COMPUTATIONS AND MEASURES OF COLLECTIVE MOVEMENT PATTERNS BASED ON TRAJECTORY DATA

LIONOV WIRATMA

Cover design by Lionov Wiratma
The cover was created using Ipe—the extensible drawing editor—and the Utrecht University Crowd Simulation Framework

This thesis was typeset using the LaTeX typesetting system

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/

ISBN: 978-90-393-7223-4

Computations and Measures of Collective Movement Patterns Based on Trajectory Data

Berekeningen en Maten voor Collectieve Bewegingspatronen in Trajectdata

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op
vrijdag 20 december 2019 des ochtends te 10.30 uur

door

Lionov Wiratma

geboren op 30 november 1977 te Bandung, Indonesië

Promotor: Prof. dr. M.J. van Kreveld
Copromotor: Dr. M. Löffler

This thesis was accomplished with financial support from the Ministry of Research, Technology and Higher Education of the Republic of Indonesia

CONTENTS

1 INTRODUCTION 1

1.1 Preliminaries 2
1.1.1 Computational Geometry 3
1.1.2 Geographic Information Systems and Science 3
1.1.3 Computational Movement Analysis 4
1.2 Trajectories 5
1.2.1 Notation 8
1.2.2 Trajectory Data 9
1.3 Trajectory Analysis 11
1.3.1 Examples of Analysis Tasks on Trajectory Data 11
1.3.2 Movement Patterns 15
1.3.3 Collective Movement 16
1.4 Outline of the Thesis 18

2 POLYLINE SIMPLIFICATION 27
2.1 Preliminaries 30
2.2 Approximation Quality of Douglas-Peucker and Imai-Iri Simplification 32
2.2.1 Hausdorff Distance 33
2.2.2 Fréchet Distance 33
2.3 Algorithmic Complexity of the Hausdorff Distance 37
2.3.1 Undirected Hausdorff Distance 37
2.3.2 Directed Hausdorff Distance: $\mathcal{P}$ to $\mathcal{Q}$ 40
2.3.3 Directed Hausdorff Distance: $\mathcal{Q}$ to $\mathcal{P}$ 48
2.4 Algorithmic Complexity of the Fréchet Distance 49
2.4.1 Observations 49
2.4.2 A Dynamic Programming Algorithm 50
2.5 Conclusions and Future Work 52

3 MEASURES FOR GROUPS 55
3.1 Measures for a Single Trajectory 56
3.1.1 Measures for a Trajectory in Isolation 57
3.1.2 Measures for a Trajectory Amidst Other Trajectories . . . 60
3.1.3 Measures for a Trajectory in an Environment 62
3.2 Measures for a Group of Trajectories 63
3.2.1 Measures for a Single Group in Isolation 64
3.2.2 Measures for a Group Amidst Other Groups 66
3.2.3 Measures for a Group of Trajectories in an Environment . 68
3.3 Applications of Group Measures 69
3.4 Conclusions and Future Work 71

4 A REFINED DEFINITION FOR GROUPS 73
4.1 Preliminaries 76
4.2 Algorithms for Entities in $\mathbb{R}^1$ 78
4.2.1 Basic Algorithm 78
4.2.2 Improved Algorithm 79
4.3 Algorithms for Entities in $\mathbb{R}^d$ 81
4.4 Algorithms with Linear Dependence on τ 82
4.5 A Lower Bound on the Maximum Number of Maximal Groups at some Time t 84
4.6 Conclusions and Future Work 87

5 AN EXPERIMENTAL EVALUATION OF GROUPING DEFINITIONS 89
5.1 The Definitions 91
5.2 Methods 95
5.2.1 Experimental Setup 95
5.2.2 Evaluation Setup 96
5.3 Experimental Evaluation 97
5.3.1 Comparisons with Annotated Groups and Social Formations 98
5.3.2 Dependence on Density 108
5.3.3 Dependence on Sampling Rate 115
5.4 Conclusions and Future Work 117

6 CONCLUSIONS 119
6.1 Summary 119
6.2 Open Problems 120
6.3 Future Research 121

BIBLIOGRAPHY 125

SAMENVATTING 143

ACKNOWLEDGEMENTS 147

CURRICULUM VITAE 151

1

INTRODUCTION

EVERYTHING around us is moving. Humans move between different locations to do their activities. Many species of animals migrate from one habitat to another to take advantage of favorable weather conditions and better food supplies. Natural phenomena like glaciers or hurricanes also move along a path influenced by various factors such as wind, temperature, and air pressure. Other objects like packages and documents are moving while being delivered to recipients by couriers. Even the universe is in motion: the earth revolves around the sun on its orbital path.

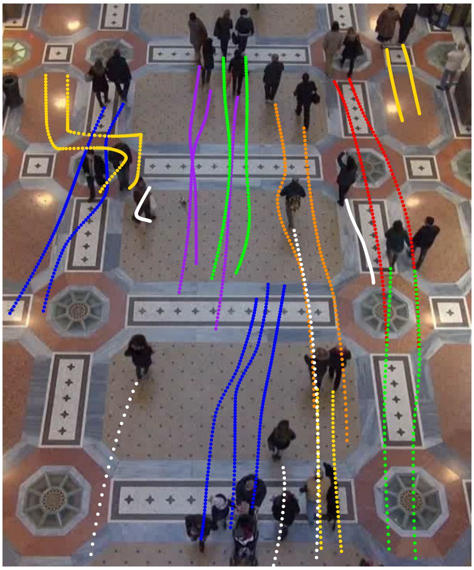


FIGURE 1.1: The movement of pedestrians. For each pedestrian, the sequence of small discs shows her/his previous positions. Intervals between consecutive discs indicate the walking speed. Pedestrians with a white trail are walking alone, while pedestrians with the same color of the trail and spatially close are probably walking together.

We show an example of movement by moving entities in Figure 1.1:[1] several pedestrians are walking in a hallway. The colored discs following each pedestrian show their previous locations. Therefore, the gap between consecutive discs indicates the walking speed of a pedestrian. If the gaps are small, then the pedestrian is moving slowly; if the gaps get larger, they speed up. Furthermore, the use of the same color and spatial proximity of the discs suggests that a few pedestrians are walking together. Identifying a set of entities (not necessarily pedestrians) moving together is a common task in research involving movement data. This thesis is primarily about one specific type of movement: *collective movement*. A collective movement can be defined informally as *multiple moving entities that travel together over a period of time*. We study various aspects of a collective movement. First, we discuss various measures that might be needed to perform different types of analysis on collective movement. Next, we introduce a new definition for collective movement and present algorithms to compute it. Finally, we compare four definitions for collective movement experimentally using different types of movement data sets.

In the rest of this chapter, we start with a short introduction to research areas related to movement analysis. A brief summary of trajectories as movement data follows it. Then, we give an overview of some analysis tasks on movement data. We present different types of movement patterns and give a more detailed explanation of the collective movement pattern. An outline of the remainder of the thesis is the last part of this chapter.

1.1 Preliminaries

Movement data from various moving entities has become a new kind of data that has gained significant interest among researchers from different research areas. For instance, by analyzing the movement of free-grazing domestic ducks, researchers can identify areas that may have the highest risk of avian influenza transmission [2].

Since movement is continuously changing the (geographic) location of an entity over time, analysis of movement data has become a mainstream topic in the field of *Geographic Information Systems and Science*. Furthermore, since movement data is essentially spatial, methods used in various types of analysis are often based on geometric objects and algorithms to compute or manipulate them. Hence, analysis of movement data also became an important research focus in the field of *Computational Geometry*.

---

[1] The background image and movement data are from [1].

1.1.1 Computational Geometry

Computational geometry, which has been developed since the nineteen-seventies, is one of many branches of theoretical computer science. It mainly focuses on the design and analysis of efficient algorithms and data structures for solving computational problems involving geometric objects (e.g., points, lines, line segments, polygons) [3]. Even though geometric problems have been studied before, the use of modern computers increases the popularity of geometric algorithms. Computers enable us to move from small instances of geometric problems (which are usually “easy” to solve) to much larger instances. Nowadays, researchers from different areas of applied sciences can analyze, process, compute, and visualize large instances of complicated geometric models representing (real) objects.

Examples of typical questions asked in computational geometry are:

- Given a set of line segments in a plane, report all intersections between any two line segments.
- What is the minimum area of a convex polygon that encloses a given set of points?

Computational geometry has many application areas that demand efficient geometric algorithms and data structures, such as Computer-Aided Design (CAD), Computer Graphics, Computer Vision, Shape Recognition, and many more [3]. Initially, the combination of the study of movement and computational geometry arose in Robotics, which results in various motion planning algorithms and kinetic data structures. Consequently, analysis of the movement of moving entities has also become an essential topic in computational geometry.

1.1.2 Geographic Information Systems and Science

Geographic Information Systems (GIS) are computer-based tools to improve the efficiency and effectiveness of handling information about objects and events located in geographic space by collecting, storing, processing, analyzing, and visualizing geographic information [4]. There are many application fields where GIS play an essential role, for example, urban planning, agriculture, tourism, defense, transportation, and many more.

While GIS focuses more on tools and technologies to solve problems involving geographic information, Geographic Information Science (GISc) focuses on the scientific knowledge behind the GIS technologies that will still be valid many years from now. Geographic Information Science is a multi-disciplinary field, with research contributions from geography, geodesy, geostatistics, cartography, and computer science.

Since geographic information contains spatial data related to space (e.g., location, shape), many computational problems in GIS (and thus, GISc) are geometric by nature. Therefore, computational geometry plays an integral part in the research in GISc. Problems in GISc where computational geometry can provide (part of) a solution include, for example, data structures for spatial data (e.g., R-Tree), map overlay, map generalization, map visualization, spatial queries, spatial interpolation, label placement, and many others [5].

Along with its spatial components, geographic information can also have various non-spatial parts, such as a temporal component (i.e., time). These components enable the geographic information to incorporate changes over time because sometimes it is necessary to record the changes as well, rather than maintain only the most recent data. For example, in transportation management, the position of vehicles may change continuously. Not only their starting and ending position but also their movement over time is an essential factor for planning and managing transportation systems. However, the spatial component of the geographic information is not necessarily influenced by the change in its temporal component. Consider the case in urban and regional planning where the use of an area may change from industrial to residential at a specific time, but the location of the area itself does not change [6, 7].

Many challenging real-world problems can be solved if we have efficient methods to extract knowledge from movement data of moving entities. For example, understanding the movement pattern of a hurricane could be useful for a better natural disaster management. For ecologists, knowledge about the collective motion of animals might give helpful insight to understand their movement behaviors. Thus, analysis of movement by entities has gradually become a significant research field in GISc.

1.1.3 Computational Movement Analysis

Although observations of various moving entities have been carried out for a long time, previous analysis of movement data at that time often produced unsatisfactory results mainly because of two reasons [8, 9]:

- Movement data was difficult to gather because of limited technology. It was collected in relatively small numbers and had poor quality in terms of sampling rate and spatial precision.
- In-depth analysis of those data was performed manually because no computational methods were available and those manual processes were feasible only for small numbers of data.

However, the presence of devices equipped with advanced tracking technologies and sensors (e.g., GPS-enabled mobile phones, RFID tags), other systems based on the Global Navigation Satellite Systems (GNSS), and video surveillance/tracking analysis, make it possible to easily record the locations and track the movement of an entity over a period of time.

The widespread use of such inexpensive devices by various types of moving entities (e.g., people, animals, vehicles) leads to the availability of a vast amount of high-quality movement data (i.e., high sampling rate and precise geographical location). Consequently, this gives rise to an increasing interest to analyze them and open new opportunities to develop useful applications in many research fields. In recent years, *Computational Movement Analysis* (CMA) has emerged as a mainstream research domain within the GISc community. It focuses on developing and applying computational tools for analyzing movement data.

Laube [10] defines Computational Movement Analysis as *the interdisciplinary research field studying the development and application of computational techniques for capturing, processing, managing, structuring, and ultimately analyzing data describing movement phenomena, both in geographic and abstract spaces, aiming for a better understanding of the processes governing that movement.*

From the practical point of view, CMA can be considered as a bridge to connect the low-level tracking data and the high-level understanding of the semantics of movement in various research fields. Examples of application domains where movement analysis is a crucial part of its research are: animal movement and behavior, traffic and transportation, defense and surveillance, oceanography, disaster management, people behavior, health management, sports science, and many others [8, 10].

1.2 Trajectories

Movement data of a moving entity is typically described as a *trajectory*: a path made by a moving entity as it travels through space over a period of time.

Abstract and Data Model. Since a trajectory is the model of a moving entity, we can use either the *abstract model* or the *data model* of a trajectory. In the *abstract model*, a trajectory is the representation of a moving entity, assumed to be a point that moves without discontinuities. With this model, the spatial position of the moving entity can be located at any moment in time. Thus, a trajectory can be regarded as a function that maps a time

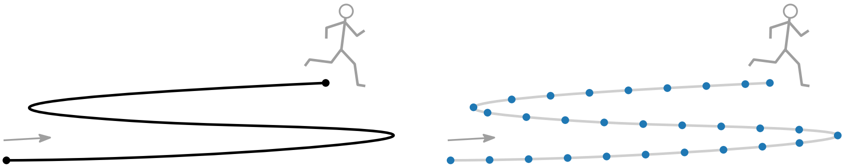


FIGURE 1.2: (Left) A path of an abstract model of a trajectory. (Right) In the data model, trajectory consist of recorded locations of the moving entity, which are ordered by time.

interval to the space in which the entity is moving. We define the path of a trajectory as the image of the function, which has a shape and annotated with a direction but has no time component.

Movement data is generally collected by tracking devices that give spatial information at a certain time when the location is sampled. Even though the movement is often continuous, due to the limitation of the technologies, tracking devices usually report the locations only at specific moments with regular or irregular time intervals in between. Hence, in the *data model*, a trajectory is a time-ordered sequence of locations where the position of the moving entity was recorded. Here, the geometric precision and the sampling rate are the two most important aspects of data quality for trajectories that exist in the data model but not in the abstract model. Figure 1.2 illustrates the two models of a trajectory.

Movement Space. There are various types of movement space in which the entity could be moving. We consider the space where entities move based on the type of entities being observed and the type of applications that use the movement data. When the height or depth of the positions is not important, we usually use the Euclidean plane $\mathbb{R}^2$. For instance, in sports analysis, the z-coordinate of soccer players in the field probably is insignificant for researchers who want to analyze their movement. However, if we want to observe the movement of migrating birds or fish in the ocean (where location includes altitude or depth), then we want to consider $\mathbb{R}^3$ as the movement space. Moreover, other types of specific space may also be used to analyze movement data. For example, researchers hypothesize a structure of political mentality (political left, liberal, and conservative) in a three-dimensional space to analyze referendum data [11].

Lagrangian and Eulerian Perspective. There are two different approaches to obtain trajectory data for the data model [10]. From the *Lagrangian* perspective, the trajectory data is determined by tracking the position of entities as they move in time. Thus, it is useful if we want to track the

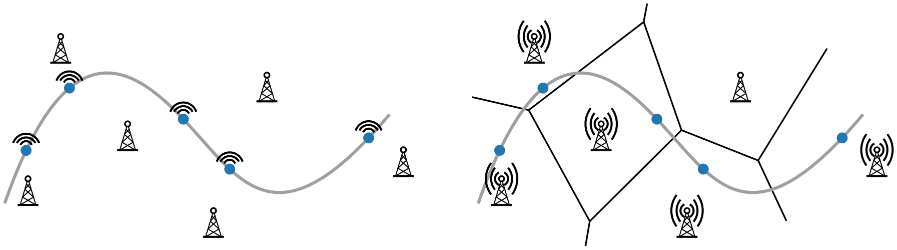


FIGURE 1.3: (Left) The Lagrangian perspective of movement: movement data is collected while the blue entity moves along the gray path. The information of the recorded locations is taken from the tracking devices attached to the entity. (Right) The Eulerian perspective: the stationary GSM towers track the movement of the entity.

movement detail of an entity. The use of GPS collars on animals is a typical example of the practical acquisition of movement data using this approach. While the Lagrangian viewpoint is an entity-based, the *Eulerian* perspective is location-based. It focuses on the movement changes by an entity which is observed at a known fixed location using technologies like RFID tag, WiFi, or GSM (through mobile phone cell towers). Figure 1.3 shows the difference between the two views.

Trajectory Interpolation. Although collecting trajectory data using the Lagrangian approach makes it possible to sample them at finer spatial resolutions (e.g., by sampling the position every second), the advantage of having such trajectory data still depends on the scale that we use in the analysis or applications. When we use a larger scale, we might still not know where the moving entity is at times in between the samples. Nevertheless, since the original movement is continuous, we may want to analyze and process the (discrete) trajectory data in its continuous form. There are various approaches to address this issue.

First, one might choose to ignore this problem and analyze the movement only at times when the location is known. This approach has at least one advantage: it usually leads to a simple and fast algorithm to process the movement data. However, if the trajectory data is not densely sampled, then we could interpolate the time and locations between consecutive recorded locations.

The simplest assumption for the location of a moving entity at any time between two consecutive sampled locations is to perform a linear interpolation between them. In this interpolation, we assume that the moving entity moves with constant speed along a straight line connecting the two locations. If the movement data is sampled sufficiently often, we can further assume that the linear interpolation does not induce a significant

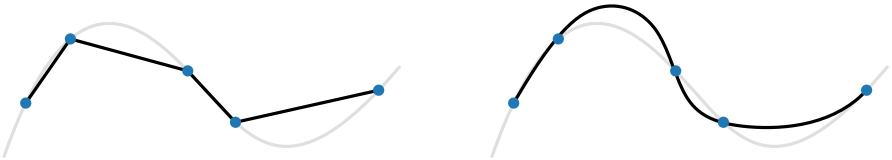


FIGURE 1.4: The actual trajectory is shown in gray. Different Interpolations for a sequence of time-stamped locations (blue discs). (Left) Linear interpolation: the entity is assumed to move along a straight line between two blue discs with a constant speed. (Right) An example of a non-linear interpolation.

error. Due to rather efficient computation of trajectories with this type of interpolation, this model is very common in GIS, Computational Geometry, and other domains.

Sometimes, a linear interpolation is not realistic in a particular scenario. Therefore, a more complicated shape (e.g., curve) is needed and probably more suitable to connect the two consecutive sampled locations. For instance, Tremblay et al. [12] suggest that curvilinear interpolations (Bézier curve, Hermite, and cubic spline) are more advantageous than linear interpolation for trajectories of entities that move in a fluid environment. Other non-linear interpolations include the constrained random-walk [13], kinematic interpolation that incorporates object kinematics (i.e., velocity and acceleration) [14] and the Brownian motion to estimate the movement path [15, 16, 17].

1.2.1 Notation

Throughout this thesis, we use $\mathcal{T}$ to denote the trajectory of an entity. Recall that in the abstract model, a trajectory $\mathcal{T}$ is a function that maps a time interval $\mathcal{I} = [t_\alpha, t_\beta]$ to space. The location at the start time t_α (or t_{start}) is the *origin* of $\mathcal{T}$ and the location at the end time t_β (or t_{end}) is the *destination* of $\mathcal{T}$.

Furthermore, we only consider the movement in $\mathbb{R}^2$ and study trajectory data from the Lagrangian perspective. Therefore, in the data model, the trajectory $\mathcal{T}$ is a sequence of time-stamped locations $(p_1, t_1), (p_2, t_2), \ldots, (p_\tau, t_\tau)$ where $p_i = (x_i, y_i)$ denotes the position of the entity at timestamp t_i and τ denotes the total number of recorded data points.

We use linear interpolation between two consecutive data points to make a continuous mapping from time to space. Hence, a trajectory $\mathcal{T}$ now is a (continuous) piecewise linear function mapping time to space. We refer to the end-points of those pieces, which are the recorded data points, as the *vertices* of a trajectory.

For most parts of this thesis, instead of analyzing only one trajectory, we study a collection of trajectories. Let $\mathcal{X}$ be a set of moving entities whose trajectories are tracked and n be the number of entities in $\mathcal{X}$. We denote $\mathrm{d}_{ij}(t)$ as the Euclidean distance between two entities $i \in \mathcal{X}$ and $j \in \mathcal{X}$ at time t. Note that we may have differently sampled trajectories: their vertices are not aligned in time.

1.2.2 Trajectory Data

Trajectory data comes from various moving entities, for example, vehicles, animals, and pedestrians. There are diverse technologies and methods to acquire their trajectory data. The quality of the acquired trajectory data depends on several factors such as the type of entities, the environment where the movement is observed, the technology used to record the locations, and many more.

There are a few methods that do not use modern technology involving electronic devices. For instance: the thread trailing method to follow trajectories of box turtles [18, 19], manual marking by humans following mangabey monkeys [20], and the use of a colored band/ring attached to the leg of blue cranes [21]. One main advantage of these methods is that they do not require any power source such as batteries. However, these methods have several disadvantages, such as the low precision on the recorded locations, the long intervals between recorded data, and only a small number of trajectory data can be collected.

Nowadays, electronic devices are used to obtain fine-grained trajectory data on a large scale for a more extended period of time. An example is the use of VHF (Very High Frequency) Radio-Telemetry to track the movement of brown bears [22]. The Starkey project uses automated radio-telemetry systems to track and collect movement data of elks and deer for three years [23].

More advanced technologies include satellite-based tracking technologies like the Argos-Doppler systems and the more popular GPS (Global Positioning System). The main difference between the two is in the type of device carried by the moving entity. In the Argos Doppler systems, the moving entity is equipped with the transmitter, while in the GPS, the entity carries the receiver. The Argos Doppler systems can be used better than GPS if the entity (usually animals) spends time under water, for example, to track the movement of whales in the Atlantic sea [24]. In recent years, the use of GPS has increasing and gradually become a standard to track various types of moving entities, not only for animals but also for vehicles [25, 26] and humans [27, 28].

Other specific types of vehicles like ships use the Satellite-AIS (Automatic Identification Systems) in conjunction with the marine radar. The resulting trajectory data can be used to predict the arrival time of a vessel in the future [29]. An example of AIS trajectory data available for download and use freely for research purposes is provided by the Danish Maritime Authority [30]. Furthermore, trajectories can also be constructed from satellite imagery, for example, to track hurricane trajectories [31]. One of the primary sources for animal movement data collected using various tracking systems is the Movebank database [32].

When the size of the environment is limited (e.g., inside a building or in a hallway), it is challenging to use previously listed technologies, which are usually intended to cover a larger area. Therefore, the use of devices with a close (or medium) range capability, such as sensors and Bluetooth, are preferable. Static 3D range sensors can be installed inside a building and then used to track the movement of humans inside the building, for example, to track trajectories of visitors inside a shopping mall [33, 34, 35]. Small sensors can also be attached to small animals (e.g., mice, turtles) inside a small monitored environment to acquire their trajectories [36]. Nevertheless, limited range tracking devices can also be useful to extract movement data with the Eulerian specification (see Section 1.2) in a larger area. For example, Michau et al. [37] proposed a model to reconstruct vehicular trajectories collected from Bluetooth detectors scattered throughout road networks.

Now, the acquisition of real-world movement data using technologies we described above depends on various aspects such as weather, battery power, and signal strength. Therefore, the recorded trajectory data might not have uniform time intervals in between consecutive recorded locations. A similar problem also arises if we want to acquire trajectories from a set of moving entities; the interval of recorded time might differ from one entity to another.

Sometimes, tracking devices are difficult to install in some environments (e.g., fluid) or to attach on the moving entities (e.g., fish). In this case, we can record the movement using video recording systems such as surveillance-type videos. Then, the trajectories of the observed entities can be extracted from the resulting videos using VCA (video content analysis) and video tracking algorithms. Some examples of trajectory data obtained using this method are trajectories of fish in the ocean [38, 39] and also inside a tank [40]. This method is also very common to get trajectories of pedestrians, either outside [41] or inside [42] a building.

Finally, if it is challenging to obtain trajectory data using any of the previous methods, then we can model the movement of moving entities. Based on the model, we can create computer programs and run simulations on computers to generate trajectories. From the simulations, we obtain *artificial* trajectory data. Although modeling the movement of entities is a complicated process, this method also has advantages over other methods. We can adjust important things related to the simulation: the duration of the simulation, the size and the type of the environment (and its obstacles), the number of moving entities, as well as various features of the moving entities (e.g., speed, behavior). Furthermore, we can produce a large amount of trajectory data of the modeled entity easily.

Huth and Wissel [43] tried to model the movement of fish and then compare it against the actual movement observed using a video camera. The agent-based modelling software Netlogo [44] provides a simulation model that attempts to mimic the movement of a flock of birds or school of fish [45, 46, 44]. There have been extensive works from various research fields to create simulations of pedestrians in crowds [47]. Moreover, various models have been used to simulate crowds: flow-based [48], ruled-based [49], and agent-based models [50, 51]. Examples of crowd simulation software in which we can record the trajectories of the (artificial) moving entities during simulations are PEDSIM [52] and the framework developed at Utrecht University [53, 54].

1.3 Trajectory Analysis

A number of methods have been developed in recent years to analyze trajectory data. Even though movement data is used in various applications, those methods usually perform generic and fundamental tasks. We present an overview of several important analysis tasks: segmentation, similarity determination, clustering, representation, and various patterns that may emerge from the movement of the entities. We discuss one of the pattern in more detail: the *collective movement pattern*, which is the main topic in this thesis.

1.3.1 Examples of Analysis Tasks on Trajectory Data

Segmentation. Trajectory segmentation is the process to partition or divide a trajectory into several sub-trajectories such that within each sub-trajectory, certain properties are uniform and satisfy a given criterion [55, 56]. The criterion used in the segmentation is usually based on typical attributes of a trajectory such as speed, acceleration, direction/heading,

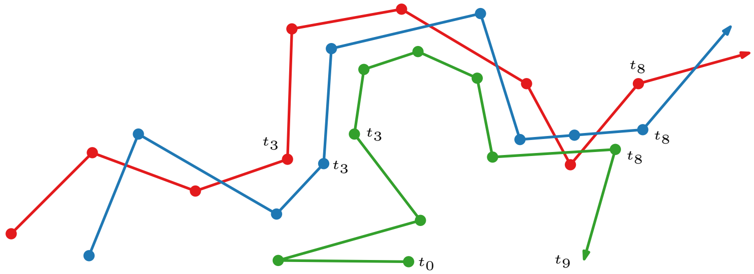


FIGURE 1.5: The blue and red trajectories are considered to be similar. For the green trajectory, only the sub-trajectory from t_3 to t_8 is similar to the other two within the same time interval.

curvature, or a combination of those attributes [57, 58]. Furthermore, a trajectory can also be divided based on its contextual spatial information [59].

Segmentation may also be used to split a trajectory into sub-trajectories with different movement behavior. In animal ecology, trajectory segmentation is useful for the detection of activities or behavioral states (e.g., resting, foraging, migration) of animals [60, 61]. In transportation related studies, trajectory segmentation is combined with machine learning to detect unusual behavior from travelers [62]. Furthermore, the trajectory of a traveler can be partitioned based on the transportation mode they used using a change-point based segmentation [63].

Similarity. Another important analysis task on trajectory data is similarity analysis, to determine whether two trajectories appear alike or not. The reasons for appearing alike can be varied. We could define similarity based on visiting the same locations, or based on having the same speed development (e.g., slow in the beginning, then linearly increasing speed). Similarity analysis can be used not only to determine how similar the (part of) trajectories of two distinct entities is [64], but also to find similar sub-trajectories from a trajectory of a single entity [65]. Usually, similarity analysis is used as a preprocessing or subroutine for other analysis tasks such as clustering or finding movement patterns.

The similarity between the two trajectories sometimes is understood as the distance between them. We present an example in Figure 1.5. There are various distance measures that can be applied to determine trajectory similarity for different applications. If we only consider the path of the trajectory, then we can use the Euclidean distance (in combination with Principal Component Analysis) [66], the Hausdorff distance [67, 68], or the Fréchet distance [68]. Other measures incorporate the temporal aspect of trajectories. These include Dynamic Time Warping [69], time-focused distance [70], edit distance [71], and the Longest Common Subsequence [72]. Additional attributes of movement such as speed, acceleration, and direction can also be used to determine trajectory similarity [73]. Furthermore, those attributes can be combined with other contextual data (e.g., land-cover, temperature) [74].

Finding similar trajectories is useful for diverse purposes. For instance, to find pedestrians with similar movement behaviors [75] or to estimate the migration route of animals [16]. Moreover, trajectory similarity is often used in video surveillance analysis [76] and sign language retrieval [77].

Clustering. Clustering is a procedure to partition objects into several clusters. Objects in the same clusters are similar to one another but dissimilar to objects in other clusters. Clustering can be applied to trajectories from various moving entities [78, 79], including in veterinary research to analyze the trajectories of animals [80]. We can cluster trajectories based on the similarity between trajectories or sub-trajectories [70, 75, 76]. Furthermore, trajectory clustering can also be used to detect the occurrence of a specific movement pattern (e.g., commuting pattern) [65, 81].

In general, clustering methods (on any type of data) can be classified into five methods: partitioning-based, hierarchical-based, density-based, grid-based, and more advanced model-based [82]. Therefore, different results on trajectory clustering can also be based on those methods. For instance, Wu et al. [83] use a partitioning-based clustering to study the movement styles of entities at different local regions due to geographic nature. To visually analyze the migration patterns of gulls, Konzack et al. [84] integrate their method with a hierarchical clustering algorithm. The density-based clustering is adapted by Nanni and Pedreschi [70] to cluster trajectories that consider the temporal component to improve the quality of the results. Mao et al. [85] develop a trajectory clustering algorithm that combines the grid-based and density-based method to realize adaptive parameter calibration. These parameter values have a great influence on the effect of clustering. Finally, Gaffney et al. [86] propose (probabilistic) model-based clustering to observe the behavior of cyclones based on their trajectory tracks.

Representation. From a cluster of similar trajectories, one might want to compute a suitable representative trajectory for that cluster. This trajectory could represent a typical route that was taken by ships in one region or describes the overall movements of a group of animals traveling together. In Figure 1.6, we illustrate some examples of a possible representative trajectory of the same cluster.It could be a single trajectory (from that particular cluster), a trajectory consisting of sub-trajectories from trajectories

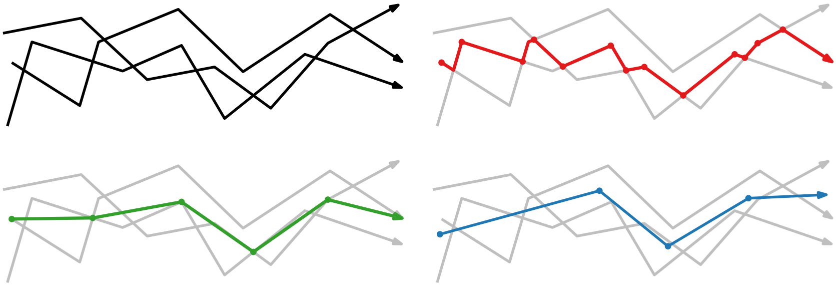


FIGURE 1.6: Examples of representative trajectories from a set of three black trajectories (top-left). The representative trajectory uses only the vertices of trajectories (in green, bottom-left), or must use parts of trajectories in the set (in red, top-right), or a completely new trajectory (in blue, bottom-right).

in the same cluster, a whole new trajectory, or a trajectory which has some parts made from pieces of trajectories in the cluster and the other parts are new (e.g., Figure 1.6, bottom-left). Nevertheless, the representative trajectory is required to be similar or close (for a given distance function) to all trajectories in a cluster.

Finding a representative trajectory is useful for several reasons. Firstly, we can reduce the amount of trajectory data to be analyzed since the representative trajectory already represents other similar trajectories. Secondly, when we want to analyze a set of trajectories visually, the representative trajectory offers better visualization since we can focus on only one or a few trajectories. Thirdly, we can use it to detect an outlier, which can be done by analyzing the similarity or the closeness between the representative trajectory and other trajectories. Finally, in the k-medoids and k-means clustering algorithm, representative trajectories can act as central objects, which are chosen before the clusters are made. Then, other objects are distributed into different clusters based on their similarity with each central object.

Lee et al. [87] proposed a simple sweep algorithm to efficiently discover a representative trajectory that describes the overall movement of trajectories from the same cluster. It is an imaginary trajectory consisting of a sequence of "average coordinate" points with respect to vertices (with the same timestamps) from the trajectories in the cluster. Therefore, the vertices of this "mean" trajectory usually do not coincide with the vertices of the input trajectories. On the contrary, Buchin et al. [88] introduce the concept of median trajectory that only uses pieces of trajectories from a given set. However, it does not consider the time information of the trajectories. Van Kreveld et al. [89] incorporate the temporal component and propose another type of representative: the central trajectory. It also uses pieces of the input trajectories but is allowed to "jump" and switches to another piece if they are within a small distance.

Several examples show that the concept of representative trajectory might be useful in different applications. For instance, it can be used to analyze the trajectory of an aircraft during take-off and landing [90]. Buchin et al. [91] use it in map reconstruction and Jiang et al. [92] use it to simplify a vast number of trajectories that do not fit into the computer memory for further processing.

1.3.2 Movement Patterns

Movement patterns of an entity refer to the interesting characteristics recognized from its trajectory [93, 94, 95]. For multiple moving entities, it usually refers to interesting interactions or relationships among them. Since various types of moving entities could have different types of relationships and characteristics, relevant patterns usually depend on the respective domain. For example, the movement patterns of visitors in a national park may provide useful information about interesting places that are regularly visited. In animal ecology, analyzing the movement patterns of groups of animals may help ecologists to understand the correlation between animal behaviors in different activities (e.g., foraging, mating) and the types of locations where the activities happened.

A Single Trajectory. From an individual entity, we can retrieve some interesting patterns based on its trajectory. The periodic pattern [96, 97] occurs when the entity shows the same spatio-temporal pattern with some periodicity (e.g., visit an area with some regularity). We can also identify popular places that are frequently visited by looking at the trajectory pattern of an entity [98, 99]. Furthermore, based on the history of the previous movement pattern, we can predict the next location to be visited by an entity [100]. Other examples of movement patterns for a single trajectory include commuting [65] and concentration patterns [93].

Multiple Trajectories. For a pair of trajectories, we can determine some particular movement patterns like chasing behavior (e.g., between predator and its prey) [101, 102] and an avoidance movement [103]. For a set of moving entities, a leadership [104, 105] pattern occurs when one entity acts as a leader and its movement is followed by others; for example, the movement of a herd of wild horses. The more specific type of leadership pattern is the single file movement [106] in which each entity is following another in a row. Other movement patterns involving multiple trajectories are meeting [107], convergence [107], and many more. We show a few examples of movement patterns in Figure 1.7.

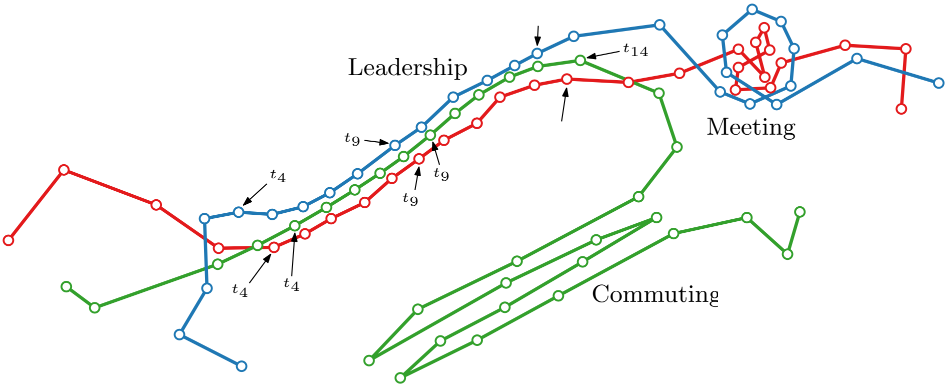


FIGURE 1.7: Three trajectories show various types of patterns: meeting, commuting, and leadership. From t_4 to t_{14}, the green entity acts as a leader and is followed by the other two entities, recognized by the timestamps on the trajectories.

In this thesis, we mostly study the *collective movement pattern*. It is a particular type of pattern which is determined by the spatial proximity of multiple moving entities over a period of time. The Leadership pattern in Figure 1.7 can be seen as a specialization of a collective movement pattern.

1.3.3 Collective Movement

One particular type of pattern that has been studied in various ways is the collective movement pattern, which occurs when multiple entities travel together (or simply are together) during a period of time. Throughout the thesis, we also use the term *group* interchangeably with *collective movement*. This pattern is related to clustering, but in clustering, we generally consider the whole trajectory when making clusters (although sub-trajectory clustering also exists), whereas in grouping, a single entity can be in different groups at different times, or even at the same time.

Identifying this type of pattern is particularly relevant in many applications, for example, to understand the behavior of groups of animals like brown lemur [108]. In veterinary science, researchers investigate whether the composition of animals in a group depends on the health level of its members [109, 80]. In social psychology, the collective movement can be identified and analyzed within human crowds and can be useful in behavioral studies [110]. In all these areas (and many more), identifying collective movement in trajectory data can provide critical new insights to understand the movement of moving entities.

There are various aspects that need to be considered when one wants to model collective movement:

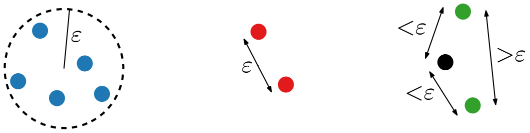


FIGURE 1.8: Different types of spatial proximity: (Left) All blue entities are contained in a disc of radius ε. (Centre) The distance between two close entities must be $\leq \varepsilon$. (Right) The two green entities are considered to be close using the black entity as an intermediate.

- *Input trajectories*
  Recall that the movement data is collected as discrete trajectories (a sequence of time-stamped locations) and obtaining its continuous version can be done using interpolation. If the model of a collective movement only considers the discrete version, then the starting and ending time of a group must coincide with the recorded times. On the other hand, if we use the continuous version, we can estimate the time and position of entities between known locations. Therefore, the same group might have a longer time interval because of the differences in the starting and/or ending time.
- *Spatial Proximity*
  Usually, we consider entities that travel together to be spatially close to one another. A straightforward approach to define closeness between two entities is to set a single value $\varepsilon > 0$ as the maximum required distance between them to be considered close. We can easily extend this requirement for the spatial requirement of a group: every pair of entities of a group must be close to each other at any time during the duration of the group.
  There are other ways to define closeness between entities in a group. For example, Gudmundsson and van Kreveld [111] use a disc of radius ε and require that entities in the same group are contained in it.
  The maximum required distance between two entities can also be used differently. Instead of considering the distance between two entities directly, we can use another entity as an intermediate: two entities i and j are close to each other at time t although $\mathrm{d}_{ij}(t) > \varepsilon$ as long as there is another entity k such that both $\mathrm{d}_{ik}(t) \leq \varepsilon$ and $\mathrm{d}_{kj}(t) \leq \varepsilon$. See Figure 1.8.
- *Size*
  Obviously, we can consider two entities moving together as a group. However, depending on the application, probably one wants to set a different value as the minimum required size (the number of entities) to form a group, or even limit the size of the group. Furthermore, at any particular moment, one can decide whether a single entity should only be part of one group, or could be a member of multiple groups.

- *Temporal Component*
  The temporal component refers mainly to the minimum time needed for entities to stay together to be viewed as a group. One important thing about the duration of a group is its continuity.
  Consider a case when members of a group are separated for a moment and then they join again to form the same group. It could be that each separate time interval, before and after the split, does not fulfill the time requirement for a group, but their cumulative time does. Therefore, allowing discontinuity on the temporal requirement might give a more robust definition of a group as it is not sensitive to short interruptions in proximity.

There are various models of collective movement with slightly different definitions. One of these definitions is flocks [111, 107, 112, 113]. Other names for closely related concepts are moving micro-clusters [114], moving clusters [115, 116], mobile groups [117], herds [118], convoys [81, 119, 120], swarms [121, 122], traveling companions [123], gatherings [124], platoons [125], and groups [126, 127].

All definitions depend on at least three parameters (there could be more): the size, the temporal parameter, and the spatial parameter. Note that while most of the definitions above consider the discrete version of a trajectory, there are also models designed to handle continuous trajectories (e.g., groups [126]).

1.4 Outline of the Thesis

The remainder of this thesis consists of several chapters. In the next chapter, we study the problem of simplifying a single polygonal line (henceforth *polyline*) under two well-known distance measures. A polyline can be seen as the image of a continuous piecewise-linear function of a trajectory (see Section 1.2.1); it is a connected sequence of line segments specified by a sequence of points.

The next three chapters cover different aspects of a collective movement: various measures for a collective movement, a new definition for a collective movement and algorithms to compute it, and experiments to compare different definitions of collective movement using various types of trajectory data sets. We conclude the thesis in the final chapter by providing an overview of our works from each chapter. Furthermore, we also discuss possible future research related to the collective movement of moving entities.

Chapter 2: Polyline Simplification

One crucial pre-processing step in trajectory analysis is the reduction of the number of vertices of a trajectory. This procedure is known as trajectory simplification. Trajectory simplification is important for at least two reasons: to reduce the use of computer resources (e.g., memory, computational power, storage) and to remove irrelevant or recurring information in a trajectory so that the analysis can be performed faster.

Andrienko et al. [9] suggest five possible approaches to trajectory simplifications: (i) geometry-based: convey the shape of the trajectory using a minimum number of vertices, (ii) density-based: use a single vertex to simplify consecutive vertices concentrated in one area (e.g., [128]), (iii) place-based: remove vertices located in the area that will not be included in the subsequent analysis (e.g., [129]), (iv) event-based: preserve only vertices when certain movements of events such as stop events occurred (e.g., [130]), and (v) attribute-based: remove consecutive vertices having similar values of some thematic attributes (e.g., [131]).

Cao et al. [132] and Gudmundsson et al. [133] show that geometry-based trajectory simplification can be done using a modification of the Douglas-Peucker algorithm [134]. It is a well-known algorithm to solve (poly)line simplification problems in computer graphics and cartography. Recall that a polyline is the image of the continuous piecewise linear function of a trajectory. Hence, a polyline only contains spatial information but does not have any temporal information. In this chapter, we shift our focus from trajectories to polylines and revisit the classical polyline simplification problem and study it using the Hausdorff distance and (continuous) Fréchet distance.

The line simplification problem takes a maximum allowed error ε and a polyline $\mathcal{P}$ defined by a sequence of points $\langle p_1, \ldots, p_n \rangle$, and computes a polyline $\mathcal{Q}$ defined by $\langle q_1, \ldots, q_k \rangle$ for which the error is at most ε in some error metric. Commonly the sequence of points defining $\mathcal{Q}$ is a subsequence of points defining $\mathcal{P}$, and furthermore, $q_1 = p_1$ and $q_k = p_n$.

There are many ways to measure the distance or error of a simplification. Among the distance measures for two shapes that are used in computational geometry, the *Hausdorff distance* and the *Fréchet distance* are probably the most well-known. They are both *bottleneck measures*, meaning that the distance is typically determined by a small subset of the input like a single pair of points (and the distances are not aggregated over the whole shapes). The Fréchet distance is considered a better distance measure, but it is considerably more difficult to compute because it requires us to optimize over all parametrizations of the two shapes [68]. The Hausdorff distance

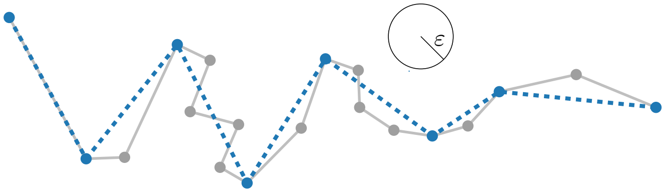


FIGURE 1.9: The input trajectory is shown in gray. The simplified trajectory (in blue) is the result of the Douglas-Peucker simplification algorithm. All vertices of the input trajectory have a distance $\leq \varepsilon$ to the blue trajectory.

between two simple polylines with n and m vertices can be computed in $O((n+m)\log(n+m))$ time [135]. Their Fréchet distance can be computed in $O(nm\log(n+m))$ time [68]. Note that the Fréchet distance is symmetric, whereas the Hausdorff distance has a symmetric and an asymmetric version (in term of the distance from the input to the simplification).

Among many polyline simplification algorithms, the ones by Douglas and Peucker [134] (see Figure 1.9) and by Imai and Iri [136] hold a special place and are frequently implemented and cited. There are many other results in line simplification. Different error measures can be used [137], self-intersections may be avoided [138], line simplification can be studied in the streaming model [139], it can be studied for 3-dimensional polylines [140], angle constraints may be put on consecutive segments [141], there are versions that do not output a subset of the input points but other well-chosen points [142], and it can be incorporated in subdivision simplification [143, 144, 142]. Some optimization versions are NP-hard [143, 142].

Interestingly, no previous authors studied line simplification under the Hausdorff distance and Fréchet distance in its pure form, namely: *for a given $\varepsilon > 0$, choose a minimum size subsequence of the vertices of the input such that the Hausdorff or Fréchet distance between the input and output polylines is at most ε*. We analyze how the well-known Douglas-Peucker and Imai-Iri simplification algorithms using both measures perform compared to the optimum possible.

Our results show that computing an optimal simplification using the undirected Hausdorff distance is NP-hard. The same holds when using the directed Hausdorff distance from the input to the output polyline, whereas the reverse can be computed in polynomial time.

Finally, to compute the optimal simplification from a polygonal line consisting of n vertices under the (continuous) Fréchet distance, we give an $O(kn^5)$ time algorithm that requires $O(kn^2)$ space, where k is the output complexity of the simplification. Very recently, Bringmann and Chaudhury [145] improved this result by giving an $O(n^3)$ time algorithm.

This chapter presents work that is published in:

[146] Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the Hausdorff and Fréchet Distance. In *Proc. of the 34th International Symposium on Computational Geometry, SoCG 2018*, pages 56:1–56:14, 2018.

Chapter 3: Measures for groups

All trajectory analysis types depend on measures defined on trajectories. In the overview of trajectory analysis in Section 1.3, we mentioned several attributes of a trajectory such as location, speed, and direction. These attributes can be defined in the abstract model and can be computed in the data model of a trajectory (see Section 1.2.1). In fact, an attribute like speed gives rise to multiple measures: average speed, variations of speed, total stationary time, etcetera.

In Chapter 3, we discuss measures for groups of trajectories. Many analysis tasks that apply to trajectories exist for groups of trajectories as well. For example, we can imagine segmentation of the whole group at once or doing a similarity analysis on two groups. To perform such analyses, we also need measures for the whole group. Groups are a natural unit of aggregation which can be analyzed at once and which can be visualized using a single stroke. With the ongoing trend of dealing with larger and larger collections of data, aggregation is one of the approaches to cope with the growth. Note that these measures are general measures for groups, and therefore, their definition does not depend on a specific model of groups (see Section 1.3.3).

We choose the abstract model of trajectory to define measures for trajectories since it is mathematically more clean and representation-independent. If needed, measures in an abstract model can be converted in various ways to measures in a data model.

We begin with an overview of measures that exist for a single trajectory. We classify them into three types: (i) measures for a single trajectory in isolation, (ii) measures for a single trajectory that require the presence of other trajectories (e.g., the centrality of a trajectory amidst other trajectories), and (iii) measures defined by combining trajectory data with data from other sources, such as the environment where the entities move. For each measure, we give the description, the unit and range of the measurement, where the measure is derived (from attributes or other measures), and related works on that particular measure.

Next, we proceed with extensions of measures for groups, while at the same time including new measures that do not exist for single trajectories like density and formation stability. We use the same three types of classification for measures for groups and use similar approaches to develop those measures. There are three views of treating a group when defining its measures: representative (consider only a single trajectory in the group), complete (consider all trajectories in the group), and area view (consider the area that the group occupied during movement). We will highlight such more general approaches because they can potentially be used for other measures needed in specific applications. Finally, we discuss several tasks: settlement selection, visualization, and segmentation, where measures on groups of trajectories are necessary.

This chapter presents work that is published in:

[147] Lionov Wiratma, Marc van Kreveld, and Maarten Löffler. On measures for groups of trajectories. In *Societal Geo-innovation - Selected Papers of the 20th AGILE Conference on Geographic Information Science*, pages 311–330, 2017.

Chapter 4: A Refined Definition for Groups

In Section 1.3.3, we describe the collective movement pattern and list different definitions for this pattern. One of those definitions is the one by Buchin et al. [126] called *groups*. We refine their definition for groups and argue that our refined definition corresponds better to human intuition in certain cases, particularly in dense environments.

Buchin et al. [126] introduce a model called the *trajectory grouping structure* which not only defines groups but also the splitting of a group into subgroups and its opposite, merging. Moreover, they analyze its mathematical structure and present efficient algorithms for computing all maximal groups[2] in a given set of trajectories. In their definition, a group is defined for both the continuous and the discrete model of a trajectory (see Section 1.2). Therefore, the algorithm to compute them considers times in between the timestamps where the locations are recorded as relevant. In between these timestamps, locations are inferred by linear interpolation over time. Thus, the computation allows trajectories to have recorded locations not collected at the same timestamps.

This grouping definition by Buchin et al. [126] relies on three parameters: one for the distance between entities, one for the duration of a group, and one for the size of a group. We propose a refined definition for groups and make a slight change in the condition for the distance parameter: the requirement for connectivity between two entities from the same group.

---

[2] A maximal group is a group that is maximal in size *and* maximal in duration.

Consequently, this change leads to different algorithms to compute the *refined groups*.

Let n denote the number of moving entities and τ the number of timestamps used for each trajectory. Hence, the input size is $\Theta(\tau n)$. Note that depending on the application, one of n or τ can be much larger than the other.

We present an $O(\tau^2 n^4)$ time algorithm to compute all maximal groups from a set of moving entities in $\mathbb{R}^1$, according to the refined definition. A similar approach applies if the timestamps of moving entities are not synchronous, at the cost of a small extra factor of α(n) in the running time, where α denotes the inverse Ackermann function.

In higher dimensions, we can compute all maximal groups in $O(\tau^2 n^5 \log n)$ time (for any constant number of dimensions), regardless of whether the timestamps of entities are the same or not. We model the moving entities and their connectivity as a graph and use an efficient method that allows $O(\log n)$ time per edge update [148] to maintain the changes of connected components over time. Furthermore, we also show that one τ factor can be traded for a much higher dependence on n by giving an $O(\tau n^4 2^n)$ time algorithm for the same problem. Consequently, we give a linear-time algorithm when the number of entities is constant and the input size relates to the number of timestamps of each entity.

Finally, we provide a construction to show that it might be challenging to develop an algorithm with polynomial dependence on n and linear dependence on τ.

This chapter presents work that is published in:

[149] Marc van Kreveld, Maarten Löffler, Frank Staals, and Lionov Wiratma. A refined definition for groups of moving entities and its computation. *International Journal of Computational Geometry & Applications.*, 28(2):181–196, 2018.

Chapter 5: Experimental Evaluation of Grouping Definitions

In this chapter, we present the results of an extensive experimental study to find small groups in pedestrian data using various data sets. Finding small groups is a crucial case in analyzing the throughput in public spaces like shopping malls, parks, and train stations, and in detecting suspect behavior in such spaces. Note that small groups can be as small as just two individuals.

We compare four definitions of groups, namely *convoys* [81] (which in our setting are the same as *traveling companions*), *swarms* [121], *groups* [126], and *refined groups* [149]. These four grouping definitions are different by one or more of three different characteristics: (i) *input*: how they model the input trajectories (as a continuous piecewise linear function or as discrete timestamps), (ii) *connectivity*: how they model when entities are considered together, and (iii) *duration*: how they measure if the entities are together long enough (it is measured cumulatively or as one contiguous time interval).

Convoys (traveling companions) are a well-known type that considers groups whose composition does not change, whereas togetherness is consecutive, and assessment is done at the timestamps themselves. Groups and refined groups distinguish themselves from the other definitions on a characteristic of their input trajectories; treating time as a continuous phenomenon may make a difference for patterns that consist of relatively few timestamps, which is the case in our pedestrian settings. The refined group definition is the only definition that measures togetherness within the group only, not having other non-group entities influence this. Swarms distinguish themselves by not requiring a contiguous grouping; interruptions are allowed.

Our study is not just an analysis of the four definitions, but also of how the input, space, and time can be treated and how this affects the results. Therefore, it may give indications to the results for other definitions. We note that it is not the objective to find the "best" definition since we typically do not have ground truth. For several of the data sets, we do have human annotations of groups, so we can compare the four definitions to the human annotations, which are by their nature subjective.

We consider the following research questions:

(1) How well do the four studied definitions of groups correspond with what humans consider a "group", and how do the characteristics mentioned (input, connectivity, and duration) influence this?
(2) How does the number of groups, as reported by the various definitions, depend on the density of the entities?
(3) How does the number of groups, as reported by the various definitions, depend on the sampling rate of the input trajectories?

To answer these questions, we perform both a quantitative and qualitative study.

We implement the algorithms to compute groups based on convoys, swarms, groups, and refined groups. We compare the outputs from all implementations, which is the same as comparing the various definitions of groups since the implementations follow the definitions exactly.

For the quantitative analysis, we compute and compare the number of reported groups and the precision, recall, and F1 scores of the various definitions with respect to human annotation. Furthermore, we compute how well the various definitions correspond to social formations used in crowd simulations; a behavior scheme used to generate synthetic movement data that represents a group of friends moving in a crowd [150]. For the qualitative analysis, we develop a novel visualization to show and compare groups in video footage showing the movement of the entities. In particular, our visualization allows easy comparison of the detected groups with human annotation or with any other group definition.

We conducted experiments using three different types of trajectory data sets:

- artificial trajectories generated by computer simulations;
- real-life trajectories of people moving in synthetic environments under laboratory conditions;
- real-life pedestrian trajectories extracted from video recording systems in a public area.

Note that our experimentation only focuses on the output and does not consider the running time or memory consumption by the implementations.

This chapter is based on the following publications:

[151] Lionov Wiratma, Maarten Löffler, and Frank Staals. An experimental comparison of two definitions for groups of moving entities (short paper). In *Proc. of the 10th International Conference on Geographic Information Science, GIScience 2018*, pages 64:1–64:6, 2018.

[152] Lionov Wiratma, Marc van Kreveld, Maarten Löffler, and Frank Staals. An experimental evaluation of grouping definitions for moving entities. In *Proc. of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019*, 2019. To appear.

2

POLYLINE SIMPLIFICATION

In this chapter, we study the classical polygonal line (polyline) simplification problem. Line simplification (polygonal approximation) is one of the oldest and best studied applied topics in computational geometry. It was and still is studied, for example, in shape analysis, in the context of computer graphics (after image to vector conversion), and in Geographic Information Science.

Let $\mathcal{P}$ be an input polyline specified by a sequence of points $\langle p_1, \ldots, p_n \rangle$ and $\mathcal{Q}$ be an output polyline, which is a subsequence of points $\in \mathcal{P}$ starting with p_1 and ending with p_n. We consider the line simplification problem using the *Hausdorff distance* [67, 68] and *(continuous) Fréchet distance* [68] in its pure form: given an input polyline $\mathcal{P}$ and $\varepsilon > 0$, compute an output polyline $\mathcal{Q}$—a minimum size subsequence of points $\in \mathcal{P}$—such that the Hausdorff or Fréchet distance between $\mathcal{P}$ and $\mathcal{Q}$ is at most ε.

Both the Hausdorff and Fréchet distance are *bottleneck measures* since a small subset of the input determines the distance (e.g., a pair of points) and the distances are not aggregated over the whole shapes. For two polylines with n vertices each, their Hausdorff distance can be computed in $O(n \log n)$ time [135] and their Fréchet distance can be computed in $O(n^2 \sqrt{\log n} (\log \log n)^{3/2})$ time [153]. Although the Fréchet distance takes more time to compute, it is considered a better distance measure [68].

Probably the two most well-known algorithms for polyline simplification are the ones by David Douglas and Thomas Peucker [134] and by Hiroshi Imai and Masao Iri [136]. Both algorithms take a constant $\varepsilon > 0$ and guarantee that the output polyline $\mathcal{Q}$ is within ε from the input polyline $\mathcal{P}$.

The Douglas-Peucker algorithm [134] is a simple and effective recursive procedure that keeps on adding vertices from $\mathcal{P}$ until the computed polyline lies within a prespecified distance ε. The procedure is a heuristic in several ways: it does not minimize the number of vertices in $\mathcal{Q}$ (although it performs well in practice) and it runs in $O(n \log n)$ time in practical cases (although in the worst case, it runs in $O(n^2)$ time). Hershberger and Snoeyink [154] overcame the worst-case running time bound by providing a worst-case $O(n \log n)$ time algorithm using techniques from computational geometry, in particular a type of dynamic convex hull.

The Imai-Iri algorithm [136] takes a different approach. It computes for every *link* $\overline{p_i p_j}$ with $i < j$ whether the sequence of vertices $\langle p_{i+1}, \ldots, p_{j-1} \rangle$, in between p_i and p_j, lie within distance ε to the segment $\overline{p_i p_j}$. In this case $\overline{p_i p_j}$ is a valid link that may be used in the output. The graph G that has all vertices $p_1, \ldots, p_n$ as nodes and all valid links as edges can then be constructed, and a minimum link path[1] from p_1 to p_n represents an optimal simplification. Brute-force, this algorithm runs in $O(n^3)$ time, but with the implementation of Chan and Chin [155] or Melkman and O'Rourke [156] it can be done in $O(n^2)$ time.

Now, the Imai-Iri algorithm is considered an optimal line simplification algorithm, because it minimizes the number of vertices in $\mathcal{Q}$, given the restriction that $\mathcal{Q}$ must be a subsequence of $\mathcal{P}$. But for what measure? It is not optimal for the Hausdorff distance, because there are simple examples (see Section 2.2) where a simplification with fewer vertices can be given that still have Hausdorff distance at most ε between $\mathcal{P}$ and $\mathcal{Q}$. This comes from the fact that the algorithm uses the Hausdorff distance between a link $\overline{p_i p_j}$ and the sub-polyline $\langle p_i, \ldots, p_j \rangle$. This is more local than the Hausdorff distance requires, and is more a Fréchet-type of criterion. But the line simplification produced by the Imai-Iri algorithm is also not optimal for the Fréchet distance. In particular, $\mathcal{P}$ and $\mathcal{Q}$ do not necessarily lie within Fréchet distance ε, because links are evaluated on their Hausdorff distance only.

The latter issue could easily be remedied: to accept links, we require the Fréchet distance between any link $\overline{p_i p_j}$ and the sub-polyline $\langle p_i, \ldots, p_j \rangle$ to be at most ε [157, 158]. This guarantees that the Fréchet distance between $\mathcal{P}$ and $\mathcal{Q}$ is at most ε. However, it does not yield the optimal simplification within Fréchet distance ε. Because of the nature of the Imai-Iri algorithm, *it requires us to match a vertex p_i in the input to the vertex p_i in the output in the parametrizations, if p_i is used in the output.* This restriction on the parametrizations considered limits the simplification in unnecessary ways. Agarwal et al. [159] refer to a simplification that uses the normal (unrestricted) Fréchet distance with error threshold ε as

---

[1] A path in G with a minimum number of edges. This term is commonly used in a polyline simplification problem [142]

TABLE 2.1: Algorithmic results

	Douglas-Peucker	Imai-Iri	Optimal
Hausdorff distance	$O(n \log n)$ [154]	$O(n^2)$ [155]	NP-hard *(our result)*
Fréchet distance	$O(n^2)$ *(easy)*	$O(n^3)$ [158]	$O(kn^5)$ *(our result)*

n: vertices in the input polyline; k: output complexity of the simplified polyline

a *weak ε-simplification under the Fréchet distance.*[2] They show that the Imai-Iri algorithm using the Fréchet distance gives a simplification with no more vertices than an optimal weak ($\varepsilon/4$)-simplification under the Fréchet distance, where the latter need not use the input vertices.

The discussion begs the following questions: How much worse do the known algorithms and their variations perform in theory, when compared to the optimal Hausdorff and Fréchet simplifications? What if the optimal Hausdorff and Fréchet simplifications use a smaller value than ε? As mentioned, Agarwal et al. [157] give a partial answer. How efficiently can the optimal Hausdorff simplification and the optimal Fréchet simplification be computed (when using the input vertices)?

Organization and results. In Section 2.1 we explain the Douglas-Peucker algorithm and its Fréchet variation; the Imai-Iri algorithm has been explained already. We also show with a small example that the optimal Hausdorff simplification has fewer vertices than the Douglas-Peucker output and the Imai-Iri output, and that the same holds true for the optimal Fréchet simplification with respect to the Fréchet variants.

In Section 2.2 we will analyze the four algorithms and their performance with respect to an optimal Hausdorff simplification or an optimal Fréchet simplification more extensively. In particular, we address the question how many more vertices the four algorithms need, and whether this remains the case when we use a larger value of ε but still compare to the optimization algorithms that use ε.

In Section 2.3 we consider both the directed and undirected Hausdorff distance to compute the optimal simplification. We show that only the simplification under the directed Hausdorff distance from the output to the input polyline can be computed in polynomial time, while the other two versions are NP-hard to compute. In Section 2.4 we show that the problem can be solved in polynomial time for the Fréchet distance.

---

[2] Weak refers to the situation that the vertices of the simplification can lie anywhere.

2.1 Preliminaries

We first describe the formal definitions of the Hausdorff and Fréchet distance.

Hausdorff Distance. For arbitrary bounded sets $A, B \subseteq \mathbb{R}^2$, the Hausdorff distance δ_H is defined as [68]:

$$\delta_H(A, B) = \max(\sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(a, b))$$

where d is the distance metric in the plane. In this chapter, we use the Euclidean distance. We refer to the two terms $\sup_{a \in A} \inf_{b \in B} d(a, b)$ and $\sup_{b \in B} \inf_{a \in A} d(a, b)$ as the *asymmetric* Hausdorff distance from A to B and from B to A, respectively.

Fréchet distance. Let a curve $f : [a, b]$ be a continuous mapping to an an arbitrary Euclidean vector space V with $a, b \in \mathbb{R}$ and $a < b$. A polygonal curve is a curve $P : [0, n] \to V$ with $n \in \mathbb{N}$, such that for each $i \in \{0, 1, ..., n-1\}$, the restriction of $\mathcal{P}$ to the interval $[i, i + 1]$ is affine, i.e. $P(i + \lambda) = (1 - \lambda)P(i) + \lambda P(i + 1)$ for all $\lambda \in [0, 1]$. Then, the Fréchet Distance δ_F between two curves $A : [a, a']$ and $B : [b, b']$ is defined as [68]:

$$\delta_F(A, B) := \inf_{\substack{\alpha[0,1] \to [a,a'] \\ \beta[0,1] \to [b,b']}} \max_{t \in [0,1]} \parallel A(\alpha(t)) - B(\beta(t)) \parallel$$

where α, β range over continuous and increasing functions with $\alpha(0) = a$, $\alpha(1) = a'$, $\beta(0) = b$ and $\beta(1) = b'$ only. Note that the Fréchet distance is symmetric.

The Douglas-Peucker algorithm for polyline simplification is a simple recursive procedure that works as follows. Let the line segment $\overline{p_1 p_n}$ be the first simplification. If all points of $\mathcal{P}$ lie within distance ε from this line segment, then we have found our simplification. Otherwise, let p_f be the furthest point from $\overline{p_1 p_n}$, add it to the simplification, and recursively simplify the polylines $\langle p_1, \ldots, p_f \rangle$ and $\langle p_f, \ldots, p_n \rangle$. Then merge their simplifications (remove the duplicate p_f). It is easy to see that the algorithm runs in $O(n^2)$ time, and also that one can expect a much better performance in practice. It is also straightforward to verify that polyline $\mathcal{P}$ has Hausdorff distance (symmetric and asymmetric) at most ε to the output. We denote this simplification by $DP_H(\mathcal{P}, \varepsilon)$, and will leave out the arguments $\mathcal{P}$ and/or ε if they are understood.

We can modify the algorithm to guarantee a Fréchet distance between $\mathcal{P}$ and its simplification of at most ε by testing whether the Fréchet distance between $\mathcal{P}$ and its simplification to a segment is at most ε. If not, we still choose the most distant point p_f to be added to the simplification (other choices are possible). This modification does not change the efficiency of the Douglas-Peucker algorithm asymptotically as deciding whether or not the Fréchet distance between a line segment and a polyline is at most ε can be done in linear time [68]. We denote this simplification by $\mathit{DP}_F(\mathcal{P}, \varepsilon)$.

Previously, we have already described the Imai-Iri algorithm. We refer to the resulting simplification as $\mathit{II}_H(\mathcal{P}, \varepsilon)$. It has a Hausdorff distance (symmetric and asymmetric) of at most ε. Similar to the Douglas-Peucker algorithm, the Imai-Iri algorithm can be modified for the Fréchet distance, leading to a simplification denoted by $\mathit{II}_F(\mathcal{P}, \varepsilon)$. Note that both $\mathit{II}_H(\mathcal{P}, \varepsilon)$ and $\mathit{II}_F(\mathcal{P}, \varepsilon)$ never have more vertices than $\mathit{DP}_H(\mathcal{P}, \varepsilon)$ and $\mathit{DP}_F(\mathcal{P}, \varepsilon)$, respectively.

We will denote the optimal simplification using the Hausdorff distance by $\mathit{OPT}_H(\mathcal{P}, \varepsilon)$, and the optimal simplification using the Fréchet distance by $\mathit{OPT}_F(\mathcal{P}, \varepsilon)$. In the case of Hausdorff distance, we require $\mathcal{P}$ to be within ε of its simplification, so we use the directed Hausdorff distance. Next, we will show a case where $\mathit{DP}_H(\mathcal{P})$ and $\mathit{II}_H(\mathcal{P})$—which are both equal to $\mathcal{P}$ itself—may use more vertices than $\mathit{OPT}_H(\mathcal{P})$. We also present a similar case with the Fréchet distance.

For the Hausdorff distance, let $P = \langle p_1, ..., p_7 \rangle = \langle (0,0), (0, 2\varepsilon - \mu), (2\varepsilon, 2\varepsilon - \mu), (4\varepsilon, -2\mu), (4\varepsilon, 2\varepsilon), (0, 2\varepsilon) \rangle$ with $\mu = \frac{1}{6}\varepsilon$ (see Figure 2.1). The Douglas-Peucker algorithm begins with the segment $\overline{p_1, p_7}$. Then, at each step of the algorithm, there is always at least one vertex that has $\delta_H > \varepsilon$ from the current simplified polyline until the algorithm stops and returns the final simplified polyline $\mathit{DP}_H(\mathcal{P}) = \mathcal{P}$. The simplification $\mathit{II}_H(\mathcal{P})$ is also equal to $\mathcal{P}$ since there is no valid *link* (shortcut) between two non-adjacent vertices that can be used in the Imai-Iri algorithm. The optimal simplification $\mathit{OPT}_H(\mathcal{P}) = \langle p_1, p_5, p_6, p_7 \rangle$ since part of $\overline{p_1, p_2}$, $\overline{p_2, p_3}$ and part of $\overline{p_3, p_4}$ have $\delta_H \leq \varepsilon$ to $\overline{p_6, p_7}$, while the remaining part of $\overline{p_1, p_2}$ and $\overline{p_3, p_4}$ has $\delta_H \leq \varepsilon$ to $\overline{p_5, p_6}$.

For the Fréchet distance, consider $\mathcal{P} = \langle p_1, ..., p_4 \rangle = \langle (-\sqrt{2}\varepsilon - \mu, 0 + \mu), (\sqrt{2}\varepsilon, 0), (0,0), (\sqrt{2}\varepsilon, -\sqrt{2}\varepsilon) \rangle$ with $\mu \ll \varepsilon$ (see Figure 2.2). All vertices except p_1 are located in the circle of radius ε and the three vertices make an isosceles triangle where its two sides of the same length are parallel to the x and y-axis. Now, $\mathit{DP}_F = \mathcal{P}$ since the Douglas-Peucker algorithm determines that $\delta_F(\overline{p_1, p_4}, p_2) > \varepsilon$ and $\delta_F(\overline{p_2, p_4}, p_3) > \varepsilon$ and subsequently add p_2 and p_3 to the simplified output. Similarly, $\mathit{II}_F = P$ since all possible links are not valid (including $\overline{p_1, p_3}$, because $\delta_F(\overline{p_1, p_3}, p_2) > \varepsilon$). However, the optimal

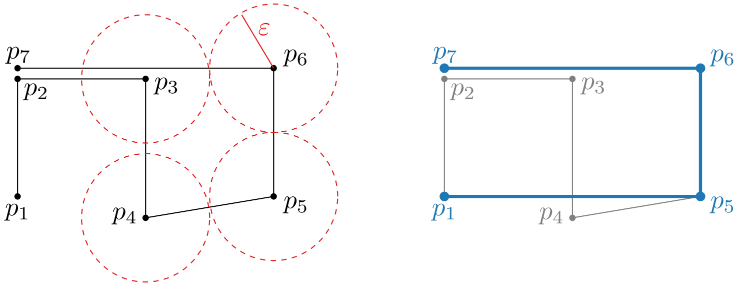


FIGURE 2.1: Simplifications DP_H and II_H (both are the same as the input, left) and OPT_H (in blue, right).

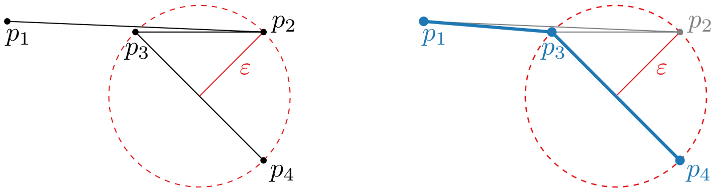


FIGURE 2.2: Simplifications DP_F and II_F (both are the same as the input, left) and OPT_F (in blue, right).

simplification only uses three vertices, $OPT_F(\mathcal{P}) = \langle p_1, p_3, p_4 \rangle$. The segment $\overline{p_1, p_3}$ has Fréchet distance $\leq \varepsilon$ to the most part of $\overline{p_1, p_2}$ and the rest of the input has Fréchet distance $\leq \varepsilon$ to $\overline{p_3, p_4}$.

2.2 Approximation Quality of Douglas-Peucker and Imai-Iri Simplification

The examples of the previous section not only show that II_H and II_F (and DP_H and DP_F) use more vertices than OPT_H and OPT_F, respectively, they show that this is still the case if we run II with a slightly larger value than ε. To let II_H use as few vertices as OPT_H, we must use 2ε instead of ε. This will ensure link $\overline{p_1, p_5}$ is a valid simplification for the part of the input until p_5, see Figure 2.1. For the Fréchet distance, the enlargement factor needed for the example in Figure 2.2 is $\sqrt{2}$. This is needed to match the distance between p_2 and p_3 (or between p_2 and p_4) and make all links involving p_2 or p_3 valid. Then, DP_F and II_F can output the simplified polyline with 3 vertices.

In this section we analyze how the approximation enlargement factor relates to the number of vertices in the Douglas-Peucker and Imai-Iri simplifications and the optimal ones. The interest in such results stems from the fact that the Douglas-Peucker and Imai-Iri algorithms are considerably more efficient than algorithms to compute OPT_H and OPT_F.

2.2.1 Hausdorff Distance

To show that *II*$_H$ (and *DP*$_H$ by consequence) may use many more vertices than *OPT*$_H$, even if we enlarge ε by any constant factor, we give a construction where this occurs. Imagine three circle-shaped regions with diameter ε ($\varepsilon \ll 1$) at the vertices of a sufficiently large equilateral triangle. Let the center of the three regions X, Y and Z be located at $(-1, 0)$, $(0, \sqrt{3})$ and $(1, 0)$, respectively.

Now we construct an input polyline $\mathcal{P}$, see Figure 2.3. We start at p_1 in region X and will go back and forth to region Z while visiting region Y intermediately. The last vertex $p_n \in \mathcal{P}$ is located at region Z. Therefore, $p_1, p_5, p_9, \ldots$ are in region X, $p_2, p_4, p_6, \ldots$ are in region Y, and the remaining vertices are in region Z. Therefore, each link in the Imai-Iri algorithm connecting two vertices in two different regions is valid if and only if it has $\delta_H \leq \varepsilon$ to the third region. An optimal simplification *OPT*$_H$ is $\langle p_1, p_i, p_n \rangle$ where i is any even number between 1 and n (any vertex in region Y). Note that all segments connecting vertices in region X and Y have $\delta_H \leq \varepsilon$ to $\overline{p_1, p_i}$ and the same also applies for segments connecting vertices in Y and Z to $\overline{p_i, p_n}$.

Since the only valid links are the ones connecting two consecutive vertices of $\mathcal{P}$, *II*$_H$ is $\mathcal{P}$ itself. If the triangle is large enough with respect to ε, this remains true even if we give the Imai-Iri algorithm a much larger error threshold than ε. For example, if we enlarge ε by a factor c for *II*$_H$, then the same construction $\mathcal{P}$ where the regions have a distance $2c$ from each other will make *II*$_H$ $(\mathcal{P}, c\varepsilon)$ use more vertices than *OPT*$_H$ $(\mathcal{P}, \varepsilon)$ (which needs only 3). Note that the example applies to both the directed and undirected Hausdorff distance.

Theorem 1. *For any* $c > 1$*, there exists a polyline* $\mathcal{P}$ *with* n *vertices and an* $\varepsilon > 0$ *such that II*$_H(\mathcal{P}, c\varepsilon)$ *has* n *vertices and OPT*$_H(\mathcal{P}, \varepsilon)$ *has* 3 *vertices.*

2.2.2 Fréchet Distance

Our results are somewhat different for the Fréchet distance; we need to make a distinction between *DP*$_F$ and *II*$_F$.

Douglas-Peucker. We construct an example that shows that *DP*$_F$ may have many more vertices than *OPT*$_F$, even if we enlarge the error threshold by any constant factor. See Figure 2.4 for the full construction. Let $\mathcal{P}$ be the input polyline with n vertices. First, p_1 is located at the origin. It is followed by a (nearly vertical) zigzag $p_2, \ldots, p_{n-4}$ such that all vertices p_i

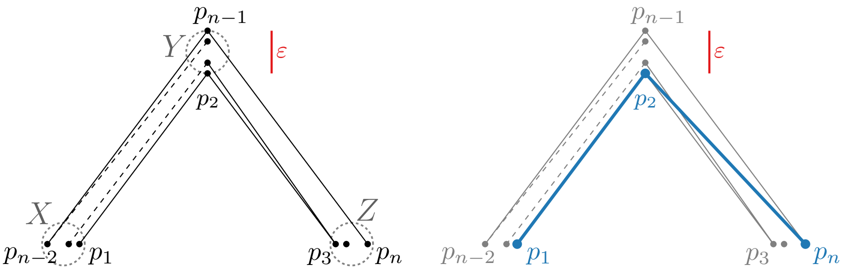


FIGURE 2.3: The Douglas-Peucker and Imai-Iri algorithms may not be able to simplify at all, whereas the optimal simplification using the Hausdorff distance has just three vertices (in blue, right).

where i is an even number are placed above the x-axis and the others are below. Vertices p_2 and p_3 have distance ε to the x-axis and the rest of the vertices gradually get closer to the x-axis. Then, we add vertex p_{n-3} which is located exactly on the x-axis.

Now, the idea is to force the Douglas-Peucker algorithm to get all vertices from the zigzag part, but the optimal simplification only needs one segment that coincides with part of the x-axis because all vertices $p_1, ..., p_{n-3}$ have $\delta_F \leq \varepsilon$ to such a part. Therefore, we place p_{n-2} on the x-axis far away to the right of p_{n-3} such that the distance between p_{n-2} and p_{n-3} is much greater than ε. Then, we place p_{n-1} close to p_{n-3} but slightly below the x-axis without intersecting the zigzag part. We finalize the input by placing p_n close to p_{n-2} and it should have the same y-coordinate as p_{n-1}.

The placement of the last three vertices will ensure that the link $\overline{p_1, p_n}$ is not close enough to any vertex above the x-axis to be valid. Since vertex p_2 is placed slightly higher than $p_4, p_6, \ldots$, it will be added first by the Fréchet version of the Douglas-Peucker algorithm and eventually, all vertices will be added. Furthermore, the (horizontal) zigzag of $p_{n-3}, ..., p_n$ is large (wide) compared to ε and $\overline{p_{n-3}, p_n}$ has $\delta_F > \varepsilon$ to both p_{n-1} and p_{n-2}. Thus, both of the vertices will be added as part of the simplified output by the Douglas-Peucker algorithm.

While DP_F consists of all vertices from $\mathcal{P}$, OPT_F has only four vertices because the link $\overline{p_1, p_{n-2}}$ has a distance $\leq \varepsilon$ to all vertices up to p_{n-2}. Since the width of the zigzag $p_{n-3}, \ldots, p_n$ can be arbitrarily much larger than the height of the vertical zigzag $p_1, \ldots, p_{n-4}$, the situation remains if we make the error threshold arbitrarily much larger.

Theorem 2. *For any $c > 1$, there exists a polyline $\mathcal{P}$ with n vertices and an $\varepsilon > 0$ such that $DP_F(\mathcal{P}, c\varepsilon)$ has n vertices and $OPT_F(\mathcal{P}, \varepsilon)$ has 4 vertices.*

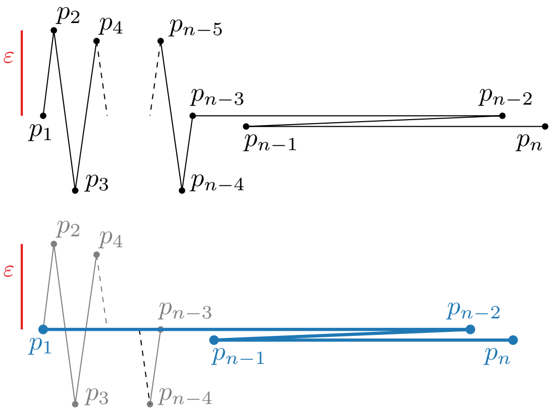


FIGURE 2.4: Top: a polyline on which the Fréchet version of the Douglas-Peucker algorithm performs poorly and the output polyline contains n vertices. Bottom: the optimal simplification contains four vertices (in blue).

Remark. One could argue that the choice of adding the furthest vertex is not suitable when using the Fréchet distance, because we may not be adding the vertex (or vertices) that are to "blame" for the high Fréchet distance. However, finding the vertex that improves the Fréchet distance most is computationally expensive, defeating the purpose of this simple algorithm. Furthermore, we can observe that also in the Hausdorff version, the Douglas-Peucker algorithm does not choose the vertex that improves the Hausdorff distance most (it may even increase when adding an extra vertex).

Imai-Iri. Finally, we compare the Fréchet version of the Imai-Iri algorithm to the optimal Fréchet distance simplification. The previous construction in Figure 2.4 shows that under the Fréchet distance, $II_F = OPT_F$ because in the Imai-Iri algorithm, $\overline{p_1, p_{n-2}}$ is a valid link. Therefore, we give another input polyline $\mathcal{P}$ to show that II_F also does not approximate OPT_F even if II_F is allowed to use a value for ε that is larger by a constant factor.

The main strategy for this construction is to prevent any occurrence of valid links in the Imai-Iri algorithm, but OPT_F must be able to skip several vertices using non-valid links. We show the construction of such input polyline in Figure 2.5. First, we start with three vertices p_3, p_4 and p_5. The three vertices are positioned at the vertices of an isosceles triangle that has its base (p_3 and p_5) in a horizontal position with distance ε to each other and p_4 is located far away to the right with a distance 6ε from the base of the triangle. Next, we put p_1 and p_2 very close to p_5 and p_4, respectively. The starting vertex p_1 is located on the base of the triangle and slightly

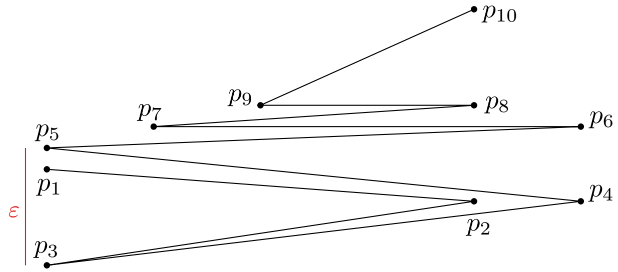


FIGURE 2.5: The Imai-Iri simplification will have all vertices because the only valid links with a Fréchet distance at most ε are the ones connecting two consecutive vertices in the polyline.

below p_5, their distance is μ where $\mu \ll \varepsilon$. The vertex p_2 is located to the left of p_4 (with the same y-coordinate) and their distance is also μ. With this construction, we prevent any valid links except for the segments of the polyline itself.

Then, the sub-polyline $\langle p_4, p_5, ..., p_{10} \rangle$ is a horizontal zigzag whose width becomes smaller at every turn. The vertex p_6 has distance ε to p_4, and similarly p_7 to p_5, p_8 to p_6 and p_9 to p_7. The last three vertices p_8, p_9 and p_{10} are located on the circle with radius ε where p_9 has distance 2ε to p_{10} and more than ε to p_8, but the distance between p_8 and p_{10} must be less than ε. This configuration ensures that $\overline{p_8, p_{10}}$ will not be a valid link.

Since there is no valid link for the Imai-Iri algorithm, $\mathit{II}_F = \mathcal{P}$. However, the consecutive links $\overline{p_1, p_4}$ and $\overline{p_4, p_5}$ have $\delta_F \leq \varepsilon$ to $\langle p_1, p_2, p_3 \rangle$, which can be seen clearly in the free space diagram in Figure 2.6 (top). Furthermore, the zig-zag part of the construction forces the simplified output to be always two vertices behind the input. Since the distance between p_5 and p_3 and between p_6 and p_4 is ε, we have $\delta_F(\langle p_1, ..., p_6 \rangle, \langle p_1, ..., p_4 \rangle) \leq \varepsilon$ and this is still true if we continue adding the next vertex p_7, and so on. Finally, in the last part, the point in the middle of $\overline{p_9, p_{10}}$ has a distance ε to the last three vertices and therefore, the simplified output can make up for the situation in which it is behind by two vertices. Since OPT_F can use $\overline{p_1, p_4}$ as a link, it can skip two vertices and consequently, use less vertices than II_F. See Figure 2.6 (bottom).

We can append multiple copies of this construction together with a suitable connection in between. This way we obtain:

Theorem 3. *There exist constants $c_1 > 1$, $c_2 > 1$ such that for any $n > 0$, a polyline $\mathcal{P}$ with n vertices and an $\varepsilon > 0$ exist such that $|\mathit{II}_F(\mathcal{P}, c_1\varepsilon)| > c_2|\mathit{OPT}_F(\mathcal{P}, \varepsilon)|$.*

From Agarwal et al. [157], we know that $|\mathit{II}_F(\mathcal{P}, 4\varepsilon)| \leq |W_F(\mathcal{P}, \varepsilon)|$ where W_F is a *weak* ε-simplification under the Fréchet distance. Since

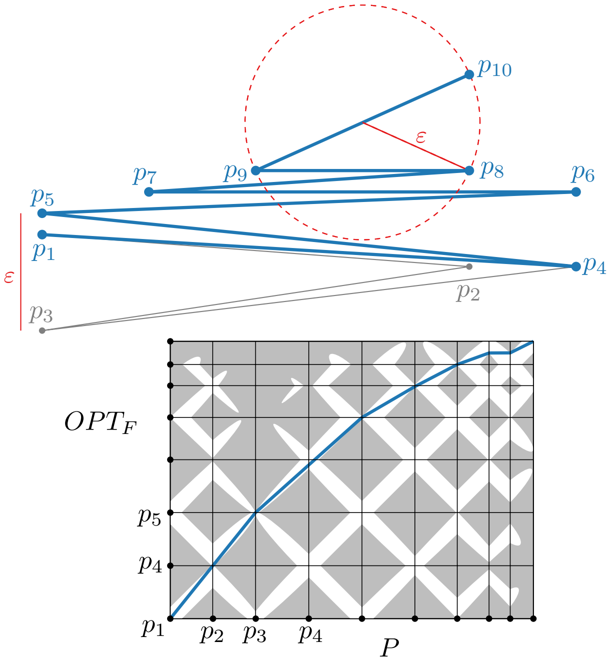


Figure 2.6: The optimal simplification can skip p_2 and p_3; in the parametrizations witnessing the Fréchet distance, OPT_F "stays two vertices behind" on the input until the end. Bottom, the free space diagram of $\mathcal{P}$ and OPT_F.

we know that $|W_F(\mathcal{P},\varepsilon)| \leq |OPT_F(\mathcal{P},\varepsilon)|$, it implies that $|II_F(\mathcal{P},4\varepsilon)| \leq |OPT_F(\mathcal{P},\varepsilon)|$. Therefore, for any value of $c_1 \geq 4$, the above theorem is not true.

2.3 Algorithmic Complexity of the Hausdorff Distance

The results in the previous section show that both the Douglas-Peucker and the Imai-Iri algorithm do not produce an optimal polyline that has Hausdorff or Fréchet distance at most ε, or even approximates it well. Naturally, this leads us to the following question: Is it possible to compute the optimal Hausdorff or Fréchet simplification in polynomial time?

In this section, we present a construction which proves that under the Hausdorff distance, computing the optimal simplified polyline is NP-hard.

2.3.1 Undirected Hausdorff Distance

We first consider the undirected (or bidirectional) Hausdorff distance; that is, we require both the maximum distance from the initial polyline $\mathcal{P}$ to the simplified polyline $\mathcal{Q}$ and the maximum distance from $\mathcal{Q}$ to $\mathcal{P}$ to be at most ε.

Theorem 4. *Given a polyline $\mathcal{P} = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence $\mathcal{Q}$ of $\mathcal{P}$ such that the undirected Hausdorff distance between $\mathcal{P}$ and $\mathcal{Q}$ is at most ε is NP-hard.*

We prove the theorem with a reduction from Hamiltonian cycle in orthogonal segment intersection graphs.[3] Akiyama et al. [161] show that Hamiltonian cycle remains NP-complete in 2-connected cubic bipartite planar graphs and Czyzowicz et al. [162] prove that there is a linear time algorithm to represent bipartite planar graphs as an intersection graph of orthogonal line segments. Hence, Hamiltonian cycle in orthogonal segment intersection graphs is NP-complete.

Let $\mathcal{L}$ be a set of n horizontal or vertical line segments in the plane. Assume the segment endpoints have integer coordinates that are linear in n and all intersections are proper (if not, extend the segments slightly). Let G be its orthogonal segment intersection graph (that is, G has a vertex for every segment in $\mathcal{L}$, and two vertices in G are connected by an edge when their corresponding segments intersect). We assume that G is connected; otherwise, clearly there is no Hamiltonian cycle in G.

We first construct an initial polyline $\mathcal{P}$ as follows (Figure 2.7 illustrates the construction). Let $\mathcal{A}$ be the arrangement of $\mathcal{L}$ (intersections in $\mathcal{L}$ become vertices in $\mathcal{A}$), let p be some endpoint of a segment in $\mathcal{L}$, and let π be any path on $\mathcal{A}$ that starts and finishes at p and must visit all vertices and edges of $\mathcal{A}$. Since π does not need to be a shortest path, it may visit the same vertices and edges more than once and clearly, π can be constructed in linear time. P is simply $3n + 1$ copies of π appended to each other. Consequently, the order of vertices in $\mathcal{Q}$ now must follow the order of these copies. We set ε to a sufficiently small value ($\varepsilon \ll 1$). Since we require all vertices to have integer coordinates, there will be no valid link, except the ones where both endpoints of a link have the same x- or y-coordinate.

Now, an output polyline $\mathcal{Q}$ with Hausdorff distance at most ε to $\mathcal{P}$ must also visit all vertices and edges of $\mathcal{A}$, and stay close to $\mathcal{A}$. If ε is sufficiently small, there will be no benefit for $\mathcal{Q}$ to ever leave $\mathcal{A}$.

Lemma 5. *A solution $\mathcal{Q}$ of length $3n + 1$ exists if and only if G admits a Hamiltonian cycle.*

---

[3] Our original reduction was from Hamilton cycle in segment intersection graphs (not necessarily orthogonal). Subsequently, van de Kerkhof et al. [160] simplified and extended the reduction to show that the problem remains hard even when the simplification vertices are not restricted to be a subset of the input vertices. Their choice to reduce from the smaller class of *orthogonal* segment intersection graphs (on which Hamiltonian cycle is still NP-hard) simplifies the argument and avoids the issue of coordinate bit complexity of an explicit representation of the graph; hence, here we also adopt this change.

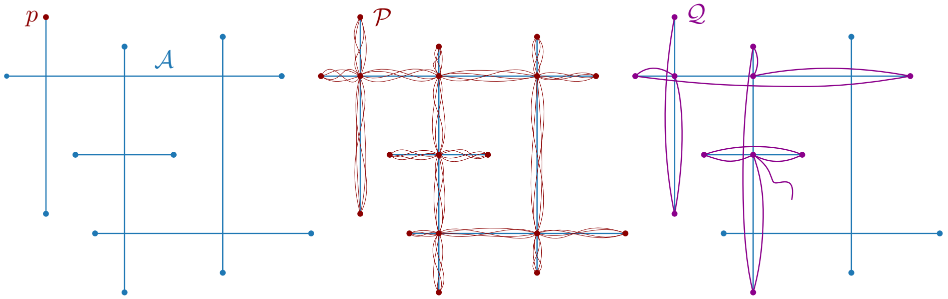


FIGURE 2.7: The construction: $\mathcal{A}$ is the arrangement of a set of segments $\mathcal{L}$. We build an input path $\mathcal{P}$ that "paints" over S completely, and we are looking for an output path $\mathcal{Q}$ that corresponds to a Hamiltonian cycle. In this case, there is no Hamiltonian cycle, and the path gets stuck.

Proof. Clearly, any simplification $\mathcal{Q}$ will need to visit the $2n$ endpoints of the segments in $\mathcal{L}$, and since it starts and ends at the same point p, will need to have length at least $2n+1$. Furthermore, $\mathcal{Q}$ will need to have at least two internal vertices on every segment $\ell \in \mathcal{L}$: once to enter the segment and once to leave it (note that we cannot enter or leave a segment at an endpoint since all intersections are proper intersections). Since a vertex to leave a segment coincides with the vertex to enter another segment, the number of these vertices is equal to the number of segments minus one. Therefore, the minimum number of vertices possible for $\mathcal{Q}$ is $3n+1$.

Now, if G admits a Hamiltonian cycle, it is easy to construct a simplification with $3n+1$ vertices as follows. We start at $p \in \mathcal{Q}$ and collect the other endpoint of the segment ℓ_1 of which p is an endpoint. Then we follow the Hamiltonian cycle to segment ℓ_2; by definition $\ell_1\ell_2$ is an edge in G so their corresponding segments intersect, and we use the intersection point to leave ℓ_1 and enter ℓ_2. In G, this means that we move from one vertex (that represents a line segment in $\mathcal{Q}$) to one of the vertices adjacent to it. We proceed in this fashion until we reach ℓ_n, which intersects ℓ_1, and finally return to p.

On the other hand, any solution with $3n+1$ vertices must necessarily be of this form and therefore imply a Hamiltonian cycle: in order to have only 3 vertices per segment the vertex at which we leave ℓ_1 must coincide with the vertex at which we enter some other segment, which we call ℓ_2, and we must continue until we visited all segments and return to p.

If there is no Hamiltonian cycle in G, then we have to visit at least one vertex $\in G$ more than once. This means $\mathcal{Q}$ will have two identical segments to simplified one segment of $\mathcal{P}$. In this case, $\mathcal{Q}$ is clearly not an optimal simplification of $\mathcal{P}$. $\square$

2.3.2 Directed Hausdorff Distance: $\mathcal{P}$ to $\mathcal{Q}$

We now shift our attention to the directed Hausdorff distance from $\mathcal{P}$ to $\mathcal{Q}$: we require the maximum distance from $\mathcal{P}$ to $\mathcal{Q}$ to be at most ε, but $\mathcal{Q}$ may have a larger distance to $\mathcal{P}$. The previous reduction does not seem to work because there is always a Hamiltonian Cycle of length $2n$ for this measure: $\mathcal{Q}$ can always leave a segment from its endpoint and move to an endpoint of another segment directly without the need to stay close to $\mathcal{P}$. Meanwhile, $\mathcal{P}$ is always close to $\mathcal{Q}$, as long as $\mathcal{Q}$ is also moving between endpoints of the same segment. Therefore, we prove the NP-hardness using a different approach.

The idea is to reduce from COVERING POINTS BY LINES, which is known to be both NP-hard [163] and APX-hard [164]: given a set $\mathcal{S}$ of points in $\mathbb{R}^2$, find a set of straight lines $\mathcal{L}$ of minimum cardinality such that every point in S lies on at least one line in $\mathcal{L}$.

The Construction of the Input Polyline $\mathcal{P}$

Let $\mathcal{S} = \{s_1, \ldots, s_m\}$ be an instance of the COVERING POINTS BY LINES problem. We fix ε based on $\mathcal{S}$ and present the construction of a polyline connecting a sequence of $n = \text{poly}(m)$ points: $\mathcal{P} = \langle p_1, p_2, ..., p_n \rangle$ such that for every $1 \leq i \leq m$, we have $s_i = p_j$ for some $1 \leq j \leq n$. The idea is to force the simplification $\mathcal{Q}$ to cover all points in $\mathcal{P}$ except those in $\mathcal{S}$, such that in order for the final simplification to cover all points, we only need to collect the points in $\mathcal{S}$ using as few line segments as possible (Figure 2.8 shows this idea). To this end, we will place a number of *forced points* $F \subset \mathcal{P}$, where a point f is *forced* whenever its distance to any line through any pair of points in $\mathcal{P}$ is larger than ε. Since $\mathcal{Q}$ must be defined by a subset of points in $\mathcal{P}$, we will never cover f unless we choose f to be a vertex of $\mathcal{Q}$. On the other hand, we need to place points that allow us to freely draw every line through two or more points in $\mathcal{S}$. Figure 2.9 shows the idea. We create two point sets L and R to the left and right of $\mathcal{S}$, such that for every line through two or more points in $\mathcal{S}$, there are a point in L and a point in R on that line. Finally, we need to build additional scaffolding around the construction to connect and cover the points in L and R.

We now discuss the construction in detail, divided into three parts with different purposes:

1. a sub-polyline that contains $\mathcal{S}$;
2. a sub-polyline that contains L and R; and
3. two disconnected sub-polylines which share the same purpose: to guarantee that all vertices in the previous sub-polyline are themselves covered by $\mathcal{Q}$.

Part 1: Placing $\mathcal{S}$

First, we assume that every point in $\mathcal{S}$ has a unique x-coordinate; if this is not the case, we rotate $\mathcal{S}$ until it is. We also assume that every line through at least two points of $\mathcal{S}$ has a slope between -1 and $+1$; if this is not the case, we vertically scale $\mathcal{S}$ until it is. Now, we fix ε to be smaller than half the minimum difference between any two x-coordinates of points in $\mathcal{S}$, and smaller than the distance from any line through two points in $\mathcal{S}$ to any other point in $\mathcal{S}$ not on the line.

We place $m+1$ forced points $f_1, f_2, ..., f_m, f_{m+1}$ such that the x-coordinate of f_i lies between the x-coordinates of s_{i-1} and s_i (except for f_1 where $f_{1_x} < s_{1_x}$ and f_{m+1} where $f_{m+1_x} > s_{m_x}$) and the points lie alternatingly above and below $\mathcal{S}$. Furthermore, the y-coordinates of these forced points must be larger or smaller (at least by 2ε) than the maximum (or minimum) y-coordinate of all points in $\mathcal{S}$. We also place these forced points such that the distance of the line segment $\overline{f_i f_{i+1}}$ to s_i is $\frac{3}{2}\varepsilon$ and the distance of $\overline{f_i f_{i+1}}$ to s_{i-1} is larger than ε. Next, we place two auxiliary points t_i^+ and t_i^- on $\overline{f_i f_{i+1}}$ such that the distance of each point to s_i is 2ε; refer to Figure 2.8. These auxiliary points make another requirement for the position of the forced points $f_1, ..., f_{m+1}$: for each forced point f_i, its distance to f_{i+1} should be at least 10ε. This ensures that the distance of $\overline{f_i, s_i}$ to t_i^+ and $\overline{s_i, f_{i+1}}$ to t_i^- is more than ε. Finally, let $\tau_1 = \langle f_1, t_1^+, s_1, t_1^-, f_2, t_2^-, s_2, t_2^+, f_3, \ldots, f_{m+1}\rangle$ be a polyline connecting all points in the construction; τ_1 will be part of the input segment $\mathcal{P}$.

The idea here is that all forced points must appear on $\mathcal{Q}$, and if only the forced points appear on $\mathcal{Q}$, everything in the construction will be covered *except* the points in $\mathcal{S}$ (and some arbitrarily short stubs of edges connecting them to the auxiliary points). Of course, we could choose to include more points in τ_1 in $\mathcal{Q}$ to collect some points of $\mathcal{S}$ already. However, this would cost an additional three vertices per collected point (note that using fewer than three, we would miss an auxiliary point instead), and in the remainder of the construction, we will make sure that it is cheaper to cover the points in $\mathcal{S}$ separately later.

Part 2: Placing and covering L and R

In the second part of the construction, we create two sets of $O(m^2)$ vertices, L and R, which can be used to make links that cover $\mathcal{S}$. Consider the set Λ of all $k \leq \frac{m^2-m}{2}$ unique lines that pass through at least two points in $\mathcal{S}$. We create two sets of k points $L = \{\ell_1, \ell_2, \ldots, \ell_k\}$ and $R = \{r_1, r_2, \ldots, r_k\}$ with the following properties:

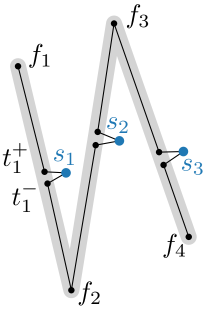


FIGURE 2.8: Example of τ_1 where $m = 3$. For a given ε, the (simplified) polyline f_1, f_2, f_3, f_4 covers the gray area but not the blue points s_1, s_2, s_3.

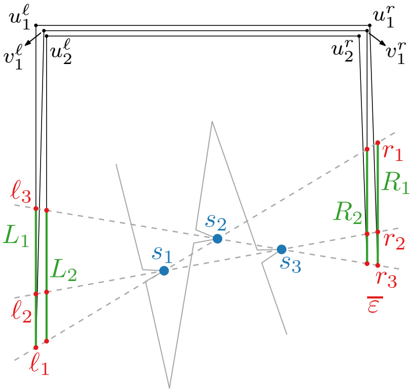


FIGURE 2.9: Construction to allow the lines that can be used to cover the points of $\mathcal{S}$. To ensure the order of vertices in $\mathcal{Q}$, we create copies of L and R. Then, $\mathcal{Q}$ can use them alternatingly.

- the line through ℓ_i and r_i is one of the k lines in Λ,
- the line through ℓ_i and r_j for $i \neq j$ has distance more than ε to any point in $\mathcal{S}$, and
- the points in L (resp. R) all lie on a common vertical line.

Clearly, we can satisfy these properties by placing L and R sufficiently far from $\mathcal{S}$. We create a vertical polyline for each set, which consists of $k - 1$ non-overlapping line segments that are connecting consecutive vertices in their y-order from top to bottom. Let R_1 and L_1 be such polylines containing k vertices each.

Now, each line that covers a subset of $\mathcal{S}$ can become part of $\mathcal{Q}$ by selecting the correct pair of vertices from R and L. However, if we want $\mathcal{Q}$ to contain multiple such lines, this will not necessarily be possible anymore, since the order in which we visit R_1 and L_1 is fixed (and to create a line, we must skip all intermediate vertices). The solution is to make h copies[4] $R_1, R_2, \ldots, R_h$ of R_1 and h copies $L_1, L_2, \ldots, L_h$ of L_1 and visit them alternatingly. Here $h = \lceil \frac{m}{2} \rceil$ is the maximum number of lines necessary to cover all points in $\mathcal{S}$ in the COVERING POINTS BY LINES problem.

We create a polyline τ_2 that contains R_1 and L_1 by connecting them with two new vertices u_1^r and u_1^ℓ located above R_1 and L_1, respectively. The x-coordinates of u_1^r and u_1^ℓ are the same as of R_1 and L_1 (for clarity in the

---

[4] The copies are in exactly the same location. If the reader does not like that and feels that points ought to be distinct, she may imagine shifting each copy by a sufficiently small distance (smaller than ε/h) without impacting the construction.

figures, we slightly move u_1^r to the left). Both u_1^r and u_1^ℓ should be located far enough from R_1 and L_1 such that a link between u_1^r and a vertex in L_1 (and u_1^ℓ with R_1) will not cover any point in $\mathcal{S}$.

To ensure that the construction ends at the last vertex in L_h after visiting the h copies of R_1 and L_1 alternatingly, we use two intermediate vertices v_1^ℓ and v_1^r and make $h-1$ copies of them. These intermediate vertices are located close to u_1^ℓ and u_1^r, see Figure 2.9.

Finally, let $\tau_2 = \langle R_1, u_1^r, u_1^\ell, L_1, v_1^\ell, v_1^r, R_2, u_2^r, u_2^\ell, L_2, v_2^\ell, \ldots, L_h\rangle$ be a polyline connecting all points in the construction; τ_2 will also be part of the input $\mathcal{P}$.

Part 3: Putting it together

All vertices in τ_1 can be covered by the simplification $\langle f_1, f_2, ..., f_{m+1}\rangle$ and a suitable choice of links that connect pairs of vertices from R and L (in τ_2). Therefore, the last part of the input $\mathcal{P}$ (τ_3) has two purposes: to definitely cover all vertices in τ_2 and as a proper connection between τ_1 and τ_2. Consequently, all vertices in τ_3 will also be *forced* and therefore be a part of the final simplified polyline.

We divide this last part into two unconnected polylines: τ_{3_a} and τ_{3_b}. The main part of τ_{3_a} is a vertical line segment e that is parallel to R_1. There is a restriction to e: the Hausdorff distance from each of R_i, u_i^r, v_j^r $(1 \le j < i \le h)$, and also from line segments between them to e should not be larger than ε. In order to force e to be a part of the simplified polyline, we must place its endpoints away from τ_2. Then, τ_1 and τ_2 can be connected by connecting $f_{n+1} \in \tau_1$ and the first vertex in R_1 to different endpoints of e.

Next, the rest of τ_2 that has not been covered yet, will be covered by τ_{3_b}. This is shown in Figure 2.10, left. First, we have a vertical line segment g that is similar to e, in order to cover L_i, u_i^ℓ, v_j^ℓ $(1 \le j < i \le h)$, and all line segments between them. Then, a horizontal line segment z is needed to cover all horizontal line segments $\overline{u_i^r u_i^\ell}$ and $\overline{v_j^\ell v_j^r}$ $(1 \le j < i \le h)$. Similar to e, the endpoints of g and z should be located far from τ_2, implying that z intersects both e and g. We complete the construction by connecting the upper endpoint of g to the left endpoint of z and the lower endpoint of g to the last vertex in L_h.

We can show that even if the input is restricted to be non-self-intersecting, the simplification problem is still NP-hard. This is shown in Figure 2.10, right. We modify the last part of the construction to remove the three intersections. Firstly, we shorten z on the right side and place it very close to u_1^r. Since the right endpoint of z is an endpoint of the input, it will always be included in a simplification. Secondly, to remove the intersection of g

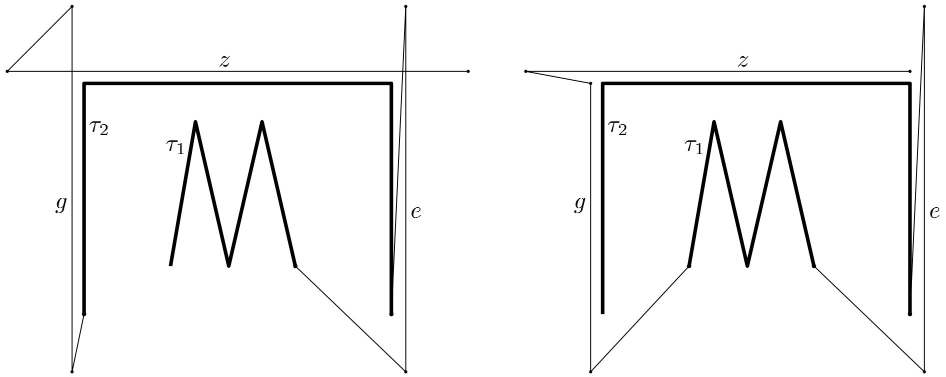


FIGURE 2.10: Schematic views of connecting up different parts of the NP hardness construction into a single polyline. The bold polylines show τ_1 and τ_2 and indicate multiple parts of $\mathcal{P}$ close together.

and z, we bring the upper endpoint of g to just below z, so very close to u_1^ℓ. To make sure that we must include g in the simplification we connect the lower endpoint of g to f_1. This connecting segment is further from g so it cannot help enough to cover the lower part of g; only g itself can do that.

Building the Construction in Polynomial Time

We will show that building the actual construction from the sketch in the previous subsection is easy and can be done in polynomial time. Recall that we have a set of points $\mathcal{S} = \{s_1, \dots, s_m\}$ as the instance of the COVERING POINTS BY LINES problem and the actual construction of the input polyline P is based on these points. We set $\varepsilon = 4$ and illustrate the full construction in Figure 2.11.

We start with a sufficiently large rectangle Γ with a size of 512×768 and set the lower-left corner of Γ at the origin $(0, 0)$.Imagine that we have two squares with sides of ε, q_ℓ and q_r, located at the upper-left and upper-right corners of Γ, respectively. The upper-left endpoint of q_ℓ coincides with the upper left endpoint of Γ and the same holds for the upper-right endpoints of q_r and Γ.

Let the lines g, z, and e (from Figure 2.10, right) coincide with the left, top, and right edges of Γ. We put the lower endpoints of g and e such that they coincide with the lower-left and lower-right corners of Γ, respectively. The upper endpoint of e must be placed upwards from the upper-right corner of Γ by more than ε. Now, we place the upper endpoint of g such that it coincides with the lower-left corner of q_ℓ. For z, its right endpoint must coincide with the upper left endpoint of q_r and its left endpoint must have smaller x-coordinate (by $> \varepsilon$) than the upper-

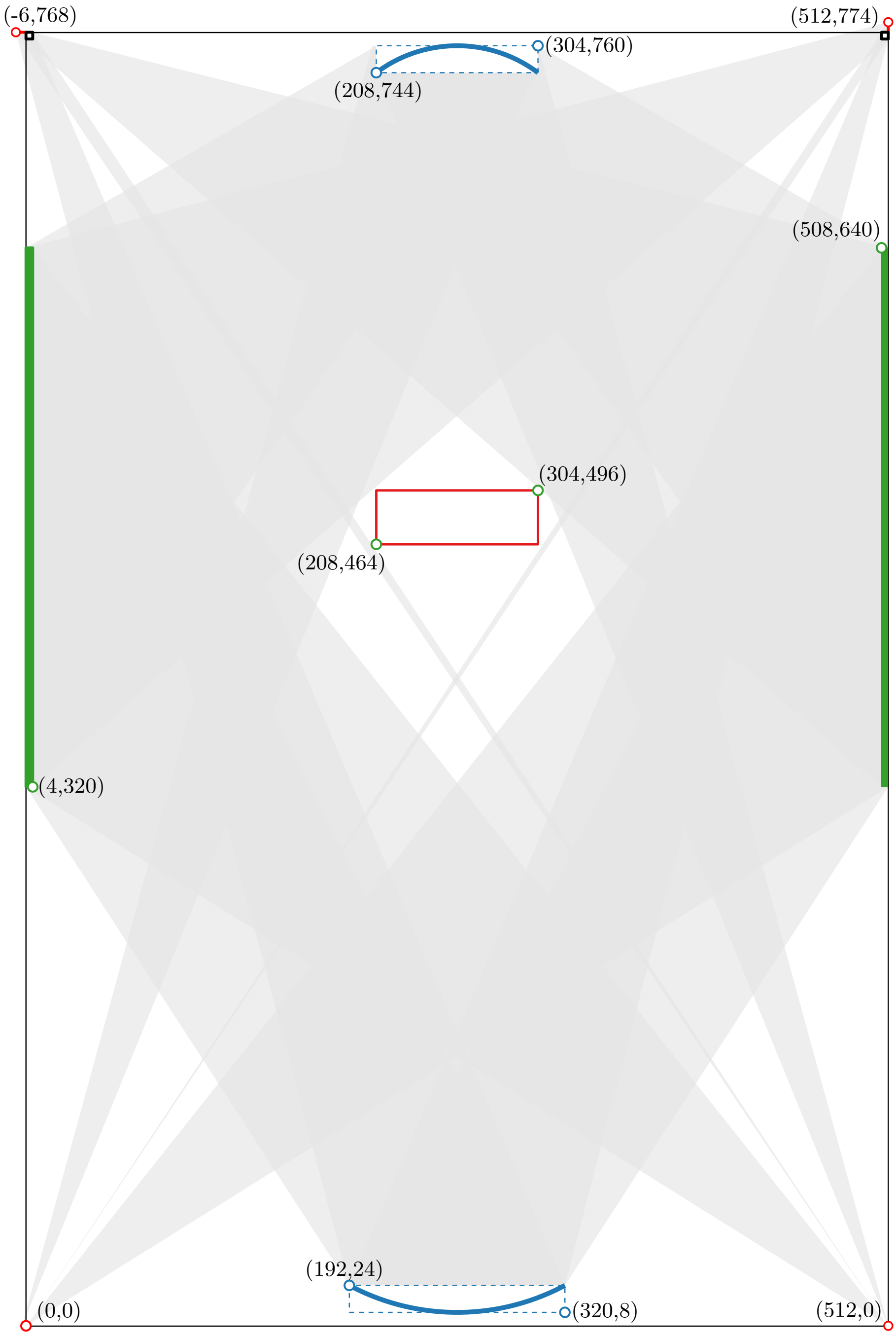


FIGURE 2.11: The construction (with several important coordinates) that can be used to create an actual input polyline $\mathcal{P}$. The large black rectangle is Γ and the red rectangle in the middle is the area to put $\mathcal{S}$. After that, we can create $\mathcal{P}$ according to the process in Section 2.3.2.

right corners of q_ℓ. Furthermore, we use q_ℓ and q_r as the locations for the connecting vertices $\{u_1^\ell, v_1^\ell, u_2^\ell, v_2^\ell, ..., u_h^\ell, v_h^\ell\}$ and $\{u_1^r, v_1^r, u_2^r, v_2^r, ..., u_h^r, v_h^r\}$, respectively. Refer to Figure 2.9.

Next, we fix two rectangles w_ℓ and w_r (shown in green) of size $320 \times \varepsilon$ and place them inside Γ. The coordinates for their lower-right endpoints are $(0, 320)$ and $(512 - \varepsilon, 320)$. Both w_ℓ and w_r are the locations for points in $L_1, L_2, ..., L_h$ and $R_1, R_2, ..., R_h$ ($h = \lceil \frac{m}{2} \rceil$), respectively (see Figure 2.9). Thus, w_ℓ, w_r, q_ℓ, and q_r are the locations of vertices of τ_2, see Figure 2.10 (right).

We can now fix the position of two small rectangles as the places for the set of m forced points $f_1, f_2, ..., f_m, f_{m+1}$. These rectangles should be placed near the top and bottom edge of Γ, in the middle, with a distance $> \varepsilon$ to the top and bottom edges. Note that the rectangle near the bottom edge is bigger than the other one because of the position of the forced points, refer to Figure 2.8. Inside each rectangle, we use a curve to place the forced points such that any shortcut connecting two forced points does not have other forced points within distance ε.

Now, shortcuts through points in $\mathcal{S}$ must be line segments that have endpoints inside w_ℓ and w_r. Since it is possible to make other shortcuts not starting and ending in both rectangles, for example, from q_ℓ to w_r, we have to place S such that it can avoid those unused shortcuts. We can identify such locations by marking areas where those shortcuts may occur and place $\mathcal{S}$ outside such areas. The gray area in Figure 2.11 denotes an area connecting different parts of the construction in which no shortcut can be useful for the output simplification. Next, we set the position for a red rectangle (to put $\mathcal{S}$) between w_ℓ and w_r, which do not intersect with the gray area.

Finally, to make sure that any lines between two points in $\mathcal{S}$ will intersect both w_ℓ and w_r, we can scale down the y-coordinates of all points in $\mathcal{S}$ by an integer factor. Again, we assume that all points in $\mathcal{S}$ have distinct x-coordinates.

We present an example of a full construction of the input polyline $\mathcal{P} = \langle \tau_{3_b}, \tau_1, \tau_{3_a}, \tau_2 \rangle$ for $m = 4$ (m is the number of vertices in $\mathcal{S}$, not all of vertices in $\mathcal{P}$) in Figure 2.12. In order to show the vertices clearly, we reduce the size of Γ to 512×480 and use a slightly larger ε ($\varepsilon = 4.5$).

NP-Hardness Proof

In the previous section, we presented the full construction for the input polyline $\mathcal{P}$. Now, we will prove the following theorem:

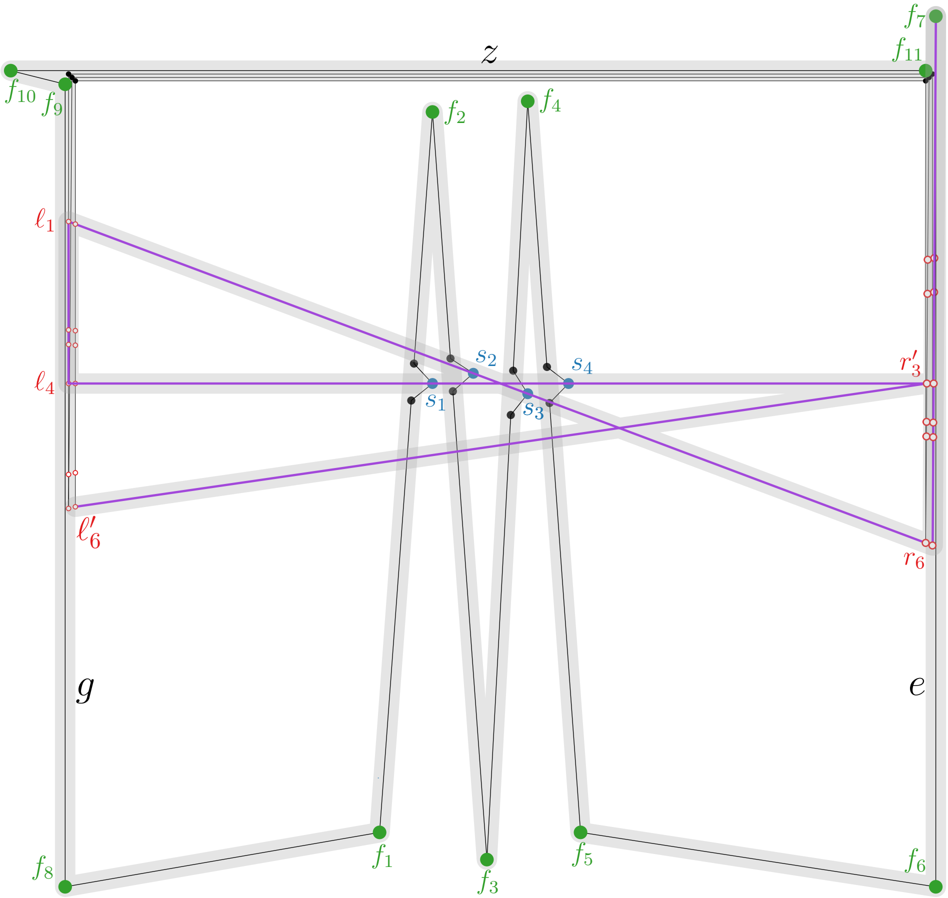


FIGURE 2.12: The full construction showing that computing OPT_H is NP-hard. τ_{3_a} is a line segment $e = \overline{f_6, f_7}$ and $\tau_{3_b} = \langle f_8, ..., f_{11} \rangle$. The endpoints of the construction are f_{11} and $\ell'_6 \in L_2$. The gray area is within ε from the sub-polyline consist of all green vertices: $\langle f_{11}, .., f_8, f_1, .., f_7 \rangle$, which is a part of the simplification. The rest of the simplification is the purple polyline $\langle f_7, r_6, \ell_1, \ell_4, r'_3, \ell'_6 \rangle$ that covers all blue points $\mathcal{S}$ ($r'_3 \in R_2$ and $l'_6 \in L_2$). In order to show the blue points S clearly, we enlarged their size. As a consequence, both s_2 and s_3 have a distance $> \varepsilon$ to a link going through s_1 and s_4.

Theorem 6. *Given a polyline $\mathcal{P} = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence $\mathcal{Q}$ of $\mathcal{P}$ such that the directed Hausdorff distance from $\mathcal{P}$ to $\mathcal{Q}$ is at most ε is NP-hard.*

Proof. The construction contains $O(m^2)$ vertices and a part of its simplified polyline with a constant number of vertices that contains $f_1, f_2, ..., f_{m+1}$ and all vertices in τ_{3_a} and τ_{3_b} can cover all vertices in the construction except for $\mathcal{S}$. Then, the other part of the simplified polyline depends on links to cover points in $\mathcal{S}$. These links alternate between going from left to right and from right to left. Between two such links, we will have exactly two vertices from some L or two from some R.

The only two ways a point s_i can be covered is by including s_i explicitly or by one of the $O(m)$ links that cover s_i and at least another point s_j. If we include s_i explicitly then we must also include t_i^+ and t_i^- or else they are not covered. It is clearly more efficient (requiring fewer vertices in the simplification) if we use a link that covers s_i and another s_j, even if s_j is covered by another such link too. The links of this type in an optimal simplified polyline correspond precisely to a minimum set of lines covering $s_1, \ldots, s_m$. Therefore, the simplified polyline of the construction contains a solution to COVERING POINTS BY LINES instance. Since $\mathcal{P}$ in the construction is simple, the theorem holds even for simple input. □

2.3.3 Directed Hausdorff Distance: $\mathcal{Q}$ to $\mathcal{P}$

Finally, we finish this section with a note on the reverse problem: we want to only bound the directed Hausdorff distance from $\mathcal{Q}$ to $\mathcal{P}$ (we want the output segment to stay close to the input segment, but we do not need to be close to all parts of the input). This problem seems more esoteric but we include it for completeness. In this case, a polynomial time algorithm (reminiscent of Imai-Iri) optimally solves the problem.

Theorem 7. *Given a polyline $\mathcal{P} = \langle p_1, p_2, \ldots, p_n \rangle$ and a value ε, the problem of computing a minimum length subsequence $\mathcal{Q}$ of $\mathcal{P}$ such that the directed Hausdorff distance from $\mathcal{Q}$ to $\mathcal{P}$ is at most ε can be solved in $O(n^3 \log n)$ time.*

Proof. We compute the region $\mathcal{R}$ with distance ε from $\mathcal{P}$ explicitly. The boundary of $\mathcal{R}$ consist of straight line segments and circular arcs with radius of ε. Computing this boundary takes $O(n \log n)$ time when $\mathcal{P}$ is simple and $O(n^2 \log n)$ time when $\mathcal{P}$ is not, by first constructing the arrangement and then the Voronoi diagram of the edges. The worst-case complexity of $\mathcal{R}$ is quadratic.

Now, for every vertex $p \in \mathcal{P}$, split circular arcs in $\mathcal{R}$ at tangent lines through p. Since a circular arc is split into at most 3 pieces, the complexity of the new boundary $\mathcal{B}$ of $\mathcal{R}$ (with split arcs) still has quadratic complexity. Sort all n shortcuts from p to other vertices in $\mathcal{P}$ and all extremes from $\mathcal{B}$ by angle around p, and perform a rotational sweep. This way, it is easy to find all valid shortcuts from p that lie inside $\mathcal{B}$ in $O(n^2 \log n)$ time. In total, we spend $O(n^3 \log n)$ time to find all valid shortcuts for all $p \in \mathcal{P}$. Finally, we build the graph G where these valid shortcuts are the edges and compute a shortest path in G, which represents an optimal simplification for this measure. □

2.4 Algorithmic Complexity of the Fréchet Distance

In this section, we show that for a given polyline $\mathcal{P} = \langle p_1, p_2, ..., p_n \rangle$ and an error ε, the optimal simplification $\mathcal{Q} = \mathit{OPT}_F(\mathcal{P}, \varepsilon)$ can be computed in polynomial time using a dynamic programming approach.

2.4.1 Observations

Note that a link $\overline{p_i p_j}$ in $\mathcal{Q}$ is not necessarily within Fréchet distance ε to the sub-polyline $\langle p_i, p_{i+1}, ..., p_j \rangle$ (for example, $\overline{p_1 p_3}$ in Figure 2.2). Furthermore, a (sequence of) link(s) in $\mathcal{Q}$ could be mapped to an arbitrary subcurve of $\mathcal{P}$, not necessarily starting or ending at a vertex of $\mathcal{P}$. For example, in Figure 2.6, the sub-polyline $\langle p_1, p_4, p_5, p_6 \rangle$ has Fréchet distance ε to a sub-polyline of $\mathcal{P}$ that starts at p_1 but ends somewhere between p_4 and p_5. At this point, one might imagine a dynamic programming algorithm which stores, for each vertex p_i and value k, the point $p(i,k)$ on $\mathcal{P}$ which is the farthest along $\mathcal{P}$ such that there exists a simplification of the part of $\mathcal{P}$ up to p_i using k links that has Fréchet distance at most ε to the part of $\mathcal{P}$ up to $p(i,k)$. However, the following lemma shows that even this does not yield optimality.

Lemma 8. *There exists a polyline $\mathcal{P} = \langle p_1, \ldots, p_{13} \rangle$ and an optimal ε-Fréchet-simplification $\mathcal{Q} = \langle p_1, p_2, p_5, p_6, p_{13} \rangle$ using 4 links, with the following properties:*

- *There exists a partial simplification $R = \langle p_1, p_3, p_5 \rangle$ of $\langle p_1, \ldots, p_5 \rangle$ and a point r on $\overline{p_6 p_7}$ such that the Fréchet distance between R and the subcurve of $\mathcal{P}$ up to r is $\leq \varepsilon$, but*
- *there exists no partial simplification S of $\langle p_5, \ldots, p_{13} \rangle$ that is within Fréchet distance ε to the subcurve of $\mathcal{P}$ starting at r that uses fewer than 7 links.*

Proof. Let $\mathcal{P} = \langle p_1, \ldots, p_{13} \rangle$. The first part of $\mathcal{P}$ consist of the first six vertices. They are positioned such that the only valid 2-link-simplifications

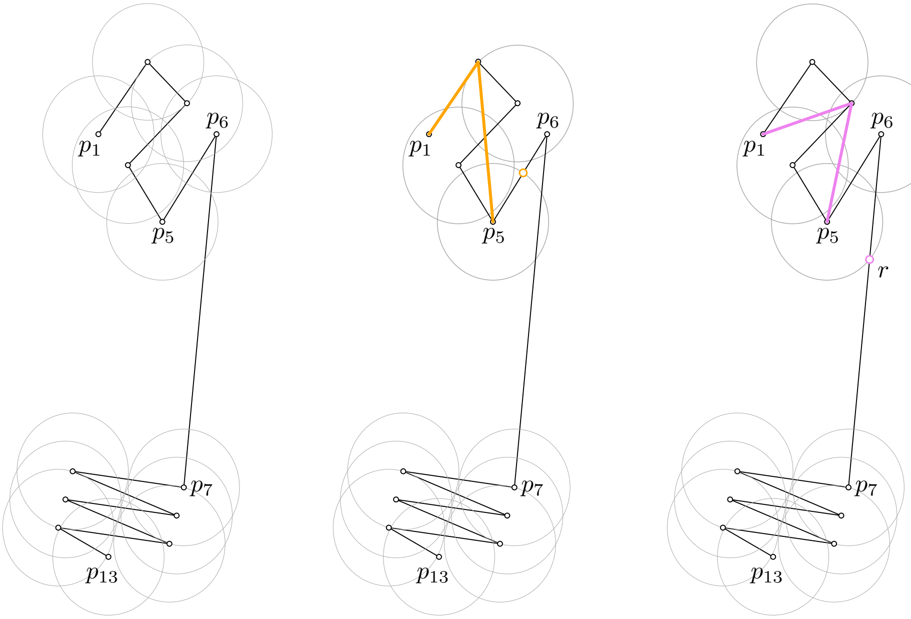


FIGURE 2.13: An example where the farthest-reaching simplification up to p_5 using 2 links is not part of any optimal solution. (Left) The input curve $\mathcal{P}$ in black, with circles of radius ε around all vertices in light gray. The optimal simplification of $\mathcal{P}$ consists of 4 links and must include p_5. (Middle) A 2-link simplification of $\langle p_1, \ldots, p_5 \rangle$ that reaches up to a point on $\overline{p_5 p_6}$ (in yellow) which can be extended to a 4-link optimal simplification of $\mathcal{P}$. (Right) A 2-link simplification of $\langle p_1, \ldots, p_5 \rangle$ that reaches point r on $\overline{p_6 p_7}$ (in pink) which does not allow an optimal simplification.

that have $\delta_F \leq \varepsilon$ to the subcurve $\langle p_1, \ldots, p_5 \rangle$ are $\langle p_1, p_2, p_5 \rangle$ and $\langle p_1, p_3, p_5 \rangle$. Both $\langle p_1, p_2, p_5 \rangle$ and $\langle p_1, p_3, p_5 \rangle$ have $\delta_F \leq \varepsilon$ to the subcurve of P up to some point on $\overline{p_5, p_6}$ and $\overline{p_6, p_7}$, respectively. The second part of P, namely $\langle p_7, \ldots, p_{13} \rangle$, is located far away from the first part and forms a zigzag configuration such that only the segment $\overline{p_6, p_{13}}$ has a distance exactly ε to the vertices in this part of P in between p_6 and p_{13}. See Figure 2.13 (left) for an example of a complete construction of P.

Now, any simplification that contains ≤ 4 links must include $\overline{p_6, p_{13}}$, otherwise it will use more links in the zigzag part of P. Since this is the case, there is no benefit to use a (partial) simplification that uses p_5 and already reaches a point on $\overline{p_6, p_7}$. See Figure 2.13 (middle and right). ☐

2.4.2 A Dynamic Programming Algorithm

Lemma 8 shows that storing a single data point for each vertex and value of k is not sufficient to ensure that we find an optimal solution. Instead, we argue that if we maintain the set of *all* points at $\mathcal{P}$ that can be "reached" by a simplification up to each vertex, then we can make dynamic programming work. We now make this precise and argue that the complexity of these sets of reachable points is never worse than linear.

First, we define π, a parameterization of $\mathcal{P}$ as a continuous mapping: $\pi : [0,1] \to \mathbb{R}^2$ where $\pi(0) = p_1$ and $\pi(1) = p_n$. We also write $\mathcal{P}[s,t]$ for $0 \leq s \leq t \leq 1$ to be the subcurve of $\mathcal{P}$ starting at $\pi(s)$ and ending at $\pi(t)$, also writing $\mathcal{P}[t] = \mathcal{P}[0,t]$ for short.

We say that a point $\pi(t)$ can be *reached* by a (k,i)-simplification for $0 \leq k < i \leq n$ if there exists a simplification of $\langle p_1, \ldots, p_i \rangle$ using k links which has Fréchet distance at most ε to $\mathcal{P}[t]$. We let $\rho(k,i,t) = \texttt{true}$ in this case, and `false` otherwise. With slight abuse of notation we also say that t itself is reachable, and that an interval I is reachable if all $t \in I$ are reachable (by a (k,i)-simplification).

Observation 9. *A point $\pi(t)$ can be reached by a (k,i)-simplification if and only if there exist a $0 < h < i$ and a $0 \leq s \leq t$ such that $\pi(s)$ can be reached by a $(k-1,h)$-simplification and the segment $\overline{p_h p_i}$ has Fréchet distance at most ε to $\mathcal{P}[s,t]$.*

Proof. Follows directly from the definition of the Fréchet distance. ☐

Observation 9 immediately suggests a dynamic programming algorithm: for every k and i we store a subdivision of $[0,1]$ into intervals where ρ is true and intervals where ρ is false, and we calculate the subdivisions for increasing values of k. We simply iterate over all possible values of h, calculate which intervals can be reached using a simplification via h, and then take the union over all those intervals. For this, the only unclear part is how to calculate these intervals.

We argue that, for any given k and i, there are at most $n-1$ reachable intervals on $[0,1]$, each contained in an edge of $\mathcal{P}$. Indeed, every (k,i)-reachable point $\pi(t)$ must have distance at most ε to p_i, and since the edge e of $\mathcal{P}$ that $\pi(t)$ lies on intersects the disk of radius ε centered at p_i in a line segment, every point on this segment is also (k,i)-reachable. We denote the farthest point on e which is (k,i)-reachable by $\hat{t}$.

Furthermore, we argue that for each edge of $\mathcal{P}$, we only need to take the farthest reachable point into account during our dynamic programming algorithm.

Lemma 10. *If k, h, i, s, and t exist such that $\rho(k-1,h,s) = \rho(k,i,t) = \texttt{true}$, and $\overline{p_h p_i}$ has Fréchet distance $\leq \varepsilon$ to $\mathcal{P}[s,t]$, then $\overline{p_h p_i}$ also has Fréchet distance $\leq \varepsilon$ to $\mathcal{P}[\hat{s},\hat{t}]$.*

Proof. By the above argument, $\mathcal{P}[s, \hat{s}]$ is a line segment that lies completely within distance ε from p_h, and $\mathcal{P}[t, \hat{t}]$ is a line segment that lies completely within distance ε from p_i.

We are given that the Fréchet distance between $\overline{p_h p_i}$ and $\mathcal{P}[s, t]$ is at most ε; this means a mapping $f : [s, t] \to \overline{p_h p_i}$ exists such that $|\pi(x) - f(x)| \leq \varepsilon$. Let $q = f(\hat{s})$. Then $|p_h - \pi(\hat{s})| \leq \varepsilon$ and $|q - \pi(\hat{s})| \leq \varepsilon$, so the line segment $\overline{p_h q}$ lies fully within distance ε from $\hat{s}$.

Therefore, we can define a new ε-Fréchet mapping between $\mathcal{P}[\hat{s}, \hat{t}]$ and $\overline{p_h p_i}$ which maps $\hat{s}$ to the segment $\overline{p_h q}$, the curve $\mathcal{P}[\hat{s}, t]$ to the segment $\overline{q p_i}$ (following the mapping given by f), and the segment $\overline{\pi(t)\pi(\hat{t})}$ to the point p_i. □

Now, we can compute the optimal simplification by maintaining a $k \times n \times n$ table storing $\rho(k, i, \hat{t})$, and calculate each value by looking up n^2 values for the previous value of k, and testing in linear time for each combination whether the Fréchet distance between the new link and $\mathcal{P}[\hat{s}, \hat{t}]$ is within ε or not.

Theorem 11. *Given a polyline $\mathcal{P} = \langle p_1, ..., p_n \rangle$ and a value ε, we can compute the optimal polyline simplification of $\mathcal{P}$ that has Fréchet distance at most ε to P in $O(kn^5)$ time and $O(kn^2)$ space, where k is the output complexity of the optimal simplification.*

2.5 Conclusions and Future Work

In this chapter, we analyzed well-known polygonal line simplification algorithms, the Douglas-Peucker and the Imai-Iri algorithm, under both the Hausdorff and the Fréchet distance. Both algorithms are not optimal when considering these measures. We studied the relation between the number of vertices in the resulting simplified polyline from both algorithms and the enlargement factor needed to approximate the optimal solution. For the Hausdorff distance, we presented a polyline where the optimal simplification uses only a constant number of vertices while the solution from both algorithms is the same as the input polyline, even if we enlarge ε by any constant factor. We obtain the same result for the Douglas-Peucker algorithm under the Fréchet distance. For the Imai-Iri algorithm, such a result does not exist but we have shown that we will need a constant factor more vertices if we enlarge the error threshold by some small constant, for certain polylines.

Next, we investigated the algorithmic problem of computing the optimal simplification using the Hausdorff and the Fréchet distance. For the directed and undirected Hausdorff distance, we gave NP-hardness proofs. Interestingly, the optimal simplification in the other direction (from output to input) is solvable in polynomial time.

Finally, we showed how to compute the optimal simplification under the Fréchet distance in polynomial time. Our algorithm is based on the dynamic programming method and runs in $O(kn^5)$ time and requires $O(kn^2)$ space. Recently, Bringmann and Chaudhury improved this result [145].

For future work, we would like to show NP-hardness of computing an optimal simplification using the Hausdorff distance when the simplification may not have self-intersections.

3

Measures for Groups

Although trajectory data is analyzed by researchers from various domain areas and used in varying applications, some fundamental analysis tasks are frequently performed. We can distinguish the following main types of trajectory analysis (see also Section 1.3):

- *Segmentation* [56, 165]: Partitioning a trajectory into sub-trajectories so that within each sub-trajectory, certain attributes of the trajectory are uniform. It may be used to split a trajectory into different movement behaviors.
- *Similarity analysis* [64, 72]: Analyzing how much two trajectories appear alike. It can be used to determine how similar movement of a single entity is on different days, or to determine how similar the movement of two different entities is. The reasons for appearing alike can be rather varied: we could define similarity based on visiting the same locations, or based on having the same speed development (e.g. slow in the beginning, then linearly increasing speed).
- *Clustering* [87]: Grouping the trajectories in a collection based on similarity.
- *Outlier detection* [166, 167]: Finding trajectories that do not belong to any cluster.
- *Classification* [78]: Assigning a trajectory to a cluster of trajectories, based on similarity.
- *Hotspots detection* [98]: Finding places that are visited frequently by many trajectories.
- *Pattern detection*: Finding interesting characteristics in one or more trajectories on any sort (e.g., leadership patterns [104], commuting patterns [65]).

- *Collective movement pattern (group) detection* [112, 126, 118]: Finding a special type of pattern determined by spatial proximity of a subset of the entities over a period of time.

All trajectory analysis types depend on *measures* defined on trajectories. In the listing above, we mentioned location, speed, and similarity as attributes that are defined in the abstract model of trajectories. If needed, then these attributes can be computed in the data model of trajectories. In fact, an attribute like location gives rise to multiple measures: traversed distance, global direction, etcetera.

In this chapter, we give a list of measures for a single trajectory and for groups of trajectories. Recently, a number of different definitions of groups have been given, accompanied by algorithms that compute them from a collection of trajectories [117, 81, 126]. Analysis tasks for trajectories like segmentation and similarity analysis can also be performed for groups of trajectories. Therefore, we also need measures for the whole group.

Results and Organization. We first present an overview of the measures that exist for a single trajectory in Section 3.1. We classify them into three types: measures for a single trajectory in isolation, measures for a single trajectory amidst other trajectories, and measures for a single trajectory in other environments. Then, we proceed with extensions of these measures to groups in Section 3.2, while at the same time including new measures that do not exist for single trajectories. We use the same three types of classification as for a single trajectory. When developing group measures, similar approaches are sometimes taken. We will highlight such more general approaches because they can potentially be used for other measures needed in specific applications. Finally, in Section 3.3, we discuss three applications involving groups where measures for a group of trajectories are necessary: settlement selection, visualization, and segmentation.

3.1 Measures for a Single Trajectory

When defining measures for trajectories we will use the abstract model, since it is mathematically more clean and also representation-independent. All measures can be converted in various ways to measures in a data model.

Since a trajectory is a function from a time interval to the plane (or to space), the image of the function gives the locations where the entity is at the relevant times. The derivative of the function is the velocity, which is a vector, and the second derivative is acceleration, which is also a vector. Velocity has two components, namely speed and heading. These at any time measurable features (location, speed, heading, ...) of a trajectory are called *attributes*, and attributes are often the basis of measures defined over a whole trajectory. For example, from speed we can derive the measures *average speed*, *maximum speed*, *standard deviation of speed*, *percentage of stand-still*, and more. These measures give a single value for the whole trajectory.

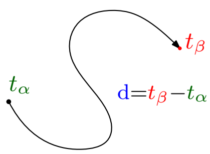
There are also attributes for trajectories amidst other trajectories. For example, for a given trajectory we have the attribute closest distance to another trajectory, which exists at any time. We can use this as the basis for a measure that intuitively corresponds to *degree of isolation* by averaging.

In this section we give an overview of various general-purpose measures that can be defined for a single trajectory. We first discuss measures for a single trajectory in isolation, then measures for a single trajectory amidst other trajectories, and then measures for a single trajectory in various contexts.

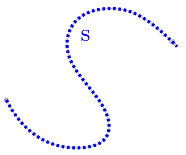
3.1.1 Measures for a Trajectory in Isolation

In this section we describe measures that relate to a single trajectory independent of other trajectories and the environment. Our list is not complete; we describe a number of existing useful, general-purpose measures. We begin with a number of basic, well-known measures capturing properties of the whole trajectory at once. Note that most of these common measures also appeared in similar lists of measures given in previous research [168, 93, 169, 9].

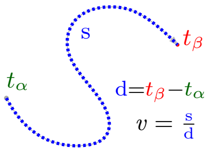
DURATION

Description	*The length of the time interval ($\mathcal{I}$) of the trajectory (d)*			
Unit	second	Range	$[0, \infty)$	
Derived from	-			
Related work	[170]			

TRAVERSED DISTANCE

Description	*The total length of the path of the trajectory (s)*			
Unit	meter	Range	$[0, \infty)$	
Derived from	location			
Related work	[25]			

AVERAGE SPEED

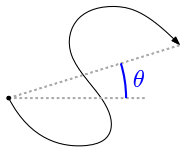
Description	*The ratio of the traversed distance and the duration (v)*			 s t_β t_α $\mathrm{d}=t_\beta-t_\alpha$ $v = \frac{\mathrm{s}}{\mathrm{d}}$
Unit	meter per second	Range	$[0, \infty)$	
Derived from	speed			
Related work	[171]			

While the average speed is a useful measure, it tells little about the variation in speed during the time interval. The entity could have been moving with constant speed equal to the average speed, or it could have stood still for half the time and then moved to the destination with speed double the average. Since this distinction is often important, we would like to have measures for it. The most obvious candidate is the standard deviation of speed over $\mathcal{I}$.

In principle speed is an attribute that has a value at any time in $\mathcal{I}$. To capture speed development over $\mathcal{I}$ in a more extensive manner, there are many options, both quantitative and qualitative. For example, we could split up $\mathcal{I}$ into k equal-duration subintervals and use the average speed in each of them. This may not be a good description of speed development because it might be more fine-grained than a chosen value of k. Ideally, one wants a partition of $\mathcal{I}$ into subintervals where the speed is similarly behaved. This can be obtained by segmentation.

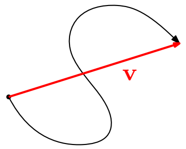
A derivative of speed is *acceleration* (and *deceleration*). Since acceleration exists at any time, we have a similar measure choices as for speed. For example, we can use the standard deviation of the acceleration to describe its variation.

GLOBAL DIRECTION

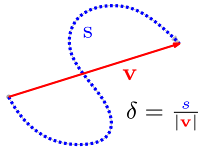
Description	*The direction of the vector from the origin to the destination of T (θ)*			
Unit	radian	Range	$[0, 2\pi]$	
Derived from	location			
Related work	[172, 173, 174]			

Also direction can be described more finely than with just a global direction, and as with speed, we can use subintervals of I to describe it. These subintervals may be obtained by equal durations or by segmentation.

GLOBAL VELOCITY

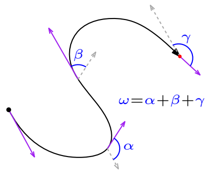
Description	*The vector from the origin to the destination of T divided by total duration ($\vec{v}$)*			
Unit	vector	Range	$\mathbb{R}^2$	
Derived from	velocity, or location and time			
Related work	[175]			

DETOUR

Description	*The ratio of the traversed distance and the length of the global velocity vector (δ)*			 s $\mathbf{v}$ $\delta = \frac{s}{\lvert\mathbf{v}\rvert}$
Unit	-	Range	$[1, \infty)$	
Derived from	-			
Related work	[165, 176]			

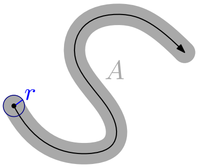
In other work, the notion of detour is described under different names, for example, sinuosity [9] and straightness index [17].

TOTAL ANGULAR CHANGE

Description	*The total change of the heading angle (ω)*			 γ β $\omega=\alpha+\beta+\gamma$ α
Unit	radian	Range	$[0, \infty)$	
Derived from	heading			
Related work	[170]			

The *total angular change* is a common shape descriptor that does not depend on time. It is the global version of the attribute representing the change in heading. It describes the shape in just one number on an angular scale.

AREA COVERED

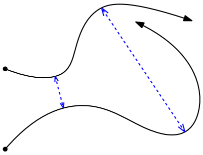
Description	*The area covered by a disc with radius r that moves along the trajectory (A)*			
Unit	square meter (m^2)	Range	$[\pi r^2, \infty)$	
Derived from	location			
Related work	[177]			

Finally, it may be useful to define the area covered by the trajectory. Andrienko et al. [9] describe this measure as the *spatial extent* of a trajectory, which can be defined in several ways. A simple definition is to use a distance parameter r and say that the moving entity covers the whole area within distance r from its location. The covered area is then the total swept area of a disk centered at the entity when it follows its path; multiply swept areas count only once.

3.1.2 Measures for a Trajectory Amidst Other Trajectories

Some trajectory measures require the presence of other trajectories, for example measures for notions like similarity and centrality. For now we only consider other trajectories that exist at the same time as the trajectory that we observe. However, it is also possible to take other trajectories into account that occur later or earlier, by adapting the measures.

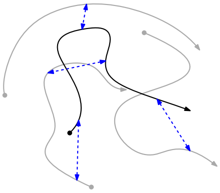
SIMILARITY

Description	*The average distance to another trajectory*			
Unit	meter	Range	$[0, \infty)$	
Derived from	-			
Related work	[70]			

Similarity has been studied extensively. It is a distance measure between two trajectories that can refer to average distance between the entities, maximum distance between the entities, or maybe just similarity in shape of the path of the trajectory. Well-known similarity measures for paths of trajectories are the Hausdorff distance [67] and the Fréchet distance [68]. Well-known similarity measures for trajectories are the time-focused distance [70], the dynamic time-warping distance [69], and the edit distance [71]. All of these similarity measures take the location into account. However, one can easily define speed similarity or acceleration similarity if location is not relevant in the application. Another similarity measure, introduced for shapes, is the turn function similarity. It is rotation-invariant, and therefore particularly useful for comparing shape only.

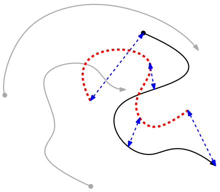
In this work, we define the similarity as the average distance between two trajectories during their time interval. To get the average distance, we can take the integral of the distance over the time interval and divide by the duration of the trajectories. Alternatively, one can take other possible options such as taking the minimum (or maximum) value, or integrating over space (reparameterize the trajectory).

CLOSENESS

Description	*Average distance to the nearest other trajectory at each time*			
Unit	meter	Range	$[0, \infty)$	
Derived from	distance to the closest trajectory			
Related work	[178]			

The closest entity to a single moving entity at a single time can change over its time interval. Therefore, the average distance can show how close a trajectory is to other trajectories.

CENTRALITY

Description	*Average distance to the central position of several trajectories*			
Unit	meter	Range	$[0, \infty)$	
Derived from	centrality			
Related work	[159, 168]			

Centrality is also an intuitive concept that allows various different definitions. The centrality of an entity amidst other entities can change over time, so we first consider centrality measures at a fixed time. In other words, we have a set of points in the plane and one specific point p, and we want to describe how central p is. One option is to define the center of mass as the most central location, and use the distance of p to the center of mass to define the centrality of p. We can do something similar with the median location, whose coordinates are the median x-coordinate and the median y-coordinate of all points. Another option is to use the distance to the center of the smallest enclosing circle of the points. For one trajectory, we take the average of centrality over its time interval, which is defined by an integral. Similar with the two previous measures, other options than taking the average are also possible.

A more combinatorial notion of centrality is the minimum number of points of the set that must be removed to bring p to the convex hull of the points.

Discussion. Other notions that can be turned into measures are isolation and sociality. To define these, we could use nearest-neighbor distance or k-nearest neighbor distances.

3.1.3 Measures for a Trajectory in an Environment

Since the environment influences how an entity moves, it is natural to consider trajectories in the context of other data. Other data can be internal or external, that is, it can refer to the moving entity or to its environment. Depending on the source of the data, internal attribute data can be heart rate, brain activity, CO_2 level in exhaust, and produced noise. External attribute data can be current land cover, elevation, weather, quality of road, water currents, and visibility information.

Environmental factors can influence how the measures defined before may be redefined. For example, two trajectories at distance 80 meters in the open field can be deemed more similar than two trajectories at 60 meters, one of which is in the field and one in the forest. A speed of $10\,m/s$ in the open field may be considered similar to a speed of $8\,m/s$ in bushland.

Moreover, the environment may give rise to measures that do not exist without it, like a measure for the diversity of landcover types crossed by the moving entity. Here, the locations of the moving entity is used to select other data (the environment) to which the measure relates.

The definitions of measures for trajectories in environments are very much application-dependent, and therefore we omit further discussion in this chapter that aims to take a general view on measures for trajectories and groups.

3.2 Measures for a Group of Trajectories

When sufficiently many moving entities are close enough for a relatively long period of time, they form a *group*. Closely related concepts are flocks [111], swarm [121], herds [118], convoys [81], and many more. Here, we assume that a group is a fixed set of entities during a fixed period of time. Intuitively, the existence of a group relies on some conditions, such as a minimum number of entities in it, a minimum entity inter-distance, a minimum duration, etcetera. It is possible to add further requirements to a group definition, like similarity of heading of the group entities.

Just like for a single trajectory, many measures exist for a group of trajectories. Some of these are direct extensions from the single trajectory case, while others only occur for a group. In general, there are three views of treating a group when defining these measures:

Representative view: A single trajectory (not necessarily from the group) is used to define the measure. For example, we can take the mean location at every time and use this to define a mean trajectory. This mean trajectory can then be used to define group speed, for instance.

Complete view: All entities in the group are used to define the measure. For example, to define group speed in a different way than in the representative view, we can take the average speed of each entity over the group duration, and average this over the entities.

Area view: The area that the group occupies is used to define the measure. For example, to define group density, we could define the area that the group occupies and let the density be the ratio of the number of entities and the area.

Not all views make sense for all measures we will define. Often, these different views give rise to different possible measures for the same concept, like for our example of group speed. The representative view gives us a way to generalize all single trajectory measures to groups.

In the following subsections we define group measures that do not require other input, group measures that exist for a group amidst other groups of moving entities, and group measures that require other types of input. The measures exist for any definition of a group, assuming that the number of entities in the group does not change in the interval during which this group exists. We also assume that the group exists during exactly one time interval, so we do not allow a group to form, break up, and later form again and call it the same group. Both assumptions can be lifted if required by the application at hand; they are made for the sake of clarity of the definitions of the measures.

3.2.1 Measures for a Single Group in Isolation

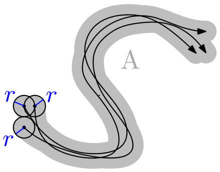
Measures extended from a single trajectory

Almost all measures for a single trajectory in isolation can be extended directly to a group. Both the representative view and the complete view work well with these measures (except for the covered area). For example, the traversed distance of a group can be defined in two ways: as the average of the traversed distance of all entities in the group or as the traversed distance of a representative.

Furthermore, each of these two views can have more variations. With the complete view, instead of taking the average of the measure from each trajectory, we can also choose to take, for example, either the minimum or the maximum value. There are also several possibilities to get a representative trajectory. For example, to define the representative trajectory, we can take the mean or median point of all entities in a group at a single time, or just pick one trajectory to represent the group.

Only one measure for a single trajectory cannot be extended directly to a group:

AREA COVERED

<table>
<tr><td>Description</td><td colspan="3">*The union of the total area covered by each trajectory (using a disc of radius r) in a group (A)*</td><td rowspan="4">
</td></tr>
<tr><td>Unit</td><td>square meter (m^2)</td><td>Range</td><td>$[\pi r^2, \infty)$</td></tr>
<tr><td>Derived from</td><td colspan="3">location</td></tr>
<tr><td>Related work</td><td colspan="3">[176]</td></tr>
</table>

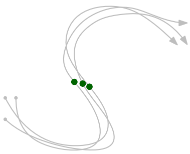
Clearly, we cannot sum up the covered area from all entities in a group because they may overlap. For the same reason, it is also not possible to aggregate the covered area by a group at each single time. Therefore, we define the area covered by a group as a union of the total area covered by its entities. Note that this measure relies on a parameter r to define the disc which is used to sweep the covered area.

The movement behavior of a group influences this measure. For instance, when entities in a group move by following each other (*single file behavior*), then the group covers roughly the same area as one of its entities. A more compact group will also yield a smaller value for this measure than a more spread out group.

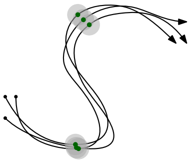
Measures that only exist for a group

In addition to attributes of a group that exist in a single trajectory, there are also attributes that only naturally exist when we observe multiple entities at a single time. For example, the number of entities in a group, (which is fixed for the whole duration of a group), or the *density* of a group.

SIZE

Description	*The number of entities in a group (n)*			
Unit	-	Range	$[2, \infty)$	
Derived from	the number of entities in a group			
Related work	[179]			

DENSITY

Description	*The average of the density of a group over its duration (ρ)*			
Unit	per square meter (m^{-2})	Range	$[\frac{2}{\pi r^2}, \infty)$	
Derived from	density			
Related work	[180, 179]			

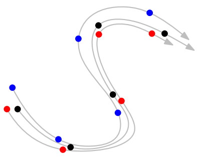
Intuitively, density relates a count to an area. Therefore, it is natural to define the density using an area view: a ratio of the size of a group and the covered area at a single time. Then, we integrate it over time to get a single value to represent the density of the group.

Other options to define the density of a group are based on the distance between entities in a group. We describe several possible definitions:

- The average of all distances between pairs of entities in a group.
- The weighted average of all distances, where small distances get a higher weight. Closer entities have more influence on the density than the farther ones.
- The ratio between the farthest distance occurring between two entities in the group and the maximum distance possible between two entities in any one group of the same size.

These distance-based measures have the advantage that they do not depend on a parameter to be determined.

FORMATION STABILITY

Description	*The average of the magnitudes of the velocity difference of each entity and the group (F)*			
Unit	meter per second	Range	$[0, \infty)$	
Derived from	velocity			
Related work	[181, 182]			

A group of moving entities may move in a mostly stable formation, for instance, the V-formation of migrating birds. Often, entities keep the same formation for several reasons, like increasing the efficiency of movement by using the least energy.

At a fixed time, we define this measure using the magnitude of the difference in velocity of each entity and the group. We can average these magnitudes over the group entities and over time. Lower values for this measure indicate that the group has a low deformation rate. The formula to compute F for a group $\mathcal{G} = \{e_1, e_2, \ldots, e_n\}$ over the duration $[t_\alpha, t_\beta]$ is:

$$F = \frac{1}{t_\beta - t_\alpha} \int_{t_\alpha}^{t_\beta} \frac{1}{n} \sum_{i=1}^{n} \|\vec{e}_i(t) - \vec{\mathcal{G}}(t)\| dt$$

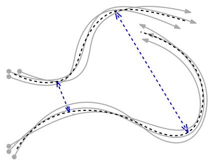
where $\vec{e}_i(t)$ is the velocity vector of entity e_i at time t and $\vec{\mathcal{G}}(t)$ is the velocity vector of $\mathcal{G}$, which is the average over all of its entities at time t.

Discussion. Many more application-specific group measures are conceivable. For instance, the occurrence of a *leader*. A leader can be defined as a single entity that is usually in front of the rest of the entities in the group, or as an entity whose movement drives the movement of the group [176, 104]. Some groups are more clearly led than others, which can be captured in a measure. Because such measures depend more on behavior interpretation than a clear objective aspect of groups, we will not suggest any formal definition in this chapter.

3.2.2 Measures for a Group Amidst Other Groups

Most of the measures for a group among other groups can be directly defined using the representative trajectory of each group and apply to measures for a trajectory amidst other trajectories.

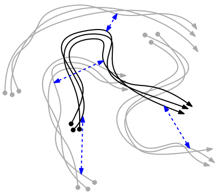
Similarity

Description	*The average distance to another group*			
Unit	meter	Range	$[0, \infty)$	
Derived from	location			
Related work	[183]			

To define similarity between two groups of entities we need to be aware of a few issues: (i) The two groups may have different size, and we still want a similarity measure that makes sense, and (ii) The trajectories in one group may be dissimilar among each other, and if this occurs in the same way in the other group, then the groups are similar. These considerations suggest a many-to-many matching-based definition. For each trajectory in each group, consider the similarity to the most similar trajectory in the other group, and average over these similarity values. If the one group has n trajectories and the other m, then we use $n + m$ similarity values to average over. Alternatively, we could take all $n \cdot m$ pairs of trajectories, one from each group, and take their average, but this does not allow groups to be similar if their trajectories are not similar internally.

Alternatively, we can compare the similarity between the two groups at a single time and then integrate the result over the duration of the groups. If we ignore the temporal component, then the area view can be used to compare the shape of the groups by comparing the shape of their covered area for a group shape similarity measure.

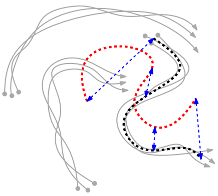
Closeness

Description	*The average distance to the nearest other group at each time*			
Unit	meter	Range	$[0, \infty)$	
Derived from	distance to the closest group			
Related work	–			

Although this measure is a straightforward extension from the measure for a single trajectory, applying it directly to the representative trajectories might not give the proper estimation about how close the two groups are. Intuitively, a group is near to the other groups if one or more entities in the group is close to entities in other groups. However, because the representative trajectory is located roughly in the middle of other entities in the group, there must be other pair of entities (each from different groups) that are closer.

Since the nearest other group might change over the duration of the group, it is better to first define the distance between two groups at a fixed time. Then, we can pick the minimum distance to one other group and get the average of this value over the whole duration of the group. At a single time, the distance between two sets of points can be defined in several ways. The simplest is to take the closest distance from any entity in one group to any entity in the other. Other options include the Hausdorff Distance [67] and the Earth Mover's Distance [184]. The question is whether we consider two groups to be very close when they are completely interspersed or when they move closely side by side as a whole group.

CENTRALITY

Description	*The average distance to the central position of other groups*			
Unit	meter	Range	$[0, \infty)$	
Derived from	centrality			
Related work	–			

Centrality can be defined using the representative trajectories and then apply the centrality measure for a single trajectory. We can also use the complete view of a group by computing the centrality of each entity w.r.t the central position of all entities and then average them over time to get the centrality of the group.

Discussion. So far in this section, we only considered measures that assume all groups start and end at the same time. It is also possible to define measures if we drop this assumption. For example, we can measure how many other groups exist (on average) during the lifetime of a group, and we can measure how similar a group moves with respect to any previously existing group.

3.2.3 Measures for a Group of Trajectories in an Environment

Similar to the case for a single trajectory, we can consider environmental factors when defining the measure of a group. For example, entities in a group may change their formation because they are moving in a single file formation while crossing a river, and may change back to the previous formation on the opposite bank. In this case, we probably want to say that the formation is consistent if the formation is the same in each terrain type. We cannot use the formation stability measure to capture this.

Different types of moving entities show different behavior when moving as a group, and this is influenced by different types of external factors. Consequently, integrating those factors into measures for a group depends on the type of moving entities that we want to analyze. Therefore, measures that take external factors into account are application-dependent, and general purpose measures are less easily defined.

3.3 Applications of Group Measures

With the vast increase of trajectory data, large collections of trajectories must be organized well and good tools for selection, visualization, and analysis must be available. We discuss how group measures may be used for these tasks. We assume that a group structure exists and has been computed.

Selection. To explore a collection of trajectories organized in groups, we may want to select the ten or twenty most important or characteristic groups. What is important about a group is application-dependent and can be discussed, but generally one can say that large groups with a long duration are more important than small groups with a short duration. Any of the other attributes can contribute to the importance of a group too: perhaps fast-moving groups (high average speed) or groups with a large area covered are important. The more measures play a role, the more dimensions the Pareto-optimal front has, and the more Pareto-optimal choices there are.

While the single most important group may be defined by weighing the different measures that contribute to importance, selecting a set of ten most characteristic and important groups is more complex. In many applications the selection should be *varied*, implying that choosing one group in the set influences which other groups should also be chosen. The corresponding issues have been studied in the classical problem of *settlement selection* [185, 186]: not necessarily the ten biggest cities should be chosen, they should also be spatially distributed over the region of interest. Various models for such selections have been suggested in the literature. Transferring these issues to groups in a set of trajectories, it may be important to get variation in where in the plane (in space) the group occurred. If the six biggest and longest duration groups all occurred in the west, the next six ones occurred in the east, and we wish to select six groups, then it would probably be undesirable to select all groups in the

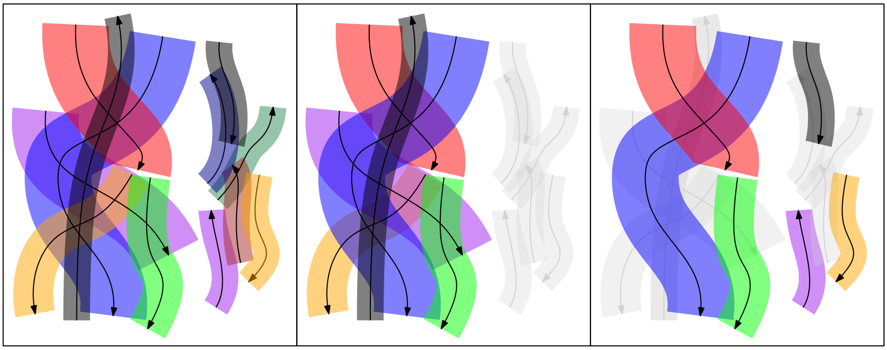


FIGURE 3.1: (Left) Each group is represented using a single black trajectory and an area with different colors. The width of the area shows the size of the group. (Middle) Only the six longest and biggest groups are selected, shown in color. (Right) A better selection of six groups for diversity of the groups with respect to location.

west; instead, three or four in the west and three or two in the east gives a better idea of the data (see Figure 3.1).

While a full treatment on when groups are important, salient or characteristic, and when a selection is an appropriate selection is beyond the scope of this work, it is clear that group measures do play an important role in the process.

Visualization. Visualization of large collections of trajectories is difficult for several reasons. First of all, it is not easy to show time in an intuitive manner, besides animations. Second, simply showing all paths of trajectories usually creates a chaos in which little can be seen. If the trajectories come in groups, it is attractive to identify these groups and use visual encodings of relevant attributes or measures. A representative can be used to show each group (determined by trajectory measures), the group size can be visualized by line thickness of the representative, and the group speed or density can be visualized by color or other visual variable.

Segmentation. The segmentation of a single trajectory has been discussed in several papers [55, 165, 58], but it may be more appropriate to *segment a group* in case the group as a whole shows certain behavior types. A herd of bison may be roaming, migrating, or sleeping. Segmentation is typically determined by within-segment similarity and across-segment dissimilarity and hence it relies on suitable measures. How to extend trajectory segmentation to trajectory group segmentation is an interesting research question in which it is clear that group measures will play an important role.

3.4 Conclusions and Future Work

In this chapter, we discussed measures for a single trajectory and a group of trajectories arising from moving entities. These measures provide extra information than just the spatial and temporal component needed to be able to analyze the trajectory data better. First, we gave basic measures for a single trajectory and differentiated them into three types: measures for a single trajectory, measures for a trajectory amidst other trajectories, and measures for a trajectory in the context of other data.

For a group of trajectories, we introduced three different views to define measures for a group: the representative view, the complete view and the area view. Most measures for a single trajectory can be extended directly to groups using at least one of the three views. We also introduced exclusive measures for a group like density and formation stability. For measures for a group amidst other groups, we discussed their differences with the same measure for a single trajectory and gave alternative approaches to define them.

Finally, we considered several tasks where group measures are essential to analyze collections of trajectory data.

Our discussion of measures gives rise to various directions of future work. Firstly, the three cases: selection, visualization, and segmentation of groups have only been addressed briefly, and a complete study of each of these tasks would be valuable. Secondly, we may examine the measures further on robustness to "outliers" within a group. Finally, measures need to be computed by algorithms, which in several cases still need to be developed.

4

A REFINED DEFINITION FOR GROUPS

Movement data may consist of just one trajectory tracked over a period of time, or a whole collection of trajectories that are all tracked over the same time period. Note that for the latter case, the locations of each trajectory are not necessarily recorded at the same time stamps. It is common to denote the number of trajectories (or moving entities) by n and the number of time stamps used for each trajectory by τ. Thus, the input size is $\Theta(\tau n)$ and depending on the application, one of n or τ can be much larger than the other.

To analyze movement data, a number of methods have been developed in recent times. These methods perform similarity analysis or compute a clustering, outliers, a segmentation, or various patterns that may emerge from the movement of the entities (see Section 1.3). These methods are often based on geometric algorithms, because the data is essentially spatial.

In this chapter we present the results of one of the important tasks in the analysis of movement data collected from moving entities: finding a collective movement pattern (or group). This pattern occurs when a set of entities travel together for a sufficiently long period of time. The informal description above suggest three important parameters to define groups:

- the *size* parameter for the minimum number of entities in a group,
- the *temporal* parameter for the duration of a group, and
- the *spatial* parameter for the distance between moving entities, which is used to define the togetherness between entities.

There are various formal definitions of groups such as flocks [111, 107, 112, 113], convoys [81, 119, 120], swarms [121, 122], groups [126, 127], and many more (see Section 1.3.3). We continue the study of such groups and propose a refined definition to the one by Buchin et al. [126].

Buchin et al. [126] introduce a model called the *trajectory grouping structure* which not only defines groups, but also the splitting of a group into subgroups and the merging of groups. The algorithmic problem of reporting all maximal groups (we give a formal definition of maximal groups later) that occur in the trajectories is solved in $O(\tau n^3 + N)$ time, where $N \in O(\tau n^4)$ is the output size (the summed size of all groups reported). The algorithm also considers times in between the τ time stamps where the locations are recorded as relevant. In between these time stamps, locations are inferred by linear interpolation over time. We refine the definition of groups by Buchin et al. [126] and motivate why our proposed definition corresponds better to human intuition in certain cases, particularly in dense environment. We also present algorithms to compute all maximal groups based on the refined definition.

Previous definition of a group. The definition of a group by Buchin et al. [126] relies on three parameters: one for the distance between entities, one for the duration of a group, and one for the size of a group. We review their definitions next.

For a set of moving entities $\mathcal{X}$, two entities x and y are *directly ε-connected* at time t if the Euclidean distance between x and y is at most ε $(\varepsilon > 0)$ at time t, $\mathrm{d}_{xy}(t) > \varepsilon$. Two entities x and y are *ε-connected in $\mathcal{X}$* at time t if there is a sequence $x = x_0, ..., x_k = y$, with $\{x_0, ..., x_k\} \subseteq \mathcal{X}$ and for all i, x_i and x_{i+1} are directly ε-connected at time t.

A *group* for an entity inter-distance ε, a minimum required duration δ, and a minimum required size m, is defined as a subset $\mathcal{G} \subseteq \mathcal{X}$ and corresponding time interval $\mathcal{I}$ for which three conditions hold [126]:

(i) $\mathcal{G}$ contains at least m entities.
(ii) $\mathcal{I}$ has a duration at least δ.
(iii) Every two entities $x, y \in \mathcal{G}$ are ε-connected in $\mathcal{X}$ at all times in $\mathcal{I}$.

Furthermore, a group $\mathcal{G}$ with time interval $\mathcal{I}$ is *maximal* if there is no time interval $\mathcal{I}' \supset \mathcal{I}$ for which $\mathcal{G}$ is also a group, and there is no proper superset $\mathcal{G}' \supset \mathcal{G}$ that is also a group during $\mathcal{I}$ [126].

Refined definition of a group. One issue with the previous definition is that it does not correspond fully to our intuition. Two entities x and y may form a (rather small) maximal group in an interval $\mathcal{I}$ even if they are always far apart, as long as there are always entities of $\mathcal{X}$ in between them to make x and y ε-connected in $\mathcal{X}$. These entities in between are not part of the

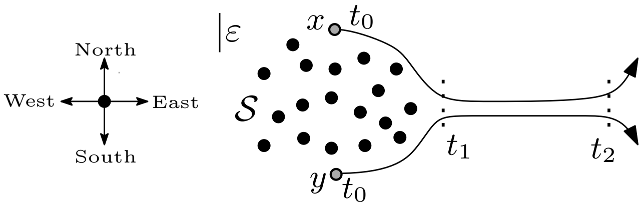


FIGURE 4.1: In the previous definition of groups by Buchin et al. [126], x and y are ε-connected during $[t_0, t_2]$.

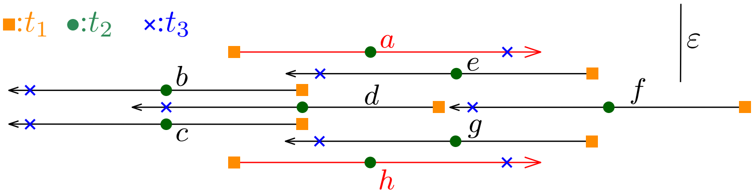


FIGURE 4.2: Entities in $\mathcal{G} = \{a, h\}$ are ε-connected using entities not in $\mathcal{G}$.

maximal group, but they do cause x and y to be ε-connected by the previous definition. This can have counter-intuitive effects especially in dense crowds. To avoid such issues, we refine the definition of a group. In particular, we replace condition (iii) above by:

(iii') Every two entities $x, y \in \mathcal{G}$ are ε-connected in $\mathcal{G}$ during $\mathcal{I}$.

We define *maximal* groups analogous to the previous definition by Buchin et al. [126], but now based on the refined definition of groups.

We give two examples that show the difference in these definitions.

First, consider a number of stationary entities $\mathcal{S}$ and two entities x and y, see Figure 4.1. Entity x starts (at time t_0) to the North of $\mathcal{S}$ and moves around its perimeter to the East. Entity y starts (at t_0) to the South and also moves around the perimeter to the East. After encountering (at t_1) each other at the East side, both continue together eastward, away from the stationary entities in $\mathcal{S}$ (ending at t_2). By the previous definition of groups by Buchin et al. [126], x and y form a maximal group in the interval $[t_0, t_2]$. By our refined definition, they form a maximal group during $[t_1, t_2]$, starting when x and y are at distance ε and actually encounter each other.

Second, the previous definition can even see groups of entities that were never close, see Figure 4.2. Here, $\{a, h\}$ is a maximal group in the interval $\mathcal{I} = [t_1, t_3]$ using the definition in [126]. At each time, a and h are ε-connected, but through different subsets of entities. By choosing the coordinates carefully, we can ensure that no supergroup of $\{a, h\}$ is also a group in the same time interval, and hence $\{a, h\}$ will be maximal. Although a and h move in the same direction with the same speed, intuitively they do not form a group because they are too far apart and separated by other entities that move in the opposite direction. With our refined definition, we do not consider $\{a, h\}$ a group in the interval $\mathcal{I}$, and hence also not a maximal group.

Results and Organization. We have refined the previous definition for a group of moving entities by Buchin et al. [126] and argued why our refined definition can give an intuitively plausible group. From now on, we will use the term “group” to denote a group of entities that comply with our refined definition.

In the following section, we show that for a set $\mathcal{X}$ of n moving entities in $\mathbb{R}^1$ with τ time stamps each, the number of maximal groups by the refined definition is $O(\tau n^3)$, which is tight in the worst case.

In Section 4.2, we present algorithms to compute all maximal groups in $\mathbb{R}^1$. First we consider the case where all trajectories have their vertices at the same time and begin with a basic algorithm for that runs in $O(\tau^3 n^6)$ time. Subsequent improvements lead to a running time of $O(\tau^2 n^4)$. When the time stamps of different trajectories are not the same, our algorithm runs in $O(\tau^2 n^4 \alpha(n))$ time.

Next, for moving entities in $\mathbb{R}^d$ $(d > 1)$, we model entities and their interdistance into graphs and show that all maximal groups can be computed in $O(\tau^2 n^5 \log n)$ time, regardless the uniformity of the time stamps in the trajectories. We show how to achieve this bound in Section 4.3.

In Section 4.4, we consider situations where the value of n is significantly smaller than τ, which is typical in real-life moving entity datasets. We give an $O(\tau 2^n n^4)$ time algorithm for entities that move in any constant dimension.

Finally, we show an exponential bound on the number of maximal groups that can contain any given time t in the last section.

4.1 Preliminaries

Let $\mathcal{X}$ be a set of n entities moving in $\mathbb{R}^1$, given by locations at τ time stamps. A trajectory of an entity in $\mathcal{X}$ can be expressed by a piecewise-linear function which maps time to a point in $\mathbb{R}^1$. If $\mathbb{R}^1$ is associated with the vertical axis and time with the horizontal axis of a 2-dimensional plane, the trajectories of entities in $\mathcal{X}$ are polylines with τ vertices each. We will use the same notation to denote an entity and its trajectory.

When $\mathrm{d}_{ij}(t) = \varepsilon$, we say that an *$\varepsilon$-event* occurs. For any ε-event v, we denote by t_v the time when v occurs and $\omega(v)$ the function that returns the two entities that create v. We assume that no two or more ε-events occur at the same time.

Consider an ε-event v; let $\omega(v) = \{i, j\}$. If the distance between i and j is more than ε immediately before t_v, then v is a *start ε-event*; if this happens after t_v, then v is an *end ε-event*. If there is no entity $k \in \mathcal{X}$ located strictly in between i and j at t_v (so $\mathrm{d}_{ik}(t_v) + \mathrm{d}_{jk}(t_v) = \varepsilon$), then we say that v is a *free ε-event*.

For clarity, we assume that there are no two parallel edges of trajectories with distance ε. This assumption can be lifted without complications; we just need to handle such situations in the appropriate manner.

Observation 12. *The number of ε-events is $O(\tau n^2)$.*

Let $\mathcal{G}$ be a group of entities in time interval $\mathcal{I}$ that is maximal in size. All entities in $\mathcal{G}$ are pairwise ε-connected in the interval $\mathcal{I}$, and hence, there are no free ε-events in $\mathcal{G}$ during $\mathcal{I}$. In the arrangement of trajectories from $\mathcal{G}$, we define the height of a face as the length of the longest vertical line segment inside the face. Thus, no face has height greater than ε.

It is also clear that $\mathcal{G}$ can begin only at a start ε-event and end only at an end ε-event. Furthermore, we observe that if a start ε-event (or end ε-event) of $\mathcal{G}$ is not a free ε-event with respect to the entities in $\mathcal{G}$, then before (or after) the interval $\mathcal{I}$, entities in $\mathcal{G}$ are still pairwise ε-connected and we can extend the interval of $\mathcal{G}$. Therefore, $\mathcal{G}$ can be a maximal group only if both the start ε-event and end ε-event are free ε-events (but this is not a sufficient condition).

Observation 13. *There can be at most one maximal group that starts and ends at a particular pair of start ε-event and end ε-event.*

Theorem 14. *For a set $\mathcal{X}$ of n entities, each entity moving along a piecewise-linear trajectory of τ edges, the maximum number of maximal groups is $\Theta(\tau n^3)$.*

Proof. Any group $\mathcal{G}$ that starts at a start ε-event contains at most n entities. When a free end ε-event involving $\mathcal{G}$ occurs, only group $\mathcal{G}$ ends but a subgroup of $\mathcal{G}$ with fewer entities may continue longer. This can happen at most $n - 1$ times. Therefore, the maximum number of maximal groups is $O(\tau n^3)$.

For the lower bound on the number of maximal groups, we use the same construction given by van Goethem et al. [187]. Although they use the previous definition by Buchin et al. [126] to define a group, the construction itself also works for our refined definition of a group. They start by building a construction in which all n entities move along lines and show that there can be $\Omega(n^3)$ maximal groups. Then, repeating the construction $\Omega(\tau)$ times gives $\Omega(\tau n^3)$ as a lower bound on the number of maximal groups. $\square$

The approach to compute all maximal groups is to work on the arrangement $\mathcal{A}$ of line segments that are the trajectories. For a subset $\mathcal{G} \subseteq \mathcal{X}$ and interval $\mathcal{I}$, we can remove entities from $\mathcal{G}$ that are separated at a face with height larger than ε in $\mathcal{I}$ (corresponding to a free ε-event). Only if there are no such faces, the remaining entities in $\mathcal{G}$ can be a group. Note that removing entities in $\mathcal{G}$ involves removing the corresponding trajectories from the arrangement $\mathcal{A}$, which can cause new faces that are free ε-events.

4.2 Algorithms for Entities in $\mathbb{R}^1$

In this section, first we consider the case where the trajectories have the same time stamps. We present a basic algorithm that computes all maximal groups in $O(\tau^3 n^6)$ time for entities moving in $\mathbb{R}^1$. Then we present a more efficient algorithm that runs in $O(\tau^2 n^4)$ time. Furthermore, we present an $O(\tau^2 n^4 \alpha(n))$ time algorithm if the vertices of the trajectories have different time stamps.

4.2.1 Basic Algorithm

We describe a simple algorithm to compute all maximal groups. Let $\mathcal{V}_s$ and $\mathcal{V}_e$ be the sets of all start ε-events and all end ε-events, respectively. Fix one event of each type: $\beta \in \mathcal{V}_s$ and $\gamma \in \mathcal{V}_e$. By Observation 13, at most one maximal group $\mathcal{G}$ that starts at β and ends at γ. Furthermore, observe that $\mathcal{G}$ necessarily contains the entities $\omega(\beta) = \{a, b\}$ and $\omega(\gamma) = \{c, d\}$, and that if $\mathcal{G}$ is a maximal group on $\mathcal{I} = [t_\beta, t_\gamma]$, then all entities in $\mathcal{G}$ are on the same side at time $t_\lambda \in (t_\beta, t_\gamma)$ when a free ε-event λ occurs. We then use the following approach to find $\mathcal{G}$ (if it exists):

1. Initialize a set $\mathcal{G}$ containing all entities in $\mathcal{X}$.
2. Build an arrangement $\mathcal{A}$ induced by the trajectories of the entities in $\mathcal{G}$ on $\mathcal{I}$.
3. A face f in $\mathcal{A}$ contains a free ε-event λ if (and only if) the height of f is more than ε. If f has height larger than ε, test if (the trajectories of) a, b, c, and d, all lie on the same side of f. If not, there is no maximal group $\mathcal{G}$ that starts at β and ends at γ. If they do pass on the same side, let $\mathcal{S}$ denote the set of entities whose trajectories lie on the other side of f. Remove these entities of $\mathcal{S}$ from $\mathcal{G}$, and remove their trajectories from $\mathcal{A}$. Observe that new free ε-events may appear because removal of a trajectory from $\mathcal{A}$ merges two faces of $\mathcal{A}$ into a larger one. See Figure 4.3. Repeat this step until there is no more free ε-event λ with $t_\lambda \in (t_\beta, t_\gamma)$.

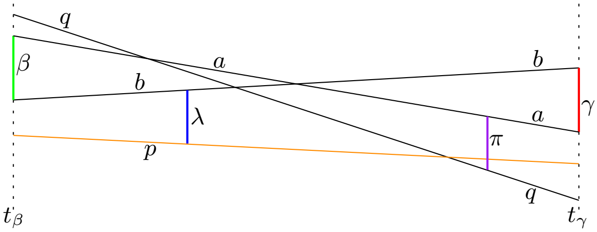


FIGURE 4.3: Removing trajectory p (due to the free ε-event λ) causes the ε-event π to become a free ε-event.

4. Check that β and γ are now free. If so, $\mathcal{G}$ is a maximal group on $\mathcal{I}$, and hence we can report it. If not, $\mathcal{G}$ is actually a group during a time interval $\mathcal{I}' \supset \mathcal{I}$. Hence, $\mathcal{G}$ may be maximal in size, but not in duration. We do not report $\mathcal{G}$ in this case.

Theorem 15. *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^3 n^6)$ time using the Basic Algorithm.*

Proof. The number of combination of a pair of start and end ε-events is $O(\tau^2 n^4)$. Building an arrangement from trajectories of entities takes $O(\tau n^2)$ time. Removing a trajectory e and checking new faces in $\mathcal{A}$ takes time proportional to the zone complexity of e: $O(\tau n)$. Since there are at most n trajectories to be removed, the whole process to remove entities for each interval $\mathcal{I}$ takes $O(\tau n^2)$ time. Therefore, the running time of the algorithm is $O(\tau^3 n^6)$ time. $\square$

4.2.2 Improved Algorithm

The previous algorithm checks every pair of possible start and end ε-events β and γ to potentially find one maximal group. To improve the running time, we fix a start ε-event β and consider the $O(\tau n^2)$ end ε-events γ in increasing order. We show that we can check for a maximal group on $[t_\beta, t_\gamma]$ in amortized $O(1)$ time.

We build the arrangement $\mathcal{A}$ for all trajectories, starting from time t_β, and sort the end ε-events γ, with $t_\gamma > t_\beta$ on increasing time. We then consider the end ε-events γ in this order, while maintaining a maximal set $\mathcal{G}$ that is ε-connected in $\mathcal{G}$ throughout the time interval $[t_\beta, t_\gamma]$.

Let $\omega(\beta) = \{a, b\}$ be the entities defining the start ε-event β, and let $\mathcal{G} \supseteq \{a, b\}$ be the largest ε-connected set on $[t_\beta, t_\gamma]$. We compute the largest ε-connected set on $[t_\beta, t_{\gamma'}]$ for the next ending event γ' as follows. Note that this set will be a subset of $\mathcal{G}$.

Let $\mathcal{S}$ be the set of entities that separate from a and b at γ. We remove all trajectories from the entities in $\mathcal{S}$ from $\mathcal{A}$. As before, this may introduce faces of height larger than ε. For every such face f, we check if a and b still pass f on the same side. If not, there can be no maximal groups that contain a and b, start at t_β, and end after t_γ. If a and b lie on the same side of f, we add all entities that lie on the other side of f to $\mathcal{S}$ and remove their trajectories from $\mathcal{A}$. We repeat this until all faces in $\mathcal{A}$ that have non-empty intersection with the vertical strip defined by $[t_\beta, t_{\gamma'}]$ have height at most ε (or until we have found a face that splits a and b). It follows that the set $\mathcal{G}' = \mathcal{G} \setminus \mathcal{S}$ is the largest set containing a and b that is ε-connected throughout $[t_\beta, t_{\gamma'}]$. If β and γ' are free with respect to $\mathcal{G}'$ then we report $\mathcal{G}'$ as a maximal group.

Building the arrangement $\mathcal{A}$ takes $O(\tau n^2)$ time, and sorting the ending-events takes $O(\tau n^2 \log(\tau n))$ time. By the Zone Theorem, we can remove each trajectory in $O(\tau n)$ time. Checking the height of the new faces can be done in the same time bound. It follows that the total running time is $O(\tau n^2(\tau n^2 + \tau n^2 \log(\tau n) + \mathcal{R}))$ where $\mathcal{R}$ is the total time for removing trajectories from the arrangement. Clearly, $\mathcal{R}$ is bounded by the complexity of the arrangement: $O(\tau n^2)$. So, the total running time is $O(\tau^2 n^4 \log(\tau n))$.

Further Improvement. We can avoid repeated sorting of end ε-events by pre-sorting them in a list, and for each start ε-event, use this list. The list will contain events that do not concern the entities involved in the start ε-event, but this can be tested easily in constant time. Thus, we conclude:

Theorem 16. *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^2 n^4)$ time.*

Next, we consider finding all maximal groups when the vertices of different trajectories do not have the same time stamps. We use the same idea as in the above algorithm: take one start ε-event β at a time and remove trajectories to find all maximal groups containing $\omega(\beta)$.

We use a similar strategy to split trajectories vertically into τ cells as in [188], where each cell now contains $O(n)$ segments of trajectories. It follows that the complexity of each cell is bounded by the number of possible intersections between segments: $O(n^2)$. Thus, building the arrangement $\mathcal{A}$ still takes $O(\tau n^2)$ time. However, by the Zone Theorem for an arrangement of line segments, removing a trajectory in each cell now takes $O(n\alpha(n))$ time [189], where $\alpha(n)$ is the inverse Ackermann Function. Therefore, the total time to remove trajectories in A is $O(\tau n^2 \alpha(n))$ time and we obtain:

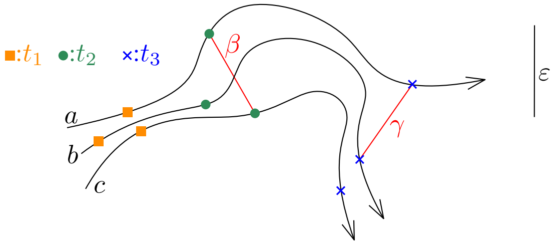


FIGURE 4.4: At t_2, β is not a free ε-event because a and c are still ε-connected through b. At t_3, a and b create the free ε-event γ.

Theorem 17. *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^1$ along a trajectory of τ edges under the condition that their vertices have different time stamps, all maximal groups can be computed in $O(\tau^2 n^4 \alpha(n))$ time.*

4.3 Algorithms for Entities in $\mathbb{R}^d$

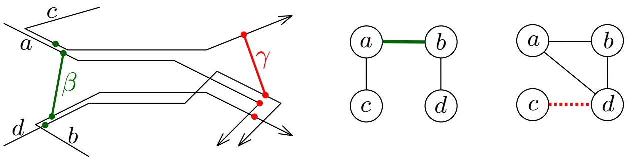
In $\mathbb{R}^d$ ($d > 1$), it is harder to test whether an ε-event really connects or disconnects because the two entities may be ε-connected through other entities in the group, see Figure 4.4. This observation immediately gives the condition for an ε-event to be *free*.

We model our moving entities as a graph where vertices represent entities and an edge exists if two entities are directly ε-connected. As shown by Parsa [148], we can maintain the graph under edge updates, while allowing same component queries, in $O(\log n)$ time per operation.

To compute maximal groups, we start at a start ε-event β, where $\omega(\beta) = \{a, b\}$, and maintain the connected component $\mathcal{C}$ throughout the sequence of sorted ε-events. At each ε-event γ, we remove any vertices that are disconnected from $\mathcal{C}$ and start again from β in case we remove anything. We stop if a and b are disconnected. If β is a free ε-event when we reach γ again, we report $\mathcal{C}$ as a maximal group and continue. See Figure 4.5 for an example of how the algorithm works.

We start at $O(\tau n^2)$ ε-events and for each, we process $O(\tau n^2)$ ε-events. We may need to restart this process up to $n-1$ times. In $\mathbb{R}^d$, our approach only examine the ε-events of entities and is not affected by whether the vertices of trajectories have the same time or not, therefore we obtain the same result for both cases:

Theorem 18. *Given a set $\mathcal{X}$ of n entities moving in $\mathbb{R}^d$ along a trajectory of τ edges, all maximal groups can be computed in $O(\tau^2 n^5 \log n)$ time.*



(i) (ii) (iii)

FIGURE 4.5: Two ε-events β and γ. (i) The maximal group $\{a, b, c, d\}$ has a time interval $[t_\beta, t_\gamma]$. (ii) Connectivity of the graph after t_β. A new edge appeared connecting a and b. (iii) Connectivity of the graph after t_γ. The edge between c and d disappeared.

4.4 Algorithms with Linear Dependence on τ

In many real-life situations, GPS is used for collecting trajectory data from human, vehicle or animal movement. Since it may record the position of an entity frequently and over a period of time, the number of vertices in each trajectory is often much larger than the number of moving entities [9]. Therefore, the dependence of the algorithm on τ is more important than the dependence on n. Next, we present a simple algorithm that is linear in τ, at the cost of an exponential dependence on n. In particular, our algorithm will compute all maximal groups in $O(\tau n^4 2^n)$ time.

We consider all 2^n subsets of $\mathcal{X}$ in order of decreasing size, while maintaining the set of maximal groups found so far (ordered by increasing starting time). For each subset $\mathcal{G}$ we determine the maximal time intervals during which $\mathcal{G}$ is ε-connected, and for each such interval $\mathcal{I}$ we check if $\mathcal{G}$ is dominated by a maximal group $\mathcal{H} \supset \mathcal{G}$ on $\mathcal{I}$. If such a set does not exist, then $\mathcal{G}$ is a maximal group on $\mathcal{I}$. Notice that we only need to know when the start ε-event and end ε-event of a particular group occurs. Therefore, this algorithm can be applied for both cases where the vertices of trajectories have the same or different time stamps.

For each subset $\mathcal{G}$, we consider the ε-events generated by the entities in $\mathcal{G}$. We can compute all these $O(\tau n^2)$ ε-events in $O(\tau n^2 \log n)$ time, by sorting the batches of $O(n^2)$ ε-events between two consecutive time stamps separately, and concatenating the resulting τ lists. We then go through the ε-events in order, and check if $\mathcal{G}$ is ε-connected at every ε-event. We can easily handle every event in $O(n^2)$ time, by naively checking if the entities in $\mathcal{G}$ are ε connected (we can easily improve on this, but the total running time will be dominated by the number of sets anyway). It follows that we can compute the sequence $\mathcal{S}_\mathcal{G}$ of maximal time intervals on which $\mathcal{G}$ is ε-connected in $O(\tau n^4)$ time. Note that $\mathcal{S}_\mathcal{G}$ contains at most $O(\tau)$ such time intervals.

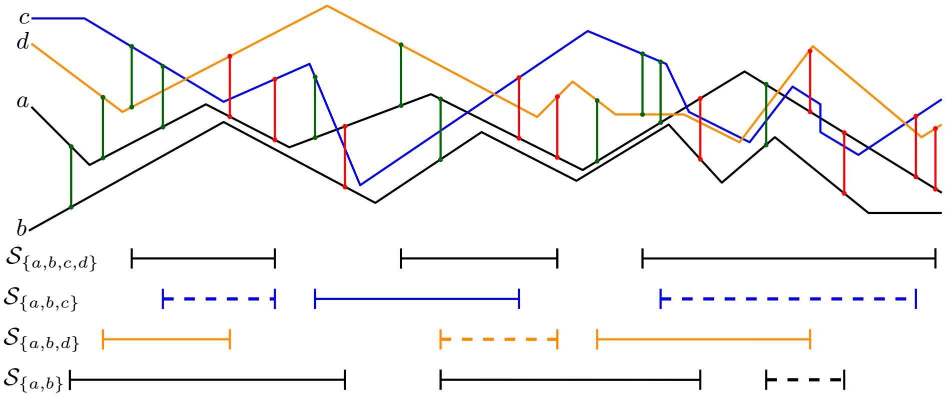

c
d
a
b
$\mathcal{S}_{\{a,b,c,d\}}$
$\mathcal{S}_{\{a,b,c\}}$
$\mathcal{S}_{\{a,b,d\}}$
$\mathcal{S}_{\{a,b\}}$

FIGURE 4.6: The sequences of maximal time intervals for $\mathcal{G} = \{a, b\}$ and all of its supersets. The three time intervals at the bottom are found when processing $\mathcal{G}$; the other ones were found earlier. The dashed time intervals do not give rise to a maximal group.

For each interval $\mathcal{I}$ in $\mathcal{S}_{\mathcal{G}}$ we now have to check if $\mathcal{G}$ is a maximal group during $\mathcal{I}$. The set $\mathcal{G}$ is a maximal group on $\mathcal{I}$ if and only if there is no maximal group $\mathcal{H} \supset \mathcal{G}$ on a time interval that contains $\mathcal{I}$. Since we maintain the maximal groups larger than $\mathcal{G}$ (and the time interval on which they are a maximal group), ordered by increasing starting time, we can iterate through them once, and extract the maximal groups that are a superset of $\mathcal{G}$. Figure 4.6 shows the sequence of maximal time intervals found by the algorithm when processing the entity set $\mathcal{G} = \{a, b\}$ and its supersets.

Since by Theorem 14 there are at most $O(\tau n^3)$ maximal groups, this takes at most $O(\tau n^4)$ time. Let $\mathcal{I}$ denote the set of time-intervals corresponding to those groups, ordered by increasing starting time. We now simply scan through $\mathcal{S}_{\mathcal{G}}$ and $\mathcal{I}$ simultaneously, while maintaining the time interval in $\mathcal{I}$ that started earliest and has not ended yet. For every interval $\mathcal{I}$ in $\mathcal{S}_{\mathcal{G}}$ we can then check if $\mathcal{G}$ is a maximal group on $\mathcal{I}$ in constant time. In total this takes $O(\tau n^3)$ time. Using a similar simultaneous scan we can add the intervals on which $\mathcal{G}$ is maximal to our set of maximal groups found so far.

It follows that we can compute all time intervals on which $\mathcal{G}$ is maximal in $O(\tau n^4)$ time. Since we do this for all subsets $\mathcal{G} \subseteq \mathcal{X}$ we obtain the following result.

Theorem 19. *Given a set $\mathcal{X}$ of n entities in which each entity moves in $\mathbb{R}^d$ along a trajectory of τ edges, we can compute all maximal groups in $O(\tau n^4 2^n)$ time, using $O(\tau n^3)$ space.*

4.5 A Lower Bound on the Maximum Number of Maximal Groups at some Time t

The result in the previous section shows that, when τ is large but n is small, we can improve the dependence on τ from quadratic to linear. However, we pay for this by having an exponential dependence on n. This naturally raises the question whether an algorithm with linear dependence on τ, but polynomial dependence on n, is possible. While we do not know the answer to this question, we present a construction which may indicate that such a result is hard to obtain, if possible at all.

We show that the number of maximal groups that contain a given time t can be exponential in n, provided that τ is sufficiently large. Without the requirement that the maximal groups must span a single moment in time, it is easy to make a construction of trajectories that has a number of maximal groups that is linear in τ, even with just two entities, so it is unbounded in n. Similarly, we can easily construct trajectories that give rise to $2^n - n - 1$ maximal groups (with a group size of at least 2) that are different in composition using roughly 2^n time stamps by making these groups consecutive. The construction that we present, where many different maximal groups occur simultaneously, is more involved, and shows that there may be $\Omega(\sqrt{2}^n)$ maximal groups simultaneously when there are $\Omega(\sqrt{2}^n)$ time stamps. While the result does not imply any lower bound for the problem of computing all maximal groups, it suggests that it may be difficult to obtain an algorithm that is linear in τ and polynomial in n. Several natural approaches to the problem (based on, for instance, divide-and-conquer) appear not to work due to this construction and the result on simultaneous maximal groups.

Theorem 20. *There exists a set $\mathcal{X}$ of n entities in $\mathbb{R}^1$ whose trajectories are defined by $\Theta(\sqrt{2}^n)$ time stamps, for which the number of maximal groups at some time t is $\Omega(\sqrt{2}^n)$.*

Proof. We use a set of $n = 2k+2$ entities, denoted $p_1, \ldots, p_k$, $p'_1, \ldots, p'_k$, and q and q'. We are interested in counting the groups that contain q, q', and for each i, exactly one of p_i and p'_i. We call any such group *k-max* and will show that they are all maximal. A k-max group $\mathcal{G}$ is encoded by a length-k bitstring where the i-th bit is 1 if $p_i \in \mathcal{G}$ and it is 0 if $p'_i \in \mathcal{G}$.

We make a construction with the following properties; the half after t_{mid} is illustrated for $k = 3$ in Figure 4.7:

1. The trajectory of p_i is the reverse of p'_i, with respect to t_{mid} (that is, mirrored in t_{mid}), and vice versa.

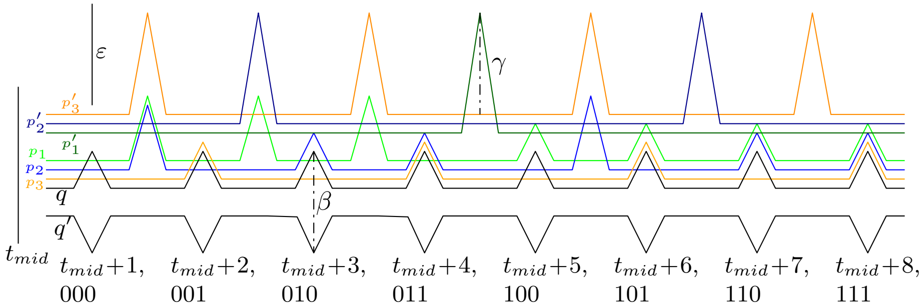


Figure 4.7: Right half of the lower bound construction for $k = 3$. The times very near the start and end ε-events of q and q' are shown together with the bitstring of the k-max group that ends there. At t_{mid}, all trajectories are in a single point (the trajectories are not shown near t_{mid}).

2. A k-max group starts and ends at free ε_q-events of q and q'.
3. A k-max group encoded by bitstring $\mathcal{B}$ starts a fraction after time $1 + \mathcal{B}$ and ends a fraction before time $t_{mid} + 1 + \mathcal{B}$, where $\mathcal{B}$ is interpreted as a binary number.
4. There are only $O(1)$ trajectory vertices of each trajectory within one time unit.
5. Each k-max group is maximal.

At t_{mid}, all trajectories pass through a single point to ensure they are continuous when mirroring, and they are pairwise directly ε-connected. It is the moment in time for which $\Omega(\sqrt{2}^n)$ maximal groups exist, as we will show. After t_{mid}, the entities q and q' will have 2^k pairs of ε-events: an end ε-event directly followed by a start ε-event. We call these events ε_q-events. Whether these ε_q-events are free for a k-max group $\mathcal{G}$ depends on the time and the bitstring, or equivalently, which entities from $p_1, \ldots, p_k$ are in $\mathcal{G}$.

The ε_q-event at a time t is free for a k-max group $\mathcal{G}$ if and only if the bitstring corresponding to time t is the same as the bitstring of $\mathcal{G}$. Hence, (assuming that no earlier ε-event ends $\mathcal{G}$) $\mathcal{G}$ will end at the time of its bitstring, so a fraction before $t_{mid} + 1 + \mathcal{B}$. By symmetry of p_i and p'_i, $\mathcal{G}$ will start a fraction after time $1 + \mathcal{B}$. In Figure 4.7, for example, at time $t_{mid} + 4$, the k-max group $\{p'_1, p_2, p_3, q, q'\}$ ends.

The other ε-events of the trajectories are between two consecutive ε_q-events. These ε-events involve the entities of $p'_1, \ldots, p'_k$ and the trajectories need three vertices between ε_q-events. Their presence ensures that only one of p_i or p'_i is in a particular k-max group. Notice that these ε-events will also be the start or end ε-events of maximal groups that are supersets of a k-max group.

Suppose p'_i is an entity that creates a free ε-event β just before a k-max group $\mathcal{G}$ containing p_i ends at $t_{mid} + \mathcal{B}$. Obviously, p'_i only needs to create such a free ε-event once and it follows this is only necessary if the previous k-max group $\mathcal{G}'$ that ends at $t_{mid} + \mathcal{B} - 1$ is not containing p_i. However, other k-max groups that will end after $\mathcal{G}$ might contain p'_i. Therefore, to prevent this ε-event becomes free in the duration of those k-max groups, we make entities of $p_1, \ldots, p_k$ that are not in $\mathcal{G}$ to keep p'_i ε-connected to all other entities. Still, not all of them are needed to prevent β to become free, but only for each entity p'_h where $h < i$, because by the ordering of the bitstrings, k-max groups contain p'_i and those entities might end after β. See Figure 4.7: before ε-event β, only p_1 prevents p'_2 from creating a free ε-event (but not p_3). Two k-max groups contain p_1,p'_2 and one of p_3 or p'_3 end after β while k-max group of $\{p'_1, p'_2, p_3\}$ ends before β.

Claim 21. If a maximal group containing time t_{mid} contains at least p_i or p'_i for all indices i, and both p_i and p'_i for at least one index i, then its time interval cannot contain both time h and $t_{mid} + h$ for any integer h.

Proof. Suppose for contradiction $\mathcal{G}$ is a maximal group which contains both p_i and p'_i, and its time interval fully contains an interval $[h, t_{mid}+h]$ for some integer h; suppose further that i is the smallest index for which this is the case. Let $\mathcal{B}$ be the bitstring that encodes the entities with indices $1 \ldots i-1$; let $\mathcal{B}^- = \mathcal{B}0111\ldots1$ be obtained from $\mathcal{B}$ by appending a single 0 and $k-i$ 1s and let $\mathcal{B}^+ = \mathcal{B}1000\ldots0$ be obtained from $\mathcal{B}$ by appending a single 1 and $k-i$ 0s. Then $\mathcal{G}$ starts not earlier than some time between $\mathcal{B}^-$ and $\mathcal{B}^+$, and ends not later than some time between $t_{mid} + \mathcal{B}^-$ and $t_{mid} + \mathcal{B}^+$. Refer to Figure 4.7. Hence, there is no integer h such that both h and $t_{mid} + h$ are contained in the time interval of $\mathcal{G}$. □

The claim directly implies that all k-max groups are maximal, because by Property 3 they start and end at some time h and $t_{mid} + h$, but adding any other trajectories will cause both p_i and p'_i to be in the group for some i.

Moreover, the ε-events created by entities of $p'_1, \ldots, p'_k$ are also the end ε-events of the $2^k - 1$ groups that have more entities than a k-max group. Let the maximal group contains all entities end at free ε-event γ at time $t_\gamma = t_{mid} + 2^{k-1} + \mathcal{T}$ $(0 < \mathcal{T} < 1)$ created by p'_i. By the symmetry of the construction and the ordering of the bitstrings, two groups of $n-1$ entities not containing either p'_i or p_i will end at time $t_\gamma - 2^{k-2}$ and $t_\gamma + 2^{k-2}$, respectively. Then, continuing the same process with the two groups recursively will results on other maximal groups with different entities. Since the start and end ε-events of these groups are always start later or end earlier than k-max groups, then these groups are maximal because their interval will not contain interval of other maximal groups. Clearly, the number of these maximal groups is fewer than k-max groups because their ε-events only occur between two consecutive ε_q events. In Figure 4.7, p'_1 defines γ, the end ε-event for a maximal group containing all entities. Then, maximal group that are not contain p_1 or p'_1 will end before or after γ, respectively.

To build the construction, all trajectories must have a constant times 2^k vertices for the ε-events of q and q' and a constant number of vertices in between those ε-events. Each trajectory in the construction has $\Theta(\sqrt{2}^n)$ vertices. We conclude that the number of maximal groups that contain time t_{mid} in this construction is at least $2^k = 2^{n/2-1} = \Omega(\sqrt{2}^n)$. $\square$

4.6 Conclusions and Future Work

In this chapter we introduced a variation on the grouping structure definition [126] and argued that it corresponds better to human intuition. The number of maximal groups that can arise in a set of n moving entities is $\Theta(\tau n^3)$ in the worst case. We have given an algorithm for trajectories moving in $\mathbb{R}^1$ that computes all maximal groups and runs in $O(\tau^2 n^4)$ time. In $\mathbb{R}^d$, our algorithm runs in $O(\tau^2 n^5 \log n)$ time. For the more general case where the input trajectories do not have time-aligned vertices, the algorithm for trajectories in $\mathbb{R}^1$ can be extended at the cost of an extra factor of $\alpha(n)$, while the same result still holds for trajectories in $\mathbb{R}^d$.

Furthermore, we presented an algorithm that has only linear dependence on τ, at the expense of exponential dependence on n. Since collections of trajectories are often very large in the number of time stamps and not necessarily in the number of trajectories, this algorithm or a practical variation on it may still be useful. This algorithm is not affected by whether or not the vertices of the trajectories are aligned in time.

In the next chapter, we implement our $O(\tau^2 n^5 \log n)$ time algorithm for entities moving in $\mathbb{R}^d$ and experimentally show the differences between the definition of convoys [81], swarms [121], groups [126] and our refined definition, both qualitatively and quantitatively.

For future work, it would be interesting to develop algorithms that take geodesic distance into account to define direct ε-connectedness instead of the straight-line distance, as was done for the previous definition of a group [188].

5

AN EXPERIMENTAL EVALUATION OF GROUPING DEFINITIONS

In the previous chapter, we refine the (original) definition of groups by Buchin et al. [126] and show that in dense environments the refined definition might give intuitively more plausible groups. Now, we compare several grouping definitions (including the original group [126] and our refined group) by conducting an extensive experimental study using various types of trajectory data sets. Our experimentation focuses on finding small groups—which can consist of just two entities—in pedestrian data, which is essential to analyze the throughput in public spaces such as train stations or airports, and in detecting suspect behavior in such spaces.

Beside our refined group and the original grouping definition by Buchin et al. [126], there are many different definitions have been suggested to model the collective movement of a “sufficiently large” set of entities that travel “together” for a “sufficiently long” period of time: moving micro-clusters [114], moving clusters [115, 116], mobile groups [117], flocks [111, 112, 113], herds [118], convoys [81], swarms [121], gatherings [124], traveling companions [123], and platoons [125].

Most of the grouping definitions listed above are different by one or more of three different characteristics: whether they use the measured sample points or the continuous movement as the input, how distance is used to decide the togetherness of entities in the same group, and whether the togetherness of entities is measured cumulatively or as one contiguous time interval. It is beyond the scope of this study to overview them all and explain their often subtle differences. We refer to the original papers, most of which introduce the new model for a collective movement, present one or more algorithms to compute it, and describe experiments where the new algorithms are run on some data sets, sometimes with a comparison to one earlier type.

We compare four of the definitions, namely *convoys* [81] (which in our setting are the same as traveling companions [123]), *swarms* [121], *original groups* [126], and our *refined groups*. Early definitions that use a shape of the cluster (flocks) in the definition are disregarded because they exhibit the lossy flock problem [81]. Since we study small groups, allowing group composition change is not so suitable, and hence we do not take definitions into account where the composition may change (like moving clusters and gatherings). Some other definitions are motivated mostly by vehicle data; we also do not consider these. We note that the four chosen definitions all use exactly three primary parameters: one for group size, one for group duration, and one for entities inter-distance. Therefore, comparing these definitions is more clean than including more complex definitions that need more parameters. There are no definitions that use fewer parameters. Note that convoys and swarms use one additional parameter for the density threshold, but we fix its value. We provide a full explanation of this parameter in the next section.

Our study is not just an analysis of the four definitions (i.e., convoys, swarms, original groups, and refined groups). We also investigate of how the input, space, and time can be treated and how this affects the results of the experimentation. Furthermore, the objective of our study is not to find the "best" definition since we typically do not have the ground truth. However, there are human annotations of groups for several real-life trajectories data sets. Therefore, we can compare the four definitions to the human annotations, which are by their nature subjective.

We consider the following research questions:

1. How well do the above definitions correspond with what humans consider a "group", and how do the characteristics mentioned (input, connectivity, and duration) influence this?
2. How does the number of groups, as reported by the various definitions, depend on the density of the entities?
3. How does the number of groups, as reported by the various definitions, depend on the sampling rate of the input trajectories?

We answer these questions by performing both a quantitative and qualitative analysis. For the quantitative analysis, we use real-life trajectory data sets to compare the reported groups by the four definitions with the human annotation groups. To evaluate the results, we use three evaluation metrics: *precision*, *recall*, and *F1-scores*. We also use a crowd simulation system to generate synthetic trajectory data from small groups (moving in a crowd) that stay in coherent and have socially-friendly formations [150]. We compute how well the various definitions correspond to this kind of group formations.

We analyze the four definitions qualitatively only for the experiments with real-life trajectory data sets. The qualitative evaluation is augmented with a novel visualization that shows the reported groups with color-coding in video footage. Furthermore, the movement traces of the pedestrians in the video are also shown. With this visualization, we offer easy comparisons between the reported groups by one definition and the human-annotated groups, or with groups from another definition.

Results and Organization. In the following section, we review the four grouping definitions that we consider and analyze how they differ in theory. Then, we describe the methods for our experimental comparison and introduce our new visualization method in Section 5.2. We present the results of our experimental evaluation in Section 5.3.

5.1 The Definitions

The four definitions rely on three parameters to define a group: the size parameter (the number of entities in a group), the temporal parameter (the time interval in which those entities form a group), and the spatial parameter (the distance between entities in the group). We formalize these parameters to define a group $\mathcal{G}$ from a set of moving entities $\mathcal{X}$ during time interval $\mathcal{I}$:

- $\mathcal{G}$ contains at least m entities.
- $\mathcal{I}$ has a duration at least δ.
- Every pair of entities $x, y \in \mathcal{G}$ is *connected* during $\mathcal{I}$.

Note that for the size parameter, the required minimum of entities to form a group is the same for all four definitions.

For the temporal parameter, the swarms [121] definition handles it differently from the others since it measures the duration of a group cumulatively. Let $\mathcal{T}$ be a set of timestamps where at each timestamp, every pair of entities $\in \mathcal{G}$ are connected. Then, swarm uses the size of $\mathcal{T}$—the number of timestamps—to define the duration of $\mathcal{G}$, rather than the duration of one contiguous time interval $\mathcal{I}$. Note that with this property, swarm allows entities $\in \mathcal{G}$ to leave $\mathcal{G}$ and join again later, as long as $\mathcal{G}$ is formed during at least k timestamps (which may be non-consecutive). For the other three definitions, the requirement for the temporal parameter is similar. Further-

TABLE 5.1: Differences between grouping definitions.

	Input	Connectivity	Duration
Original Groups (OG)	continuous	free	consecutive
Refined Groups (RG)	continuous	within group	consecutive
Convoys (CO)	discrete	free	consecutive
Swarms (SW)	discrete	free	cumulative

more, the original and refined group definitions consider trajectories in its continuous form and interpolate positions between timestamps using the linear interpolation. Therefore, the start and end time of the duration of a group will typically not be at any timestamp.

For the spatial parameter, we take a close look at each definition. The original group definition uses ε-connectivity between two entities as follows [126]: two entities x and y ($x, y \in \mathcal{X}$) are *directly ε-connected* if at any particular timestamp t, the Euclidean distance between x and y is at most ε ($\varepsilon > 0$). Furthermore, x and y are *ε-connected in $\mathcal{X}$* at time t if there is a sequence $x = x_0, ..., x_k = y$, with $x_0, ..., x_k \in \mathcal{X}$ and for all i, x_i and x_{i+1} are directly ε-connected at time t. This definition has the advantage that we need to consider the locations of all entities at time t only, to decide whether two of them are ε-connected.

One may claim that it is more natural if connectivity for x and y at time t can only be provided by entities who are in the same group, which is the approach taken by the refined group definition. More specifically, to decide if x and y are ε-connected in in a group $\mathcal{G}$, we ignore all entities not in $\mathcal{G}$ and require a sequence $x = x_0, ..., x_k = y$, with $x_0, ..., x_k \in \mathcal{G}$ where x_i and x_{i+1} are directly ε-connected at time t (see Chapter 4). Computing groups using this refined definition appears more complex since we cannot decide just from the locations at time t whether x and y are ε-connected. We need the location history and future as well.

The convoy [81, 190] and swarm [121] definitions use the concept of density connection [191], which is similar to the requirement of the spatial parameter for the original group, but with a slight difference. Let the ε-neighborhood of an entity $x \in \mathcal{X}$ is defined by $\mathcal{N}_\varepsilon(x)$, the number of other entities in $\mathcal{X}$ that have the Euclidean distance at most ε ($\varepsilon > 0$) from x (at any given timestamp t). Now, given a density threshold μ ($\mu > 0$), an entity $y \in \mathcal{X}$ is *directly density-reachable* from x if $y \in \mathcal{N}_\varepsilon(x)$ and $|\mathcal{N}_\varepsilon(x)| \geq \mu$. Furthermore, y is *density-reachable* from x if a sequence of entities $\in \mathcal{X}$ exists where each consecutive pair of entities in the sequence from x to y is directly density-reachable. Clearly, if $\mu = 1$ then the notion of (directly)

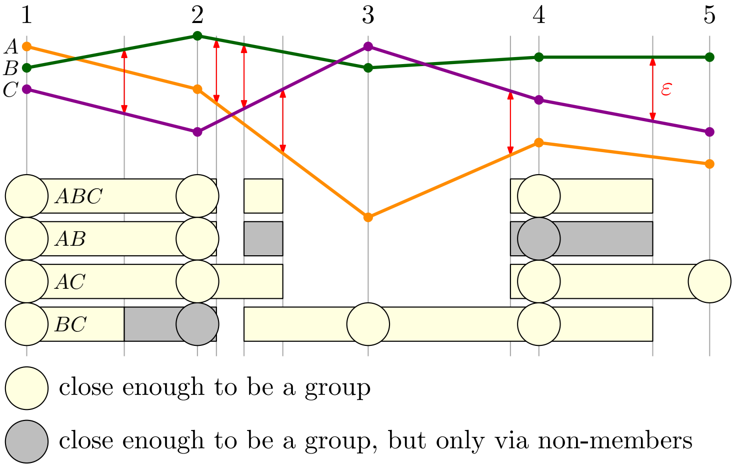

1
2
3
4
5
A
B
C
ε
ABC
AB
AC
BC
close enough to be a group
close enough to be a group, but only via non-members

Figure 5.1: Maximal groups according to: $(m = 2, \delta = 1)$
CO: $ABC[1,2], ABC[4], AC[4,5], BC[1,4]$
SW: $ABC(1,2,4), AC(1,2,4,5), BC(1,2,3,4)$
OG/RG: $ABC[1,2.1], AC[1,2.5], AC[3.8,5], BC[2.3,4.6]$

density-reachable is exactly the same as the (directly) ε-connected in the original group definition. Henceforth, we only use $\mu = 1$ since $\mu > 1$ prevents the convoy and swarm definitions to identify groups that contain only two entities.

We summarize the differences between the definitions in Table 5.1. We note that no two of the four definitions we consider are the same on all three aspects.

Maximal Groups. In the original and refined group definitions (see Chapter 4), a group $\mathcal{G}$ is a *maximal group* during time interval $\mathcal{I}$ if there is no time interval $\mathcal{I}' \supset \mathcal{I}$ for which $\mathcal{G}$ is also a group and there is no $\mathcal{G}' \supset \mathcal{G}$ that is also a group during $\mathcal{I}$. Moreover, swarms also has exactly the same concept as the maximal group, namely *closed swarm*. On the other hand, the definition of convoys only considers the ones that are maximal. Henceforth, we also use the term maximal group to describe the (maximal) convoy and the closed swarm. Figure 5.1 illustrates the concept for a small example. Note that the same set of entities can appear multiple times (at different moments in time) as a maximal group under all definitions except swarms.

Differences. Now, the differences shown in Table 5.1 affect how each definition specifies maximal groups from a set of trajectories. We demonstrate this using examples. First, we present an example in Figure 5.2, where a maximal group containing exactly the same entities may have different time

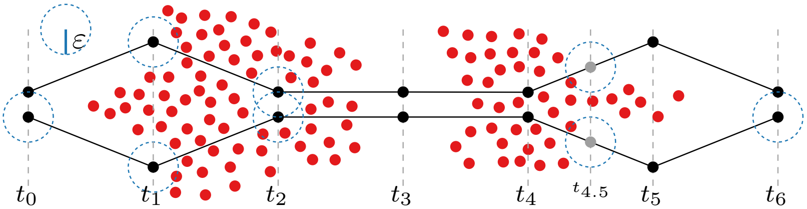


FIGURE 5.2: According to different definitions, the black entities are a group at different times.

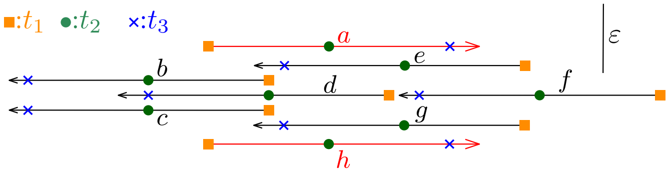


FIGURE 5.3: Entities a and h are not a refined group during $[t_1, t_3]$, but they are an original group, convoy, and swarm during $[t_1, t_3]$ or t_1, t_2, t_3.

durations, depending on which definition we use. Let two black entities x and y be the only entities that move; all red entities are stationary. Furthermore, trajectories of x and y consist of the shown positions at $t_0, t_1, ..., t_6$, and we set $\delta = 2$. Since x and y are not ε-connected (or density reachable) at t_5, the pair $\{x, y\}$ is a convoy during the time interval $[t_0, t_4]$, while the swarm $\{x, y\}$ is formed during the timestamps of $\{t_0, t_1, t_2, t_3, t_4, t_6\}$. With the original group definition, x and y are a group starting at t_1 and ending at $t_{4.5}$. Finally, the refined group of $\{x, y\}$ can only start when the distance between x and y is $\leq \varepsilon$ because they cannot be ε-connected through the red entities. Therefore, they can only start at t_2 and must end at t_4.

Next, we show that the type of connectivity between entities in a group such as in the refined group definition can result in a completely different grouping. In particular, we use the same example from the previous chapter. In Figure 5.3, two entities a and h are moving in the same direction, opposite to the other entities. At any time during the time interval $\mathcal{I} = [t_1, t_3]$, a and h are ε-connected through other entities. As a consequence, the convoy and the swarm definitions consider $\{a, h\}$ to be a group at timestamps t_1, t_2, t_3, or during interval $\mathcal{I}$ for the original group definition. In the refined group definition, $\{a, h\}$ is not a group during $\mathcal{I}$ because their connectivity is only through (changing) entities not in the group itself. There are several refined groups that include a and h that have a considerably shorter duration.

5.2 Methods

To answer the research question described previously, we conduct extensive experiments by computing all maximal groups from various trajectory data sets according to the different group definitions. We evaluate the results both quantitatively and qualitatively.

5.2.1 Experimental Setup

Data sets. To conduct our experiments, we need trajectory data. We use various data sets, which we divide into three *categories* based on the source of the trajectories:

- Real-life trajectories extracted from videos surveillance in a public area: the *NYC Grand Station* [192, 193], *ETH Walking Pedestrian* [194, 195], *Crowds by Examples* [196, 197, 198, 1] and *Vittorio Emanuele II Gallery* data sets [199, 197, 1].
- Real-life trajectories of pedestrians walking in a laboratory environment: the *Pedestrian Dynamics* data set [200, 201].
- Artificial trajectories generated by a computer simulation: the *Netlogo Flocking* data set [44, 46] and the *Utrecht University Crowd Simulation* data set [150].

We describe each data set in more detail along with the results of experiments using them in Section 5.3. The real-life data sets are captured from video surveillance; hence their raw coordinates are frame (pixel) coordinates from the videos. These coordinates are first converted to world coordinates using a homography matrix to be able to make fair distance comparisons. Most real-life trajectory data sets also come with a list of *human-annotated* groups; only the *NYC Grand Station* and the *Pedestrian Dynamics* data set do not.

Implementations. In order to compute all maximal groups according to the different notions of groups, we implement all algorithms ourselves:

- the Smart-and-Closed algorithm [123] to compute *convoys*,
- the ObjectGrowth algorithm [121] to compute *swarms*,
- the algorithm to compute *original groups* [126], and
- the algorithm in Section 4.3 to compute *refined groups*.

Note that since the swarm algorithm has an exponential running time we were unable to compute all swarms for some of the parameter values in our experiments.

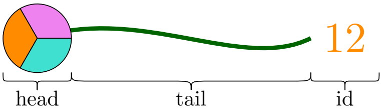


FIGURE 5.4: A moving entity is shown in a schematic manner.

5.2.2 Evaluation Setup

Quantitative Evaluation. We analyze and evaluate the results from all experiments quantitatively. We compute and count all maximal groups in our data sets according to the four definitions while varying the parameters of the definitions: the distance ε, the minimum time duration δ, and the minimum group size m.

- For several data sets, we provide the precision, recall, and F1-score as measurements to show the relevance between the groups found by each definition with the human-annotated or computer-generated groups. Note that we avoid the terms "correctness" and "ground truth": we can only test to what extent the groups found agree with the human-annotated or computer-generated groups. In particular, human-annotated data is likely to be influenced by personal interpretation and therefore not a ground truth.
- We vary the *density* of the environment by considering different numbers of entities moving in the same bounded space. We compute the number of groups for each definition and study how it changes with the number of entities.
- We vary the sampling rate, or level of detail, of the trajectories by ignoring a fraction of the vertices in each trajectory. We count how many groups are identified by the different definitions, and analyze the consistency of these numbers.

Qualitative Evaluation. We also qualitatively evaluate the results of our experiments by visualizing the trajectories of pedestrians integrated in the videos from the data sets.

Conceptually, we represent each moving entity by a schematic figure that is overlaid on the video material; refer to Figure 5.4. Each entity consists of three parts. The *head* is a disk which shows the *current location* of the entity. The *tail* is a piece of curve which shows the *previous locations* of the entity during a set duration. The *id* is a unique identifier of the entity.

Grouping information is encoded by the *color* of the head and tail. We use the color of the *tail* to show a base grouping: depending on the data set this can be either the "ground truth", a human-annotated grouping, or a grouping computed by one of the methods. Entities belonging to the same

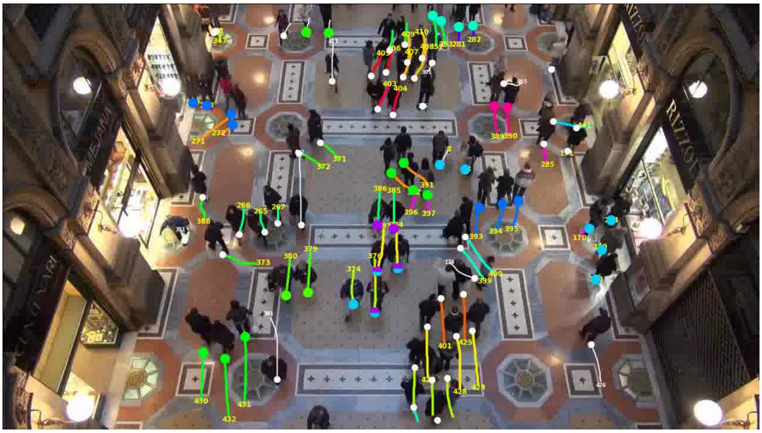


Figure 5.5: A snapshot from a video.

group have the same color, and every entity can only belong to at most one group, which cannot change over time. Entities that do not belong to any group in the "ground truth" have a white tail. The color of the *head* indicates the grouping as computed by the method currently under study. The computed groups can in principle overlap, and they do change over time. As a result, the head of an entity can have multiple colors, and the color of a head may change as time progresses.

The combination of colors of the tails and heads gives insight into the matching between annotated groups and groups based on a grouping definition. Note that colors are chosen at random; even when a method produces a group that exactly matches with one of the annotated groups, the color of the head may be different than the color of the tail.

We applied this scheme to all our data sets and generated videos for various parameter settings; see Figure 5.5 for an example. Our implementation of the visualization is based on the work by Maurice Marx [202]. In the remainder of this chapter, we supply some snapshots of interesting configurations. The complete collection of videos from this chapter can be found on our website [203].

5.3 Experimental Evaluation

In this section we evaluate the results of our experiments. We focus our evaluation on the differences of the four definitions, and thus on the maximal groups that are reported, rather than the differences between the algorithms and their implementation. All implementations are non-optimized prototypes and therefore, comparing statistics like running times is meaningless.

TABLE 5.2: Information on the trajectories in the data sets and parameters used in the experiments. Here, g denotes the number of annotated groups, and G their size range. The video length is specified in minutes and seconds; the values for ε and δ are in meters and frames, respectively.

	ETH	HTL	VEIIG	CBE
video length	08:39	12:54	05:00	03:36
FPS	25	25	8	25
#entities	360	389	630	434
avg τ	143.37	159.2	189.82	400.12
g	58	39	207	115
G	2-3	2-6	2-7	2-4
ε	1.24	0.94	0.963	{1.22,1.52}
δ	{72,89,105}	{20,58,96}	{17,38,58}	{36,57,78}

5.3.1 Comparisons with Annotated Groups and Social Formations

We aim to establish how well the definitions capture the human intuition of grouping. In our first experiment, we compute the groups, as reported by the various definitions, and compare them to human annotation. We then report the *precision* (the percentage of the groups reported by the algorithm that also occur in the annotated data), the *recall* (the percentage of the human-annotated groups also found by the definition), and the corresponding *F1*-score. The F1-score is the harmonic mean of the recall (R) and the precision (P), and can be computed using the following formula: $F1 = 2\frac{P\times R}{P+R}$.In our second experiment, we use a crowd simulation framework to generate trajectories including a set of entities traveling in a "social formation". Intuitively, this is a group. We test if the various definitions identify these entities as a group.

Annotated Groups

We use four data sets consisting of real-life trajectories from video surveillance: ETH Walking Pedestrian (ETH-EWAP and HTL-EWAP) [195], Vittorio Emanuele II Gallery (VEIIG) [1], and Crowds by Example (CBE) [198, 1]. See Table 5.2 for details of each data set. Besides trajectories of pedestrians, these data sets are supplemented with homography matrices and lists of groups that are annotated manually by the authors. The annotations specify only *which* entities appear in a group, not *when*, or *how long* the entities form a group. Moreover, unlike in the four definitions, an entity occurs in at most one group in the human annotation.

TABLE 5.3: Comparative results on the ETH-EWAP data set.

		Precision	Recall	F1 Score
$\delta = 72$	OG	0.410	0.741	0.528
	RG	0.467	0.741	0.573
	CO	0.453	0.741	0.562
	SW	0.380	0.793	0.514
$\delta = 89$	OG	0.411	0.638	0.500
	RG	0.463	0.638	0.537
	CO	0.446	0.638	0.525
	SW	0.372	0.724	0.491
$\delta = 105$	OG	0.418	0.569	0.482
	RG	0.471	0.569	0.515
	CO	0.452	0.569	0.504
	SW	0.404	0.655	0.500

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

For each definition, we count how many maximal groups match exactly with the annotated groups and evaluate the correctness using the precision, recall, and F1-score. For each data set we set the minimum required number of entities m to 2. The values for the inter-entity distance ε are chosen based on a study by Solera et al. [197], who analyze the average distance between people in the same group in a human crowd. Finally, we determine three different values for required minimum time δ that a group is together, based on the group annotations. In particular, we assume that a set of people cannot form a group when not all members are present in the video. Hence, we compute the time interval during which all members of an annotated group are present, and define the duration of the group to be the length of this interval. The minimum such duration over all groups gives us one choice of δ. The other two are chosen based on the average such duration $\bar{\delta}$ and the standard deviation σ. In particular, we pick $\bar{\delta} - \sigma$ and $\bar{\delta} - \frac{1}{2}\sigma$.

The ETH Walking Pedestrian. The data set contains two bird-eye view videos from two locations: in front of a university building at ETH Zurich (ETH) and a sidewalk near a tram stop (HTL). We consider this data sets to have low to medium density pedestrian. The results of the experiments using the ETH-EWAP and HTL-EWAP data sets are presented in Table 5.3 and Table 5.4, respectively.

As expected, if the density of the crowd is relatively low, all definitions have similar results, which we can see from their F1-score from both data sets. Furthermore, as we increase the value of δ, the recall values are decreased because now many annotated groups with a short duration cannot be found by all definitions.

TABLE 5.4: Comparative results on the HTL-EWAP data set.

		Precision	Recall	F1 Score
$\delta = 20$	OG	0.400	0.974	0.567
	RG	0.418	0.974	0.585
	CO	0.409	0.974	0.576
	SW	0.463	0.974	0.628
$\delta = 58$	OG	0.660	0.897	0.760
	RG	0.673	0.897	0.769
	CO	0.660	0.897	0.760
	SW	0.706	0.923	0.800
$\delta = 96$	OG	0.690	0.744	0.716
	RG	0.707	0.744	0.725
	CO	0.707	0.744	0.725
	SW	0.705	0.795	0.747

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

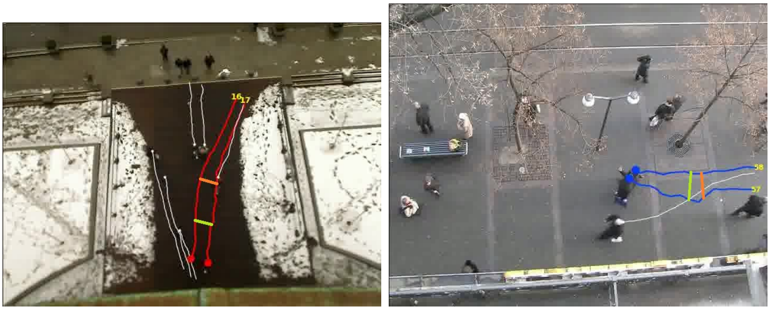


FIGURE 5.6: Examples from ETH (left) and HTL (right) data set, orange and light green lines shows the end and start of a group, respectively. Swarm can detect both groups, while other definitions cannot because the length of contiguous time interval is below required δ.

We observe that swarm can match more annotated groups than other definitions but with a low precision because of swarm output more maximal groups. The other definitions output less number of maximal groups (and different between each definition), but the number of annotated groups that they are able to match is the same. Here, the strict connectivity requirement makes the refined definition gets better precision scores.

Now, we look into details on the maximal groups found by each definition. It turns out that all maximal groups found by the three other definitions are the same, while the swarm also finds them and more. From the ETH data set, we give an example of a maximal group that is found by swarms but not by the others (see Figure 5.6 (left)). This example is taken when we set δ to 90 or 120, while for $\delta = 72$, all definitions consider the two entities in red as a maximal group.

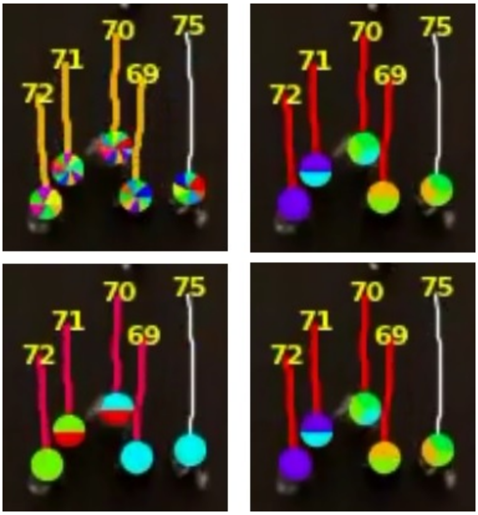


FIGURE 5.7: (Top-left) Swarm detects an annotated group of {69,70,71,72} but also many other groups compare to the other definitions. (Top-right) The result from convoy. (Bottom-right) The result from original group, which is the same with convoy. Notice that {70,75} is also a group. (Bottom-left) The result from refined group. Since their connectivity must be within group only, {70,75} is not a group.

After 83 timestamps, they reach a position where their distance to each other is $> \varepsilon$ (see the orange line in Figure 5.6 (left)) and no other entities exist to keep their connectivity. Therefore, all grouping definitions that require contiguous time interval as a duration of a group, fail to detect the group for $\delta > 83$. Later, they resume their ε-closeness (starting at the light green line) for a short time which enables swarm to detect this as a maximal group. The same observation also applies to the HTL data set, see Figure 5.6 (right).

We present another example in Figure 5.7 where swarms (the top-left figure) find one particular maximal group (i.e., {69,70,71,72}) that other definitions can not. We show the reason in Figure 5.8. The left and right figures in Figure 5.8 show the events before and after the moment in Figure 5.7, respectively. In Figure 5.8 (left), the white pedestrian walks through the group and separate them far enough that the original group and convoy find the two sub-groups are disconnected. Although the two subgroups join again afterward, they do not stay together long enough since the leftmost pedestrian (72) moves faster and leaves the others, see Figure 5.8 (right). The time interval of these two events is not long enough for the required minimum duration of $\delta = 72$.

From Figure 5.7 (top-right) and (bottom-right), we can see that convoys and original groups have the same results, but the refined group (Figure 5.7 (bottom-left)) is different. Since the refined group only realizes connectivity

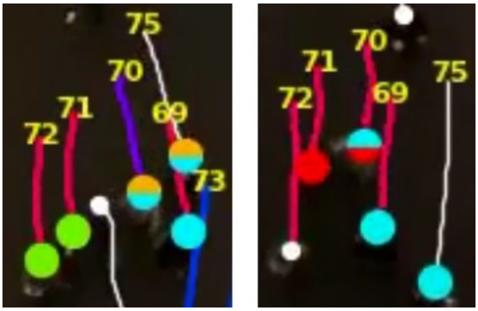


FIGURE 5.8: (Left) The white trajectory separates colored trajectories into two sub-groups. (Right) The leftmost pedestrian (72) moves faster and leaves others.

TABLE 5.5: Comparative results on the Vittorio Emanuele II data set with human annotation.

		Precision	Recall	F1 Score
$\delta = 17$	OG	0.199	0.952	0.329
	RG	0.223	0.947	0.361
	CO	0.202	0.952	0.333
	SW	0.125	0.957	0.221
$\delta = 38$	OG	0.357	0.884	0.509
	RG	0.414	0.884	0.564
	CO	0.362	0.884	0.514
	SW	0.238	0.932	0.379
$\delta = 58$	OG	0.444	0.778	0.565
	RG	0.503	0.778	0.611
	CO	0.451	0.778	0.571
	SW	0.315	0.870	0.463

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

within members of groups, it does not consider the group of {70,75}. However, convoys and original groups use another entity (69) as an intermediate to keep the two pedestrians (70 and 75) to be ε-connected.

Vittorio Emanuele II Gallery. The data set is taken from the video surveillance in a hallway inside the Vittorio Emanuele II Gallery in Milan, Italy. The flow of entities in the video is mostly bidirectional. The results of our experiments are in Table 5.5.

First, we note that for all definitions, the precision values are relatively small. This can be explained by the fact that all definitions (except for swarm) consider a set of entities that is together during two disjoint but sufficiently long time intervals as two different (maximal) groups. Also, a group of 3 (or more) entities is often also found as one or two subgroups

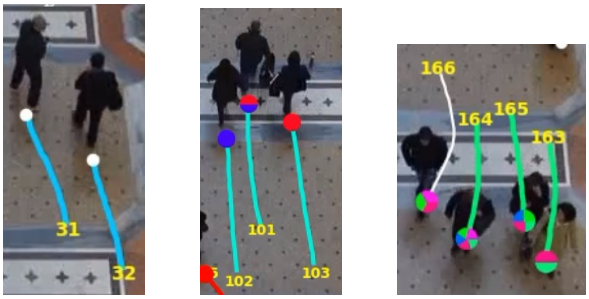


FIGURE 5.9: (Left) The pedestrians in a group are not within distance ε long enough. (Middle) The pedestrian in the middle is close to the left pedestrian and not close to the right one during this snapshot, but the reverse was the case earlier in the video. The annotation has all three in a group. (Right) The pedestrian on the left is always close to a group but the annotation does not include it.

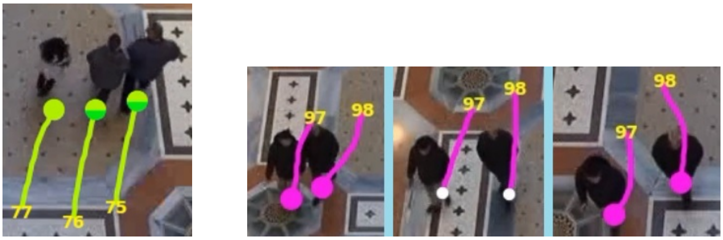


FIGURE 5.10: (Left) In this group of 3, the two pedestrians on the right appear earlier and disappear later in the videos, making a maximal group of 2 that is a subgroup of the 3. (Right) Frames in sequence from left to right show a group that separated for a while, resulting in two different maximal groups by the grouping definitions, except for swarm.

of 2 entities with slightly longer duration. Therefore, we focus on relative precision values. This also holds for the other data sets in our experiments.

We observe that in this data set, the refined group corresponds better to human annotation than the others based on their F1-score. This is mostly since the refined group definition has the highest precision out of the definitions considered. The swarm definition has the best recall value, while the maximal groups by the original group and convoy definitions find the same number of annotated groups; refined group misses one in total. The higher recall of swarm is related to the lower precision: swarm outputs many more groups, some of which correspond to human annotation. These may be groups with interrupted duration.

Our qualitative review shows several reasons why the grouping definitions cannot match all annotated groups, see Figure 5.9. One main reason is that the members of an annotated group are not within ε distance for a duration δ. This results in (i) annotated groups not recognized at all, or (ii) grouping definitions only found subgroups of annotated groups. There are also situations where entities are always within distance ε, but they were not annotated as a group. It possible to increase the recall by increasing ε,

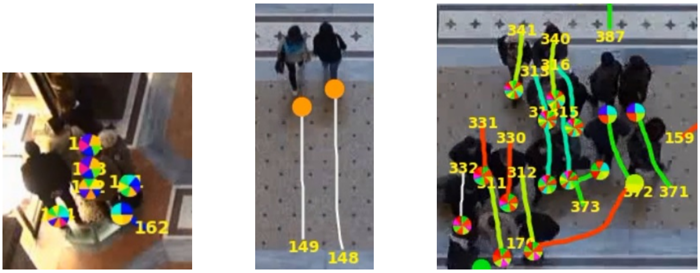


FIGURE 5.11: (Left) Two groups of pedestrians standing close together, making different maximal groups when they walk again. (Middle) A group found by all four definitions that was not annotated. (Right) In a dense environment, many more groups are produced by all grouping definitions.

TABLE 5.6: Comparative results on the CBE data set; $\varepsilon = 1.22$.

		Precision	Recall	F1 Score
$\delta = 36$	OG	0.172	0.565	0.264
	RG	0.181	0.565	0.274
	CO	0.167	0.600	0.261
	SW	0.176	0.652	0.277
$\delta = 57$	OG	0.243	0.461	0.318
	RG	0.254	0.452	0.325
	CO	0.245	0.470	0.332
	SW	0.269	0.574	0.366
$\delta = 78$	OG	0.307	0.339	0.322
	RG	0.315	0.339	0.327
	CO	0.291	0.357	0.321
	SW	0.326	0.522	0.401

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

but the precision is likely to go down. Figures 5.10 and 5.11 show several scenarios that help to explain the low precision of all four definitions.

Crowds by Example. The Crowds by Example (CBE) data set records pedestrian outside a university building. The flow of pedestrians is different than in the GVEII data sets: pedestrians move in various directions with varying speed. In this data set, vertices are sampled once every 6 frames. For experiments using this data set, we set ε based on Proxemics Theory [204], rather than the theory by Solera et al. [197] which seems to suggest an unrealistically small value for ε (namely 0.41m). Instead, the maximum far phase for a personal distance between pair of individuals from Proxemics Theory gives $\varepsilon = 1.22$m. We present the results in Table 5.6 Although swarms have the same discrete handling of the input, it performs better on recall because it appears there are groups with interrupted duration that are not found by the other definitions.

TABLE 5.7: Comparative results on the CBE data set; $\varepsilon = 1.52$.

		Precision	Recall	F1 Score
$\delta = 36$	OG	0.131	0.817	0.226
	RG	0.138	0.817	0.236
	CO	0.130	0.835	0.225
	SW	0.097	0.861	0.174
$\delta = 57$	OG	0.180	0.722	0.288
	RG	0.203	0.722	0.317
	CO	0.182	0.713	0.290
	SW	0.135	0.800	0.231
$\delta = 78$	OG	0.224	0.609	0.328
	RG	0.260	0.591	0.361
	CO	0.228	0.617	0.333
	SW	0.164	0.757	0.270

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

In dynamic crowds, we expect that entities from the same group will not be close to each other all the time, which is one reason why swarms can have a high recall score. Hence, we do the same experiment with different ε, to see how the other definitions that find maximal groups by a longest consecutive timestamp will perform. We set $\varepsilon = 1.52$ and present the results in Table 5.7. As expected, we see an increase in recall and decrease in precision for all definitions. However, the F1 results from the swarm get worse and other definitions now perform better for all δ. Our qualitative evaluation of the Crowds by Example data shows similar situations as for the VEIIG data set.

Comparison to Generated Social Formations

We now compare the grouping definitions to other notions of grouping. In particular, we consider "socially-friendly formations" in crowd simulation [150].

We use a crowd simulation framework developed at Utrecht University [205], aimed at generating realistic crowd behavior. The framework allows agents (entities) to traverse a virtual environment, using global "indicative" routes on an underlying navigation mesh [206]. The framework supports generating routes for a set of entities that attempt to stay in a socially-friendly formation throughout the motion (and to re-establish such a formation when it is lost) [150]. We use the framework to generate trajectories in which there is exactly one such a socially-friendly formation $\mathcal{G}$. We then test how well the four definitions capture $\mathcal{G}$.

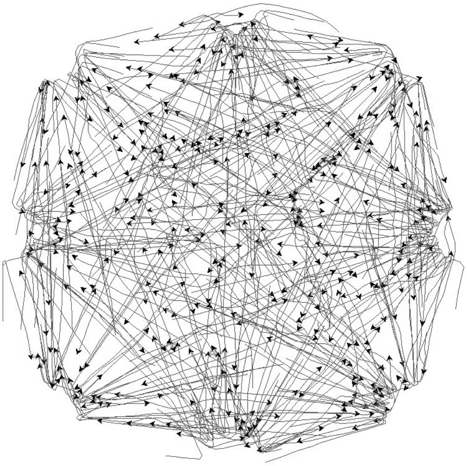


FIGURE 5.12: A set of trajectories generated using the UU Crowd Simulation.

The input trajectories are generated as follows: we construct a virtual environment in which we place a set $\mathcal{P}$ of nine "points of interest". We choose the eight corners of an octagon and its center point as $\mathcal{P}$. Each entity has a global route visiting these points of interest. These global routes are picked randomly, i.e., when an entity gets close to its point of interest we randomly select a new target point of interest. We fix a set of four entities $\mathcal{G}$ that behave like a social group, make sure that they start at the same point in $\mathcal{P}$, and follow the same global route. All remaining $n' = n - 4$ entities do not have any social group behavior and pick their global route individually. Figure 5.12 shows an example of trajectories in this data set. For each choice of $n' \in \{100, 200, 300, 400, 500\}$, we run the simulation for 1000 time steps, thus producing trajectories with 1000 vertices each. Furthermore, for each n', we repeat the simulation ten times. Note that even though the entities in $\mathcal{G}$ use the same global route, the individual trajectories will differ as the entities in $\mathcal{G}$ have to avoid colliding with other entities.

Finding Social Formations. We compute the maximal groups on the resulting data sets, selecting $\varepsilon = 1.6$ and $m = 4$, and we compare when $\mathcal{G}$ is a maximal group according to all definitions. Figure 5.13 shows the percentage of time during which $\mathcal{G}$ is a maximal group as a function of the number of dummy entities n', and thus of the density in the environment. First note that, even though the entities in $\mathcal{G}$ use the same global route, this is not a guarantee that the entities remain close together (and thus form a group) throughout the entire time interval considered. Indeed, we see that as we increase the number of dummy entities, the entities in $\mathcal{G}$ may be forced to spread out in order to avoid collisions. We see that for all densities considered, $\mathcal{G}$ occurs as a maximal group longest in the swarm definition and shortest in the refined group. An explanation for this is that in all definitions except for the refined group, the entities in $\mathcal{G}$ can stay in a group longer by using the non-group entities to remain connected. Note, however, that this does not guarantee that $\mathcal{G}$ is a maximal group for a longer period of time, since these dummy entities could also create a larger maximal group $\mathcal{H} \supset \mathcal{G}$ that prevents $\mathcal{G}$ from being maximal. As expected, swarm has a larger percentage where $\mathcal{G}$ is a group than the other definitions, especially when the environment becomes more dense.

In Figure 5.14, we see the number of time intervals during which $\mathcal{G}$ is a maximal group. For up to $n' = 300$ we see a steady increase in the number of time intervals—and thus also the number of interruptions—during which $\mathcal{G}$ is a maximal group. So, it appears that as the crowd becomes more and more dense, the entities in $\mathcal{G}$ have to separate to avoid colliding with other entities more frequently. When we increase the number of dummy entities even further, the number of interruptions seems to remain constant, or even decline a bit. A possible explanation is that there are now so many entities in the environment that it also takes more time for the entities in $\mathcal{G}$ to regroup.

The convoy has the fewest disjoint time intervals, although it has a similar percentage of $\mathcal{G}$ being maximal group with the refined group definition. We believe the reason for these results is because the convoy only considers the sampled point of trajectories. This characteristic has two consequences. First, the convoy might not realize that a group is being separated or joined between timestamps, while the refined group does. Therefore, the contiguous time interval in the convoy is longer than the original group and it makes the number of time intervals also less than the original group's. Second, in the dense environment, this easily leads to the situation where at a particular time interval, a maximal group $\mathcal{H} \supset \mathcal{G}$ exists. Thus, while the contiguous time interval of the convoy is longer, its total duration is similar to the original group, because $\mathcal{G}$ in the original group can avoid $\mathcal{H}$ by starting and ending as a group just before and after $\mathcal{H}$, respectively.

Research question (1): correspondence to human annotation and other models. Over all experiments, the refined groups have a slightly higher F1-score in the correspondence to human annotation than the original groups and convoys definitions, but they are usually close. The higher F1-score is caused by a better precision. The swarms definition sometimes corresponds better and sometimes worse to human annotation. It appears to depend on the precise parameter settings. We also observe that human annotation

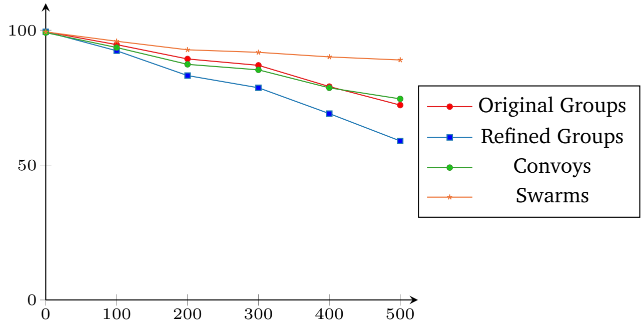


FIGURE 5.13: The percentage of time during which G is a maximal group as a function of the number of entities n' (averaged over ten simulations).

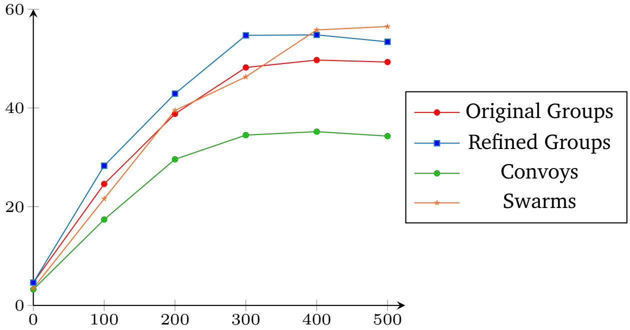


FIGURE 5.14: The number of disjoint time intervals during which $\mathcal{G}$ is a maximal group (averaged over all ten simulations).

is likely somewhat subjective. When analyzing the retrieval of a socially-friendly formation, we observe that all definitions notice interruptions of the groups with increasing density. The swarms definition suffers least from this and the refined groups definition most.

5.3.2 Dependence on Density

For the second experiment, we investigate how the maximal groups produced by each definition is affected by the density of the environment. Therefore, for each data set, we consider grouping in increasing density situations. Arguably, dense situations are especially difficult for identifying groups.

The Netlogo Flocking Model

We generated several data sets using an adapted version of the NetLogo Flocking model [44, 46]. This data is convenient because we can easily

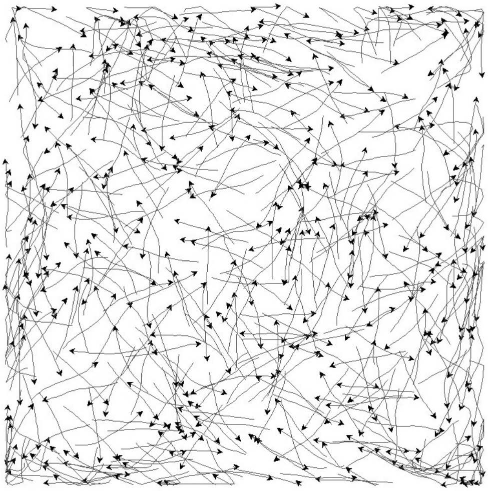


FIGURE 5.15: Trajectories from the Netlogo Flocking data set.

produce data sets of varying densities. In the adapted model the entities start to turn when they approach the border (instead of wrapping around), and there is a small random component in the new direction of the entities. This same model was used by Buchin et al. [126] to test the definition of original groups.

In all our experiments, the size of the environment in which the entities move is fixed and set to 256×256 units. See Figure 5.15 for a general impression of the moving entities in these data sets. We consider different densities by varying the number of entities n to be 200, 300 or 400, and generate data sets with 500 time stamps each. For each generated data set, we compute all maximal groups for all four definitions, with a fixed $\delta = 10$ and $m = 10$, but using three values of ε, namely 4, 5, and 6. We chose to vary ε because this distance value is related to density, the characteristic under investigation. Each experiment is performed 10 times and the average and standard deviation are computed. The results of these experiments are shown in Table 5.8. There are no results for swarm when $\varepsilon = 6$ and $n = 400$ due to the computation time needed: the swarm algorithm has exponential running time.

Generally, for all definitions the number of maximal groups increases as the density increases or when ε increases, which is not a surprise. We see that in most settings the refined group produces fewer maximal groups than the other definitions, and swarm produces more. All definitions show roughly a 20-fold increase from $n = 200$ to $n = 300$ when $\varepsilon = 5$. From $n = 300$ to $n = 400$, the swarm definition has a larger than 20-fold increase

TABLE 5.8: The average number of maximal groups for $m = 10$ and $\delta = 10$, and the standard deviation in the Netlogo data set for 10 generated sets.

		average (10 sets)				std. dev.			
$\varepsilon = 3$	n	OG	RG	CO	SW	OG	RG	CO	SW
	200	0	0	0	0	0	0	0	0
	300	3.0	1.7	3.2	9.1	3.99	2.86	3.63	16.27
	400	23.1	12.3	31.1	112.2	23.48	17.95	21.65	252.86
$\varepsilon = 4$	n	OG	RG	CO	SW	OG	RG	CO	SW
	200	2.9	2.2	3.0	10.8	3.58	2.00	4.60	11.00
	300	411.6	38.4	61.7	229.0	12.17	6.50	12.79	77.89
	400	396.1	259.0	410.8	5299.6	64.15	47.61	53.61	2363.13
$\varepsilon = 5$	n	OG	RG	CO	SW	OG	RG	CO	SW
	200	33.9	25.9	36.6	229.3	9.63	7.98	9.83	45.00
	300	396.1	304.5	418.6	5017.3	108.00	64.43	106.38	2466.90
	400	1905.7	1357.0	1830.4	-	250.68	204.28	226.49	-

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

while the other three definitions have a less than 10-fold increase. For $\varepsilon = 4$, the values are too small for such observations. For $\varepsilon = 6$, we notice that the increase for swarm from $n = 200$ to $n = 300$ is much larger than for the other three definitions. Hence, it seems that swarm has a larger increase in the number of maximal groups than the other three definitions when the density increases. We notice a similar effect when ε increases rather than the density.

Pedestrians in a Synthetic Environment

The purpose of this experiment is to measure the conformity of a grouping definition in a dense environment. We use the data set consists of trajectories extracted from video recordings of people walking in a synthetic environment [200, 201]. These trajectories are recorded by the Institute for Advanced Simulation of Jülich Supercomputing Centre to study the dynamics of pedestrians.

The particular data set we use consist of two sets of people walking in opposite directions through a corridor that is 8 meters long and 3.6 meters wide [207]. See Figure 5.16.[1] The density inside the corridor is controlled by the width w, in centimeters, of the two entrances to the corridor: a larger width w means that more people can enter the corridor simultaneously. The

---

[1] The frame is taken from http://ped.fz-juelich.de/experiments/2009.05.12_Duesseldorf_Messe_Hermes/export/bot-360-250-250v2.mp4

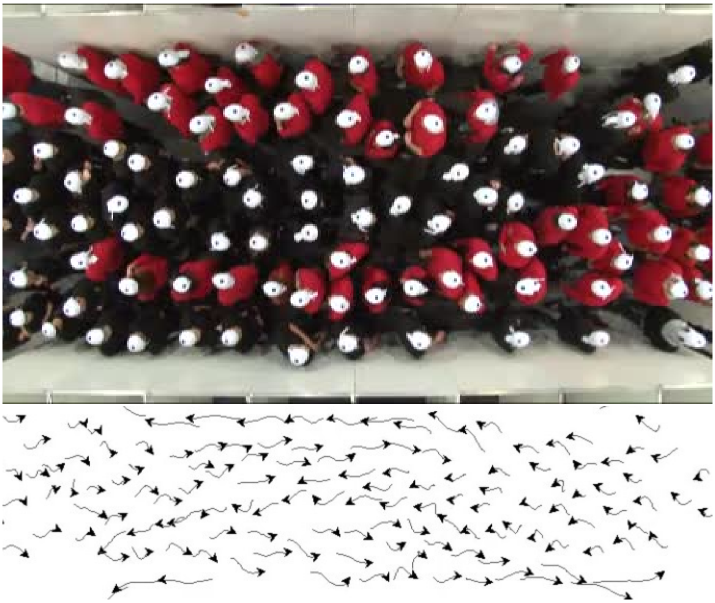


FIGURE 5.16: (Top) Frame from video recording of the corridor. (Bottom) Extracted trajectories from the video.

considered widths w are taken from $\{120, 160, 200, 250\}$. Each experiment consists of 300 trajectories, each of approximately 300 vertices as well.

In our experiments we fix the inter-entity distance ε to $80\,\mathrm{cm}$, and choose the minimum group size m from $\{3, 6, 9\}$. For the minimum required duration δ we consider values in the range $[60, 180]$. This corresponds to a minimum group duration roughly between four and twelve seconds. For comparison, the average time $\bar{t}$ it takes a person to cross the corridor ranges from roughly twelve to twenty-three seconds. Note that since the density of the environment is extremely high, the swarm algorithm—whose running time is exponential—unable to output any maximal group after a reasonable period of time (around 6 hours). Therefore, we exclude the swarm definition in these experiments.

The Number of Maximal Groups. The numbers of maximal groups for the considered parameter values are in Table 5.9. We first consider the number of maximal groups as a function of w, and thus of the density of the environment. As Figure 5.17 highlights for the case $m = 6$ and $\delta = 150$, we see that up to $w = 200$, the number of reported maximal groups increase as a function of w. This applies for the three definitions of a group, although the number of maximal groups according to the original group and the convoy increases much faster than for the refined group. For even bigger

TABLE 5.9: The number of maximal groups in the pedestrian data set.

	$m = 3$				$m = 6$				$m = 9$			
$w = 120, \bar{t} = 251.71$	δ	OG	RG	CO	δ	OG	RG	CO	δ	OG	RG	CO
	60	901	341	886	60	691	177	682	60	582	116	579
	90	600	190	595	90	431	68	431	90	348	26	351
	120	394	117	394	120	261	31	264	120	200	7	203
	150	240	55	241	150	132	13	134	150	88	5	91
	180	144	35	141	180	59	7	55	180	40	3	37
$w = 160, \bar{t} = 318.42$	δ	OG	RG	CO	δ	OG	RG	CO	δ	OG	RG	CO
	60	4718	1126	4313	60	4416	867	4025	60	4200	747	3813
	90	3991	580	3669	90	3722	382	3414	90	3518	286	3215
	120	3371	387	3111	120	3118	232	2868	120	2923	154	2678
	150	2802	290	2595	150	2569	170	2370	150	2394	105	2204
	180	2246	229	2075	180	2037	138	1874	180	1871	85	1713
$w = 200, \bar{t} = 369.23$	δ	OG	RG	CO	δ	OG	RG	CO	δ	OG	RG	CO
	60	9749	2233	8018	60	9452	1959	7734	60	9194	1825	7488
	90	8700	1449	7164	90	8426	1232	6902	90	8180	1123	6669
	120	7697	959	6337	120	7431	768	6083	120	7198	672	5863
	150	6787	700	5616	150	6540	535	5378	150	6325	457	5168
	180	5948	538	4926	180	5722	397	4708	180	5510	338	4503
$w = 250, \bar{t} = 374.38$	δ	OG	RG	CO	δ	OG	RG	CO	δ	OG	RG	CO
	60	9277	2834	8396	60	9030	2611	8158	60	8777	2487	7921
	90	8205	1888	7438	90	7972	1693	7212	90	7725	1574	6981
	120	7280	1153	6617	120	7054	976	6400	120	6819	861	6182
	150	6406	680	5809	150	6190	524	5605	150	5963	422	5394
	180	5580	429	5064	180	5381	302	4876	180	5156	231	4667

values of w, the number of maximal groups flattens off, or sometimes even decreases. These results are more apparent for larger values of δ. It seems that if the density becomes higher, pedestrians that are far apart are more likely to form a group. Since the speed of each entity also become slower, then these groups are also bigger (and longer) from other maximal groups.

The number of maximal groups reported by the refined group definition is generally much smaller than the number of maximal groups reported by the other two definitions. This is also clearly visible in Figure 5.18, where we show the number of maximal groups, with $m = 6$, and $w = 200$, as a function of δ. The graphs for different settings of m and w are similar. Here, we also see that the number of maximal groups decreases as we increase the minimum required duration (which is to be expected).

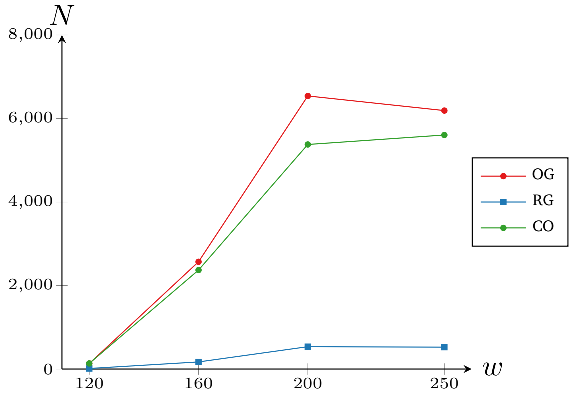


FIGURE 5.17: The number of maximal groups (N) for $m = 6$ and $\delta = 150$ as a function of the width w of the corridor entrance, which influences density.

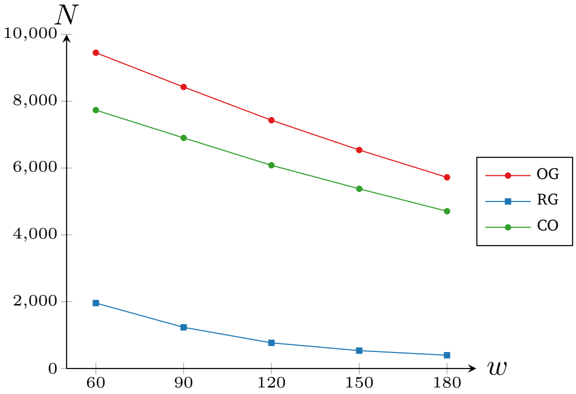


FIGURE 5.18: The number of maximal groups (N) for $m = 6$ and $w = 200$ as a function of δ. There are much fewer maximal groups according to refined groups when compared with original groups and convoys.

Measuring the Conformity of a Group. Since all entities (pedestrians) completely cross the corridor, we can classify each entity as type going “left to right” (type R), or “right to left” (type L). We can extend this notion to groups of entities by taking the type of the majority of its members (in case of ties we pick arbitrarily). We then define the *conformity* $c(\mathcal{G})$ of a group $\mathcal{G}$ as the percentage of its members that have the same type as the type of the group. Hence, the conformity of $\mathcal{G}$ is a value varying from 50, half of the members of $\mathcal{G}$ cross the corridor each way, to 100, all members of $\mathcal{G}$ go in the same direction. Intuitively, we expect that a set of people that act as a group (in the social sense) travel in the same direction, and thus we expect the conformity to be high in a good grouping definition.

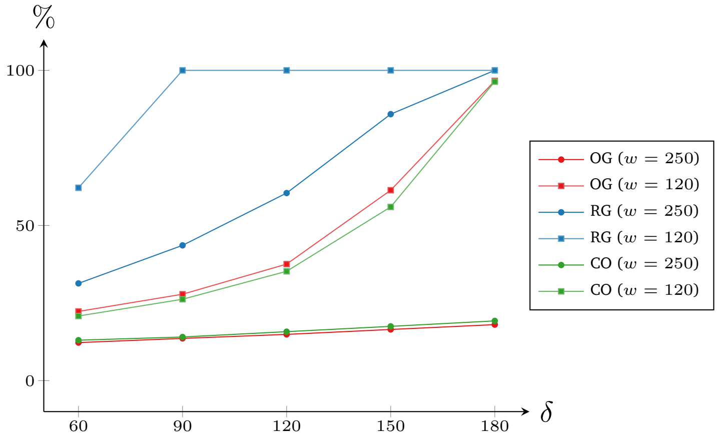


FIGURE 5.19: The percentage of maximal groups with conformity 100 in the pedestrian data for $m = 6$ and $w = 120$ as a function of δ.

We now measure the conformity of all maximal groups reported by all three definitions. Specifically, we consider the percentage of maximal groups that have conformity 100, that is, all group members travel in the same direction. We say that such a group is *uni-directional*. The results are in Figure 5.19. Consider the case where $m = 6$ and $w = 120$. For all definitions, we see that as the minimum required duration increases, so does the percentage of uni-directional maximal groups. However, the refined definition generally has a much higher percentage of uni-directional maximal groups. In particular, for a duration as short as 90 time units (about 5 seconds), all maximal groups are uni-directional. For the original group and the convoy, this requires a minimum duration threshold of more than 180. These results are even more clearly visible as we increase the width of the corridor. For example, for $w = 250$, all maximal groups with a duration of at least $\delta = 90$ are uni-directional, whereas in the original group and the convoy definition, less than 20% of the reported maximal groups are uni-directional, even if we increase the minimum required duration to 180. We expect that this is mostly due to the fact that the original group and the convoy reports many more maximal groups than the refined group.

Research question (2): dependence on density. As expected, all grouping definitions find more groups when the density of entities increases or when connectedness is satisfied at larger distances. The swarm definition has a larger increase in the number of maximal groups than the other definitions. Since we do not have a ground truth or human-annotated groups data, we cannot draw further conclusions from these observations.

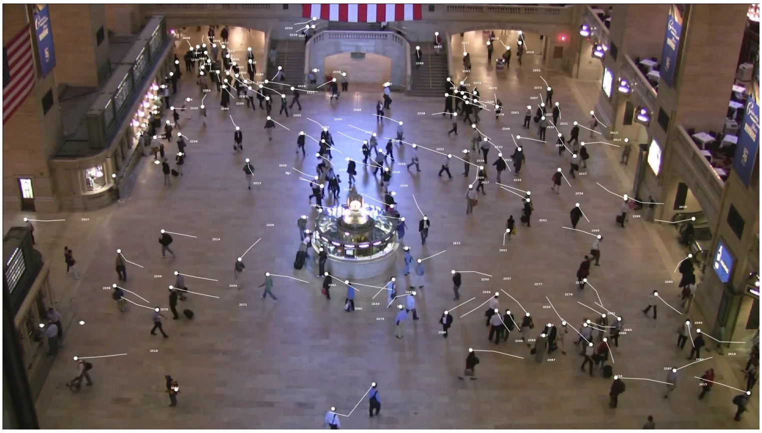


Figure 5.20: The Grand Station and trajectories of pedestrians.

However, in an extreme case where the density is very high and entities only move in opposite directions, the refined group appears to be more natural. The original group and convoy report many groups consisting of entities that move in opposite directions, whereas the refined group finds only a few of them. Moreover, such groups often have a short duration. Another interesting observation is that the refined group gives fewer groups. It is not clear whether this is an advantage or a disadvantage since the nature of all definitions gives rise to groups that share entities at the same time.

5.3.3 Dependence on Sampling Rate

The purpose of our last experiment is to examine how different sampling rates of trajectories affect the maximal groups produced by each definition. We conduct experiments by gradually removing vertices from trajectories, thus decreasing their sampling rate. For each new data set consisting of trajectories with a lower sampling rate, we count how many maximal groups result.

The data set consists of trajectories from pedestrians inside the Grand Central Terminal in New York City, USA (see Figure 5.20).[2] The data set contains 6000 video frames at which data points are generated manually. This is once every 0.8 seconds. There are 12,684 pedestrians, with an average of 105.52 pedestrians in each frame. For our experiment, we choose two sets of 800 consecutive frames that have a high density. The first set contains 2591 trajectories while the second contains 3313 trajectories. The average number of vertices in a trajectory are 46.57 and 46.85, respectively.

---

[2] The background image and movement data are from [193].

TABLE 5.10: The number of maximal groups from 2591 trajectories in the Grand Station data set with different sampling rate.

		25%	50%	100%
$\delta = 8s$	OG	174	178	170
	RG	173	177	169
	CO	140	158	177
	SW	153	180	249
$\delta = 12s$	OG	127	118	116
	RG	127	118	117
	CO	111	122	121
	SW	115	138	199
$\delta = 16s$	OG	97	96	96
	RG	97	97	96
	CO	92	94	97
	SW	98	111	162

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

TABLE 5.11: The number of maximal groups from 3313 trajectories in the Grand Station data set with different sampling rate.

		25%	50%	100%
$\delta = 8s$	OG	276	268	257
	RG	269	262	255
	CO	222	256	264
	SW	229	259	379
$\delta = 12s$	OG	211	206	203
	RG	206	200	200
	CO	190	203	204
	SW	196	215	304
$\delta = 16s$	OG	172	168	159
	RG	167	163	157
	CO	160	156	161
	SW	166	183	252

OG = Original Groups, RG = Refined Groups, CO = Convoys, SW = Swarms

First, we create a homography matrix to map frame coordinates from the data set into ground coordinates. We choose $\varepsilon = 0.76$m for a personal distance between pairs, based on the maximum *close phase* from Proxemics Theory [204]. We vary the required minimum duration for a maximal group $\delta \in \{8, 12, 16\}$ in seconds. Finally, we consider different sampling rates for the two sets of trajectories by taking $25\%, 50\%, 100\%$ of the vertices of the trajectories. Some trajectories may be removed because less than 2 vertices remain. The results of our experiments are in Tables 5.10 and 5.11.

We notice that the number of original groups and refined groups is stable or increases slightly when reducing the sampling rate. In contrast, the number of convoys decreases slightly when reducing the sampling rate, and the number of swarms decreases substantially. This trend is related to the cumulative version of the time duration of swarms. Imagine a swarm with several disconnected time intervals on its time duration (with a sampling rate of 100%). By reducing the sampling rate, some of these time intervals will likely to disappear, or their length is reduced. Therefore, the swarm may not meet the required δ anymore. On the other hand, the other definitions which use consecutive timestamps are not affected much by this situation.

Research question (3): dependence on sampling rate. In general it is preferable when a definition of grouping is not influenced too much by the sampling rate, so in this respect the original and refined group definitions perform a bit better than convoys and much better than swarms.

5.4 Conclusions and Future Work

We experimentally evaluated four definitions for grouping in trajectory data: (original) groups, refined groups, convoys, and swarms. We tried to establish how well these definitions correspond to the human intuition of a group, how the number of groups depends on the density of the entities in their environment, and how the number of groups depends on the sampling rate of the trajectories. In our experiments, the groups, refined groups, and convoys perform similar in terms of recognizing all sets of entities that were a group according to the human annotations, with the refined groups typically having the highest F1-score. On occasion the swarm definition outperforms the other definitions. To be more conclusive in these experiments, we first of all need better human annotation, and second of all, test more data sets and settings.

We observe that the definitions that consider the trajectories to be continuous mappings from time to space (original groups and refined groups) are more stable than the definitions considering the trajectories as discrete input (convoys and swarms) when we consider the number of reported groups under reductions of the sampling rate.

In general, it appears that swarm is most different among the definitions, suggesting that taking group duration cumulatively has a larger effect on grouping than the discrete or continuous handling of the data, or the type of connectedness (see Table 5.1).

In terms of qualitative assessment, we developed a style of video annotation that allows us to compare two different grouping definitions. It is best suited for comparisons to groups from human annotation. Videos using this visualization can be found on our website [203].

For future work, it is possible to extend the experimentation using other pedestrian data sets in different environments such as inside a museum [208]. However, probably it is more interesting to perform experimental evaluations on animals trajectory data set (e.g., from Movebank [32]) since they may exhibit different behavior when grouping for various activities (e.g., foraging, migration) than pedestrians walking together.

6

CONCLUSIONS

In this final chapter, we summarize the results of our work and describe several open problems based on those results. Furthermore, we also discuss possible directions for future research on the study of the collective movement pattern.

6.1 Summary

Although our work was focused on the collective movement pattern, we also considered the problem of simplifying a polyline (polygonal curve) in Chapter 2. We studied the polyline simplification problem such that the output simplification has Hausdorff [67] or Fréchet distance [68] at most $\varepsilon > 0$ to the input polyline. We first compared the output simplifications from the two well-known polyline simplification algorithms, by Douglas-Peucker [134] and by Imai-Iri [136], with the optimum simplification possible. Then, we showed that the polyline simplification problem is NP-Hard under the undirected Hausdorff distance and the directed Hausdorff distance from the input polyline to the simplified output. For the reverse case under the directed Hausdorff distance from the output to the input, and also under the Fréchet distance, we presented algorithms that run in polynomial time to solve the problem.

We started our study on the collective movement pattern in Chapter 3, where we presented various measures for a single trajectory and a group of trajectories. For each measure, along with the name and its description, we also gave more information: the unit and range of the measurements, from which attribute or other measure it is derived, and its related work. First, we described measures for a single trajectory and classified them into three different types: for a single trajectory in isolation, a single trajectory amidst other trajectories, and a single trajectory in an environment. Then, we extended those measures for a group of trajectories using three different views (i.e., representative, complete, and area) and added other measures that do not exist for an individual trajectory. We also discussed several analysis tasks where measures for groups of trajectories may play an important role.

We refined the previous definition of groups by Buchin et al. [126] in Chapter 4. We made a small change in the spatial requirement for groups in the previous definition and presented two scenarios in which we argued that our refined definition will perform better. We provided algorithms to compute the refined groups for entities that move in $\mathbb{R}^1$ or $\mathbb{R}^d (d > 1)$, also for the case when the time component of the input trajectories are not aligned. Both algorithms have polynomial dependence on n (the number of input trajectories) and τ (the number of vertices on each trajectory). Furthermore, we showed that the algorithm for moving entities in $\mathbb{R}^d$ can have linear dependence on τ, although the dependence on n becomes exponential. Improving the dependence on n from exponential to polynomial while keeping the linear dependence on τ appears to be difficult.

Based on our work in Chapter 4, we conducted experiments to compare four definitions of a collective movement pattern: convoys, swarms, groups, and refined groups. The results of the experimentation were presented in Chapter 5. We implemented the algorithm to compute each definition and used various data sets in the experiments. The data sets include artificial trajectories from computer simulations and real-life trajectories of pedestrians in different environments: real-life and synthetic. The experimentation focused on the differences between definitions and human-annotated data in the pedestrian trajectories data (if available). We also analyzed how each definition performed on different crowd densities and on different sampling rates. We presented the results of the quantitative evaluations and developed a visualization that shows groups with a color coding for qualitative evaluation.

6.2 Open Problems

Our work described in this thesis is far from complete and left us with several open problems. From the polyline simplification problem in Chapter 2, we are still interested in the computational status of the optimal simplification under the Hausdorff and the Fréchet distance when the simplification does not need to use the vertices of the input. Note that Kostitsyna et al. [209] and van de Kerkhof et al. [160] already have partial results on these problems. Furthermore, we can use other distance measures to consider the optimal polyline simplification.

From the study of measures for a single trajectory and a group of trajectories in Chapter 3, we have at least two open problems that are worth mentioning. First, we can consider real-world trajectory data and investigate the need for measures beyond the ones we have given. Second, we can analyze how external factors like geographic context should influence measures in various applications.

The running time of our algorithms to compute the refined groups in Chapter 4 depend on the input parameters of n and τ. We have shown that the algorithms for moving entities in $\mathbb{R}^d$ could have either (i) polynomial dependence on both n and τ or (ii) linear dependence on τ but at the expense of exponential dependence in n. The trade-off in the dependence on n and τ gives rise to interesting open problems. Most importantly, is it possible to develop an algorithm whose running time is linear in τ and polynomial in n? Similarly, can we realize subquadratic dependence on τ without having exponential dependence on n? In general, what trade-offs are possible? Furthermore, it would be interesting to develop an output-sensitive algorithm that uses considerably less time if the output is small, or under realistic input assumptions.

The experimentation in Chapter 5 showed that by counting duration cumulatively rather than consecutively, the swarm definition is more robust to noise than the other methods, but at the same time finds more doubtful groups that arise from several short, by-chance encounters. Therefore, more robust grouping definitions can be developed and compared, which would depend on a fourth parameter that describes how noise is handled. Examples are platoons [125] and robust groups [126]. The extra parameter makes proper experimentation harder, however.

6.3 Future Research

Most of this thesis presents our contribution to the study on the analysis of movement data, especially of the collective movement pattern. There are numerous possible directions for interesting future research. Below, we briefly discuss several directions that reach further beyond the scope of this thesis.

Our work in this thesis only used Lagrangian-based trajectory data (see Section 1.2). However, we can expect that the availability of Eulerian-based trajectory data will increase rapidly in the near future. This is due to the use of more and more diverse types of location-aware technologies such as WiFi and RFID tags. For instance, the positions of human entities can be tracked whenever they use WiFi access, pay using digital payment systems in a supermarket, or even check-in or out from public transport systems using digital gates. Recorded tracking data of this type is very different from individual movement traces. We realize that the distinctions between these two types of trajectory data will pose a unique challenge, especially when we want to identify collective movement patterns from the Eulerian-based trajectory data.

In Chapter 5, we used four definitions of collective movement pattern and listed many other definitions. Each one of these definitions has its own advantages and disadvantages. All definitions depend heavily on the spatial component to define the "togetherness" between entities (at certain times). However, the spatial component only concerns the distance between entities. Nevertheless, we can further explore other features to extend the approach to define togetherness so that it is not influenced solely by the distance between entities. We may improve the definition of a collective movement pattern in several ways:

- *Measures*
  We can include measures (see Chapter 3) when defining a collective movement pattern. This can be done, for example, by considering measures of a trajectory when the associated entity is being examined at any given time. There are several measures that we can use in this scenario such as average speed, global direction, or global velocity. Moreover, we can also consider measures (for groups) as additional requirements for a set of moving entities to be considered as a group. For example, we can set a fixed value f and require that the formation stability value of a group (see Section 3.2.1) is smaller than f. In the same way, the density measure for a group can also be used.
- *Contextual Information*
  So far, most definitions focus on raw trajectories containing only the spatial and temporal information. However, it is crucial to take contextual information (e.g., environment data, weather data) into account and integrate them into trajectory data [74, 210]. Consequently, incorporating contextual information to enrich trajectory data will lead to various new approaches to define a collective movement pattern.
- *Interactions*
  Naturally, entities from the same group will have interactions with one another. The form of the interaction can be of various kinds. For example, human pedestrians will probably have a conversation or hold hands, whereas animals communicate through vocalization. We can consider interaction differently: when an entity adjusts its movement in response to the change of movement and position of

other entities in the group. Hence, the interaction between two entities can be defined, for example, as the difference in their turning angle or in their heading direction. Recently, Loglisci [211] define a new model for a collective movement pattern called *crews* that take interactions into account. He used parameters such as the distance ratio and the tortuosity ratio to express interactions between two entities. However, we believe that the opportunities are still wide open to expand the research on the collective movement pattern that considers interactions among entities.

Integrating some or all features listed above to the computation of collective movement patterns may significantly increase the computing time. On the other hand, the availability of the massive amount of trajectory data already lets researchers face the challenge to develop more efficient algorithms to process and analyze them. Therefore, the combination of both situations may lead to algorithmic problems whose solutions are computationally very expensive. Hence, it is interesting to explore the possibility to apply some heuristic methods in the algorithms to compute groups. However, these heuristics should be tested extensively using real-life trajectory data sets and most likely will depend on specific applications. Furthermore, one can also develop algorithms that only work on trajectory data that has specific features (e.g., geographic context information) to solve a domain-specific problem.

Next, we also consider the visualization of collective movement patterns that integrate features that we discussed above. In Chapter 5, we gave a qualitative analysis and developed a visualization to show groups in video footage. However, the visualization can only show the movement traces and groups from various definitions. Therefore, it may be interesting to visually include contextual information like geographic data or interactions like the direction or the difference of heading between entities from the same group. We believe a proper visual encoding for different aspects of groups will allow researchers to analyze trajectory data accurately in a more convenient way.

Finally, we want to emphasize the necessity to collaborate with researchers from other domain areas (e.g., ecology, behavioral sciences) to expand the study of collective movement patterns. In a domain-specific case, enriching trajectory data with context information or using measures to define groups of entities may require knowledge from experts in the respective domains. Furthermore, how we visualize the collective movement pattern is also crucial for experts to clearly analyze their trajectory data and formulate new hypotheses through visual observation.

BIBLIOGRAPHY

[1] Francesco Solera. group-detection. http://imagelab.unimore.it/group-detection/. Last accessed: August 1, 2019.

[2] Diann Prosser, Eric Palm, John Takekawa, Delong Zhao, Xiangming Xiao, Peng Li, Ying Liu, and Scott H. Newman. Movement analysis of free-grazing domestic ducks in Poyang lake, China: a disease connection. *International Journal of Geographical Information Science*, 30(5):869–880, 2016.

[3] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[4] Paul Longley, Michael Goodchild, David Maguire, and David Rhind. *Geographic Information Science and Systems*. Wiley, 4th edition, 2015.

[5] Marc van Kreveld. Geographic information systems. In Jacob Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 1293–1314. Chapman & Hall/CRC, 2nd edition, 2004.

[6] Donna Peuquet. Time in GIS and geographical databases. In Paul Longley, Michael Goodchild, David Maguire, and David Rhind, editors, *Geographical Information Systems: Principles, Techniques, Management and Applications, Abridged*. Wiley, 2nd edition, 2005.

[7] May Yuan. Temporal GIS and applications. In Shashi Shekhar and Hui Xiong, editors, *Encyclopedia of GIS*. Springer US, 2008.

[8] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Computational movement analysis. In Wolfgang Kresse and David Danko, editors, *Handbook of Geographic Information*, pages 725–741. Springer, 2012.

[9] Gennady Andrienko, Natalia Andrienko, Peter Bak, Daniel Keim, and Stefan Wrobel. *Visual Analytics of Movement*. Springer, 2013.

[10] Patrick Laube. *Computational Movement Analysis*. Springer, 2014.

[11] Patrick Laube, Stephan Imfeld, and Robert Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005.

[12] Yann Tremblay, Scott Shaffer, Shannon Fowler, Carey Kuhn, Birgitte McDonald, Michael Weise, Charle-André Bost, Henri Weimerskirch, Daniel Crocker, Michael Goebel, and Daniel Costa. Interpolation of animal tracking data in a fluid environment. *Journal of Experimental Biology*, 209(1):128–140, 2006.

[13] Elizabeth Wentz, Aimee Campbell, and Robert Houston. A comparison of two methods to create tracks of moving objects: Linear weighted distance and constrained random walk. *International Journal of Geographical Information Science*, 17(7):623–645, 2003.

[14] Jed A. Long. Kinematic interpolation of movement data. *International Journal of Geographical Information Science*, 30(5):854–868, 2016.

[15] Kevin Buchin, Stef Sijben, Jean Marie Arseneau, and Erik Willems. Detecting movement patterns using Brownian bridges. In *Proc. of the 2012 International Conference on Advances in Geographic Information Systems, SIGSPATIAL'12*, pages 119–128, 2012.

[16] Jon Horne, Edward Garton, Stephen Krone, and Jesse Lewis. Analyzing animal movements using brownian bridges. *Ecology*, 88(9):2354–2363, 2007.

[17] Simon Benhamou. How to reliably estimate the tortuosity of an animal's path: straightness, sinuosity, or fractal dimension? *Journal of Theoretical Biology*, 229:209–220, 2004.

[18] Lucille Stickel. Populations and home range relationships of the box turtle, terrapene c. carolina (linnaeus). *Ecological Monographs*, 20(4):351–378, 1950.

[19] Dennis Claussen, Michael Finkler, and Meghan Smith. Thread trailing of turtles: Methods for evaluating spatial movements and pathway structure. *Canadian Journal of Zoology*, 75(12):2120–2128, 1997.

[20] Peter Waser. Monthly variations in feeding and activity patterns of the mangabey, cercocebus albigena (lydekker). *African Journal of Ecology*, 13(3-4):249–263, 1975.

[21] Julia Van Velden, Res Altwegg, Kevin Shaw, and Peter Ryan. Movement patterns and survival estimates of blue cranes in the Western Cape. *Ostrich*, 88(1):33–43, 2017.

[22] Petra Kaczensky, Felix Knauer, Blaz Krze, Marco Jonozovic, Miha Adamic, and Hartmut Gossow. The impact of high speed, high volume traffic axes on brown bears in Slovenia. *Biological Conservation*, 111(2):191–204, 2003.

[23] Pacific Northwest Research Station. The starkey project. https://www.fs.fed.us/pnw/starkey/index.shtml. Last accessed: August 1, 2019.

[24] Rui Prieto, Mónica Silva, Gordon Waring, and João Gonçalves. Sei whale movements and behaviour in the North Atlantic inferred from satellite telemetry. *Endangered Species Research*, 26(2):103–113, 2014.

[25] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. In *Proc. of the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010*, pages 99–108, 2010.

[26] Yu Zheng. T-drive trajectory data sample. https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample. Last accessed: August 1, 2019.

[27] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–39, 2010.

[28] Microsoft. Geolife GPS trajectories. https://www.microsoft.com/en-us/download/details.aspx?id=52367. Last accessed: August 1, 2019.

[29] Andrej Dobrkovic, Maria-Eugenia Iacob, and Jos van Hillegersberg. Maritime pattern extraction and route reconstruction from incomplete AIS data. *International Journal of Data Science and Analytics*, 5(2-3):111–136, 2018.

[30] The Danish Maritime Authority. AIS data. https://www.dma.dk/SikkerhedTilSoes/Sejladsinformation/AIS. Last accessed: August 1, 2019.

[31] National Hurricance Center and Central Pacific Hurricance Center. NHC data archive. https://www.nhc.noaa.gov/data. Last accessed: August 1, 2019.

[32] Max Planck Institute for Animal Behavior, North Carolina Museum of Natural Sciences, and University of Konstanz. Movebank for animal tracking data. https://www.movebank.org. Last accessed: August 1, 2019.

[33] Dražen Brščić, Takayuki Kanda, Tetsushi Ikeda, and Takahiro Miyashita. Person tracking in large public spaces using 3-D range sensors. *IEEE Transactions on Human-Machine Systems*, 43(6):522–534, 2013.

[34] Dražen Brščić and Takayuki Kanda. Changes in usage of an indoor public space: Analysis of one year of person tracking. *IEEE Transactions on Human-Machine Systems*, 45(2):228–237, 2015.

[35] ATR Intelligent Robotics and Communication Laboratories. Datasets from the ATC shopping center. https://irc.atr.jp/crest2010_HRI/ATC_dataset. Last accessed: August 1, 2019.

[36] Apurva Joshi, Naga VishnuKanth, Navkar Samdaria, Sumit Bagla, and Prabhat Ranjan. GPS-less animal tracking system. In *Proc. of the 4th International Conference on Wireless Communication and Sensor Networks, WSCN 2008*, pages 120–125, 2008.

[37] Gabriel Michau, Alfredo Nantes, Ashish Bhaskar, Edward Chung, Patrice Abry, and Pierre Borgnat. Bluetooth data in an urban context: Retrieving vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2377–2386, 2017.

[38] Cigdem Beyan and Robert Fisher. Detecting abnormal fish trajectories using clustered and labeled data. In *Proc. of the IEEE International Conference on Image Processing, ICIP 2013*, pages 1476–1480, 2013.

[39] Robert Fisher. Fish trajectory ground truth dataset. https://groups.inf.ed.ac.uk/vision/FISH4KNOWLEDGE/WEBSITE/GROUNDTRUTH/BEHAVIOR. Last accessed: August 1, 2019.

[40] Gang Xiao, Yi Li, Tengfei Shao, and Zhenbo Cheng. Prediction of individual fish trajectory from its neighbors' movement by a recurrent neural network. In *Proc. of the 12th International Symposium on Neural Networks, ISNN 2015*, pages 390–397, 2015.

[41] Robert Fisher. Edinburgh overhead camera person tracking dataset. https://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING. Last accessed: August 1, 2019.

[42] Robert Collins. PSU HUB dataset. http://vision.cse.psu.edu/data/data.shtml. Last accessed: August 1, 2019.

[43] Andreas Huth and Christian Wissel. The simulation of the movement of fish schools. *Journal of Theoretical Biology*, 156(3):365 – 385, 1992.

[44] Uri Wilensky. Netlogo. http://ccl.northwestern.edu/netlogo. Last accessed: August 1, 2019.

[45] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *Proc. of the International Conference on Complex Systems*, volume 21, pages 16–21, 2004.

[46] Uri Wilensky and William Rand. *An Introduction to Agent-based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press, 2015.

[47] Daniel Thalmann and Soraia Raupp Musse. *Crowd Simulation*. Springer, 2nd edition, 2013.

[48] Wesley Kerr and Diana Spears. Robotic simulation of gases for a surveillance task. In *Proc. of the International Conference on Intelligent Robots and Systems IEEE/RSJ 2005*, pages 2905–2910, 2005.

[49] Craig Reynolds. Steering behaviors for autonomous characters. In *Proc. of the Game Developers Conference*, volume 1999, pages 763–782, 1999.

[50] Sujeong Kim, Stephen Guy, and Dinesh Manocha. Velocity-based modeling of physical interactions in multi-agent simulations. In *Proc.of The ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA '13*, pages 125–133, 2013.

[51] Nuria Pelechano, Jan Allbeck, and Norman Badler. Controlling individual agents in high-density crowd simulation. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007*, pages 99–108, 2007.

[52] Christian Gloor. Pedestrian crowd simulation. http://pedsim.silmaril.org/. Last accessed: August 1, 2019.

[53] Wouter van Toll, Norman Jaklin, and Roland Geraerts. Towards believable crowds: A generic multi-level framework for agent navigation. In *ICT.OPEN 2015*, 2015.

[54] Roland Geraerts. Crowd simulation software. https://www.staff.science.uu.nl/~gerae101/UU_crowd_simulation_software.html. Last accessed: August 1, 2019.

[55] Richard Mann, Allan Jepson, and Thomas El-Maraghi. Trajectory segmentation using dynamic programming. In *Proc. of the 16th International Conference on Pattern Recognition, ICPR 2002*, pages 331–334, 2002.

[56] Aris Anagnostopoulos, Michail Vlachos, Marios Hadjieleftheriou, Eamonn Keogh, and Philip Yu. Global distance-based segmentation of trajectories. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2006.

[57] Maike Buchin, Anne Driemel, Marc van Kreveld, and Vera Sacristán. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proc. of the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010*, pages 202–211, 2010.

[58] Boris Aronov, Anne Driemel, Marc van Kreveld, Maarten Löffler, and Frank Staals. Segmentation of trajectories on nonmonotone criteria. *ACM Transactions on Algorithms*, 12(2):26:1–26:28, 2016.

[59] Katarzyna Sila-Nowicka, Jan Vandrol, Taylor Oshan, Jed Long, Urska Demsar, and Stewart Fotheringham. Analysis of human mobility patterns from GPS trajectories and contextual information. *International Journal of Geographical Information Science*, 30(5):881–906, 2016.

[60] Hendrik Edelhoff, Johannes Signer, and Niko Balkenhol. Path segmentation for beginners: an overview of current methods for detecting changes in animal movement patterns. *Movement Ecology*, 4(1):21, 2016.

[61] Maike Buchin, Helmut Kruckenberg, and Andrea Kölzsch. Segmenting trajectories by movement states. In Sabine Timpf and Patrick Laube, editors, *Advances in Spatial Data Handling: Geospatial Dynamics, Geosimulation and Exploratory Visualization*, pages 15–25. Springer Berlin Heidelberg, 2013.

[62] Monika Sester, Udo Feuerhake, Colin Kuntzsch, and Lijuan Zhang. Revealing underlying structure and behaviour from movement data. *Künstliche Intelligenz*, 26(3):223–231, 2012.

[63] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web*, 4(1):1:1–1:36, 2010.

[64] Kevin Buchin, Maike Buchin, Marc van Kreveld, and Jun Luo. Finding long and similar parts of trajectories. *Computational Geometry*, 44(9):465–476, 2011.

[65] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(3):253–282, 2011.

[66] Faisal Bashir, Ashfaq Khokhar, and Dan Schonfeld. Segmented trajectory based indexing and retrieval of video data. In *Proc. of the 2003 International Conference on Image Processing, ICIP 2003*, pages 623–626, 2003.

[67] Günter Rote. Computing the minimum Hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127, 1991.

[68] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1-2):75–91, 1995.

[69] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[70] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[71] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. Discovering similar multidimensional trajectories. In *Proc. of the 18th International Conference on Data Engineering*, pages 673–684, 2002.

[72] Hechen Liu and Markus Schneider. Similarity measurement of moving object trajectories. In *Proc. of the 3rd ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS@SIGSPATIAL 2012*, pages 19–22, 2012.

[73] Somayeh Dodge, Robert Weibel, and Ehsan Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, 33(6):419–434, 2009.

[74] Maike Buchin, Somayeh Dodge, and Bettina Speckmann. Similarity of trajectories taking into account geographic context. *Journal of Spatial Information Science*, 9(1):101–124, 2014.

[75] Gavin McArdle, Urska Demsar, Stefan van der Spek, and Seán McLoone. Classifying pedestrian movement behaviour from GPS trajectories using visualization and clustering. *Annals of GIS*, 20(2):85–98, 2014.

[76] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR 2006)*, pages 1135–1138, 2006.

[77] Eamonn Keogh and Chotirat Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[78] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. TraClass: Trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[79] Li Cai, Sijin Li, Shipu Wang, and Yu Liang. GPS trajectory clustering and visualization analysis. *Annals of Data Science*, 5(1):29–42, 2018.

[80] Marcus Doherr, Tim Carpenter, David Wilson, and Ian Gardner. Evaluation of temporal and spatial clustering of horses with Corynebacterium pseudotuberculosis infection. *American Journal of Veterinary Research*, 60(3):284–291, 1999.

[81] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[82] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[83] Huey-Ru Wu, Mi-Yen Yeh, and Ming-Syan Chen. Profiling moving objects by dividing and clustering trajectories spatiotemporally. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2615–2628, 2013.

[84] Maximilian Konzack, Pieter Gijsbers, Ferry Timmers, Emiel van Loon, Michel Westenberg, and Kevin Buchin. Visual exploration of migration patterns in gull data. *Information Visualization*, 18(1), 2019.

[85] Yingchi Mao, Haishi Zhong, Hai Qi, Ping Ping, and Xiaofang Li. An adaptive trajectory clustering method based on grid and density in mobile pattern analysis. *Sensors*, 17(9):2013, 2017.

[86] Scott Gaffney, Andrew Robertson, Padhraic Smyth, Suzana Camargo, and Michael Ghil. Probabilistic clustering of extratropical cyclones using regression mixture models. *Climate Dynamics*, 29(4):423–440, 2007.

[87] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007.

[88] Kevin Buchin, Maike Buchin, Marc van Kreveld, Maarten Löffler, Rodrigo Silveira, Carola Wenk, and Lionov Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

[89] Marc van Kreveld, Maarten Löffler, and Frank Staals. Central trajectories. *Journal of Computational Geometry*, 8(1):366–386, 2017.

[90] Willem Eerland and Simon Box. Trajectory clustering, modeling and selection with the focus on airspace protection. In *AIAA Infotech@ Aerospace*, page 1411. ARC, 2016.

[91] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristán, Rodrigo Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In *Proc. of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016*, pages 14:1–14:10, 2017.

[92] Wei Jiang, Jie Zhu, Jiajie Xu, Zhixu Li, Pengpeng Zhao, and Lei Zhao. A feature based method for trajectory dataset segmentation and profiling. *World Wide Web*, 20(1):5–22, 2017.

[93] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008.

[94] Hoyoung Jeung, Man Lung Yiu, and Christian Jensen. Trajectory pattern mining. In Yu Zheng and Xiaofang Zhou, editors, *Computing with Spatial Trajectories*, pages 143–177. Springer, 2011.

[95] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Movement patterns in spatio-temporal data. In Shashi Shekhar, Hui Xiong, and Xun Zhou, editors, *Encyclopedia of GIS*, pages 1362–1370. Springer, 2017.

[96] Bojan Djordjevic, Joachim Gudmundsson, Anh Pham, and Thomas Wolle. Detecting regular visit patterns. *Algorithmica*, 60(4):829–852, 2011.

[97] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 236–245, 2004.

[98] Joachim Gudmundsson, Marc van Kreveld, and Frank Staals. Algorithms for hotspot computation on trajectory data. In *Proc. of the 21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013*, pages 134–143, 2013.

[99] Marc Benkert, Bojan Djordjevic, Joachim Gudmundsson, and Thomas Wolle. Finding popular places. *International Journal of Computational Geometry & Applications*, 20(1):19–42, 2010.

[100] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Proc. of the 24th International Conference on Data Engineering, ICDE 2008*, pages 70–79, 2008.

[101] Fernando de Lucca Siqueira and Vania Bogorny. Discovering chasing behavior in moving object trajectories. *Transactions in GIS*, 15(5):667–688, 2011.

[102] Philip Blythe, Geoffrey Miller, and Peter Todd. Human simulation of adaptive behavior: Interactive studies of pursuit, evasion, courtship, fighting, and play. In Pattie Maes, Maja Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (Complex Adaptive Systems)*, pages 13–22. MIT Press, 1996.

[103] Francesco Lettich, Luis Otávio Alvares, Vania Bogorny, Salvatore Orlando, Alessandra Raffaetà, and Claudio Silvestri. Detecting avoidance behaviors between moving object trajectories. *Data & Knowledge Engineering*, 102:22–41, 2016.

[104] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

[105] Bertrand Dumont, Alain Boissy, Cècile Achard, Angela Sibbald, and Hans Erhard. Consistency of animal order in spontaneous group movements allows the measurement of leadership in a group of grazing heifers. *Applied Animal Behaviour Science*, 95(1):55 – 66, 2005.

[106] Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. Detecting single file movement. In *Proc. of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008*, page 33, 2008.

[107] Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica*, 11(2):195–215, 2007.

[108] Armand Jacobs, Cédric Sueur, Jean Louis Deneubourg, and Odile Petit. Social network influences decision making during collective movements in brown lemurs (Eulemur fulvus fulvus). *International Journal of Primatology*, 32(3):721–736, 2011.

[109] Frances Colles, Russell Cain, Thomas Nickson, Adrian Smith, Stephen Roberts, Martin Maiden, Daniel Lunn, and Marian Stamp Dawkins. Monitoring chicken flock behaviour provides early warning of infection by human pathogen Campylobacter. *Biological Sciences*, 283(1822):20152323, 2016.

[110] Kiran Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter Rentfrow, Chris Longworth, and Andrius Aucinas. Emotionsense: A mobile phones based adaptive platform for experimental social psychology research. In *Proc. of the 12th ACM International Conference on Ubiquitous Computing*, pages 281–290, 2010.

[111] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, pages 35–42, 2006.

[112] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

[113] Marcos Vieira, Petko Bakalov, and Vassilis Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009*, pages 286–295, 2009.

[114] Yifan Li, Jiawei Han, and Jiong Yang. Clustering moving objects. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–622, 2004.

[115] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *Proc. of the Advances in Spatial and Temporal Databases, 9th International Symposium, SSTD 2005*, pages 364–381, 2005.

[116] Christian Jensen, Dan Lin, and Beng Chin Ooi. Continuous clustering of moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1161–1174, 2007.

[117] San-Yih Hwang, Ying-Han Liu, Jeng-Kuen Chiu, and Ee-Peng Lim. Mining mobile group patterns: A trajectory-based approach. In *Proc. of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD'05*, pages 713–718, 2005.

[118] Yan Huang, Cai Chen, and Pinliang Dong. Modeling herds and their evolvements from trajectory data. In *Proc. of the 5th International Conference of Geographic Information Science, GIScience*, pages 90–105, 2008.

[119] Htoo Htet Aung and Kian-Lee Tan. Discovery of evolving convoys. In *Proc. of the Scientific and Statistical Database Management, 22nd International Conference, (SSDBM 2010)*, pages 196–213, 2010.

[120] Jeremy Yeoman and Matt Duckham. Decentralized detection and monitoring of convoy patterns. *International Journal of Geographical Information Science*, 30(5):993–1011, 2016.

[121] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1):723–734, 2010.

[122] Zhenhui Li, Jiawei Han, Ming Ji, Lu-An Tang, Yintao Yu, Bolin Ding, Jae-Gil Lee, and Roland Kays. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Transactions on Intelligent Systems and Technology*, 2(4):37:1–37:32, 2011.

[123] Lu-An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Chih-Chieh Hung, and Wen-Chih Peng. On discovery of traveling companions from streaming trajectories. In *Proc. of the 2012 IEEE 28th International Conference on Data Engineering (ICDE '12)*, pages 186–197, 2012.

[124] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, Shuo Shang, and Xiaofang Zhou. Online discovery of gathering patterns over trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1974–1988, 2014.

[125] Yuxuan Li, James Bailey, and Lars Kulik. Efficient mining of platoon patterns in trajectory databases. *Data & Knowledge Engineering*, 100:167–187, 2015.

[126] Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. *Journal of Computational Geometry*, 6(1):75–98, 2015.

[127] Yida Wang, Ee-Peng Lim, and San-Yih Hwang. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering*, 57(3):240–282, 2006.

[128] Hengfeng Li, Lars Kulik, and Kotagiri Ramamohanarao. Spatio-temporal trajectory simplification for inferring travel paths. In *Proc. of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information*, pages 63–72, 2014.

[129] Haizhong Qian and Yongmei Lu. Simplifying GPS trajectory data with enhanced spatial-temporal constraints. *ISPRS International Journal of Geo-Information*, 6(11):329, 2017.

[130] Luis Felipe Sánchez-Heres. Simplification and event identification for ais trajectories: The equivalent passage plan method. *The Journal of Navigation*, 72(2):307–320, 2019.

[131] Katerina Vrotsou, Halldor Janetzko, Carlo Navarra, Georg Fuchs, David Spretke, Florian Mansmann, Natalia Andrienko, and Gennady Andrienko. Simplifly: A methodology for simplification and thematic enhancement of trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 21(1):107–121, 2015.

[132] Hu Cao, Ouri Wolfson, and Goce Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *International Journal on Very Large Data Bases*, 15(3):211–228, 2006.

[133] Joachim Gudmundsson, Jyrki Katajainen, Damian Merrick, Cahya Ong, and Thomas Wolle. Compressing spatio-temporal trajectories. *Computational Geometry*, 42(9):825–841, 2009.

[134] David Douglas and Thomas Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973.

[135] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, 1995.

[136] Hiroshi Imai and Masao Iri. Polygonal approximations of a curve - Formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology: A Computational Geometric Approach to the Analysis of Form*, volume 6, pages 71–86. North-Holland, 1988.

[137] Lilian Buzer. Optimal simplification of polygonal chains for subpixel-accurate rendering. *Computational Geometry*, 42(1):45–59, 2009.

[138] Mark de Berg, Marc van Kreveld, and Stefan Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and Geographic Information Systems*, 25(4):243–257, 1998.

[139] Mohammad Ali Abam, Mark de Berg, Peter Hachenberger, and Alireza Zarei. Streaming algorithms for line simplification. *Discrete & Computational Geometry*, 43(3):497–515, 2010.

[140] Gill Barequet, Danny Chen, Ovidiu Daescu, Michael Goodrich, and Jack Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002.

[141] Danny Chen, Ovidiu Daescu, John Hershberger, Peter Kogge, Ningfang Mi, and Jack Snoeyink. Polygonal path simplification with angle constraints. *Computational Geometry*, 32(3):173–187, 2005.

[142] Leonidas Guibas, John Hershberger, Joseph Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 03(04):383–415, 1993.

[143] Regina Estkowski and Joseph Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Proc. of the 17th Annual ACM Symposium on Computational Geometry*, pages 40–49, 2001.

[144] Stefan Funke, Thomas Mendel, Alexander Miller, Sabine Storandt, and Maria Wiebe. Map simplification with topology constraints: Exactly and in practice. In *Proc. of the 19th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 185–196, 2017.

[145] Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In *Proc. of the 35th International Symposium on Computational Geometry, SoCG 2019*, pages 18:1–18:16, 2019.

[146] Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the Hausdorff and Fréchet distance. In *Proc. of the 34th International Symposium on Computational Geometry, SoCG 2018*, pages 56:1–56:14, 2018.

[147] Lionov Wiratma, Marc van Kreveld, and Maarten Löffler. On measures for groups of trajectories. In *Societal Geo-innovation - Selected Papers of the 20th AGILE Conference on Geographic Information Science*, pages 311–330, 2017.

[148] Salman Parsa. A deterministic O(m log m) time algorithm for the reeb graph. *Discrete & Computational Geometry*, 49(4):864–878, 2013.

[149] Marc van Kreveld, Maarten Löffler, Frank Staals, and Lionov Wiratma. A refined definition for groups of moving entities and its computation. *International Journal of Computational Geometry & Applications*, 28(2):181–196, 2018.

[150] Angelos Kremyzas, Norman Jaklin, and Roland Geraerts. Towards social behavior in virtual-agent navigation. *SCIENCE CHINA Information Sciences*, 59(11):1–17, 2016.

[151] Lionov Wiratma, Maarten Löffler, and Frank Staals. An experimental comparison of two definitions for groups of moving entities (short paper). In *Proc. of the 10th International Conference on Geographic Information Science, GIScience 2018*, pages 64:1–64:6, 2018.

[152] Lionov Wiratma, Marc van Kreveld, Maarten Löffler, and Frank Staals. An experimental evaluation of grouping definitions for moving entities. In *Proc. of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019*, 2019. To appear.

[153] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017.

[154] John Hershberger and Jack Snoeyink. An $O(n \log n)$ implementation of the douglas- peucker algorithm for line simplification. In *Proc. of the 10th Annual ACM Symposium on Computational Geometry*, pages 383–384, 1994.

[155] W.S. Chan and Francis Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry & Applications*, 06(01):59–77, 1996.

[156] Avraham Melkman and Joseph O'Rourke. On polygonal chain approximation. In Godfried T. Toussaint, editor, *Computational Morphology: A Computational Geometric Approach to the Analysis of Form*, pages 87–95. North-Holland, 1988.

[157] Pankaj Agarwal, Sariel Har-Peled, Nabil Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3-4):203–219, 2005.

[158] Michael Godau. A natural metric for curves - Computing the distance for polygonal chains and approximation algorithms. In *Proc. of the 8th Annual Symposium on Theoretical Aspects of Computer Science, STACS 91*, pages 127–136, 1991.

[159] Pankaj Agarwal, Mark de Berg, Jie Gao, Leonidas Guibas, and Sariel Har-Peled. Staying in the middle: Exact and approximate medians in R1 and R2 for moving points. In *Proc. of the 17th Canadian Conference on Computational Geometry, CCCG'05*, pages 43–46, 2005.

[160] Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *Proc. of the 27th Annual European Symposium on Algorithms, ESA 2019*, pages 67:1–67:14, 2019.

[161] Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *Journal of Information Processing*, 3(2):73–76, 1980.

[162] Jurek Czyzowicz, Evangelos Kranakis, and Jorge Urrutia. A simple proof of the representation of bipartite planar graphs as the contact graphs of orthogonal straight line segments. *Information Processing Letters*, 66(3):125–126, 1998.

[163] Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

[164] Anil Kumar, Sunil Arya, and Hariharan Ramesh. Hardness of set cover with intersection 1. In *Proc. of 27th International Colloquium of the Automata, Languages and Programming, ICALP 2000*, pages 624–635, 2000.

[165] Maike Buchin, Anne Driemel, Marc van Kreveld, and Vera Sacristán. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, 3(1):33–63, 2011.

[166] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory outlier detection: A partition-and-detect framework. In *Proc. of the 24th International Conference on Data Engineering, ICDE 2008*, pages 140–149, 2008.

[167] Xiaolei Li, Zhenhui Li, Jiawei Han, and Jae-Gil Lee. Temporal outlier detection in vehicle traffic data. In *Proc. of the 25th International Conference on Data Engineering, ICDE 2009*, pages 1319–1322, 2009.

[168] Patrick Laube, Todd Dennis, Pip Forer, and Mike Walker. Movement beyond the snapshot – Dynamic analysis of geospatial lifelines. *Computers, Environment and Urban Systems*, 31(5):481 – 501, 2007.

[169] Natalia Andrienko, Gennady Andrienko, Nikos Pelekis, and Stefano Spaccapietra. Basic concepts of movement data. In Fosca Giannotti and Dino Pedreschi, editors, *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*, pages 15–38. Springer, 2008.

[170] Clément Calenge, Stéphane Dray, and Manuela Royer-Carenzi. The concept of animals' trajectories from a data analysis perspective. *Ecological Informatics*, 4(1):34–41, 2009.

[171] Elodie Buard and Mickaël Brasebin. Visual exploration of large animal trajectories. In *Proc. of the 25th International Cartographic Conference (ICC'11)*, page 7, 2011.

[172] Scott LaPoint, Paul Gallery, Martin Wikelski, and Roland Kays. Animal behavior, cost-based corridor models, and real corridors. *Landscape Ecology*, 28(8):1615, 2013.

[173] Kamran Safi, Bart Kranstauber, Rolf Weinzierl, Larry Griffin, Eileen Rees, David Cabot, Sebastian Cruz, Carolina Proaño, John Takekawa, Scott Newman, Jonas Waldenström, Daniel Bengtsson, Roland Kays, Martin Wikelski,

and Gil Bohrer. Flying with the wind: Scale dependency of speed and direction measurements in modelling wind support in avian flight. *Movement Ecology*, 1(1):4, 2013.

[174] Peter Ranacher and Katerina Tzavella. How to compare movement? a review of physical movement similarity measures in geographic information science and beyond. *Cartography and Geographic Information Science*, 41(3), 2014.

[175] Ephraim Hanks, Mevin Hooten, Devin Johnson, and Jeremy Sterling. Velocity-based movement modeling for individual and population level inference. *PLoS ONE*, 6(8):1–17, 08 2011.

[176] Sue Boinski and Paul Garber. *On the Move: How and Why Animals Travel in Groups*. The University of Chicago Press, 2000.

[177] Luca Giuggioli, Jonathan Potts, and Stephen Harris. Animal interactions and the emergence of territoriality. *PLoS Computational Biology*, 7(3):1–9, 2011.

[178] Irene Giardina. Collective behavior in animal groups: theoretical models and empirical studies. *Human Frontier Science Program Journal*, 2(4):205–219, 2008.

[179] Guy Beauchamp. Flock size and density influence speed of escape waves in semipalmated sandpipers. *Animal Behaviour*, 83(4):1125–1129, 2012.

[180] Stefan Peters and Jukka Krisp. Density calculation for moving points. In *Proc. of the 13th AGILE International Conference on Geographic Information Science*, pages 43–46, 2010.

[181] Frank Heppner. Three-dimensional structure and dynamics of bird flocks. In Julia Parrish and William Hamner, editors, *Animal Groups in Three Dimensions: How Species Aggregate*, page 68–89. Cambridge University Press, 1st edition, 1997.

[182] Steven Viscido, Julia Parrish, and Daniel Grünbaum. Individual behavior and emergent properties of fish schools: A comparison of observation and theory. *Marine Ecology Progress Series*, 273:239–249, 2004.

[183] José Bento. A metric for sets of trajectories that is practical and mathematically consistent. *CoRR*, abs/1601.03094, 2016.

[184] Yossi Rubner, Carlo Tomasi, and Leonidas Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[185] Marc van Kreveld, René van Oostrum, and Jack Snoeyink. Efficient settlement selection for interactive display. In *Proc. of the Auto-Carto XIII International Symposium on Computer-Assisted Cartography*, pages 287–296, 1997.

[186] Timofey Samsonov and A. Krivosheina. Joint generalization of city points and road network for small-scale mapping. In *Proc. of the 7th International Conference on Geographic Information Science, GIScience*, 2012.

[187] Arthur van Goethem, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals. Grouping time-varying data for interactive exploration. In *Proc. of the 32nd International Symposium on Computational Geometry, SoCG 2016*, pages 61:1–61:16, 2016.

[188] Irina Kostitsyna, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals. Trajectory grouping structure under geodesic distance. In *Proc. of the 31st International Symposium on Computational Geometry, SoCG 2015*, pages 674–688, 2015.

[189] Pankaj Agarwal and Micha Sharir. Davenport-schinzel sequences and their geometric applications. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. North Holland / Elsevier, 2000.

[190] Lu An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Wen-Chih Peng, and Thomas La Porta. A framework of traveling companion discovery on trajectory data streams. *ACM Transactions on Intelligent Systems and Technology*, 5(1):3:1–3:34, 2013.

[191] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[192] Shuai Yi, Hongsheng Li, and Xiaogang Wang. Understanding pedestrian behaviors from stationary crowd groups. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pages 3488–3496, 2015.

[193] Shuai Yi. Pedestrian walking path dataset. https://www.dropbox.com/s/7y90xsxq0l0yv8d/cvpr2015_pedestrianWalkingPathdataset.rar. Last accessed: August 1, 2019.

[194] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. of the 12th IEEE International Conference on Computer Vision, ICCV 2009*, pages 261–268, 2009.

[195] ETH Zürich Computer Vision Lab. ETHZ - computer vision lab: Datasets. http://www.vision.ee.ethz.ch/en/datasets/. Last accessed: August 1, 2019.

[196] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[197] Francesco Solera, Simone Calderara, and Rita Cucchiara. Socially constrained structural learning for groups detection in crowd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(5):995–1008, 2016.

[198] UCY Computer Graphics Lab. Crowd data. https://graphics.cs.ucy.ac.cy/research/downloads/crowd-data/. Last accessed: August 1, 2019.

[199] Stefania Bandini, Andrea Gorrini, and Giuseppe Vizzari. Towards an integrated approach to crowd analysis and crowd synthesis: A case study and first results. *Pattern Recognition Letters*, 44:16–29, 2014.

[200] Jun Zhang, Wolfram Klingsch, Andreas Schadschneider, and Armin Seyfried. Ordering in bidirectional pedestrian flows and its influence on the fundamental diagram. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(02):P02002, 2012.

[201] Forschungszentrum Jülich Institute for Advanced Simulation (IAS). corridor4 [pedestrian dyanmics data archive]. http://ped.fz-juelich.de/da/2009bidirSym/. Last accessed: August 1, 2019.

[202] Maurice Marx. Python 2d/3d trajectory visualization library. https://github.com/marximus/trackviz/. Last accessed: August 1, 2019.

[203] Lionov Wiratma. Visualization of grouping definitions. https://tiny.cc/groupingvideos/. Last accessed: August 1, 2019.

[204] Edward Hall. *The Hidden Dimension*. Anchor Books, 1992.

[205] Wouter van Toll, Norman Jaklin, and Roland Geraerts. Towards believable crowds: A generic multi-level framework for agent navigation, 2015. ICT.OPEN 2015.

[206] Ioannis Karamouzas, Roland Geraerts, and Mark Overmars. Indicative routes for path planning and crowd simulation. In *Proc. of the 4th International Conference on Foundations of Digital Games, FDG 2009*, pages 113–120, 2009.

[207] Christian Keip and Kevin Ries. Dokumentation von versuchen zur personenstromdynamik, projekt "hermes". http://ped.fz-juelich.de/experiments/2009.05.12_Duesseldorf_Messe_Hermes/docu/VersuchsdokumentationHERMES.pdf, 2009.

[208] MICC Media Integration and University of Firenze Communication Center. Museumvisitors dataset for pedestrian and group detection. https://www.micc.unifi.it/resources/datasets/museumvisitors/. Last accessed: August 1, 2019.

[209] Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals. On the complexity of minimum-link path problems. *JoCG*, 8(2):80–108, 2017.

[210] Gennady Andrienko, Natalia Andrienko, and Marco Heurich. An event-based conceptual model for context-aware movement analysis. *International Journal of Geographical Information Science*, 25(9):1347–1370, 2011.

[211] Corrado Loglisci. Using interactions and dynamics for mining groups of moving objects from trajectory data. *International Journal of Geographical Information Science*, 32(7):1436–1468, 2018.

SAMENVATTING

Geavanceerde tracking technologieën zoals Global Positions Systems (GPS) maken het mogelijk om eenvoudig locaties op te slaan en de verplaatsingen van mensen, dieren of andere bewegende objecten te volgen. Het toenemende gebruik van dit soort technologieën heeft geleid tot grote hoeveelheden verplaatsingsdata. Als gevolg hiervan is er ook meer vraag naar efficiënte methoden om ze te bewerken en analyseren, want de analyse van verplaatsingen is essentieel in verschillende onderzoeksgebieden. Voorbeelden zijn ecologie en transport.

Gewoonlijk wordt verplaatsingsdata beschreven door een traject: een pad gemaakt door een bewegende entiteit wanneer het zich verplaatst door de ruimte in een tijdsbestek. Daarom kan een traject gezien worden als een functie die een tijdstip afbeeldt op de positie op dat moment. Tracking technologieën zijn echter alleen in staat om de positie op specifieke momenten te bepalen, met regelmatige of onregelmatige tijdsintervallen ertussen. In dit model is een traject een reeks van posities voorzien van het tijdstip waarop die positie werd waargenomen. We gebruiken lineaire interpolatie tussen de waargenomen posities, en nu is een traject een continue functie die lineair is op tijdsintervallen. De waargenomen posities zijn de vertices van het traject.

Er zijn vele verschillende analysetaken die we kunnen uitvoeren op trajectdata. Een van de meest belangrijke is het identificeren van diverse patronen die tevoorschijn komen als de entiteiten zich verplaatsen. Deze verplaatsingspatronen zijn essentieel want ze stellen het gedrag voor van een entiteit, of ze stellen de relaties voor tussen meerdere zich verplaatsende entiteiten. In dit proefschrift concentreren we ons vooral op een specifiek patroon: gezamenlijke verplaatsingspatronen, ofwel groepsverplaatsingen. Dit patroon treedt op als meerdere entiteiten zich gezamenlijk verplaatsen gedurende een voldoend lange tijdsduur.

We beginnen onze studie naar groepsverplaatsing met de analyse van diverse maten voor een groep trajecten. Als eerste beschouwen we maten voor een enkel traject. Deze maten geven een enkele waarde voor een heel traject, zoals gemiddelde snelheid of globale richting. Er zijn drie klassen voor deze maten: voor een traject op zich, voor een traject tussen andere trajecten, en voor een traject in een omgeving of context. We breiden deze maten voor een traject uit naar maten voor een groep trajecten en klassificeren ze op dezelfde wijze. Daarnaast voegen we nieuwe maten toe die niet bestaan voor een enkel traject, zoals de dichtheid van een groep. We laten vervolgens zien dat diverse taken gerelateerd aan trajectanalyse, zoals de visualisatie van grote hoeveelheden trajecten, deze maten kunnen gebruiken om betere resultaten te verkrijgen.

Vervolgens beschrijven we een nieuwe definitie om groepsverplaatsingen te modelleren. Deze definitie is een verfijning van een eerdere definitie. We maken daarbij een kleine wijziging in wat het betekent om gezamenlijk te zijn, de ruimtelijke parameter. Met voorbeelden laten we zien dat in omgevingen met veel bewegende entiteiten, de verfijnde definitie beter overeenkomt met de menselijke intuïtie. We formaliseren dit model en geven efficiënte algoritmen om de groepen in een dataset met trajecten te berekenen. De efficiëntie van deze algoritmen hangt af van het aantal trajecten en het aantal vertices per traject. Het algoritme kan ook gebruikt worden als de vertices van de trajecten niet met overeenkomende tijdstippen zijn opgeslagen.

Tenslotte vergelijken we vier definities van groepsverplaatsingen op een experimentele wijze, waaronder onze nieuwe definitie. Daartoe hebben we de algoritmen geïmplementeerd die groepen volgens deze definities berekenen. In de experimenten gebruiken we diverse soorten trajectdata: kunstmatige data van computersimulaties en echte trajecten van voetgangers, zowel in laboratorium- als in natuurlijke omgevingen. We beschouwen de verschillen tussen elke definitie en door de mens geannoteerde data kwantitatief. We bestuderen ook de afhankelijkheid van de vier definities van de dichtheid (van de entiteiten) en van de frequentie waarmee de posities zijn opgeslagen. Met een nieuw visualisatiesysteem bieden we een manier om de verschillende definities kwalitatief te vergelijken door middel van kleurcoderingen in videomateriaal.

Naast de studie van groepsverplaatsingen behandelt dit proefschrift ook het polygonale lijnsimplificatieprobleem. Een polygonale lijn is het beeld van een continue functie die lineair is op tijdsintervallen. We beschouwen het probleem om voor een invoer polygonale lijn een uitvoer polygonale lijn te berekenen die goed op de invoer lijkt en zo weinig mogelijk vertices bevat. Deze uitvoer dient een voorgeschreven maximale afstand tot de invoer hebben. We gebruiken twee afstandsmaten: de Hausdorff afstand en de Fréchet afstand.

De Hausdorff afstand tussen twee verzamelingen punten is het maximum van alle afstanden van een punt in de ene verzameling tot het meest dichtbije punt in de andere verzameling. Om de Fréchet afstand te begrijpen beschouwen we een persoon en haar hond die elk een eigen route lopen. Beiden kunnen hun snelheid willekeurig aanpassen, maar ze kunnen alleen vooruit op hun route. De kleinste lengte van een hondenriem die nodig is om de routes te belopen is de Fréchet afstand.

Met deze twee afstandsmaten vergelijken we eerst twee bekende lijnsimplificatiealgoritmen, te weten het Douglas-Peucker algoritme en het Imai-Iri algoritme, met de optimale simplificatie. Daarna beschouwen we het berekeningsprobleem om de optimale simplificatie te berekenen met de Hausdorff afstand en de Fréchet afstand.

ACKNOWLEDGEMENTS

This thesis marks the end of one of the most important chapters of my life: my time as a PhD student. Over the course of my PhD, many people have directly or indirectly made this thesis possible and have supported me in various ways. I know that words are not enough to express all of my gratitude, but this is the part of my thesis where I can convey my thanks to those who have helped me get where I am today.

First of all, I would like to express my deepest and sincere gratitude to my supervisors, Marc van Kreveld and Maarten Löffler. I have always considered myself extremely lucky to have them as my supervisors. Their vision, knowledge, and innovative ideas during our discussions will always impress me.

Marc, thank you for being a very friendly and thoughtful supervisor during my PhD, as well as during my master's study years ago. I will always be grateful that you have provided me with so much advice regarding scientific matters. I am also thankful that you trusted me and gave me such a great opportunity to conduct my PhD research with you. I hope I do not disappoint you in the end. Maarten, besides your guidance and supervision, I also really appreciate your thinking, your explanations, and your feedback. Most of the time, I spent a rather long time following your thoughts and ideas, mainly because of my limited (geometric) intuition, but eventually, you enabled me to gain new knowledge and new perspectives on how to solve problems. Above all, I want to thank both of you for being incredibly helpful, supportive, and very patient with me. I have such a great time doing research with you, and most importantly, I learned a great deal from both of you along the way. Dank je wel voor alles!

Next to Marc and Maarten, I would like to extend my gratitude to Frank Staals. Frank, I cannot thank you enough for all your helpful suggestions and direction while doing research with me and being super-friendly with answering all my questions. Without your invaluable contributions, this thesis would not have become what it is today.

The experimentation part presented in this thesis was made possible with help from Roland Geraerts, Angelos Kremyzas, Arne Hillebrand and Wouter van Toll. Roland allowed me to use the UU Crowd Simulation Framework, while Angelos, Arne, and Wouter helped me to understand how to use it. Arne also pointed out the pedestrian data sets that I finally used in the experiments. Many thanks to all of you!

For the members of the reading committee: Maike Buchin, Mark de Berg, Jantien Stoter, Hans Bodlaender, and Remco Veltkamp – thank you for your time and effort to carefully read and evaluate my thesis. Moreover, almost all work in this thesis has been published in conference proceedings or journals. Therefore, I would also like to thank all anonymous referees for their comments, corrections, and advice that shaped my thesis into what it is now.

Furthermore, I like to thank everyone in the Geometric Computing group (and Virtual Worlds division) for a great working atmosphere. In particular, for all fellow participants of WASWEG, thank you all for sharing your knowledge. I really learned a lot from every discussion in our weekly sessions. For the last four years, I have been sharing my office with Ron Vanderfeesten, and Christyowidiasmoro, who joined us later. Thank you for being so easygoing and for making our office such a pleasant and enjoyable place to do research.

Outside my academic life, I am deeply grateful that I have friends who will always be there on (almost) a daily basis: Ivonne Martin, Husnul Hakim, and Mariskha Tri Adithia. Thank you for your companionship (from a distance) and support in many ways throughout these years. Also, for my friends in the Netherlands: Ameling Keymans, Don Duell & Elvira Duell, and Matthias Helder & Karen Nieuwenhof. Thank you for your friendship and your moral support from the first time I came to the Netherlands in 2008. My special thanks also go to my long-time friend, Kim Siung, who introduced me to the Voronoi diagram (and the world of computational geometry) in the first place.

Although my research keeps me busy and at a distance, I owe immense thanks to my family. For my cousins and their partners in the Netherlands, thank you for always supporting me and making me feel at home every time we meet. Of course, I wish to express my loving gratitude for my family in Indonesia: my mother, my father, my sister, and my parents-in-law, for their relentless support, care, and love throughout these years. I could not have made it this far without you all.

Finally, for the most important acknowledgement: my love, Febi. You always encourage me to go abroad for my graduate studies and you let me go all the way to Utrecht (twice!), even when you know that our life together would not be the same during those times. I owe you at least six years that we were supposed to spend together, but now I have a whole life ahead of us to pay this debt. Obviously, I could write endlessly on how much you mean to me, but I know that I only need these five words: *thank you for loving me*.

CURRICULUM VITAE

Lionov Wiratma was born on 30 November 1977 in Bandung, Indonesia. From 1993 to 1997, he went to BPK Penabur Christian Senior High School in Bandung, where he obtained his secondary education degree.

He then continued studying computer science at Parahyangan Catholic University in Bandung and received his Bachelor's degree in 2002 on the topic of visualization of the Fortune's algorithm for computing the Voronoi diagram of point sites.

In 2008, he started as a Master student in the Game and Media Technology master program at Utrecht University in the Netherlands, with a full scholarship from the Ministry of Communications and Information Technology of the Republic of Indonesia. His Master thesis, "Following the Majority: a New Algorithm for Computing a Median Trajectory", was supervised by prof. dr. Marc van Kreveld. He received his Master of Science degree in August 2010.

As of October 2015, he started as a PhD student in the Virtual Worlds Group, and later the Geometric Computing group, at the Department of Information and Computing Sciences of Utrecht University. The Ministry of Research, Technology and Higher Education of the Republic of Indonesia awarded his PhD fellowship. He completed his PhD thesis in 2019 under the supervision of prof. dr. Marc van Kreveld and dr. Maarten Löffler.