneider, Gustav Stahlke, Christian Vanderheijden, Andreas Weinlein, Andy Wong, and Michelle Zeuner. I am thankful that you accept me for who I am and for the time that we have shared with each other.

I strongly believe that all my strengths, shortcomings, talents, and other qualities have a single origin: my family. You shaped me like no one else. Your pragmatism, discipline, curiosity, tenacity, and patience helped me to be who I am and to pursue this degree. I am grateful for all of the past generations and current members of my family. In particular, I want to thank my parents, Gabriele Konzack and Eberhard Konzack, for providing a good home and raising me to be the person that I am. Danke, Mama und Papa!

I want to thank my math teacher, Frau Böer, who encouraged me fifteen years ago to move away from economics toward something more technical like mathematics or computer science. I think I finally listened. Vielen Dank, Frau Böer!

Finally, I thank all the people who directly and indirectly helped me reach this milestone in my life. Within this limited amount of space, it is impossible to name them all, which is why I thank you all.

Last but not least, I want to thank you, reader, for reading the results of my PhD journey.

*Maximilian Peter Konzack
Eindhoven, April 2018*

Contents

Acknowledgments	ii
1 Introduction	1
1.1 Computational Movement Analysis	2
1.2 Algorithms	3
1.3 Visualization	5
1.4 Trajectory Analysis	6
1.5 Overview	8
2 Background	11
2.1 Introduction	11
2.2 Related Work	12
2.3 Scope and Focus	13
2.4 Typology for Analyzing Movement Data	14
2.5 Overview of Trajectory Analysis Tasks	18
2.5.1 Alignment	18
2.5.2 Transform	21
2.5.3 Categorization	24
2.5.4 Representation	28
2.6 Discussion	31
2.6.1 Workflow	31
2.6.2 Scale and Uncertainty	31
2.6.3 Context	32
2.6.4 Space and Time	32
2.6.5 Interdisciplinarity	33
2.6.6 Specificity of Data Analysis	33
2.7 Conclusions	34
3 Computational Complexity of Problems on Trajectories	35
3.1 Introduction	35
3.2 Preliminaries	36

3.3	Single Curve Problems	39
3.3.1	Simplification	39
3.3.2	Lower Bound on Simplification	44
3.4	Problems on Two Trajectories	48
3.4.1	Overview	48
3.4.2	Alignment Methods	49
3.5	Problems on Multiple Trajectories	53
3.5.1	Overview	53
3.5.2	Lower Bound on the Fréchet Distance	55
3.6	Conclusions	59
4	Progressive Simplification	61
4.1	Introduction	61
4.2	Related Approaches	64
4.3	Computing Simplifications Progressively	65
4.3.1	Optimal Progressive Simplifications	66
4.3.2	Greedy Heuristics	71
4.4	Constructing the Shortcut Graph for Arbitrary Scale	72
4.5	Compressing the Shortcut Graph	74
4.5.1	Shortcut Graph Construction	75
4.5.2	Finding Shortest Paths	76
4.6	Experimental Evaluation	77
4.7	Conclusions	81
5	Visual Analytics of Delays and Interaction	83
5.1	Introduction	83
5.2	Interaction and Similarity Measures	87
5.3	Fast Computation of Global Delays	91
5.3.1	Correlations and the Fast Fourier Transform	92
5.3.2	Approximation of Similarity Measures	93
5.4	Visual Analytics for Local Analysis of Delays	95
5.4.1	Requirements for Analyzing Interactions	96
5.4.2	Computing Matchings	96
5.4.3	Interactive Analysis of Delays in Matchings	99
5.5	Experiments	103
5.5.1	Analysis of the Global Delay	104
5.5.2	Analysis of Delays on Ultimate Frisbee Data	107
5.5.3	Analysis of Delays on Pigeon Data	110

5.5.4 Analysis of Delays on a Triplet of Pigeons	112
5.6 Conclusions	116
6 Visual Exploration of Migration Patterns in Gull Data	119
6.1 Introduction	119
6.2 Related Work	122
6.3 Problem Definition and Requirement Analysis	125
6.3.1 Ecological Research Questions on Migration	126
6.3.2 Requirements for Analysis Tasks.	127
6.4 Visual Analytics Approach	129
6.4.1 Computational Methods	130
6.4.2 Visualization Techniques.	132
6.5 Exploratory Analysis Process	134
6.6 Evaluation	136
6.6.1 Dataset of Migrating Gulls	137
6.6.2 Expert User Evaluation	138
6.6.3 Reflections	145
6.7 Conclusions	147
7 Conclusion	148
7.1 Contributions	149
7.2 Looking Forward	151
References	153
List of Publications	171
Summary	172
Curriculum Vitæ	175

1

Introduction

Everything moves. Humans move from home to work; animals move to forage; packages move from warehouses to customers; hurricanes emerge over water and cause a path of devastation when they hit land; this enumeration of examples of movement can be extended by many more. Consequently, movement analysis is central to understanding the causal links in time and space between a moving entity and its surrounding resources. Since movement is so ubiquitous, movement analysis has many applications.

One such application is *movement ecology*. Movement ecology is an interdisciplinary scientific area which aims at understanding cues and causes related to movement, specifically in regard to an organism and its environment. Understanding an organism's movement concerns internal factors – why, how, when and where an organism moves to – and external factors that link an organism to its environment [Nathan et al., 2008].

Tracking individuals manually is tedious, and it does not allow continuous tracking or tracking over long distances. The availability of new sensor technology has made it possible to track individuals remotely. Sensor technology, e.g., GPS, has drastically improved the ability to capture movement [Kays et al., 2015; Nathan and Giuggioli, 2013]. These technological advances have led ecologists and biologists, among other researchers, e.g., urban planners or sports analysts, to collect more and more movement data in recent years. These collections offer a great opportunity to understand movement.

Consequently, domain experts, for instance ecologists, urban planners, sports analysts, or biologists, need methodologies that allow them to generate knowledge from movement data. Quantification of spatio-temporal properties is pertinent if the analyst knows what she is searching for. Qualitative methods, such as exploration, presentation, or detection, help when the analyst knows little of her dataset or does not yet have a specific objective in mind. Both quantitative and qualitative methods are central to the movement analysis and complement each other in it. All too often researchers consider only one or the other; this is true for contributions from algorithms, which typically focus on quantitative methods, as well as for contributions from visualization, which focus mainly on qualitative methods. Combining quantitative and qualitative means is therefore crucial to allow generating a wide range of knowledge from movement data.

In this thesis, we explore how movement analysis can be advanced by enhancing the interplay between algorithms and visualization. In the remainder of this chapter, we will first review how the fields of algorithms and visualization contributed to the analysis of movement before we give a brief summary of analysis tasks for movement data. We then provide an overview of this thesis's contributions by relating them to the discussed analysis tasks for movement data.

1.1 Computational Movement Analysis

To deal with the increasing amount of movement data, researchers have strong need to analyze movement data using automated means. *Computational movement analysis* (CMA) is concerned with developing new computational methods, that detect patterns and structures in movement data [Laube, 2014]. The goal of CMA is to develop insight into the behavior of moving phenomena from raw movement data. Laube [2015] regards closing the semantic gap between low-level tracking data and high-level concepts that an analyst wants to investigate as today's main challenge in CMA.

The way in which movement data is being collected determines and influences substantially its computational analysis. There are different paradigms and manners to collect data. This thesis focuses on a specific type of tracking data, which we will refer to as a *trajectory*. A trajectory is a sequence of locations over time. Each location lies in the plane (or can be projected onto the plane from geographic coordinates) and has a corresponding time stamp indicating when the location was captured. The movement between any two points in a trajectory is unknown, so, in this thesis, we will use only the discrete locations and their time stamps as a representation of trajectories.

CMA is an interdisciplinary research field which involves various methodological research areas to analyze movement: geographic information sciences, computer science, and statistics [Laube, 2014]. Even within computer science for instance, different fields of expertise are needed in CMA to understand movement: data mining, visualization, machine learning, computational geometry, database modeling among others. Overlaps and synergies among those fields in CMA help to elevate tooling and knowledge in CMA.

This dissertation explores the synergies between algorithms and visualization. Given a dataset, algorithms are needed to process the data before visualizing them. Also, visualizations are central to understand how an algorithm works on a dataset. Thus, we are interested in this thesis to investigate how techniques from both fields can be used in concert to strengthen the analysis of trajectory data.

1.2 Algorithms

An algorithm in computer science is a sequence of actions that transforms an input into an output with desired properties [Cormen et al., 2009]. Ideally, an algorithm has provable guarantees on its output and on its performance in terms of running time and storage requirements.

Designing, developing, and analyzing time-efficient and space-efficient algorithms on geometric data is the focus of compu-

tational geometry [de Berg et al., 2008]. Application domains that involve spatial information, e.g., geographic information systems, robotics, cartography, and computer vision, have always played an important role in computational geometry. Naturally, movement data is an important topic in computational geometry due to its spatio-temporal properties. Researchers from computational geometry have thus developed new techniques and methods that have contributed to movement analysis across fields [Demšar et al., 2015].



Figure 1.1: Segmentation of the trajectories of two migratory geese from Alewijnse et al. [2014]. The yellow segments are stopovers, and the blue glyphs indicate the end of a stopover.

However, the resulting graphical representations in these computational approaches often do not go beyond plotting the trajectories as line segments and points on a geographic map (if the approaches are implemented and evaluated experimentally at all). For example, Alewijnse et al. [2014] propose a new method to segment a trajectory into homogeneous pieces of similar movement characteristics, e.g., the same speed. They visualized their experimental results by simply plotting colored trajectories on top of an interactive geographic map representing the segmentation, as shown in Figure 1.1. Presuming a large quantity of trajectories in a dataset, the visual abstractions from plain trajectories are required to declutter the results and to enhance analytical reasoning.

1.3 Visualization

Visualization is a field in computer science which is concerned with the graphical representation of knowledge in form of data, and provides means to manipulate, explore, and alter the representations to help the user in creating new knowledge. Card et al. [1999] formally define visualization as “the use of computer-aided, interactive, visual representation of data to amplify cognition.” Munzner [2014] argues that visualization helps users to carry out their tasks more effectively. Ultimately, the purpose of visualization systems is to gain insights [van Wijk, 2006]. Hence, visualization enables movement analysts to foster knowledge generation from collected data.

Since more and more analysts are collecting an increasing amount of data in various applications, it has become important for visualization systems to integrate users’ knowledge and inference capabilities into analytical and visual data analysis processes to turn this information overload into an opportunity [Keim et al., 2010b]. *Visual analytics* is the research area that supports analytical reasoning with interactive visual interfaces [Cook and Thomas, 2005]. Visual analytics couples (see Figure 1.2) the analyst’s domain knowledge with methodologies and visualizations [Keim et al., 2010a]. Sacha et al. [2016] argue that visual analytics helps users in building trust in their generated knowledge base and in generating knowledge gained from large and often complex data. Because of this cross-disciplinary nature of visual analytics, it has been applied to many domains: sports analytics [Losada et al., 2016; Pileggi et al., 2012], biological networks [Dinkla et al., 2014], fraud detection [Leite et al., 2018; Zhao et al., 2016], eye-tracking data [Kurzhals and Weiskopf, 2013], and many more. Sun et al. [2013] provide a broad overview of the current state-of-the-art in visual analytics and its numerous applications.

However, current visualization systems and visual analytics systems focus mainly on developing new visualizations based on specific domain knowledge. They often do not allow users the flexibility to replace or to experiment with various computational methods. Such visualizations are generally adapted to only one application or one spe-

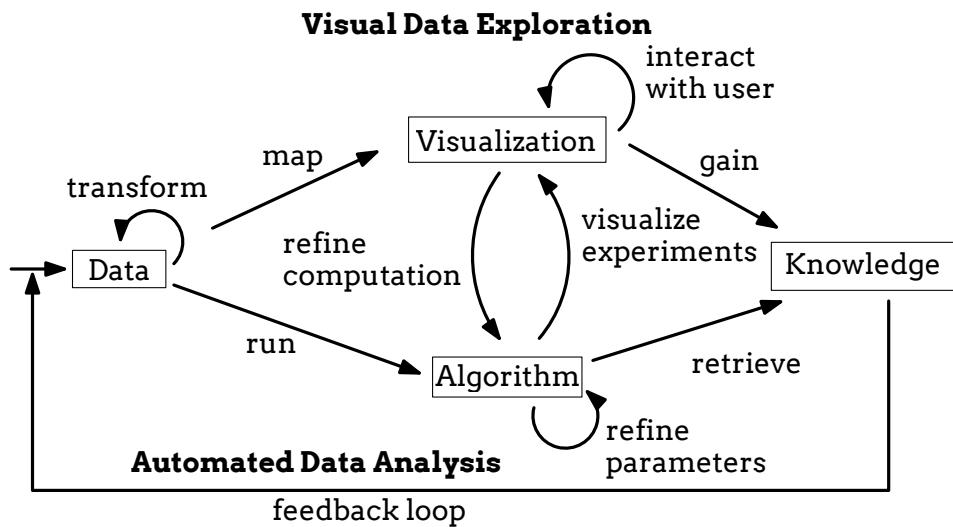


Figure 1.2: The visual analytics process of this thesis, adapted from Keim et al. [2010a]

cific dataset. For example, Andrienko et al. [2013] devised a comprehensive visual analytics tool for analyzing the collective movement of a group of individuals. In their system, they make use of a group's center and presume that a center can be computed at equal time stamps for each trajectory without allowing variability in time and space. Depending on the application, for instance trajectories with high sampling rate, another method to compute a center might be worthwhile. Consequently, a strong need exists to make visual analytics systems more flexible to support a wide range of computational methods.

1.4 Trajectory Analysis

While movement data is used in many, varying applications, there is a common set of frequently occurring fundamental analysis tasks. Each analysis task has distinct computational requirements and visual design considerations. Munzner [2014] argues that task abstraction helps researchers to contemplate on the tasks' similarities and differences, to determine the tasks' different goals, and to guide fur-

ther data abstraction. Munzner's argument thus motivated us to identify and structure analytical tasks for movement data based on the number of trajectories in the data. This structure on the analytical tasks is shown in Figure 1.3.

In Chapter 2, we devise a typology for movement data by grouping analytical tasks with similar characteristics and means. We can group analytical tasks in multiple ways. Our typology provides a way to abstract analytical tasks for trajectory data. In Section 2.2, we discuss alternative ways of abstracting analytical tasks for movement data.

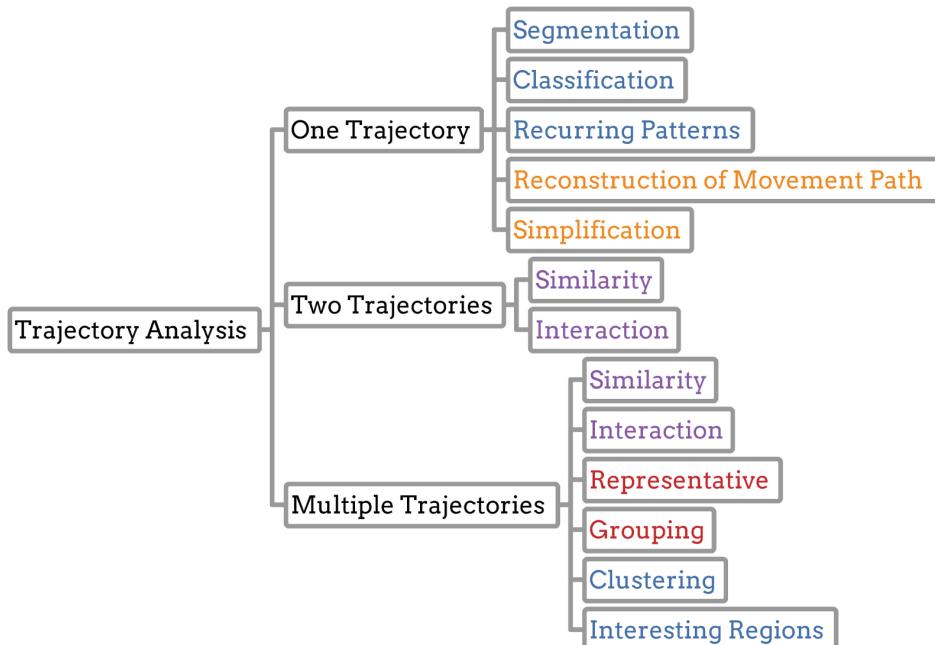


Figure 1.3: Overview of analyses on trajectory data based on the number of input curves. The colors indicate the task abstraction that we use in Chapter 2: alignment methods are in purple, categorizations in blue, transforms in orange, and representations in red.

Given a single trajectory, we can *segment* the trajectory into pieces that are homogeneous with respect to a parameter, such as speed or direction of movement. Assigning labels to pieces of a trajectory that are homogeneous based on a movement state is known as *classification*. Such a movement state can be walking, flying, resting, etc. When we compute *recurring patterns*, we identify subtrajectories

with repeating and similar movements, such as commuting or migration. *Reconstructing a movement path* is concerned with computing an approximation of the actual movement from discrete locations and possibly additional data. Reducing the number of points within a trajectory while approximating the input trajectory up to a given error is known as *simplification*.

The most prominent analysis on a pair of trajectories is to determine their *similarity*. It deals with quantifying distances between two trajectories. Determining *interaction* is concerned with detecting whether trajectories are influencing each other, for example, the individuals of the trajectories are avoiding or attracting each other.

Similarity can also be computed among more than two trajectories, for instance, to identify parts of the trajectories with similar characteristics from multiple individuals. Likewise, we can identify *interaction* events among more than two trajectories. Grouping trajectories or pieces of trajectories based on similarity is called *clustering*. *Classification* assigns labels to parts of trajectories based on shared characteristics. A *representative* is a single trajectory that approximates the movement of all input trajectories. Identifying places that have often been visited or where individuals stayed a certain duration are known as *interesting regions*.

1.5 Overview

Hamming [1987] said: “The purpose of computation is insight, not numbers.” Insight has diverse manifestations in human cognition, is usually the first phase in problem solving, and requires restructuring an input quaintly [Mayer, 1995]. In this thesis, we contribute to computational movement analysis by enhancing the interplay between (theoretical) algorithms and (dataset-driven) visualizations using integrated computational and visual analytics systems, as shown in Figure 1.2. Our approaches foster new opportunities for analyzing trajectory data and allow movement analysts to enrich their knowledge basis and to gain new insights. This thesis makes several contributions

to the analysis of movement, which we will now survey.

In Chapter 2 we provide a background on movement analysis and present a taxonomy of existing approaches from fields such as algorithms, GIS, and visualization. The goals of this survey are to create an awareness of the variety of existing methodologies and to help in advancing interdisciplinary research on movement data in general.

Chapter 3 concerns the computational complexity of analysis tasks on trajectory data. Specifically, we discuss the running times of algorithms for movement data as well as lower bounds for certain analysis tasks. In addition to known bounds, we prove two new lower bounds. First, we show for the simplification problem that an optimal simplification cannot be computed in subquadratic time assuming the *Strong Exponential Time Hypothesis* (SETH) when the points of the input curve lie in $\Omega(\log n)$ -dimensional space, where n is the number of points of the input curve. Secondly, we show that the discrete Fréchet distance between k curves, each consisting of n points, cannot be computed in $O(n^{k-\varepsilon})$ time for any $\varepsilon > 0$ again assuming SETH. Our results permit us to understand which problems can be solved efficiently and for which analysis tasks we need approximation algorithms or heuristics. This chapter is partially based on joint work with Kevin Buchin, Maike Buchin, Wolfgang Mulzer and André Schulz [Buchin et al., 2016].

Shneiderman [1996] posed the visual information-seeking mantra: “Overview first, zoom and filter, then details-on-demand”. In interactive maps, it is essential to draw a trajectory at different scales without unnecessary flickering while zooming in or out. A progressive simplification addresses this problem by enforcing a consistent simplification which only remove vertices when zooming out (and which only adds vertices when zooming in). In Chapter 4 we present the first efficient algorithm for the progressive simplification problem that provides a guarantee on the complexity of the simplification and runs in $O(n^3 m)$ time, where n is the number of points in the input curve and m is the number of scales. For continuous scaling, our algorithm computes a progressive simplification in $O(n^5)$ time. Our second contribution to the simplification problem concerns shortcut graphs. Shortcut graphs are used within many simplification algorithms as data structure. We show how to efficiently compute shortcut graphs at

multiple scales. A shortcut graph encodes at which scale a certain subtrajectory is sufficiently well represented by only its start and end point. We present a method to compute this information for all subtrajectories in $O(n^2 \log n)$ time. This improves upon a running time of $O(n^3)$, which would be needed if we compute this information for each subtrajectory individually. Furthermore, we show how to compactly represent shortcut graphs. The results of this chapter are based on joint work with Kevin Buchin and Wim Reddingius [Buchin et al., 2018].

In Chapter 5 we present a visual analytics tool that computes and visualizes interaction events between two (or three) trajectories in which delayed responses occurred. Our approach aligns the trajectories based on spatial properties and exploits the temporal differences that occur in the alignment. By allowing the analyst to choose various alignment methods, including dynamic time warping and the Fréchet distance, we enable users to also quantify the progression of similarity over time. Our approach consists of multiple coordinated views in which we use the temporal difference that occurs in the alignment to visualize potential delayed responses. Furthermore, we provide a novel approach to compute a global delay between two trajectories in $O(n \log n)$ time by using Fast Fourier Transforms. Chapter 5 is based on joint work with Kevin Buchin, Maike Buchin, Luca Giuggioli, Joachim Gudmundsson, Jed Long, Thomas McKetterick, Trisalyn Nelson, Tim Ophelders, Michel Westenberg and Georgina Wilcox [Konzack et al., 2017, 2015].

Next, we present a visual analytics tool in Chapter 6 that explores migratory trajectory data interactively. In our approach, we compute, aggregate, and visualize stopovers, which are breaks from migration, on top of a geographic map in addition to a density map and a calendar view. We applied our visual analytics tool to a dataset of 75 lesser black-backed gulls, and validated our approach through an expert user interview. Our evaluation suggests that our tool enables ecologists to visually explore migratory patterns in trajectory data. This chapter is based on joint work with Kevin Buchin, Pieter Gijsbers, Emiel van Loon, Ferry Timmers and Michel Westenberg [Konzack et al., 2018].

2

Background

2.1 Introduction

Researchers from many domains and applications collect trajectory data to understand the cues and drivers behind an individual's movements. Methodologies for analyzing trajectory data benefit from the numerous concepts and theories from multiple disciplines. Thus, it is important to get an overview of the existing approaches. Therefore, we will survey methodologies from various disciplines in this chapter. We abstract low-level analytical tasks and survey their contributions, challenges and opportunities, and high-level goals as expressed as means and characteristics.

Such an overview helps in designing and developing new methods to analyze trajectory data as well as in identifying similarities and differences between the analytical tasks. By connecting the high-level goals that analysts often have in mind with the low-level analytical tasks for trajectory data, we aim to provide a comprehensive and holistic survey on the analysis of movement data.

The contributions in this chapter are:

- a survey of methods used to analyze and visualize trajectory data,
- a typology in which we arrange all approaches based on a task's means and characteristics, and

- a discussion of general challenges in movement analysis.

2.2 Related Work

Taxonomies help in guiding researchers to abstract, to synthesize, and to contemplate about aspects of their research problems and to develop new methodologies. Here, we give an overview of past surveys and taxonomies for visualization, for computational approaches, and for movement data in general.

Within the visualization community, Munzner [2014] discusses the link between why and how visualization systems work and what they visualize. Brehmer and Munzner [2013] proposed a typology that bridges the gap between low-level tasks and high-level tasks. Their domain-agnostic typology is based on both why and how a visualization task is performed.

Similarly, Schulz et al. [2013] explored the design space for visualization tasks. They characterized tasks in a taxonomy of five dimensions: the task's goal (why), the task's means (how), data characteristics (what), the target and cardinality of data entities (where), the order of tasks (when), and the (type of) users (who). As an example, Schulz et al. [2013] applied their taxonomy to tasks for climate impact research. We use two of the dimensions in this chapter, the task's means (how) and the task's characteristics, to categorize methodologies for movement data.

Lam et al. [2018] extended the work by Schulz et al. [2013] on visualization tasks by organizing design study papers into a two-axes taxonomy on the specificity, which spans from exploring via describing and explaining to confirming as task's goals, and on the number of data populations, which can be either a single entity or multiple entities.

Andrienko and Andrienko [2013] surveyed the state of the art of visual analytics approaches for movement data. They divided visual analytics systems into four categories: looking at trajectories, looking inside trajectories, showing a bird's eye view of movement data, and

investigating movement within its context.

Andrienko et al. [2011] modeled the links between the dimensions space, time, and the moving entity of the trajectory as a taxonomy for analytical methods on movement data. They defined tasks by combining properties from these three dimensions. Furthermore, they differentiated whether analyses operated on these properties directly (elementary analysis) or whether they worked on (sub)sets of these properties (synoptic analysis).

In geography, the *Space Time Cube* (STC) is a representation for movement data in which time is modeled explicitly as a third dimension in addition to the locations in the Euclidean plane [Hägerstrand, 1970]. Bach et al. [2014] gave an overview of all possible operations on an STC. By defining these operations, they were able to describe visualization systems for time series as sequences of operations on an STC. They addressed how existing approaches can be operationalized on STCs, including interactive 2D visualizations that use animation.

Long and Nelson [2013b] surveyed quantitative methods for analyzing movement data, in which they classified existing approaches into seven categories: time geography, path descriptors, similarity indices, pattern and cluster methods, individual-group dynamics, spatial field methods, and spatial range methods.

2.3 Scope and Focus

In this section, we describe how we devised our typology from studying existing approaches on movement data. First, we studied existing survey papers on this topic from the preceding section in addition to handbooks for computational geometry [Goodman et al., 2017; Sack and Urrutia, 1999] and a survey dedicated to the contributions of the MOVE project [Demšar et al., 2015]. From these sources, we collected valuable information from an array of references from various disciplines, such as computational complexity, data mining, computational geometry, GIS, visualization, movement ecology, and visual

analytics. Then, we compiled an exhaustive list of analytical tasks in a bottom-up fashion from these papers, as shown in Figure 1.3. We merged common topics into a single theme, e.g., reconstructing a movement path contains approaches for interpolation of trajectories as well as map matching.

The starting points for our typology are the works by Munzner [2014] and Schulz et al. [2013]. From the five dimensions distinguished by Schulz et al. [2013], we selected *means* because they model actions and are expressed as verbs, and we chose *characteristics* because they cover aspects of the data that interest analysts. We dropped and merged some values for means and characteristics because they were not generally applicable to movement data, but rather only to visualization systems. By analyzing the relations between the low-level tasks in terms of task A “is specialization of” B, we introduced a new abstraction layer. For instance, *interesting regions* is a specific form of a *clustering* or *segmentation*, so we aggregated these types of approaches into *categorization*.

2.4 Typology for Analyzing Movement Data

In Chapter 1, we covered why researchers collect movement data and what motivates them to analyze trajectory data. We also introduced (low-level) analytical tasks that researchers can use to compute or visualize trajectory data.

The objective of the typology in this chapter is to connect the low-level analytical tasks to the high-level goals that analysts have, e.g., to discover, explore, or identify patterns in a trajectory dataset. We use two high-level goals to categorize and abstract the low-level analytical tasks to types of analyses with similar characteristics and means:

- How and by which means can we conduct analytical tasks?
- What do these tasks aim to reveal in the data?

These two questions are inspired by the taxonomies by Schulz et al. [2013] and Munzner [2014]; we adapted them for the spatio-temporal

properties of trajectory data. The first question is concerned with the *means* of an analytical task and deals with the operations to reach a goal. A mean is sometimes referred to as action or task, and it gives insights into how an analytical task is carried out. The second question deals with the *characteristics* that describe particular facets of the trajectory data which the analytical task aims to reveal. Munzner [2014] refers to the characteristics as targets. Schulz et al. [2013] distinguished between low-level characteristics, which a user can perceive and detect easily, and high-level characteristics, which require more sophisticated techniques to mine.

We adapted and simplified the categories for the means and characteristics to model aspects specific to movement data. In addition to means and characteristics, we aggregated analytical tasks into types of tasks with similar means and characteristics. Such a typology allows us to study the structural properties between the analytical tasks and to identify additional, more abstract topics for movement analysis. In Table 2.1, we show our typology. An X denotes that the analyses use these means and characteristics. An (X) denotes an implicit dependency or usage of the given means or characteristics. For example, all types of analytical tasks in our typology use similarity measures.

We now describe each part of our typology in more detail. First, we give an overview of the means to analyze trajectory data. Next, we review the characteristics of analytical tasks for movement data. Then, we discuss the different types of analyses in depth.

Adding metadata, such as a new attribute, by a manual user action is *annotating*. Contrarily, *deriving* new attributes from existing ones is carried out computationally.

An analysis can have various scopes, and we identified the following five query operations. Filtering or (sub)sampling of data points is known as *extracting*. The *identification* of regions or individuals is concerned with computationally detecting and finding associations in trajectory data. In visualization systems, it is common to give an overview of the data. Such a bird's eye view gives a *summary* of the dataset. Another important query operation is to *compare* the move-

Table 2.1: Our taxonomy for the analysis of movement.

	<i>Alignment</i>	<i>Transform</i>	<i>Categorization</i>	<i>Representation</i>
Means				
Annotate			(X)	(X)
Derive			X	
Extract		X		
Identify	X		X	
Summarize				X
Compare		X		X
Relation-seeking	X		X	X
Characteristics				
Trends	X		X	
Outliers		X		
Features	X	X	X	X
Distribution				X
Dependencies				X
Correlation	X		X	
Similarity	X	(X)	(X)	(X)
Network				X

ments of two (or more) individuals by juxtaposing the results of both trajectories. Contrarily, *relation-seeking* deals with analyzing possible links between two (or more) individuals.

Munzner [2014] identified three high-level characteristics that each analytical task can have and which also apply to trajectory data. High-level patterns in a dataset can be described as *trends*. On the contrary, *outliers* are data points which do not fit well (or at all) into structures and which represent anomalies with respect to other data points. The third high-level characteristic is *features* which constitute particular structures of interest.

Beyond these high-level characteristics, we identified other characteristics for trajectory datasets. A *distribution* encompasses counting occurrences of spatio-temporal properties with respect to their spatial and temporal extent, such as regions or frequencies for different years or months, to spot typical values as well as anomalies. With *dependency*, we refer to occurrences when movements in one trajectory influence the movement in another trajectory. The *correlation* between (the values of) two trajectories quantifies the (statistical) relationship between the trajectories' movements. *Similarity* characterizes how a trajectory resembles another trajectory. A *network* expresses individuals' movement in terms of states and the relations between these states, enabling analysts to reason on higher-level abstractions from the movement data.

We divided analytical tasks on trajectory data into four types in our typology: alignments, transforms, categorizations, and representations. Within *alignments*, we survey approaches that quantify the inter-dependency between two (or more) trajectories. *Transforms* modify the number of points within a trajectory either by reducing or by adding locations to the input trajectory. *Categorization* approaches are concerned with identifying meaningful parts of trajectories, regions of interest, or moving entities with specific spatio-temporal properties. *Representations* summarize movement by abstracting from plain trajectories.

2.5 Overview of Trajectory Analysis Tasks

High-level research questions and goals are related to interpreting the cues and mechanisms behind movement. New methodologies achieve these goals to some extent by focusing on low-level research questions related to a specific phenomenon and by limiting their attention to only certain aspects of high-level goals. For example, take the work by Alewijnse et al. [2014]. Their visualization prototype segments trajectories and allows an interactive selection of parameters. Alewijnse et al. [2014] mention only their high-level goals within a case study (to distinguish between movement states, such as migration and stopovers). They do not describe other operations that an analyst might want to investigate, such as *seeking relations* or *identifying trends*. To fully understand why and how a method works, it is essential to connect both. Our typology links low-level analytical tasks to types of analyses.

We now survey each type using the following structure:

Background addresses the setting and the underlying motivation for applying this analysis type to trajectory data;

Means and Characteristics concern the links between the aims and the actions needed to for this analysis type;

State of the Art gives an overview of computational approaches as well as visualizations for this analysis type (Here we survey the approaches as such. For a discussion of the computational complexity of the approaches we defer to Chapter 3);

Challenges and Opportunities deals with limits and constraints for this analysis type as well as possibilities for future work.

2.5.1 Alignment

Background. The movements of individuals often influence each other, for instance, when pursuing a particular goal, e.g., foraging, or

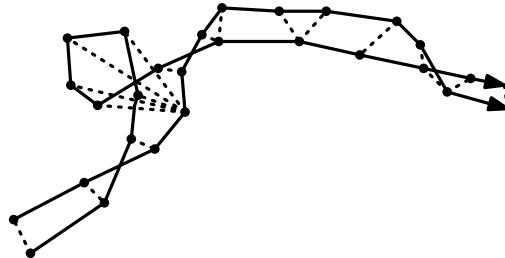


Figure 2.1: Aligning two trajectories in a monotonous matching (dashed).

when traveling together as a flock or in a particular formation, like a single file. Therefore, the movements of one individual correlate to some extent with the movement of others. Given two (or more) trajectories, a basic problem is quantifying how similar or dissimilar these trajectories are to each other. Such a quantification gives insights into how the movements of one individual are related to or depend on the movements of another individual. Quantifying the interdependency between moving entities allows analysts to express and trace the influence between trajectories over time. An *alignment* captures the interdependency as a mapping from each point of one trajectory to a point of the other trajectory that is similar.

Means and Characteristics. Alignment methods quantify *similarity* and correlate locations from two (or more) trajectories. To align similar features in different trajectories, it is necessary to *identify features*. Furthermore, the *dependencies* between points of one trajectory and points of another trajectory are also of interest in the computation of alignments.

State of the Art. A variety of alignment methods from many applications exist. Alignment methods commonly interpret a trajectory as a curve or a sequence parameterized/indexed by time. These alignments continuously map one trajectory onto another trajectory without reversing time during the mapping. Naturally, these alignments starts at the first points of both trajectories and ends at their endpoints.

An alignment method originally used for curve matching is the Fréchet distance which minimizes the maximum distance between two trajectories. The Fréchet distance has been applied to computa-

tional movement analysis [Buchin et al., 2008, 2010]. Lower bounds are known (see Section 3.4.2) as well as algorithms to compute the Fréchet distance exactly (see the works by Alt and Godau [1995], Buchin et al. [2012], or Rote [2014]) or approximately [Driemel et al., 2012; Dumitrescu and Rote, 2004].

Dynamic time warping is another alignment method [Berndt and Clifford, 1994]. It minimizes the sum of distances between two trajectories within an alignment. Dynamic time warping is popular within data mining and database systems. To bypass the near-quadratic lower bound [Gold and Sharir, 2016], Salvador and Chan [2007] and Al-Naymat et al. [2009] have developed heuristics.

The edit distance [Wagner and Fischer, 1974] and the longest common subsequence [Maier, 1978] are also popular alignment methods, which have been applied to trajectory data [Chen et al., 2005; Vlachos et al., 2002]. All these alignment methods share (conditional) near-quadratic lower bounds, which makes it challenging to compute optimal alignments for large datasets.

Wang et al. [2013] reviewed various alignment methods for time series experimentally. Among these techniques were dynamic time warping, the longest common subsequence, and two versions of the edit distance, but not the Fréchet distance.

Many movement patterns are closely linked to alignments. Andersson et al. [2008] developed an algorithm to detect the movement pattern of leadership. Leadership is when one entity is followed by sufficiently many other entities, but is not following another entity itself. Here an entity is considered to follow another entity if the other trajectory is in the *front region*, which is a circular sector facing the direction of movement. Similar to leadership, a single file captures a follow-behind relationship. Buchin et al. [2008, 2010] defined a single file as a group of moving entities in which one leads the group, and all others are following each other.

Similarly, the dynamic interaction measure developed by Long and Nelson [2013a] detects follow-behind relationships between trajectories by making utilization of spatial displacements. They applied the dynamic interaction measure to movement datasets from grizzly bears

and ultimate frisbee players on global, local, and episodic temporal scales.

In Chapter 5, we use alignment methods to quantify action-reaction patterns for three datasets: two different pigeon datasets and ultimate frisbee data. We use the computed alignments to visualize the inter-dependency between the trajectories.

Challenges and Opportunities. Defining and computing an alignment for multiple trajectories is challenging since the running time for computing such alignments grows exponentially in the number of trajectories. Thus, efficient approximation algorithms are needed here. Another challenge is visualizing alignments of three or more trajectories on two-dimensional screens.

To investigate when an individual does not interact with another individual, it would be useful to define a reference model for no interaction. Such a no-interaction model might vary between applications. Exploring how computed interaction events are related to actual observed interaction might be of help to define null models for interaction.

2.5.2 Transform

Background. In applications like sports analysis, locations are tracked with a high sampling rate (multiple locations per second). Such high resolutions allow researchers to describe movement almost continuously. However, this results in large datasets. That makes it difficult to handle this data computationally, in particular when methods with high computational complexity need to be used (see Chapter 3). But applications also often face the opposite problem, that is, movement datasets which have been captured with low precision or a sparse sampling rate. Movement analysts are therefore in need of *transforms*, techniques to reduce the number of locations within a trajectory while preserving the shape of the trajectory to some extent, on the one hand, and on the other methods to enrich

trajectories by additional locations to improve the quality of the analysis on the other hand.

Means and Characteristics. Comparing features of a trajectory is needed in transforms. Transforms also extract features to enhance the results of the analysis. Similarly, detecting *outliers* is central to transforming trajectories to differentiate regular locations from anomalies.

State of the Art. Applications need to find a way to reduce the complexity of a raw trajectory, such that the trajectory can be further processed efficiently. Curve simplification then addresses the problem of reducing the complexity of a trajectory by minimizing the number of vertices of the curve while preserving the original shape of the curve up to a specific error ε . We discuss curve simplification in Section 3.3.1 in more detail.

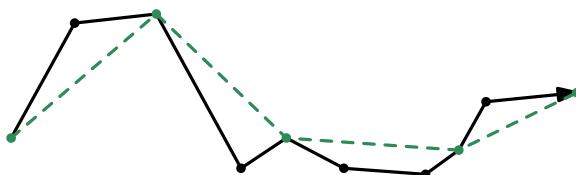


Figure 2.2: Computing a simplification (in green) from a trajectory.

For some applications, such as cartography, it is important to compute a series of simplifications that are consistent across different scales [Cao et al., 2006; Visvalingam and Whyatt, 1993]. Consistency here means that whenever a trajectory/curve is simplified further (moving from a finer to a coarser scale), the simplification removes only vertices. We refer to such a simplification as *progressive simplification*. In Chapter 4, we give the first algorithm to compute minimum-complexity progressive simplifications.

Since trajectories are collected as a sequence of discrete locations, the movement between two locations of a trajectory remains unknown to the observer if the locations are not sampled frequently enough. The reconstruction of the original movement path is, therefore, an important task because it enables analysts to understand the actual movement by exploiting the exhaustive set of all possible move-

ment patterns on the movement path. For instance, interpolation of movement is central to handle intermediate positions between two locations in sparsely sampled movement data. Linear interpolation usually works sufficiently well for densely sampled trajectories. By employing the *Brownian bridge movement model* (BBMM), Buchin et al. [2012] were able to compute movement patterns, including encounter, avoidance, attraction, regular visits, and following, from movement data with low sampling rate and thus high uncertainty. The BBMM assumes random movement between measured locations. Buchin et al. [2015] showed how to integrate environmental context into a BBMM-based analysis. They demonstrated on two ecological datasets that the derivation of movement parameters and the movement parameters' spatial distribution via BBMM is a powerful technique for computational movement analysis.

Reassigning locations of a trajectory to positions on a road network of a digital map is known as *map matching*. This transform remedies the problem of inaccuracies in the tracking of individuals moving on a known network. Greenfeld [2002] reviewed several approaches for map matching. A map matching algorithm can either locally adapt locations iteratively or remap locations of a trajectory globally. Brakatsoulas et al. [2005] devised an approach that allows both paradigms and employs the Fréchet distance to match trajectories onto a street network. They applied their approach to 45 trajectories from vehicles in Athens, Greece. Similarly, Lou et al. [2009] developed a global map matching approach that uses geometric and topological structures of the road network, which furthermore allows to define spatio-temporal constraints. They compared the performance of their algorithm with other map matching algorithms on synthetic data and the GeoLife dataset [Zheng et al., 2009]. Newson and Krumm [2009] employed Hidden Markov Models in their map matching algorithm for sparse trajectories.

Challenges and Opportunities. Because global map matching algorithms generate curves that yield a smaller Fréchet distance to the original trajectory than incremental map matching algorithms, and because incremental algorithms run faster than global ones, it would be worthwhile to investigate how this trade-off could be bal-

anced by, for instance, integrating multiple sensors into a map matching algorithm.

Computing optimal simplifications in subquadratic (or ideally, near-linear) time remains an open problem, except for in some specific settings [Agarwal and Varadarajan, 2000].

2.5.3 Categorization

Background. Understanding the cues and drivers behind movement motivates researchers to collect and to interpret trajectory data. A partitioning of (sub)trajectories allows analysts to discern which regions or (pieces of) trajectories are of interest. Assigning categories to these partitions can be either explicit or implicit. We refer in both cases to this problem as *categorization*.

Means and Characteristics. Categorization *identifies features* of trajectories that break them into categories. A methodology for categorization *derives trends* that are prevalent in the data. Furthermore, computing such partitions helps in *seeking relations* between categories. It is likely that a *correlation* of spatio-temporal properties exists within a category.

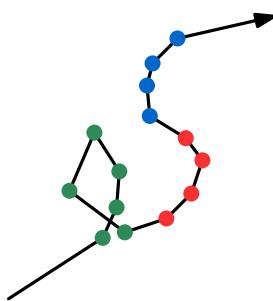


Figure 2.3: Segmenting a trajectory into three segments: the circular movement (in green), a left turn (in red), and a right turn (in blue).

State of the Art (Segmentation). Dividing a trajectory into pieces with similar movement parameters, such as speed, heading, or turning angle, is known as *segmentation*. Segmentation allows us to detect

subtrajectories, that describe behavioral states of a moving entity, as shown in Figure 2.3. There is a trade-off between the number of such segments and how similar the (sub)trajectory is within a segment. Given a spatio-temporal criterion on the similarity within a segment, it is natural to formulate this problem as an optimization problem. To avoid overfitting, we aim to minimize the number of segments.

In the past, either heuristics or optimizations of a global criterion have been considered rather than the minimization of the number of segments [Aronov et al., 2016]. Aronov et al. [2016] devised a framework for optimal segmentation, given a so-called start-stop diagram, which is a representation of valid and invalid segments on a given trajectory. The start-stop diagram naturally leads to a quadratic-time algorithm if a trajectory can be segmented only at data points. Alewijnse et al. [2014] tackled the quadratic barrier on segmentation for a wide range of criteria. Alewijnse et al. [2014] developed a visualization to allow an interactive parameter selection for the segmentation criteria, and they applied their prototype to a dataset of migrating geese.

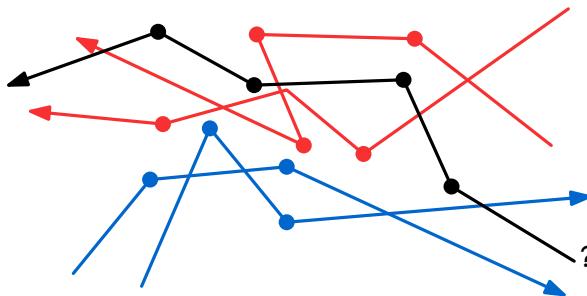


Figure 2.4: Classification of trajectories into two classes: red and blue. Given an unclassified trajectory (in black), the classification problem asks to compute a class for this trajectory.

State of the Art (Classification). Classification is concerned with assigning (pieces of) trajectories to classes. The goal of a classifier is to determine the discriminator between the classes. A classification can have binary classes, e.g., walking or non-walking, or have multiple categories, such as walking, resting, foraging, etc. The assignment of classes can have different spatial extents. A class can be assigned to either entire trajectories or pieces of them. Classification was originally defined in statistics and later applied in machine learn-

ing [Bishop, 2007] and data mining [Zheng, 2015]. The original definition asks to learn from labels of a training dataset to classify a test dataset which did not have labels, as shown in Figure 2.4.

Zheng et al. [2008a] and Zheng et al. [2008b] developed approaches to learn modes of transportation and people's motion modes and to experimentally infer the transition between different modes from trajectory data by using techniques from machine learning. TrajClass [Lee et al., 2008] classifies (sub)trajectories based on the density of their locations and allows tuning for specific regions. Alewijnse et al. [2017] classify subtrajectories based on the parameter of a movement model.

State of the Art (Clustering). Similar to classification, *clustering* algorithms aggregate pieces of trajectories with similar characteristics, see Figure 2.5. Computing clusters works either top-down or bottom-

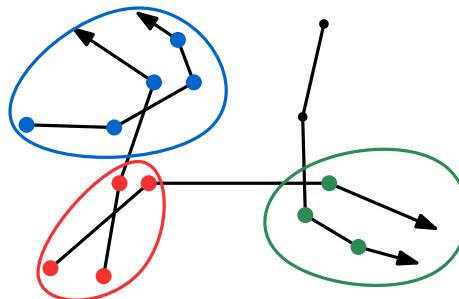


Figure 2.5: Clustering of four trajectories. Each cluster (in red, green, and blue) captures subtrajectories with similar movements.

up. One type of top-down clustering algorithms are density-based approaches. The DBSCAN [Ester et al., 1996] and the OPTICS [Ankerst et al., 1999] algorithms are the most prominent algorithms for point-based clustering. Andrienko et al. [2018, 2009, 2007] implemented the OPTICS algorithm to visualize clusters in trajectory data experimentally. Lu et al. [2015a] employed a modified version of the DBSCAN algorithm in their visual analytics system to detect clusters of specific regions in urban areas.

Lee et al. [2008, 2007] devised approaches that combine region-based classification with density-based clustering to improve the accuracy of the resulting clusters.

Gaffney and Smyth [1999] developed a clustering algorithm for trajectories that uses a probabilistic mixture regression model by applying an EM algorithm on the trajectories. They applied their clustering algorithm to both simulated data and video data.

The subtrajectory clustering by Buchin et al. [2011] employed the Fréchet distance to detect clusters of predetermined subtrajectory lengths and with a specified number of moving entities within each cluster. Gudmundsson and Valladares [2015] explored how to use GPUs for the subtrajectory clustering algorithm.

In Chapter 6, we present a hierarchical clustering algorithm that is integrated into a visual analytics system. We applied our approach to a dataset of migratory gulls.

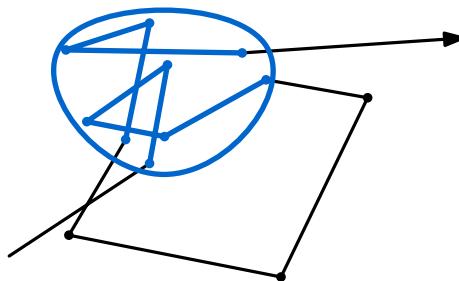


Figure 2.6: A recurring pattern (in blue) within a trajectory. Such patterns can be either shape-based or distance-based.

State of the Art (Recurring Patterns). Moving entities often follow the same or comparable similar routes; they also often pursue these routes repeatedly. The consequent periodicity is observable along different temporal scales: daily (commuting), annually (migration), or seasonally. These *recurring patterns* are often modeled as sequences, see Figure 2.6. Detecting recurring patterns in a trajectory helps analysts to find subtrajectories with similar movement characteristics. The approach by de Berg and Mehrabi [2016] reports all subtrajectories that are similar to a given line segment with respect to dilation and direction deviation and has been applied to a dataset of soccer players.

Cao et al. [2007] used a density-based clustering with DBSCAN to compute recurring patterns. In their approach, user-specified periods (days, weeks, or months) define the periodicity of the pattern. Their approach considers time shifts and distortions in the trajectories, and they evaluated their approach experimentally on synthetic data.

State of the Art (Interesting Regions). Some places that individuals visited are more crucial for pursuing a goal than others. Movement in relation to a place or a region provides an indication about the relevance of a place, for example: how many individuals visited a place or how much time individuals spent in such a place. *Interesting regions* can be viewed and modeled as a specific form of a clustering.

Regions that have been visited by many entities are so-called popular places. Benkert et al. [2010] studied the problem of finding popular places. They devised an $O(\hat{n} \log \hat{n})$ -time algorithm for the discrete model, given a set of trajectories with a total number of points of \hat{n} (and at most nk , where k is the number of trajectories and n the number of points within a trajectory). Gudmundsson et al. [2013] explored various optimization problems of square-shaped regions, so-called hotspots, and devised algorithms for them.

Challenges and Opportunities. A common trait that all approaches of this type share is that they require a valid discriminator which appropriately describes the desired categorization. This presupposes prior knowledge of the drivers and mechanisms that underlie movement. Visualizations could help to guide the exploration of viable parameterizations experimentally.

Because clusters, interesting regions, and recurring patterns share that they describe regions in the plane, it could be worthwhile to investigate how to model and visualize the links and dependencies among these categories, e.g., by incorporating temporal properties.

2.5.4 Representation

Background. When a movement dataset has many trajectories, it is more prevalent to summarize the trajectories by abstracting

them conceptually from their locations. A *representation* remedies this by superimposing a structure upon all input trajectories. Such an abstraction helps to reduce visual clutter in the data and the cognitive load of processing it.

Means and Characteristics. Representations *summarize features*, and they *extract dependencies* implicitly or explicitly. Comparing the *distribution* of the individuals' locations is also required in representations. Representations help analysts in *seeking relations* between trajectories as well as identifying movement states and relationships between movement states (*networks*).

State of the Art. In statistics, an average describes a typical value within a set of values. A *representative* for such a set is calculated to express the typical characteristics. It is calculated from a population of values. Similar abstractions are needed for movement data. Given a collection of trajectories, a representative approximates the move-

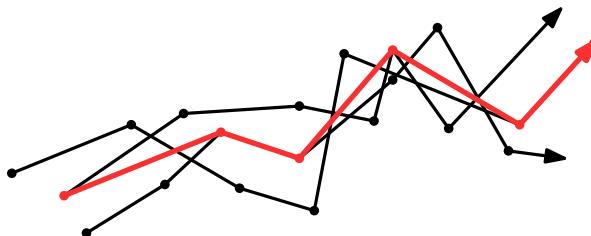


Figure 2.7: Computing a representative from a collection of trajectories.

ments of all trajectories to a certain degree by being "in the middle" of all trajectories, see Figure 2.7. A representative needs to capture the progression of individuals' movements either by using only locations from the trajectories or by adding derived points from the trajectories that express centrality.

Buchin et al. [2013a] defined a *median trajectory* intuitively by using pieces of input trajectories and by staying in the middle of them. A *mean trajectory* averages locations, one from each trajectory, like a center of gravity. A median trajectory stays central with respect to the number of given trajectories; whereas, a mean trajectory does not. Buchin et al. [2013a] devised two algorithms to compute a median trajectory: a simple median, based on the arrangement of lines and

which is computable in $O((nk)^2)$ time, where k is the number of trajectories and n the number of points in a trajectory, and a homotopic median, based on geometric and topological concepts and which can be constructed in $O((nk)^{2+\varepsilon})$ time for $\varepsilon > 0$. The running times for these two medians are optimal because a median curve is composed of $O(nk)$ line segments, which can yield output sizes of up to $O((nm)^2)$.

However, the algorithms by Buchin et al. [2013a] are not sensitive to the time stamps of the data points of the trajectories. By computing a *central trajectory*, van Kreveld et al. [2015] tackled the issue of finding a time-sensitive representative. Their central trajectory consists of points from the input trajectories and switches from one trajectory to another when the smallest enclosing disk of the data points at time t needs improvements. In \mathbb{R}^1 , a central trajectory of complexity $\Theta(nk^2)$ is computable in $O(nk^2 \log k)$ time. For trajectories in d -dimensional space with $d > 1$, van Kreveld et al. [2015] devised another algorithm to compute a central trajectory with a complexity of at most $O(nk^{5/2})$ in $O(nk^3)$ time.

Andrienko et al. [2013] designed a linear-time algorithm to compute a centroid by iterating through all trajectories simultaneously and finding a central location among the trajectories at corresponding time stamps. Their technique presumes that all trajectories have the same length.

In a large group of moving entities, we are interested in questions like who traveled together in a subgroup and for how long. Flocks, swarms, and moving clusters [Dodge et al., 2008] address these questions partially, but little work has been done on considering merging and splitting between different groups. Buchin et al. [2013b] addressed this gap by introducing the *trajectory grouping structure* and presenting an algorithm that computes it efficiently. Kostitsyna et al. [2015] extended this work by incorporating contextual information into the grouping structure. They used the geodesic distance to measure the distance between entities where obstacles, e.g., buildings, lakes, or walls, occur. The interactive analysis of grouping structures for varying parameters was investigated by van Goethem et al. [2016].

Challenges and Opportunities. Context, such as geographical regions or weather, provides important information, which can influence, how we formulate a suitable representative. Integrating contextual information into the computation of representations and visualizing representations in relation to the context is of high importance.

Representations might be of help when visualizing alignments among multiple trajectories because a suitable representation can serve as the basis of a visual abstraction of many trajectories.

2.6 Discussion

We reflect on our typology, analysis tasks, and methods surveyed by relating gaps and challenges that we synthesized from previous research.

2.6.1 Workflow

Often analytical tasks work in concert with each other to (pre)process data or to improve computational results. We decided not to arrange them into workflows since it is tedious to enumerate all meaningful workflows and some analytical methods, e.g., segmentation can either be a pre-computational step or the desired end result. Zheng [2015] surveyed methodologies for trajectory data within the context of data mining by identifying a generic workflow from preprocessing and indexing of trajectory data to mining of patterns.

2.6.2 Scale and Uncertainty

Trajectory data consists of a sequence of locations over time. Uncertainty of the path between those locations is inevitable, even when the temporal sampling rate during recording is increased. Therefore,

granularity and uncertainty can significantly influence an analysis's interpretation [Shamoun-Baranes et al., 2011a]. Dodge et al. [2016] argue that it is essential to analyze movement at multiple scales because different movement patterns are expressed and characterized at different spatial and temporal scales. Laube et al. [2007] identified four temporal scales in the analysis of movement: globally, computed on the entire trajectory; locally, at a specific time stamp; episodic, computed on a subtrajectory where a spatio-temporal movement parameter of the trajectory – for instance the heading – is homogeneous; and intervallic, on a fixed window of time stamps. Wood et al. [2010] defined a three-level analysis for collective motion that encompasses both spatial and temporal granularity: how individual members move within a collective, how the collective moves as a single entity, and how the collective's footprint evolves (individuals entering and leaving the collective).

2.6.3 Context

Not only intrinsic factors explain the movements of an individual and motivate entities to move; external factors also influence how an individual moves. Analysts, e.g., ecologists, use this contextual information implicitly or explicitly in their research [Shamoun-Baranes et al., 2011a]. Storing information about how moving entities interact with their environment is central to improve computational results, e.g., in segmentation or statistical analysis. Dodge et al. [2016] claim that only by incorporating and considering all factors in an analysis a deeper understanding of patterns in movement data can be gained. Integrating data from multiple sensors, like accelerator or weather data, can help in filling the gap in capturing a moving entity's behavior more thoroughly.

2.6.4 Space and Time

A trajectory's data points show high auto-correlation for space and time [Demšar et al., 2015; Shamoun-Baranes et al., 2011a]. The higher

the sampling rate a trajectory has, the higher the correlation is between space and time. In our survey, we did not distinguish between approaches that interpret trajectories as discrete sequences of timestamped locations and approaches that (by suitable interpolation) interpret trajectories as continuous movements over time. However, the computational complexity might differ for approaches depending on how the approach interprets trajectories (see Chapter 3).

2.6.5 Interdisciplinarity

Because movement analysis has many applications, collaborations between researchers working on new methodologies and researchers from application domains are crucial to foster innovative, exciting, and unusual new concepts for the analysis of movement [Demšar et al., 2015; Dodge et al., 2016]. Disseminating these interdisciplinary efforts can be challenging because finding suitable venues for a specific target audience across disciplines is difficult.

2.6.6 Specificity of Data Analysis

Lam et al. [2018] proposed a taxonomy for visualization systems that includes the specificity of an analysis. They delineated a spectrum of four categories for specificity that ranges from *exploring* and *describing* to *explaining* and *confirming*. Demšar et al. [2015] argue that methods from information sciences, e.g., visualization, which are apt for data exploration, could contribute to confirmatory (hypothesis-driven) approaches in ecology that enhance the (presentation of the computed) results. In addition, Dodge et al. [2016] point out that simulation and predictive models for movement will become more important in the future.

2.7 Conclusions

We surveyed approaches that analyze movement data in this chapter. These methods come from various backgrounds: geographic information science, computational geometry, data mining, and visualization. Our contribution is a typology for the analysis of movement. We aggregated all approaches into four types: *alignments* that arrange trajectories to quantify the inter-dependency between them; *transforms* that change the sequence of points in a trajectory to boost the results for a subsequent analysis; *representations* that find structures from multiple trajectories; *categorizations* that identify (pieces of) trajectories with similar properties.

Furthermore, we identified and connected high-level research questions to our typology by taking the means and characteristics of analysis tasks into account. For each type of analysis, we discussed limits and future work. Then, we synthesized an overview of challenges on movement data from previous papers.

In the subsequent chapters, we present approaches for some of the aforementioned analytical tasks. We elaborate on the computational complexity in next chapter.

3

Computational Complexity of Problems on Trajectories

3

3.1 Introduction

Within the last decade, researchers from different fields worked on developing new techniques and algorithms for movement data [Demšar et al., 2015]. Each application of movement data has its specific requirements that concern computational aspects as well as constraints related to how to present and to communicate scientific results.

In this chapter, we study the computational complexity of analytical methods on movement data for a single trajectory, two trajectories, and multiple trajectories. We discuss state-of-the-art algorithms for computational problems on trajectory data.

The importance of the efficiency of algorithms is immanent if we want to solve problems for large trajectory datasets. To illustrate the importance of lowe bounds, consider the problem of computing an alignment between two trajectories.

We know little about the factors that determines the hardness of a problem on curves. In the past years, previous research explored and proved quadratic bounds for problems concerning two curves. For

large-scale datasets, researchers now need to either restrict their input by limiting the computations to small instances or by imposing additional constraints on the input trajectories, or they need to devise near-linear time heuristics or approximation algorithms. For problems concerning a large number of trajectories, however, less is known about how to solve problems in polynomial time, and which problems admit such a solution. Take the work by Buchin et al. [2011] to cluster subtrajectories as an example of a problem on multiple subtrajectories for which more is known. First, they formulated this problem as an optimization problem on multiple subtrajectories. Then, they showed that it is infeasible to solve it optimally in polynomial time. This insight led them to resort an existing approximation algorithm [Dumitrescu and Rote, 2004] to compute clusters efficiently. Therefore, it is essential to use concepts and results from problems on two trajectories in the context of computational complexity for problems with multiple trajectories.

Analyzing the computational complexity of problems on trajectories, discussed in this chapter, will serve as a basis for the design of methodologies for trajectory data, in the subsequent chapters, which are efficient with respect to time and space.

3.2 Preliminaries

In this section, we review common definitions, conjectures, and problems that have been used in the past to prove lower bounds. A lower bound for a computational problem provides insights into how efficiently the problem can be solved by any algorithm. If a lower bound exists for computational problem, then we know that we cannot develop an algorithm for this problem that, in general, solves it faster than the given lower bound.

Lower bounds need to make assumptions about the computations that are possible and their efficiency. For instance, lower bounds can be proven in a restricted model of computation like algebraic computation trees [Ben-Or, 1983], which have been applied to show an

$\Omega(n \log n)$ lower bound for computing the Fréchet distance between polygonal curves [Buchin et al., 2007]. Using algebraic computation trees for other problems to analyze movement, however, has so far not been promising.

Often the model of computation is not made explicit; instead, a lower bound is proven by reduction, that is, by showing that solving the problem at hand would also provide a fast solution to a different, difficult problem. A classical example of this are NP-hardness proofs which show that a problem is NP-hard by reducing an NP-hard problem to it.

NP-hardness is useful in determining whether we can hope to find a polynomial-time algorithm. But for large datasets, the difference between linear and quadratic running time may already determine whether we can compute a solution. To prove (near-)quadratic lower bounds, *conditional bounds* have proven to be useful. These are again based on reductions, but not necessarily from an NP-hard problem.

The 3SUM problem asks to find three numbers which sum to zero given a set of n real numbers. Since the 3SUM problem is assumed to have a quadratic lower bound, it has been popular to show conditional quadratic lower bounds [Gajentaan and Overmars, 1995]. However, to the best of our knowledge, there are no known reductions from 3SUM to problems concerning polygonal curves, which are of high interest to trajectory analysis.

A new series of *conditional lower bounds* emerged which makes use of the hypothesis that the satisfiability problem for CNF formulas cannot be solved much faster than by exhaustive search. These lower bounds seem promising for revealing the computational complexity of problems in the analysis of movement.

Definition 3.1. The *Strong Exponential Time Hypothesis* (SETH) states that for every $\varepsilon > 0$, there is a $k \in \mathbb{N}$ such that the satisfiability problem on k -CNF formulas with n variables and m clauses cannot be solved in $m^{O(1)}2^{(1-\varepsilon)n}$ time.

Using SETH that conjectures an exponential running time allows us to prove polynomial time lower bounds for a problem of interest. We

use the k -Orthogonal-Vectors problem, as defined by Abboud et al. [2015], to prove lower bounds on certain problems with k trajectories. It is defined as follows, using the notation $[n] := \{1, \dots, n\}$.

Definition 3.2 (k -Orthogonal-Vectors (kOV)). Suppose we are given k lists $\{\alpha_i^1\}_{i \in [n]}, \{\alpha_i^2\}_{i \in [n]}, \dots, \{\alpha_i^k\}_{i \in [n]}$ of vectors in $\{0, 1\}^d$. We need to decide whether there are k vectors $\alpha_{i_1}^1, \alpha_{i_2}^2, \dots, \alpha_{i_k}^k$ with

$$\sum_{h=1}^d \prod_{t \in [k]} \alpha_{i_t}^t[h] = 0.$$

Any such collection of vectors is called *orthogonal*.

For example, given these $\{0, 1\}$ vectors $\alpha^1, \alpha^2, \dots, \alpha^k$ with the following values

$$\begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^k \end{pmatrix} = \begin{pmatrix} 0 & 1 & \dots & 0 & 1 \\ 1 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 1 & 1 \end{pmatrix},$$

then $\alpha^1, \alpha^2, \dots, \alpha^k$ are *orthogonal* vectors.

By using a specialized version of kOV, the 2-Orthogonal-Vectors problem, we can prove lower bounds on problems on a single trajectory and on two trajectories. These proofs encode orthogonal vectors and non-orthogonal vectors differently in the reduction so that they yield specific values of the computational problem to which they want to reduce to.

Definition 3.3 (2-Orthogonal-Vectors (2OV)). Suppose we have two lists $\{\alpha_i\}_{i \in [n]}$ and $\{\beta_i\}_{i \in [n]}$ of vectors $\alpha_i, \beta_i \in \{0, 1\}^d$. We want to decide whether there is a pair α_i, β_j satisfying

$$\sum_{h=1}^d \alpha_i[h] \cdot \beta_j[h] = 0.$$

We call such a pair of vectors *orthogonal*.

The k -Orthogonal Vectors problem is linked with the SETH by the following lemma, which has been studied by Abboud et al. [2015] and Williams [2005].

Lemma 3.1. If there is an $\varepsilon > 0$ such that k -Orthogonal Vectors on n vectors in $\{0, 1\}^d$ with $d = \Omega(\log n)$ can be solved in $O(n^{k-\varepsilon})$ time, then SETH is false.

3.3 Single Curve Problems

Some of the most basic problems on movement data have been formulated as computational problems for a single trajectory. These include simplifying a trajectory, segmenting a trajectory into pieces with similar movement parameters, and reconstructing the original movement path from a trajectory with discrete locations.

An important task is simplifying a trajectory to obtain a trajectory of lower complexity which approximates the (original) trajectory sufficiently well. We give a more detailed overview of simplification in Section 3.3.1. In Section 3.3.2, we show that a subquadratic-time algorithm is unlikely to exist for the simplification problem in high dimensions.

3.3.1 Simplification

Many applications need to deal with a vast amount of movement data. New sensor technology allows researchers to collect trajectory data at an increased sampling rate. Some analyses need to reduce the complexity of a raw trajectory so that the trajectory can be compressed, stored, visualized, and analyzed further more efficiently.

To simplify a trajectory based on its geometry, the trajectory can be interpreted as a polygonal curve. Then, the problem of simplifying a trajectory can be seen as the problem of minimizing the number of vertices of the curve while sufficiently approximating and preserving the original shape of the trajectory. The extent of the approximation relies on a tolerance value that controls the quality of the simplification. The simplification problem is defined formally as follows:

Definition 3.4 (Simplification). Given

1. a *trajectory* \mathcal{T} as a sequence of n time-stamped points $(p_i, t_i) \in \mathbb{R}^{d+1}$, where p_i denotes a location in the d -dimensional plane, $t_i \geq 0$ a time stamp, and $t(p_i) = t_i$,
2. an upper bound of the number of points included in the simplification: $M \in \mathbb{N}^+$,
3. a *tolerance value* $\varepsilon > 0$ thresholding the error of the simplification, and
4. an *error criterion* $\delta(p_i p_j, \mathcal{T}[p_i p_j])$ comparing a line segment $p_i p_j$ to the corresponding subtrajectory

$$\mathcal{T}[p_i p_j] := \langle (p_i, t_i), (p_{i+1}, t_{i+1}), \dots, (p_j, t_j) \rangle$$

to decide whether $\mathcal{T}[p_i p_j]$ is at most ε far from the line segment $p_i p_j$, with respect to a distance measure, e.g., the Hausdorff distance or the Fréchet distance,

we want to find an ordered subsequence $S = \langle s_1, s_2, \dots, s_m \rangle$ of the points $\langle p_1, \dots, p_n \rangle$, such that

1. $m \leq M$
2. $t(s_j) < t(s_{j+1})$ for $1 \leq j < m$
3. $s_1 = p_1$ and $s_m = p_n$,
4. If $s_j = p_i$ and $s_{j+1} = p_k$, then $\delta(s_j s_{j+1}, \mathcal{T}[p_i p_k]) \leq \varepsilon$ for $1 \leq j < m$.

We then call S an (M, ε) -simplification of \mathcal{T} .

Another way of defining a simplification is to allow points/vertices in S which are not points from the sequence \mathcal{T} . This version of the simplification problem permits a more flexible composition of a simplification and finds simplifications of smaller tolerance values ε .

We can define the error criterion δ in various ways [Imai and Iri, 1988]. With an error criterion δ , we can choose from many similarity measures to determine the error between a subtrajectory $\mathcal{T}[p_i p_j]$ and a line segment $p_i p_j$ of the computed simplification S . Next, we present the most common error criteria for the simplification problem.

A common way to define a *tolerance zone* is by displacing the line segment $p_i p_j$ by ε and then extending this zone by half circles around

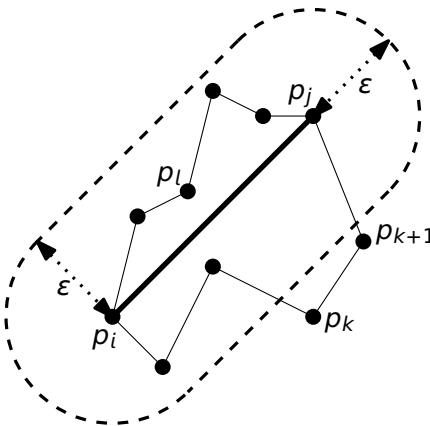


Figure 3.1: Example for the line segment model on $p_i p_j$ using the Euclidean distance d_2 . The subtrajectory $\mathcal{T}[p_i p_j]$ including p_k and p_{k+1} is not approximated by $p_i p_j$. $p_i p_j$ approximates the subtrajectory that includes p_l .

p_i and p_j with ε as the radius, as shown in Figure 3.1. This model is known as the *line segment model* [Imai and Iri, 1988]. If all points of the subtrajectory $\mathcal{T}[p_i p_j]$ lie within the tolerance zone of $p_i p_j$, then $p_i p_j$ is a valid simplification for that subtrajectory, i.e., $p_i p_j$ *approximates* $\mathcal{T}[p_i p_j]$. The line segment model corresponds to taking the Hausdorff distance [Hausdorff, 1914] as error criterion δ . Other tolerance regions have been considered. For instance, the line model takes the region between the lines displaced by ε as its tolerance zone for a line segment of the simplification. A popular method to measure similarity between curves is the Fréchet distance [Alt and Godau, 1995]. In this setting, the error criterion δ measures the Fréchet distance between $\mathcal{T}[p_i p_j]$ and $p_i p_j$, and ε represents the Fréchet distance.

The simplification problem, as defined in Definition 3.4, is so far not stated as an optimization problem. There are two ways of defining simplification as an optimization problem.

Definition 3.5 (min-# Simplification). Given \mathcal{T} , ε , and δ from Definition 3.4, a min-#-simplification is an (M, ε) -simplification for the minimal M that admits an (M, ε) -simplification. We refer to the problem of computing such a simplification as the min-# problem.

Definition 3.6 (min- ε Simplification). Given \mathcal{T}, M , and δ from Definition 3.4, a min- ε -simplification is an (M, ε) -simplification for the minimal ε that admits an (M, ε) -simplification. We refer to the problem of computing such a simplification as the min- ε problem.

For common distance measures δ , the min- ε problem can be solved using an additional $O(\log n)$ factor by the min-# problem [Imai and Iri, 1988]. Thus, a min- ε simplification can be solved by any algorithm for the min-# problem [Imai and Iri, 1988]. We therefore focus on the min-# problem and refer to a min-# simplification as the *optimal simplification*.

We now survey previous results for the problem of computing a simplification on a polygonal curve.

Algorithms for the min- ε and the min-# problems with running times $O(n^2 \log n)$ and $O(n^2)$, respectively, are known for polygonal curves in the plane [Chan and Chin, 1996]. For the L_1 -metric, Agarwal and Varadarajan [2000] presented an $O(n^{4/3+\varepsilon})$ -time algorithm.

Imai and Iri [1988] proposed one of the first optimal simplification algorithms by computing a shortest path in a directed graph as an optimal simplification. As graph we take the complete graph with the points p_i as vertices and edges oriented from smaller to larger indices. Each edge has a weight that captures the error measure of the corresponding line segment, that is, the measure δ between the line segment and the corresponding subcurve. An edge of this graph is also referred to as a *shortcut*. For a given $\varepsilon > 0$, an edge is a valid shortcut when the weight of the edge is smaller or equal to ε . The *shortcut graph* is the graph containing only valid shortcuts. A simplification can be computed by constructing the shortcut graph, and then computing the path from the first to the last vertex that uses as few edges as possible. Thus, a simplification can be computed in $O(f(n) + n^2)$ time where $f(n)$ describes the costs of constructing the shortcut graph. Chan and Chin [1996] showed that $f(n) = O(n^2)$ for the Hausdorff distance (the line segment model). The graph approach is flexible; it can be used with other error measures, such as the Fréchet distance and covering rectangles [Imai and Iri, 1986].

Applications often use non-optimal algorithms, specifically, the heuristic by Douglas and Peucker [1973]. The heuristic works as follows: it determines whether for a line segment $p_i p_j$ (initial call: $i = 1, j = n$) each $p_k \in \mathcal{T}[p_i p_j]$ is at distance ε to $p_i p_j$. If so, it returns $p_i p_j$ as a valid simplification. If not, it identifies the point p_k in $\mathcal{T}[p_i p_j]$ farthest from $p_i p_j$, then recurses on the two subproblems: $\langle p_i, p_{i+1}, \dots, p_k \rangle$ and $\langle p_k, p_{k+1}, \dots, p_j \rangle$, and outputs the concatenation of the simplifications from the subproblems as the simplification from p_i to p_j . The output of this algorithm is neither a min-# simplification nor a min- ε simplification. The worst case running time of this heuristic is $O(n^2)$. Hershberger and Snoeyink [1994] showed that the Douglas-Peucker heuristic can be implemented to run in $O(n \log n)$. The algorithm by Hershberger and Snoeyink [1998] improved that running time to $O(n \log^* n)$ for non-self-intersecting polygonal curves using the line model.

Cao et al. [2006] investigated simplifying trajectories and how to design sound query spatio-temporal operations by modeling 2-dimensional trajectories in \mathbb{R}^3 , incorporating the time stamps in an additional dimension, and projecting the data points in \mathbb{R}^3 back into the plane. Gudmundsson et al. [2009] proved the soundness of all operations for the line segment model. Additionally, Gudmundsson et al. [2009] devised an approximative version of the Douglas-Peucker heuristic for trajectories by applying the Douglas-Peucker simplification to projections of the trajectory in \mathbb{R}^3 . The running time for this algorithm is $O(n \log^2 n)$ for the line model and $O(n \log^3 n)$ for the line segment model.

Agarwal et al. [2005] devised a greedy approximation algorithm for curve simplification that runs in $O(n \log n)$ time and returns a simplification for a given ε that does not have more vertices than a min- $\varepsilon/2$ simplification. Their algorithm works with various error criteria, e.g., the Fréchet distance.

For curves in \mathbb{R}^d , Barequet et al. [2002] developed efficient algorithms. Their algorithms run in near-quadratic time for $d = 3$ and in subcubic time for $d = 4$. When the distance is measured according to the L_1 - or the L_∞ -metric, then their algorithms achieve a running time of $O(n^2)$ and $O(n^2 \log n)$ for min- ε and min-#, respectively, in any fixed dimension. In particular, for L_∞ the dependency on the dimension is only a

small-degree polynomial. It is possible to use any L_p norm. By transforming the simplification problem into the off-line ball-inclusion testing problem, Barequet et al. [2002] enhanced the graph construction for shortcuts. An efficient data structure for the off-line ball-inclusion testing problem enabled Barequet et al. [2002] to obtain the preceding running times for simplification in higher dimensions.

3.3.2 Lower Bound on Simplification

3

It is a longstanding open problem whether the (near-)quadratic running time can be improved for finding the optimal simplification, $\min\#$, for the line segment model [Agarwal and Varadarajan, 2000].¹ Furthermore, to the best of our knowledge, there has not been any lower bound established for the simplification problem so far.

We prove that, at least in a sufficiently high (non-constant) dimension, a $\min\#$ simplification cannot be computed in subquadratic time unless SETH fails. For L_∞ , our construction shows that the algorithm by Barequet et al. [2002] essentially is optimal in high dimensions, assuming SETH. We focus in this proof on a conditional lower bound on the Hausdorff distance, although the reduction also applies to the Fréchet distance.

Theorem 3.1. Assuming SETH, there is no $O(n^{2-\varepsilon})$ -time algorithm that optimally, $\min\#$ or $\min\epsilon$, simplifies a polygonal curve with n edges in \mathbb{R}^d with $d = \Omega(\log n)$ dimensions for any $\varepsilon > 0$ using ε -tolerance zones in the L_1 -, L_2 - or L_∞ -metric.

We prove this theorem by reducing the 2-Orthogonal Vectors problem to the simplification problem. Given two lists of 0/1-vectors $\{\alpha_i\}_{i \in [n]}$ and $\{\beta_i\}_{i \in [n]}$ in dimension d , we interpret each vector as a point in dimension $d + 1$, as follows: we define $\hat{\alpha}_i[h] := \alpha_i[h]$ for $1 \leq h \leq d$ and $\hat{\alpha}_i[d + 1] := -\delta$ with $\delta = 2d^2$. We define $\hat{\beta}_i[h]$ analogously, except that $\hat{\beta}_i[d + 1] := \delta$.

The idea of the reduction is illustrated in Figure 3.2. We construct a curve that moves from a starting point through all $\hat{\alpha}_i$, then passes

¹See also <http://cs.smith.edu/~orourke/TOPP/P24.html>.

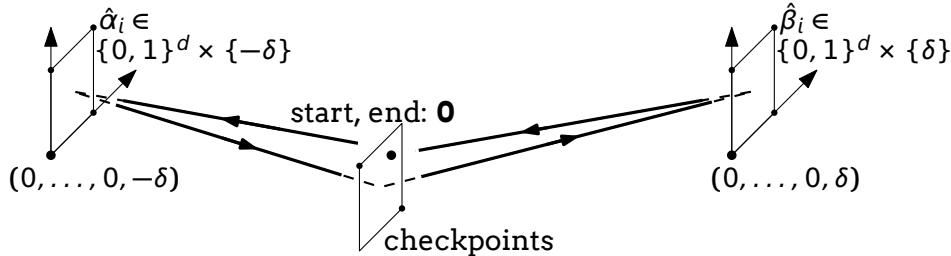


Figure 3.2: Construction for the simplification lower bound.

through d checkpoints, continues through all $\hat{\beta}_i$, and finally reaches an endpoint. The threshold ε for the simplification is chosen such that all points $\hat{\alpha}_i$ have a pairwise distance smaller than ε , and similarly for the points $\hat{\beta}_i$.

If the two corresponding vectors α and β are orthogonal, then the checkpoints q will have a distance of at most ε to the line segment from $\hat{\alpha}_i$ to $\hat{\beta}_j$, see Figure 3.3. In this case, the resulting simplification uses the starting point, one point $\hat{\alpha}_i$, one point $\hat{\beta}_j$, and the endpoint, thus four points in total.

If α and β are non-orthogonal, then some checkpoints lie outside of the tolerance zone for the embedded line segment $\hat{\alpha}_i\hat{\beta}_j$ where $\hat{\alpha}_i = 1$ and $\hat{\beta}_j = 1$, so at least one checkpoint q needs to be included in a simplification, see Figure 3.3. Hence, a simplification then consists of at least five points: the starting point, a point $\hat{\alpha}_i$, a checkpoint q , a point $\hat{\beta}_j$, and the endpoint.

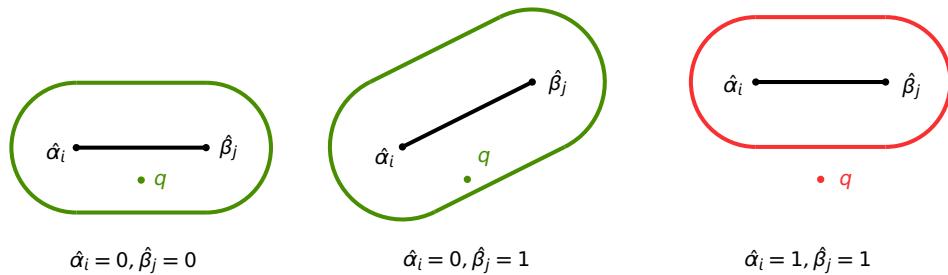


Figure 3.3: Examples for different values of $\{0, 1\}$ vectors $\hat{\alpha}_i, \hat{\beta}_j$: (i) and (ii) show that the checkpoints q for orthogonal vectors are approximated by the tolerance zone of the line segment $\hat{\alpha}_i\hat{\beta}_j$. For non-orthogonal vectors, see (iii), the checkpoints lie outside of the tolerance zone.

To complete our embedding, we need to define how we compose the input curve A for the simplification problem. Let us define $A = \langle a_0, \dots, a_m \rangle$ with $m = 2n + 2 + d$ vertices by $a_0 = a_m = (0, \dots, 0)$, $a_i = \hat{\alpha}_i$, for $1 \leq i \leq n$, $a_{n+i} = q_i$, for $1 \leq i \leq d$, and $a_{n+d+i} = \hat{\beta}_i$, for $1 \leq i \leq n$. The checkpoints $q_i \in \mathbb{R}^{d+1}$ are defined as

$$q_i[h] = \begin{cases} 0 & \text{for } h = d + 1 \\ -\delta' & \text{for } h = i \\ \frac{1}{4} & \text{otherwise,} \end{cases}$$

for $1 \leq i \leq d$, where δ' will be chosen later depending on the metric.

Proof of Theorem 3.1 under the L_∞ -metric. By setting $\varepsilon = 1$ and $\delta' = 1/2$, we impose that every simplification needs to include at least a_0 , one point $\hat{\alpha}_i$, one point $\hat{\beta}_j$, and a_m .

Assume there are orthogonal vectors α_i and β_j . Let $\ell(t)$ be the line segment between $\hat{\alpha}_i$ and $\hat{\beta}_j$ parameterized by t in the $(d+1)$ -th coordinate. For the midpoint $\ell(0)$ of the segment, we have

$$\ell(0)[h] = \begin{cases} \frac{\hat{\alpha}_i + \hat{\beta}_j}{2} \in \left\{0, \frac{1}{2}\right\} & \text{for } 1 \leq h \leq d \\ 0 & h = d + 1. \end{cases}$$

Thereby, all q_i have a distance less than 1 to $\ell(0)$ and are therefore within distance ε to the segment $\hat{\alpha}_i \hat{\beta}_j$.

In contrast, let us assume α_i and β_j are non-orthogonal. In this case, there is a coordinate $1 \leq h \leq d$ such that $\hat{\alpha}_i[h] = \hat{\beta}_j[h] = 1$. It follows that $\ell(t)[h] = 1$ for all $t \in [-\delta, \delta]$, and therefore

$$d_\infty(\ell(t), q_h) \geq 1 - q_h[h] > 1 - \varepsilon.$$

Thus, for the line segment $\hat{\alpha}_i[h] \hat{\beta}_j[h]$, q_h has a distance larger than ε to this line segment. Consequently, if there is no pair of orthogonal vectors, a simplification for distance ε requires at least five vertices because we need to include q_h . \square

Proof of Theorem 3.1 under the L_1 -metric.

We set $\varepsilon = d$ and $\delta' = 3/4d - 1/4$ to enforce to include at least four points in the simplification: a_0 , a point $\hat{\alpha}_i$, a point $\hat{\beta}_j$, and a_m .

By the same argument as for L_∞ , we induce a simplification with four vertices if α and β are orthogonal since

$$d_1(q_i, \ell(0)) \leq \frac{d-1}{4} + \delta' + \frac{1}{2} = d = \varepsilon.$$

Now again consider the case that all α_i and β_j are non-orthogonal, so there is a coordinate $1 \leq h_0 \leq d$, such that $\hat{\alpha}_i[h_0] = \hat{\beta}_j[h_0] = 1$. We show that $d_1(\ell(t), q_h) > \varepsilon$ for all $t \in [-\delta, \delta]$. We can restrict our attention to $t \in [-\varepsilon, \varepsilon]$ due to the $(d+1)$ -th coordinate. Now consider a coordinate $h \neq h_0, d+1$. If $\alpha_i[h] = \beta_j[h] = 0$, then $\ell(t)[h] = 0$. Otherwise, $\ell(0)[h] \geq 1/2$ and $\ell(t)[h] \geq \ell(0)[h](1 - \varepsilon/\delta)$ for $t \in [-\varepsilon, \varepsilon]$. Consequently, for any t we obtain

$$\begin{aligned} d_1(\ell(t), q_h) &\geq (d-1) \left(\frac{1}{2} \left(1 - \frac{\varepsilon}{\delta} \right) - \frac{1}{4} \right) + 1 - \delta' \\ &= \left[\frac{d-1}{4} + \delta' + \frac{1}{2} \right] + \frac{1}{2} - \frac{(d-1)\varepsilon}{\delta} \\ &= \varepsilon + \frac{1}{2} - \frac{(d-1)\varepsilon}{\delta} > \varepsilon. \end{aligned}$$

Thus, there is an optimal min-# simplification that uses exactly four vertices if there is an orthogonal pair. \square

Proof of Theorem 3.1 under the L_2 -metric. In this case, we set $\varepsilon = \sqrt{d}$, and we further fix $\delta' = -1/2 + \sqrt{15d + 1/4}$, which implies that $\delta' > 0$ and that $\sqrt{(d-1)/4 + (1/2 + \delta')^2} = \varepsilon$. By the choice of δ' and assuming orthogonal vectors α and β , we induce points $\hat{\alpha}$ and $\hat{\beta}$ that approximate all q_i with distances of at most ε .

Now again consider a pair of non-orthogonal vectors with $\alpha_i[h_0] = \beta_j[h_0] = 1$. It is sufficient then to prove that $d_2(\ell(t), q_h)^2 > \varepsilon^2 = d$ for $t \in [-\varepsilon, \varepsilon]$. Using the same derivation as for L_1 , we obtain

$$d_2(\ell(t), q_h)^2 \geq (1 + \delta')^2 + (d-1) \left(\frac{1}{4} - \frac{\varepsilon}{\delta/2} \right)^2.$$

The first summand is larger than $15/16d + 1/16 + 1/4$ while the second is larger than $(d-1)/16 - (d-1)\varepsilon/\delta/4 > (d-1)/16 - 1/8$. Hence, q_h has a distance larger than ε to the segment, lies outside of the line segment's tolerance zone, and enforces that the optimal simplification consists of at least five points. \square

Theorem 3.1 follows from these proofs. As a result of this, we have shown that we can reduce the 2-Orthogonal Vectors problem in d dimensions to an optimal curve simplification, min-# or min- ε , in $d+1$ dimensions for the L_1 , L_2 , and L_∞ metrics. By this lower bound, we can therefore assume that for sufficiently large dimensions, we cannot compute a simplification in subquadratic time.

3.4 Problems on Two Trajectories

Trajectories do not only occur in isolation. Researchers in application fields, including biology, geography, and sports analysis, usually track several moving entities at the same time. In this section, we survey the computational complexity of problems involving two trajectories at a time.

Determining the correlation and dependency of one moving entity to another one is fundamental to understanding the relationship between those trajectories. As mentioned earlier, we view trajectories as (time-stamped) polygonal curves, and we thus discuss computational problems concerning pairs of polygonal curves in the following.

3.4.1 Overview

The computation of similarity between two curves has been studied extensively within the last two decades. Computing similarity is the most important problem between two curves, and the analysis of interaction between trajectories makes heavy use of existing similarity measures.

First, we review approaches on computing interaction between trajectories briefly. Then, we survey methods for computing similarity in more detail by addressing how they can be used to construct an alignment between the trajectories.

Interaction is the inter-dependency between moving objects [Doncaster, 1990]. Various aspects of interaction can be captured by detecting the underlying movement patterns of leadership, single file, or following. Naturally, these types of movement patterns apply to two and more trajectories.

Andersson et al. [2008] developed an algorithm to detect the movement pattern of leadership by following a leader in the vicinity of a circular sector. Their algorithm to detect leadership runs in linear time and space for two trajectories. For k trajectories, their algorithm runs in $O(k^2n)$ time and $O(nk)$ space for discrete time and in $O(k^2n \log k)$ time and $O(nk)$ space for continuous time.

Similar to leadership, a single file captures a follow-behind relationship by a leadership of one moving entity and all others follow each other [Buchin et al., 2008, 2010]. Buchin et al. [2008] expressed a follow-behind relationship by one trajectory moving along a similar path as the other, but with a (possibly varying) time shift. This follow-behind relationship is computable for two trajectories in $O(nk_{avg})$ time and $O(n + k_{max})$ space, where k_{avg} (resp. k_{max}) is the average (resp. maximum) number of data points from the second trajectory that have a suitable time shift relative to a data point of the first curve.

3.4.2 Alignment Methods

A central problem for two trajectories is computing their similarity in shape in order to capture the inter-dependency between the movements. We survey the most common techniques to align two trajectories. These methods usually compute only a single value to express the similarity. Any of the methods, that we discuss, however, can be modified to output a monotone matching between the trajectories. A monotone matching starts with an edge at the first point of each trajectory and ends with an edge at the last point of each trajectory.

Monotone matchings require that if a point p on one trajectory \mathcal{T}_1 is matched to a point q of another trajectory \mathcal{T}_2 , then any point after p on \mathcal{T}_1 needs to be matched to either q or a point succeeding q on \mathcal{T}_2 . Such a matching provides insights into the structure of the inter-dependency between the trajectories because the monotonicity allows us to capture local events between the moving entities while optimizing a global criterion for the alignment. A discrete matching contains only locations of the input trajectories; whereas for a continuous matching, we allow all points that yield a monotone matching.

We discuss distances between trajectories instead of similarity measures because a distance can be easily converted to a similarity measure and vice versa by taking the inverse.

The *Fréchet distance* is an intuitive measure that describes the resemblance of shapes between two curves, which, naturally, applies to two trajectories. It minimizes the maximum distance for any monotone matching between the curves. Alt and Godau [1995] provided the first known algorithm for the continuous Fréchet distance between polygonal curves which runs in $O(n^2 \log n)$ time. Eiter and Mannila [1994] developed an $O(n^2)$ -time algorithm for the discrete Fréchet distance.

The *fine-grained complexity* of the (discrete) Fréchet distance between two curves has recently attracted a lot of attention. After a long period without major progress, Agarwal et al. [2014] devised a sub-quadratic $O\left(\frac{mn \log \log n}{\log n}\right)$ -time algorithm for the discrete Fréchet distance on the word RAM. Buchin et al. [2014] developed a randomized algorithm for the continuous Fréchet distance with a running time slightly better than the classic bound of $O(n^2 \log n)$ [Alt and Godau, 1995]. Buchin et al. [2012] introduced a new notion of locally correct Fréchet matchings where every submatching of the matching between the curves minimizes its local Fréchet distance. A locally correct Fréchet matching is computable in $O(n^3 \log n)$ time for the continuous Fréchet distance and in $O(n^2)$ time for the discrete Fréchet distance.

Answering a question posed by Buchin et al. [2014], Bringmann [2014] showed that the (discrete) Fréchet distance cannot be computed in

$O(n^{2-\varepsilon})$ time, for any $\varepsilon > 0$, assuming the *Strong Exponential Time Hypothesis* (SETH). Bringmann and Mulzer [2016] refined and extended this result. They proved this lower bound by reducing 2OV to the discrete Fréchet distance. Moreover, Bringmann and Mulzer [2016] showed that there cannot be an 1.399-approximation for the Fréchet distance in subquadratic running time assuming SETH.

An important alignment method for time series is *Dynamic Time Warping* (DTW). DTW minimizes the sum of distances on a monotone matching between the trajectories. We can transform the classical $O(n^2)$ -time dynamic program for the discrete Fréchet distance into DTW by replacing the max-operation by a sum. Berndt and Clifford [1994] defined DTW and the original algorithm using dynamic programming to compute an alignment in $O(n^2)$ time.

Applications in data mining, speech recognition, and database systems have used DTW extensively in the past. Due to growing datasets, heuristics and approximation schemes for DTW have been developed to boost the running time. Salvador and Chan [2007] developed FastDTW, a linear-time heuristics for DTW, assuming the time series can be simplified and subsampled in linear time. Their multi-level approach which recursively projects a DTW path of a lower resolution to one of higher resolution performed well in practice. The approximation error depends on the chosen resolution of the recursion. Al-Naymat et al. [2009] proposed a heuristic, SparseDTW, which uses pruning to omit cells in the DTW computation. The running time, however, can still be $O(n^2)$ in the worst case.

Recently, Gold and Sharir [2016] devised an exact algorithm for computing DTW on one-dimensional time series that runs in $O(n^2 \log \log \log n / \log \log n)$ time. It is possible to extend their algorithm to compute DTW in \mathbb{R}^d with $d > 1$ assuming the distance metric used is polyhedral.

The work by Bringmann [2014] triggered proofs of a conditional lower bound also for dynamic-time warping [Abboud et al., 2015; Bringmann and Künnemann, 2015]. Assuming SETH, DTW cannot be computed in less than $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$. Abboud et al. [2015] proved this lower bound from a variant of the 2OV problem on 0/1-strings over an

alphabet of size five. Bringmann and Künemann [2015] gave a proof for the same lower bound on a 1D curve of binary strings of values $\{0, 1, 2, 4, 8\}$ in \mathbb{R} , in which they encoded each coordinate of a curve in a four-bit representation.

The *Edit Distance* (ED) is a popular method to align strings, correct spelling, or process natural languages [Wagner and Fischer, 1974]. We can employ a similar dynamic program as for DTW where we substitute the sum of the distances by the sum of unit costs. In addition, we require checking whether two elements $p_i \in \mathcal{T}_A$ and $q_j \in \mathcal{T}_B$ are the same or close $d(p_i, q_j) \leq \varepsilon$. If this is the case, we do not charge the simultaneous movement of both entities within the matching with a unit. The dynamic program for ED runs in $O(n^2)$ time.

Since ED has primarily been designed for and applied to mostly discrete sequences, such as strings, and not for numerical values, Chen et al. [2005] adapted ED for such sequences as *Edit Distance on Real Sequences* (EDR).

Masek and Paterson [1980] devised an $O(n^2 / \log n)$ -time algorithm for ED. After a long period, Gold and Sharir [2016] improved the upper bound on ED to an $O(n^2 \log \log \log n / \log \log n)$ running time for a geometric version of the ED, and they adapted their algorithm for DTW to ED.

Similar to DTW, Abboud et al. [2015]; Bringmann and Künemann [2015] showed that a subquadratic-time algorithm for ED cannot exist assuming the SETH.

The *Longest Common Subsequence* (LCSS) is the classic method to find the longest subsequence of one string in another string [Maier, 1978]. For trajectories, LCSS essentially translates into finding the longest subtrajectory that is similar to another trajectory. LCSS can be seen as a restricted and simpler version of ED because the dynamic programs are identical apart from that only two of the three operations in the ED are allowed within LCSS; namely, LCSS does not permit *insertion* operations. This yields an $O(n^2)$ running time for the dynamic program.

To show that LCSS cannot be computed in subquadratic time assuming SETH, Bringmann and K  nnemann [2015] and Abboud et al. [2015] reduced from a version of the 2OV problem to LCSS. Abboud et al. [2015] used in their proof strings over an alphabet of size seven, whereas Bringmann and K  nnemann [2015] used binary strings, in which each coordinate value of the curve is encoded by a five-bit string.

3.5 Problems on Multiple Trajectories

Most datasets involve more than just one or two individuals. Some analyses can easily be extended to multiple moving entities, but for others, a novel, specific analytical methodology needs to be designed. Usually, these methods are computationally more involved because computations on multiple trajectories at the same time are expensive. However, little is known about what determines whether a polynomial-time algorithm can exist for such problems and when faster alternatives, such as approximations or heuristics, needs to be devised.

In this section, we therefore survey both upper and lower bounds for problems involving multiple trajectories. We conclude by proving a lower bound on the Fr  chet distance for multiple trajectories.

3.5.1 Overview

Buchin et al. [2011] devised an algorithm to cluster similar subtrajectories by employing an approximation algorithm for the Fr  chet distance on the concatenation of the trajectories. Their algorithm under the discrete Fr  chet distance runs in $O(n^2 + nk\ell)$ time and uses $O(n\ell)$ space, where ℓ is the length of each subtrajectory measured in time stamps. ℓ is bounded by n/k , and $k\ell$ is in order of n . Under the continuous Fr  chet distance, we can compute a cluster of subtrajectories in $O(n^2\ell)$ time that uses $O(n\ell^2)$ space.

If a clustering algorithm assigns labels to clusters, then this problem in the context of trajectories is also known as classification. A classi-

fication may stem from either a clustering or a segmentation, but it can also be directly computed. Alewijnse et al. [2017] gave an $O(m^2 + km \log m)$ -time algorithm to classify k trajectories using m parameter values in a parameterized movement model. Buchin et al. [2016] explored a new concept of flow diagrams to fill the gap between classification and clustering. A flow diagram is a sequence of activities that represents many state sequences. They showed that computing a minimal flow diagram is $\text{W}[1]$ -hard if the number of state sequences is variable, developed several heuristics for flow diagrams and evaluated them experimentally on varying parameterizations.

Naturally, we can define an alignment between k trajectories at the same time (see Section 3.4.2). To find an alignment among multiple moving entities simultaneously, we need to extend our notion of a distance norm to multiple trajectories.

Dumitrescu and Rote [2004] defined the Fréchet distance on k curves. The Fréchet distance is the minimal longest pairwise distance over all monotonous matchings among the trajectories then. Dumitrescu and Rote [2004] devised a 2-approximation algorithm by constructing the Fréchet distance from the pairwise distances on the k curves.

To compute a matching among k trajectories simultaneously, we can adapt the aforementioned $O(n^2)$ -time algorithms for computing DTW, LCSS, the Edit distance, and the Fréchet distance on two curves into dynamic programs with an $O(n^k)$ running time.

Little is known about lower bounds for those alignment methods on multiple trajectories. Abboud et al. [2015] showed a lower bound on LCSS that there cannot be an algorithm in sub $O(n^{k-\varepsilon})$ time for any $\varepsilon > 0$ and k strings over an alphabet of size $O(k)$ assuming SETH.

It is still an open problem to show conditional lower bounds for other alignment methods, such as DTW and the Edit distance, on multiple curves. We show a similar lower bound as Abboud et al. [2015] for the discrete Fréchet distance on k curves, as discussed in the following section.

3.5.2 Lower Bound on the Fréchet Distance

Theorem 3.2. For any $\varepsilon > 0$, the discrete Fréchet distance of k planar point sequences of length n cannot be computed in $O(n^{k-\varepsilon})$ time, unless SETH fails.

We show the lower bound on the discrete Fréchet distance between k curves by a reduction from the k -Orthogonal Vectors problem. First, we need to introduce some notation, so that we can prove Theorem 3.2.

Let A_1, \dots, A_k be k sequences of points in the plane, and for each $i = 1, 2, \dots, k$, we set $A_i = \langle a_1^i, \dots, a_{n_i}^i \rangle$. By $a_j^i[h]$, for $h = 1, 2$, we denote the h -th coordinate of a_j^i . We set $S = [n_1] \times [n_2] \times \dots \times [n_k]$.

We define a *coupling* of length m on S as a sequence $\mathcal{C} = \langle \mathcal{C}_1, \dots, \mathcal{C}_m \rangle$ such that we have $\mathcal{C}_i \in S$, $\mathcal{C}_1 = (0, 0, \dots, 0)$, $\mathcal{C}_m = (n_1, n_2, \dots, n_k)$, and $\mathcal{C}_{i+1}[h] = \mathcal{C}_i[h]$ or $\mathcal{C}_{i+1}[h] = \mathcal{C}_i[h] + 1$, for all $i = 0, \dots, m-1$ and $h = 1, \dots, k$. A coupling \mathcal{C} defines an alignment of the curves A_1, \dots, A_k , and we define the *coupled distance* as

$$d_{\mathcal{C}}(A_1, \dots, A_k) := \max \left\{ d(a_{\mathcal{C}_i[h]}^h, a_{\mathcal{C}_i[h']}^{h'}) \mid 0 \leq i \leq m, 1 \leq h, h' \leq k \right\}$$

where d denotes the Euclidean distance. Let C be the set of all possible couplings on A_1, \dots, A_k ; then, the *discrete Fréchet distance* is defined as

$$d_F(A_1, \dots, A_k) := \min \{ d_{\mathcal{C}}(A_1, \dots, A_k) \mid \mathcal{C} \in C \}.$$

Next, we describe our reduction.

Proof of Theorem 3.2. Suppose we have k lists $\{\beta_i\}_{i \in [n]}, \{\alpha_i^t\}_{i \in [n]}, t \in [k-1]$, of vectors $\alpha_i^t, \beta_i \in \{0, 1\}^d$. From these vector lists, we want to construct k curves $B, A^1, A^2, \dots, A^{k-1}$. The encoding of curve B is slightly different from the ones of the other $k-1$ curves. The discrete Fréchet distance among $B, A^1, A^2, \dots, A^{k-1}$ will be 1 if the given vector lists contain a collection of k orthogonal vectors, and strictly larger than 1, otherwise.

The coordinates of the vectors are encoded by *coordinate gadgets* (CG), see Figure 3.4. Set $\delta := 1/100$, and for $i = 1, \dots, k - 1$, let

$$CG_i(0) := \langle (-0.5 - \delta, 0), (0.5, 0), (-0.5 - \delta, 0), \dots, (0.5, 0), (-0.5 - \delta, 0) \rangle$$

be a curve with $2k - 1$ vertices. We define $CG_i(1)$ as having the same vertices as $CG_i(0)$, except that the $2i$ -th vertex is replaced by $(0.5 + \delta, 0)$. Further, we define

$$CG_B(0) := \langle (-0.5, 0), (0.5, 0), (-0.5, 0), \dots, (0.5, 0), (-0.5, 0) \rangle$$

with $2k - 1$ vertices and $CG_B(1)$ in the same way, but with only $2k - 3$ vertices. We call the vertices at $(0.5, 0)$ *short spikes* and the vertices at $(0.5 + \delta, 0)$ *long spikes*.

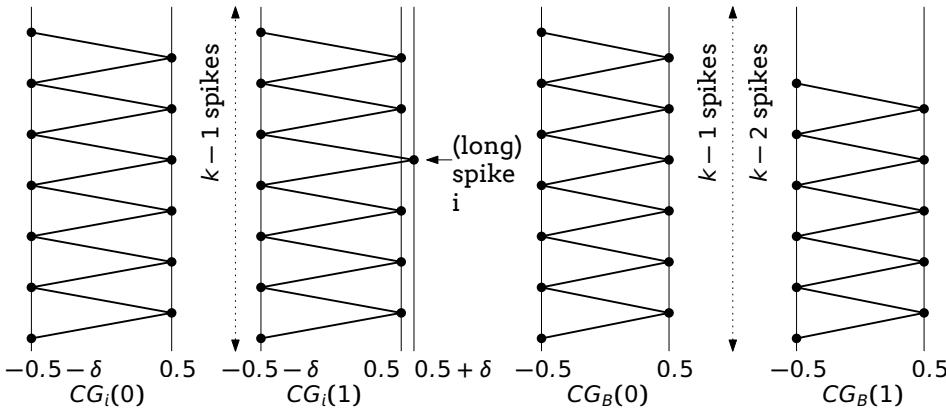


Figure 3.4: Coordinate gadgets (distorted vertically for the purpose of illustration).

Suppose that there were a coupling of $CG_1(1), CG_2(1), \dots, CG_{k-1}(1), CG_B(1)$ achieving a distance of at most 1. Then, we need to couple each of the $k - 1$ long spikes of $CG_1(1), \dots, CG_{k-1}(1)$ with a different spike of $CG_B(1)$. But this is not possible since $CG_B(1)$ has only $k - 2$ spikes; thus, $d_F(CG_1(1), \dots, CG_B(1)) > 1$. If we replace any $CG_*(1)$ with a respective curve $CG_*(0)$, the distance becomes 1.

If $CG_B(0)$, then $CG_B(0)$ has $k - 1$ spikes and thus can accommodate all long spikes; if $CG_i(0)$, then there are at most $k - 2$ long spikes. These can all be accommodated by CG_B , and short spikes of the CG_i can simply be coupled with a $(-0.5, 0)$ on CG_B .

Next, we encode the vectors and the vector lists into the curves A_1, A_2, \dots, A_{k-1} and B by concatenating the coordinate gadgets. The construction is depicted in Figure 3.5.

For each coordinate of the vectors, we use the coordinate gadgets as aforementioned. Between the coordinates of a vector, we use a point $c := (0, 0.8661)$ to “synchronize” among the coordinate gadgets from different vectors. The start of the vectors will be demarcated by $v_A := (-0.499, -1)$ and $v_B := (0, -0.8661)$. Additionally, we will use the points $t_A = (0.48, -0.01)$ and $t_B = (0.57, 1.005)$ to mark a successful synchronized traversal, and $s = (-0.499, 0)$ as a point that is close to all except t_B , see Figure 3.5.

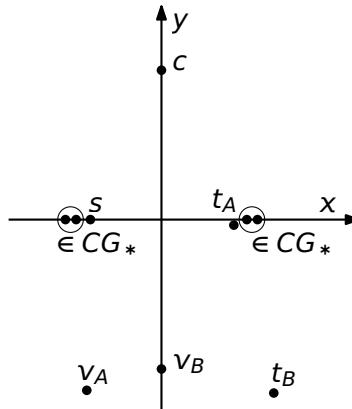


Figure 3.5: The points used as vertices of the curves.

Two points are said to be *close* if their distance is at most 1: s is close to all points except t_B , and t_A is close to all points except v_A . The point c is close only to s and t_A (and itself); t_B is close only to t_A and v_B , v_A is close only to s and v_B ; v_B only to s , t_A and t_B .

Now we can compose the curves A_1, A_2, \dots, A_{k-1} and B from the vectors. We denote \circ as the operation of adding a vertex to a curve or of concatenating curves. Let $A_i^j := s \circ v_A \circ \bigcup_{h=1}^d (CG_j(\alpha_i^j[h]) \circ c) \circ t_A$ be the curve of the vector element α_i^j . By concatenating the constructed curves A_i^j , we set $A^j := \left(\bigcup_{i=1}^n A_i^j \right) \circ s$ as the representation of the vector α^j . Furthermore, we define $B_i := v_B \circ \bigcup_{h=1}^d (CG_j(\beta_i[h]) \circ c)$ and $B := s \circ v_A \circ \bigcup_{i=1}^n B_i \circ t_B \circ s$ similarly.

First, we argue that k vectors $\alpha_{i_1}^1, \alpha_{i_2}^2, \dots, \alpha_{i_{k-1}}^{k-1}, \beta_{i_k}$ are orthogonal if and only if the corresponding concatenated coordinate gadgets have Fréchet distance of at most 1. If the vectors are orthogonal, then in each coordinate, at least one vector has a 0-entry, and a coupling with a distance of at most 1 is possible. On the other hand, if the vectors are not orthogonal, there is one coordinate in which all vectors have 1-entries. The c vertices then force us to traverse all coordinates simultaneously so that we will have to couple k one-coordinate gadgets, yielding a Fréchet distance larger than 1.

Now let us consider the vector lists and the complete curves when they are orthogonal. If the vector lists contain a k -tuple $\alpha_{i_1}^1, \alpha_{i_2}^2, \dots, \alpha_{i_{k-1}}^{k-1}, \beta_{i_k}$ of orthogonal vectors, then the corresponding curves A^1, \dots, A^{k-1}, B have a Fréchet distance of at most 1. We can observe this by the following coupling: first, A^1 walks to the first point s of $A_{i_1}^1$ while all other curves wait at s . Then, A^2 walks to the first point s of $A_{i_2}^2$ while all other curves wait at s , etc. Finally, B walks to the first point v_B of B_{i_k} while all other curves wait at s . Since s is close to all points except for t_B , the distance so far is 1. Then, the A^j curves simultaneously jump to v_A while B waits at v_B , and subsequently the coordinate gadgets are traversed simultaneously. Next, the A^j curves wait at t_A while B walks to the last point s . The A^j curves then simultaneously move to the next s , and finish the traversal to the final vertex one after another while the other curves wait at s .

Next, suppose that the curves A^1, \dots, A^{k-1}, B have a Fréchet distance larger than 1. We argue then that there is a k -tuple of orthogonal vectors. Indeed, suppose that no such k -tuple exists, and consider the first time that B reaches t_B . Since t_B is close only to t_A and v_B , at this point, all A^j must be at t_A . It follows that before that, all A^j 's must have been simultaneously at v_A because on the construction of A^j 's, v_A comes before t_A , and v_A is close only to s and v_B . For the same reason, at this point, B also must be at v_B . Thus, the coordinate gadgets of a k -tuple of vectors are traversed simultaneously, leading to a Fréchet distance larger than 1 since all k -tuples are non-orthogonal. Theorem 3.2 follows. \square

Our construction also rules out a faster polynomial-time approximation scheme for the Fréchet distance on k curves unless SETH fails. We computed the coordinates by hand, and they could be optimized to prove a specific approximation lower bound.

3.6 Conclusions

By studying the computational complexity of specific analyses, we surveyed many computational problems on trajectory data. This view enabled us give insights for visual analytics systems into what kind of analyses can be computed efficiently in terms of running time. By reviewing existing lower bounds and proving new ones, we guide researchers in their decision-making when to resort or to employ approximation algorithms or heuristics. We will now reflect on this chapter's application to the thesis as a whole.

Simplification of polygonal curves is applicable to many domains and problems. Since new sensors collect data at higher sampling rates, preprocessing of trajectory data becomes also pertinent to any analyses on trajectories. However, little is known whether simplifications can be computed in near-linear time. Our lower bound suggests that this might be not the case at least in sufficient high dimensions. Little is also known about how we can compute a simplification consistently across many scales. In Chapter 4, we explore how to compute simplifications progressively which has been motivated by the visual information-seeking mantra [Shneiderman, 1996] from information visualization. Furthermore, the lack of lower bounds prompted us to explore new efficient representations for shortcut graphs that apply to progressive as well as non-progressive simplification algorithms.

For two trajectories, finding an alignment between long sequences in near-quadratic time for all important similarity measures is very involved. Furthermore, our lower bound on the Fréchet distance among multiple curves and the lower bound for LCSS by Abboud et al. [2015] suggest that finding an alignment among multiple individuals is a challenging task. These insights led us to focus on small-scale

datasets with a sampling rate, mimicking continuous movement, in computing interaction events (see Chapter 5). We will also use a dataset of three pigeons moving together to show how an alignment can be constructed on a triplet.

Computing an optimal clustering for a variable state space is W[1]-hard [Buchin et al., 2016]. Thus, to compute a clustering efficiently, we are in need of approximation algorithms or heuristics. We therefore implemented a heuristic to compute a clustering, in this case an aggregation of stopovers, for gull data in Chapter 6.

4

Progressive Simplification

4.1 Introduction

4

Given a polygonal curve as input, the *curve simplification* problem asks for a polygonal curve that approximates the input well and that uses as few vertices as possible. Because of the importance of data reduction, curve simplification has a wide range of applications. Cartography is one such application in which the visual representation of line features like rivers, roads, or boundaries of regions needs to be reduced. Nowadays, maps are interactive, so we need curve simplification that works with different levels of details. The visual information-seeking mantra [Shneiderman, 1996] states “Overview first, zoom and filter, then details-on-demand”. A natural way to follow this mantra would be to simplify for each zoom level independently. This however would have the drawback that the resulting simplifications would not be consistent among different scales. Therefore, we require *progressive simplification*, that is, a series of simplifications for which the level of detail is progressively increased for higher zoom levels. This is shown in Figure 4.1a.

Existing progressive algorithms, e.g., [Cao et al., 2006], work by simplifying a curve, then simplifying the previous simplification, and so on. More concretely, a common approach is to first discard the vertex, whose removal introduces the smallest error (according to some criterion); then, we proceed by removing the vertex with the small-

est error from the simplified curve and so on. For instance, the algorithm by Visvalingam and Whyatt [1993] always removes the vertex, that together with its neighboring vertices forms the smallest area triangle.

Such approaches stand in stark contrast to (non-progressive) curve simplification algorithms, which aim to minimize the complexity of the simplification while guaranteeing a (global) bound on the error introduced by simplifying. The most prominent algorithm with a global error bound is the algorithm by Douglas and Peucker [1973]. However, while heuristically aiming at a simplification with few vertices, it does not actually minimize the number of vertices. Imai and Iri [1988] introduced a general approach for the problem of minimizing the number of vertices in a simplification. Their approach uses *shortcut graphs*, which we describe in more detail below. In line with these algorithms, the goal of our work is to develop algorithms that solve progressive simplification as an optimization problem.

A (vertex-restricted) *simplification* \mathcal{S} of a polygonal curve \mathcal{C} is an ordered subsequence of \mathcal{C} (denoted by $\mathcal{S} \subseteq \mathcal{C}$) that includes the first and the last point of \mathcal{C} . An ε -*simplification* \mathcal{S} is a simplification that ensures that each edge of \mathcal{S} has a distance of at most ε to its corresponding subsurve, wherein the distance measure can, for instance, be the Hausdorff or the Fréchet distance [Alt and Godau, 1995]. For an ordered pair of points (p_i, p_j) of \mathcal{C} , we denote the distance between the segment (p_i, p_j) and the corresponding subchain by $\varepsilon(p_i, p_j)$. We denote by $(p_i, p_j) \in \mathcal{S}$ that (p_i, p_j) is an edge of \mathcal{S} .

We now define the *progressive simplification problem* in the plane:

Definition 4.1. Given a polygonal curve $\mathcal{C} := \langle p_1, p_2, \dots, p_n \rangle$, where each point p_i of \mathcal{C} lies in the plane \mathbb{R}^2 , and a sequence $\langle \varepsilon_1, \dots, \varepsilon_m \rangle$, where $0 < \varepsilon_1 < \varepsilon_2 < \dots < \varepsilon_m$, we want to compute a sequence of (vertex-restricted) simplifications $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m$ such that

1. $\mathcal{S}_m \subseteq \mathcal{S}_{m-1} \subseteq \dots \subseteq \mathcal{S}_1 \subseteq \mathcal{C}$ (*monotonicity*),
2. \mathcal{S}_i is an ε_i -simplification of \mathcal{C} , and
3. $\sum_{i=1}^m |\mathcal{S}_i|$ is minimal.

We refer to a sequence of simplifications fulfilling the first two conditions as *progressive simplification*. A *minimal progressive sim-*

plification fulfills all three conditions, and the problem of computing such a sequence is called the *progressive simplification problem*. In this chapter, we present an $O(n^3m)$ -time algorithm for the progressive simplification problem in the plane.

The cornerstone of progressive simplification is that we require monotonicity (see 1. in Definition 4.1) between simplifications of different scales: $S_i \subseteq S_{i-1}$, as illustrated in Figure 4.1a. This guarantees that, when “zooming out”, only vertices are removed and cannot (re)appear. As an error measure, we will mostly use the Hausdorff distance [Hausdorff, 1914]. This is not essential to the core algorithm, and we will discuss how to use the Fréchet distance [Alt and Godau, 1995] or area-based measures [Daneshpajouh et al., 2012] without affecting the worst-case running time. Furthermore, our algorithm generalizes to the *weighted* version of the problem in which $\sum_{i=1}^m w_i |S_i|$ with positive weights w_i is minimized, and to the *continuous* version, where S_ε needs to be computed for all $0 \leq \varepsilon < \varepsilon_M$ wherein ε_M is the error at which we can simplify the curve by the single line segment (p_1, p_n) ; thus, we have $\varepsilon_M = \varepsilon(p_1, p_n)$. As in the discrete setting, we require $S_{\varepsilon'} \subseteq S_\varepsilon$ for $\varepsilon' > \varepsilon$; the resulting algorithm minimizes $\int_0^{\varepsilon_M} |S_\varepsilon| d\varepsilon$ in $O(n^5)$ time.

In our algorithms, we will make use of the *shortcut graph* as introduced by Imai and Iri [1988]. Given a polygonal curve C , a *shortcut* is an ordered pair $(i < j)$ of vertices. Given an error $\varepsilon > 0$, a shortcut (p_i, p_j) is *valid* if $\varepsilon(p_i, p_j) \leq \varepsilon$. The shortcut graph $G(C, \varepsilon)$, as shown in Figure 4.1b, represents all valid shortcuts (p_i, p_j) with $1 \leq i < j \leq n$. A bottleneck in computing (progressive) simplifications is the construction and space usage of these graphs. We therefore devise sev-

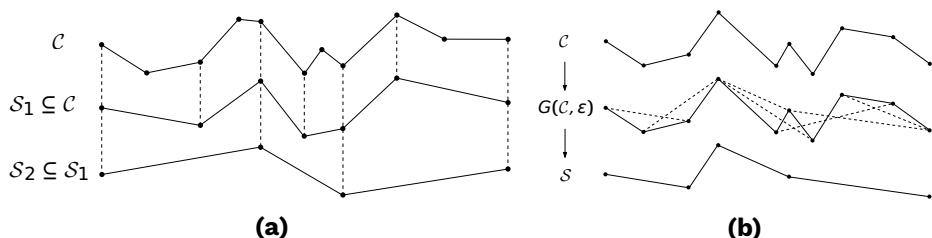


Figure 4.1: Examples for (a) progressive and (b) global curve simplification.

eral new techniques for computing these graphs more efficiently with respect to time and space. These techniques apply to progressive simplifications as well as (non-progressive) min-# simplifications. A min-# simplification algorithm minimizes the number of vertices in a simplification for a given ε .

First, we present an algorithm that efficiently constructs shortcut graphs independently of an error ε , which makes computations on different scales more efficient as well. To date, it has been known only how to compute shortcut graphs in subcubic time if an error ε is given upfront. We show how $\varepsilon(p_i, p_j)$ can be computed for an arbitrary line segment (p_i, p_j) without fixing an error in advance by employing convex hulls in the computation of the graph. This construction is also of interest for min- ε simplification, in which ε needs to be minimized for a given bound on the number of points in the simplification.

Secondly, we introduce a representation of the shortcut graph that employs so-called *shortcut intervals*. Shortcut graphs are computed to perform shortest path calculations, and as we will later argue and demonstrate in an experimental evaluation, computing shortest paths using shortcut intervals typically takes only $O(n \log n)$ time. This result tends to be more generally applicable for min-# simplifications. In our experiments, we compare our minimal progressive simplification algorithm with several natural heuristics and evaluate our techniques for shortcut graphs.

4.2 Related Approaches

Curve simplification is a well-studied problem in the past 30 years due to its importance to applications in various domains. We survey the most-related previous results on this problem.

Imai and Iri [1988] introduced a general approach to min-# simplification using the shortcut graph, and they provided corresponding algorithms. Similar algorithms were presented in a series of papers [Melkman and O'Rourke, 1988; Toussaint, 1985], resulting in algorithms with a running time of $O(n^2)$ for min-# simplifications, and with a

running time of $O(n^2 \log n)$ for min- ε simplifications [Chan and Chin, 1996] under the Hausdorff distance. These algorithms are based on computing the shortcut graph for a specific error and then using binary search for the min- ε problem.

For the L_1 -metric, Agarwal and Varadarajan [2000] presented an $O(n^{4/3+\varepsilon})$ -time algorithm. For this, they use a clique-cover representation for the shortcut graph. Agarwal et al. [2005] devised a greedy approximation algorithm that runs in $O(n \log n)$ time which given $\varepsilon > 0$ guarantees that it does not have more vertices than an optimal $\varepsilon/2$ -simplification.

Applications often use heuristics, in particular the Douglas-Peucker simplification [Douglas and Peucker, 1973]. The output of this algorithm is neither a min-# simplification, nor a min- ε simplification, nor is the algorithm progressive.

Progressive simplifications are used in cartography [Qingsheng et al., 2002]. A popular progressive algorithm is the one by Visvalingam and Whyatt [1993] that always removes the point in a series of simplifications, that is part of the triangle with the smallest area. Inspired by this, Daneshpajouh et al. [2012] defined an error measure for non-progressive simplification by measuring the sum or the difference in area between a simplification and the input curve. Cao et al. [2006] referred to progressive curve simplification as “aging”. They developed a heuristic to this problem by iteratively simplifying previously computed simplifications instead of the input curve.

4.3 Computing Simplifications Progressively

We will show how to solve the progressive simplification problem for curves in the plane in $O(n^3 m)$ time in Section 4.3.1. The same running time holds for the weighted version, and based on this fact we are able to show that the continuous progressive simplification problem can be solved in $O(n^5)$ time. Our proofs apply to polygonal curves in higher dimensions. As faster alternatives, we discuss greedy heuristics in Section 4.3.2, which we evaluate experimentally in Section 5.5.

4.3.1 Optimal Progressive Simplifications

By the monotonicity property of the progressive simplification problem (see 1. in Definition 4.1), we require that all points within a simplification S_k of the sequence must also occur within all subsequent simplifications S_l with $k < l$. Adding points to a simplification thus is a crucial step as well as maintaining shortcuts. We, therefore, associate a cost value $c_{ij}^k \in \mathbb{N}$ for each shortcut (p_i, p_j) in the shortcut graph $G(\mathcal{C}, \varepsilon_k)$ of a simplification S_k . In Sections 4.4 and 5.5, we use the Hausdorff distance as an error measure to determine whether a shortcut is valid, but because the shortcut graph is flexible in using any error measure, we can employ any other distance measure for our algorithms. In particular, for the Fréchet distance [Alt and Godau, 1995] and area-based distances [Daneshpajouh et al., 2012], we can simply compute whether a shortcut is valid in $O(n)$ time and therefore use these measures without changing the worst-case running time. We obtain a cost value c_{ij}^k for a shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$ by minimizing the costs of all possible shortcuts in $\langle p_i, \dots, p_j \rangle$ at lower scales recursively. The dynamic program is defined as follows:

$$c_{ij}^k = \begin{cases} 1 & \text{if } k = 1 \\ 1 + \min_{\pi \in \prod^{k-1}(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} c_{xy}^{k-1} & \text{if } 1 < k \leq m \end{cases}$$

We use $\prod^k(p_i, p_j)$ to denote the set of all paths in $G(\mathcal{C}, \varepsilon_k)$ from p_i to p_j .

We compute all cost values from scale $k = 1$ up to m by assigning a weight c_{ij}^k to each shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$. For each shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$, we compute c_{ij}^k by finding a shortest path π in $G(\mathcal{C}, \varepsilon_{k-1})$ from p_i to p_j , minimizing $\sum_{(p_x, p_y) \in \pi} c_{xy}^{k-1}$ thereby.

We can use any shortest path algorithm. We opted to employ Dijkstra's algorithm that we need to run per scale k on $O(n)$ source nodes of $G(\mathcal{C}, \varepsilon_k)$. This yields a worst case running time of $O(n^3m)$ because this algorithm takes in $O(n^2)$ time on graphs with integer weights.

We increment $c_{ij}^k = c_{ij}^{k-1} + 1$ for any shortcut $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_{k-1})$. By doing so, we avoid recomputations of shortest paths and reuse cost values whenever necessary.

We construct the sequence of simplifications from \mathcal{S}_m down to \mathcal{S}_1 . First, we compute \mathcal{S}_m by returning the shortest path from p_1 to p_n in $G(\mathcal{C}, \varepsilon_m)$ using the computed cost values at scale m . Next, we compute a shortest path P from p_i to p_j in $G(\mathcal{C}, \varepsilon_{m-1})$ for all shortcuts $(p_i, p_j) \in \mathcal{S}_m$. Simplification \mathcal{S}_{m-1} is then constructed by linking these paths P with each other. We build all other simplifications in this manner until \mathcal{S}_1 is constructed.

Correctness

We will now prove that this algorithm returns a valid and minimal solution for the progressive simplification problem.

By constructing the simplifications from scale m down to 1, it follows that, for any shortcut $(p_i, p_j) \in \mathcal{S}_k$ with $1 < k \leq m$, there exists a subsequence $\langle p_i, \dots, p_j \rangle \subseteq \mathcal{S}_{k-1}$. We therefore have $\mathcal{S}_k \subseteq \mathcal{S}_{k-1}$. Furthermore, each simplification \mathcal{S}_k has a maximum Hausdorff distance ε_k to \mathcal{C} since we link only edges from $G(\mathcal{C}, \varepsilon_k)$.

It remains to show that we minimize $\sum_{i=1}^m |\mathcal{S}_i|$. We therefore define a set of shortcuts \mathcal{S}_k^{ij} for any $1 \leq i < j \leq n$ and $1 \leq k \leq m$ as:

$$\mathcal{S}_k^{ij} = \{ (p_x, p_y) \in \mathcal{S}_k \mid x \leq i < j \leq y \}.$$

Thus, \mathcal{S}_k^{ij} includes all line segments of \mathcal{S}_k that span the subcurve $\langle p_i, \dots, p_j \rangle$ with an error of at most ε_k to \mathcal{C} . $|\mathcal{S}_k^{ij}|$ then is the number of shortcuts in simplification \mathcal{S}_k covering (p_i, p_j) .

Lemma 4.1. If the line segment (p_i, p_j) is part of simplification \mathcal{S}_k , then the associated cost value c_{ij}^k satisfies $c_{ij}^k = \sum_{\ell=1}^k |\mathcal{S}_\ell^{ij}|$ for any $1 \leq k \leq m$ and $1 \leq i < j \leq n$.

Proof. We show $c_{ij}^k = \sum_{\ell=1}^k |\mathcal{S}_\ell^{ij}|$ by induction on k using the following inductive hypothesis: for any $n \geq y > x \geq 1$, if $(p_x, p_y) \in \mathcal{S}_k$, then $c_{xy}^k = \sum_{\ell=1}^k |\mathcal{S}_\ell^{xy}|$ (IH).

Base $k = 1$: Take any shortcut $(p_i, p_j) \in S_1$. We therefore observe that $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_1)$. Furthermore, we know $S_1^{ij} = \{(p_i, p_j)\}$, and therefore $|S_1^{ij}| = 1$. This yields $c_{ij}^1 = 1 = \sum_{\ell=1}^k 1 = \sum_{\ell=1}^k |S_1^{ij}|$.

Step $k > 1$: Take any line segment $(p_i, p_j) \in S_{k+1}$. Thus, we observe $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_{k+1})$, $S_{k+1}^{ij} = \{(p_i, p_j)\}$, and $|S_{k+1}^{ij}| = 1$.

Consider any $1 \leq \ell \leq k$ and a path $\pi \in \prod^k(p_i, p_j)$ such that $\sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}|$ is minimal. We now derive that $\pi = S_\ell^{ij}$ such that S_ℓ^{xy} is minimal for all $(p_x, p_y) \in \pi$. Note that $\pi = S_\ell^{ij} \subseteq G(\mathcal{C}, \varepsilon_\ell) \subseteq G(\mathcal{C}, \varepsilon_k)$ since $\varepsilon_k \geq \varepsilon_\ell$. We observe that π is both in $\prod^\ell(p_i, p_j)$ and $\prod^k(p_i, p_j)$. It thus follows that:

$$\min_{\pi \in \prod^k(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}| = \min_{\pi \in \prod^\ell(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}|. \quad (4.1)$$

From $\pi = S_\ell^{ij}$, it follows that $S_\ell^{xy} \cap S_\ell^{yz} = \emptyset$ for any (p_x, p_y) and (p_y, p_z) in π . Combining S_ℓ^{xy} for all $(p_x, p_y) \in \pi$ yields a non-overlapping sequence of shortcuts from p_i to p_j . This gives us:

$$|S_\ell^{ij}| = \min_{\pi \in \prod^\ell(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}|. \quad (4.2)$$

We now derive the following:

$$\begin{aligned} c_{ij}^{k+1} &\stackrel{(IH)}{=} 1 + \min_{\pi \in \prod^k(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} \sum_{\ell=1}^k |S_\ell^{xy}| \stackrel{(4.1)}{=} 1 + \sum_{\ell=1}^k \min_{\pi \in \prod^\ell(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}| \\ &\stackrel{(4.2)}{=} 1 + \sum_{\ell=1}^k |S_\ell^{ij}| \quad |S_{k+1}^{ij}| = \{(p_i, p_j)\} \sum_{\ell=1}^{k+1} |S_\ell^{ij}| \end{aligned}$$

□

Theorem 4.1. Given a polygonal curve with n points in the plane and $0 \leq \varepsilon_1 < \dots < \varepsilon_m$, a minimal progressive simplification can be computed in $O(n^3m)$ time under distance measures for which the validity of a shortcut can be computed in $O(n)$ time. This includes the Fréchet, Hausdorff, and area-based measures.

Proof. It remains to be proven that the combined size of the simplifications computed by our algorithm is minimal. Let $\langle S'_1, \dots, S'_m \rangle$ be a sequence of simplifications of a minimal progressive simplification, and let $\langle S_1, \dots, S_m \rangle$ be the sequence computed by the algorithm. We need to prove that $\sum_{\ell=1}^m |S_\ell| \leq \sum_{\ell=1}^m |S'_\ell|$ holds.

Let us derive the following:

$$\begin{aligned} \min_{\pi \in \prod^m(p_1, p_n)} \sum_{(p_x, p_y) \in \pi} c_{xy}^m &\stackrel{\text{Lemma 4.1}}{=} \min_{\pi \in \prod^m(p_1, p_n)} \sum_{(p_x, p_y) \in \pi} \sum_{\ell=1}^m |S_\ell^{xy}| \\ &\stackrel{(4.1)}{=} \sum_{\ell=1}^m \min_{\pi \in \prod^\ell(p_1, p_n)} \sum_{(p_x, p_y) \in \pi} |S_\ell^{xy}| \stackrel{(4.2)}{=} \sum_{\ell=1}^m |S_\ell|. \end{aligned}$$

Hence, the algorithm produces a simplification that minimizes the overall cost function; our algorithm produces a progressive set of simplifications in which each simplification consists of edges from the corresponding shortcut graph such that the cumulative number of vertices is minimized. We also know that any minimal simplification S'_k is a path in $G(\mathcal{C}, \varepsilon_k)$ since it strictly connects shortcuts with an error of at most ε_k . We conclude from this that $\sum_{\ell=1}^m |S_\ell| \leq \sum_{\ell=1}^m |S'_\ell|$ holds. \square

We now consider two variants of the progressive simplification problem: the *weighted* progressive simplification for which the objective is to minimize the weighted cumulative size of the simplifications; and the *continuous* progressive simplification. They are formally defined as follows:

Definition 4.2 (Weighted Progressive Simplification). Given a polygonal curve $\mathcal{C} := \langle p_1, p_2, \dots, p_n \rangle$, where each point p_i of \mathcal{C} lies in the plane, a sequence $\langle \varepsilon_1, \dots, \varepsilon_m \rangle$, where $0 \leq \varepsilon_1 < \varepsilon_2 < \dots < \varepsilon_m$, and a sequence of corresponding weights w_1, w_2, \dots, w_m , where each $w_j \in \mathbb{R}^+$, we want to compute a sequence of (vertex-restricted) simplifications S_1, S_2, \dots, S_m of \mathcal{C} such that

1. $S_m \subseteq S_{m-1} \subseteq \dots \subseteq S_1 \subseteq \mathcal{C}$ (*monotonicity*),
2. S_i is an ε_i -simplification of \mathcal{C} , and
3. $\sum_{i=1}^m w_i |S_i|$ is minimal.

Definition 4.3 (Continuous Progressive Simplification). Given a polygonal curve $\mathcal{C} := \langle p_1, p_2, \dots, p_n \rangle$, where each point p_i of \mathcal{C} lies in the plane and we want to compute a sequence S_1, S_2, \dots, S_m with $\{\varepsilon_1, \dots, \varepsilon_m\}$, where ε_m is the error at which we can simplify \mathcal{C} to the line segment (p_1, p_n) and $0 \leq \varepsilon_1 < \varepsilon_2 < \dots < \varepsilon_m$, such that

1. $S_m \subseteq S_{m-1} \subseteq \dots \subseteq S_1 \subseteq \mathcal{C}$ (*monotonicity*),
2. S_i is an ε_i -simplification of \mathcal{C} , and
3. $\int_0^m |S_i| d\varepsilon$ is minimal.

For both of these problems, we can employ our preceding algorithm to compute simplifications progressively. We first show how to adapt our algorithm for the weighted progressive simplification problem; then we prove how to solve the continuous simplification problem.

Recall that in the *weighted* progressive simplification problem, we are additionally given weights w_1, \dots, w_m and need to minimize $\sum_{\ell=1}^m w_\ell |S_\ell|$ (see Definition 4.3). We achieve this by adapting the definition of c_{ij}^k : For each shortcut (p_i, p_j) in $G(\mathcal{C}, \varepsilon_1)$, we have a cost of $c_{ij}^1 = w_1$ instead of 1; furthermore, $c_{ij}^k = w_k + \min_{\pi \in \prod^{k-1}(p_i, p_j)} \sum_{(p_x, p_y) \in \pi} c_{xy}^{k-1}$, for each (p_i, p_j) in $G(\mathcal{C}, \varepsilon_k)$ where $k > 1$. Note that the proofs above are trivially extended to apply to this updated cost function. The weighted progressive simplification allows us to solve the continuous progressive simplification problem. We now show the following Theorem:

Theorem 4.2. Given a polygonal curve with n points in the plane, a minimal continuous progressive simplification can be computed in $O(n^5)$ time under distance measures for which the validity of a shortcut can be computed in $O(n)$ time. This includes the Fréchet, Hausdorff, and area-based measures.

Proof. Consider the maximal errors $\varepsilon(p_i, p_j)$ of all possible line segments (p_i, p_j) with $i < j$ with respect to the Hausdorff distance (or another distance measure). Then let $\varepsilon := \langle \varepsilon_1, \dots, \varepsilon_{\binom{n}{2}} \rangle$ be the sorted sequence of these errors based on their value. Let M be the index of the corresponding ε_M in this sorted sequence ε for the line segment (p_1, p_n) ; thus, $\varepsilon_M = \varepsilon(p_1, p_n)$. Note that it is possible that $M < \binom{n}{2}$, but there is no reason to use any $\varepsilon > \varepsilon_M$ since at this point we already have simplified the curve to a single line segment, (p_1, p_n) .

In a minimal-size progressive simplification, it holds that $S_\varepsilon = S_{\varepsilon_i}$ for all $\varepsilon \in [\varepsilon_i, \varepsilon_{i+1})$. This can be shown by contradiction: if S_ε would be smaller, we could decrease the overall size by setting all $S_{\varepsilon'}$ with $\varepsilon' \in [\varepsilon_i, \varepsilon]$ to S_ε . Therefore, in a minimal continuous progressive simplification we have

$$\int_0^{\varepsilon_M} |S_\varepsilon| d\varepsilon = \sum_{k=1}^{M-1} (\varepsilon_{k+1} - \varepsilon_k) |S_{\varepsilon_k}|.$$

Thus, we can solve the continuous progressive simplification problem by reducing it to the weighted progressive simplification problem with $O(n^2)$ values ε_k . \square

4.3.2 Greedy Heuristics

4

Ideally, we would like to be able to browse between simplifications of different scales in any direction. To accomplish this, we need an efficient way to compute not only from scale 1 up to m , which we define as *bottom-up* ordering, but also from scale m down to 1, which we refer to as *top-down* ordering.

Greedy approaches for simplification have helped in the past to construct efficient approximation algorithms [Agarwal et al., 2005] and heuristics [Cao et al., 2006] for (non-progressive) curve simplification. By ignoring all costs, we can reuse our optimal algorithm top-down and greedily. For a bottom-up construction, we just need to ensure that we reuse points from the previous scale.

Due to the greedy choice and the monotonicity constraint within the progressive simplification, many potential shortcuts are pruned. We integrate a new pruning technique into the efficient shortcut graph construction by Chan and Chin [1996]. When constructing a simplification top-down progressively, we want to obtain all shortcuts $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$ where $x \leq i < j \leq y$ for some $(p_x, p_y) \in S_{k+1}$. Similarly, for constructing simplifications progressively bottom-up, we prune the search space to all shortcuts $(p_i, p_j) \in G(\mathcal{C}, \varepsilon_k)$ where $(p_i, p_j) \in S_{k-1}$. A bottom-up construction prunes more drastically than a top-down construction by excluding particular points.

Cao et al. [2006] devised an efficient heuristic for simplifying polygonal curves progressively from a finer to a coarser scale. Instead of pruning graph $G(\mathcal{C}, \varepsilon_k)$ progressively, they construct a simplification \mathcal{S}_k for scale ε_k by computing it on $G(\mathcal{S}_{k-1}, \varepsilon_k)$. Thereby, the simplifications cascade recursively, which means that any guarantees on the error of these simplifications with respect to \mathcal{C} are lost. We therefore obtain a new bound $\sum_{\ell=1}^k \varepsilon_\ell$ on the error between \mathcal{S}_k and \mathcal{C} . This heuristic works with any min-# simplification.

4.4 Constructing the Shortcut Graph for Arbitrary Scale

In this section, we present a novel algorithm to compute the maximum error $\varepsilon(p_i, p_j)$ for any ordered ($i < j$) pair (p_i, p_j) with respect to the Hausdorff distance (or any other distance measure) in $O(n^2 \log n)$ time instead of $O(n^3)$ time [Chan and Chin, 1996]. This technique can be applied to both non-progressive simplification algorithms and progressive simplification algorithms to speed up the computation of the maximum error $\varepsilon(p_i, p_j)$ of any line segment (p_i, p_j) with $i < j$.

Cao et al. [2006] showed experimentally that the shortcut computation by Chan and Chin [1996] is fastest for the Euclidean distance. This shortcut representation, however, requires fixing an error value in advance. To facilitate the construction of shortcut graphs for various error criteria, we are interested in computing the maximum error $\varepsilon(p_i, p_j)$ between any shortcut (p_i, p_j) and its induced subcurve $\langle p_i, \dots, p_j \rangle$. This allows us to efficiently determine for any given error bound whether a shortcut is valid or not. This is particularly useful for continuous progressive simplification, for which we would otherwise need to compute a quadratic number of shortcut graphs, thus spending $O(n^4)$ time on computing shortcut graphs.

We can easily compute such a shortcut graph by annotating each edge (p_i, p_j) with its maximum error $\varepsilon(p_i, p_j)$. This would take $O(n^3)$

time if we compute $\varepsilon(p_i, p_j)$ for each edge separately. In this section, we discuss how this can be improved to $O(n^2 \log n)$.

For a given $p_i \in \mathcal{C}$, we construct a convex hull $CH = \{q_1, \dots, q_\ell\}$ where $CH \subseteq \mathcal{C}$ in which we incrementally insert all points p_j where $i \leq j \leq n$. After inserting a point p_j , we find the extreme points X_t^{ij} and X_b^{ij} on CH using the upward normal \vec{n}_{ij} and downward normal $-\vec{n}_{ij}$ of the line segment (p_i, p_j) . An extreme point on CH in the direction of a vector \vec{n} is any point $x \in CH$ such that there cannot be a point $y \in CH$ with $\vec{n} \cdot (\vec{y} - \vec{x}) > 0$. See Figure 4.2a for an example.

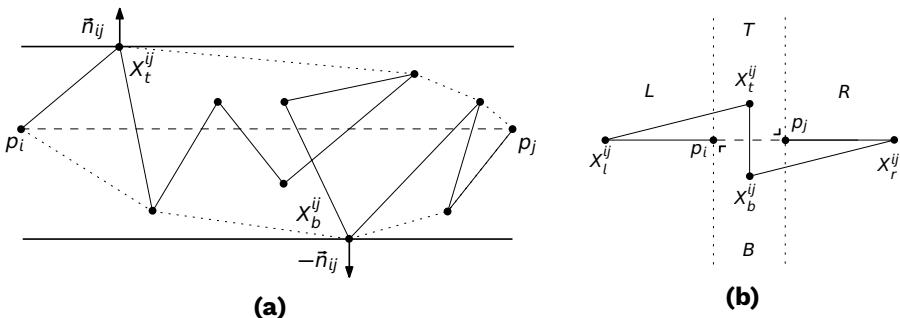


Figure 4.2: The convex hull of $\{p_i, \dots, p_j\}$. (a) Extreme points X_t^{ij} and X_b^{ij} with respect to (p_i, p_j) . (b) Division of the convex hull into regions L , R , T and B .

The two extreme points on the convex hull for line segment (p_i, p_j) do not always yield the furthest point on the corresponding subcurve. An example of this is shown in Figure 4.2b. To resolve this, we need to subdivide the area around (p_i, p_j) into four regions: T (op), B (ottom), L (eft), and R (ight). The maximum error value of shortcut (p_i, p_j) is the maximum distance from the furthest point in each region to (p_i, p_j) .

For region T , we compute the furthest point X_t^{ij} by finding an extreme point in the upper convex hull CH_t . The furthest point X_b^{ij} within region B for the lower convex hull CH_b is computed analogously. Both convex hulls are represented as balanced binary search trees ordered x -monotonically. We obtain X_t^{ij} and X_b^{ij} by a binary search on the normals of the line segments of CH_t and CH_b respectively.

By obtaining candidates X_{tl}^{ij} and X_{bl}^{ij} on CH_t and CH_b respectively, we compute X_t^{ij} . We continue our discussion for computing X_{tl}^{ij} on CH_t ,

which is analogous to obtaining X_{bl}^{ij} on CH_b .

To compute X_{tl}^{ij} , we annotate each root node p_r of a subtree $T_r \in CH_t$ with the furthest point in T_r to p_i . The root of the binary search tree is therefore annotated with the point in CH_t furthest from p_i . An example of such a tree annotation is shown in Figure 4.3.

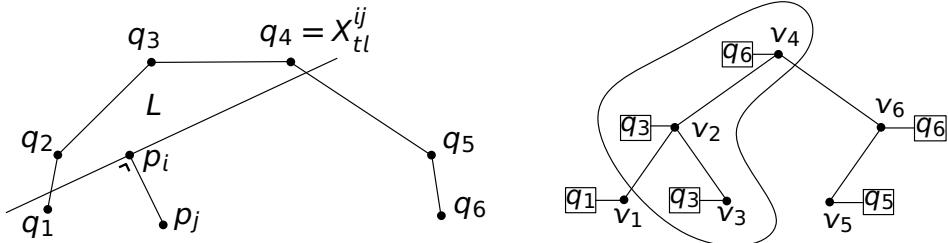


Figure 4.3: Annotating the binary search tree of the convex hull CH_t to find X_{tl}^{ij} .

4

We employ range queries to isolate subtrees which lie completely inside L . To isolate those subtrees, we traverse the search tree and check for every subtree whether the left-most and right-most point contained in the subtree are inside L . In this case, then by the convexity of the hull, the entire subtree lies inside L . This then means we can consider its annotation as a candidate for X_{tl}^{ij} .

To determine X_r^{ij} , we cannot reuse this approach since p_i is the first point added to the convex hull and p_j the last. Therefore, we need to run the annotation algorithm on the reversed sequence of \mathcal{C} as well, similarly to Chan and Chin [1996]. During the forward traversal, we compute X_t^{ij} , X_b^{ij} and X_l^{ij} incrementally; during the backward traversal, we obtain X_r^{ij} .

4.5 Compressing the Shortcut Graph

For many types of spatial data, such as movement trajectories with a high sampling rate, consecutive points p_i and p_j are expected to be spatially close. We therefore presume that, if (p_x, p_i) is a valid shortcut for some point p_x , then (p_x, p_j) might be a shortcut too. This ob-

servation inspired us to exploit so-called *shortcut intervals*, which are contiguous subsequences of \mathcal{C} with which a particular point forms shortcuts. By employing a set of shortcut intervals, we can typically find a shortest path for an arbitrary pair of points in $O(n \log n)$ time instead of $O(n^2)$. A similar approach has been used by Alewijnse et al. [2014] to speed up trajectory segmentation. Our new shortcut representations can be used with any simplification algorithm that uses shortcut graphs.

4.5.1 Shortcut Graph Construction

We propose a novel representation of a shortcut graph called a *shortcut interval set*. It is a minimally sized set of shortcut intervals covering all shortcuts: $I(\mathcal{C}, \varepsilon) = \langle I_1(\varepsilon), \dots, I_n(\varepsilon) \rangle$ in which $I_i(\varepsilon)$ is a set of shortcut intervals starting in p_i . Formally, any shortcut interval for a point p_i and tolerance value ε is a maximal interval $[x, y]$ where all shortcuts from p_i to p_j for $x \leq j \leq y$ are valid for ε .

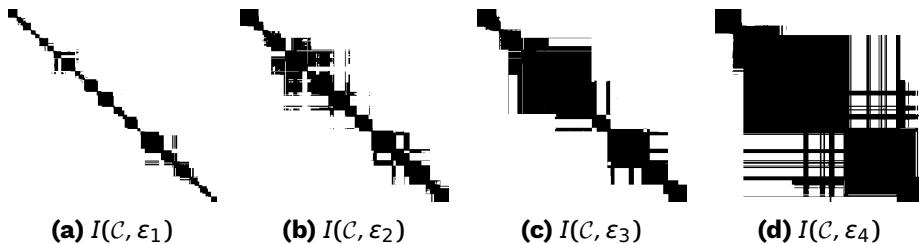


Figure 4.4: Shortcut intervals sets of a polygonal curve for four different error criteria.

In Figure 4.4, we show a shortcut interval set as a matrix; the shading of a cell (i, j) with $1 \leq i, j \leq n$ indicates whether (p_i, p_j) is a valid shortcut. Observe that independent of the tolerance value ε , every column or row within the matrix has only a few shaded regions. Hence, we expect $|I_i(\varepsilon)|$ to be of constant size in practice, so $I(\mathcal{C}, \varepsilon)$ has linear space complexity in experimental settings. This representation is thus typically an order of magnitude smaller than explicitly storing the shortcut graph $G(\mathcal{C}, \varepsilon)$.

To handle degenerated cases, Chan and Chin [1996] used in their algorithm to compute shortcut graphs efficiently in $O(n^2)$ time two sets of shortcuts. After computing them, they intersected these sets to obtain the shortcut graph. Since both have sizes of $O(n^2)$ in the worst case, computing such an intersection takes $O(n^2)$. We can speed up this computation of the intersection of these sets by not representing them explicitly but by employing shortcut intervals instead. Given the two shortcut interval sets $I'(\mathcal{C}, \varepsilon)$ and $I''(\mathcal{C}, \varepsilon)$, we want to compute the intersection by obtaining an interval set $I_i(\varepsilon)$ such that each $I_i(\varepsilon)$ contains the overlap of the intervals in $I'_i(\varepsilon)$ with the intervals of $I''_i(\varepsilon)$. An example of such an intersection is shown in Figure 4.5a. We can compute this efficiently in $O(n)$ time by sweeping over the shortcut intervals of both sets $I'_i(\varepsilon)$ and $I''_i(\varepsilon)$ simultaneously, wherein computing each encountered overlap takes $O(1)$ time. Since each set contains typically $O(n)$ shortcut intervals, this operation runs in $O(n)$ time in practice.

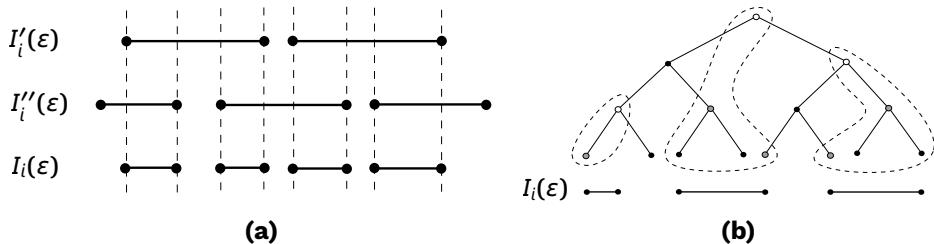


Figure 4.5: Exploiting shortcut intervals. (a) Intersecting $I'(\mathcal{C}, \varepsilon)$ and $I''(\mathcal{C}, \varepsilon)$ to obtain $I(\mathcal{C}, \varepsilon)$. (b) Finding the shortest path from p_i to p_t by performing multiple range queries on T for each interval in $I_i(\varepsilon)$.

4.5.2 Finding Shortest Paths

To compute a simplification as a shortest path from some point p_s to another point p_t , we can use breadth-first search in $O(n^2)$ time in unweighted shortcut graphs. By using shortcut intervals, we can improve the running time to $O(n \log n)$ in practice. Our construction of shortest paths can be employed in any simplification algorithm.

First, we construct a balanced binary search tree T containing all points $\langle p_s, \dots, p_t \rangle$ ordered by their indices. Suppose we have a subtree T_r rooted at $p_r \in T$; our objective is to annotate p_r with the shortest path from p_r to p_t as well as the shortest path from any point in T_r to p_t . Hence, every node and subtree in T has an annotation.

Next, we annotate T by inserting all points from p_t down to p_s . We annotate each node p_i by the result of a range query on T for every shortcut interval in $I_i(\varepsilon)$. A range query for a shortcut interval $[x, y] \in I_i(\varepsilon)$ isolates all subtrees for which (p_i, p_j) is a valid shortcut, $p_j \in T$ and $x \leq j \leq y$. We can then determine the shortest path from a point p_i to p_t by retrieving the subtree annotations. See Figure 4.5b for an example. After inserting all points, we return the annotation of node p_s which is the shortest path from p_s to p_t .

Depending on the input curve and the error value, we presume that $|I_i(\varepsilon)|$ is typically of size $O(1)$ for an arbitrary point p_i . There are cases when the shortcut intervals in $I_i(\varepsilon)$ cover a small number of points. In such a case, a shortcut interval is so small that performing the range query is more time-consuming than simply checking the node annotation of all points of that interval. There are $O(n^2)$ intervals in the worst case, yielding a worst-case running time of $O(n^2 \log n)$, which is slower than breath-first search in $O(n^2)$ time.

Therefore, we compute the shortest path for shortcut intervals $[x, y]$ where $y - x < c \log n$ for some constant $c \in \mathbb{R}^+$ by brute force in $O(y - x)$ time. By doing so, we obtain a worst-case running time of $O(n^2)$ for this method.

4.6 Experimental Evaluation

We experimentally evaluated various simplification algorithms in the progressive setting and on different shortcut representations. Our motivation for studying this problem was the visualization of trajectories at varying scales, and so we used trajectory data from a migrating griffon vulture [Schmidt-Rothmund, 2017] to evaluate our algorithms.

This trajectory has high granularity and spans a large distance, making it highly suitable for progressive simplification. We conducted our experiments on a 64-bit Intel Core i7-2630QM machine with 8 gigabytes of DDR3 SDRAM. All code was written in C# 6.0.

We ran our progressive simplification algorithm on a sample of 5000 points and on ten scales. The associated error criteria are sampled linearly from the 10% smallest errors of shortcuts on that input. All shortcut graphs are constructed using the algorithm by Chan and Chin [1996] and represented as shortcut interval sets. We used Dijkstra's algorithm and employed pairing heaps as priority queues [Fredman et al., 1986] in our progressive simplification algorithm.

The minimal progressive simplification algorithm (*II Opt.*), depicted in red in Figure 4.6a, has a similar cumulative simplification size as the minimal simplification in a non-progressive simplification setting (*II No Hier.*). Because greedy choices at earlier (coarser) scales propagate errors on the construction of simplifications at finer scales, the cumulative size of the progressive simplification for a top-down ordering (*II TD*) is significantly larger than for any other algorithm. The bottom-up construction (*II BU*) shows better performance because it starts with the least aggressive greedy choice while pruning the shortcut graph the most. *II BU Naive* is the approximate version of *II BU* in which we progressively simplify previous simplifications instead of using \mathcal{C} (see [Cao et al., 2006] and Section 4.3.2). This algorithm outperforms all other Imai and Iri [1988] algorithms in terms of the cumulative size of the simplification and the running time. The Douglas-Peucker simplification (*DP*) [Douglas and Peucker, 1973] has the fastest running time for any length of the input curve (Figure 4.6b), but has larger simplification sizes.

We wanted to explore which shortcut graph construction algorithm facilitates simplifying curves on many scales, so we compared the running time of constructing a shortcut graph using convex hulls with the construction from Chan and Chin [1996] for a varying ε . We evaluated how sensitive a construction algorithm for the shortcut graph is with respect to multiple scales. To perform range queries on the annotated convex hulls, we used left-leaning red-black trees [Sedgewick,

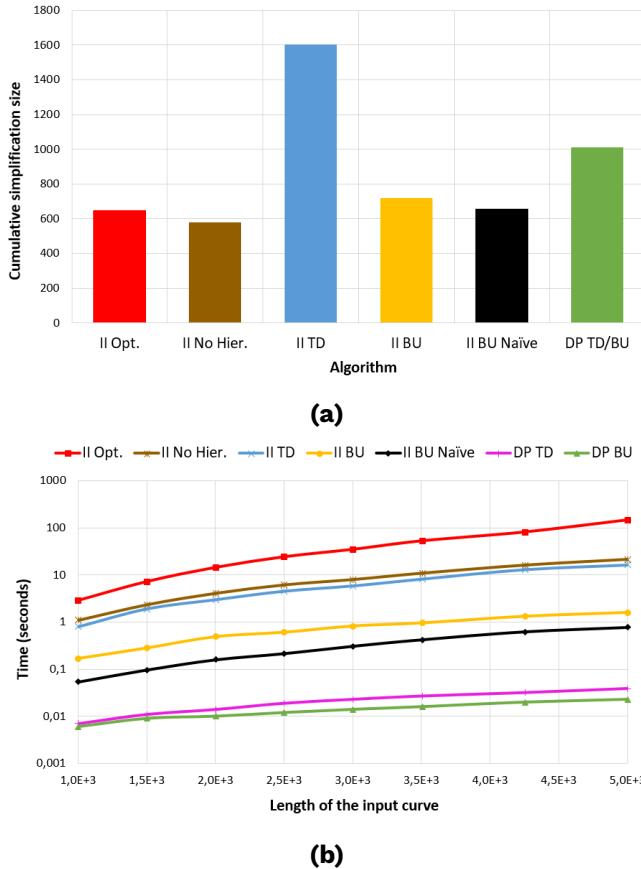


Figure 4.6: Performance of progressive simplification algorithms on 10% of the smallest errors using 10 scales. (a) Comparison of the cumulative simplification size for 5000 points. (b) Running time in seconds on a log-scale with respect to the length of the input curve.

2008]. Employing convex hulls involves precomputation costs for building them; for a few scales, this can take up to 10 minutes (see Figure 4.7a). However, these computational costs for building convex hulls amortize for more than 65 scales which is when our construction starts to outperform the algorithm by Chan and Chin [1996].

Finally, we investigated the link between time and space complexity when representing shortcut graphs using shortcut intervals. We observed that constructing shortcut interval sets has a near-linear running time for varying lengths of the input curve; whereas, construct-

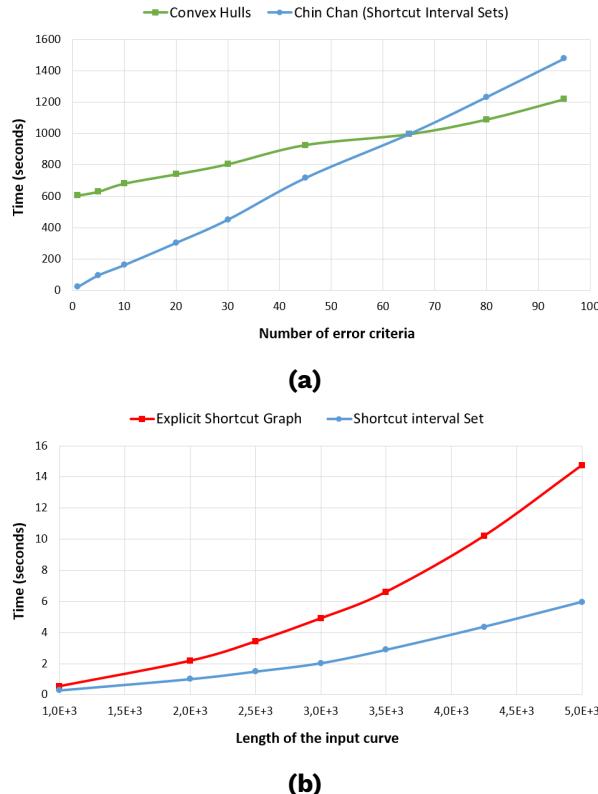


Figure 4.7: Running times for constructing the shortcut graph. (a) Constructing different types of shortcut graphs on 5000 points. (b) Comparison of the shortcut interval sets to explicit shortcut graphs for varying lengths of the input curve. We construct shortcut graphs with the smallest 10% of errors on the input curve.

ing shortcut graphs explicitly shows a running time that is above $O(n)$ (see Figure 4.7b). For varying ε , our experiments suggest that shortcut intervals use linear storage, as opposed to superlinear space complexity that the representation by Imai and Iri [1988] uses (Figure 4.8a). Our experiments indicate a linear running time for range queries on shortcut interval sets (see Figure 4.8b), which is drastically faster than running a quadratic-time breadth-first search (BFS).

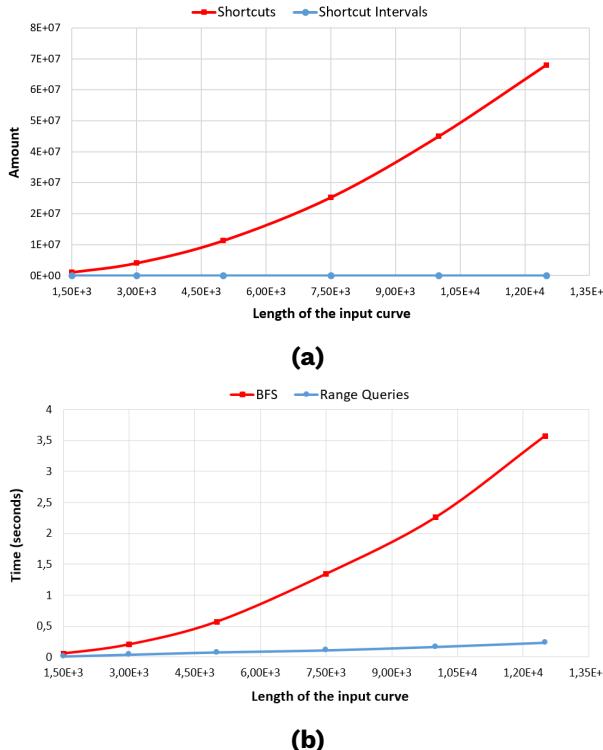


Figure 4.8: Results for exploiting shortcut intervals for error criterion $\varepsilon = 0.06$ and varying lengths of the input curve. (a) Comparing the space complexity between using explicit shortcut graphs versus shortcut interval sets. (b) Running time comparison of finding a shortest path from the start to the end of the curve, using breadth-first search (BFS) or range queries on shortcut interval sets.

4.7 Conclusions

We have presented the first algorithm that computes minimum-complexity progressive simplifications given a polygonal curve with n points in the plane. Our algorithm runs in $O(n^3m)$ time for m discrete scales and $O(n^5)$ time for continuous scaling. To facilitate progressive simplifications on many scales, we present a technique for efficiently computing the maximum error for every potential shortcut on \mathcal{C} in $O(n^2 \log n)$ time. This technique facilitates computing both

non-progressive simplifications and progressive simplifications. Furthermore, we developed a storage-efficient representation for the shortcut graph that is capable of finding shortest paths in $O(n \log n)$ time, which is also applicable to any simplification algorithm that uses a shortcut graph.

Our experimental evaluation shows that our progressive algorithm is effective, yet too slow for larger trajectory data. As a future work, it would be interesting to improve the worst-case running time for both the discrete and the continuous case. Our experiments further indicate that greedy heuristics that simplify bottom-up provide a reasonable alternative in practice.

In general, it would be interesting to develop a near-linear time simplification algorithm, which might be of benefit to compute progressive simplifications faster. Our results in Chapter 3.3.2 suggest that we depend on new approximation algorithms and heuristics for the simplification problem to accomplish this because it might not be possible to compute an optimal simplification in subquadratic time.

Finally, the experiments show that our new representations of the shortcut graph are efficient both in computing the error ε for all edges and in computing shortest paths. It would be of interest to compute shortcut intervals faster because this would make the min-# simplification more viable for large data.

5

Visual Analytics of Delays and Interaction

5.1 Introduction

In a wide range of applications, such as biology, urban planning, sports, or ecology, movement data are being collected in the form of trajectories. In recent years, technological advances have led to a rapid improvement in the ability to record trajectory data [Nathan and Giuggioli, 2013]. New technology allows scientists to collect data of high resolution, over long durations, and for a large number of simultaneously moving entities. Coupled with methodological advancements, movement data offer an opportunity to better understand the mechanisms and behavioral ecology guiding collective motion. In order to explore such data interactively, this research integrates interaction and delayed responses on movement data in a new visual analytics tool.

Interaction is the inter-dependency in the movements of two trajectories [Doncaster, 1990]. The computation of interaction events is motivated by understanding combined movements of separation, attraction, and mutual repulsion that occurs between moving objects. One way to identify interaction involves computing an alignment between the trajectories. Within an alignment, any point of one moving ob-

ject is mapped to either a point or a range of points from the other moving object. We are interested in exploring delayed responses in the movement over time, where one moving object moves in a new direction, an action pattern, and this is followed by an adaption of another moving object, a reaction pattern. To capture such patterns, we need so-called interaction measures, which are similarity measures adapted to cover aspects of interaction. An example of such an *action-reaction pattern* is depicted in Figure 5.1.

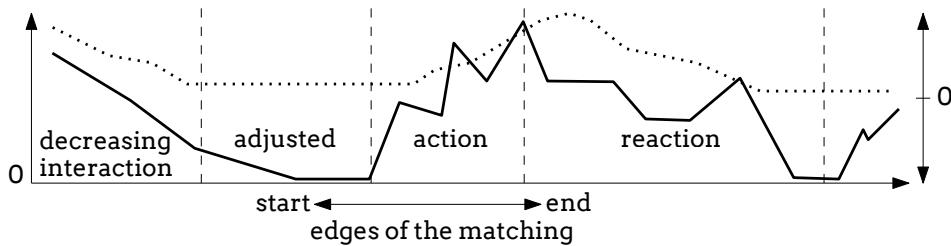


Figure 5.1: A sketch for an action-reaction pattern in an overlayed distance and delay plot. The progression of the delay is depicted as a dotted line, and the distances as a black line. At first, the interaction decreases to an adjusted level. An action then results in a peak of the distances, where one trajectory moves in a new direction. The delay increases rapidly at this point, and the delay reaches its maximum at the beginning of the reaction phase. The reaction of the other trajectory results in a shrinking of the distances again. After having reached its maximum, the delay decreases accordingly. Finally, a new adjusted level is reached until a new action invokes another action-reaction pattern.

Delay is the temporal difference for a pair of points from the trajectories. In Figure 5.1, the progression of delays is visualized as a dotted curve. Both positive and negative delays can occur over time, depending on whether the first point of the pair has a larger time stamp than the second one or vice versa. A positive delay corresponds to the first point being delayed, while a negative delay corresponds to the second point being delayed.

Using the direction of movement and the displacement at corresponding time stamps of the trajectories at equal times, Long and Nelson [2013a] defined the *dynamic interaction measure* for the calculation of strength and degree of interaction between moving objects. However, interaction often includes a delayed response. For ex-

ample, delays are expected in interaction movement patterns for behaviors associated with pursuit and escape, confrontation, and avoidance [Merki and Laube, 2012].

Computational methods for detecting a delayed response often search for movement episodes from two trajectories with similar characteristics, but with a delay for one of the trajectories. Reaction delay is a key parameter in ascertaining both leadership and causality since a moving entity requires time to perceive, process, and respond to its neighbor's motions. Hence, movement patterns are characterized by periods of interactive and non-interactive behavior. Such an episode refers to a period of time, and it might, therefore, be expressed as a sequence of points ranging from one point to all of them. This depends on the level of analysis of the interactive behavior. The scope of the analysis can be varied from a local analysis over episodes to a global analysis. A local analysis reveals the times and locations of dynamic interactions, and allows a finer treatment of the interaction itself [Long and Nelson, 2013a].

A 'follow-behind' pattern is detected in Buchin et al. [2008] by finding episodes where two trajectories move through approximately the same locations, but with a small delay. Nagy et al. [2010] compute a similar pattern by looking at how one trajectory copies the direction of movement of another with a delay. Using a time-ordering procedure to analyze the cross-correlation of velocity and distance, Giuggioli et al. [2015] extract interaction delays and are able to classify copying patterns in both space and direction.

There is a wide range of alignment methods, e.g., *Dynamic Time Warping* [Berndt and Clifford, 1994], *Edit Distance on Real Sequences* [Chen et al., 2005], and the *Fréchet distance* [Alt and Godau, 1995], that aim at simply identifying similar movement. We discuss these methods in more detail in Section 5.4.2.

In movement analysis there is a growing interest in analysis methods that include visualizations. Andrienko et al. [2013] analyze delayed responses in the context of group movement. To identify a group ordering over time, they first precompute a centroid on the collection of trajectories and then rank the interaction of a trajectory and the cen-

troid for each trajectory of the group. One pattern that they detect is the direction-based delay pattern by Nagy et al. [2010]. Its occurrence is visualized in space and time.

Our aim is a visual analytics approach that explores delayed responses in the form of action-reaction patterns. For this purpose, we propose an approach to analyze and visualize delays on two trajectories. We expect that the trajectories have been recorded simultaneously and with the same sampling rate such that the input data captures spatial and temporal relationships at the same time.

Although our focus is on pairs of trajectories, we also show the application of our methodology to a set of three simultaneous trajectories. Computing and visualizing an alignment on k trajectories is cumbersome since the computational time of current state-of-the-art techniques for alignments grows exponentially in the number of trajectories. Thus, it is inherently demanding to develop an interactive visualization for k trajectories that captures interaction events.

To determine whether two trajectories have interactions at all, we develop a new approach to compute a global correlation in sub-quadratic time in the length of the trajectories (see Section 5.3). The global measure captures the overall interaction between the moving objects by computing the correlation via the Fast Fourier Transform and approximates the global delay under an interaction measure.

We have implemented our approach in a prototype visual analytics tool. Our approach shows that quantification of movement patterns complements qualitative knowledge discovery, such as visualizations, so that we support movement analysts who are in need of both. We also introduced a novel similarity measure between trajectories, which incorporates spatial and directional characteristics. Crucial to the approach to analyze delays locally is the computation of a matching between simultaneously recorded trajectories. The matching algorithm optimizes the alignment of the matching with respect to certain features of the trajectories. The temporal alignment is then used, which is induced by a matching, to analyze the delay. This approach scales up to hundred points in the trajectories since it visualizes the trajectories and their interaction patterns as a whole. We summarize

our visual analytics approach in Section 5.4.

In this chapter, we conduct new, extensive experiments on both the FFT-approach and the matching-based approach on three datasets (see Section 5.5). We compare our results to those obtained by the time-ordering approach by Giuggioli et al. [2015], DTW and others. We extend the matching-based approach to three trajectories in order to relate the result on the interaction among the three moving objects with a pair-wise analysis of the triplet.

5.2 Interaction and Similarity Measures

Measuring interaction is closely linked to measuring similarity. In this section, we describe properties of these measures and review interaction measures used in our tool. Trajectory data can be represented as a sequence of points over time. Hence, it consists of directions and locations. Interaction measures can be distinguished into these types. We are interested in several, different facets of interaction. By combining existing measures into a single measure, the combined measure expresses more complex interactions. In this work, we analyze only discrete trajectories.

Definition 5.1. A *discrete trajectory* \mathcal{T} is a sequence of n time-stamped points $\langle(p_1, t_1), (p_2, t_2), \dots, (p_n, t_n)\rangle$, where each $p_i \in \mathbb{R}^d$ and each $t_i \in \mathbb{R}^+$ for $i \in \{1, \dots, n\}$. $\mathcal{T}(t_i)$ selects the corresponding point in the trajectory for a valid time stamp t_i , so $\mathcal{T}(t_i) = p_i$.

Given a pair of trajectories, we define a *delay* as the time difference for the corresponding time stamps of a pair of points. Without loss of generality, we assume that a delay τ has a discrete value between 0 and $n - 1$, thus $\tau \in \{0, \dots, n - 1\}$. The sampling rate of the underlying dataset is then multiplied with τ to obtain an actual delay, e.g., in seconds.

Before elaborating on the interaction measures that we used in our algorithms for computing a matching, we look at common properties that these measures should provide. Without loss of generality, we presume that the measures are given as a distance, i.e., smaller dis-

tances values correspond to higher similarity. A distance can be easily transformed into a similarity measure and vice versa.

Definition 5.2. A *metric* on a set X with $d : X \times X \rightarrow \mathbb{R}$ satisfies the following conditions for all $x, y, z \in X$

$$d(x, y) \geq 0 \quad (5.1)$$

$$d(x, y) = 0 \text{ if and only if } x = y \quad (5.2)$$

$$d(x, y) = d(y, x) \quad (5.3)$$

$$d(x, z) \leq d(x, y) + d(y, z). \quad (5.4)$$

We require the properties of a *premetric (with symmetry)*. It is essentially a metric, for which the distance might be zero for some $x \neq y$, see equation (5.2), and the triangle inequality does not have to hold, see equation (5.4).

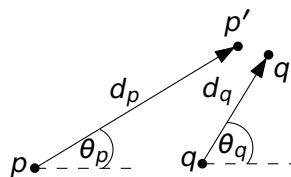


Figure 5.2: A metric space with $p \in \mathcal{T}_1$ and $q \in \mathcal{T}_2$ in \mathbb{R}^2

Given a pair of points $(p, q) \in \mathbb{R}^d \times \mathbb{R}^d$ on trajectories \mathcal{T}_1 and \mathcal{T}_2 , a similarity measure can be computed from spatial properties or the direction of the movement (p, q) . The movement vector pp' is the directed line segment from p to the consecutive point p' with length δ_p . The same holds for q respectively.

The *direction* of p in \mathcal{T}_1 measures an angle θ_p on the movement vector pp' with regard to some other criteria. Very frequently, the *heading* is used, which is the angle θ_p that spans between pp' and an axis, usually the x-axis [Long and Nelson, 2013a], as depicted in Figure 5.2 for two-dimensional trajectories. Other notions for measuring angles are possible as well. The *turning angle* is the angle between pp' and its consecutive movement vector [Kareiva and Shigesada, 1983]. The angle between the line segment pq and the movement vector pp' is the so-called *exposure angle* [Giuggioli et al., 2015]. It can be used to identify leadership between the trajectories because it expresses rel-

ative headings with respect to a line segment of a pair of points from the moving objects.

Direction-based measures calculate a correlation from the two angles θ_p and θ_q from a pair of points (p, q) . A simple directional similarity measure for (p, q) is the cosine of the difference between θ_p and θ_q [Long and Nelson, 2013a]. If both entities are moving in exactly the same direction, its value is 1. When the directions from the movement vectors are pointing into opposite directions, the similarity value is -1 .

A similarity measure for spatial properties uses either the locations of a pair of points or the movement vectors of a pair of points. By using the Euclidean distance, we are able to measure the similarity of point locations.

The *displacement* [Long and Nelson, 2013a] expresses a similarity between δ_p and δ_q , the lengths of movement vectors pp' and qq' . The parameter α controls the behavior of the similarity measure.

$$\text{displacement}(p, q) := 1 - \left(\frac{|\delta_p - \delta_q|}{\delta_p + \delta_q} \right)^\alpha. \quad (5.5)$$

By using a large α , it restricts the displacement to regard large differences as more similar. To consider large differences as dissimilar, α is set to one as a default.

Next, we survey premetrics that combine both directional and spatial properties. The *dynamic interaction measure* proposed by Long and Nelson [2013a] is one of these premetrics; it multiplies the displacement with the cosine on the difference of the headings.

$$s(p, q) := \cos(\theta_p - \theta_q) \cdot \left[1 - \left(\frac{|\delta_p - \delta_q|}{\delta_p + \delta_q} \right)^\alpha \right].$$

If one of the factors becomes zero, then either the impact of the heading or the movement vector is suppressed. This means, therefore, that the other part does not contribute anymore (see Figure 5.3).

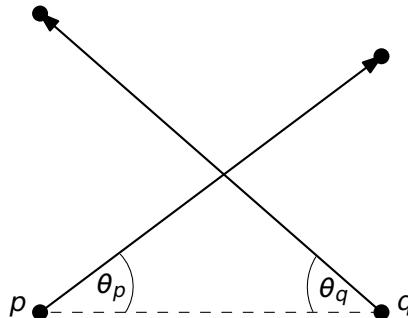


Figure 5.3: Cosine on heading between θ_p and θ_q yields zero for two-dimensional trajectories.

This issue can be resolved by scaling the distance $d(p, q)$ between (p, q) by the similarity of the headings θ_p and θ_q . The distance $d(p, q)$ can be an arbitrary L^p norm on (p, q) . We call this the *directional distance*:

$$d_{\text{dir}}(p, q) := d(p, q) \cdot [2 - \cos(\theta_p - \theta_q)].$$

Thus, the directional distance scales the actual distance on the pair of points (p, q) by the similarity of directions. In contrast to this, the dynamic interaction measure by Long and Nelson [2013a] uses the movement vectors instead of a distance norm. The more the angles of p and q deviate, the more the distance between them stretches.

In order to measure similarity on k moving objects simultaneously, we need to extend our notion of a premetric to k trajectories. Given points (p_1, p_2, \dots, p_k) in $\mathbb{R}^d \times \mathbb{R}^d \times \dots \times \mathbb{R}^d$, we want to have an interaction measure on k trajectories simultaneously: $d(p_1, p_2, \dots, p_k)$.

One way to compose such a measure is to use either the sum, the squared sum, or the maximum of a pairwise interaction measure, e.g., for the directional distance and the sum, the combined distance measure is the following:

$$d(p_1, p_2, \dots, p_k) := \sum_{i=1}^n \sum_{j=i+1}^n d_{\text{dir}}(p_i, p_j)$$

When we choose the max-value of the pairwise Euclidean distance d_2 , the distance measure is the following:

$$d(p_1, p_2, \dots, p_k) := \max_{1 \leq i \leq n; i+1 \leq j \leq n} \{d_2(p_i, p_j)\}$$

An interesting question is how to define an interaction measure that is sensitive to the movement in the trajectories. In Figure 5.4, a zig-zag movement occurs in one trajectory while the other trajectory progresses in the global direction. It is, therefore, desirable to design similarity measures that capture such irregular movements as either a special type of interaction, or as a particular similarity value.

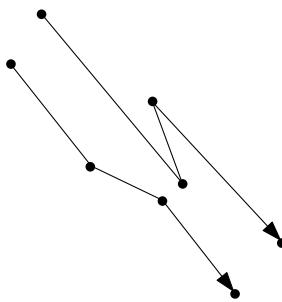


Figure 5.4: Zig-zag movement of one trajectory while the other moves towards the global direction

5.3 Fast Computation of Global Delays

Given a similarity measure d for interaction of time-stamped trajectories, the global delay (τ_{global}) between trajectories \mathcal{T}_1 and \mathcal{T}_2 is the time shift τ that maximizes the similarity between \mathcal{T}_1 and a copy of \mathcal{T}_2 whose points have been delayed by τ time units. Hence, the global delay τ_{global} has $O(n)$ possible time shift values within the range $[0, n - 1]$.

$$\tau_{global}(\mathcal{T}_1, \mathcal{T}_2) = \operatorname{argmax}_{\tau} \frac{1}{n} \sum_{(i, i+\tau) \in [0, n-1]^2} d(\mathcal{T}_1(i), \mathcal{T}_2(i + \tau)). \quad (5.6)$$

For trajectories with many points, it is desirable to compute such global delays efficiently. A simple but time-consuming approach would be to individually compute the similarity between trajectories for all possible time shifts. As an alternative to this $O(n^2)$ time algorithm, it turns out that for the similarity measures defined in Section 5.2, the global delay between two trajectories can be approximated in time sub-quadratic in their length. Specifically by using a *Fast Fourier Transform* (FFT), we can efficiently approximate the similarities for all $O(n)$ candidate values of the global delay in a total of $O(n \log n)$ time. From those candidate values, we extract the global delay in linear time, which is the delay with the maximum similarity of the candidate values.

5.3.1 Correlations and the Fast Fourier Transform

First, we will introduce the basic notions on Fast Fourier Transforms, see for instance [Brigham, 1988], then we apply them to trajectories. The correlation between two complex numbers a and b is defined as the complex number $\text{corr}(a, b) = \bar{a} \cdot b$ where \bar{a} is the complex conjugate of a . More generally, the correlation between sequences $A = [a_0, a_1, \dots, a_{n-1}]$ and $B = [b_0, b_1, \dots, b_{n-1}]$ of complex numbers is defined as:

$$\text{corr}(A, B) = \sum_{i=0}^{n-1} \bar{a}_i \cdot b_i.$$

The cross-correlation $A \star B$ defines a correlation for every time shift $\tau \in [0, 1, \dots, n - 1]$, such that when $\tau = 0$, we have that $(A \star B)_\tau$ is exactly $\text{corr}(A, B)$.

$$(A \star B)_\tau = \sum_{i=0}^{n-1} \bar{a}_i \cdot b_{(i+\tau) \bmod n}.$$

The FFT \mathcal{F} and its inverse \mathcal{F}^{-1} take a sequence of n complex numbers as input and return a sequence of n complex numbers. The cross-correlation theorem states that $A \star B$ can be computed for the n values

of τ using the Fast Fourier Transform as follows:

$$A \star B = \mathcal{F}^{-1} \left(\overline{\mathcal{F}(A)} \cdot \mathcal{F}(B) \right).$$

\bar{X} and $X \cdot Y$ operations on sequences act in an element-wise fashion. Since \mathcal{F} and \mathcal{F}^{-1} both take $O(n \log n)$ time to compute, the n values

$$[(A \star B)_0, \dots, (A \star B)_{n-1}]$$

of the cross-correlation are computable in $O(n \log n)$ time.

To obtain a global delay τ_{global} , see equation (5.6), we now apply these fundamentals to trajectories. Trajectories contain a finite amount of points. The cross-correlation, however, is based on the assumption that the trajectories will repeat indefinitely. We discuss this conversion of the trajectories in the following.

To account for trajectories that do not repeat, we must correlate time stamps that are outside the range (i or $i + \tau \notin [0, n - 1]$) with a value of zero. This is achieved by padding both sequences $c(\mathcal{T}_1)$ and $c(\mathcal{T}_2)$ of complex values with n zeros. The function c resembles a trajectory under a similarity measure as a sequence of complex numbers. For \mathcal{T}_1 , this yields a sequence $C(\mathcal{T}_1) = [c(\mathcal{T}_1(0)), c(\mathcal{T}_1(1)), \dots, c(\mathcal{T}_1(n - 1)), 0, 0, \dots, 0]$ of length $2n$. Computing the cross-correlation between the padded sequences then correctly handles delayed time stamps that are out of range. Correlations for negative delays $-n < \tau < 0$ are now stored at index $2n + \tau$ of $C(\mathcal{T}_1) \star C(\mathcal{T}_2)$, and correlations for delays $0 \leq \tau < n$ are simply stored at index τ . The interaction for a delay is given by the corresponding correlation divided by n .

5.3.2 Approximation of Similarity Measures

We wish to use the FFT to compute the global delay between trajectories under the similarity measures of Section 5.2. To accomplish this, we convert both trajectories into a sequence of complex numbers such that the similarity under time shifts can be derived from their cross-correlation. In particular, for a given interaction measure $d(p, q)$, we are looking for a function c that converts a data point of a

trajectory into a complex number such that $d(p, q) \approx \text{corr}(c(p), c(q)) = \bar{c}(p) \cdot c(q)$ for all pairs of points p and q on the trajectories that we want to compare. It is often convenient to write the function c in polar coordinates, such that the magnitude of $c(p) : \mathbb{C}$ is $f(p) : \mathbb{R}$ and its angle in the complex plane is $g(p) : \mathbb{R}$.

$$c(p) = f(p)e^{g(p)i}.$$

The correlation between $c(p)$ and $c(q)$ is then given by

$$\text{corr}(c(p), c(q)) = f(p)e^{-g(p)i} \cdot f(q)e^{g(q)i} = f(p)f(q)e^{(g(q)-g(p))i}.$$

As an example, consider the direction-based similarity measure $d(\theta_p, \theta_q) = \cos(\theta_p - \theta_q)$. In this case, taking $f(\theta_p) = 1$ and $g(\theta_p) = \theta_p$ gives us $c(\theta_p) = e^{\theta_p i}$, and we obtain $\text{corr}(c(\theta_p), c(\theta_q)) = e^{(\theta_q - \theta_p)i} = \cos(\theta_q - \theta_p) + i\sin(\theta_q - \theta_p)$ from which the exact original similarity measure $\cos(\theta_p - \theta_q)$ can be derived. We show that if an interaction measure d can be derived from the correlation in this way, the global delay between two trajectories can be computed in $O(n \log n)$ time.

For displacement-based similarity measures it is challenging to extract the original similarity measure from such a correlation. Instead, we approximate displacements δ_p, δ_q from the displacement similarity measure, see equation (5.5), by

$$f(\delta_p) = 1, \text{ and } g(\delta_p) = \frac{\pi \log(\delta_p + \beta)}{2 \log(1 + \frac{1}{\beta})}.$$

To approximate the displacements δ_p and δ_q well, β is chosen depending on an α from the displacement similarity measure.

Here δ_q and δ_p are normalized to lie in the range $[0, 1]$. The difference between the original measure $d(\delta_p, \delta_q)$ and our approximation

$$\text{corr}(c(\delta_p), c(\delta_q))$$

is shown in Figure 5.5. One clear difference is how small displacements are correlated. Whereas the original measure barely correlates small displacements, our approximation treats such

displacements as similar, which seems more natural. If this is not intended, the function f can be tuned to suppress this correlation of low displacements. In particular, taking $f(x) = x^\gamma$ allows us to discriminate more between small and large displacements, depending on the parameter γ .

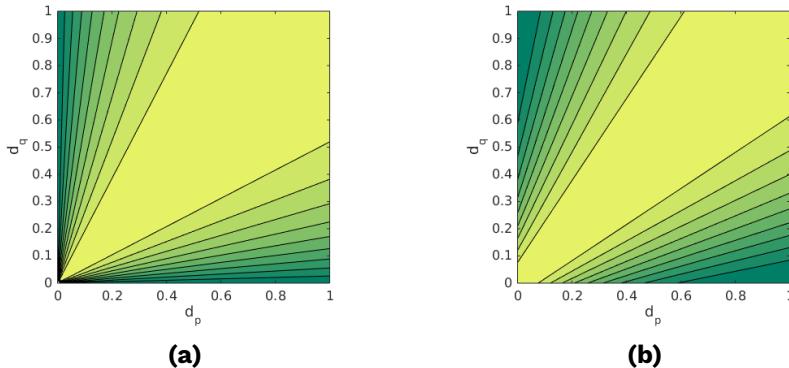


Figure 5.5: Comparison of displacements with our approximation (a) Displacements $\alpha = 2$ [Long and Nelson, 2013a] (b) Approximation using $\beta = 0.15$, $f(d_p) = 1$ and $g(d_p) = \frac{\pi}{2} \frac{\log(d_p + \beta)}{\log(1 + \frac{1}{\beta})}$

5.4 Visual Analytics for Local Analysis of Delays

Varying the temporal scale in analyzing movement data is important to extract movement patterns [Laube and Purves, 2011]. The global analysis of delays, discussed in Section 5.3, enables to quantify whether and how much the trajectories are correlated. To explore interaction events in detail, an approach over time and space on a local level is necessary.

In this section, we summarize our work from Konzack et al. [2015] in which a matching between two trajectories was used to compute local movement patterns on a so-called delay space. To capture them in a visually salient way, we bundle the edges as patches to indicate the source and relevance of the interaction event. Our focus lies on episodes of movement in which two moving entities show similar characteristics, but possibly with a delay.

5.4.1 Requirements for Analyzing Interactions

The main analysis task concerns the interpretation of interaction events and patterns in two trajectories. Since interaction can be defined in different ways depending on the context, our visual analytics tool should allow the analyst to *identify interaction patterns/events in the data (R1)* for various interaction measures.

There can be many events, which may impede visual analysis if they are all shown in detail. The visualization should, therefore, allow an aggregation of the surroundings of a movement pattern to help an analyst focus on the progression of the interaction before and beyond the current event. This leads to the requirement that *critical events should be visually salient (R2)*.

Interaction between moving objects occurs at different scales: globally over the trajectories as a whole, locally at either a specific point in time or over some time interval, or in episodes (a partitioning) of particular patterns [Laube et al., 2007]. An analysis tool should, therefore, *provide means to analyze interaction at different scales (R3)*.

Our computational approach relies on determining a matching between two trajectories in what we call a delay space. For the purpose of the analysis, it is necessary to *understand the spatial structure of the delay space and its relation to the corresponding matching in space and time (R4)*.

5.4.2 Computing Matchings

To identify a potential interaction between two moving entities, we are looking for pairs of data points, one from each trajectory, that are similar according to one of the distance measures discussed in Section 5.2. For this purpose, we first define the *delay space* as a grid of distances for all pairs of points on the two trajectories, as shown in Figure 5.7(b). More formally, given two trajectories of lengths $|\mathcal{T}_1| = m$ and $|\mathcal{T}_2| = n$, the delay space $DL: [1, m] \times [1, n] \rightarrow \mathbb{R}^+$ is a grid of $m \times n$ points $(p_i, q_j) \in \mathcal{T}_1 \times \mathcal{T}_2$.

To identify delayed interactions, we compute a plausible alignment between the two trajectories. Such an alignment should map any point of one of the trajectories to either another point or a contiguous range of points of the other trajectory. We call such an alignment a *matching* between the trajectories. In the delay space, this corresponds to a bi-monotone curve from the lower left corner (starting points of both trajectories) $DL[1, 1]$ to the upper right corner (end points of both trajectories) $DL[m, n]$. Such a matching is shown as a green curve in Figure 5.7(b).

As a basis for our analysis, we pick an alignment of overall high similarity, a matching between two trajectories. There is a wide range of similarity measures for trajectories that are based on finding such an alignment.

Dynamic Time Warping (DTW) aligns two trajectories – or more generally two time series – as to minimize the (squared) sum of distances between matched elements. An DTW alignment can be computed in quadratic time using dynamic programming [Berndt and Clifford, 1994].

The *Edit Distance* (ED) is a widely used measure for similarity between two strings [Wagner and Fischer, 1974]. It has been applied in bioinformatics and language processing. The *Edit Distance on Real Sequences* (EDR) is an adaption of ED for sequences with numerical values, for instance trajectories [Chen et al., 2005]. The EDR tackles the problem of transforming numerical values from ED into integer values by defining a match on a pair of points. A pair of points is matched in EDR when the distance between the pair is less than or equal to an ε threshold. To compute the EDR, a similar dynamic program as for DTW is used. The ε threshold restricts the choices within the dynamic program of EDR. DTW uses distances in the dynamic program; whereas, EDR uses unit costs. The *Longest Common Subsequence* (LCSS) is, essentially, a restricted version of the ED, in which only two of the three operations used in the ED are allowed [Maier, 1978].

Another alignment method minimizes the maximum distance along the trajectories. For curves, it is based on the *Fréchet distance*, and

the corresponding matching can be computed in near-quadratic time [Alt and Godau, 1995]; whereas, the *direction-based Fréchet distance* minimizes the maximum direction difference between the two curves [de Berg and Cook, 2011]. Specifically, we use a so-called *locally-correct Fréchet matching* (LCFM). Such a matching has the following property: if we take any sub-curve of the matching (starting at a pair of time stamps (i_s, j_s) and ending at a pair (i_e, j_e)) and consider the sub-grid of the delay space restricted to the corresponding time stamps ($\text{DL}^*: [i_s, i_e] \times [j_s, j_e] \rightarrow \mathbb{R}^+$), then an LCFM also minimizes the maximum distance restricted to this sub-grid [Buchin et al., 2012]. Similar to this restriction, a profile function determines the structure of such a matching in a *lexicographic Fréchet matching* [Rote, 2014]. In our tool, we use the dynamic programming algorithm from Buchin et al. [2012] to compute a discrete matching based on the Fréchet distance. The main idea is to construct a tree, with $\text{DL}[1, 1]$ as the root, to all valid, subsequent, and monotonous paths towards the root. All vertices on the path from $\text{DL}[m, n]$ to the root are the edges of the matching. This algorithm works with any premetric (see Section 5.2) though Buchin et al. [2012] used it only with Euclidean distances.

To compute a matching on three or more trajectories, we need to extend the notion of a Fréchet matching from two trajectories to a set of trajectories. Dumitrescu and Rote [2004] proposed a definition of the Fréchet distance on a set of curves. In their definition, the Fréchet distance is the longest leash in the set of curves, such that the length of the longest leash is minimized over all tuples of points from the input curves. Dumitrescu and Rote [2004] showed that a 2-approximation of the Fréchet distance on the set of curves can be constructed from all pairwise Fréchet distances.

Our visual analytics tool supports any of the alignment methods above and any similarity measures (see Section 5.2), which allows to use a premetric in the delay space.

5.4.3 Interactive Analysis of Delays in Matchings

The visual analytics approach allows users to explore matchings on two trajectories, the degree of interaction, and delays. The proposed approach was implemented in a prototype visual analytics tool that supports multiple coordinated views of the trajectories, matching, delay space, delay plot, and distance plot for the purpose of visual exploration (requirement R4). To avoid visual clutter in the trajectory plot, we bundle the edges of a matching in colored patches. Doing so makes changes of the delay visible over time (requirements R2 and R3), so that important local movement patterns can be perceived (requirement R1). These local patterns might reflect changes in direction or distance, or show who is ahead and who is behind. A screen shot of the tool is shown in Figure 5.6.

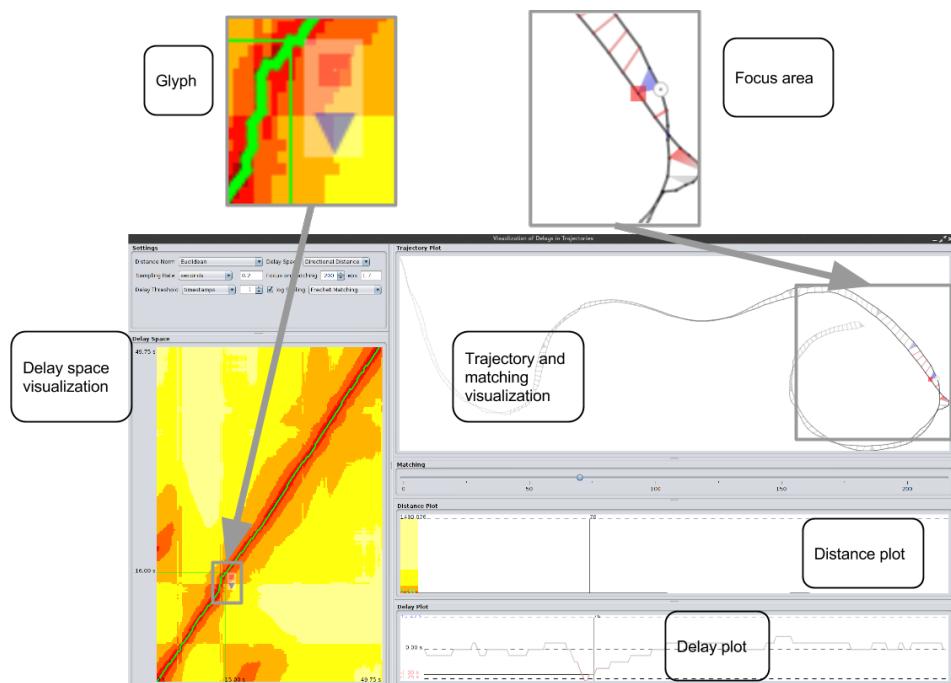


Figure 5.6: Screen shot of the tool, showing the delay space, the trajectory and matching visualization with an enlarged section of it, the distance plot, and the delay plot. Ahead/behind behavior is visualized by a glyph.

The analytical process behind our visual analytics tool consists of the following steps:

1. Open and load a dataset
2. Select a similarity measure and an alignment method
3. Analyze the structure of the matching
4. Refine the parameters for the alignment method and the similarity measure
5. Identify interaction events by browsing the edges

First, we need to open and load the dataset of the moving objects into our visual analytics tool (step (1)). Next, we pick a similarity measure, to construct the delay space, and an alignment method, that will drive the structure of the computed matching (step (2)).

Then, we analyze the characteristics of the matching in the delay space and the trajectory visualization (step (3)) by navigating through the edges of the matching.

A matching enables an interactive, local analysis of delays by sliding through the edges of the matching. The main interaction component is the slider between the trajectory visualization and the distance plot, which browses the edges of the matching (although not the time points), see Figure 5.6.

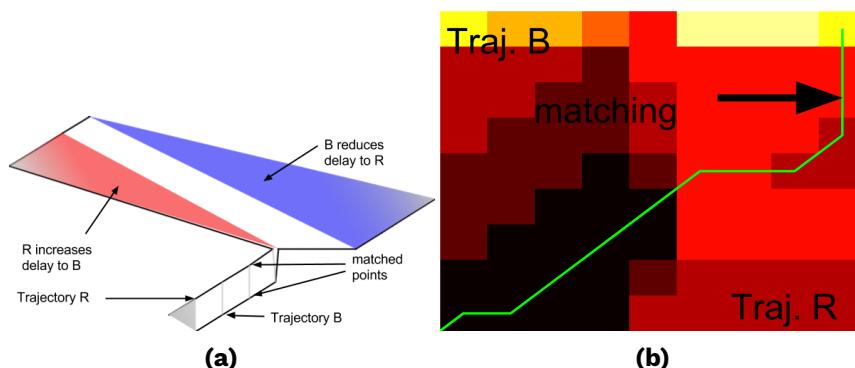


Figure 5.7: Visualizations for a matching based on the directional distance similarity measure. (a) Trajectories and a corresponding matching (b) The delay space and a matching

A delay space is depicted in Figure 5.7(b). Trajectory B is on the y-axis and trajectory R on the x-axis. The values are shown by a linear

heated body color scale, e.g., see [Munzner, 2014], and the matching is visualized as a green path through the delay space. Diagonal line segments correspond to simultaneous movement of both objects, horizontal line segments correspond to movement of the object in R and stationary behavior in B, and vertical line segments correspond to movement in B and stationary behavior in R. In Figure 5.6, a cursor in the delay space points to the currently selected edge of the matching – indices of the pair of points on the axes – accompanied by a glyph to encode that trajectory B, symbolized by the blue triangle, is behind trajectory R – the red square (see Figure 5.6 for an enlargement). This stacking reverses when trajectory B is ahead. If no delay occurs, both symbols appear side-by-side.

A straightforward way to visualize the trajectories and the corresponding matching would be to plot the trajectories in the plane, and to connect matched points between the trajectories by a line segment. This would result in a very dense visualization and visual clutter, however. A monotonous matching has a nice property that has been used to simplify the visualization: in the matching, situations in which one point of one trajectory (the actor) is matched with several points of an other trajectory (the reactor) occur often. This corresponds to an interaction pattern with a delayed response. These edges then can be bundled into a single patch to avoid visual clutter. We color the patches to show source, the actor, and the relevance of the corresponding reaction events (requirement R2).

An example visualization for two short trajectories, B and R, is shown in Figure 5.7(a). Trajectory B is below R, and the reaction events of B and R are associated with the colors blue and red, respectively. Both moving entities start at the bottom of the plane, progress in the same direction, approach each other, split into almost opposite directions, and finally move in the same direction although at a large distance. The matching is based on the directional distance. At the beginning, almost no interaction occurs since both are moving along. The gray patch indicates a delayed response by R, in which one point of B is matched to the first two points of R. As the moving objects approach each other, the structure of the patches becomes different. Within the red patch, the color changes from gray to red indicating that R

increases the delay to B. The blue patch expresses that B reduces the delay to R until the delay vanishes (the color becomes gray).

The trajectory visualization, in Figure 5.6, provides an overview of all interactions between both trajectories in translucent colors. The selected edge and its direct surroundings in the focus area are shown in saturated colors. This helps to untangle the plot of the trajectories since the locations from different time stamps may overlap in space. The delay is visualized by glyphs: a circle for the actor and a red square or blue triangle for the reactor (the same visual encoding as in the delay space visualization). If no delay is observable, both points are depicted as circles.

The delay space visualization can be used to detect specific movement patterns (requirements R1 and R4). An example is a zig-zag movement, as depicted in Figure 5.8 for the upper trajectory, while the other trajectory proceeds in a straight line. The delay space based on the directional distance similarity measure shows a clear color change (a vertical stripe) at the point of this movement. As the trajectory is short, this pattern can also be observed in the trajectory plot. However, for longer trajectories, the trajectory plot does not scale very well, and it becomes more difficult to read; whereas, the matrix-based visualization of the delay space easily scales up to hundreds of points.

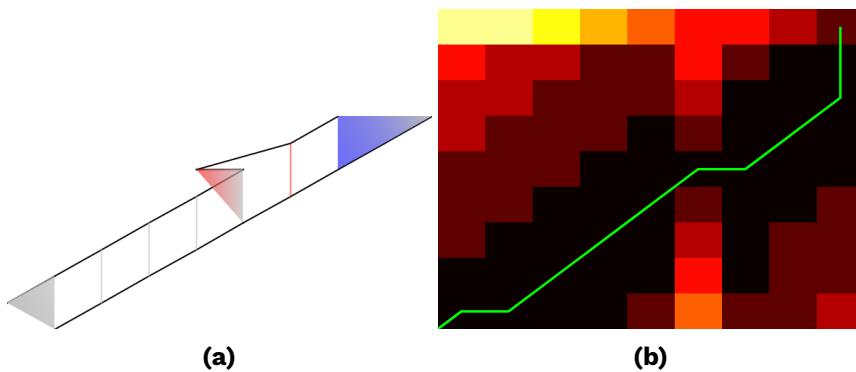


Figure 5.8: (a) Trajectory with a zig-zag movement pattern and a straight trajectory. (b) Delay space based on the directional distance similarity measure. The strong color change corresponds to the zig-zag pattern.

As soon as we perceive the overall structure of the patches within the matching and its delay space as unfruitful, we refine the alignment method and/or the similarity measure (step (4)) so that we obtain a new delay space and a novel matching. For this new setting, we apply steps (2) to (4) until we are satisfied with the overall structure of the matching.

Finally, we identify interaction events by browsing the edges of the matching (step (5)) in more detail. The aim is to find consecutive episodes in both trajectories that correspond to high interaction. By tracing the patches in form of action-reaction patterns in the trajectory visualization and rapid color changes in the delay space visualization, we are able to read off the time-stamps for those events in the delay space visualization.

Additionally, the distance and delay plot, see Figure 5.6, help to detect action-reaction patterns, which globally look as depicted in Figure 5.1, since significant changes in the progression of delays (requirement R3) are related to changes in the interaction between the moving objects. The distance and delay plot show progression of distance and delay over time, respectively. The cursors help to read off exact values at the left side of the plots. The distances are also color-coded on a heated body color scale, and the delays are plotted in the colors of the trajectory.

5.5 Experiments

To evaluate our approach, we applied it on three datasets. First, we analyze the interaction between two Ultimate Frisbee players [Long and Nelson, 2013a]. On this dataset, we compute global delays with our FFT-based approach on subsamples (episodes) to determine how well these episodes correlate. Then, we analyze the covering performance between the attacker and the defender locally with our matching-based visual analytics tool.

The second dataset is a pair of homing pigeons in collective flight [Petit et al., 2013]. We are interested here in extracting interactive move-

ment episodes by applying our visual analytics tool to a 2D projection of the moving pigeons.

On the third dataset, a flock of homing pigeons [Santos et al., 2014b], we selected three trajectories from the flock to analyze the interaction within a matching among these three moving objects. We compare the triplet matching with the pairwise matchings on the three pigeons.

5.5.1 Analysis of the Global Delay

The trajectories in the Ultimate Frisbee dataset from Long and Nelson [2013a] describe the movements of a defender, trajectory B, covering an attacker, trajectory R, who tries to get a pass. The defender wants to intercept or dissuade passes from being completed. The defender's movement is a reaction to the attacker's movements. In this scenario, the delay can be used as a metric to evaluate the defender's performance in covering the attacker.

The trajectories were simultaneously recorded with a sampling rate of 5 Hz, and each trajectory consists of 276 GPS locations. The duration of the recording is 60 s. Some missing locations occurred when the players were stationary. To resolve this issue, we interpolated the trajectories linearly over time for those missing locations. This interpolated dataset with 300 locations per trajectory has been used in our analysis.

Long and Nelson [2013a] propose a global measure for dynamic interaction to determine whether a substantial amount of interaction between the players occurred. This global measure is derived from the values from local dynamic interaction events. We compute a related measure, the global delay by using our FFT-based approach (see Section 5.3).

We analyze the global interaction using displacements because displacements are sensitive to the time shifts that are used for finding the global delay. This also allows us to evaluate the performance of approximations for displacements from Section 5.3.2.

The main goal is to decide whether the trajectories have a substantial amount of interaction or not. In order to distinguish between interactive and non-interactive behavior, we selected subranges of the Frisbee data, such that we can perceive the difference in the global correlation among the trajectories. In Figure 5.9, the displacements of the two players are plotted over time. The selected subranges are indicated in saturated colors. We use the displacement measure with $\alpha = 1$ throughout this section. We first compare the global delay between B1 and R1; then, we compute the global delay between B1 and R2. To overlay the displacements, we shift the time stamps of R2 onto B1 by -30 s on each point of R2.

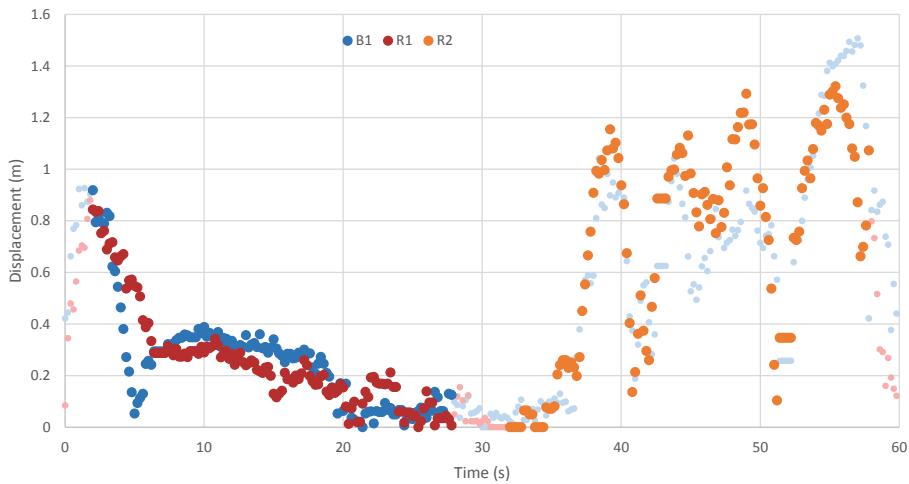


Figure 5.9: Displacements of Ultimate Frisbee data. The selected subranges are indicated in saturated colors. The displacements for the subrange B1 from trajectory B are blue. The values for the subranges R1 and R2 from trajectory R are red and orange.

The set of trajectories B1 and R1, which we analyze for the global delay, are temporally aligned, and they consist of a significant amount of interaction within the first 20 seconds [Long and Nelson, 2013a]. In Figure 5.10(a), the correlation of displacements between B1 and R1 are computed for all possible time shifts and plotted as a black line. The observable highest correlation is at 0 seconds, which constitutes the global delay under the displacement similarity measure. Since either one or the other trajectory has been time-shifted for an align-

ment in the global delay, the plots for the global delay are asymmetrical. We also ran our FFT-based approach to approximate the displacements globally. The results are shown in Figure 5.10(a) for two values of β as dashed lines ($\beta = 0.0$ and $\beta = 0.15$). We observe that these values yield good approximations. However, there are some deviations at the boundaries of the time-shifting. The best approximation of displacements is when our FFT approach has a β between 0.0 and 0.15.

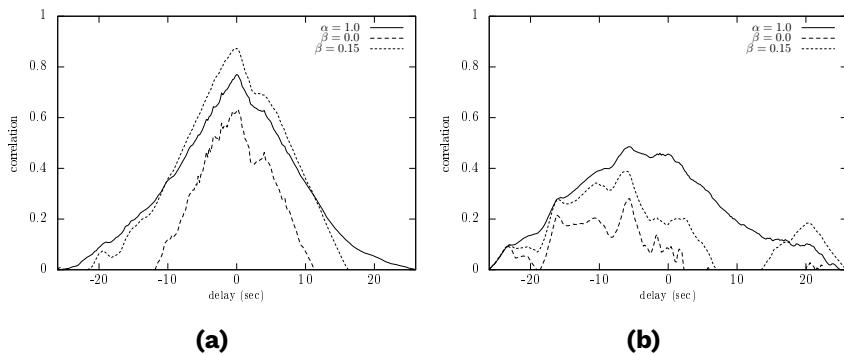


Figure 5.10: Displacements and our FFT-based approximations on subranges from Ultimate Frisbee data (a) Similarity of displacements between trajectories B1 and R1 (b) Similarity of displacements between trajectories B1 and R2

In the second set of trajectories B1 and R2 from Figure 5.9, we compare trajectories that are not temporally aligned. In order to compute a global delay between them, the time stamps of R2 have been shifted by -30 s. The correlation of the displacements between B1 and R2 is plotted in Figure 5.10(b) as a black line. No remarkable correlation is observable, and the maximum delay, the global delay, is not at 0 seconds. Our FFT-approximations of the displacements, with $\beta = 0.0$ and $\beta = 0.15$, yield oscillating values (dashed lines). The correlation's values are below the displacements', and we omitted negative values in the plot.

5.5.2 Analysis of Delays on Ultimate Frisbee Data

In team sports, such as ultimate frisbee, players engage in bursts of movement that can be characterized by different movement patterns. Therefore, a local analysis is more informative than a global analysis for the Ultimate Frisbee dataset [Long and Nelson, 2013a]. Multiple episodes of different levels of interaction occur over time in this dataset. Our aim is to segregate these segments into episodes of high and low interaction.

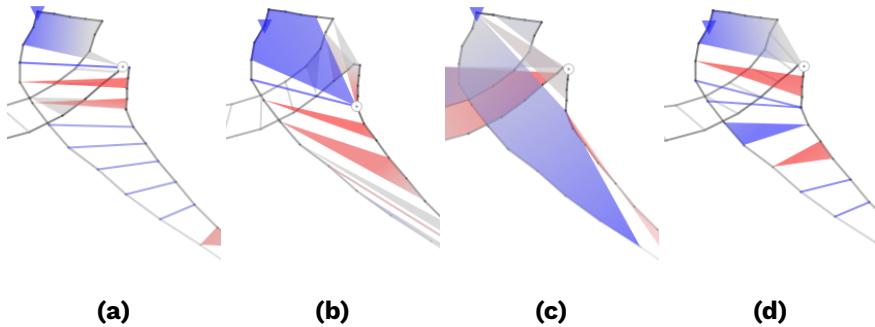


Figure 5.11: Matchings under different similarity measures to detect a loop pattern in the Ultimate Frisbee data: (a) based on the Euclidean distance measure, (b) based on the similarity of headings, (c) based on the dynamic interaction similarity measure, and (d) based on the directional distance similarity measure. (a) highlights it as an interaction event. For (b) in the delay space, only the adjustment of the defender B towards the attacker R is found because it highlights the loop pattern partially as two patches. Using the dynamic interaction measure in the delay space (c) yields three patches, but the adjustments of the defender after the loop are not captured as separate events. (a) and (d) highlights the loop as an interaction event by a blue patch followed by some smaller patches during the adjustments within the reaction of the defender B.

5

To detect interaction events in our matching-based visual analytics tool, we need to choose a similarity measure (see Section 5.2) for the delay space that captures the movement pattern in a visually salient manner (step (2) in Section 5.4.3) as well as an alignment method (see Section 5.4.2). In Figure 5.11(a)-(d), matchings have been computed for the defender B's reaction pattern to the attacker R (step (4)). We employed, as Long and Nelson [2013a], an $\alpha = 1$ in the dynamic in-

teraction measure. The loop and its adjustment between the players afterwards are captured clearly in matchings based on the Euclidean distance and the directional distance. As alignment method, we first use an LCFM.

We opt to analyze the Ultimate Frisbee dataset under the directional distance because this delay space is sensitive to direction-based and spatial properties. During the episodes 0–25 s and 36–45 s, both players show a low distance, demonstrating a high interaction. This is consistent with the findings of Long and Nelson [2013a]. Our measures detect an additional episode of interaction: during the interval 45–57 s, a turn by R is followed by a reaction of B, see Figure 5.11.

The progression of the distance over time, see Figure 5.12(a), follows the structure of an action-reaction-pattern, as discussed in the previous section. In the delay plot, this event can also be seen in the form of a significant change of the local delay, see Figure 5.12(b). Long and Nelson [2013a] analyzed the covering performance by the dynamic interaction measure on pairs of points with the same time stamp; that is why our notion of delay therefore does not occur in their analysis.

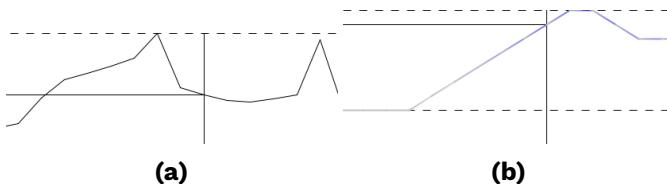


Figure 5.12: Distance and delay plot for matching in Figure 5.11(d) follows the structure of an action-reaction-pattern with delayed response. (a) The distance plot within a reaction. (b) The delay plot shows a significant rise at the point where the defender reacts as a loop movement.

Next, we performed the same analysis using different alignment methods, specifically DTW and EDR. We computed matchings on the two trajectories using DTW and EDR. These algorithms have the same running time of $O(n^2)$ that computing an LCFM has. In the dynamic programs for DTW and EDR, we maintained predecessor graphs to construct a matching as well as to compute the distance value. The original definition of EDR uses an epsilon on the absolute difference

on each dimension of a pair of points. However, we used an ε threshold on the actual distance norm used in the delay space to match a pair of points.

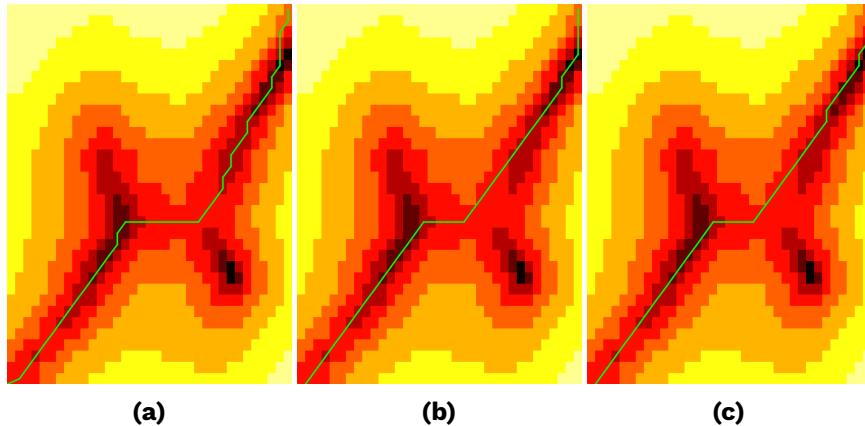


Figure 5.13: Delay spaces for the loop pattern in the Ultimate Frisbee data in a log scaling using the Euclidean distance: (a) shows a Fréchet matching. The matching in (b) is based on Dynamic Time Warping. In (c) a matching has been computed on the Edit Distance for Real Sequences with an $\varepsilon = 1.6$ on the delay space. (a) stays longer in low distances after the horizontal stripe, within the reaction event, than (b) and (c).

In Figure 5.13, we computed matchings in the delay space using the Euclidean distance for the loop pattern of the Ultimate Frisbee data. The horizontal stripe in the matchings of all three methods corresponds to the reaction movement of the defender. All three methods capture the loop. EDR recognizes the loop prematurely with respect to the other methods over time. The progression of the matching curve in DTW and EDR are somehow similar. However, an LCFM consists of more vertical segments in the matching than DTW and EDR. Hence, the LCFM has more bends to move through regions with low distances in the delay space; whereas, the matchings based on DTW and EDR tend to stay shorter in this area of low distances.

The threshold parameter ε to match a pair of points heavily influences the structure of a matching based on EDR. A relatively large or low value forces the matching to prefer diagonal movements in the delay space. A pre-analysis of the distribution of the distances that may occur in a possible matching needs to be conducted in order to spot a

suitable ε value because a relatively large or low value for ε forces the matching to prefer diagonal movements even when it is possible to take vertical or horizontal movements with lower distances. Hence, EDR suffers from having to determine an appropriate ε before computing a matching; whereas, other techniques yield better results without the need of a parameter at all.

5.5.3 Analysis of Delays on Pigeon Data

We analyzed the global delay first to confirm that the trajectories are correlated (results not included) and then applied our matching-based approach on a segment of paired homing pigeon flight trajectories collected as part of a larger pairwise dataset by Pettit et al. [2013]. The time span of the data is 140 s using a sampling rate of 5 Hz. The trajectories have not been tuned or optimized. Pigeon B flies a distance 1.62 km at an average speed of $16.75 \pm 5.02 \text{ ms}^{-1}$ to land 634 m from its starting point, and pigeon R flies 1.33 km at an average speed of $15.36 \pm 4.92 \text{ ms}^{-1}$ to land a distance 572 m from its start point. In this analysis, we used directional distance in combination with LCFM.

An overview visualization is shown in Figure 5.14 for a selected event, in which pigeon R reacts to the movement of pigeon B. The reaction of pigeon R consists of a transition from a right turn to a left turn. This event is visible as a large blue patch in Figure 5.14(b). In the delay space, as shown in Figure 5.14(a), this movement pattern is indicated by the horizontal segment within the matching. Due to the adjustment of pigeon R to the movement of pigeon B as a reaction, there is a decline in the distance, see Figure 5.14(d). The delay continues to increase at this point until the maximum delay is reached within the reaction movement of pigeon R, see Figure 5.14(e). The movements of the pigeons and their matching visualization are outlined in Figure 5.14(c); however, only the surroundings of the selected edge are in the focus of the visualization.

Between 0 – 12 s and 20 – 66 s, the pigeons have low directional distance, indicating a high level of interaction, as can be seen in both the

distance plot and the delay plot (see Figure 5.14). The delay plot indicates that the leadership switches between these two episodes. The directional distance then generally increases with the exceptions between from 95 s to 106 s and starting again at 120 s, when the pigeons have a high level of interaction again. Between 106 and 120 s, one pigeon makes a loop while the other progresses in a straight line.

To evaluate these findings informally with those from another technique, we applied the time-ordering approach by Giuggioli et al. [2015] on the pigeon dataset, see Figure 5.15. For this approach, we have focused on the velocity cross-correlation to identify and classify copying patterns in direction. The optimal movement episodes, based on a directional separation, are similar to the episodes detected by our approach. For computational convenience, the maximum allowed delays were capped at 4.5 s.

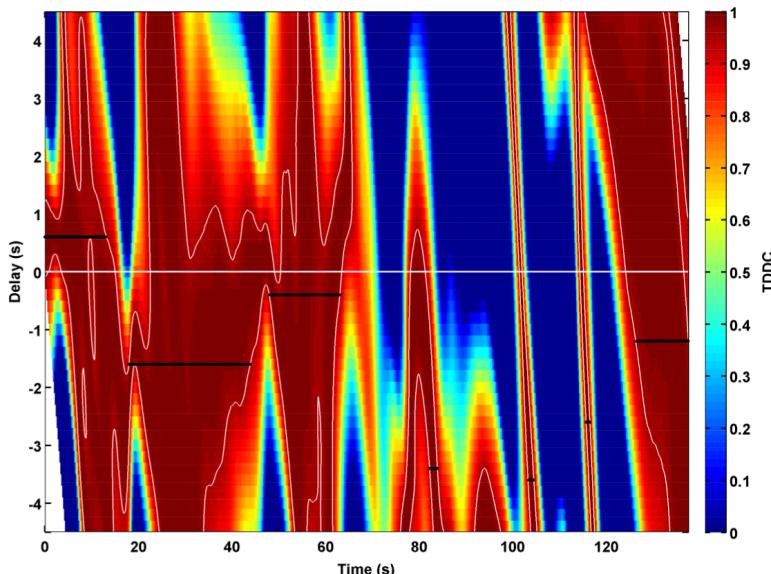


Figure 5.15: Delay plot for the time-ordering approach. The white contour lines are a threshold for motion that is sufficiently aligned to represent interaction. The black segments are the optimal interaction intervals representing the best delayed interaction that can be extracted using this method. Both positive and negative delays are detected within this method as well.

The episodes 0–18 s, 20–65 s, and 120–130 s are consistent with those from our approach. At around 105 s and 118 s, two short movement episodes of interactive behavior have been found. They capture the beginning and the end of the loop pattern, but it is not detected as a whole. The deviations are probably due to the delay parameters of the time-ordering approach since in our matching-based approach quite large delays (max. 9.6 s) occur at around 120 s.

Compared to the time-ordering procedure, our approach exhibits various advantages. Our approach does not require a pre-smoothing of the dataset, and it can be extended beyond pairwise analysis.

5.5.4 Analysis of Delays on a Triplet of Pigeons

Our approach has focused so far on the analysis of interaction between two trajectories. However, it is common in applications to track several moving objects as a collective movement, such as a flock or a group. A procedure to analyze the interaction among more than two trajectories simultaneously is, in fact, very much in need.

There are two reasons why a pairwise procedure on more than two trajectories is likely to fail: one is practical and the other methodological. The practical aspect is that the computational cost increases very rapidly if one accounts for all possible pairs of trajectories. The methodological problem is due to potential inconsistencies in identifying the correct sequence of events. For example, to explain it, let us consider three simultaneously moving objects, A, B, and C. Suppose that a pairwise procedure between A and B and between A and C indicates, respectively, that B reacts to A with a (positive) delay τ_{BA} and that C reacts to A with a (positive) delay $\tau_{CA} > \tau_{BA}$. If τ_{BA} is quite different from τ_{CA} , the pairwise analysis between B and C most likely will extract a delay $\tau_{CB} > 0$, consistent with the fact that individual C has responded to individual A later than individual B. But if the pairwise delays extracted are small, then it is not ensured that $\tau_{CB} > 0$. To avoid these issues, one ought to extract the delay for all individuals at once.

To do so, we extend our approach to compute a matching on three simultaneously moving objects, and we compare how the triplet analysis differs from a pairwise analysis as well as how well a triplet matching captures action-reaction patterns on a local scale.

We use a dataset of a flock of 10 pigeons from Santos et al. [2014b], which has been made available on Movebank [Santos et al., 2014a]. It consists of simultaneously recorded GPS data using a sampling rate of 4 Hz. The pigeons have been tracked for several trips.

All pigeons from the flock are likely to interact with each other since they are all socially familiar to each other [Santos et al., 2014b]. The transitive, pairwise comparison of the pigeons in Santos et al. [2014b] has shown results with significant repeatability, such that we can validate the pairwise matchings with the one obtained directly from the triple.

We have selected three trajectories from pigeons M, S, and U because they represent different roles within the flock. Pigeon M has a high rank for leadership while pigeons S and U have a low negative leadership rank within the flock. Pigeons S and U are thus likely to exhibit follower behavior. Santos et al. [2014b] explain that leadership roles are stable within a flock across different flights.

A pairwise matching between pigeon M and S is shown in Figure 5.6. The matching captures a circular movement wherein the distances between the pigeons vary, and changes in position, heading, and turns occur. At the beginning of a flight, there are many adjustments among the pigeons to the collective movement of the flock, which is why we use a sample of 200 points from Flight 5 from time-stamp 800 to 999; whereas, Flight 5 consists of 3845 time-stamped points in total.

To measure similarity between a triple of points, we used the maximum of the pairwise Euclidean distance (see Section 5.2) in our methods, LCFM and DTW, to compute a triplet matching. For the pairwise analysis, we used the Euclidean distance on the selected pair of trajectories.

We projected the triplet matching into our visual analytics tool by showing only the coordinates of the selected pair and leaving out the

values of the non-selected trajectory. An LCFM on a triplet results in a similar matching as a pairwise computation on the three trajectories separately. This can be seen in Figure 5.16. A pairwise matching is visualized as a blue curve in the delay space and the projected triplet matching in green.

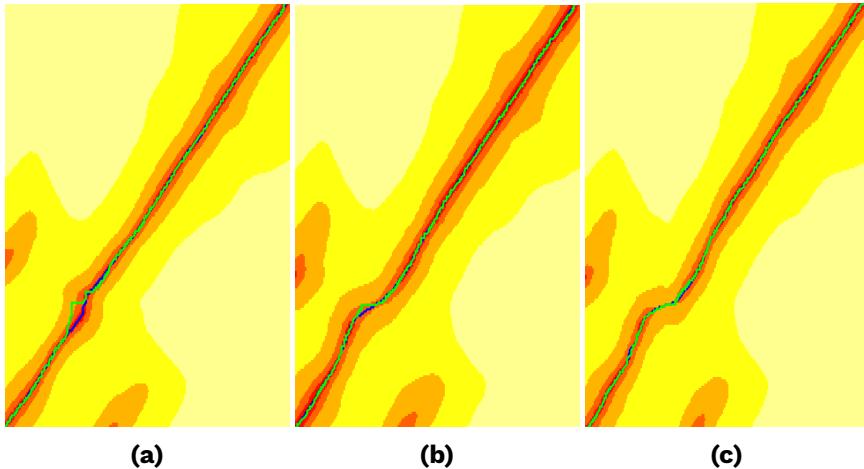


Figure 5.16: Delay spaces, in a logarithmic scale, for projections of a Fréchet matching on three pigeons. The projections are in green, and the pairwise LCFM is in blue beneath it. (a) shows the matchings for the pair (M,U). The matchings in (b) are between pigeons M and S. In (c) the matchings are shown for the pair (S,U). The Fréchet distance is from (c), where the largest distances within the triplet are obtained since the color in the delay space there is the brightest of all projections of the triplet.

The triplet matching on the pair (S,U) coincides with the LCFM on that pair of trajectories. The S-shape of the delay space is due to a sharp turn by S within the circular, anticlockwise movement of the flock. The longest leash is from the pair (S,U) since this projection (see Figure 5.16) of the triple matching moves through a short segment within the S-turn in the brightest color of all three projections. It hereby contains the largest Euclidean distance of the triplet matching. The longest leash gives the pairs (M,S) and (M,U) slack for their edges within the triplet matching. This fact is likely to explain why the pairwise projections deviate by having horizontal stripes in the triplet matching. However, these projections still avoid large values in the delay space.

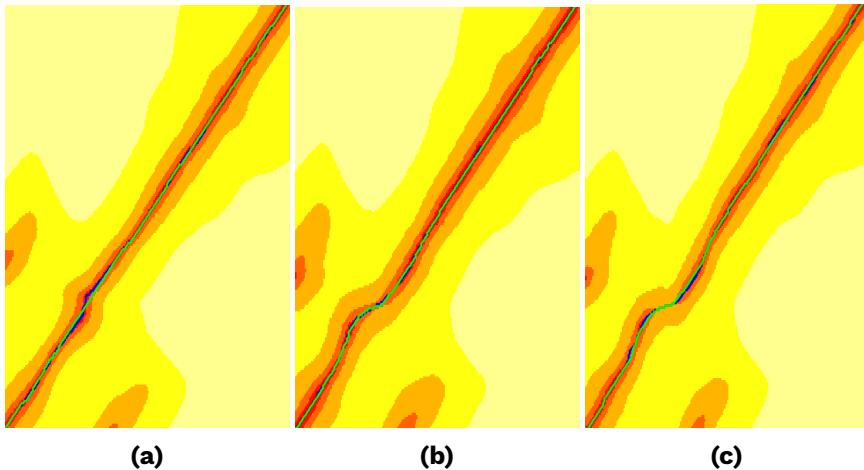


Figure 5.17: Delay spaces, in a logarithmic scale, for projections of a triplet matching on pigeons based on DTW. The projections are in green, and the pairwise LCFM is in blue beneath it. (a) shows the matchings for the pair (M,U). The matchings in (b) are between pigeons M and S. In (c) the matchings are shown for the pair (S,U). The maximum distance value for the triplet matching is obtained from (c) since the distance values within the S-movement of the matching have the brightest color in the delay space, i.e. the largest values.

To evaluate this LCFM on a triple of trajectories, we have computed a triplet matching based on DTW. Figure 5.17 shows the projections in green and beneath it the LCFM on the selected pair in blue. The projections for the pairs (M,U) and (S,U) follow the progression of the pairwise LCFM. The triplet matching on the pair (M,S) clearly deviates from the LCFM since it prefers diagonal movement in the delay space. This can be explained by the fact that DTW optimizes the squared sum of Euclidean distances. As a result, DTW prefers to minimize large distance values. The pairs (M,U) and (S,U) preserve a significantly large distance with respect to (M,S).

Overall, the matchings between triplets provide results similar to those obtained by pairwise matchings. Although, in general, such equality may not be ensured as we explained earlier, for this dataset it holds true independently of the alignment method used (DTW or LCFM).

5.6 Conclusions

We proposed a new approach to analyze interaction events between two trajectories. First, we determined with our FFT-based approach whether any interaction between the trajectories took place. Afterwards, we applied our main technique on the dataset, a versatile visual analytics tool, that enables time delays to be incorporated in interaction movement analysis.

Delayed responses play a key role in detecting interaction events in movement data on a global and local scale. These events are modeled within the so-called delay space wherein we compute a matching between the moving objects. The purpose of our methodology is to gain insight into action-reaction patterns by analyzing those matchings.

Our visual analytics approach uses multiple coordinated views to explore the movement data interactively. The edges of the matching are visualized as colored patches to convey the structure of the interaction events. The relevance of the movement pattern is determined by the delay among the moving entities, which is used as a color saturation for the patches.

The experiments show that the structure of a matching provides important insights into action-reaction patterns in movement data. The computation of a matching relies on the alignment method. We evaluated various types of state-of-the-art methods to compute a matching. All of them are supported in our visual analytics tool. In general, DTW and LCFM gave good results in our experiments. If the movement of trajectories is relatively aligned, then both DTW and LCFM yielded good results. However, LCFM gives better alignments as soon as delayed responses appear within the movement data since DTW tends to prefer diagonal movements in the delay space.

We surveyed various interaction measures in our tool as well that combine movement vectors, distance norm, or headings of the trajectories. By combining the Euclidean distance norm and the difference in the headings, we introduced the novel premetric, the direc-

tional distance. In our experiments, using the directional distance yielded large patches; e.g., in the loop movement of the Ultimate Frisbee dataset. The directional distance captured action-reaction relationships better as visible colored patches in the visual analytics tool than other similarity measures.

The concept of a delay space and a matching can be generalized to more than two trajectories. We showed this on a triplet of trajectories. In these experiments, the results on the triplet were consistent with those from a pairwise analysis. They did not provide additional insights to our datasets, however.

Visualizing the delay space and matching between simultaneously moving objects is challenging since the delay space then has at least three dimensions. Our delay space visualization is limited to support only two trajectories as x and y-axes. Therefore, a novel technique needs to be developed to visualize a matching on more than two trajectories. Beyond the examples presented in our experiments, our methods are widely applicable to the analysis of individual human movement in a crowd, prey-predator interaction, wilding mating behavior, as well as sports analysis.

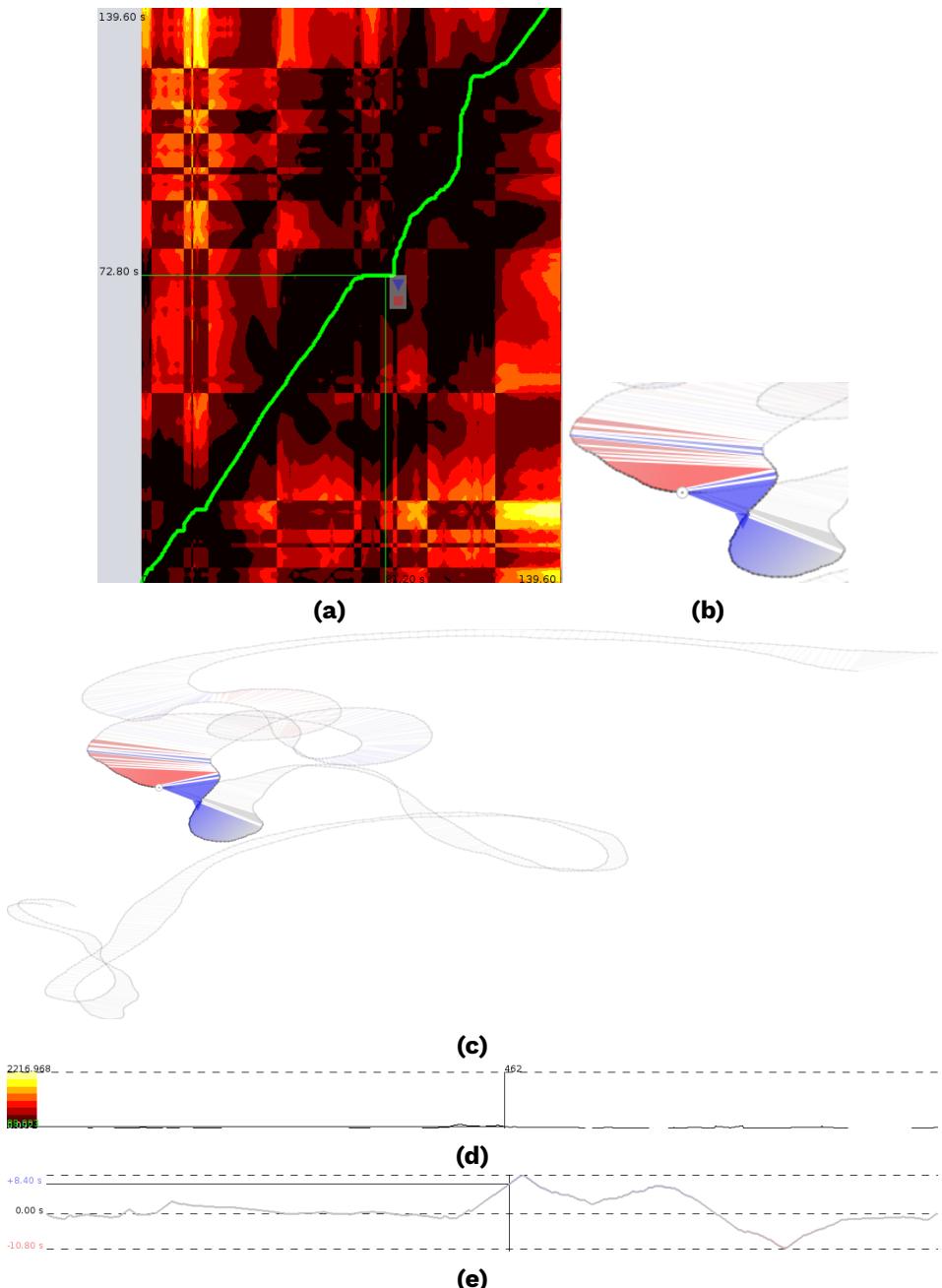


Figure 5.14: Visualization of pigeon trajectories and corresponding delay space based on the directional distance similarity measure (a) The delay space. (b) The focus area. (c) The trajectories and their corresponding matching. (d) The distance plot. (e) The delay plot.

6

Visual Exploration of Migration Patterns in Gull Data

6.1 Introduction

The study of animal movement has long been of interest. Starting in the 1990s, the availability of new technology has led to increasingly detailed and diverse types of data relating to movement [Rutz and Hays, 2009]. GPS-based movement tracks are currently among the most frequently collected types of data. Because of its relative accuracy and high sampling rate, the GPS technology drastically improved the ability to describe and gain new insights about animal movement. The continuing miniaturization allows at the same time to collect data for an increasing range of species and conditions [Kays et al., 2015]. Today, animal movement tracks form one of the main data sources when studying the mechanics of movements, navigational cues and drivers of movement, constantly leading to new insights on animal physiology, behavior, and demography [Dingle, 2014]. Through those advances, ecologists are beginning to ask novel questions about the causes of movement and its consequences for individuals, populations, and ecosystems for which

formal analysis techniques are not always readily available. Kays et al. [2015] claim that interdisciplinary research between data scientists – computer scientists, statisticians – and ecologists will be required to develop new tools efficiently, which will eventually lead to new insights and scientific breakthroughs.

The goal of data visualization is to provide insights into data [van Wijk, 2006]. New visualization techniques and visual encodings help users to understand their datasets. Furthermore, visual analytics can be used to generate knowledge from large and often complex datasets by developing and deploying analytical and visualization techniques [Sacha et al., 2016].

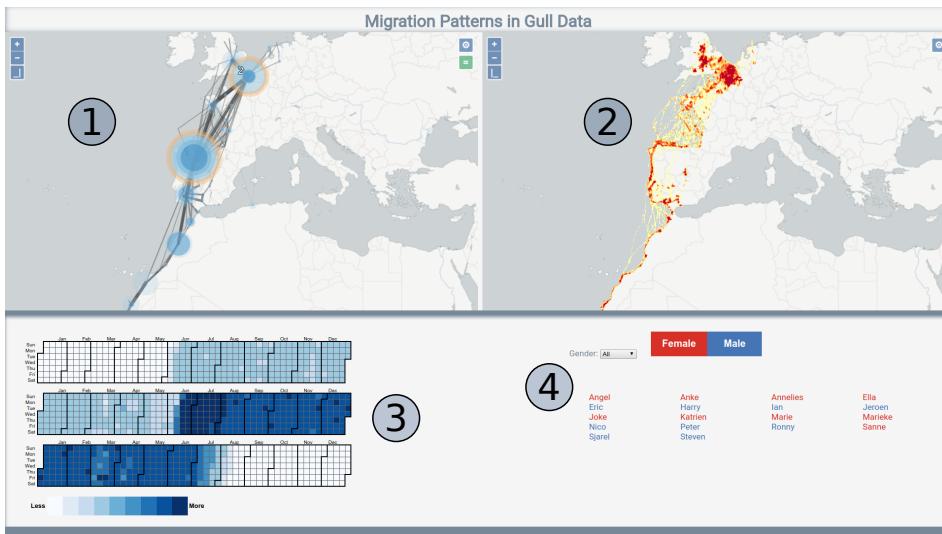


Figure 6.1: Overview of the visual analytics tool: The stopover aggregation visualization (1) enables the user to investigate and select stopovers. The density map (2) shows the spatial distribution of the selected moving entities. Within the calendar view (3) the temporal distribution of the stopovers of the selected entities over time is visualized. The list of gulls (4) shows the names and genders of the selected entities.

To support an ecologist in her search for new knowledge, a visualization expert needs to spend time and effort to understand the relevant questions, data and the general ecological context of a study-system – knowledge which has often not been acquired a priori. Ecologists do, on the other hand, not commonly have an overview of the analysis

and visualization possibilities that are now technically feasible; nor do they know how these could help answer certain ecological research questions. Thus, a knowledge gap between domain experts and visualization designers exists [Slingsby and Dykes, 2012; van Wijk, 2006]. Explorative visualization can help fill this gap as it provides a means for ecologists to discover new trends, to present a dataset visually, to identify pertinent subsets, to compare the movement of individuals, and to locate moving entities, among other tasks. Abstracting such tasks helps to reason about the similarities and differences between them, to distinguish between different goals, and to further guide data abstraction [Munzner, 2014]. Eventually, exploratory visualization provides a novel analytical means that leads to new ecological insights. In addition, it may help users build trust in their generated knowledge base [Sacha et al., 2016].

The current practice of ecologists to investigate and visualize movement is by developing and using Matlab or R libraries [Slingsby and Dykes, 2012; Slingsby and van Loon, 2016]. Those results are either analyzed and visualized statically or plotted on top of satellite maps. Few bird ecologists examined movement data in interactive visualizations with multiple coordinated views, or used a Google Earth-based tool for exploring GPS data from a bird's eye perspective, both of which have been of limited value [Slingsby and Dykes, 2012]. It seems that a tight integration of different spatio-temporal views of the data, with a flexible selection would be beneficial to ecologists to focus on data analysis mechanisms rather than on laborious coding [Spretke et al., 2011].

In this chapter, we present such a technique. It comprises a novel visual analytics approach to help explore animal migration patterns interactively. Our approach provides analytical and visual means to understand different aspects of migration through an aggregation at various spatial scales, with interlinked geographical maps, and views on spatio-temporal events. Migration is ubiquitous in ecology. It is the seasonal displacement of individuals between sites. In our approach, we identify and aggregate stopovers. A stopover is a break within a migratory trajectory. Functionally, stopovers are important for foraging, resting, or socializing with conspecifics, but stopovers can also be

used diagnostically to recognize different migration strategies: along the coast, over the sea, or overland.

Past research on visualizing gull migration lack an aggregation of the trajectories [Slingsby and van Loon, 2016], or impose visual clutter by drawing the results of the clustering as colored data points on a map [Spretke et al., 2011]. Our visual analytics approach remedies this by employing a stopover aggregation visualization, a density map, and a calendar view (see Figure 6.1). The stopover aggregation and density map are plotted on top of interconnected, zoomable geographic maps. The user can select stopovers, and impose constraints on spatio-temporal properties of the selection.

We applied our approach to a dataset of 75 migrating Lesser Black-backed Gulls (*Larus fuscus*), which we will henceforth denote as 'gulls' in this chapter. We evaluated our tool by consulting an expert user [Tory and Moller, 2005] whose expertise on bird migration to assess the strengths and weaknesses of our approach. We identified both ecological research questions and the requirements for the visual design, and mapped them to analytical tasks that the expert user completed. In this chapter we use the terms moving entities and individuals interchangeably.

Our qualitative evaluation confirms that our approach helps ecologists in their analysis of migration patterns so that they are able to visually identify and isolate groups of individuals with a certain migration behavior rather than in non-visual computations. This speeds up and fosters their analytical workflow because our approach empowers ecologists to focus on interpreting the data and on developing new questions without being distracted by coding or by algorithmic technicalities.

6.2 Related Work

To identify different homogeneous movement episodes in trajectory data, trajectories are commonly segmented, i.e. cut into parts, according to characteristics of the movement. Segmentation together with

classification or clustering of movement data helps to summarize and visualize large trajectory datasets. Visualization techniques for movement data support users in identifying new trends within datasets. We survey past research on visualization and clustering of movement data from various application backgrounds to give an overview of recent achievements in these fields.

Segmentation algorithms have been successfully applied to migrating geese [Alewijnse et al., 2014; Buchin et al., 2013c] and gulls [Spretke et al., 2011] among many other studies [Beyer et al., 2013; Guararie et al., 2016; Lavielle, 1999; Le Corre et al., 2014; Madon and Hingrat, 2014; Thiebault and Tremblay, 2013; Zhang et al., 2015]. These approaches segment individual trajectories into pieces of similar movements. Buchin et al. [2016] presented algorithms to summarize segmentations of a larger number of trajectories in a flow diagram, which they then applied to trajectories of football players to analyze spatial formations and plays.

To aggregate movement data, Andrienko and Andrienko [2011] transformed trajectories into aggregate flows between spatial regions in their visual analytics approach. This type of aggregation allows groupings of essential characteristics of the movement. In their approach, the user can interactively control the overall level of abstraction of the visualization. Andrienko and Andrienko [2011] applied their approach to deer and stork data and to trajectories in an urban context, where the aggregated results are visualized as flow maps or as transition matrices.

Density visualization is a powerful visualization technique for analyzing many trajectories. One such example is the work by Willems et al. [2009], which presents a multi-scale density visualization for trajectories. They demonstrate it on vessel trajectories. The visualization of the density fields is derived from the convolution of the dynamic vessel position with a kernel that takes the speed of the movement into account. The density fields are illuminated as height maps on top of a heat map. Additionally, Willems et al. [2009] visualize the individual's contribution of a moving point within the overall density.

Scheepens et al. [2011] use a similar approach to visualize densities of maritime trajectories, for which they apply a cascade of filtering and selection mechanisms on top of density maps. Their selection mechanisms are sensitive to the user’s role, either a domain expert or an operator, and to the task at hand. An operator is usually only concerned with events connected to his work task, such as surveillance of a particular port.

Densities of movement data do not have to be necessarily visualized as a raster map on top of a geographical map. Slingsby et al. [2008] deploy hierarchical, interactive treemaps to explore spatio-temporal movement patterns of couriers in London.

Clustering on urban data has been studied by Lu et al. [2015a,b] on taxi data. Lu et al. [2015a] explore *Origin Destination* (OD) pairs as an interactive selection of clustered regions. The underlying summarization uses a modified DBSCAN algorithm, which utilizes a density computation. Lu et al. [2015b] ranked trajectories of taxi data based on their similarity of travel time. In their visual analytics tool, they visualized the ranking as bands over time. Before ranking the trajectories, they are mapped onto the underlying street network first, and then segmented.

Slingsby and van Loon [2016] present an exploratory visual analysis approach for animal movement data. They applied it, as we did, on gull data, and they devised ecological and visualization requirements on the analytical process. Their visual encodings range from point plots over density maps to tile maps and, thus, cover partially our visualization techniques. The visual analytics software consists of a central view, a satellite map with an overlaid visual encoding and user interactions, and two interconnected timelines: one for a sequence of days and the other one for the times during those days. They did not, however, consider a summary and aggregation of the gulls’ trajectories, which we provide and use as the main technique to interact with the user in our study.

Spretke et al. [2011] developed a visual analytics approach for migrating seagulls. It supports interactive data exploration and enrichment of movement data by adding attributes dynamically from existing

ones, and incorporating weather information, such as wind and temperature. They clustered the trajectories of gulls into three movement states: day migration, night migration, and stopover. They also segmented the trajectories based on spatio-temporal characteristics (45 minutes resting and less than two km continuous flight). Their visualization with interconnected views lacks an abstraction for visualizing clusters since Spretke et al. [2011] mapped the clustering to only colors, and plotted the clusters as plain data points yielding visual clutter.

Kölzsch et al. [2013] reflected on the visual design of migrating birds by exploring different visualization techniques to encode spatio-temporal characteristics of migration. They related ecological research questions to their visual design which inspired us to link our ecological research questions to the requirements of our visual design.

6.3 Problem Definition and Requirement Analysis

Animal migration is an intriguing phenomenon in nature and has as a consequence always received much attention as a research topic in biology. It is increasingly being studied through visual and quantitative techniques due to the availability of tracking data in combination with relevant environmental data layers, see the works by Buchin et al. [2013c]; Klaassen et al. [2011]; Shamoun-Baranes et al. [2011b]; Slingsby and van Loon [2016].

In our study, we focus on the interactive analysis of animal migration tracks. In this section, we lay out a number of important ecological research questions that can be partially answered through these means. By relating those domain research questions to analytical requirements, we aim for a holistic problem analysis [Brehmer and Munzner, 2013] of migratory animal movement.

6.3.1 Ecological Research Questions on Migration

Even though there is a large body of knowledge about bird migration [Berthold, 2001; Newton, 2008] and some common principles are generally recognized (a migrating organism would maximize its fitness behaviorally by minimizing either energy consumption, time expenditure, or the risk incurred during migration) [Alerstam, 2011; Alerstam and Lindström, 1990; Hedenstrom, 1993]; our understanding of underlying drivers as well as the (behavioral, ecological, and physiological) mechanisms is still far from complete.

In the case of the focal species in this study (the Lesser Black-backed Gull), for instance, exact energy budgets are unknown, the comparative advantage of migration (versus overwintering in the breeding area) is unclear, and the reason for the wide spread in overwintering sites by individuals from a single colony are also unknown [Shamoun-Baranes et al., 2017]. On the other hand, some aspects of the migration of this species have been studied, leading, e.g., to the conclusion that it minimizes the energetic costs rather than time spent during migration [Klaassen et al., 2011].

In this study, we explore how an interactive analysis of trajectory data can help us gain more insight into the possible role of individual differences, sex, and time-dependent conditions (such as weather patterns or ephemeral food resources) as well as the characteristics of stopover sites during migration. The ecological questions that we consider are listed in Table 6.1. These questions have been designed to cover insights into a summary of stopovers and the exploration of the relation between trajectories and the actual movement.

Questions E1 to E3 focus on the relation between attributes of the movement tracks (which can be considered as predictor variables) and migration decisions (which can be considered as response variables). However, both the predictor and the response variables have not been operationalized; therefore, inferential testing through, e.g., multiple regression is yet not feasible. Rather, an explorative analysis is required to help to define those variables.

Questions E4 to E6 deal with the uniqueness of stopover sites. For these questions, the reference (e.g., direct surroundings, other individuals) is not clearly defined; hence, part of the challenge here is to discover the type and scale of reference that is meaningful.

Table 6.1: Overview of the ecological research questions that we explore in our visual analytics tool for migration.

Ecological Research Question	
E1	How do individuals' differences, sex, and temporal conditions relate to the route choice?
E2	How do individuals' differences, sex, and temporal conditions relate to the choice of the stopover site?
E3	How do individuals' differences, sex, and temporal conditions relate to the timing of stopping and commencing to migrate?
E4	What is special about the places to where migrating individuals move relative to the direct surroundings (at the same time/within the same time window)?
E5	What is special about the places to where migrating individuals move relative to the place where they come from (at the same time/within same time window)?
E6	What is special about the places to where migrating individuals move relative to other individuals (at the same time/within same time window)?

6.3.2 Requirements for Analysis Tasks

We now map these ecological research questions to more abstract requirements that our approach needs to support. These requirements are listed in Table 6.2 and help in abstracting generic tasks from the spatio-temporal characteristics of migratory trajectory data.

In our approach, we want to *identify spatial patterns (T1)*. This requirement covers a comprehensive visualization of all trajectories, allowing the user to understand and compare spatial patterns in migration (E1) across different scales. A grouping of gulls with similar movements provides insights into different categories of migratory behaviors (E6). A sequence of stops from an origin to a destination (E5) can be expressed as such a grouping. The behavior of individuals should be distinguishable from overall group patterns to investigate how the

migration strategy of an individual deviates from the group movements (E6).

We want to *identify temporal patterns* (*T2*) across several scales, ranging from day/night patterns over a period of days to seasonal patterns. A visualization should allow to specify an episode to constrain the selection to lie within a start and end date (E4, E5, E6).

Another analytical requirement is to *identify stopovers* (*T3*). This requirement deals with a more aggregate view of the data to identify important or often used places (E2) where migrating gulls come together (E6). Stopovers can also be considered at the level of an individual to visualize its migration strategy on top of an exchangeable map, such as a geographic or a topographic map, to investigate the surroundings of a stopover (E4) with different visual cues. A visualization should, furthermore, provide insight into the proximity of the stopover. Statistics on a stopover help us understand the nature of the stopover (E2).

Within our analytical framework, we want to *compare groups and individuals* (*T4*). This requirement concerns grouping individuals that show similar migration strategies (E1), e.g., travel mostly along a coastline, over land, or over sea. A visualization should enable a visual linkage between these groups (E5), but also the comparison of one or more individuals with a group (E6). Subgroups can be selected by the user individually or by characteristics of the gulls, such as gender (E1, E2, E3).

Table 6.2: Summary of the requirements for our visual analytics tool.

Requirements for the Visual Analysis	
T1	Identify spatial patterns.
T2	Identify temporal patterns.
T3	Identify stopovers.
T4	Compare groups and individuals.

6.4 Visual Analytics Approach

Our visual analytics approach (see Figure 6.1) enables users to explore migration patterns interactively. We identify stopovers, and aggregate them in a visualization, so that the user can investigate and interact with stopovers. The tool allows the user to select a sequence of stopovers from an origin to a destination. Such a selection imposes a direction of movement for moving entities within a migration. The selected group of moving entities is rendered in the density map, the calendar view, and the list of gulls. In the density map, the spatial usage of the selected gulls is shown while the calendar displays the counts of stopovers per day of selected or all stopovers. The geographic maps for the density map and the stopover aggregation are interconnected.

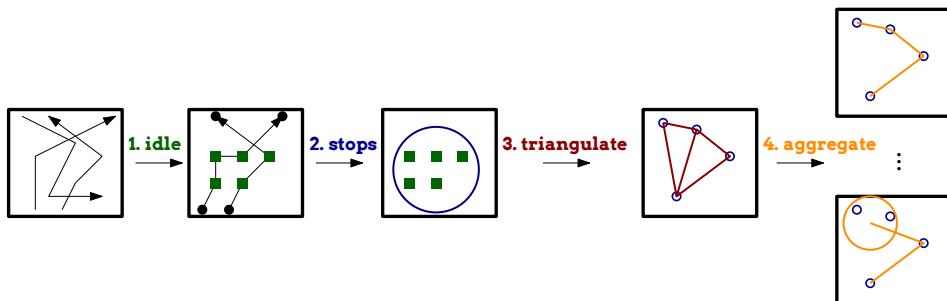


Figure 6.2: The computational process to aggregate stopovers consists of four phases: (1) the identification of idle points; (2) the computation of stops; (3) the triangulation of stops; (4) the aggregation of stopovers on various scales. The first two phases identify stopovers (T3). The resulting triangulation from phase (3) is used in phase (4) to compute summaries on different spatial scales from fine to coarse (T1).

Our visual analytics approach is implemented as an interactive website¹ so that it is widely accessible to researchers who want to explore migration patterns.

By visually exploring a dataset, trust into the knowledge base of the dataset can be built [Sacha et al., 2016], and it allows ecologists to fo-

¹<http://www.win.tue.nl/~kbuchin/proj/gullmigration>

cus on data analysis instead of implementing source code to isolate groups with different migration behavior [Spretke et al., 2011].

In this section, we will first discuss the algorithmic techniques which underlie our visual analytics tool; then, we will relate these to the requirements of our tool. Finally, we will link the requirements to our visualization design.

6.4.1 Computational Methods

To allow a clustering across different spatial scales (T1), we have applied a single-linkage agglomerative clustering to aggregate stopovers. Since gulls have frequent stops along their migration routes which are heterogeneous in duration and local detours, we cannot readily apply stopover criteria used in existing algorithms [Buchin et al., 2013c] for segmenting the trajectories. Our algorithm supports two parameters to facilitate flexible stopover definitions. These parameters are thresholds on the speed of a point to its successor within a trajectory, defaulted to $3.5 \frac{\text{km}}{\text{h}}$, and a maximum distance between two moving entities, defaulted to 500 meters. The distance threshold determines whether two points from distinct trajectories are within the same stopover. Our chosen defaults provide sensible parameters to describe stopover criteria for gulls.

In Figure 6.2, we show the aggregation algorithm that we employed; it has four phases. The first two phases focus on identifying stopovers (T3), while phases (3) and (4) exploit spatio-temporal characteristics of migration (T1, T2). The first three phases are executed sequentially. After that, phase four is executed for different spatial scales. We will discuss each phase in more detail in the following.

First, we classify points as *idle* if the speed of a point with respect to the previous point of the trajectory of an individual is below the given threshold. This allows us to distinguish between movement and non-movement for an individual.

Next, we compute stopovers (T3) by employing Ritter's Bounding Sphere algorithm [Ritter, 1990] onto the idle points and thresholding

the distance between two idle points. A stop is the smallest disk containing idle points (Figure 6.2 shows five idle points that together define a stop). Ritter’s algorithm, with a running time of $O(nd)$ in general for n points in d dimensions, is exceptionally efficient with a linear running time in our case because we compute the sphere in the plane and n points at a stop. A drawback of this algorithm is that the obtained disk is approximately 5% larger than the optimal minimum-radius circle. By identifying these stops, we are able to represent them as a visual abstraction.

In the third phase, we take the centers of all disks, which represent stopovers of idle points, and compute a Delaunay triangulation [de Berg et al., 2008] on those centers. Because a Delaunay triangulation maximizes the minimum angle within the triangulation, i.e. it avoids narrow triangles, it is a suitable means to compute all possible edges between stopovers.

Finally, to aggregate the stops, we perform a single-linkage clustering by applying Kruskal’s algorithm [Kruskal, 1956] on the Delaunay triangulation from a fine to a coarse scale (T1), where the distances between the corresponding stops – disks – serve as edge weights. We span an edge uv between the centers of the smallest enclosing balls at u and v for only the moving entities incident to this edge. We exclude in this computation those individuals who are not traveling along that edge uv . This step allows us to construct a summary of the stopovers that reduces visual clutter across multiple spatial scales.

The density map is dynamically computed on a set of gulls, and allows comparisons between different groups (T4). The computation of a density map consists of three steps in our approach: first, we interpolate the data linearly for each individual, using a sample resolution of 15 minutes to bypass irregular sample intervals. Then, we bin the counts for each individual on a grid of all possible locations. The counts of a cell are the occurrences of all moving entities within that grid cell. Eventually, we discretize the binned values to five quintiles (see Figure 6.3) and compute the contour lines of the density map. Such a density map allows users to perceive the distribution and the spatial extent of the trajectories (T1).



Figure 6.3: Color scale for the quintiles in the density map.

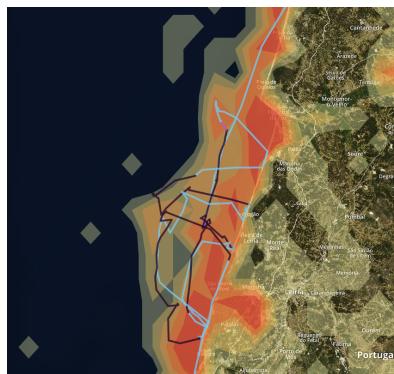


Figure 6.4: Activity at night of gull Sanne. The coloring of the individual gull, Sanne, shows considerable movement at night, in black, during the migration.

As we are interested in investigating day and night patterns of trajectories (T2), we need to classify subtrajectories as day or nighttime. To determine whether a data point of a trajectory, given as longitude, latitude, and a time stamp, occurs during day, night, or twilight, we used the method described in Forsythe et al. [1995]. This method is robust across the latitude, and the computational error ranges from a maximum of one minute near the equator up to two minutes near 60 degrees north latitude.

6

6.4.2 Visualization Techniques

The stopover aggregation (T3) provides an overview of the stopovers and the segmented trajectories, moving among the stopovers. We represent each stopover as a disk, of which the radii encode the quintiles on the number of trajectories at the stopover. This encoding allows us to visualize the spatial distribution of the stops within a stopover. The selected stopovers have an orange halo, and the number indicates the sequential ordering of the stopovers from the origin to the destination (see Figure 6.1). Edges are colored in a light shade of

gray per default without any selection. The more moving entities of the selection travel along an edge, the stronger it will be saturated in a darker shade of gray. Our encoding enables users to have a visually salient selection.

To change the spatial layout of the stopover aggregation from coarse-grained to fine-grained (T1), we allow the user to select an aggregation level (see phase four in Section 6.4.1) by providing a slider, which is only visible if no selection has been made. The map type can be changed to a satellite view (see Figure 6.4) to investigate the vicinity of the stopover (T3), as it is provided by other state-of-the-art map services.

The density map provides an overview of the spatial usage of a set of moving entities or all of them (T1). We ensure that individual trajectories that deviate substantially from others are clearly visible by using an appropriate kernel size. This way, the density map also supports task T4. As for the stopover aggregation, the map type can be changed to a satellite view as well (T3).

To investigate the migration strategy of an individual, we provide a trajectory visualization. The trajectory is drawn on top of the density map (Figure 6.4). Day, night, and twilight are color-coded in light blue, black, and purple (T2), respectively. This color-coding allows a user to see whether a gull travels long or short periods on a particular day and also how many days the entire journey takes. The text labels on top of the trajectory show the stopovers of an individual and can be turned off and on. They indicate the sequence and the direction of the movement for an individual.

The calendar view is shown when a stopover, a set of gulls, or an individual gull has been selected (T4). It provides information on the distribution of stops per day and which gulls stopped on a specific day (T2). Such temporal information can be encoded in various ways, such as a timeline or a punch card chart. A calendar has the advantage of showing multiple years at the same time. Our tool allows users to toggle between showing the distribution at the selected stopovers or at all stopovers. The number of stops are visualized in the same saturated scale of blue as the one used in the stopover aggregation. Se-

lecting a time frame within the calendar helps to exploit temporal patterns (T2), and such a selection is visualized as a contour in the same shade of orange as the one used for the selection in the stopover aggregation (see Figure 6.6).

After evaluating the metadata of the used dataset [Stienen et al., 2016], we focused on visualizing only the gulls' names and gender in addition to their trajectories because the ancillary data did not provide any pertinent statistics (categorical or numerical) beyond gender and name. The gulls within the selection are sorted alphabetically by their names of the gulls. Males' names are colored in blue and females in red. Subselections of previous selections are supported in various ways (T4).

6.5 Exploratory Analysis Process

Our visual analytics tool provides two ways to explore migration: analyzing migration patterns at the level of stopovers, and investigating the spatio-temporal characteristics of a single moving entity. Within the more comprehensive analytical process of exploring stopovers, we also support the inspection of an individual at any time.

The analysis process for exploring stopovers is the following:

1. Spot interesting stopovers
2. Select a promising stopover
3. Investigate the selected stopover(s)
4. Refine the selected stopover(s)
5. Inspect the individual trajectories of the resulting set

Users start by attaining a general overview of the dataset by zooming, panning, and inspecting the interconnected maps. By hovering over interesting stopovers (see Figure 6.5(a)), users gain insight on the structure and the relevance of the stopovers (step 1).

Next, users select a stopover of their choosing by clicking on it (step 2). This immediately updates all other views: the density map, the calendar, and the list of gulls.

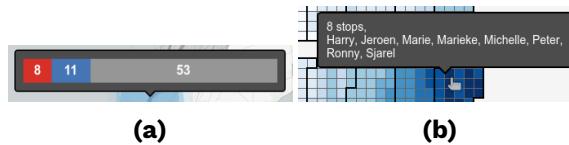


Figure 6.5: Hovering over a stopover (a) shows the amount of gulls at that stopover and their genders as a tooltip. Within the calendar (b), a tooltip shows the number of stops at this day for the selection and the names of the gulls stopping at that day.

Subsequently, users explore the nature of the selection (step 3). In the calendar view, users can toggle between showing the counts of all stopovers or just the selected ones. By hovering a day in the calendar, a tooltip (Figure 6.5(b)) is shown with the names of the gulls stopping at that day. Alternatively, the inspection process on a single gull can help to gain insights here, too (step 5).

The current selection of stopover(s) may be incomplete or inconclusive, so users can refine the selected stopover(s) (step 4). By adding another stopover to the selection, users define a sequence from an origin, the first selected stopover, to a destination, the most recently selected stopover. In the stopover aggregation (2) of Figure 6.1, we show a stopover sequence from Spain/Portugal to the Netherlands/-Belgium. A deselection of stopovers is supported, and restricting the selection to a specific gender is supported too so that users can investigate gender-specific differences (see Figure 6.7). By using the slider for the aggregation level, users are able to adjust the granularity of the stopover aggregation.

Additionally, users can define a time range wherein the sequence of stopovers must lie (see Figure 6.6). This imposes a temporal restriction in which each individual within the selection must have at least one sequence of stopovers matching to the selection sequence, a stopover at the origin after or during the start date of the time frame, and a stopover at the destination before or during the end date of the time frame. On a single stopover selection, the origin and the destination then coincide.

It is also possible to define a subgroup manually from the current selected entities.

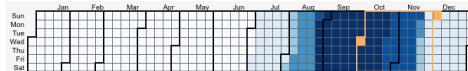


Figure 6.6: A selected time range from October to November within the calendar view at a stopover.

The users apply steps 3 and 4 until they are satisfied with their findings. Eventually they discern the individual trajectories from the result set of moving entities (step 5).

The analytical process to inspect a trajectory of a single moving entity is defined by selecting the individual first, and then investigating the movements of the trajectory on top of the density map. The visual encoding can be altered to exploit geographical characteristics and stopovers along the trajectory.

6.6 Evaluation

To assess the effectiveness of our visual analytics tool in terms of strengths and weaknesses, we applied it to a dataset of migrating gulls over a period of three years and evaluated the visual analytics tool by interviewing an expert user in a two-hour session.

Our domain expert (one of the co-authors) has a background in ecology who has studied bird migration in the past. He had seen a previous prototype of our approach, but he had not experimented with the visual analytics tool nor had he studied the dataset we used beforehand.

We conducted the evaluation with the expert user in three phases. First, we gave instructions and explanations on the visual design, the user interaction, the computation of stopovers, and the analytical features. This phase took 25 minutes. After that, we dealt with specific analytical questions on the dataset. We developed a catalog of analytical tasks, see Table 6.3, that covers different facets of the tool and links ecological research questions with the requirements for our approach, see Table 6.1 and 6.2 in Section 6.3. In the final phase, we asked

about applying our tool on an area of interest and then we asked questions about the usability of the tool. Those reflections will serve as a discussion of our approach.

6.6.1 Dataset of Migrating Gulls

The lesser black-backed gull is an interesting and challenging species to analyze since it has a broad diet and can feed on many resources (both terrestrial and marine), can fly efficiently in many different weather conditions, and can rest on both land and sea. Consequently, migration can take place across almost any landscape, and foraging is possible almost anywhere along its migration route. This species generally adopts a fly-forage migration strategy which avoids carrying loads and instead switches frequently between flying and feeding [Klaassen et al., 2011].

We applied our visual analytics tool to a comprehensive dataset collected by Stienen et al. [2016] to investigate our ecological research questions about the lesser-black backed gull. Stienen et al. [2016] collected almost 2.5 million data points from 101 gulls. This dataset is unique with respect to other gull datasets since such a large number of gulls have not been tracked over three years at such a fine scale resolution thus far. All of the gulls in the dataset have been tracked for at least ten days, and more than half of the gulls had locations for more than 100 days. This dataset contains 75 lesser black-backed gulls as well as 26 herring gulls. Since the herring gulls stayed at their breeding site, we have focused on the lesser black-backed gulls in our study. Their breeding sites were at the Belgian and Dutch coast, and during autumn the lesser black-backed gulls migrated to southern Spain, Portugal, and northern Africa. Further details about the dataset and the ecological studies that it supports and are provided in Stienen et al. [2016].

6.6.2 Expert User Evaluation

All analytical tasks (see Table 6.3) were completed successfully. The ecologist completed most of the tasks within two to five minutes. However, task A5 took nine minutes in total since this task relied heavily on several inspection processes and the expert user's having to contemplate on the meaning of "interesting movements during nighttime".

We discuss the analytical tasks sequentially from top to bottom and elaborate on the differences in solving similar tasks as well.

To accomplish A1, the expert user first inspected different aggregation levels (T1) and zoomed in at different levels within the stopover aggregation visualization. Next, he investigated the whole map by hovering over several stopovers (E4, T3). Eventually, he identified the stopover in northern Spain and Portugal as a stopover with many gulls.

Tasks A3 and A7 are similar tasks to A1, but the expert user did not use the aggregation slider. For A3, the user browsed over a couple stopovers (E4), and spotted three stopovers in Brittany (northern France) where only female gulls stayed (E2, T4). The expert user instantly recalled the stopovers from A3 in A7, and started to hover over several stopovers (E2, E4) until he found suitable stopovers in Brittany, England, and other parts of France, where more females than males stopped. He implicitly assumed there might be stops where there are more males than females (compare Figure 6.7).

To solve task A2, the user immediately selected the singleton stopover in England and noted that there are multiple stopovers in northern Africa (E4). He next used the slider for the aggregation level after that until there was only one stopover in northern Africa left and selected this one as a destination (E5). Subsequently, the domain expert inspected the individual trajectories from the selection using this route (T1, T4).

Task A4 differs from task A2 by adding a temporal constraint (E3, T2) on an origin-destination selection (E5). The domain expert wanted to

Table 6.3: Analytical tasks that the expert user completed. Those have been linked to ecological research questions and the requirements of our approach.

Analytical Task	Completion Time	Ecological search Questions	Re- Requirement
A1 Find a stopover with a lot of gulls.	2 min	E4	T1, T3
A2 Which gulls are migrating from England to northern Africa?	2 min	E4, E5	T1, T4
A3 Find a stopover with only female gulls.	2 min	E2, E4	T4
A4 Which gulls are migrating from Brittany (northern France) to England during the period beginning June 2013 until end of June 2014?	3.5 min	E3, E5	T1, T2, T3
A5 Find a gull with interesting movements during nighttime.	9 min	E1, E5	T1, T2, T3, T4
A6 Which gulls are migrating along the coastline in southern France?	4 min	E6	T1, T3, T4
A7 Find a stopover with more female than male gulls.	1 min	E2, E4	T4
A8 Which gulls are migrating from the breeding spot via northern Spain to England?	2 min	E4, E5	T1, T4
A9 What has been the migration strategy of gulls visiting Madrid (Spain)?	5 min	E1, E2, E5	T2, T4
A10 How would you describe the migration route of gulls at the stopover in Gibraltar?	2 min	E1, E2, E5	T2, T4
A11 Describe the migration pattern over France.	5 min	E2, E3, E4	T1, T2, T3

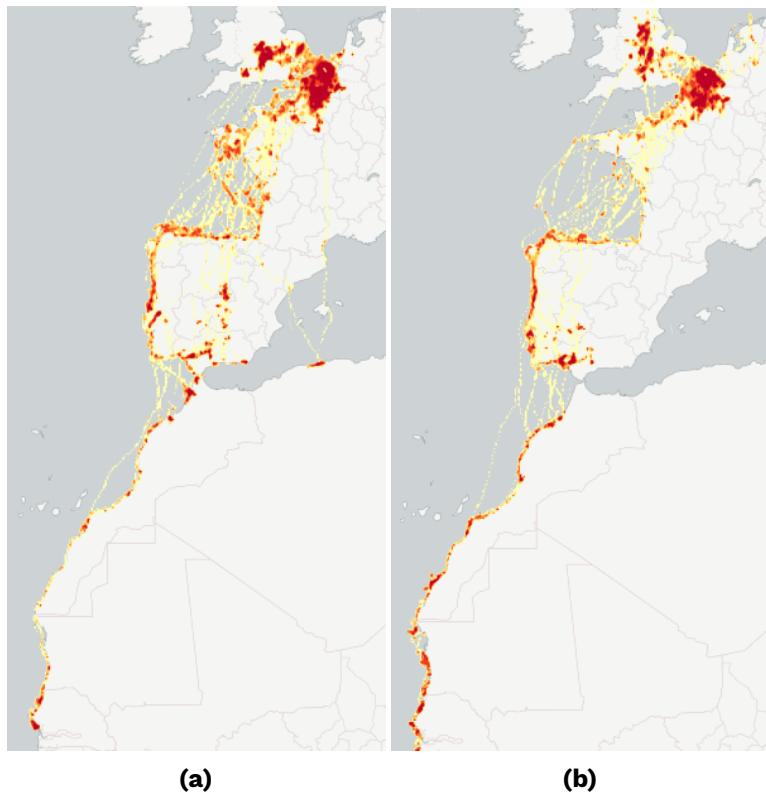


Figure 6.7: Density maps for females (a) and males (b) reveal that females stop more often in Brittany, the most northwestern part of France, than males and that males tend to take larger detours during migration along the sea.

6

use the aggregation slider similarly as in A2 for Brittany, but he did not find a unique way because, if the clustering is too coarse, there is no stopover in England, and, if it is too fine, there are many stopovers in Brittany. The expert user accepted this trade-off and selected a sequence from England to the largest stopover in Brittany (T1). Next, he restricted the time frame within the calendar appropriately (T2), enumerated the gulls Harry and Sanne, and expressed that he wanted to select a region, Brittany as such, because he thought that he otherwise might miss some birds from other stopovers in Brittany.

In task A8, he again shared the same desire to be able to select by region after having selected yet another origin-destination pair, over an intermediate stopover in this case. Thus, a weakness of our ap-

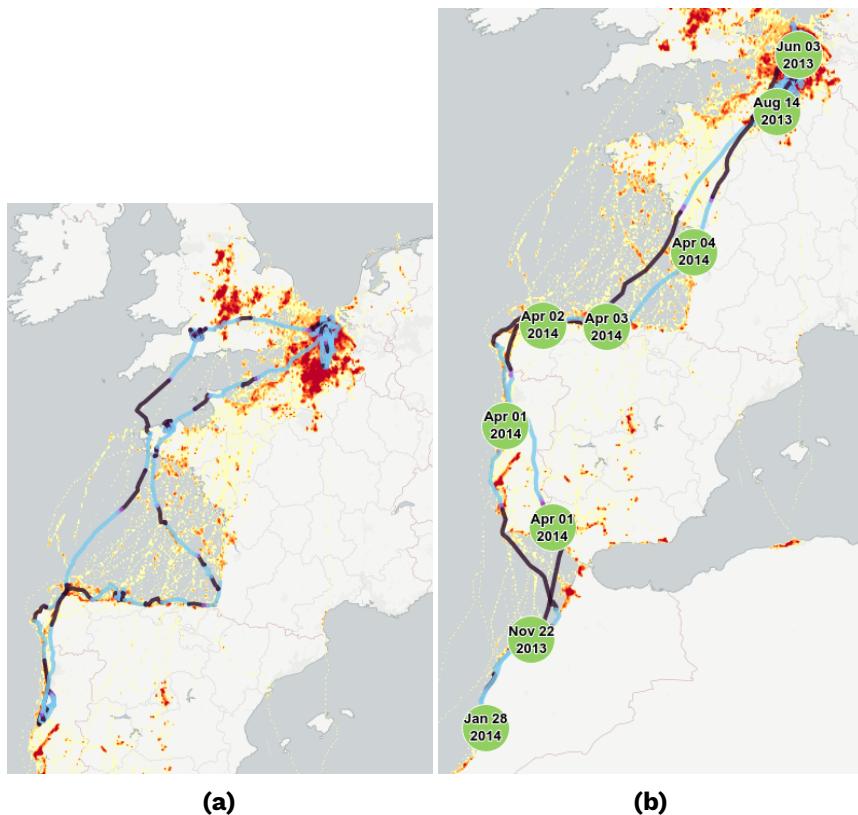


Figure 6.8: Overview of the movements of gulls (a) Angel and (b) Eric. Both show considerable movement at night in France, Spain, and Portugal.

6

proach is that our visual analytics tool does not support a selection by region.

In task A5, finding a gull with interesting movements during nighttime (E1), the user first investigated the complete list of gulls by hovering over them (T4). He was surprised that he sometimes could not see long trajectories. After discovering that half of the gulls stayed at the breeding spot, he wanted to select all of the gulls from all stopovers excluding the breeding spot (E5, T4) to investigate their night movements outside of the breeding spot since he presumed that those motions at the breeding spot are probably due to human interference. He moved further through the list of gulls, noting that selecting a stop-over (T3) might be more effective, but then he found the gull Angel

who had a suitable long trajectory (E1) (see Figure 6.8). Then, he defined “interesting” in this context as distinctively different patterns in travel duration, speed, or trajectory shape between a series of consecutive days and nights (E5).

He assumed implicitly that gulls could float on the sea and be moved by tidal forces, as known from the literature [Shamoun-Baranes et al., 2011b; Slingsby and van Loon, 2016]. Therefore, these patterns were not considered as interesting *per se*; only when occurring at a different rate during day and night.

We classified the locations of a trajectory based on the time stamps and the corresponding geographical position as twilight, daytime, or nighttime (see Section 6.4.1), and visualized this on the trajectory. The visualization tool, therefore, does not provide direct information about the actual time beyond the day and night time periods for a given date. In order to compare trajectory lengths between day and night straightforwardly, the domain expert sought for periods around the equinox since daytime and nighttime are then almost equally distributed (based on this the ecologist did, e.g., skip a stopover of Angel in November since then the nighttime is longer). He, subsequently, switched to another individual, Eric, and zoomed out to get an overview of Eric’s overall movements (E1). After some panning and zooming, he found it interesting that Eric, as many others, traveled as much during the day as during the night. He hypothesized that there might always be light at certain landmarks available to help the gulls navigate.

While Angel’s and Eric’s trajectories (see Figure 6.8) do not show much difference between the lengths of day/night stretches, there are enormous differences in the lengths of the tracks between Angel and Eric. This poses the question whether Angel was coping with adverse wind-conditions (during the segment with shorter distances per day/night) while Eric possibly had strong wind support.

In task A6, the user investigated the coastline of the Atlantic ocean, and selected the stopover covering this area in southern France. After wondering whether he missed some gulls within the selection, he inspected the list of individual gulls (E6, T4). The domain expert dis-

tinguished between gulls traveling up north at the coastline, Anke, Sjarel, and Hilbran, southwards, Roxanne and Jasmin (partially), in both directions, Lea, or not all, Joke and Ian. To investigate the migration of the gull Marie, a partial coastline migration, he needed to zoom in and out further to obtain a higher resolution for the text labels of the stopovers. He wondered whether he had covered all of them, and he hovered and selected other stopovers south of the previous selection (T3). Eventually, he noticed that those gulls are a subset of the previously analyzed gulls.

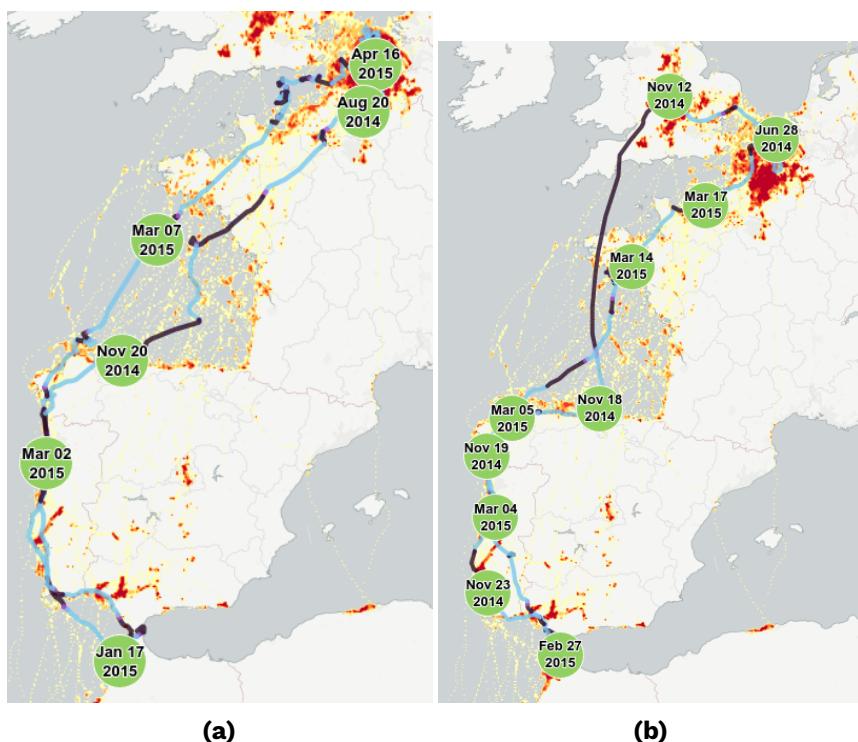


Figure 6.9: Trajectory visualization for (a) gull Annelies, who is migrating in a clockwise two-way migration, and (b) gull Ella, who visits England before migrating south (anticlockwise).

Tasks A9 and A10 dealt with migration strategies at specific locations (E1), Madrid, and Gibraltar. To obtain a more detailed aggregation, the domain expert first used the slider in task A9. Then, he noted movements within the density map of Madrid, and selected the stopover at Madrid. He enumerated Joke, Lea, and Michelle, and noticed

that only females visited the stopover (E2, T4). The expert user then defined a migration strategy for a group or an individual by identifying the most southern stopover during their/its migration (E5). By inspecting the individual trajectories, he traced that Madrid was the most southern stopover for Joke during the migration. Lea, in contrast, visited a stopover south of Madrid, and traveled a week later than Joke had. The ecologist noticed instantly that Michelle, who had been recorded for two years and pursued different migration strategies. At the stopover in Gibraltar (A10), the expert user switched often between Annelies' and Ella's trajectories, the two gulls who stopped in Gibraltar, since he was interested whether both were visiting England (E5). He summarized that Annelies migrates in a clockwise two-way migration (E1), and Ella who migrates a year later than Annelies, visits England before migrating south (in an anticlockwise two-way migration) (see Figure 6.9). The ecologist used date information within the green stopover of an individual trajectory (T2) to accomplish this. He remarked that both pursued a coastline migration, and he assumed that gulls learn to shortcut during migration through experiences based on learned landmarks and resources from previous visits at the coastline. In ecology, it is commonly believed that bird species gain a better navigation capacity over the years through past learning experiences. Therefore, older birds are assumed to have better navigation abilities.

The final analytical task A11 of describing the migration pattern over France was driven by the user's interest in whether a stopover is used during a spring or autumn migration. He reflected first on the meaning of a migration pattern and defined it as how many stops the gulls take (E2) and at what times they stop (E3, T3). To accomplish this task, he selected the largest stopover in Brittany and analyzed the temporal distribution of the selection (T2) in the calendar view by toggling between showing the distribution at all stopovers or at this selected one. After investigating some other stopovers (E4), the domain expert concluded that the majority of stopovers in France is used during spring migration, including the large stopover in Brittany, and seldom during autumn migration. He found this fact interesting because he had previously assumed that the gulls would use the same stopovers in

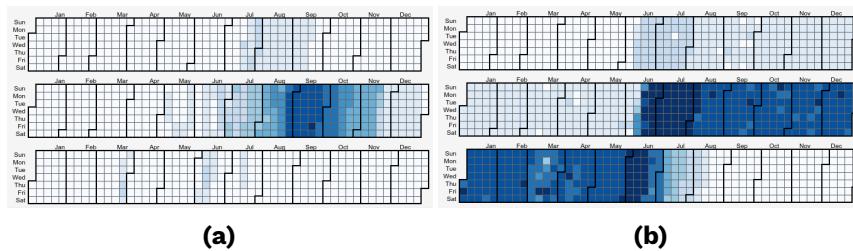


Figure 6.10: Stopovers in the calendar view for all gulls visiting England (a) at the stopover in England and (b) at all stopovers.

both directions.

6.6.3 Reflections

After the task-oriented questions, we asked the expert user to apply the visual analytics tool to an area in which he is interested. He chose the stopover in England because he wanted to know whether the visits to England were before, during, or after the breeding season (E2). He zoomed in on the area around England and analyzed the movements within the density map (E1). By using the aggregation slider, he obtained the finest resolution of the clustering. Next, by toggling between the temporal distribution at all stopovers and at the selected stopover, it became clear that the gulls visited England after breeding (E2), as shown in Figure 6.10. The ecologist was surprised that the gulls had visited England mainly in 2014. He hypothesized that this fact might be weather-related and/or dependent on wind conditions (E5). Using a temporal restriction from mid-July to September, he found that the gulls Harry, Jules, and Sanne were regular visitors to England. He was surprised that these gulls had visited England that early (E3). The ecologist was also interested in investigating the differences among the entities (E6) during two time range selections. He used the tooltips in the calendar view heavily to perform this comparison.

Afterwards, we asked the ecologist to elaborate upon his reasoning behind the selections. Regarding origin-destination selections, he would not select more than three stopovers at the same time. He

would like to be able to select by region only at a very aggregate level. In particular, a gridded map to select areas of interest, such as cities or agricultural regions, would be beneficial to the expert user. This indicates that a hierarchical clustering, as we employed it, is crucial to aggregating stopovers dynamically on different spatial scales.

The time restriction within the calendar view has been used by the expert user to isolate yearly cycles, and this feature enabled him to identify pre-, post-, and peribreeding visits in this way.

Then, we asked him to outline his workflow after finding something of interest in our tool, and to contemplate on the purpose of our approach. The expert user saw our approach as “a visual data querying tool” to generate subsets of the dataset. He would select individuals showing a certain behavior, and look at their space-time usage after finding something of interest. Subsequently, he would continue his research by computing some metrics, correlations, and statistics on the selected individuals in R to test a hypothesis on these groups. This confirms that our approach helps ecologists to visually and interactively explore and identify migration patterns before they proceed with non-visual analytical tasks.

By exploiting spatio-temporal relations between stopover sites, showing characteristics of a particular stopover, and visualizing spatio-temporal properties of an individual’s trajectory in our tool, we enhance such analytical tasks for ecologists from manually extraction through custom prototyping – which would consume several hours – to a visual user interaction that takes a couple of minutes. Furthermore, ecologists can identify individuals who share a certain migration strategy, and they also infer migration routes of individuals [Shamoun-Baranes et al., 2017] at a selected stopover site. Hence, our approach enables ecologists to visualize movement datasets with many movement tracks as well as individuals in the trajectory aggregation and thus speeds up the inspection process of discovering interesting stopover sites, individuals, or tracks drastically.

6.7 Conclusions

This study presents a novel approach to visually explore migratory trajectory data. We computed an aggregation of stopovers from those trajectories along with the movements between them in different spatial scales, and visualized it interactively together with a density map and a calendar view. To investigate the tracks interactively, we enable users to select stopovers, and to add restrictions on spatio-temporal properties of the selection. By applying our approach to a dataset of 75 migrating Lesser Black-backed Gulls and by evaluating our approach with an expert user, we validated our visual analytics tool.

Our findings show that this exploratory visual analytics tool supports ecologists to investigate research questions on migration interactively. Our tool especially enhances the identification of (groups of) individuals exhibiting similar spatio-temporal migratory behavior, and it, additionally, facilitates the discovery of stopover sites with environmentally conditions which stand out.

Since we used a single criterion for all stopover sites and individuals in our clustering, we think it is worthwhile to investigate varying rules per individual and per region. This would enhance the flexibility in the stopover aggregation.

As part of future work, we plan to integrate environmental data, such as information on land use, weather conditions (primarily wind), sea currents and daylight, to facilitate the spatial exploration in the context of relevant variables.

7

Conclusion

The miniaturization of tracking devices, an increase in their accuracy, and cheaper production costs of those sensors have led to a rise in collections of movement data. The development of new analytical tooling has not kept pace with these ongoing trends of tracking movement.

Understanding the drivers behind movement is the ultimate goal for movement analysts. Many researchers from various fields contributed to computational movement analysis through interdisciplinary collaborations to analyze movement by automated means [Demšar et al., 2015].

The contributions of this thesis are in line with those interdisciplinary advances. We first elaborated on the methodological gap between algorithms and visualization in the analysis of movement. Then, we explored how the interplay between geometric algorithms and visual analytics can be enhanced in the analysis. By combining algorithms and visualizations into integrated approaches, as in our visual analytics tools, we showed that algorithms and visualization complement each other in the analysis of movement.

7.1 Contributions

We revisit this thesis's contributions in this section. We state for each chapter open problems and future work.

In Chapter 2, we gave an overview of existing approaches in computational movement analysis as well as in the visualization of trajectory data. Our typology helps researchers to become aware of technologies and methods outside of their specialization to create new innovative tools and by that closing the semantic gap between trajectory data and concepts on movement that an analysts wants to investigate [Laube, 2015].

Our second contribution, described in Chapter 3, concerns theoretical results on the computational complexity of trajectory analysis. Assuming the Strong Exponential Time Hypothesis, we proved two lower bounds: that a simplification cannot be computed in subquadratic time for polygonal curves with n points that lie in $\Omega(\log n)$ dimensions; and that we cannot compute the Fréchet distance for k curves, each with n points, in $O(n^{k-\varepsilon})$ time for any $\varepsilon > 0$. In our survey of computational problems on movement data, we stated known lower and upper bounds. It is worth to note that there are no lower bounds known to us for dynamic time warping and the Edit distance with k curves. Furthermore, it would be interesting to investigate whether a simplification can be computed in subquadratic time for fixed dimensions, such as \mathbb{R}^2 , which has both practical and theoretical ramifications.

Then, we studied a novel formulation of the simplification problem in Chapter 4: progressive simplification, which is a series of simplifications that are consistent across different scales. A progressive simplification makes it possible to zoom in and zoom out of an interactive map without unnecessary flicker. We proposed the first algorithm to compute a minimum-complexity simplification progressively which runs in $O(n^3 m)$ time for m scales and an input curve with n points. For continuous scales, our algorithm runs in $O(n^5)$ time. Our second contribution in Chapter 4 is a new representation of shortcut graphs that

applies to any simplification algorithm using shortcuts. We proposed an algorithm to compute the error for all shortcuts in $O(n^2 \log n)$ time, which is an improvement over $O(n^3)$ time, and we devised a compressed shortcut graph that, under reasonable assumptions, allows finding a shortest path in $O(n \log n)$ time. It would be interesting to apply these representations in a non-progressive setting and further evaluate them experimentally. Moreover, it is beneficial to investigate a lower bound to this type of simplification and to explore whether a more efficient algorithm can be devised.

In Chapter 5, we presented a visual analytics tool for exploring interaction events between two (or three) trajectories. By computing an alignment, we identify delayed responses which we used to visualize action-reaction patterns. In our delay space visualization, we were limited to two trajectories because we use each trajectory on one axis. Thus, we need novel visualization techniques that work for multiple trajectories. Even visualizing a matching among multiple trajectories is a challenging task because an 'edge' is spanned among multiple individuals. Furthermore, it remains a challenging task to compute such alignments among k individuals in $O(n^{k-\varepsilon})$ time for any ε since we expect that most of the commonly used alignment methods have such a lower bound. Developing new approximation schemes for alignments among k individuals would be of interest from an algorithmic perspective and fruitful for applications that analyze interaction in large groups of trajectories. Visualizing the resulting alignments among many individuals continues to be a challenging problem.

Our final contribution of this thesis (see Chapter 6) is an exploratory visual analytics approach to investigate migration patterns of animal tracking data interactively. We have seen that segmenting, clustering, and aggregating trajectory data help eliminating visual clutter and that such aggregations can serve as a beneficial means for analyzing migration routes interactively. Our qualitative evaluation, by applying our approach to a dataset of 75 migratory lesser black-backed gulls and by consulting an expert user, suggests that our visual analytics tool empowers ecologists to be able to focus on visual data exploration instead of laborious non-visual prototyping. As we stated

in Chapter 6, it is beneficial to integrate environmental data so that users can explore the proximity of migration routes. Moreover, it would be interesting to aggregate stopovers with more flexibility to tailor our aggregation to specific spatio-temporal properties and to visualize such stopovers for instances as regions. To date, clustering algorithms for movement data use mostly spatio-temporal properties to aggregate subtrajectories. This presents an opportunity to investigate how integrating environmental data into the clustering computation can improve the clustering results.

7.2 Looking Forward

We now reflect on how our contributions may influence the analysis of movement data as a whole and what themes are beyond the scope of this thesis.

In the context of this thesis, we focused on trajectory data in the plane. Researchers increasingly collect data from multiple sensors, in addition to location-based trajectory data, to integrate context and to describe the underlying cues of the movement more precisely. The computational approaches of this thesis can be extended and applied to multidimensional trajectories by adapting the corresponding similarity measures. Visualizing such multidimensional datasets is challenging because it considerably widens the design space for presenting and exploiting the relationships between multidimensional sensor data. For instance, linking and brushing of two-dimensional views that exploit the causal links between attributes of sensor data might be of help. Another possibility would be to investigate how three-dimensional visualizations can be used in a meaningful way since there are trade-offs in visualizing data in three dimensions versus two dimensions [Munzner, 2014]. Designing and exploring such visualizations, however, lay beyond the scope of this thesis.

For movement analysis, we have seen that “the whole is greater than the sum of its parts”. Enhancing the interplay between algorithms and visualization allows analysts to interpret their datasets as

integrated approaches. Visualizations are essential to understand how an algorithm works on datasets. Given a dataset, an algorithm is necessary to transform and to visualize data. A knowledge gap and an interest gap usually exist between a visualization researcher and another (computer) scientist according to van Wijk [2006]. Typically, knowing only the basics of the other field(s) is not enough to advance or contribute to the other area. Thus, to evolve computational methods for movement analysis holistically, we need to foster intradisciplinary advances, as in this thesis. Furthermore, intra- and interdisciplinary advances are essential in closing the semantic gap between the low-level tracking data and the high-level concepts that analysts 'speak' and understand [Laube, 2015]. This thesis fosters such intradisciplinary advances, and the contributions we made here demonstrate the benefits of doing so.

We have seen in this thesis that disciplines benefit from the exchange of concepts and ideas between them. Enriching knowledge among disciplines allows researchers to investigate a wider range of research questions and to ask more diverse research questions on the causal links between an individual, its movement path, its drivers, and its proximity. Therefore, it is interesting to ask how common knowledge of movement analysis can be defined and taught among multiple disciplines. For instance, in recent years, it has become popular to teach computational and programming basics to non-computer scientists as "software carpentry". Another example is the concept of literacy; for instance, visualization literacy assesses the abilities of a person to read and understand visualizations [Boy et al., 2014]. These are examples of teaching methodologies to scientists in other fields, which may help to foster and educate common knowledge among multiple disciplines. But this exchange of knowledge needs to also go from applications to computer science. This thesis is a stepping stone to move computer science and its applications in the analysis of movement closer to each other.

References

- Abboud, A., Backurs, A., and Williams, V. V. (2015). Tight hardness results for LCS and other sequence similarity measures. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 59–78.
- Agarwal, P. K., Avraham, R. B., Kaplan, H., and Sharir, M. (2014). Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449.
- Agarwal, P. K., Har-Peled, S., Mustafa, N. H., and Wang, Y. (2005). Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3–4):203–219.
- Agarwal, P. K. and Varadarajan, K. R. (2000). Efficient algorithms for approximating polygonal chains. *Discrete and Computational Geometry*, 23(2):273–291.
- Al-Naymat, G., Chawla, S., and Taheri, J. (2009). SparseDTW: A novel approach to speed up dynamic time warping. In *Proceedings of the 8th Australasian Data Mining Conference (AusDM 2009)*, volume 101, pages 117–127. Australian Computer Society, Inc.
- Alerstam, T. (2011). Optimal bird migration revisited. *Journal of Ornithology*, 152(1):5–23.
- Alerstam, T. and Lindström, Å. (1990). Optimal bird migration: the relative importance of time, energy, and safety. In *Bird Migration*, pages 331–351. Springer-Verlag, Berlin, Germany.
- Aleijnse, S., Buchin, K., Buchin, M., Kölzsch, A., Kruckenberg, H., and Westenberg, M. A. (2014). A framework for trajectory segmentation by stable criteria. In *Proceedings of the 22nd ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM GIS 2014)*, pages 351–360. ACM.

- Alewijnse, S. P. A., Buchin, K., Buchin, M., Sijben, S., and Westenberg, M. A. (2017). Model-based segmentation and classification of trajectories. *Algorithmica*.
- Alt, H. and Godau, M. (1995). Computing the Fréchet distance between two polygonal curves. *Computational Geometry: Theory and Applications*, 5(1–2):78–99.
- Andersson, M., Gudmundsson, J., Laube, P., and Wolle, T. (2008). Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528.
- Andrienko, G., Andrienko, N., Bak, P., Keim, D., Kisilevich, S., and Wrobel, S. (2011). A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages & Computing*, 22(3):213–232.
- Andrienko, G., Andrienko, N., Fuchs, G., and Garcia, J. M. C. (2018). Clustering trajectories by relevant parts for air traffic analysis. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):34–44.
- Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D., and Giannotti, F. (2009). Interactive visual clustering of large collections of trajectories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 3–10. IEEE.
- Andrienko, G., Andrienko, N., and Wrobel, S. (2007). Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46.
- Andrienko, N. and Andrienko, G. (2011). Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):205–219.
- Andrienko, N. and Andrienko, G. (2013). Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization*, 12(1):3–24.
- Andrienko, N., Andrienko, G., Barrett, L., Dostie, M., and Henzi, P. (2013). Space transformation for understanding group move-

- ment. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2169–2178.
- Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). OPTICS: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, volume 28, pages 49–60.
- Aronov, B., Driemel, A., van Kreveld, M., Löffler, M., and Staals, F. (2016). Segmentation of trajectories on nonmonotone criteria. *ACM Transactions on Algorithms*, 12(2):26.
- Bach, B., Dragicevic, P., Archambault, D., Hurter, C., and Carpendale, S. (2014). A review of temporal data visualizations based on space-time cube operations. In *Proceedings of the Eurographics Conference on Visualization*.
- Barequet, G., Chen, D. Z., Daescu, O., Goodrich, M. T., and Snoeyink, J. (2002). Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167.
- Ben-Or, M. (1983). Lower bounds for algebraic computation trees. In *Proceedings of the 15th ACM Symposium on Theory of Computing (STOC 1983)*, pages 80–86. ACM.
- Benkert, M., Djordjevic, B., Gudmundsson, J., and Wolle, T. (2010). Finding popular places. *International Journal of Computational Geometry and Applications*, 20(01):19–42.
- de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany.
- de Berg, M. and Cook, A. F. (2011). Go with the flow: The direction-based Fréchet distance of polygonal curves. In *Theory and Practice of Algorithms in (Computer) Systems*, pages 81–91. Springer-Verlag, Berlin, Germany.
- de Berg, M. and Mehrabi, A. D. (2016). Straight-path queries in trajectory data. *Journal of Discrete Algorithms*, 36:27–38.
- Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the Workshop on*

- Knowledge Discovery in Databases (KDD 1994)*, volume 10, pages 359–370.
- Berthold, P. (2001). *Bird Migration: A General Survey*. Ornithology series. Oxford University Press, Oxford, United Kingdom.
- Beyer, H. L., Morales, J. M., Murray, D., and Fortin, M.-J. (2013). The effectiveness of bayesian state-space models for estimating behavioural states from movement paths. *Methods in Ecology and Evolution*, 4(5):433–441.
- Bishop, C. M. (2007). *Pattern recognition and machine learning*. Springer-Verlag, Berlin, Germany.
- Boy, J., Rensink, R. A., Bertini, E., and Fekete, J.-D. (2014). A principled way of assessing visualization literacy. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1963–1972.
- Brakatsoulas, S., Pfoser, D., Salas, R., and Wenk, C. (2005). On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, pages 853–864. VLDB Endowment.
- Brehmer, M. and Munzner, T. (2013). A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385.
- Brigham, E. O. (1988). *The Fast Fourier Transform and its applications*. Prentice Hall, Upper Saddle River (NJ), United States of America.
- Bringmann, K. (2014). Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS 2014)*, pages 661–670.
- Bringmann, K. and Künemann, M. (2015). Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 79–97.
- Bringmann, K. and Mulzer, W. (2016). Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76.

- Buchin, K., Buchin, M., and Gudmundsson, J. (2008). Detecting single file movement. In *Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM GIS 2008)*, pages 288–297. ACM.
- Buchin, K., Buchin, M., and Gudmundsson, J. (2010). Constrained free space diagrams: a tool for trajectory analysis. *International Journal of Geographic Information Science*, 24(7):1101–1125.
- Buchin, K., Buchin, M., Gudmundsson, J., Horton, M., and Sijben, S. (2016). Compact flow diagrams for state sequences. In *Proceedings of the 15th International Symposium on Experimental Algorithms (SEA 2016)*, pages 89–104. Springer-Verlag, Berlin, Germany.
- Buchin, K., Buchin, M., Gudmundsson, J., Löffler, M., and Luo, J. (2011). Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry and Applications*, 21(3):253–282.
- Buchin, K., Buchin, M., Knauer, C., Rote, G., and Wenk, C. (2007). How difficult is it to walk the dog. In *Proceedings of the 23rd European Workshop on Computational Geometry (EuroCG 2007)*, pages 170–173.
- Buchin, K., Buchin, M., van Kreveld, M., Löffler, M., Silveira, R. I., and Wenk, C. (2013a). Median trajectories. *Algorithmica*, 66(3):595–614.
- Buchin, K., Buchin, M., van Kreveld, M., Speckmann, B., and Staals, F. (2013b). Trajectory grouping structure. In *Proceedings of the 13th Symposium on Algorithms and Data Structures (WADS 2013)*, pages 219–230. Springer-Verlag, Berlin, Germany.
- Buchin, K., Buchin, M., Meulemans, W., and Mulzer, W. (2014). Four Soviets walk the dog - with an application to Alt’s conjecture. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 1399–1413.
- Buchin, K., Buchin, M., Meulemans, W., and Speckmann, B. (2012). Locally correct Fréchet matchings. In *Proceedings of the 20th*

- European Symposium on Algorithms (ESA 2012)*, pages 229–240. Springer-Verlag, Berlin, Germany.
- Buchin, K., Sijben, S., van Loon, E. E., Sapir, N., Mercier, S., Arseneau, T. J. M., and Willems, E. P. (2015). Deriving movement properties and the effect of the environment from the brownian bridge movement model in monkeys and birds. *Movement Ecology*, 3(1):18.
- Buchin, M., Kruckenberg, H., and Kölzsch, A. (2013c). Segmenting trajectories by movement states. In *Advances in Spatial Data Handling: Geospatial Dynamics, Geosimulation and Exploratory Visualization*, pages 15–25. Springer-Verlag, Berlin, Germany.
- Cao, H., Mamoulis, N., and Cheung, D. W. (2007). Discovery of periodic patterns in spatiotemporal sequences. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):453–467.
- Cao, H., Wolfson, O., and Trajcevski, G. (2006). Spatio-temporal data reduction with deterministic error bounds. *The VLDB Journal – The International Journal on Very Large Data Bases*, 15(3):211–228.
- Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann, Burlington (MA), United States of America.
- Chan, W. S. and Chin, F. (1996). Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, 6(01):59–77.
- Chen, L., Özsü, M. T., and Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*, pages 491–502. ACM.
- Cook, K. A. and Thomas, J. J. (2005). *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society, Los Alamitos, CA, United States of America.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT Press, Cambridge, MA, United States of America.

- Daneshpajouh, S., Ghodsi, M., and Zarei, A. (2012). Computing polygonal path simplification under area measures. *Graphical Models*, 74(5):283–289.
- Demšar, U., Buchin, K., Cagnacci, F., Safi, K., Speckmann, B., Van de Weghe, N., Weiskopf, D., and Weibel, R. (2015). Analysis and visualisation of movement: an interdisciplinary review. *Movement Ecology*, 3(1):5.
- Dingle, H. (2014). *Migration: the biology of life on the move*. Oxford University Press, Oxford, United Kingdom.
- Dinkla, K., El-Kebir, M., Bucur, C.-I., Siderius, M., Smit, M. J., Westenberg, M. A., and Klau, G. W. (2014). eXamine: Exploring annotated modules in networks. *BMC Bioinformatics*, 15(1):201.
- Dodge, S., Weibel, R., Ahearn, S. C., Buchin, M., and Miller, J. A. (2016). Analysis of movement data. *International Journal of Geographic Information Science*, 30(5):825 – 834.
- Dodge, S., Weibel, R., and Lautenschütz, A.-K. (2008). Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252.
- Doncaster, C. P. (1990). Non-parametric estimates of interaction from radio-tracking data. *Journal of Theoretical Biology*, 143(4):431–443.
- Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.
- Driemel, A., Har-Peled, S., and Wenk, C. (2012). Approximating the Fréchet distance for realistic curves in near linear time. *Discrete and Computational Geometry*, 48(1):94–127.
- Dumitrescu, A. and Rote, G. (2004). On the Fréchet distance of a set of curves. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG 2004)*, pages 162–165.
- Eiter, T. and Mannila, H. (1994). Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Technical University of Vienna.

- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Workshop on Knowledge Discovery in Databases (KDD 1996)*, volume 96, pages 226–231.
- Forsythe, W. C., Rykiel, E. J., Stahl, R. S., Wu, H.-i., and Schoolfield, R. M. (1995). A model comparison for daylength as a function of latitude and day of year. *Ecological Modelling*, 80(1):87–95.
- Fredman, M. L., Sedgewick, R., Sleator, D. D., and Tarjan, R. E. (1986). The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1(1):111–129.
- Gaffney, S. and Smyth, P. (1999). Trajectory clustering with mixtures of regression models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999)*, pages 63–72.
- Gajentaan, A. and Overmars, M. H. (1995). On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5(3):165–185.
- Giuggioli, L., McKetterick, T. J., and Holderied, M. (2015). Delayed response and biosonar perception explain movement coordination in trawling bats. *PLOS Computational Biology*, 11(3):1–21.
- van Goethem, A., van Kreveld, M., Löffler, M., Speckmann, B., and Staals, F. (2016). Grouping time-varying data for interactive exploration. *arXiv preprint arXiv:1603.06252*.
- Gold, O. and Sharir, M. (2016). Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *arXiv preprint arXiv:1607.05994*.
- Goodman, J. E., O'Rourke, J., and Toth, C. D. (2017). *Handbook of discrete and computational geometry*. CRC Press, Boca Raton (FL), United States of America.
- Greenfeld, J. S. (2002). Matching gps observations to locations on a digital map. In *81th Annual Meeting of the Transportation Research Board*, volume 1, pages 164–173.
- Gudmundsson, J., Katajainen, J., Merrick, D., Ong, C., and Wolle, T.

- (2009). Compressing spatio-temporal trajectories. *Computational Geometry: Theory and Applications*, 42(9):825–841.
- Gudmundsson, J., van Kreveld, M., and Staals, F. (2013). Algorithms for hotspot computation on trajectory data. In *Proceedings of the 21st ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM GIS 2013)*, pages 134–143.
- Gudmundsson, J. and Valladares, N. (2015). A GPU approach to subtrajectory clustering using the Fréchet distance. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):924–937.
- Gurarie, E., Bracis, C., Delgado, M., Meckley, T. D., Kojola, I., and Wagner, C. M. (2016). What is the animal doing? Tools for exploring behavioural structure in animal movements. *Journal of Animal Ecology*, 85(1):69–84.
- Hägerstrand, T. (1970). What about people in regional science? In *Papers in Regional Science Association*, volume 24, pages 6–21. Springer-Verlag, Berlin, Germany.
- Hamming, R. (1987). *Numerical methods for scientists and engineers*. Dover Publications, Mineola (NY), United States of America.
- Hausdorff, F. (1914). *Grundzüge der Mengenlehre*. Verlag Von Veit & Comp., Leipzig, Germany.
- Hedenstrom, A. (1993). Migration by soaring or flapping flight in birds: the relative importance of energy cost and speed. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 342(1302):353–361.
- Hershberger, J. and Snoeyink, J. (1994). An $O(n \log n)$ implementation of the douglas-peucker algorithm for line simplification. In *Proceedings of the 10th Annual Symposium on Computational Geometry (SoCG 1994)*, pages 383–384. ACM.
- Hershberger, J. and Snoeyink, J. (1998). Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time. *Computational Geometry: Theory and Applications*, 11(3-4):175–185.
- Imai, H. and Iri, M. (1986). Computational-geometric methods for

- polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41.
- Imai, H. and Iri, M. (1988). Polygonal approximations of a curve – formulations and algorithms. In Toussaint, G. T., editor, *Computational Morphology*, pages 71–86. Elsevier, Oxford, United Kingdom.
- Kareiva, P. and Shigesada, N. (1983). Analyzing insect movement as a correlated random walk. *Oecologia*, 56(2-3):234–238.
- Kays, R., Crofoot, M. C., Jetz, W., and Wikelski, M. (2015). Terrestrial animal tracking as an eye on life and planet. *Science*, 348(6240):aaa2478.
- Keim, D. A., Kohlhammer, J., Ellis, G., and Mansmann, F. (2010a). *Mastering the information age solving problems with visual analytics*. Eurographics Association.
- Keim, D. A., Mansmann, F., and Thomas, J. (2010b). Visual analytics: how much visualization and how much analytics? *ACM SIGKDD Explorations Newsletter*, 11(2):5–8.
- Klaassen, R. H., Ens, B. J., Shamoun-Baranes, J., Exo, K.-M., and Bairlein, F. (2011). Migration strategy of a flight generalist, the lesser black-backed gull larus fuscus. *Behavioral Ecology*, 23(1):58–68.
- Kölzsch, A., Slingsby, A., Wood, J., Nolet, B., and Dykes, J. (2013). Visualisation design for representing bird migration tracks in time and space. In *Short Papers of the Workshop on Visualisation in Environmental Sciences (EnvirVis 2013)*.
- Kostitsyna, I., van Kreveld, M., Löffler, M., Speckmann, B., and Staals, F. (2015). Trajectory grouping structure under geodesic distance. In *Proceedings of the 31st Annual Symposium on Computational Geometry (SoCG 2015)*, volume 34, pages 674–688.
- van Kreveld, M. J., Löffler, M., and Staals, F. (2015). Central trajectories. *arXiv preprint arXiv:1501.01822*.
- Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50.

- Kurzhals, K. and Weiskopf, D. (2013). Space-time visual analytics of eye-tracking data for dynamic stimuli. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2129–2138.
- Lam, H., Tory, M., and Munzner, T. (2018). Bridging from goals to tasks with design study analysis reports. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):435–445.
- Laube, P. (2014). *Computational movement analysis*. Springer-Verlag, Berlin, Germany.
- Laube, P. (2015). The low hanging fruit is gone: achievements and challenges of computational movement analysis. *SIGSPATIAL Special*, 7(1):3–10.
- Laube, P., Dennis, T., Forer, P., and Walker, M. (2007). Movement beyond the snapshot—dynamic analysis of geospatial lifelines. *Computers, Environment and Urban Systems*, 31(5):481–501.
- Laube, P. and Purves, R. S. (2011). How fast is a cow? cross-scale analysis of movement data. *Transactions in GIS*, 15(3):401–418.
- Lavielle, M. (1999). Detection of multiple changes in a sequence of dependent variables. *Stochastic Processes and their Applications*, 83(1):79–102.
- Le Corre, M., Dussault, C., and Côté, S. D. (2014). Detecting changes in the annual movements of terrestrial migratory species: using the first-passage time to document the spring migration of caribou. *Movement Ecology*, 2(1):19.
- Lee, J.-G., Han, J., Li, X., and Gonzalez, H. (2008). TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094.
- Lee, J.-G., Han, J., and Whang, K.-Y. (2007). Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD 2007)*, pages 593–604. ACM.
- Leite, R. A., Gschwandtner, T., Miksch, S., Kriglstein, S., Pohl, M., Gstreich, E., and Kuntner, J. (2018). EVA: Visual analytics to identify

- fraudulent events. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):330–339.
- Long, J. A. and Nelson, T. A. (2013a). Measuring dynamic interaction in movement data. *Transactions in GIS*, 17(1):62–77.
- Long, J. A. and Nelson, T. A. (2013b). A review of quantitative methods for movement data. *International Journal of Geographic Information Science*, 27(2):292–318.
- Losada, A. G., Therón, R., and Benito, A. (2016). Bkviz: A basketball visual analysis tool. *IEEE Computer Graphics and Applications*, 36(6):58–68.
- Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., and Huang, Y. (2009). Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 352–361. ACM.
- Lu, M., Wang, Z., Liang, J., and Yuan, X. (2015a). OD-Wheel: Visual design to explore od patterns of a central region. In *Proceedings of the 8th IEEE Pacific Visualization Symposium (PacificVis 2015)*, pages 87–91. IEEE.
- Lu, M., Wang, Z., and Yuan, X. (2015b). Trajrank: Exploring travel behaviour on a route by trajectory ranking. In *Proceedings of the 8th IEEE Pacific Visualization Symposium (PacificVis 2015)*, pages 311–318. IEEE.
- Madon, B. and Hingrat, Y. (2014). Deciphering behavioral changes in animal movement with a ‘multiple change point algorithm-classification tree’ framework. *Frontiers in Ecology and Evolution*, 2:30.
- Maier, D. (1978). The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25(2):322–336.
- Masek, W. J. and Paterson, M. S. (1980). A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31.
- Mayer, R. E. (1995). The search for insight: Grappling with gestalt psy-

- chology's unanswered questions. In *The Nature of Insight*, pages 3 – 32. The MIT Press.
- Melkman, A. and O'Rourke, J. (1988). On polygonal chain approximation. In Toussaint, G. T., editor, *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 87–95. Elsevier, Oxford, United Kingdom.
- Merki, M. and Laube, P. (2012). Detecting reaction movement patterns in trajectory data. In *Proceedings of the 15th AGILE International Conference on Geographic Information Science (AGILE 2012)*, pages 25 – 27.
- Munzner, T. (2014). *Visualization Analysis and Design*. CRC Press, Boca Raton (FL), United States of America.
- Nagy, M., Ákos, Z., Biro, D., and Vicsek, T. (2010). Hierarchical group dynamics in pigeon flocks. *Nature*, 464(7290):890–893.
- Nathan, R., Getz, W. M., Revilla, E., Holyoak, M., Kadmon, R., Saltz, D., and Smouse, P. E. (2008). A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences*, 105(49):19052–19059.
- Nathan, R. and Giuggioli, L. (2013). A milestone for movement ecology research. *Movement Ecology*, 1(1):1.
- Newson, P. and Krumm, J. (2009). Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM GIS 2009)*, pages 336–343.
- Newton, I. (2008). *The Migration Ecology of Birds*. Elsevier, Oxford, United Kingdom.
- Pettit, B., Perna, A., Biro, D., and Sumpter, D. J. T. (2013). Interaction rules underlying group decisions in homing pigeons. *Journal of The Royal Society Interface*, 10(89):20130529.
- Pileggi, H., Stolper, C. D., Boyle, J. M., and Stasko, J. T. (2012). Snapshot: Visualization to propel ice hockey analytics. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2819–2828.

- Qingsheng, G., Brandenberger, C., and Hurni, L. (2002). A progressive line simplification algorithm. *Geo-spatial Information Science*, 5(3):41–45.
- Ritter, J. (1990). An efficient bounding sphere. In *Graphics Gems*, pages 301–303. Academic Press Professional, Inc.
- Rote, G. (2014). Lexicographic Fréchet matchings. In *Proceedings of the 30th European Workshop on Computational Geometry (EuroCG 2014)*.
- Rutz, C. and Hays, G. C. (2009). New frontiers in biologging science. *Biology Letters*, 5(3):289 – 292.
- Sacha, D., Senaratne, H., Kwon, B. C., Ellis, G., and Keim, D. A. (2016). The role of uncertainty, awareness, and trust in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):240–249.
- Sack, J.-R. and Urrutia, J. (1999). *Handbook of computational geometry*. Elsevier, Oxford, United Kingdom.
- Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.
- Santos, C. D., Neupert, S., Lipp, H.-P., Wikelski, M., and Dechmann, D. K. (2014a). Data from: Temporal and contextual consistency of leadership in homing pigeon flocks. *Movebank Data Repository*.
- Santos, C. D., Neupert, S., Lipp, H.-P., Wikelski, M., and Dechmann, D. K. (2014b). Temporal and contextual consistency of leadership in homing pigeon flocks. *PLOS ONE*, 9(7):1–5.
- Scheepens, R., Willems, N., van de Wetering, H., Andrienko, G., Andrienko, N., and van Wijk, J. J. (2011). Composite density maps for multivariate trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2518–2527.
- Schmidt-Rothmund, D. (<http://www.movebank.org> accessed in June 2017.). Griffon Vulture NABU Moessingen, 186178781. *Movebank: archive, analysis and sharing of animal movement data. World Wide Web electronic publication*.

- Schulz, H.-J., Nocke, T., Heitzler, M., and Schumann, H. (2013). A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375.
- Sedgewick, R. (2008). Left-leaning red-black trees. In *Dagstuhl Workshop on Data Structures*, page 17.
- Shamoun-Baranes, Judy and van Loon, E. E., Purves, R. S., Speckmann, B., Weiskopf, D., and Camphuysen, C. J. (2011a). Analysis and visualization of animal movement. *Biology Letters*.
- Shamoun-Baranes, J., Bouts, W., Camphuysen, C. J., and Baaij, E. (2011b). Riding the tide: intriguing observations of gulls resting at sea during breeding. *Ibis*, 153(2):411–415.
- Shamoun-Baranes, J., Burant, J. B., van Loon, E. E., Bouts, W., and Camphuysen, C. (2017). Short distance migrants travel as far as long distance migrants in lesser black-backed gulls larus fuscus. *Journal of Avian Biology*, 48(1):49–57.
- Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages 1996*, pages 336–343. IEEE.
- Slingsby, A. and Dykes, J. (2012). Experiences in involving analysts in visualisation design. In *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors-Novel Evaluation Methods for Visualization*, pages 1–4. ACM.
- Slingsby, A., Dykes, J., and Wood, J. (2008). Using treemaps for variable selection in spatio-temporal visualisation. *Information Visualization*, 7(3-4):210–224.
- Slingsby, A. and van Loon, E. (2016). Exploratory visual analysis for animal movement ecology. In *Computer Graphics Forum*, volume 35, pages 471–480. John Wiley & Sons, Inc., Hoboken (NJ), United States of America.
- Spretke, D., Bak, P., Janetzko, H., Kranstauber, B., Mansmann, F., and Davidson, S. (2011). Exploration through enrichment: a visual analytics approach for animal movement. In *Proceedings of the 19th ACM SIGSPATIAL International Symposium on Advances in Ge-*

- ographic Information Systems (ACM GIS 2011), pages 421–424. ACM.
- Stienen, E. W., Desmet, P., Aelterman, B., Courtens, W., Feys, S., Vannermen, N., Verstraete, H., Van de Walle, M., Deneudt, K., Hernandez, F., et al. (2016). GPS tracking data of Lesser Black-backed Gulls and Herring Gulls breeding at the southern North Sea coast. *ZooKeys*, (555):115 – 124.
- Sun, G.-D., Wu, Y.-C., Liang, R.-H., and Liu, S.-X. (2013). A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867.
- Thiebault, A. and Tremblay, Y. (2013). Splitting animal trajectories into fine-scale behaviorally consistent movement units: breaking points relate to external stimuli in a foraging seabird. *Behavioral Ecology and Sociobiology*, 67(6):1013–1026.
- Tory, M. and Moller, T. (2005). Evaluating visualizations: do expert reviews work? *IEEE Computer Graphics and Applications*, 25(5):8–11.
- Toussaint, G. T. (1985). On the complexity of approximating polygonal curves in the plane. In *Proceedings of the International Symposium Robotics and Automation IASTED*.
- Visvalingam, M. and Whyatt, J. D. (1993). Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51.
- Vlachos, M., Kollios, G., and Gunopulos, D. (2002). Discovering similar multidimensional trajectories. In *Proceedings of 18th International Conference on Data Engineering (ICDE 2002)*, pages 673–684. IEEE.
- Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.
- Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation meth-

- ods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309.
- van Wijk, J. J. (2006). Bridging the gaps. *IEEE Computer Graphics and Applications*, 26(6):6–9.
- Willems, N., van de Wetering, H., and van Wijk, J. J. (2009). Visualization of vessel movements. In *Computer Graphics Forum*, volume 28, pages 959–966. John Wiley & Sons, Inc., Hoboken (NJ), United States of America.
- Williams, R. (2005). A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365.
- Wood, Z., Galton, A., Bhatt, M., Guesgen, H., and Hazarika, S. (2010). Zooming in on collective motion. In *Proceedings of 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 25–30.
- Zhang, J., O'Reilly, K. M., Perry, G. L., Taylor, G. A., and Dennis, T. E. (2015). Extending the functionality of behavioural change-point analysis with k -means clustering: a case study with the little penguin (*eudyptula minor*). *PLOS ONE*, 10(4):e0122811.
- Zhao, J., Glueck, M., Chevalier, F., Wu, Y., and Khan, A. (2016). Egocentric analysis of dynamic networks with egolines. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI 2016)*, pages 5003–5014. ACM.
- Zheng, Y. (2015). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):29.
- Zheng, Y., Chen, Y., Xie, X., and Ma, W.-Y. (2009). GeoLife2.0: a location-based social networking service. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM 2009)*, pages 357–358. IEEE.
- Zheng, Y., Li, Q., Chen, Y., Xie, X., and Ma, W.-Y. (2008a). Understanding mobility based on GPS data. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp 2008)*, pages 312–321. ACM.

- Zheng, Y., Liu, L., Wang, L., and Xie, X. (2008b). Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*, pages 247–256. ACM.

List of Publications

- Buchin, K., Buchin, M., Konzack, M., Mulzer, W., and Schulz, A. (2016). Fine-grained analysis of problems on curves. In *Proceedings of the 32nd European Workshop on Computational Geometry (EuroCG 2016)*.
- Buchin, K., Konzack, M., and Reddingius, W. (2018). Progressive simplification of polygonal curves. In *Proceedings of the 34nd European Workshop on Computational Geometry (EuroCG 2018)*.
- Konzack, M., Gijsbers, P., Timmers, F., van Loon, E., Westenberg, M. A., and Buchin, K. (2018). Visual exploration of migration patterns in gull data. *Information Visualization*, page 1473871617751245.
- Konzack, M., McKetterick, T., Ophelders, T., Buchin, M., Giuggioli, L., Long, J., Nelson, T., Westenberg, M. A., and Buchin, K. (2017). Visual analytics of delays and interaction in movement data. *International Journal of Geographic Information Science*, 31(2):320–345.
- Konzack, M., McKetterick, T. J., Wilcox, G., Buchin, M., Giuggioli, L., Gudmundsson, J., Westenberg, M. A., and Buchin, K. (2015). Analyzing delays in trajectories. In *Proceedings of the 8th IEEE Pacific Visualization Symposium (PacificVis 2015)*, pages 93–97.

Summary

Smaller and cheaper tracking devices with at the same time higher accuracy allow researchers to track large numbers of individuals to in their quest to understand the phenomena behind movement. The availability of this technology led to the continuing trend to collect, publish, and share movement datasets in data repositories. Because movement is ubiquitous, researchers from various disciplines are involved in tracking individuals, analyzing their trajectories, and ultimately, understanding the connection between movement and its drivers.

Methods to analyze these datasets have been developed in a variety of research areas. For example, researchers in algorithms are concerned with analyzing geometric problems and designing new algorithms for trajectory data, and visualization researchers develop novel visual representations from trajectory data to gain insights. Domain experts, from ecology, urban planning, or sports analytics, need new analytical methods for both: methods to quantify spatio-temporal properties and means to analyze movement data qualitatively, for instance exploration. To date, new methods in algorithms and visualization are being developed only within their field. However, both algorithms and visualization are crucial and complement each other in the analysis of movement data.

In this thesis, we explore how combining algorithms and visualization can enhance the analysis of movement data. Thus, we aim to fill the methodological gap between algorithms and visualization by integrating computations, their context, and their visual representations more closely. Filling this gap will help movement analysts to externalize their cognition by integrating algorithmic means, visual means, and their domain knowledge into a holistic tooling. The contributions of this thesis make it possible for movement analysts to benefit from

the exchange of ideas and concepts between algorithms and visualization.

Our contributions to the analysis of movement are manifold. We provide a thorough overview of the state-of-the-art in movement analysis in which we survey results to computational movement analysis as well as advances in the visualization of movement. We present a new taxonomy for the analysis of trajectory data, that can aid researchers in designing new analytical methods.

The second contribution of this thesis are results on the computational complexity of movement analysis tasks. We present two new lower bounds. We show that a simplification cannot be computed in subquadratic time for a trajectory of n points, presuming those points lie in $\Omega(\log n)$ dimensions and assuming the Strong Exponential Time Hypothesis. Then, we prove that the discrete Fréchet distance for k trajectories, each of length n , cannot be computed in $O(n^{k-\varepsilon})$ time for any $\varepsilon > 0$ assuming the Strong Exponential Time Hypothesis. Furthermore, we provide an overview of previous results on algorithms and their algorithmic complexity for analyzing movement data.

Subsequently, we present a new algorithm to compute progressive simplifications, i.e., a series of simplifications that are consistent across multiple scales, in $O(n^3m)$ time with n as the number of points in the input curve and m as the number of scales. Progressive simplifications are particularly important for showing trajectories (or other line features) on interactive, zoomable maps. Our algorithm computes a progressive simplification for a range of continuous scales in $O(n^5)$ time. A core element in simplification algorithms is the so-called shortcut graph. It stores for any line segment whether the segment approximates its induced subcurve given an error value. Moreover, we developed a new representation for such shortcut graphs allowing us to compute (non-)progressive simplifications more efficiently: a technique for computing the maximum error for all shortcuts in $O(n^2 \log n)$ time instead of $O(n^3)$ time, and a compressed shortcut graph allowing us to find shortest paths typically in $O(n \log n)$ time.

Our fourth contribution in this thesis is a versatile visual analytics tool to explore interaction events as action-reaction patterns between two (or three) trajectories. We use alignment methods that allow us to capture delayed responses; we visualize these delays in addition to statistics on the alignment. Furthermore, we present a novel approach for computing a global delay between two trajectories, each of consisting of n points, in $O(n \log n)$ time, by employing Fast Fourier Transforms. We applied our approach to three different datasets and compared it to existing approaches on dynamic interaction.

The final contribution of this thesis is a visual analytics approach that helps ecologists to explore animal migration patterns interactively. We identify and aggregate stopovers, which are breaks from migration. The aggregation is visualized on top of a geographic map for varying spatial scales in addition to interconnected views to explore spatio-temporal events. We evaluated our approach with an expert user on a dataset of 75 lesser black-backed gulls.

Curriculum Vitæ

Maximilian Konzack was born on 17 February 1987 in Schwabach, Germany. He completed his secondary education at Fachoberschule Fürth (Germany) in 2005. After the alternative civilian service of 10 months at a sheltered workshop, he studied computer science for his Bachelor's degree at the University of Applied Science in Nuremberg, Germany from 2006 until 2010. Next to a sabbatical in social work of six months, he continued his studies in the Masters' program for computer science at the Free University Berlin. His Master's thesis was on crossing numbers of spanning trees, supervised by prof dr. Wolfgang Mulzer. Then, he joined the Google's Summer of Code 2013 for a three months project, before he started working as a PhD student in the Algorithms Group at the Eindhoven University of Technology in 2013. The outcomes of this work are elaborated in this thesis.

Titles in the IPA Dissertation Series since 2015

G. Alpár. *Attribute-Based Identity Management: Bridging the Cryptographic Design of ABCs with the Real World.* Faculty of Science, Mathematics and Computer Science, RU. 2015-01

A.J. van der Ploeg. *Efficient Abstractions for Visualization and Interaction.* Faculty of Science, UvA. 2015-02

R.J.M. Theunissen. *Supervisory Control in Health Care Systems.* Faculty of Mechanical Engineering, TU/e. 2015-03

T.V. Bui. *A Software Architecture for Body Area Sensor Networks: Flexibility and Trustworthiness.* Faculty of Mathematics and Computer Science, TU/e. 2015-04

A. Guzzi. *Supporting Developers' Teamwork from within the IDE.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-05

T. Espinha. *Web Service Growing Pains: Understanding Services and Their Clients.* Faculty of Electrical Engineering, Math-

ematics, and Computer Science, TUD. 2015-06

S. Dietzel. *Resilient In-network Aggregation for Vehicular Networks.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-07

E. Costante. *Privacy throughout the Data Cycle.* Faculty of Mathematics and Computer Science, TU/e. 2015-08

S. Cranen. *Getting the point — Obtaining and understanding fixpoints in model checking.* Faculty of Mathematics and Computer Science, TU/e. 2015-09

R. Verdult. *The (in)security of proprietary cryptography.* Faculty of Science, Mathematics and Computer Science, RU. 2015-10

J.E.J. de Ruiter. *Lessons learned in the analysis of the EMV and TLS security protocols.* Faculty of Science, Mathematics and Computer Science, RU. 2015-11

Y. Dajsuren. *On the Design of an Architecture Framework and*

Quality Evaluation for Automotive Software Systems. Faculty of Mathematics and Computer Science, TU/e. 2015-12

J. Bransen. *On the Incremental Evaluation of Higher-Order Attribute Grammars.* Faculty of Science, UU. 2015-13

S. Picek. *Applications of Evolutionary Computation to Cryptology.* Faculty of Science, Mathematics and Computer Science, RU. 2015-14

C. Chen. *Automated Fault Localization for Service-Oriented Software Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-15

S. te Brinke. *Developing Energy-Aware Software.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-16

R.W.J. Kersten. *Software Analysis Methods for Resource-Sensitive Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2015-17

J.C. Rot. *Enhanced coinduction.* Faculty of Mathematics and Natural Sciences, UL. 2015-18

M. Stolikj. *Building Blocks for the Internet of Things.* Faculty of Mathematics and Computer Science, TU/e. 2015-19

D. Gebler. *Robust SOS Specifications of Probabilistic Processes.* Faculty of Sciences, Department of Computer Science, VUA. 2015-20

M. Zaharieva-Stojanovski. *Closer to Reliable Software: Verifying functional behaviour of concurrent programs.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-21

R.J. Krebbers. *The C standard formalized in Coq.* Faculty of Science, Mathematics and Computer Science, RU. 2015-22

R. van Vliet. *DNA Expressions – A Formal Notation for DNA.* Faculty of Mathematics and Natural Sciences, UL. 2015-23

S.-S.T.Q. Jongmans. *Automata-Theoretic Protocol Programming.* Faculty of Mathematics and Natural Sciences, UL. 2016-01

S.J.C. Joosten. *Verification of Interconnects.* Faculty of Mathematics and Computer Science, TU/e. 2016-02

- M.W. Gazda.** *Fixpoint Logic, Games, and Relations of Consequence.* Faculty of Mathematics and Computer Science, TU/e. 2016-03 TU/e. 2016-08
- S. Keshishzadeh.** *Formal Analysis and Verification of Embedded Systems for Healthcare.* Faculty of Mathematics and Computer Science, TU/e. 2016-04 T. van Dijk. *Sylvan: Multi-core Decision Diagrams.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2016-09
- P.M. Heck.** *Quality of Just-in-Time Requirements: Just-Enough and Just-in-Time.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2016-05 I. David. *Run-time resource management for component-based systems.* Faculty of Mathematics and Computer Science, TU/e. 2016-10
- Y. Luo.** *From Conceptual Models to Safety Assurance – Applying Model-Based Techniques to Support Safety Assurance.* Faculty of Mathematics and Computer Science, TU/e. 2016-06 A.C. van Hulst. *Control Synthesis using Modal Logic and Partial Bisimilarity – A Treatise Supported by Computer Verified Proofs.* Faculty of Mechanical Engineering, TU/e. 2016-11
- B. Ege.** *Physical Security Analysis of Embedded Devices.* Faculty of Science, Mathematics and Computer Science, RU. 2016-07 A. Zawedde. *Modeling the Dynamics of Requirements Process Improvement.* Faculty of Mathematics and Computer Science, TU/e. 2016-12
- A.I. van Goethem.** *Algorithms for Curved Schematization.* Faculty of Mathematics and Computer Science, J.N. van Rijn. *Massively Collaborative Machine Learning.* Faculty of Mathematics and Natural Sciences, UL. 2016-14

M.J. Steindorfer. *Efficient Immutable Collections.* Faculty of Science, UvA. 2017-01

W. Ahmad. *Green Computing: Efficient Energy Management of Multiprocessor Streaming Applications via Model Checking.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2017-02

D. Guck. *Reliable Systems – Fault tree analysis via Markov reward automata.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2017-03

H.L. Salunkhe. *Modeling and Buffer Analysis of Real-time Streaming Radio Applications Scheduled on Heterogeneous Multiprocessors.* Faculty of Mathematics and Computer Science, TU/e. 2017-04

A. Krasnova. *Smart invaders of private matters: Privacy of communication on the Internet and in the Internet of Things (IoT).* Faculty of Science, Mathematics and Computer Science, RU. 2017-05

A.D. Mehrabi. *Data Structures for Analyzing Geometric Data.* Faculty of Mathe-

matics and Computer Science, TU/e. 2017-06

D. Landman. *Reverse Engineering Source Code: Empirical Studies of Limitations and Opportunities.* Faculty of Science, UvA. 2017-07

W. Lueks. *Security and Privacy via Cryptography – Having your cake and eating it too.* Faculty of Science, Mathematics and Computer Science, RU. 2017-08

A.M. Sutii. *Modularity and Reuse of Domain-Specific Languages: an exploration with MetaMod.* Faculty of Mathematics and Computer Science, TU/e. 2017-09

U. Tikhonova. *Engineering the Dynamic Semantics of Domain Specific Languages.* Faculty of Mathematics and Computer Science, TU/e. 2017-10

Q.W. Bouts. *Geographic Graph Construction and Visualization.* Faculty of Mathematics and Computer Science, TU/e. 2017-11

A. Amighi. *Specification and Verification of Synchronisation Classes in Java: A Practical Approach.* Faculty of

Electrical Engineering, Mathematics & Computer Science,
UT. 2018-01

S. Darabi. *Verification of Program Parallelization.* Faculty of Electrical Engineering, Mathematics & Computer Science,
UT. 2018-02

J.R. Salamanca Tellez. *Coequations and Eilenberg-type Correspondences.* Faculty of Science, Mathematics and Computer Science, RU. 2018-03

P. Fiterau-Brosteau. *Active Model Learning for the Analysis of Network Protocols.* Faculty of Science, Mathematics and Computer Science,
RU. 2018-04

D. Zhang. *From Concurrent State Machines to Reliable Multi-threaded Java Code.* Faculty of Mathematics and Computer Science,

TU/e. 2018-05

H. Basold. *Mixed Inductive-Coinductive Reasoning Types, Programs and Logic.* Faculty of Science, Mathematics and Computer Science, RU. 2018-06

A. Lele. *Response Modeling: Model Refinements for Timing Analysis of Runtime Scheduling in Real-time Streaming Systems.* Faculty of Mathematics and Computer Science,
TU/e. 2018-07

N. Bezirgiannis. *Abstract Behavioral Specification: unifying modeling and programming.* Faculty of Mathematics and Natural Sciences,
UL. 2018-08

M.P. Konzack. *Trajectory Analysis: Bridging Algorithms and Visualization.* Faculty of Mathematics and Computer Science,
TU/e. 2018-09