NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Bjørn Anders Flågen

Utilising locality sensitive hashing for efficient trajectory similarity computation

Master's thesis in Computer Science
Supervisor: Svein Erik Bratsberg
June 2023

Bjørn Anders Flågen

Utilising locality sensitive hashing for efficient
trajectory similarity computation

Master's thesis in Computer Science
Supervisor: Svein Erik Bratsberg
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Utilising locality sensitive hashing for efficient trajectory similarity computation

Bjørn Anders Flågen

June 9, 2023

Abstract

The growing amount of GPS-compatible electronic devices capable of recording and storing users' position and movement data, has led to an increased interest in the analysis of this type of data. Fundamental in the analysis of observational data is the need for quantification of similarity between the observations. Several similarity measures exist for this purpose, but their time complexity limit the analysis of larger amounts of data.

In this thesis, we examine whether *locality sensitive hashing functions* can be utilised as an alternative to traditional similarity measures, with the aim of improving computation time when computing similarities between *GPS-trajectories*. As a starting point, we adapt and implement two techniques originally developed for top-k queries. We examine aspects such as accuracy and efficiency, in addition to evaluating how well the generated similarities performs in terms of *clustering*.

Sammendrag

Den stadig voksende mengden av GPS-kompatible elektroniske enheter i stand til å registrere og lagre brukernes posisjons- og bevegelsesdata, har medført en økende interesse for analyse av denne type data. Fundamentalt i analyse av observasjondata er behovet for kvantifisering av similaritet mellom observasjonene. Det eksisterer flere similaritetsmål for dette formålet, men deres tidskompleksitet legger begrensninger for analyse av større mengder data.

I denne avhandlingen undersøker vi hvorvidt *lokalitets-sensitive hashefunksjoner* kan benyttes som alternativ til tradisjonelle similaritetsmål, med formål å forbedre tidsforbruket ved similaritetsberegning av *GPS-spor*. Vi tar utgangspunkt i to teknikker opprinnelig designet for topp-k spørringer, adapterer og implementerer disse, og analyserer aspekter som nøyaktighet og effektivitet, samt evaluerer de genererte similaritetenes ytelse under *clustering*.

Preface

This thesis was written as a final deliverable in the course *TDT4900 - Computer Science, Master thesis* at the Norwegian University of Science and Technology in the spring of 2023.

I want to thank my supervisor, Professor Svein Erik Bratsberg, for his guidance, commitment and feedback, which has truly been most helpful during my research.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

The last decade's technological advancements have led to an enormous increase in the number of devices and gadgets with location-aware tracking, such as phones, smartwatches and navigation equipment to name some. The combined magnitude of these devices have further led to a huge acquisition of large volumes of movement data, including recordings of objects' locations over time, so-called *trajectories* [1]. With the rising availability of trajectory data, the need to analyse and gather valuable and meaningful insights of objects' movements also rises.

A common way of analysing trajectory data is to compute how similar trajectories are, i.e. their similarity [2]. This computation lays the foundation for many different applications, and is often the first step for more complicated analysis tasks. For instance, applications of trajectory analysis include the prediction of hurricanes' movements for disaster alert [3], finding where cyclists' routes are similar to plan for the construction of bicycle paths in cities, military surveillance and security or sports analysis to name some [4].

Even though trajectories have a rigid data-format, similar trajectories can be very different, which complicates the process of computing their similarity. As a result of this, algorithms designed for this purpose are resource-intensive, and the computation cost increases drastically with the number of trajectories to be analysed, making it a time-consuming process. In fact, most similarity distance algorithms today have quadratic computation time, which have become a bottleneck for analysis of the ever-growing volumes of trajectories [5]. With the rising amount of available trajectory data combined with the shortcomings of classic similarity algorithms in terms of efficiency, there is a need for techniques that can compute similarities for larger volumes of trajectories more efficiently.

A fundamental aspect of computer science is the art of approximation to reduce computation time for resource-intensive operations. A classic and well-known technique developed for the purpose of efficiently retrieving similar items in large and complex data-sets is called locality sensitive hashing (LSH)[6]. With this research we want to explore whether locality sensitive hashing can be applied for more efficient computation of trajectories' similarities, to accommodate the growing amount of data. The overall aim of this thesis is to research whether LSH can be utilised for similarity computation to cope with the problem of scalability, and still function as a proper alternative to traditional similarity measures in terms of accuracy.

1.2 Research Questions

In this thesis, we wish to explore the potential of LSH to cope with the problem of scalability when computing similarities for larger volumes of trajectories. More specific, we want to explore whether two LSH-schemes designed for top-k search among trajectories can be adapted to compute similarities. We want to understand how the LSH-generated similarities perform in terms of accuracy and computation time, and whether they can be used for practical data analysis tasks like clustering. An important factor in our research is to understand whether LSH-generated similarities can be used as an alternative to traditional similarity measures. From these objectives, we have defined the following research questions that we seek to explore and answer in this thesis:

RQ1 Are the LSH-schemes provided by Driemel and Silvestri [7] and Astefanoaei *et al.* [8] adaptable to similarity computation?

RQ2 How well do the approximated similarities based on these schemes perform compared to traditional similarity measures for various data sizes?

RQ3 How suited are the LSH-generated similarities for creating clusters?

1.3 Outline

This thesis is a continuation of the preliminary work [9] that was conducted in the course *TDT4501 - Computer Science, Specialization Project*. We do not assume that the reader of this thesis is familiar with the content of the report from TDT4501, and its content is therefore restated in the early chapters of this thesis. This thesis is structured the following way:

Chapter 2 is largely based on the above-mentioned report and introduces the theoretical background of trajectories and the art of traditional similarity computation. We also present the theory behind locality sensitive hashing and clustering analysis.

In **chapter 3**, which is also largely based on [9], we present the existing LSH work that our research is based upon. We also describe and discuss other existing approaches to solving the problem of scalability when computing similarities for larger volumes of trajectories.

Chapter 4 presents the data-sets, methodology, implementation details, evaluation and analysis methods of the LSH-schemes. The chapter also describes in detail how we proceeded to answer the defined research questions.

In **chapter 5**, the results and findings from the research are presented. First, we present general findings from a variety of configurations of the LSH-schemes, before more specific results for a given configuration are presented.

Chapter 6 discusses the presented results. We consider different aspects of the LSH-schemes and discuss the results against theoretical expectations and practical significance.

Finally, in **Chapter 7**, the conclusions based on the discussed aspects are summarised. The conclusions are presented and linked up to the research questions of this thesis.

For the written code and further implementation details of the conducted experiments in this research, we refer to our code-repository, which can be found in https://github.com/bjorafla/master.

Chapter 2

Theory

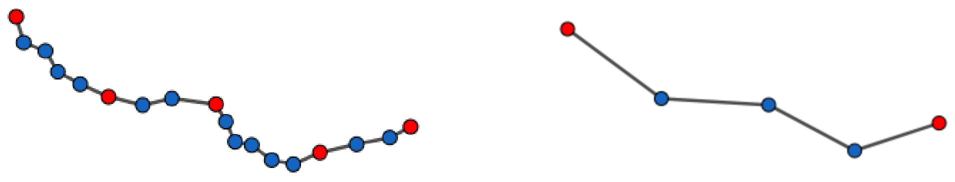
This chapter will largely present some preliminary theory and background information related to trajectories and the art of computing their similarity using distance measures. The locality sensitive hashing technique will also be presented. In addition, this chapter will also elaborate on data clustering and cluster evaluation measures.

2.1 Trajectory

Collection of movement data and the analysis of such data is a major application in a variety of fields and domains, like sports, finance, medicine, city planning and navigation to name some [10]. The movement of an object is often represented as a trajectory, which can be viewed as a series of recorded datapoints ordered by time, where each datapoint is a position or a value. Phrased differently, trajectories are traces containing data of a moving objects' positions, and in simple turns a trajectory can be seen as an ordered list of position-aware datapoints. This could be the share price value of a stock, traces of shipments, or the geographical movement of taxis in a city environment.

For the purpose of this thesis we will address trajectories that contain points of spatial location, so-called *spatial trajectories*. A precise and formal definition of a spatial trajectory was given by Zheng [11]: "A *spatial trajectory* is a trace generated by a moving object in geographical spaces, usually represented by a series of chronologically ordered points, for example, $p_1 \longrightarrow p_2 \longrightarrow \dots \longrightarrow p_n$, where each point consist of a geospatial coordinate set and a time stamp such as $p = x, y, t$". Throughout this thesis, we will use a similar definition, where the timestamp is omited and the datapoints are ordered chronologically by their sequence in their trajectory as shown in Definition 1:

Definition 1 (Trajectory) *A trajectory $T = [[x_0, y_0], [x_k, y_k], \dots, [x_n, y_n]]$ where x_k, $y_k \in \mathbb{R}$, and k is a strictly increasing sequence number $\in [1, \dots, n]$ where n is the length of the trajectory.*



(a) Higher sampling **(b)** Lower sampling

Figure 2.1: Two different trajectory representations of the same movement from one location to another. The beginning and endpoint of both trajectories are marked in red, as well as the the points in **(a)** that are likely to correspond with the inner vertices of the trajectory in **(b)**.

Despite the rigid definition of a trajectory, similar trajectories may be significantly different when comparing them point-wise. The sampling rate of datapoints may vary, as well as the collection method. For instance, datapoints may be registered every second, favouring data quality, or they could be registered only when the moving object makes a significant turn to save storage space on the recording device. Naturally, the quality of the equipment used for collection may also differ, as well as the local quality of GPS-signals. These are just a selection of factors influencing the recorded trajectory representation of an object's movement. This implies that similar movements can be represented by relatively different, yet similar trajectories. This effect is visualised in Figure 2.1.

2.1.1 Notations

Throughout this thesis we will use some notations describing trajectories and their properties. We have provided the notations in Table 2.1 together with a description explaining the notation.

Table 2.1: Notations used throughout this thesis.

Notation	Description
T	The set of trajectories.
T_i	The i-th trajectory of set T.
p_j^i	Point j of trajectory T_i. Consisting of coordinates x, y and a timestamp t.
$dis(X, Y)$	A distance (similarity) function between $X$and $Y \in T$.

2.2 Similarity of trajectories

When analysing trajectory data, there is often a need to understand how equal or similar pairs of trajectories are to each other, i.e. a need of computing their so-called similarity. In fact, the computation of these similarity-values (similarities) lays the foundation for many important trajectory mining tasks. The similarities are computed using what we call similarity measures, which is a function that takes two trajectories T_1 and T_2 as input, and quantifies their similarity to a non-negative numerical value. Due to the diverse nature of trajectories as described in section 2.1, similar trajectories may have significant variations, which must be taken into account and get captured by the similarity measures.

When computing the similarity between two trajectories, we are interested in the relationship between them, and it is therefore important that the distance measures is *symmetric*. We will use the definition of a distance function for trajectories given by Besse *et al.* [12] in our thesis.

Definition 2 (Distance function) *Let T be a set of trajectories. A distance function* $dis(X,Y) \longrightarrow \mathbb{R}$ *is called a* dissimilarity *on T if all* $T_i, T_j \in T \mid T_i \neq T_j$*:*

1. $dis(T_i, T_j) \geq 0$
2. $dis(T_i, T_j) = dis(T_j, T_i)$
3. $dis(T_i, T_i) = 0$

When condition 3. is satisfied for T_i *and* T_j*, it implies that* $T_i = T_j$*. We say that* $dis(X,Y)$ *is symmetric if all of the above conditions are satisfied. If the following condition is fulfilled, the distance function is considered to be a metric:*

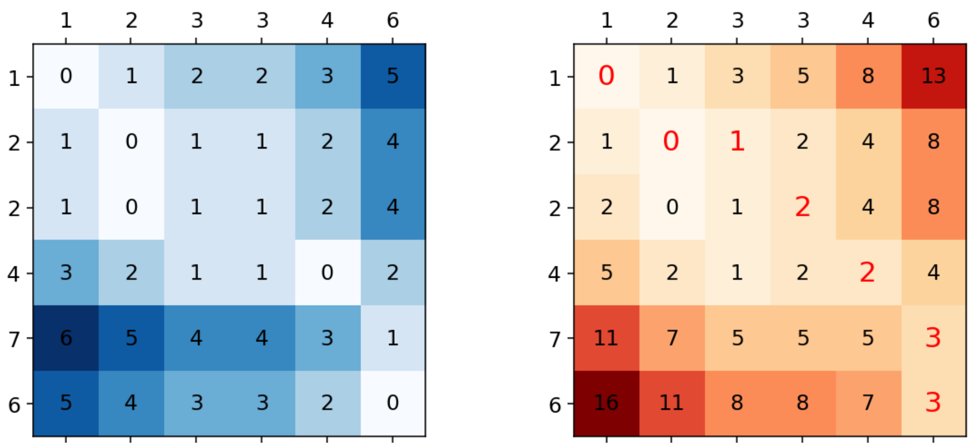
4. $dis(T_i, T_k) \leq dis(T_i, T_j) + dis(T_j, T_k)$

As stated in Definition 2, a distance function is formally a dissimilarity measure, meaning that the greater the quantified value produced by the distance function, the less similar the input trajectories are. We still use the term *similarity measure*, but one should be aware of this formality. There are different aspects that can be emphasised when computing similarities among trajectories, such as speed, directions or spatio-temporal to name some. For this thesis we will focus on the **spatial similarity** of trajectories, i.e how equal trajectories are in shape and location.

There have been proposed many different distance measures in the literature which have been thoroughly described and discussed in several studies [12–15]. Some of the most popular distance measures for computing spatial similarity of trajectories from these studies are *Dynamic Time Warping*, *Frèchet Distance* and *Edit distance*. These measures have shown to be both robust and accurate for trajectory similarity computation. However, as we shall see next, computing similarities between trajectories using these measures is a resource-intensive task with poor time-complexity.

2.2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is one of the most common discrete symmetric distance measures for trajectories, and as the name might imply it is designed to align items by warping the time dimension, meaning that trajectories are getting "stretched" and "shrunk" to find an optimal alignment. DTW computes the sim-



(a) Cost matrix **(b)** Accumulated cost matrix

Figure 2.2: Dynamic Time Warping. The computed cost matrix **(a)** and accumulated cost matrix **(b)** for two vectors. The red-coloured path in **(b)** is the warped path and the accumulated cost is 3. A darker background colour indicates a larger distance in **(a)** and a more expensive path in **(b)**. The cost function is the absolute value of their natural distance. Start-point is the upper left corner and the end-point is the lower right corner.

ilarity of trajectories, T_1 and $T_2 \in T$, by creating a $m \times n$-matrix, C, where m and n is the length of T_1 and T_2, and each point in T_1 corresponds to a row in C and each point in T_2 corresponds to a column. For each cell $c_{i,j} \in C$, the pairwise cost between p_i^1 and p_j^2 is computed and inserted [16]. The cost function may vary based on implementation, but a L^p-norm such as euclidean distance is often used. Next, the lowest accumulated cost to each cell is computed and the values in C are updated so that it now represents the accumulated costs. By traversing C from the end-point, $c_{m,n}$, to the start-point, $c_{0,0}$, an optimal warped path between T_1 and T_2 can be found by always choosing the previous point in C such that $c_{prev} = min(c_{i-1,j-1}, c_{i-1,j}, c_{i,j-1})$. The path generated by repeating this process is the optimal warped path between the trajectories, and the corresponding accumulated cost of this path represents their similarity. Figure 2.2 illustrates the generation of the accumulated cost matrix and the optimal path between two one-dimensional vectors.

A great advantage of DTW is that it can align trajectories with different speed and sampling. Even though DTW is widely common to use, it is not robust to noise as that may affect the cost matrix and thus also the optimal path. In addition, and more important for this thesis, DTW computes the pairwise cost between every pair of points in the trajectories, resulting in a run-time of $O(n^2)$, where n is the average length of the trajectories. This makes the algorithm scale poorly, and less suited for efficient computation of similarities in large data-sets.

2.2.2 Fréchet Distance

Another common and well-documented distance measure for trajectories is the Fréchet distance (FD), which also takes into account both the ordering of the points and their location. This similarity measure is a shape-based measure that aims to find the geometrical similarity between trajectories, and is often referred to as the "walking-dog distance". The reason for this is that the Fréchet distance can be seen as the minimum distance a dog leash can be when walking a dog, where the dog and owner have slightly different paths. Unlike DTW, the Fréchet distance is a continuous measure as it aligns interpolated representations of the trajectories, which enables it to align non-uniform sampled trajectories [15]. However, this complicates the algorithm and adds another factor to the time-complexity compared to DTW, resulting in a run-time of $O(n^2 log(n^2))$ where n is the average length of the two input trajectories.

The Fréchet distance can be computed by calculating a so-called free-space diagram between the two trajectories [17]. A free-space diagram can be seen as a spatial area between two trajectories where the area hold all pairs of points between the two trajectories that are within a given distance, ϵ, from each other. The free-space diagram between two trajectories, f and g, can be visualised by aligning the trajectories along each axis in a plane as shown in Figure 2.3, and mark the area where f and g are within ϵ of each other. If there exists a path from the lower left corner to the top right corner that is monotone in both dimensions, we have a valid free-space path. The Fréchet distance is the smallest possible ϵ that has a valid free-space path.

As stated, computing the Fréchet distance is a costly process with a run-time of $O(n^2 log(n^2)$, where n denotes the average length of the input trajectories. The construction of the free-space diagram and the search of a valid path have a combined run-time of $O(n^2)$ alone, and the algorithm must also find the optimal path. Furthermore, due to the high computation cost, computing the Fréchet distance quickly gets expensive for analysis of larger volumes of trajectory data. The advantage of using Fréchet distance is that it fulfils the fourth condition from Definition 2, meaning that the triangle inequality is preserved, which could be preferable during further analysis.

A simpler version of the Fréchet distance is the discrete Fréchet distance (DFD), which is a discrete and symmetric version of Fréchet distance. It was introduced by Eiter and Mannila [19] in 1994, and is fairly similar to DTW. Like DTW, DFD creates a pairwise cost-matrix, C, between every point in the two trajectories, using a domain-appropriate cost-measure which could be the euclidean distance for two-dimensional spaces. DFD finds the optimal lowest cost path by traversing the cost matrix from the start-point $c_{0,0}$ to the end-point $c_{n,m}$. When traversing, the path is constructed by adding the next point such that $c_{next} = min(c_{i+1,j+1}, c_{i+1,j}, c_{i,j+1})$, until the end-point is reached. The maximum value of the costs in the computed optimal path is the discrete Fréchet distance between the two input trajectories.

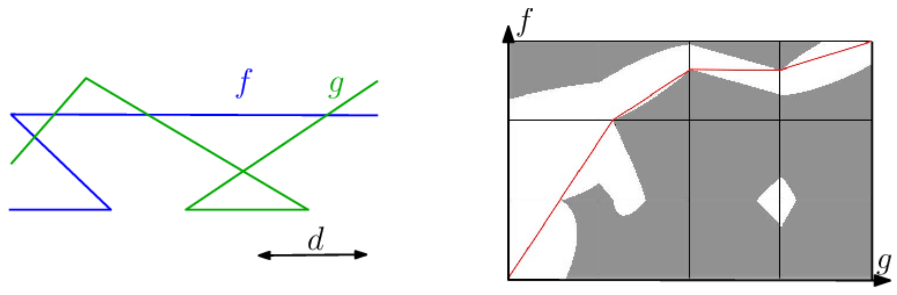


Figure 2.3: Visualisation of a free-space diagram between trajectories f and g. The red line represents a valid free-space path from bottom left corner to top right corner. Figure reprinted from [18].

Like DTW, the creation of a full $n \times m$-matrix is necessary for the optimal path to be found, thus resulting in a run-time of $O(n^2)$ where n is the average length of the input trajectories. The presented run-time is for one comparison between a set of two trajectories, which result in poor performance when using DFD for large data-sets.

2.2.3 Edit distance

Another well-known similarity measure is the edit distance (EDR). It was originally created for comparing the similarity of strings by doing so-called *edits*, and is a common algorithm present in the literature [13]. EDR distance is the number of edits needed to change one object into another, where an edit could be either insertion, deletion or substitution of a point [10]. To exemplify, the edit distance between the strings $s_1 = "abc"$ and $s_2 = "ac"$ are 1, since only the deletion of $"b"$ in s_2 is necessary to change s_1 into s_2.

To compute the symmetric EDR distance between trajectories, a threshold value, ϵ, and a sub-cost function is needed to evaluate whether two points are considered equal. Two points are considered equal if they are within ϵ from each other. EDR differs from other similarity measures by the fact that it counts edits, thus making a binary decision whether to include points or not. This approach makes the algorithm robust to noise, but as with both DTW and DFD the time-complexity of EDR is $O(n^2)$, where n is the average length of the compared objects.

2.3 Locality-sensitive hashing

Understanding data is the driving force behind the field of data-mining, and the problem of finding similar items in data-sets is a fundamental aspect of this field [20]. A naive approach to solve this particular problem is to compute and compare the similarity of every possible pair of items in the data-set using a suitable similarity measure. For large data-sets, this approach is simply not feasible, as the computation cost of generating the similarities would be too expensive and time consuming. Instead, we can utilise a technique called locality sensitive hashing (LSH), which allows us to find and compare items that are likely to be similar, without comparing the entire data-set.

Locality sensitive hashing was first introduced in 1999, and was designed as an approximation technique developed for efficiently finding similar items in large data volumes in sub-linear time [6]. As the name might suggests, the technique is based on hashing the data-items and then comparing their hashes. The basic idea behind LSH is that items that are similar or close to each other are likely to have similar hashes. This is an essential property of LSH, and is achieved by carefully choosing hashing functions that preserve the locality and characteristics of the data. It is this property that have made the technique one of the most popular approaches to solve problems related to finding similar items or so called near-neighbour problems [21]. However, the preservation of data-locality through hashing also makes it possible to use the hashes to approximate similarities among the data-items since the hashes can be seen as rough compressions of the original data-entities. It is this idea that lays the foundation for this research and thesis.

The traditional locality sensitive hashing technique used to find similar items, takes as input a matrix M, where each column, c, represents an item in the data-set. The matrix are then divided into b bands, where each band consists of r rows. Next, each band hashes its columns, which represent a piece of the original data-item, to one of k buckets depending on the hash value. Items that are hashed to the same bucket is considered to be a candidate pair, i.e similar items.

In contrast to normal hash function applications, locality sensitive hashing functions does not seek to transform the input data into random output values. Instead, since the preservation of data locality is a most crucial property, the choice of hash functions deeply affects the quality of the hashes and the technique's ability to achieve the desired result. Despite choosing a reasonable hash functions, there is no guarantee that two similar items' hashes are likely to be similar, because there is a significant information loss during hashing. However, the hashing process can be repeated multiple times with different hashing functions to increase the probability of hashes being hashed to the same bucket.

There are usually several hash-functions involved when performing LSH, and together they create a hash-family. As stated in Definition 3, the hash-family must preserve the locality of the original data in order to be suitable for the task. The definition formalises the already stated fact that two similar data-points in the original data-set should also be similar in the projected result set after hashing.

Definition 3 (Locality-sensitive hash family) *A hash family H, is said to be (R, cR, p_1, p_2)-sensitive, if all hash functions $h \in H$ satisfies:*

1. *if $dis(x, y) \leq R, \rightarrow Pr[h(x) = h(y)] \geq p_1$*
2. *if $dis(x, y) > cR, \rightarrow Pr[h(x) = h(y)] \leq p_2$*

Where x, y are datapoints in a d-dimensional data-set D in $\mathbb{R}^d$. p_1 and p_2 are *probabilities where $p_1 > p_2$, c is an approximation ratio > 1, r is a distance > 0 and dis is a distance function for similarity.*

LSH consist of a broad spectre of algorithms and different methods, having in common that the chosen hash family preserves the locality of the data. As we shall see in the coming parts of this thesis, there are different ways to build LSH-schemes of trajectories by abstracting the concepts of bands and buckets, which enables us to efficiently do more complex approximations on the data, including approximating similarities of data.

2.4 Clustering techniques

Clustering is a technique in data mining for grouping similar objects in a data-set into so-called clusters (groups). It allows us to characterise object into groups so that patterns in the data can become more evident. Since clustering is a well-known technique for data-mining, we will utilise *Hierarchical Cluster Analysis* (HCA) to evaluate the performance of the approximated similarities from the LSH schemes.

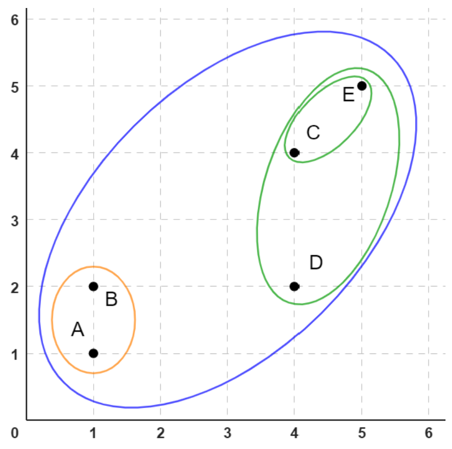
2.4.1 Hierarchical Cluster Analysis

Hierarchical Cluster Analysis (HCA) is a family of clustering algorithms which are designed to create clusters based on a hierarchical approach. The HCA family consists of *divisive* and *agglomerative* algorithms, where the former has a top-down approach and the latter having a bottom-up approach [22]. For this thesis, we will utilise the agglomerative linkage approach.

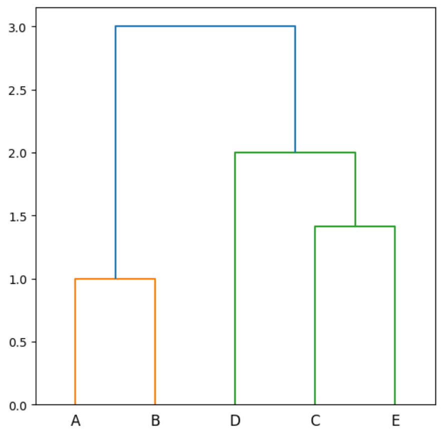
In agglomerative linkage HCA, every object start as a single cluster, and for each step of the algorithm two clusters are merged in a bottom-up approach. The selection of which clusters to be merged are decided by a linkage-method, which serves as a distance measure between clusters. Common linkage methods are *single*, *complete* and *ward*, where *single* denotes the minimum distance between any two points in two respective clusters, *complete* denotes the maximum distance between any two points in the two respective clusters, and *ward* denotes how much the total distance for all points to the clusters' centroids will change if merged. Independent of linkage-method, the two clusters with the shortest linkage will be merged at each step of the algorithm. If no end-parameter is given, HCA will merge clusters until there is only one cluster left [23]. The HCA process and especially the distances between clusters can be visualised using a dendogram, as shown in Figure 2.4.

2.4.2 Evaluating clusters

When creating clusters, and especially clusters based on approximations, we need a way to determine how well the clusters perform. A good clustering method creates clusters of objects that truly are similar and the created clusters should



(a) Nested clusters



(b) Dendogram of the clusters

Figure 2.4: Agglomerative single-linkage HCA of five points shown in **a**. By looking at the dendogram in **b** we see that the first cluster is created by merging A and B, meaning they are the most similar. The colour in the dendogram indicates the clusters.

be clearly distinct from another. There are several methods to evaluate clusters, both ground-truth based and methods based on internal performance. The first type of methods can be utilised to evaluate how well a clustering technique works when there is a known ground truth, thus being a good suit for evaluating how clusters based on approximations performs. The latter type, based on internal performance, usually evaluates clusters based on intra- and inter-cluster distance [24].

Introduced back in 1971, the rand index is a ground-truth based method that evaluates how well a cluster algorithm performs by looking at the content of the clusters [25]. Let $C = C_1, ..., C_r$ be the output of a clustering algorithm containing r clusters and $T = T_1, ..., T_q$ be the ground truth containing q clusters. The Rand index is a measure that is based on how many times two pairs of objects appear in the same cluster in C and in the same cluster in T, as well as the number of times two pairs of objects belong in different clusters in C and T. The rand index are the sum of these factors divided by the total number of object pairs in the data-set, where a score of 1 indicates a clustering output identical to the the ground truth. It is important to be aware that the rand index is sensitive to chance, meaning that an agreement between two clustering outputs can be the result of chance.

Despite being a good algorithm for comparing two clustering algorithms, the rand index does not give other information than a score presenting how close the two outputs are. A good result could potentially hide the fact that some clusters could be overlapping. As mentioned at the beginning of this section, there are measures that internally evaluate the output of a clustering algorithm by looking at intra- and inter-cluster distances, thus being suitable methods for evaluating cluster sharpness. The Davies-Bouldin index (DBI) is one of these measures [26], and it utilises two distances, *within-cluster-* and *between-cluster* distance to calculate its score. The *within-cluster* distance (WCD) is simply the average distance from each cluster's centroid to the corresponding objects, and the *between-cluster* distance (BCD) describes the distance between the centroids of two clusters. For every pair of clusters in the clustering a difference measure is calculated by dividing the sum of the clusters WCD's by their BCD. The Davies Bouldin index can then be computed by finding the average of the maximum difference measure for each cluster.

The main drawback with using this index is the need to compute centroids for every cluster. This is usually done by computing the average data-point of each cluster, which could be infeasible when the clusters contain complicated data-items. Furthermore, the method was designed to evaluate clustering methods, and in order to compare DBI scores from different clusters, the similarities used for clustering must be the same. However, the fact that the Davies-Bouldin index is a composition of both WCD and BCD means that we can use them individually when analysing the performance of a clustering output. The formal definition of within-cluster-distance, between-cluster-distance and the Davies Bouldin index can be found respectively in Definition 4, Definition 5 and Definition 6:

Definition 4 (Within-cluster distance) *The within-cluster-distance of a cluster C_k is defined as:*

$$WCD_{C_k} = \frac{1}{n_k}\sum_{a_i \in C_k} dis(a_i, c_k)$$

Where n_k is the number of objects in cluster k, c_k is the centroid of C_k and dis is a distance function.

Definition 5 (Between-cluster distance) *The between-cluster-distance of cluster C_k and C_l is defined as:*

$$BCD_{C_k C_l} = dis(c_k, c_l)$$

Where c_k and c_l are the centroids of C_k and C_l and dis is a distance function.

Definition 6 (Davies-Bouldin index) *The Davies Bouldin index of a clustering output C with K clusters is defined as:*

$$DBI_C = \frac{1}{K}\sum_{k=1}^{K} \max_{C_l \neq C_k} \frac{WCD_{C_k} + WCD_{C_l}}{BCD_{C_k C_l}}$$

Where c_k and c_l are the centroids of C_k and C_l and dis is a distance function.

We wrap up this section by looking at the definition of the Davies-Bouldin index and the description of rand index. For the Davies-Bouldin index we observe that a smaller output value indicates that the output clusters are compact and well separated. For the rand-index a higher score indicates a better result, meaning that a greater portion of the items can be found in the same cluster as the benchmark.

Chapter 3

Existing work

In this chapter, we cover some of the existing work where locality sensitive hashing have been proposed and used for efficient retrieval of similar trajectories. We first present the two articles that this research are built upon, before other approaches for efficient trajectory similarity computation are presented. Generally in the literature, to the best of our knowledge, the utilisation of LSH for efficient similarity computation of trajectories is a relatively unexplored field.

3.1 LSH of curves

In 2017, Driemel and Silvestri published a theoretical article [7] where they presented LSH schemes for the purpose of solving the (c, r)-near-neighbor problem of polygonal curves. Their work elaborated on how one can retrieve similar curves to a given query curve efficiently, using an approximation of the classic Fréchet distance as a similarity measure. They developed a data structure which allowed them to store curves and efficiently retrieve similar curves. They also presented some variations of the data structure to cope with different constraints, such as speed and anchored distances. Their work improved what was long considered as the *state of the art* from 2002 by Indyk [27] for solving the (c, r)-near-neighbor problems.

The basic LSH scheme presented in this paper is constructed by displaying curves over a randomly shifted grid and snapping the vertices of the curves to the corresponding underlying grid-cell. This creates a new representation of the curve, consisting of a series of grid cells where consecutive duplicate cells are removed. The process of snapping a curve to the underlying grid in this LSH scheme, defines a hash-function on the curves, where their locality are preserved through hashing. The grid is shifted in both dimensions by a random parameter $t \in [0, \delta)$, where δ denotes the resolution of the grid, and thus, by generating more grids with a random t, they are able to construct a family of hash functions.

Driemel and Silvestri also presented another, fairly similar scheme. Instead of a shifting grid, the vertices of curves are distorted by a parameter, t, randomly and independently chosen from a sequence of variables, t_p, that are uniformly distributed on the interval $[-\delta/2, \delta/2]$. Trajectories vertices' are then snapped to a fixed grid, and consecutive duplicates of grid points are removed. The hash function for this approach is defined by the sequence t_p. A family of hash functions is constructed by varying this sequence, which will distort the points in the curves differently.

For both schemes, the computation of the curves' hash values are stored in hash tables, where each hash function corresponds to a unique hash table. A similarity search is done by applying the hash functions to the query curve, and compare the resulting hash to the already existing hashes in the tables for matches. Compared to the state of art, the LSH structure reduced both query time and the needed space to store the structure for the problem. As mentioned in the early parts of this section, the presented LSH-schemes were originally created for solving the (c, r)-near-neighbor problem, however, we believe that it can be adapted and utilised for approximating similarities as well. It is the basic scheme from this article that this research is built upon, and our adaption is further described in chapter 4.

3.1.1 FRESH

A year later, in 2018, the creators of the grid-based LSH scheme contributed to the creation of a framework, FRESH, that were constructed to approximate answers to the r-range search problem under the Fréchet distance [28]. The r-range search problem is the problem of returning all entries in a data-set, given a distance r and a query object q, that are within distance r from q.

To address this problem, the authors constructed a framework based on the described LSH schemes, and adapted it by applying multiply-shift hashing to increase the probability of collision for close trajectories, and decrease it for distant trajectories. For a query curve, the framework returns all curves that collides with the query curve in at least one hash function, and the number collisions between two trajectories is used as an approximation. This scheme are likely to return some false positives, and thus several verification heuristics are applied to eliminate curves that are further away than r.

The FRESH framework cannot be directly adapted to similarity computation, since the key concept of FRESH is to prune computation of distant pairs. However, the LSH-scheme that lays the foundation for FRESH is similar to the one that we will utilise, such that the article itself and its results are valuable to us. An important result from the paper was that that LSH is considered to be the most effective heuristic to prune distant trajectory pairs. Furthermore, it is also considered being the cheapest to compute.

3.2 LSH with randomly deployed disks

The same year, in 2018, Astefanoaei *et al.* [8] published a study where they developed a disk-based LSH scheme specifically designer for trajectories. Compared to Driemel and Silvestri, Astefanoaei et al. developed a slightly simpler ground scheme, by randomly deploying a number of disks in a plane which covers the span of the trajectories. Their scheme allowed them to compute a variety of tasks, including both (c-r)-near and c-approximate-nearest neighbor queries, as well as distance estimation between trajectories and classification in an efficient manner.

In the article, the authors presented two LSH hashing schemes designed to approximate DTW and the Fréchet distance. The schemes are constructed by deploying n random disks with radius r in several layers l that cover a plane that spans all the trajectories. The first structure, referred to as *binary sketches*, is constructed by snapping trajectories to their intersecting disks. The idea is that close trajectories should intersect common disks, and thus distant trajectories should not intersect common disks. A bit vector is created for each trajectory, representing the disks that intersected with the original trajectory at one layer. The hash function of this LSH scheme is the process of converting a trajectory to a bit vector, and thus a hash family consisting of i hash functions is created by repeating the bit-vector computation of trajectories for each layer l_i in l.

As described in 2.2, the Fréchet distance and DTW maintains the ordering of trajectories points (vertices), which is not supported by the binary sketches. To cope with this problem, the authors presented a second scheme, *ordered sketches*. This scheme is fairly similar, but converts the trajectories into the sequence of the disks it intersects, such that a trajectory is represented sequentially by the disks it moves through. As with the binary sketches, Astefanoaei et al. constructed a family of hashes for the ordered sketches by repeating the creation of ordered sketches for multiple layers, l. The authors described multiple operations that could be performed on the ordered sketches, including approximation of the similarities between the original trajectories. They proposed *edit distance* on the output of the ordered sketches to be a good measure for this purpose.

The article included an experimental part where the performance in terms of accuracy and approximation was found to be increasing with the number of disks and the number of layers. However, for every extra disk and layer added, the run-time of the algorithm increases. The authors also compared their approach to the grid-based LSH-scheme provided by Driemel and Silvestri, and found that the disk-based approach scores slightly better on accuracy. They also conclude that the presented sketches is effective for trajectory processing, leading to us adapting their concept of ordered sketches for our research.

3.3 Other existing work

In the literature, there exists several research articles which explore and discuss how trajectory analysis can be accelerated. Different approaches and techniques have been proposed for this case, and we can roughly categorise them into approximation-based approaches and pruning-based approaches. The first aims at reducing time complexity through approximation, and the latter seeks to reduce the total number of computations to speed up the computation time.

Approximation-based work

In 2016, Aye *et al.* [29] explored the effect of implementing LSH and distance-based hashing (DBH) to accelerate similarity computations among trajectories. They implemented simple versions of LSH and DBH, that transform trajectories into binary representations where each bit represents one of the possible buckets that trajectories can be hashed to. A similarity score between the trajectories can then be calculated based on the euclidean distance between their binary representations; an approach close to the binary sketches in [8]. The approximated similarities were then evaluated by how well they formed clusters under different clustering algorithms. The approximations performed well for LSH, both in terms of speed and accuracy, however DBH did not show sufficient accuracy.

Their approach with LSH did not include any way to maintain the ordering of points and movement direction of the tracked object, thus creating big simplifications as the ordering of points often are interesting. Furthermore, two of the three data-sets used are considered simple and include few complex trajectories. The results also show a drastic reduction in accuracy for the last more complex data-set.

Another approximation-based approach is the well-known Fast-DTW algorithm, which reduces the similarity computation time for DTW from quadratic to linear time-complexity [30]. The algorithm utilises multiple layers of the cost matrix described in subsection 2.2.1 with increasing resolution, such that the top-layer is a coarse representation of the cost matrix and the bottom layer is a full representation. The algorithm iterates through the layers, beginning with the computation of an initial path at the top level, and for each underlying layer, the path is projected and a more correct path is searched for within a given distance from the previous path. This process goes on until the original bottom layer is reached. Despite reducing the time complexity to a linear factor and showing a generally good approximation factor, the true potential of fast-DTW is limited to longer trajectories as the amount of overhead for shorter trajectories are relatively larger than for longer trajectories. In addition, as shown by Wu and Keogh [31], fast-DTW also fails to find good approximations under certain conditions.

Other recent work apply a more modern approach by using artificial intelligence to accelerate the computation of similarity among trajectories. In 2019, Yao *et al.* [5] introduced a neural metric learning method called *NeuTraj*, which supports several trajectory similarity measures. Simplified, NeuTraj calculates the similarity between a number of seed trajectories from the database using a desired distance measure, and uses these similarities to train a neural network that can approximate similarities between all trajectories in the database. By utilising artificial intelligence, the authors achieved significant acceleration of computation of similarities, with a factor of minimum 3 times better than other approximation methods. However, the time cost for training NeuTraj is not reflected by that factor. Furthermore, the approach is dependent on seed samples and the computation of their pair-wise true distances, which is also another training-related factor.

A similar approach to solving the problem of efficient computation was published the same year by Ruobing *et al.* [32]. They developed a deep network structure, *DTSM*, to calculate similarities between trajectories. The network was trained by providing pair-wise trajectory samples with their pre-computed corresponding DTW similarity as labels. DTSM calculates geohashes of the trajectories points, and use these hashes as input to the deep network to compute the similarities. Their model showed over 90% accuracy with a tolerance of 2.5% from their true similarities, and a computation speed significantly faster than the Fast-DTW algorithm. However, like [5], the network requires training and thus also the computation of DTW similarities of the samples. In addition, it is crucial that the training-data reflects the characteristics of the entire data-set, so that the networks computations will approximate the true similarity values.

Pruning-based work

Within the field of distance-based data analysis of trajectories, there have been several approaches that seeks to reduce the total number of trajectory comparisons to achieve faster computation time. Strategies seeking this approach often include pruning and indexing, and are generally adapted to trajectory problems related to top-k similarity search.

Xie *et al.* [33] created a distributed index to be used for trajectory similarity search for a large trajectory data-set. The distributed index was created so that distributed computing could be applied to the problem, thus creating a high-performance distributed framework. The index was constructed so that several techniques could be applied to prune the search area and accelerate the search process, which performed well in terms of scalability.

Another study that utilised indexing for accelerating similarity search among trajectories was conducted by Gowanlock and Casanova [34]. They developed different indexes that could be utilised by GPUs to compute top-k similarity searches, and compared them to a classic CPU-implementation of a R-tree index. Their work achieved significantly shorter response-times than classic CPU-implementation, and especially relatively shorter times for larger volumes of trajectories.

Even though these strategies achieved good scalability for the top-k search problems, they remain unfit for efficiently computing distances of all trajectory pairs, since the similarity between every pair of trajectories in the data-set must be computed, thus comparing all trajectories pair-wise.

3.4 Contribution

With this thesis, we seek to explore whether the LSH can be utilised for more efficient computation of similarities to cope with the problem of scalability when dealing with larger volumes of trajectories. To the best of our knowledge, utilising LSH schemes for this purpose is a relatively unknown field with only a few existing studies presented earlier in this chapter. We seek to explore the usability of LSH techniques that preserve the direction of trajectory movement directly related to the problem of computing trajectory similarities. We take the schemes and some of the results from [7] and [8] as a starting point, and further explore the potential of these LSH schemes.

Additionally, as this research is exploratory, both the methods and the results is valuable and minor contributions itself, since they might inspire and guide other future work within this field.

Chapter 4

Methodology

This chapter describes the working methodology used to answer the research questions provided in section 1.2. In short terms, we adapted the LSH schemes provided by Driemel and Silvestri [7] and Astefanoaei *et al.* [8] to approximate trajectory similarities of two data-sets. We evaluated performance of the LSH schemes in terms of accuracy and efficiency by comparing against their true values. Lastly, we generated clusters from the approximated trajectories to understand the similarities performance of a practical usecase.

4.1 Setup

The experiments carried out in this thesis were all conducted on a personal computer with a generation 12 Intel Core i7 processor and 16 GB of RAM, running Microsoft Windows 11 Home. All experiments were set up and ran in local Jupyter Notebook instances using Python 3.10 as programming language. A Python environment was chosen due to the availability of modules and research packages, and we have chosen to utilise *traj-dist* [35] for trajectory similarity algorithms and *sklearn* [36] for data analysis.

4.1.1 Data-sets

To answer the research objectives presented in section 1.2, we have chosen to utilise two rich publicly available data-sets containing real-life movement data. Both data-sets contain trajectories over taxis' movements in two major European cities. As elaborated in section 2.3, one of the key features of LSH is the ability to reduce time complexity when comparing complex objects, which is why taxi-data was chosen for this research. We chose to utilise two data-sets for this research in order to get a deeper understanding of the schemes and strengthen the results.

In order to fully evaluate the schemes according to the research questions, especially RQ2, we generated ten data-sets from both data-sources with varying sizes that were used in the experiments. The smallest sets contained 100 trajectories, and the largest set contained 1000 trajectories, where the sets between increased with an interval of 100 trajectories. All data-sets were constructed so that a larger data-set is a superset of all smaller data-sets. Additionally, We also generated a test set for both data-sources (Porto and Rome) containing 50 trajectories that was utilised to find the scheme configurations for this research as shown in section 5.1.

Rome Taxi data-set

The Rome taxi data-set, available at CRAWDAD [37], contains movement data from 320 taxis collected over a time period of 30 days in 2014. The data were collected with a sampling frequency approximately every seventh second, resulting in relatively fine-coarse data and a total of 21.8 million individual data-points. The data-set are not divided into natural trajectories representing trips etc., but contains rows with a driver id, a timestamp and a geographical location. The data was collected as long as the taxis were active, both with and without passengers.

To be able to utilise the data-set, we extracted smaller trajectories with a minimum size of 40 data-points and a distance between four and six kilometres in an effort to mimic taxi-trips. With inspiration from Astefanoaei *et al.* [8], we only extracted trajectories that had a minimum total distance of 2.5 times the distance between the start and end point. Furthermore, to reduce the possible geographical extent of the trajectories, we only extracted trajectories that lied entirely within a bounding rectangle of approximately 6 x 8 km over Rome's city centre. An overview of the total 1000 extracted trajectories are shown in Figure 4.1.

Porto Taxi data-set

The Porto data-set, which can be publicly found at Kaggle [38], consists of trajectories collected from 442 taxis in Porto. Opposite to the Rome data-set, this set is already structured and divided into trajectories representing a single trip with a customer. The datapoints were sampled roughly every 15 seconds, and were collected from July 2013 to June 2014. The set was published as a part of taxi trajectory prediction competition and contains over 1.7 million trips. Each trajectory contains some metadata fields, including timestamp and a field indicating if the trajectory has missing data. Since the data-set was already split into trips, we extracted 1000 trajectories with a length of 40 datapoints, bounded by a 6 x 8 km rectangle over Porto city centre. The extent of the extracted trajectories are visualised in Figure 4.2. As can be seen in this figure and Figure 4.1, the underlying road-network in Porto seems to be slightly more coarse-grained than in Porto, resulting in a slightly less complicated data-set than the Rome set.

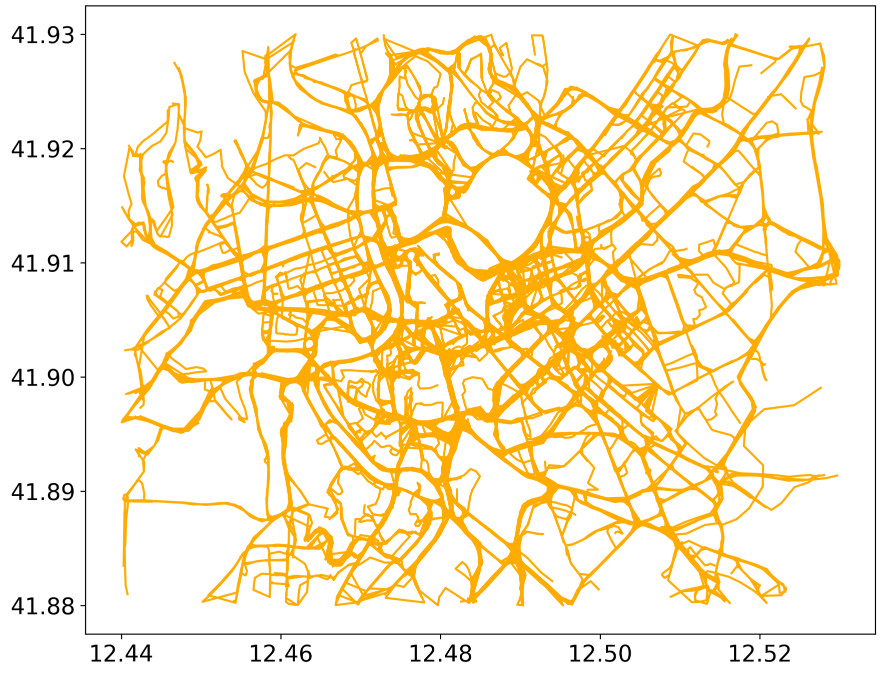


Figure 4.1: Overview of the extracted trajectories from the city centre of Rome.

4.1.2 Data cleaning and preparation

The data-sets that we used are gathered from vehicles in an urban environment where GPS-signals and the collected data-points are likely to be distorted by noise. It was therefore essential to filter out noisy data when we extracted the trajectories from the data-sets. To prevent possible noise- and outlier-datapoints in the extracted trajectories from both data-sets, we removed all trajectories where the distance between two consecutive points were greater than a threshold. To ensure overall good quality of the extracted trajectories, we did not extract trajectories that was marked with missing data from the Porto set. During data extraction from the Rome set, we removed trajectories where two consecutive points had a time difference of more than 32 seconds; indicating that data were missing between the points.

4.2 Creation of benchmarks

One of the key element of this thesis was to evaluate whether the grid-based and disk-based LSH-schemes are adaptable and suitable for similarity computation (RQ1), and to understand their performance (RQ2 & RQ3). To fully answer the research questions, we generated benchmarks that was used as a basis for comparison.

We computed the similarity values between the trajectories in both data-sets.

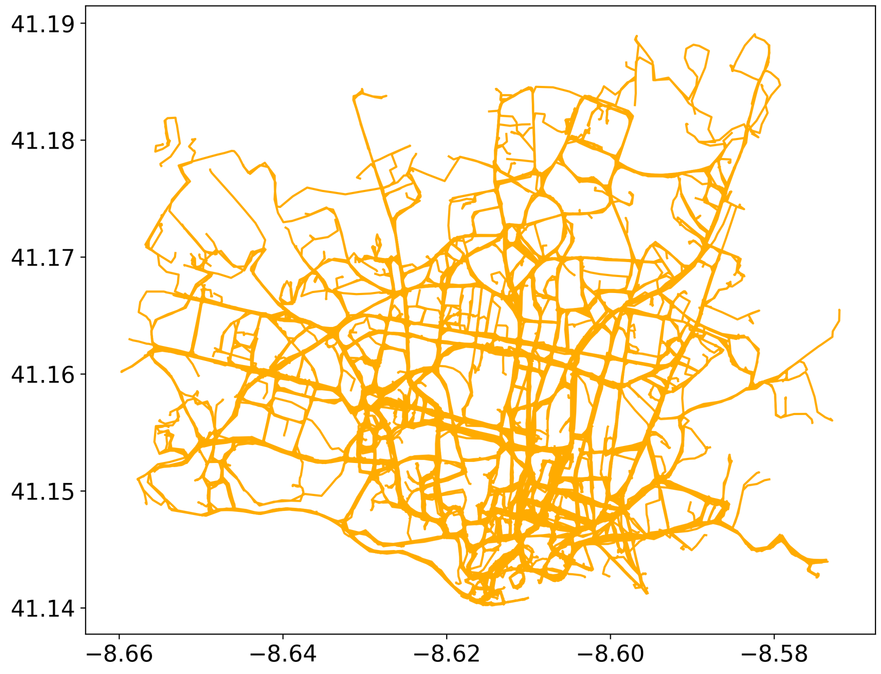
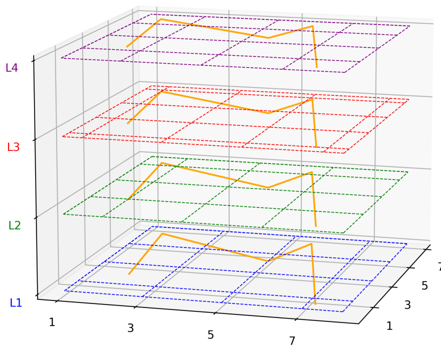


Figure 4.2: Overview of the extracted trajectories from the city centre of Porto.

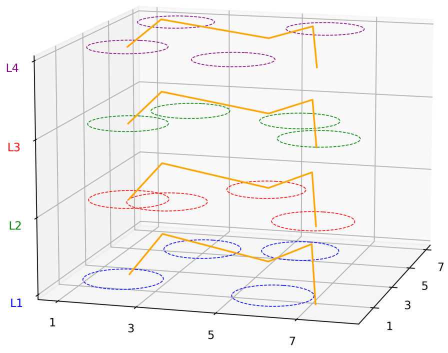
Since trajectory similarity measures emphasise different properties of the trajectories, we computed similarity values using both Dynamic Time Warping and Fréchet distance as distance measures. These similarities were stored to be used as benchmarks, and we will later in this thesis refer to them as the *true similarities* of the data-sets. Additionally, we measured the computation time required for computing the true DTW similarities for the different sized data-sets. This was done by running 10 independent parallel processes and measured the process time of each process.

We used the true similarities (DTW and Fréchet) from both Porto and Rome to generate benchmarks clusters to be used when answering RQ3. We utilised the hierarchical agglomerative clustering algorithm provided by *sklearn* [36], using *ward* as the linking method for this task. As described in section 2.4, HCA will merge clusters until it reaches the stop criterion, which is given by the number of desired output clusters. We chose to set this number to be 30 clusters, without any knowledge of possible underlying patterns in the data-sets. The number was chosen with hope that it could provide clusters that are likely to be clear, but still few enough for proper visual inspection.

The usage of these benchmarks are further elaborated on later in this chapter.



(a) Grid-based LSH scheme with a grid-tile resolution of 2.



(b) Disk-based LSH scheme with 4 disks and a diameter of 2.

Figure 4.3: Visualisation of a hash family of four hash functions for the grid-based and disk-based LSH techniques. A trajectory is represented by by the orange line, and can be seen intersecting with grid tiles and disks in the layers for the respective approaches.

4.3 Implementation of LSH-schemes

First, we implemented the basic grid scheme from [7], where the grid is randomly shifted with a parameter $t \in [0, \delta)$. As elaborated in section 3.1, δ represents the resolution (grid tile width) of the grid, i.e. the granularity of the hashing functions, and was set individually for both data-sources. Layers were implemented in the scheme, such that the layers constructed a family of hashes. We implemented the hash functions to return the set of grids that the trajectories' points are located in, ordered chronologically and with consecutive duplicates removed. Furthermore, the grid scheme was constructed so that a complete trajectory hash consisted of a list of hashes where each list element corresponded to a layer in the scheme. A visualisation of the implemented grid scheme are shown in Figure 4.3a.

The disk-based scheme from [8] was also implemented and slightly adapted. We constructed the disk-based scheme by deploying n random disks of a given size s in a plane which span the trajectories. We repeated this process for l layers, resulting in l different hashing functions. In the paper, the hashing functions were originally designed to return the sequence of deployed disks trajectories moved through, leaving us with no context indicating the relative distance between any two disks. Therefore, an option for the disk-based scheme was implemented where the hash output are similar, but now represented by the intersecting disks' geo-coordinate. Like the grid-based scheme, the disk-based scheme was implemented so that the output of hashed trajectories consisted of a list of hashes where each element corresponds to a layer (hash function) in the scheme. A visual representation of this scheme can be seen in Figure 4.3b.

Compared to the grid-based scheme, the process of hashing trajectories are slightly more complicated. For every trajectory, each point could possibly intersect multiple disks, meaning that every point in the trajectory must be controlled for intersecting each disk. This is far more comprehensive than snapping points to a fixed grid, where simple arithmetic are applied. To prevent unnecessary time spent computing the hashes, we experimented with two different approaches. A simple quad-structure and a KD-tree were implemented as alternatives to more efficiently finding points laying within disks.

4.3.1 Choice of parameters

Both the grid-based and the disk-based schemes required parameters to be set. The first required the grid-tile width (resolution) and the numbers of layers to be set, and the latter required the number of disk and their diameter, as well as the number of layers to be set. The ideal configuration of the schemes' parameter values are dependent on the underlying data and its' granularity. Additionally, as we shall see, setting the parameter values was a trade-off between precision and efficiency, since a fine-grained grid and large disks will produce longer hashes containing more information about the hashed trajectories. To find a working configuration of the parameters for our experiments, we ran 20 parallel jobs with different combinations of the parameters using the test-data-set from both cities and for both schemes. From the output hashes, we computed their similarities and compared them to their true similarities, as presented in Section 5.1.

From these observations, the grid-based scheme over Porto and Rome was configured respectively with 5 layers with a grid tile width of 1.6 km and 4 layers with a tile width of 1.2 km. The disk-based scheme over Porto was configured using 4 layers with 60 disks with a diameter of 2.2km, and the configuration over Rome was set to 5 layers with 50 disks with a diameter of 1.6 km. These values was chosen for the remaining experiments, and was used for all subsets of a data-set. The justification of the configurations, along with the effects of different parameter values are further discussed in chapter 6.

4.3.2 Choice of hash similarity measure

The computation of similarities between the disk hashes was originally proposed by [8] to be computed using edit distance for strings, since the hashes were sequences of disks. However, this approach will treat two trajectories that are close but with no common disks as equal as two trajectories that are distant from each other. Dynamic Time Warping was therefore implemented on the location-aware optional hashes, using euclidean distance as cost function. For the grid-based hashing scheme, DTW was also implemented as similarity measure where the Manhattan-distance between the grid tiles was used as cost function. In the early phases of this research, we compared the original edit-distance against the optional DTW, and then decided to focus this research on the optional DTW for both schemes, since it achieved better results. The choice of hash similarity measure is further elaborated on in chapter 6. Which similarity measures that best suits the schemes is an interesting study in itself, but no further measures were tested in our research as it falls outside the scope of this thesis.

4.3.3 Creation of LSH-similarities

The hash-similarities was computed by applying the above-mentioned similarity measure to the hashes generated by both the grid-based and disk-based scheme with the specified parameter-configurations. For the grid-based approach, similarities was computed using DTW with Manhattan-distance between grid tiles. For the disk-based approach, similarities were computed using DTW with euclidean-distance. The similarities between trajectories were generated by the sum of the trajectories' layers pair-wise DTW similarity. With two different data-sets and two different LSH-schemes, a total of four unique hash-similarity matrices was generated in total.

4.4 Analysis of LSH-similarities

The performance of the LSH generated similarities was evaluated according to their efficiency and accuracy. In terms of efficiency, the time needed to generate the hashes was measured, and more importantly the time needed to compute the hashes. The accuracy was evaluated by computing the correlation against the true similarities.

We utilised the python core module *timeit* to measure the efficiency of the LSH schemes. Timeit is a library created with the purpose of timing minor bits of python code. To prevent distortions and the timing of being affected by other running processes, we configured timeit to measure the *process time*, which is the total elapsed CPU time of a process. To understand the efficiency of the LSH schemes under various data-size inputs (RQ2), the time required for computing the similarities was measured for all subsets, with sizes from 100 trajectories to 1000. Additionally, the time needed to generate the trajectory hashes for the largest data-set was also measured. This process was repeated for both schemes and for both cities/data-sets. Each timing-experiment was conducted by running a total of 10 individual parallel processes and computing the average run-time.

As mentioned at the beginning of this section, the accuracy and effectiveness of the LSH hashes was evaluated by computing the correlation between the hashed similarities and the true similarities. As an opposite from the true similarities, the computed hash similarities are not symmetric, since the hashes they are computed from are based on random variables. However, we believe that the similarities in most cases are symmetric-like, and that pearson correlation coefficient gives a good indication of the linear accuracy between the LSH-similarities and the true similarities. Furthermore, to get a deeper understanding of the effectiveness of the LSH-schemes, we generated *2d-histograms* over the similarities. A

2d-histogram helps visualise the relationship between two numerical variables, and visualises the distribution of the values for both variables. We generated 2d-histograms between the hashed similarities from both schemes and both cities, and the true similarities from DTW and Fréchet distance.

4.5 Practical analysis using clusters

To answer RQ3 and understand the practical performance of the similarities, we utilised *sklearn* [36], a library which contains, among other things, functionality for clustering and methods for clustering analysis. Based on the generated similarities, we generated clusters using sklearn's hierarchical cluster model.

4.5.1 Cluster creation

For both data-sets and both LSH schemes, we generated clusters using the agglomerative model. As elaborated in subsection 2.4.1, agglomerative clustering continues to merge clusters until the provided number of clusters is given, and for this research we configured the final number of clusters to be 30. We had no knowledge of underlying patterns in the data-sets and with the data being unlabeled as well, the number was chosen arbitrary primarily due to fact it being small enough for easy analysis and visual inspection, but still large enough to capture peculiarities in the different clustering outputs.

The agglomerative clusters was created using *ward* as the linking method between the clusters, which minimises the variance between the merging clusters. The clustering model was configured to use "euclidean" distance as measure when deciding which clusters to merge. The model was given the similarities generated by the LSH-schemes as input, and generated the clusters based on these similarities.

4.5.2 Cluster evaluation

We mainly evaluated the clustering performance of the LSH-similarities after three measures: rand index, Davies-Bouldin index and visual inspection. Both indexes, described in subsection 2.4.2, was used as basis for the evaluation, in addition to a visual inspection of the generated clusters for performance evaluation.

The rand index was computed to quantify the resemblance between the clustering output of the benchmarks and the clustering output where the LSH-similarities was used. We utilised sklearn's *metrics* module in our work to compute the rand index between the clustering outputs. Rand index is a comparison-based measure, and can only indicate how close two clustering outputs are. In order to understand the internal performance of the clustering outputs, we implemented an alternative trajectory version of the Davies-Bouldin index.

As elaborated in subsection 2.4.2, Davies Bouldin index computes the centroids of clusters. This is usually done by computing the average value of the objects in a cluster, but with complex cluster-objects like trajectories, computing an average value is infeasible. Instead, a cluster's centroid was implemented to be represented by the trajectory with the smallest combined distance to every other trajectory in a cluster. The within-cluster distance, between-cluster distance and the final DBI was then implemented and later computed using this representation of a cluster's centroid. Since DBI is an internal evaluation measure, we cannot compare the DBI of clusters from true similarities and hashed similarities. Therefore, we repeated this process using the hashed clustering outputs in combination with the true similarities, so that direct comparisons can be made independent of the similarities used for clustering.

When evaluating clustering outputs, indexes and measures provide a good understanding. However, with the complex structures of trajectory data, a visual inspection of the clustering outputs can reveal other interesting aspects hidden from the indexes. The clustering outputs was therefore visually examined to evaluate the generated clusters properly.

4.6 Simplifications

We wrap up this chapter by mentioning the simplifications that was made through the implementation and experimental phase of this thesis. Most of the simplifications were done because of implementation reasons, with only minor effects.

During the computation of disk-based similarities and the computation of the true similarities, euclidean distance was used as measure between trajectory points. Since the data-sets contain coordinates on a sphere, the correct measure would have been the Haversine distance, which provides the surface distance between the points. However, since the trajectories were constrained to a relatively small areal extent, we chose to use euclidean distance since the difference between euclidean and Haversine distance are tiny in small geographical areas.

Furthermore, the disks in the disk-bases LSH scheme were implemented using geographical coordinates with a fixed radius. Due to the fact that the distance covered by one longitudinal degree changes depending on the current latitude, means that the implemented disks were slightly more spherical than round.

A tiny number of the trajectories did not intersect with any disk in a layer. When the similarity score between two hashes that had at least one empty layer was computed, the distance between the two layers was arbitrary set to be 0.5.

Lastly, the experiments measuring the time needed to compute the true similarities was extremely time consuming, and we therefore chose to discard the measurements of process time for the largest sub-sets. Instead, we interpolated the run-times for the largest sub-sets based on the measured time for the smaller sub-sets.

Chapter 5

Results

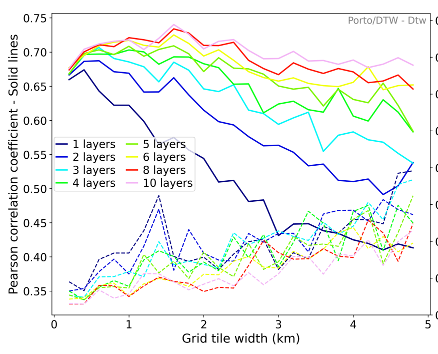
In this chapter, we present the observations and results from the conducted experiments. First, the results regarding parameter choice are presented, before the findings related to the understanding of the schemes' performance are given. At the end of this chapter, we present the results from the clustering processes along with the belonging index scores results.

5.1 Parameter configuration

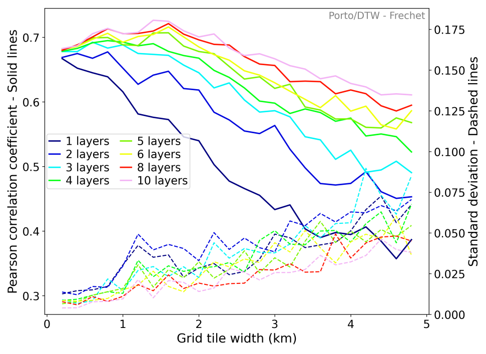
Figures 5.1 to 5.5 shows how the LSH schemes' approximated similarities correlate with the true similarities (DTW and Fréchet) under different parameter configurations. The figures display the average correlation from 20 parallel jobs for both schemes and both test-data-sets, and the standard deviation from the jobs are presented with dashed lines in the figures. In Appendix A, we have included corresponding figures where the originally proposed edit-distance was used as similarity measure.

Even though the figures were generated to determine the parameter values to be used in further analysis of the LSH schemes, we remark some general observations on parameter choice. For both schemes and cities we observe greater correlation with the true similarities when the number of layers increase, as well as a lower standard deviation which indicates a more predictable configuration. From Figures 5.1 and 5.2, we notice that the grid-similarities correlation with the true similarities drop as the grid tile width increases (lower grid resolution), and that the standard deviation also increases. Additionally, we observe peak correlation values slightly above 0.7 for the grid schemes, except for the Rome-test-set having a peak correlation of 0.5 with the true DTW similarities.

In Figures 5.3 and 5.4, we observe a similar phenomenon for the disk schemes, where a greater disk diameter corresponds to higher correlation with the true similarities, and a lower standard deviation. We remark that the peak correlation values for the disk schemes vary between 0.6 and 0.8 for the test-sets. In

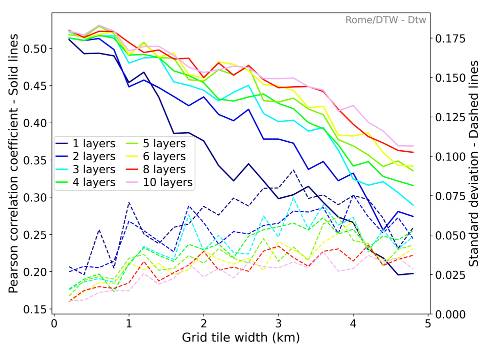


(a) Correlation with DTW

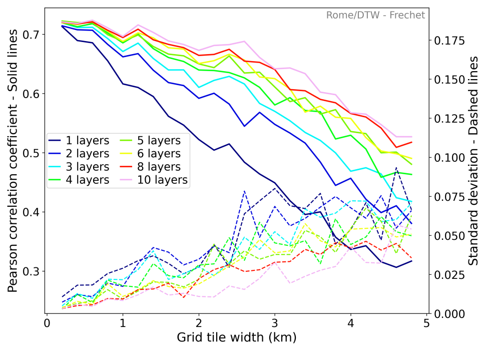


(b) Correlation with Fréchet

Figure 5.1: Correlation with the true similarities for the grid similarities with varying grid resolution and number of layers. Results from the test-set of Porto. Standard deviation marked with dashed lines.



(a) Correlation with DTW

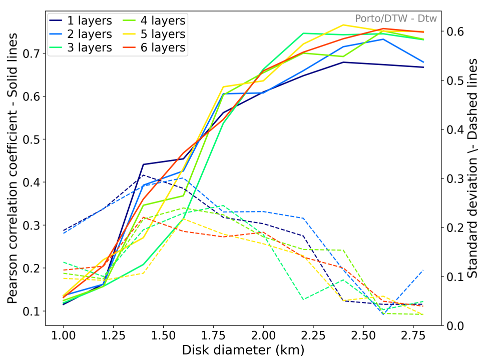


(b) Correlation with Fréchet

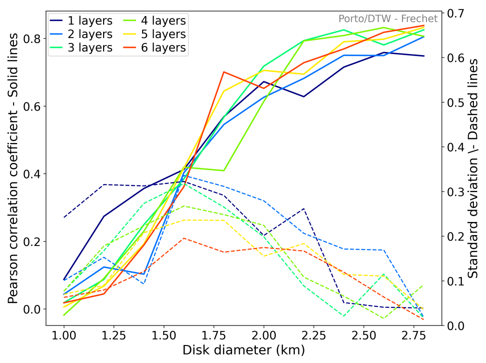
Figure 5.2: Correlation with the true similarities for the grid similarities with varying grid resolution and number of layers. Results from the test-set of Rome. Standard deviation marked with dashed lines.

Figure 5.5, we observe that an increased number of disks corresponds to higher correlation with true similarities and a lower standard deviation. We also observed a relatively high standard deviation for the Porto set for all diameters up to approximately 2.25 km, and a similar observation for the Rome set up to 1.5 km disk diameter. We also note that the number of layers in the disk Porto set have no clear connection to correlation until the diameter reaches 2.25 km.

From these results, we chose the values presented in subsection 4.3.1 to be used for further analysis. For readability of this thesis, we restate the values here. The grid configuration of Porto was set to 5 layers with a grid tile width of 1.6 km, and the grid over Rome consisted of 4 layers with a tile width of 1.2km. The disk configuration over Porto consisted of 4 layers with 60 disks with a diameter of 2.2 km, and the disk scheme over Rome consisted of 5 layers of 50 disks with a diameter of 1.6km. The values were chosen from the observation of having good

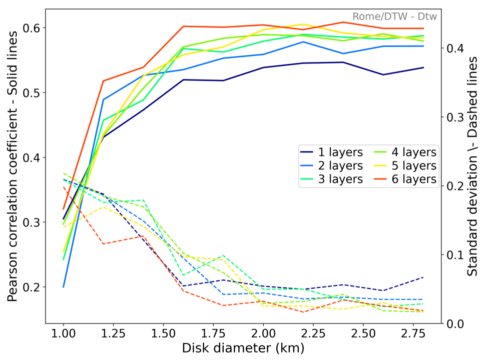


(a) Correlation with DTW

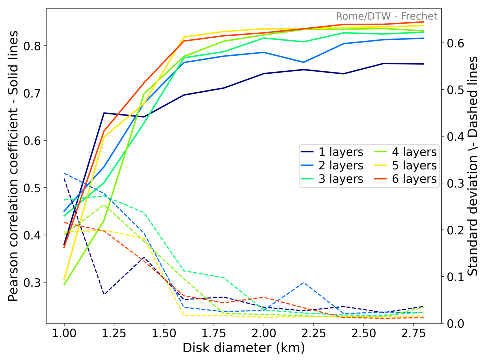


(b) Correlation against Fréchet

Figure 5.3: Correlation with the true similarities for the disk similarities with varying disk diameter and number of layers, computed with 50 disks. Results from the test-set of Porto. Standard deviation marked with dashed lines.



(a) Correlation with DTW



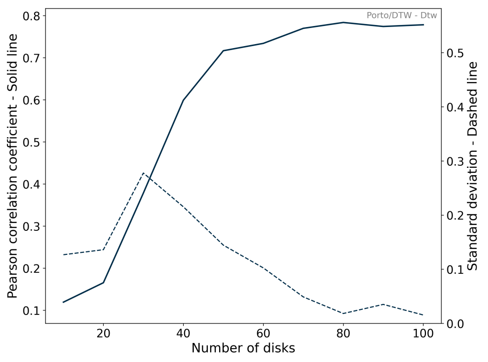
(b) Correlation against Fréchet

Figure 5.4: Correlation with the true similarities for the disk similarities with varying disk diameter and number of layers, computed with 50 disks. Results from the test-set of Rome. Standard deviation marked with dashed lines.

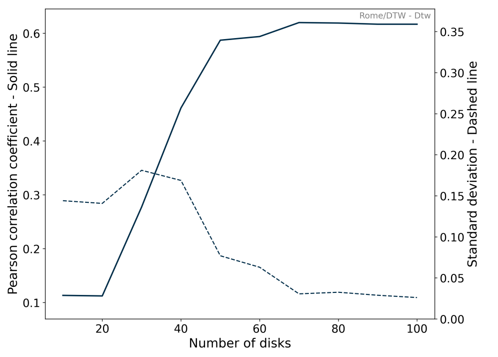
correlation with both true similarities and low standard deviation. We remark that these chosen values not necessarily corresponds to the parameter configurations with the best correlation. The choice of parameter values are therefore further discussed in chapter 6.

5.2 Performance analysis

In this section, we present the findings related to RQ2, where the schemes have been configured with the chosen parameters derived from section 5.1.



(a) Porto - Computed with 4 layers with disk diameter of 2.2 km.



(b) Rome - Computed with 5 layers with disk diameter of 1.6 km

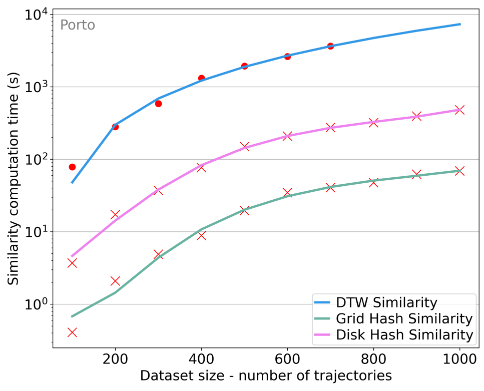
Figure 5.5: Correlation with the true DTW similarities for the disk similarities with varying number of disks. Results from the test-sets of Porto and Rome. Standard deviation marked with dashed lines.

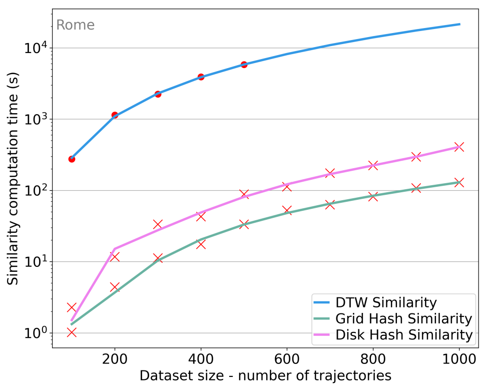
5.2.1 Efficiency

In figure Figure 5.6, we have presented the computation-times required to compute the similarity-values using the grid-hash and the disk-hash schemes for varying trajectory input loads, in addition to DTW as reference. The run-times of Fréchet have not been computed due to immense computational load. Even though both the grid and the disk schemes computation times are presented in the the same figures, we remark that a comparison of the different schemes' similarity computation-times must take into account that the times are directly influenced by the configuration parameters.

There are mainly three significant observations that are prominent in this figure. Firstly, we observe that the run-time for grid similarity computation over both Porto and Rome are roughly 100 times faster than computing the true DTW similarity for all data-sizes. Secondly, we observe a similar slightly weaker, but still strong trend for the disk-based similarity computation-times, where the respective speed-up of the Porto and Rome data-sets are approximately 13 and 21 times faster than the true DTW similarity computation times. Lastly we observe that the disk-scheme over Porto has the worst computation time of the all the schemes, even worse than the disk-scheme over Rome. This is contra-intuitive since the reference similarity computation time of Rome is much greater than the reference of Porto, and is further discussed in chapter 6. By implication from the first two observations, we note that both hashing schemes scale far better with increasing input-load of trajectories than true DTW does. We remark that the exact run-time measurement of the highest trajectory loads for DTW was omitted due to exhaustive computation. Fréchet distance was omitted as benchmark for the same reason.

In order to compute the similarities between trajectory hashes, the hashes itself must be generated. We have presented the run-times required to compute





(a) Porto **(b)** Rome

Figure 5.6: Comparison between the computation time required to generate the hashed similarities and the true DTW similarities. The red dots and crosses are the average value measured from 10 parallel runs, and the plotted lines are interpolated from these values.

1000 trajectory hashes over Porto and Rome in Table 5.1. We observe extremely low overhead when generating the grid hashes, where the average computation time for both data-sets being less than half a second which is significantly less than 1% of the schemes' required time for similarity computation. We observe a slightly higher computation time for the disc-scheme, where the naive approach on average takes vaguely less than 5 seconds for the Porto-set, and approximately 12 seconds for the Rome-set. These values correspond to around 1% and 3% of the schemes' similarity computation time. We observe that by applying either a quad-structure or a KD-tree to the disk schemes, the time spent generating the hashes can be reduced significantly.

Table 5.1: The measured computation time from the generation of 1000 trajectories' hashes. Measurements from 10 parallel computations.

Hash generation times in seconds (s)			
LSH Scheme and data-set (Alternative generation)	Minimum	Maximum	Average
Grid Porto	0.09375	0.265625	0.167187
Grid Rome	0.18750	0.468750	0.371875
Disk Porto	4.562500	5.390625	4.812500
Disk Porto (Quadrants)	2.109375	2.703125	2.384375
Disk Porto (KD-tree)	1.250000	1.609375	1.471875
Disk Rome	11.218750	12.312500	11.834375
Disk Rome (Quadrants)	5.250000	6.125000	5.771875
Disk Rome (KD-Tree)	2.015625	2.703125	2.370312

5.2.2 Effectiveness

In Figures 5.7 to 5.10, we have presented 2d-histograms visualising how the hashed similarities are distributed in relation to their true similarities. For the hashed similarities from both data-sets and both schemes, we have presented their relation to both reference similarities (DTW and Fréchet) alongside each other. The correlation between the hashed similarities and the true similarities are shown in the lower right corner of each figure. Note that the 2d-histograms are visualisations of single runs. In Table 5.2 we have provided the minimum, maximum and average correlation values from 10 independent runs.

From the 2d-histograms there are some interesting general observations that are important to us. We remark that 2d-histograms essentially visualises correlation, where a dense increasing line-shape indicates a strong relationship between the two variables. Firstly, from all 2d-histograms, we observe that both schemes have a significantly denser line-shape when compared to Fréchet distance than DTW, which naturally entails higher correlation with Fréchet distance.

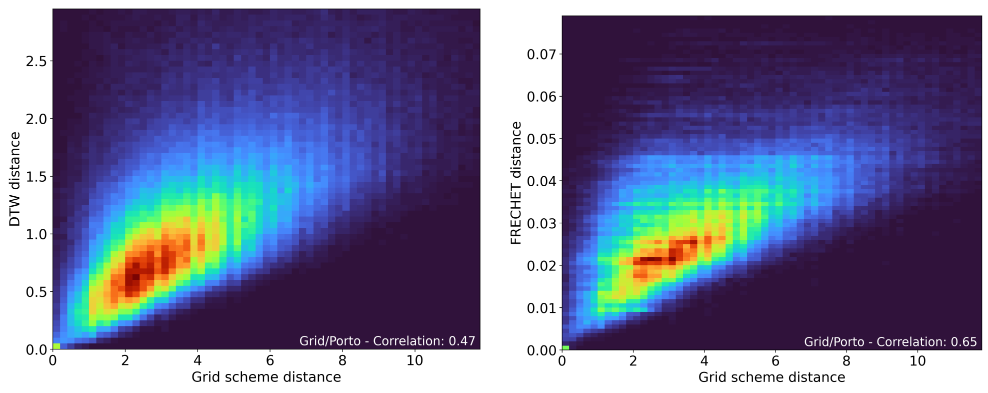
Furthermore, we observe that the disk-based similarities tend to outperform the grid-based similarities with narrower 2d-shapes. As stated earlier, the different schemes have different configurations, and a direct comparison is not necessarily meaningful. However, this observation is further supported by the consistent observation that larger true similarity values have a wider spread of corresponding grid-similarities (Figures 5.7 and 5.8) than their corresponding disk-similarities (Figures 5.9 and 5.10).

A final interesting observation which stands out from the 2d-histograms is the higher spread of the schemes similarities' corresponding DTW distance. Clear examples of this are visible in Figures 5.8a and 5.10a, where the distribution of a grid-similarity of 4, and the distribution of a disk-similarity of 1 are spread over almost the entire y-axis. This observation becomes evident when looking at the corresponding histograms with Fréchet distance in Figures 5.8b and 5.10b.

In Table 5.2, we have presented both schemes correlation with the true similarities for the largest data-set of Porto and Rome. We observe stable correlation values with relatively low standard deviation values. Even though the standard deviation values are low, we note that the difference in correlation between the minimum and maximum values are still not insignificant.

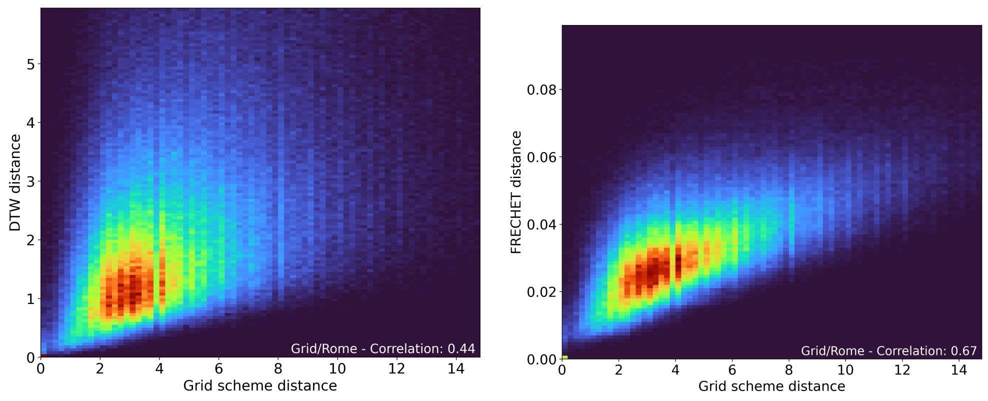
5.3 Analysis of clusters

In this section, we present the results and index measures related to RQ3. We emphasise that the interest for this specific analysis is to evaluate whether the hashed similarities are suited for a practical task, and not strive to generate the most optimal clustering of the underlying data. We have combined the clustering outputs using HCA based on the hashed and true similarities for each data-set in two respective figures for easier visual comparisons. A full-size figure of each



(a) DTW **(b)** Fréchet

Figure 5.7: Relationship distribution of trajectory similarity between grid- and true similarities over Porto. Trajectories are grouped into tiles by their similarities. A tile's colour intensity indicates the number of trajectories present in the tile.



(a) DTW **(b)** Fréchet

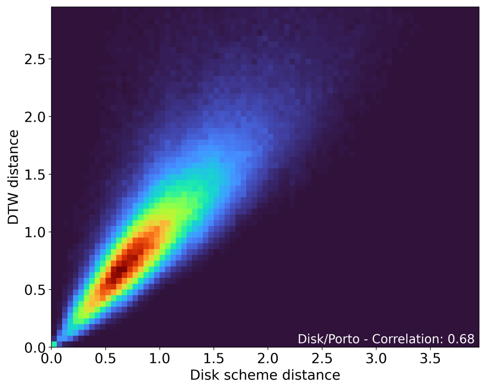
Figure 5.8: Relationship distribution of trajectory similarity between grid- and true similarities over Rome. Trajectories are grouped into tiles by their similarities. A tile's colour intensity indicates the number of trajectories present in the tile.

clustering output are included in Appendix B.

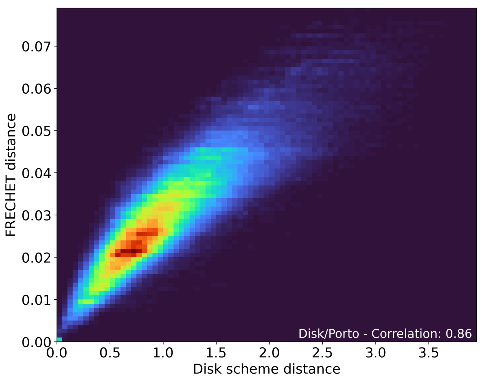
5.3.1 Clustering output

From the visual inspection of Figures 5.11 and 5.12, respectively Porto and Rome clustering outputs, we observe that the two reference outputs from DTW and Fréchet distance are slightly different for both data-sets. Both reference clustering outputs appear approximately equal in terms of density and cluster sizes, but Fréchet distance tends to maintain the trajectories' direction of movement slightly better, which are shown by the gradient colour.

We remark that this is simply an interesting observation of the benchmarks,

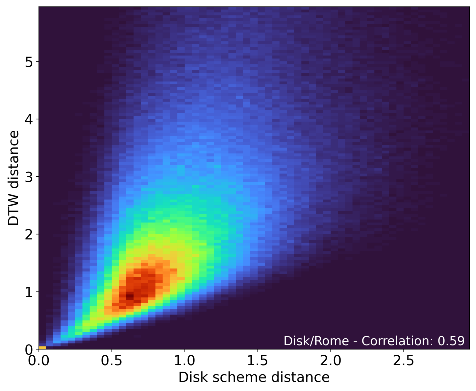


(a) DTW

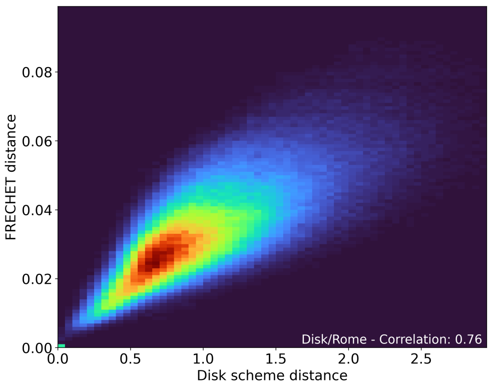


(b) Fréchet

Figure 5.9: Relationship distribution of trajectory similarity between disk- and true similarities over Porto. Trajectories are grouped into tiles by their similarities. A tile's colour intensity indicates the number of trajectories present in the tile.



(a) DTW



(b) Fréchet

Figure 5.10: Relationship distribution of trajectory similarity between disk- and true similarities over Rome. Trajectories are grouped into tiles by their similarities. A tile's colour intensity indicates the number of trajectories present in the tile.

and must not be seen as a result in this thesis. However, our subjective visual observation is that both the grid and disk hash similarities tends to maintain the trajectories direction of movement slightly better than DTW, having more in common with the Fréchet distance. We also note that the disk-based clustering output tends to preserve this property better than the grid-based clustering, especially for the Rome data-set (Figure 5.12b).

Furthermore, we see that the grid-based clusters for both Porto and Rome (Figures 5.11c and 5.12c) appear slightly more spread and less compact than both the true clustering outputs. On the other hand, we observe that the disk-based clusters (Figures 5.11d and 5.12d) are hardly distinguishable from the true clustering outputs in terms of density and size.

Table 5.2: Minimum, maximum and average correlation values between the hashed similarities and the true similarities for both Porto and Rome. The values including standard deviation are the results from 10 independent runs.

Correlation with true similarities for 10 parallel runs				
Similarity type / Reference	Minimum	Maximum	Average	Std. Deviation
Grid Porto / DTW	0.440	0.472	0.457	0.010
Grid Porto / Fréchet	0.619	0.654	0.640	0.010
Disk Porto / DTW	0.638	0.694	0.666	0.017
Disk Porto / Fréchet	0.829	0.859	0.843	0.012
Grid Rome / DTW	0.447	0.468	0.456	0.007
Grid Rome / Fréchet	0.687	0.712	0.702	0.010
Disk Rome / DTW	0.570	0.616	0.601	0.012
Disk Rome / Fréchet	0.764	0.798	0.777	0.009

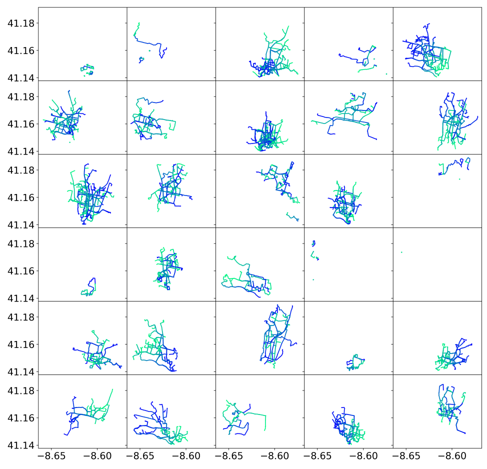
5.3.2 Clustering index analysis

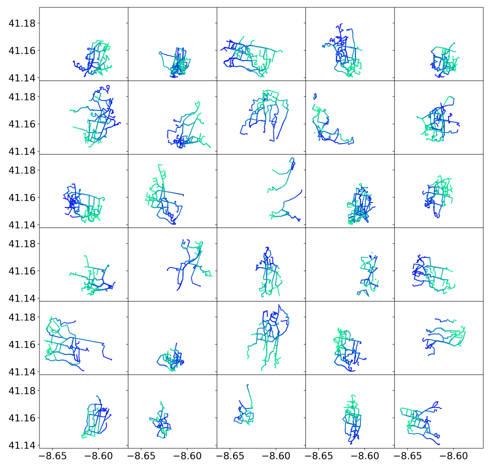
In Table 5.3, we have presented the internal Davies-Bouldin index for each clustering output, decomposed into within-like and between-like distance. From subsection 4.5.2, we remark that there is no meaningful way of directly comparing the values in this table, and that they should be emphasised in combination with the clusters they are representing. In this table, and Table 5.4, the computed Davies-Bouldin indexes are presented, along with the decomposed within-like and between-like values for the schemes' clustering outputs.

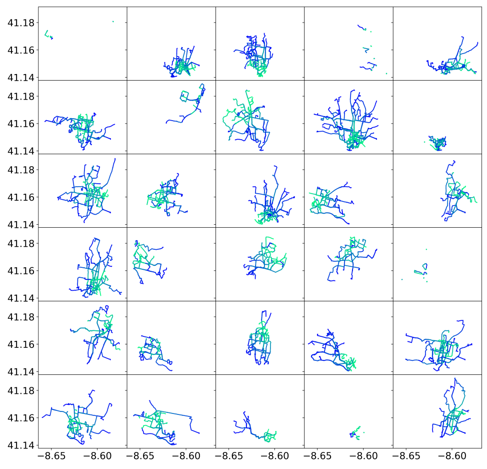
Table 5.3: Davies-Bouldin and the decomposed within-like and between-like scores over the generated clusters.

Decomposed Davies Bouldin index			
Similarity type and data-set	WL ($\frac{WL*100}{WL+BL}$)	BL ($\frac{BL*100}{WL+BL}$)	DB
DTW Porto (reference)	23.0316	76.9684	0.00997
Fréchet Porto (reference)	31.7614	68.2386	0.01551
Grid Porto	44.8063	55.1937	0.02706
Disk Porto	29.2465	70.7535	0.01378
DTW Rome (reference)	30.1179	69.8821	0.01437
Fréchet Rome (reference)	32.7586	67.2413	0.01624
Grid Rome	29.8108	70.1892	0.01416
Disk Rome	38.4741	61.5259	0.02084

The relative Davies-Bouldin index scores of the schemes cluster, where the true similarities have been laid as basis for computation, are presented in Table 5.4. The schemes' adjusted DB scores relative to its reference have been included in the rightmost column. Again, we remark that these values are a direct result of


(a) True similarities - DTW


(b) True similarities - Fréchet


(c) Grid hash similarities

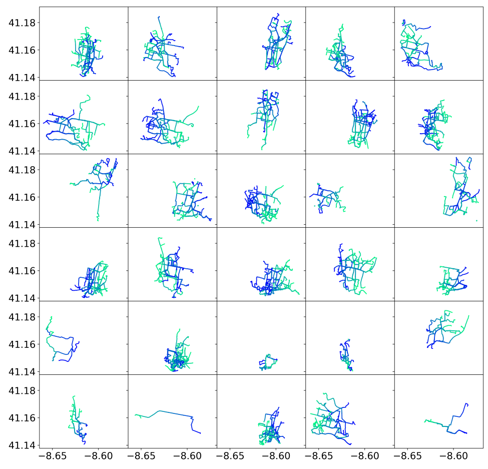
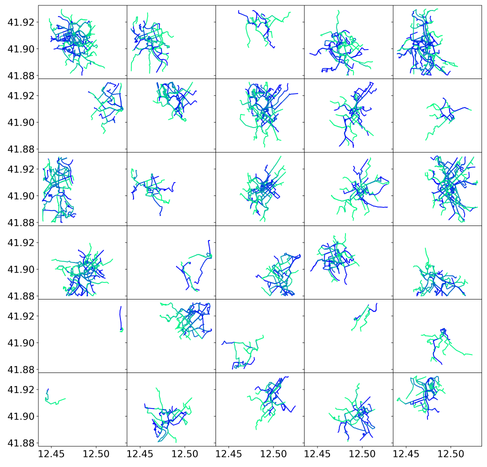

(d) Disk hash similarities

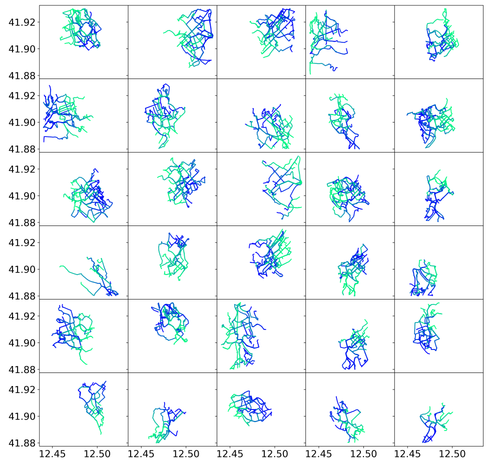
Figure 5.11: Trajectory clustering outputs over Porto generated from different similarities. The upper two outputs are generated using the trajectories' true similarities, and the lower two outputs are generated using the similarities from the Grid LSH scheme and the Disk LSH scheme.

the schemes configurations, which must not be forgotten when comparing the schemes. One of the observations that stands out in this table is the schemes closer proximity to the Fréchet clusters than the DTW clusters. The disk-scheme over Porto using Fréchet as reference scored the best with a relative score of 1.1315 to the Fréchet DB index, having both WL and BL distances close to the reference values. The good clustering performance of Disk Porto is further substantiated when the DTW reference is also laid as basis.

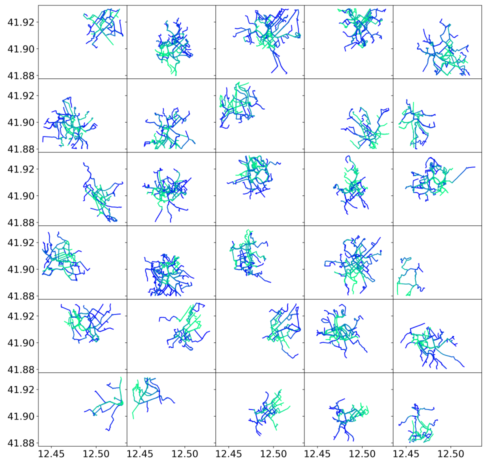
From the Rome data-set, we observe slightly better index scores for the grid-schemes. Despite the slightly better performance of the grid scheme with a relative



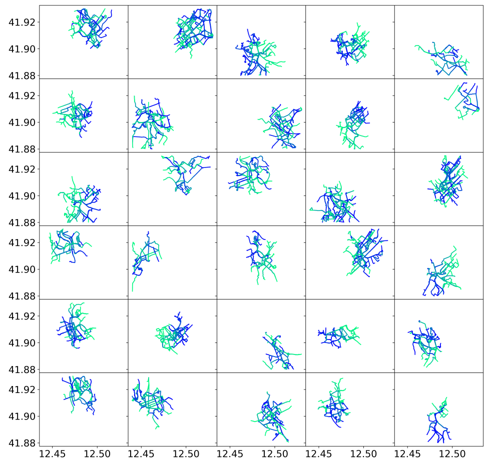
(a) True similarities - DTW



(b) True similarities - Fréchet



(c) Grid hash similarities



(d) Disk hash similarities

Figure 5.12: Trajectory clustering outputs over Rome generated from different similarities. The upper two outputs are generated using the trajectories' true similarities, and the lower two outputs are generated using the similarities from the Grid LSH scheme and the Disk LSH scheme.

DBI score of 1.2826 to Fréchet distance, the grid performance in Rome is still not as good as the disk performance in Porto. Another stand-out observation is the grid- and disk-schemes scores when DTW is used as reference, having the two worst relative DBI scores.

In Table 5.5, we have presented the rand index scores from the clustering outputs. There are two observations that we emphasise from this table. Firstly, both the clusters from the disk-based schemes over Porto and Rome have a slightly higher rand index score than the grid-based schemes. This result means that the composition of disk-clusters are more similar to the true clustering outputs based

Table 5.4: The relative Davies-Bouldin scores decomposed into within-like and between-like factors over the generated clusters. Computed using the hash similarities clusterings with the true similarities output.

Decomposed Relative Davies Bouldin index				
Similarity type and data-set / reference	WL ($\frac{WL*100}{WL+BL}$)	BL ($\frac{BL*100}{WL+BL}$)	DB	DB/DB_{ref}
DTW Porto (reference)	23.0316	76.9684	0.00997	-
Grid Porto / DTW	36.9071	63.0929	0.01950	1.9559
Disk Porto / DTW	26.6860	73.3140	0.01213	1.2166
Fréchet Porto (reference)	31.7614	68.2386	0.01551	-
Grid Porto / Fréchet	41.7700	58.2300	0.02391	1.5416
Disk Porto / Fréchet	34.4870	65.5129	0.01755	1.1315
DTW Rome (reference)	30.1179	69.8821	0.01437	-
Grid Rome / DTW	45.8131	54.1869	0.02818	1.9610
Disk Rome / DTW	46.2353	53.7647	0.02867	1.9951
Fréchet Rome (reference)	32.7586	67.2413	0.01624	-
Grid Rome / Fréchet	38.4629	61.5371	0.02083	1.2826
Disk Rome / Fréchet	40.2960	59.7040	0.02250	1.3855

on DTW and Fréchet than the grid-based clusters. The second observation that we remark is that both schemes in both cities have a higher rand index score for Fréchet-distance than DTW-distance. The rand index score results in addition to the Davies-Bouldin index results will be discussed in chapter 6.

Table 5.5: Rand index scores computed from the generated clustering outputs.

Rand index		
Similarity type / Reference	DTW	Fréchet
Grid Porto	0.9143	0.9324
Disk Porto	0.9262	0.9482
Grid Rome	0.9145	0.9391
Disk Rome	0.9202	0.9439

Chapter 6

Discussion

In this chapter, we will discuss the results and observations from the conducted experiments in the context of the expected behaviour of the schemes. We also discuss some of the choices made through this research and evaluate the schemes holistically. At the end of this chapter, we present some reflections and discuss the impact of the simplifications made in this research.

6.1 Effect of Parameter configuration

In this section, we first generally discuss the theoretical expectations we had for parameter configuration against the presented results, and include some discussions related to computation times. Then, with the general discussions in mind, we argue and reflect on the parameter values that was chosen for the remaining experiments of this thesis. Finally, we include some minor reflections on the choice of similarity measure for the adapted schemes.

We remark that the hashes from both the grid-based and the disk-based schemes can be seen as compressions of the original trajectories, where the compression ratio depends on the number of layers and the granularity of the grids and disks. From a theoretical perspective, a higher grid resolution (smaller tile width) will preserve more information through hashing, since trajectories' vertices will be snapped to a greater number of grid tiles, resulting in longer hashes and likely better correlation. The same theoretical perspective can be applied to the disk-based scheme, where we expected that a greater disk diameter are likely to decrease the hash compression ratio, since a trajectory will travel through a larger amount of disks, resulting in longer hashes and likely higher correlation. We note that the general observations presented in section 5.1 reflect these expectations, and that an increased amount of layers in both schemes have the expected effect on the correlation.

Following this derivation, we find it interesting that the grid-based scheme over Porto (Figure 5.1) has an apparent maximum correlation peak with a resolution value of 1.6 km for 5 layers and more. We expected the correlation to gradually decrease with poorer resolution, and for resolutions greater than 1.6km we note that the scheme follows this expected behaviour. Since we did not observe a similar prominent peak in the grid over Rome, we presume that the peak could potentially be explained by underlying patterns in the data-set in combination with a fixed grid size, but this still remains an open question to us. However, as a rule of thumb, we state that a higher resolution and a higher number of layers result in better correlation for the grid scheme.

We remark that the grid-based scheme are distorted by a random parameter t derived from the resolution. This is reflected by the standard deviation, which grows in line with the grid tile width. As the tile width increases (resolution lowers), the potential values t can take will also be larger. Larger values of t affects the similarities to a greater extent than smaller values of t, which results in varying similarities and higher deviations from one computation to another.

We also note that the expectation that a greater disk diameter and a larger number of disks in general results in higher correlation for the disk-based scheme, are reflected in the results given in section 5.1. However, as shown in the results, there is no clear connection between the number of layers in the disk-scheme over Porto until the disk diameter reaches 2.25 km. We presume that this could be explained by the unnatural high standard deviation for the same disk diameters, which we assume indicates that the disks are too small to cover and hash all trajectories in this data-set properly. This is further justified by the fact that for disk diameter values greater than 2.25 km, we observe significantly lower standard deviation values and a clear connection between the number of layers and the high correlation. In addition, the fact that diameters less than 1.5 km in Porto corresponds to extremely poor correlation values further justifies that the diameters less than 1.5 km is too small to cover the trajectories properly. We observe a similar, yet lighter, phenomenon for the Rome set, where the standard deviation drops significantly when the disk diameter is larger than 1.5 km. This phenomenon underpins the importance of choosing sufficient diameters, to ensure consistent and accurate correlation with the true similarities.

We remark that the disk scheme computes similarities by using the coordinates of the disks' centres. This implies that for a given number of disks, the correlation will converge when the disks' diameters increases. Trajectories at one point will begin to intersect disks that are too far away, meaning that the added disks no longer include valuable information to the hashes. This effect is reflected in the Figures 5.3 and 5.4, where the correlation curves flatten at 2.5 km diameter for Porto and 1.75 km for Rome.

By increasing the number of disks in the disk-based scheme, a greater geographical area can be covered by disks without increasing the diameter of the original disks. In theory, with a higher number of disks covering a greater area, the high standard deviation shown for disk diameters up to a certain size should be shifted towards lower diameters. In addition, a higher number of disks should also raise the convergence value slightly, since a denser disk-scheme means that trajectories have a higher probability of being snapped to disks that are closer to them, thus making the disk hashes more precise. We presume that this will have a positive effect on the correlation with the true values, and remark that our assumption is based on the results reflected in Figure 5.5, showing a higher correlation and lower standard deviation with an increasing number of disks.

From chapter 2, we restate that the time-complexity of DTW is $O(n^2)$, where n is the average length of the input hashes. Since DTW is the similarity measure used to compute the similarities between the schemes' hashes, the configuration of the schemes initially becomes a trade-off between accuracy and efficiency. As discussed in general, both a more fine-grained grid- and disk-scheme will increase the hash lengths, which inevitably entails higher similarity computation times. We remark that the extra information added to the hashes with a finer grid- or disk-scheme are multiplied for every layer, meaning that adding a layer is likely to have a higher run-time cost than tuning the granularity of grids and disks at one layer. Furthermore, with the quadratic computation time of DTW, longer and more detailed hashes quickly become costly. Finding the optimal configurations for one particular data-set with respect to accuracy and time-consumption would have been an interesting addition to this thesis, but was omitted due to time-constraints for this research.

Choice of parameters

The experiments related to evaluate the performance of the LSH-schemes (RQ2 and RQ3), required the parameter values to be set. As stated, the configuration of the schemes is initially a trade-off between accuracy and efficiency, where every extra added layer increases the accuracy of the scheme, but also raises the computation time. For the grid-based scheme over Porto, we chose a configuration of 5 layers with a tile width of 1.6 km, since 5 layers provided a significantly better correlation than 4 layers at the described correlation peak of Porto. We considered the possible correlation gain by adding more layers, but concluded that a correlation gain of barely a few hundredths (Figure 5.1) would be to small considering the additional computation time.

We applied the same argument to the choice of using 4 layers for the grid over Rome. Since the grid-scheme over Rome (Figure 5.2) did not have a distinct peak correlation value, we opted for a til width of 1.2 km, since the correlation with Fréchet distance had a hint of a peak at this resolution, which then dropped gradually after. Again, we remark that finding the optimal configurations would be an interesting addition to this thesis and for the rest of this work. Instead we opted for a close-to-optimal configuration for the rest of this research, which in general happened to be the largest grid tile width and the smallest number of layers with correlation close to the maximum correlation configuration.

For the disk-based scheme over Porto, we opted with 4 layers with 60 disks having 2.2 km in diameter, as the correlation was relatively high for this configuration. With respect to computation time, we chose 2.2 km despite the fairly high deviation for the chosen diameter. Instead of increasing the diameter, which would only give a small correlation gain due to the correlation being close to the converge value, we chose 60 disks, which would slightly increase the correlation but more importantly, as discussed, reduce the standard deviation (Figure 5.5a). We remark that a higher correlation and a greater reduction in standard deviation could be achieved by choosing a higher number of disks, but 60 was chosen as a trade-off with respect to computation time.

The same argument was applied to the disk-scheme over Rome, which was configured with 5 layers with 50 disks with a diameter of 1.6 km. We emphasised the correlation with Fréchet distance when choosing 5 layers, as it had significantly higher correlation than any lower number for the given diameter (Figure 5.4). In general for the disk-based schemes, we opted to use what we presume is the close-to-optimal configuration which happened to be the smallest disk diameter, the lowest number of disks and the lowest number of layers that were close to the correlation converge value.

By choosing the close-to-optimal, "working", configurations for both schemes and both cities, we presume that the remaining experiments and results related to performance are representative for the schemes. We still remark that by choosing other values, the results could be slightly different. However, our presumption is still that the results will be representative for the understanding of the schemes, which will be further discussed later in this chapter.

Choice of similarity measure

As a closing remark for this section, we remark that the originally proposed similarity measure, edit distance for strings, was opted out in favour of DTW, which is a more accurate distance measure for quantifying the similarity between the information stored in the hashes. This is reflected in the figures shown in Appendix A, where the correlation is significantly poorer than the corresponding hash-scheme figures in section 5.1. where DTW was used as similarity measure. Due to the shortcomings of edit distance for strings, this result correlated well with our expectations.

6.2 Performance analysis

In this section, we first discuss the run-time performance in order to compute the similarities from the schemes' hashes, as well as the time required to hash the trajectories. Then, the accuracy and effectiveness of the schemes will be discussed in detail, before we wrap this section off with some reflections regarding the consistency of the schemes.

6.2.1 Efficiency

Again, we remark that the run-times are a direct result of the schemes' parameter configuration, which must be kept in mind during this discussion. Since the schemes' hashes are compressions of the trajectories, we expected the schemes' similarity computation time to be significantly faster than the reference benchmark of DTW. However, the magnitude of speed-up was a bit surprising, but can be explained from a theoretical point of view. Due to the run-time of DTW of $O(n^2)$, two hashes with lengths of 1/10th of the original trajectories, should be computed 100 times faster than the original trajectories. Thus, the run-times also tell something about the compression ratio of the schemes, i.e. the information held by the hashes. In general, the speed-up of the schemes must be said to be more than satisfying for the purpose of similarity computation.

As noted in the chapter 5, the grid-based schemes have significantly lower computation times for both data-sets. This result must however be seen in context of the parameter configurations and the accuracy of the schemes, and will be further discussed under effectiveness below. With the current analysis and parameter configuration, it is not possible to directly compare the run-times of grid and disk similarity computation. However, an interesting possible future analysis would be to re-configure the schemes to have approximately the same correlation for a more fair comparison of run-times.

We assume that the contra-intuitive observation that the computation time of the disk-based Porto similarities is significantly slower than the disk-based similarity computation over Rome, could reveal a potential weakness in the disk-based scheme. By intuition, the Porto-disk-scheme should have faster computation time, since the Rome-disk-scheme is more complicated which is shown by the run-times of the true similarities. However, the high computation-time is explained by its configuration of 60 large disks of 2.2 km, which we observed was necessary to achieve an acceptable correlation score. Therefore, we state that a possible weakness of the disk-based scheme is that there is no guarantee that the schemes maintain the complexity of data-sets through hashing. We note that there are too many varying parameters for this observation to be conclusive, but it is still an interesting aspect. Nevertheless, the reduction in run-time is still extremely large.

Understanding the time consumption of the schemes is a vital part of RQ2. Since both schemes are based on the idea of generating compressed hashes from the trajectories, we expected them to scale better for larger data-set sizes. Again, we remark that both the true and hashed similarities are computed using the same algorithm with a time-complexity of $O(n^2)$. From a theoretical perspective, we therefore expect both schemes' time-reduction (compared to the benchmark) to follow an approximately constant value for the different data-set-sizes. This presumption is reflected in the results, where the grid and disk similarities of Porto are computed approximately 100 and 13 times faster than the benchmark values of the true similarities for the different data-set sizes. As a clarification, the presented approximated magnitudes of time reduction are derived from Figure 5.6

and not necessarily exactly accurate. However, the constant time-reduction factor is still clearly reflected in the figures, where the graphs have the same relationship for the different data-set sizes.

To finish the topic of similarity computation times, we remark that the the computation times of the true similarities for the largest sub-sets over both Porto and Rome have been interpolated and not measured. The reason was the exhaustive computation times of 10 parallel runs, which was practically very difficult to measure on a personal computer without influence from the OS. With the known time-complexity of DTW and roughly equal trajectory sizes, we believe that the interpolated graph displays timing values that are close to the correct values. The run-times of Fréchet distance was not computed for the same reason due to the higher time complexity of $0(n^2 log(n^2))$, meaning that the gain is theoretically significantly larger for this measure.

Hash generation efficiency

As expected, the overhead related to generating the grid hashes is very low for both data-sets and significantly faster than generating the disk hashes. The explanation lies in the fact that the generation of the grid hashes are solely based on pure arithmetic operations, while the generation of disk hashes involves finding which disks the trajectories travel through, which is a far more comprehensive task. From Table 5.1, we observed that by applying a KD-tree during hash-generation, the computation time can be reduced significantly. This was also the case when the quad-structure was applied, however with poorer time reduction than the KD-tree. We remark that strategies for faster hash generation does not really fall under the scope of this thesis, but have been included as a curiosity that was discovered during our research. Since KD-trees are designed for this type of problem, we believe that the gain from implementing the structure will be higher for larger number of disks, but we will not discuss the effect of the structures further. We remark that the hash generation times are influenced by the complexity of the data-sets, which is why see more exhaustive generation times for the Rome-set. All in all, there is still very low overhead in generating both the grid and disk hashes compared to their similarity computation time.

6.2.2 Effectiveness

An important aspect of this thesis is to understand how well the similarities computed from the schemes perform in terms of accuracy. This was measured in correlation with DTW and Fréchet distance, where DTW was chosen due to its popularity and Fréchet distance was chosen for its known precision as a metric. We remark that DTW is only a symmetric, and not a metric. The observation in subsection 5.2.2 that both the grid- and disk-based schemes had better visibly denser line-shapes and significantly better correlation with Fréchet distance than DTW, was therefore interesting but still expected with background in the results presented during parameter configuration. We note that this can be explained by the observed higher vertical spread of the shape in the 2d-histograms for DTW, which indicates that the same scheme distances can correspond to a greater interval of true similarity values. As stated, this is an interesting observation, but we will not discuss it further as it is not directly related to our research questions.

More importantly, we will discuss the observation that the disk-based similarities presented in Figures 5.9 and 5.10 tend to outperform the grid-based similarities. We restate that the histograms over the disk-based similarities are both visibly narrower and denser than the grid-based histograms, which is further reflected by the correlation values presented in the lower right corner of the histograms. We observe that the disk-based correlation values are clearly higher than the grid-based. By looking at the wider horizontal shape of the grid-based 2d-histograms, it becomes evident that this is the reason the disk-based schemes have higher correlations with the true similarities, since the true similarity values have a wider spread of corresponding grid-distances.

A potential explanation to the poorer accuracy that we see in Figures 5.7 and 5.8 could be the configuration of the schemes' parameters. We remark that the computation time of both the grid similarities were significantly faster than than the computation of the disk similarities for both data-sets. As stated earlier, a direct comparison between the schemes can not be justified, but as future work, it would be interesting to study the schemes accuracy by adjusting for computation time. However, since we chose parameters that apparently seemed to have relatively high correlation with the true similarities, the gain from this adjustment is likely limited without increasing the computation time drastically.

Another possible explanation to the wider horizontal spread in the grid-based histograms can be the different cost functions used by DTW in the grid-scheme and the disk-scheme. We remark that the grid-based schemes uses Manhattan distance and that the disk-based schemes uses euclidean distance. The fact that Manhattan distance is computed by the combined horizontal and vertical distance between two points, and not the shortest diagonal distance like the euclidean distance, implicates that different Manhattan distance values can correspond to the same euclidean value. Since both DTW and Fréchet distance are computed with euclidean distance, we find it likely that this effect could partially explain the wider spread.

All in all, by looking at the colour intensity of the 2d-histograms, we observe that the lsh generated similarities follow a clear correlation-pattern with the true similarities. Despite the grid similarities' generally weaker correlation pattern, the correlation with Fréchet is still at an acceptable level of around 0.65 and 0.67 for Porto and Rome which is still fairly high considering the reduction in computation time. In general, we presume that both the grid- and disk-based schemes are having sufficiently high accuracy for their similarities.

We restate that both the grid-based and disk-based schemes are based on random variables for hashing. In order to perform well, it is important that the schemes have sufficient correlation with the true similarity values, but also that they are capable of generating consistent similarity values. We have already discussed that the schemes are capable of delivering sufficient accuracy, and the results presented in Table 5.2 substantiates that the schemes are capable of producing similarities with a fairly consistent correlation with the true similarities. Even though the standard deviation mostly have a relatively low value around 0.01, we note that the relative difference between the maximum and minimum correlation values lie between 3.5% and 8.4% as shown in Table 6.1. Despite the fact that the values are small, they are still not insignificant for the accuracy of both schemes, which must therefore be said to be slightly fluid. We wrap up this section by stating that the schemes' standard deviation is a result of their configuration, and that the impact of the effect discussed in this paragraph can be lowered by a more detailed configuration.

Table 6.1: The relative difference between the maximum and minimum correlation values with the true similarities for both schemes over Porto and Rome. Computed with the formula: $\frac{Maximum-Minimum}{Average}$ using the values in Table 5.2

Correlation with true similarities for 10 parallel runs		
Similarity type / Reference	Std. deviation	Relative difference
Grid Porto / DTW	0.010	7.0%
Grid Porto / Fréchet	0.010	5.4%
Disk Porto / DTW	0.017	8.4%
Disk Porto / Fréchet	0.012	3.5%
Grid Rome / DTW	0.007	4.6%
Grid Rome / Fréchet	0.010	3.6%
Disk Rome / DTW	0.012	7.6%
Disk Rome / Fréchet	0.009	4.4%

6.3 Analysis of clusters

For the practical performance of the schemes, in terms of clustering, we will first discuss the results from the visual inspection against the previously discussed results in this chapter. Afterwards, the index analysis of the clustering outputs will be discussed in detail.

The most important result from the visual inspection is the general observation for both schemes that the disk-based clustering outputs are slightly more dense and less spread than the grid-based clusters. Furthermore, as stated in chapter 5, the disk-based clusters seem to preserve the direction of movement better, where the colour gradient of the clusters' trajectories move more holistically. With basis in the previously discussed higher correlation for the disk-based schemes, the fact that the disk-based clusters visually looked sharper than the grid based was expected. However, the fact that the disk-based schemes hardly were distinguishable from the true clustering outputs in terms of internal spread and direction of movement is promising in terms of practical performance. With argumentation in the observed correlation values, it was expected that the grid-based schemes would have more spread clusters than the disk-based. However, when comparing them to the true similarity clusters, they do not appear to be too far off in terms of spread and size.

The observation that the schemes clustering outputs appear to have more in common with the clustering outputs from Fréchet than DTW for both Porto and Rome (Figures 5.11 and 5.12), can also be explained by the corresponding correlation values, which we have seen was higher for Fréchet distance. By looking at the clustering outputs for the schemes and comparing them to the true valued clustering outputs, we believe that a partial explanation for the higher correlated Fréchet distance is the schemes ability to maintain the trajectories' direction of movement. We note that this is an inference based on our subjective visual inspection, but we still believe it is important to remark this when evaluating the practical performance of the schemes.

To fully understand the practical clustering performance of the schemes a discussion of the internal cluster evaluation index scores is necessary. As stated, the internal Davies-Bouldin indexes presented in Table 5.3, are computed using the schemes' similarities, and the computed indexes cannot be compared in a meaningful way. The reason is that the index is designed as a metric for evaluating the performance of clustering algorithms, and not the performance of varying similarities. We will therefore not discuss these values further, which was only included as a curiosity of internal evaluation which must be seen in the context of the corresponding clustering output only.

However, by combining the clustering output from the LSH-schemes with the true similarities, the Davies-Bouldin index can be computed and compared in a meaningful way, since the similarities used for computation are consistent (Table 5.4). We then see that the visual observation that the schemes' clustering outputs were slightly closer to Fréchet distance than DTW, are reflected in the adjusted relative Davies-Bouldin scores, which consistently were lower for Fréchet distance. This is naturally explained by the lower correlation with Fréchet. Nevertheless, and more importantly, the adjusted Davies-Bouldin index is never more than two times any of its reference values, and never more than 1.386 (Disk Rome / Frechet) for the best reference value of the schemes. Despite the lower grid correlation over Rome, the DB scores are slightly better for the grid-based similarities than the disk-based similarities, which show that the practical performance of the schemes not necessarily are fully related to the correlation.

When looking at the decomposed within-like and between-like factors of the Davies-Bouldin index (Table 5.4), it becomes evident that the schemes' clustering outputs are not as dense as the true similarities since the factors are slightly poorer. However, for the disk-scheme over Porto, we observe little loss in terms of within-cluster and between-cluster distances. On the other hand, with the limited data base with only two data-sets and two reference clustering outputs, we cannot state if any of the schemes are better suited for clustering, nor draw any other comparative conclusions. However, even though the within-like distances are slightly higher than the references, they are also a result of configuration, which can be adjusted.

Based on the discussed visual analysis and Davies-Bouldin index scores, we argue that both the schemes show satisfying practical performance. This is further supported by the rand index scores for the schemes, with scores well over 0.9, close to the maximum score of 1.0. In terms of practical performance, the scores mean that more than 90% of the trajectories are treated the same way as the reference clustering output for both the grid- and disk-based schemes over Porto and Rome. However, we still have to take into account that the rand index does not differentiate between trajectories that are agreed by chance and trajectories that are agreed correctly. We believe that this effect cannot be ignored, since the clusters from the schemes are visually fairly different from the true clustering outputs in terms of its content. On the other hand, we observe fairly different clusters between the two reference clustering outputs as well. The fact that the size and shape of the clusters are fairly similar underpins that the schemes are fairly suitable for a practical task like clustering.

6.4 Remarks and Reflections

In this section, we have included some thoughts and reflections regarding the performance and usability of the examined LSH-schemes.

As stated in several sections in this chapter, the configuration of the schemes are essential in terms of performance. However, the configuration also offers a trade-off between accuracy and efficiency, which can be utilised according to the needs of the user. By increasing the number of layers and the accuracy at each layer, the effectiveness of the schemes increases, but so does the time consumption during similarity generation. At several times in this thesis, we have noted that a direct comparison of the schemes accuracy are unfair due to their different configurations. It would therefore have been interesting to examine whether the grid-schemes' accuracy would have been closer to the disk-schemes' accuracy if the hash similarity computation times were adjusted for. A more fine-grained grid than we have used, would have led to an increase in computation-time, and we therefore presume that the different schemes' accuracy would have been closer if they were adjusted for computation time. The important knowledge from the configuration of the schemes is their ability to be tuned for accuracy or efficiency by the needs of the implementation.

We remark that an essential part of our argumentation that the true similarity measures are slow, is their computation time. However, both the grid- and disk-based schemes in this research have been implemented with the same DTW-algorithm used to compute the true similarities. As stated in subsection 6.2.1, both the hashed similarities and the true DTW similarities have a run-time of $O(n^2)$. We therefore note that the true run-time gain of the schemes are achieved through shorter input hashes instead of algorithmic differences. As a comment on the shorter hashes, we therefore note that the theoretical maximum time consumption of the grid hashes should be the same as the computation time of the true similarities multiplied by the number of layers. This is due to the fact that the compression of the grid scheme is given by the removal of consecutive duplicate tiles, but in a highly detailed grid, every trajectory point will be hashed to its own tile giving the output hash the same length as the input trajectory. Unfortunately, for the disk-based scheme, there are no theoretical maximum time consumption value other than the number of disks, with the reason being that a trajectory could be hashed to more disks than it has points. This effect will result in more exhaustive computation at every layer than computing the true DTW similarity once. Both the schemes could in the worst case end up with worse time consumption than the true similarities, which is important to be aware of when setting the parameter values of the schemes.

6.4.1 Simplifications

As stated in section 4.6, some simplifications were made during the research of the grid- and disk-based schemes. The simplifications and their assumed effect are shortly summarised in this section.

Firstly, we remark that the euclidean distance have been used for several tasks like computing distances between trajectory points, the computation of grids and disks in addition to the length of the trajectories. Due to the fact that all geographical points are located on a sphere, the values computed by the euclidean distance will be slightly inaccurate, and should for correctness have been computed using the haversine distance for spheres. However, due to the fact that the extracted data from both data-sources are limited to a tiny geographical rectangle of approximately 6 x 8 kilometres, we believe that the curvature of the earth can be ignored for all practical reasons in our research.

Furthermore, in the disk-based scheme, disks were included in trajectories' hashes if the euclidean distance between a disk's centre and a trajectory point was less than the given radius parameter. This was computed using the geographical coordinates of the disks and points, and the radius was recalculated to be the corresponding latitudinal distance. Due to the reason that the distances between longitudes become smaller further to the north, the disks will therefore have more of an elliptical shape than a perfect circle over Porto and Rome. At 40 degrees north, the distance between two longitudes are 85 kilometres, compared to approximately 111 kilometres at equator. This is a significant reduction, but for the purpose of examining whether the schemes are usable for similarity computation, we believe that it can be seen through, even though a more circular shape would likely entail better accuracy.

As stated in subsection 6.2.1, the computational load of measuring the true similarity computation time for the largest sub-sets was high and the process turned out to be very exhaustive. Therefore, as discussed, we opted to use the measured computation times for the smaller sub-sets and interpolate to find an approximation of the larger sub-sets. In terms of correctness, this approach is open to criticism since we are comparing it to the measured computation times of the schemes. However, we believe that the interpolated values are relatively close to the true values since the trajectories in the data-sets are of approximately the same lengths. We therefore believe that the run-times should follow a predictable curve, also for the largest sub-sets, and still be practically usable for comparison.

We remark that the test-sets of both Porto and Rome, which was used to determine the parameter configuration of the schemes (section 5.1), only consisted of 50 trajectories each. Due to the high computational load of computing the true similarities, we found it most efficient to determine the parameters using these test-sets. However, a drawback of using only 50 trajectories is that the computed correlation values are likely to be exposed to chance, since both schemes are based on random variables and that the selected 50 trajectories could not be representative for the entire data-set. From the simplifications made in this research, we believe that this simplification could be the most influential, which could be a reason for the slightly varying correlations presented in the 2d-histograms and the correlations presented in section 5.1. Despite this, we still believe the presented results of this research are highly valid.

We wrap up this chapter by noting that the general evaluation of the schemes are that they provides fairly accurate similarity values, under the assumption that the schemes are properly configured. Furthermore, the results have shown that the schemes are suitable for similarity computation and that they are usable for practical task like clustering.

Chapter 7

Conclusion

In this thesis, we have studied whether two locality-sensitive hashing schemes designed for trajectories are suitable for efficient similarity computation. In order to examine the main motivation of improving the exhaustive run-times of traditional trajectory similarity computation, we have experimentally evaluated the aspects accuracy, efficiency and time consumption of the schemes. Additionally, to evaluate whether the similarities computed by the schemes are usable for practical tasks, we generated clusters from the schemes' similarity values and evaluated the output.

The chosen evaluation criteria laid a solid foundation for the understanding of the schemes' performance, and the observed results gave interesting aspects for the discussion. We have summarised the conclusive results from this research in the context of this thesis' research questions:

The aim of RQ1 was to examine whether the LSH-schemes presented by Driemel and Silvestri [7] and Astefanoaei *et al.* [8] were adaptable to similarity computation between trajectories. We implemented the schemes as described in the papers and adjusted them slightly by changing their distance function to DTW. We found that the schemes were able to produce highly acceptable similarities for a variety of configurations, and with a proper configuration they were both able to deliver fairly strong accuracy when compared to DTW and Fréchet distance.

In terms of scaling, which was the main objective of RQ2, we have shown that the schemes are capable of reducing the computation time required by DTW a two digit number of times. For the grid-based scheme, we observed similarity computation up to 100 times faster than the true similarities. We also observed that this speed-up was consistent and consequent for the various data-sizes, which shows that both schemes are highly competitive in terms of scalability. With the scheme configurations used for this thesis, we observed that the grid-based schemes performed better in terms of efficiency, but poorer in terms of accuracy, but we remark that the configurations must be seen as a trade-off between accuracy and efficiency which left this observation open for further research to be conclusive. We have also observed that the overhead related to generating the schemes' hashes are very low, almost negligible when compared to the similarity computation times.

For RQ3, we have proven that the similarities are suitable for a practical task like clustering. We have seen that the generated clusters visually were almost as dense and tight as the true clustering outputs, The clustering index scores however were slightly more critical to the density of the clusters based on the schemes' similarities, but in general we evaluated the practical usability of the schemes' similarities to be satisfying.

In general, we have seen that the similarities generated by both the grid-based and disk-based schemes are usable, but with a somewhat reduction in accuracy compared to DTW and Fréchet, which can be controlled by their configuration. The schemes provides a significant speed-up orders of magnitude faster than the reference algorithms, and are also suitable for practical tasks. As a closing conclusive remark, which have been stated at numerous times through this thesis, we remark that the accuracy and efficiency of the schemes are a result of the configuration which can be adjusted according to the area of usage.

7.1 Further Work

In this research we have focused on examining whether the locality-sensitive schemes are a suitable faster alternative trajectory similarity computation. During this work, we have found several important aspects that we believe would be highly interesting to research further.

As mentioned, an important feature of the schemes are their configuration, and we believe that a more comprehensive study of the influence from these configurations would be rewarding. If we had time, we would have extended this research to include questions related the configurations. For instance, we would like to be able to answer questions related to the extra run-time cost of adding another layer to schemes.

Furthermore, a more comparative study between the two schemes would also be interesting topics to research. For instance, adjusting the grid- and disk-based schemes for run-time and comparing their correlation with true values was aspects that we wanted to examine, but fell outside the scope of our research questions. It would also make sense to include a comparative study of possible similarity measures that could be applied to the hashes in order to find an optimal algorithm. As we saw in this research, DTW happened to be far more precise than edit-distance, but we did not examine other alternatives.

We would also like to mention that the paper provided by Driemel and Silvestri [7] contained descriptions of other similar grid-based schemes, which would have been interesting to include in our study. The article by Astefanoaei *et al.* [8] also contained a more detailed disk-based scheme with different resolutions at different layers which also would have been interesting to include in a more comparative study.

Finally, we have evaluated the schemes on data-sets based on the movement of taxis which is connected to an underlying traffic-network. Even though the results were promising, we believe that it would be valuable to evaluate the schemes on other types of data as well.

Bibliography

[1] A. R. Mahmood, A. M. Aly, T. Kuznetsova, S. Basalamah and W. G. Aref, ‘Disk-based indexing of recent trajectories,’ *ACM Trans. Spatial Algorithms Syst. 4*, vol. 3, Sep. 2018.

[2] J. D. Mazimpaka and S. Timpf, ‘Trajectory data mining: A review of methods and applications,’ *Journal of Spatial Informatian Science*, no. 13, pp. 61–99, 2016. DOI: `http://dx.doi.org/10.5311/JOSIS.2016.13.263`.

[3] S. Dodge, R. Weibel and P. Laube, ‘Trajectory similarity analysis in movement parameter space,’ Apr. 2011. DOI: `10.5167/uzh-53069`.

[4] J. Gudmundsson, P. Laube and T. Wolle, ‘Movement patterns in spatio-temporal data,’ in *Encyclopedia of GIS*, S. Shekhar, H. Xiong and X. Zhou, Eds. Cham: Springer International Publishing, 2017, pp. 1362–1370, ISBN: 978-3-319-17885-1. DOI: `10.1007/978-3-319-17885-1_823`. [Online]. Available: `https://doi.org/10.1007/978-3-319-17885-1_823`.

[5] D. Yao, G. Cong, C. Zhang and J. Bi, ‘Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach,’ in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1358–1369. DOI: `10.1109/ICDE.2019.00123`.

[6] P. Indyk and R. Motwani, ‘Approximate nearest neighbors: Towards removing the curse of dimensionality,’ *Proceedings of the 30th Symposium on Theory of Computing*, pp. 604–613, May 1998.

[7] A. Driemel and F. Silvestri, ‘Locality-sensitive hashing of curves,’ *CoRR*, vol. abs/1703.04040, 2017. arXiv: `1703.04040`. [Online]. Available: `http://arxiv.org/abs/1703.04040`.

[8] M. Astefanoaei, P. Cesaretti, P. Katsikouli, M. Goswami and R. Sarkar, ‘Multi-resolution sketches and locality sensitive hashing for fast trajectory processing,’ Oct. 2018. DOI: `10.1145/3274895.3274943`.

[9] B. A. Flågen, ‘Utilising locality sensitive hasing for efficient trajectory similarity computation,’ Report, Norwegian University of Science and Technology, Dec. 2022.

[10] H. Su, S. Liu, B. Zheng, X. Zhou and K. Zheng, 'A survey of trajectory distance measures and performance evalutation,' *The VLDB Journal*, vol. 29, pp. 3–32, Jan. 2020. DOI: https://doi.org/10.1007/s00778-019-00574-9.

[11] Y. Zheng, 'Trajectory data mining: An overview,' *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 7, p. 41, May 2015.

[12] P. C. Besse, B. Guillouet, J.-M. Loubes and F. Royer, 'Review and perspective for distance-based clustering of vehicle trajectories,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3306–3317, 2016. DOI: 10.1109/TITS.2016.2547641.

[13] N. Magdy, M. A. Sakr, T. Mostafa and K. el-Bahnasy, 'Review on trajectory similarity measures,' in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec. 2015, pp. 613–619. DOI: 10.1109/IntelCIS.2015.7397286.

[14] K. Toohey and M. Duckham, 'Trajectory similarity measures,' *SIGSPATIAL Special*, vol. 7, pp. 43–50, May 2015. DOI: 10.1145/2782759.2782767.

[15] Y. Tao, A. Both, R. Silveira, K. Buchin, S. Sijben, R. Purves, P. Laube, D. Peng, K. Toohey and M. Duckham, 'A comparative analysis of trajectory similarity measures,' *GIScience & Remote Sensing*, vol. 58, pp. 1–27, Jun. 2021. DOI: 10.1080/15481603.2021.1908927.

[16] 'Dynamic time warping,' in *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84, ISBN: 978-3-540-74048-3. DOI: 10.1007/978-3-540-74048-3_4. [Online]. Available: https://doi.org/10.1007/978-3-540-74048-3_4.

[17] H. Alt and M. Godau, 'Computing the fréchet distance between two polygonal curves,' *Int. J. Comput. Geometry Appl.*, vol. 5, pp. 75–91, Mar. 1995. DOI: 10.1142/S0218195995000064.

[18] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler and J. Luo, 'Detecting commuting patterns by clustering subtrajectories,' vol. 21, Dec. 2008, pp. 644–655, ISBN: 978-3-540-92181-3. DOI: 10.1007/978-3-540-92182-0_57.

[19] T. Eiter and H. Mannila, 'Computing discrete fréchet distance,' Apr. 1994.

[20] J. Leskovec, A. Rajarman and J. D. Ullman, *Mining of Massive Datasets*, 3rd ed. Cambridge University Press, 2010, ch. 3.

[21] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam and C. Crushev, 'A survey on locality sensitive hashing algorithms and their applications,' *CoRR*, vol. abs/2102.08942, Feb. 2021. [Online]. Available: https://arxiv.org/abs/2102.08942.

[22] W. A. Barbakh, Y. Wu and C. Fyfe, 'Review of clustering algorithms,' in *Non-Standard Parameter Adaptation for Exploratory Data Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 7–28, ISBN: 978-3-642-04005-4. DOI: `10.1007/978-3-642-04005-4_2`. [Online]. Available: `https://doi.org/10.1007/978-3-642-04005-4_2`.

[23] F. Murtagh and P. Contreras, 'Algorithms for hierarchical clustering: An overview,' *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012. DOI: `https://doi.org/10.1002/widm.53`. eprint: `https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.53`. [Online]. Available: `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.53`.

[24] "Manimaran". 'Clustering evaluation strategies.' (May 2019), [Online]. Available: `https://towardsdatascience.com/clustering-evaluation-strategies-98a4006fcfc` (visited on 13/11/2022).

[25] W. M. Rand, 'Objective criteria for the evaluation of clustering methods,' *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. DOI: `10.1080/01621459.1971.10482356`. eprint: `https://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356`. [Online]. Available: `https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356`.

[26] D. Davies and D. Bouldin, 'A cluster separation measure,' *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-1, pp. 224–227, May 1979. DOI: `10.1109/TPAMI.1979.4766909`.

[27] P. Indyk, 'Approximate nearest neighbor algorithms for frechet distance via product metrics,' *SCG '02': Proceedings of the eighteenth annual symposium on Computational geometry*, pp. 102–106, Jun. 2002. [Online]. Available: `https://dl.acm.org/doi/pdf/10.1145/513400.513414`.

[28] M. Ceccarello, A. Driemel and F. Silvestri, 'FRESH: fréchet similarity with hashing,' *CoRR*, vol. abs/1809.02350, 2018. arXiv: `1809.02350`. [Online]. Available: `http://arxiv.org/abs/1809.02350`.

[29] Z. Aye, I. Sanchez, B. Rubinstein and R. Kotagiri, 'Fast trajectory clustering using hashing methods,' Jul. 2016. DOI: `10.1109/IJCNN.2016.7727674`.

[30] S. Salvador and P. Chan, 'Toward accurate dynamic time warping in linear time and space,' vol. 11, Jan. 2004, pp. 70–80.

[31] R. Wu and E. J. Keogh, 'Fastdtw is approximate and generally slower than the algorithm it approximates,' *CoRR*, vol. abs/2003.11246, 2020. arXiv: `2003.11246`. [Online]. Available: `https://arxiv.org/abs/2003.11246`.

[32] Z. Ruobing, G. Jiayi, H. Jianming and P. Xin, 'Deep trajectory similarity model: A fast method for trajectory similarity computation,' in *International Conference on Transportation and Development 2019*, pp. 13–23. DOI: 10.1061/9780784482582.002. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/9780784482582.002.

[33] D. Xie, F. Li and J. M. Phillips, 'Distributed trajectory similarity search,' *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1478–1489, Aug. 2017, ISSN: 2150-8097. DOI: 10.14778/3137628.3137655. [Online]. Available: https://doi.org/10.14778/3137628.3137655.

[34] M. Gowanlock and H. Casanova, 'Distance threshold similarity searches: Efficient trajectory indexing on the gpu,' *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2533–2545, 2016. DOI: 10.1109/TPDS.2015.2500896.

[35] B. Guillouet. 'Trajectory_distance.' (Apr. 2020), [Online]. Available: https://github.com/bguillouet/traj-dist (visited on 05/12/2022).

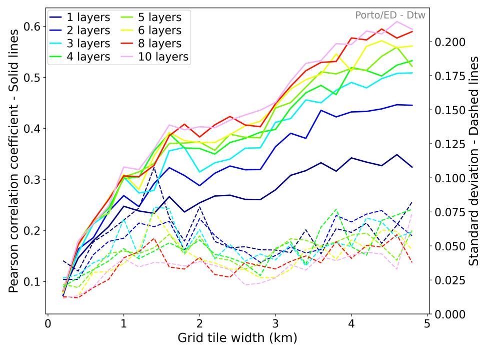
[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, 'Scikit-learn: Machine learning in Python,' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[37] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici and A. Rabuffi, *CRAWDAD dataset roma/taxi (v. 2014-07-17)*, Downloaded from https://crawdad.org/roma/taxi/20140717, Jul. 2014. DOI: 10.15783/C7QC7M.

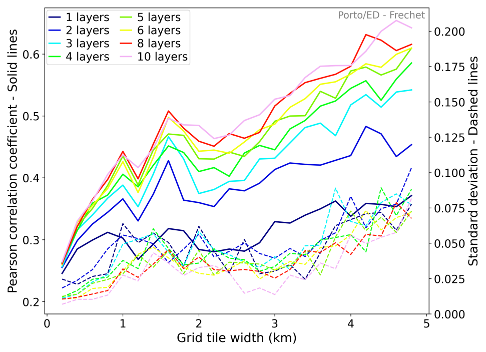
[38] *ECML/PKDD porto taxi dataset*. [Online]. Available: https://www.kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i.

Appendix A

Edit distance as hash similarity measure

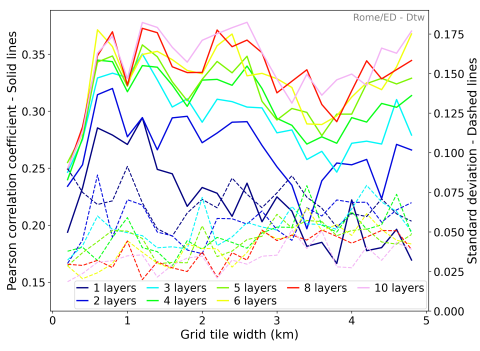


(a) Correlation with DTW

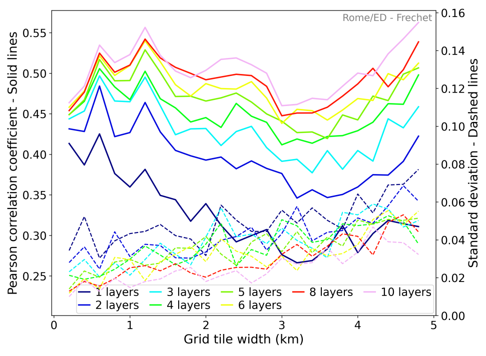


(b) Correlation with Fréchet

Figure A.1: Correlation with the true similarities for the grid similarities with varying grid resolution and number of layers. Edit distance used as similarity measure. Results from the test-set of Porto. Standard deviation marked with dashed lines.

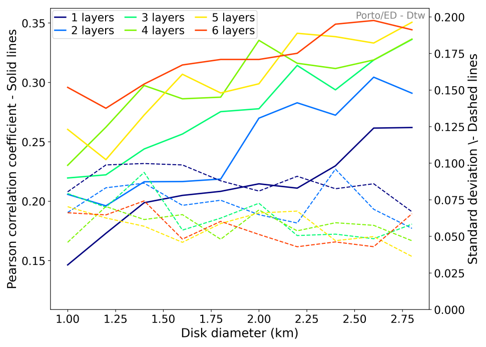


(a) Correlation with DTW

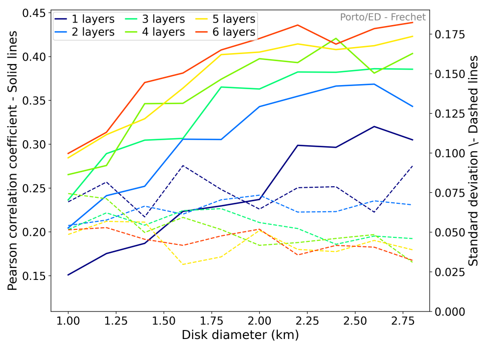


(b) Correlation with Fréchet

Figure A.2: Correlation with the true similarities for the grid similarities with varying grid resolution and number of layers. Edit distance used as similarity measure. Results from the test-set of Rome. Standard deviation marked with dashed lines.

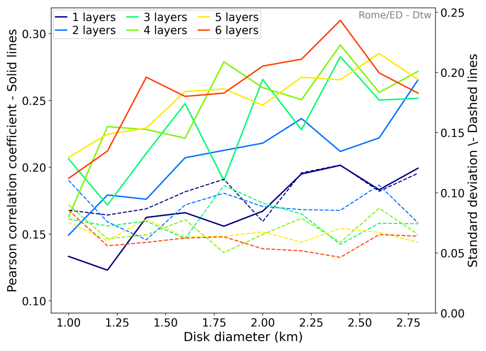


(a) Correlation with DTW

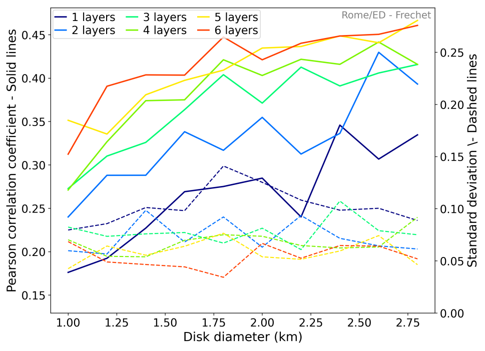


(b) Correlation with Fréchet

Figure A.3: Correlation with the true similarities for the disk similarities with varying disk diameter and number of layers, computed with 50 disks. Edit distance used as similarity measure. Results from the test-set of Porto. Standard deviation marked with dashed lines.



(a) Correlation with DTW



(b) Correlation with Fréchet

Figure A.4: Correlation with the true similarities for the disk similarities with varying disk diameter and number of layers, computed with 50 disks. Edit distance used as similarity measure. Results from the test-set of Rome. Standard deviation marked with dashed lines.

Appendix B

Clustering outputs

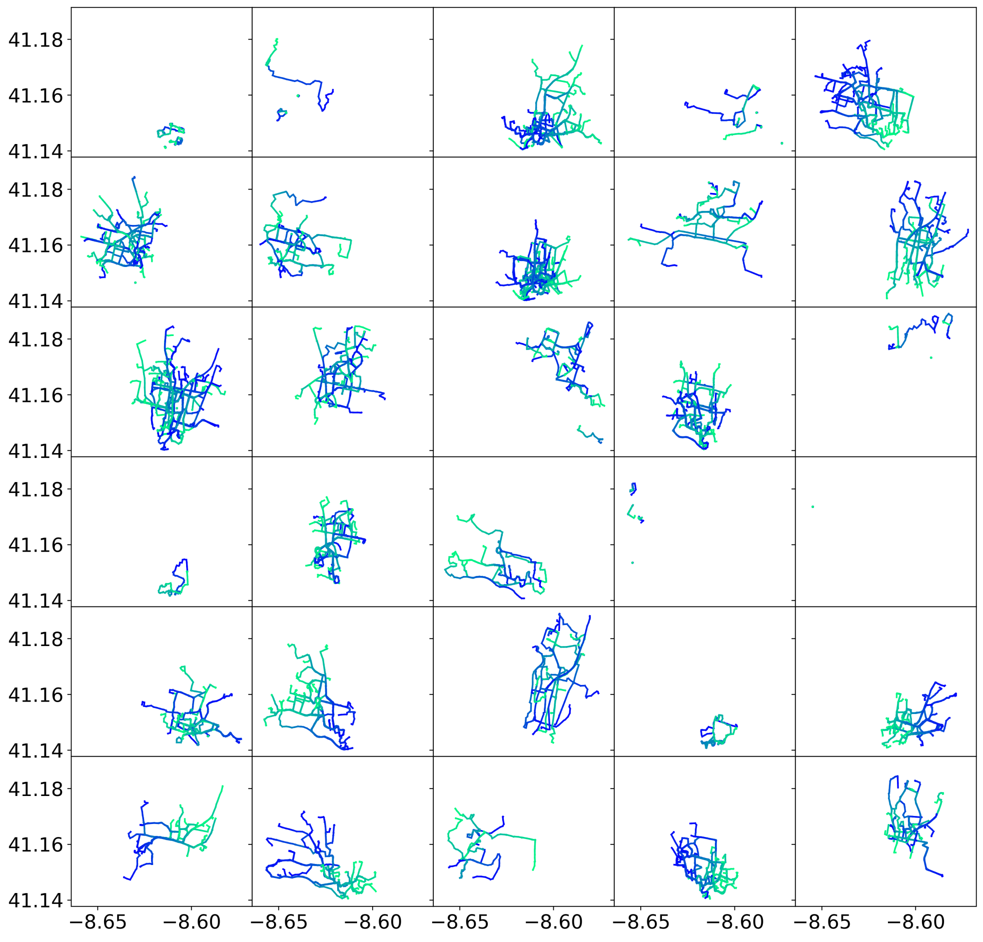


Figure B.1: Clustering output over Porto generated from DTW distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

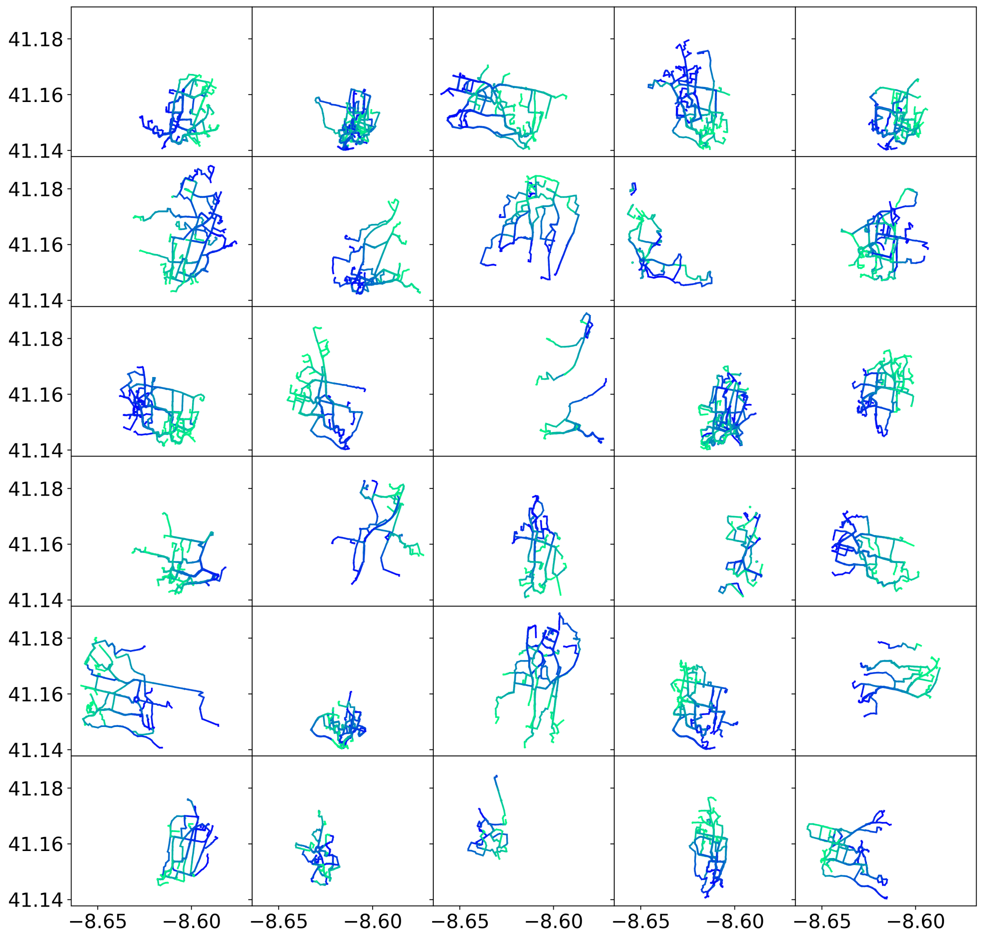


Figure B.2: Clustering output over Porto generated from Fréchet distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

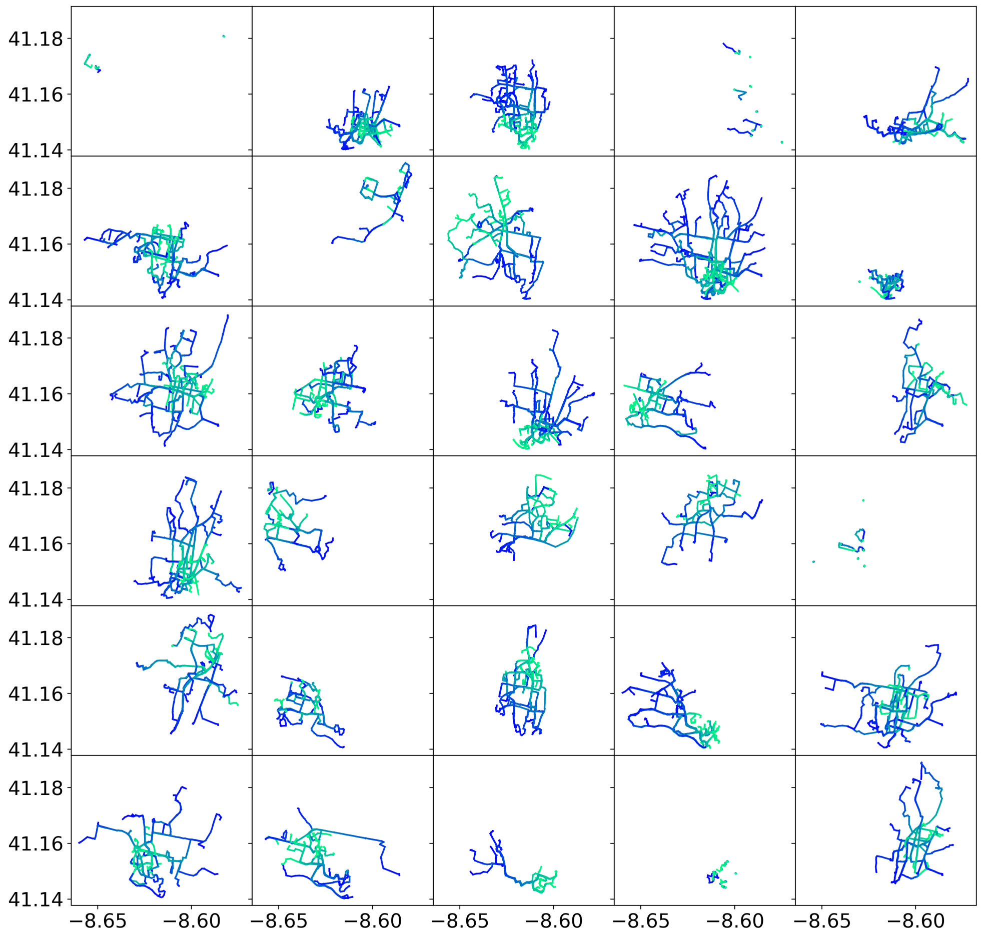


Figure B.3: Clustering output over Porto generated from Grid distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

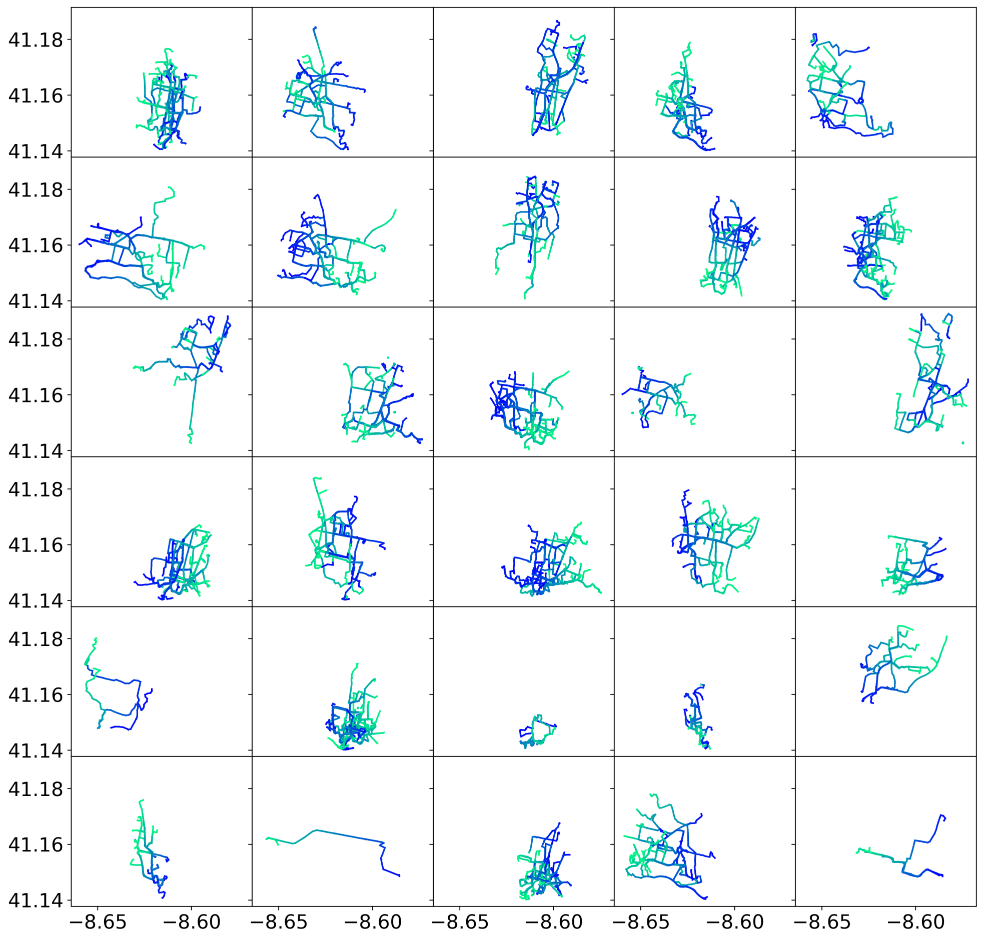


Figure B.4: Clustering output over Porto generated from Disk distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

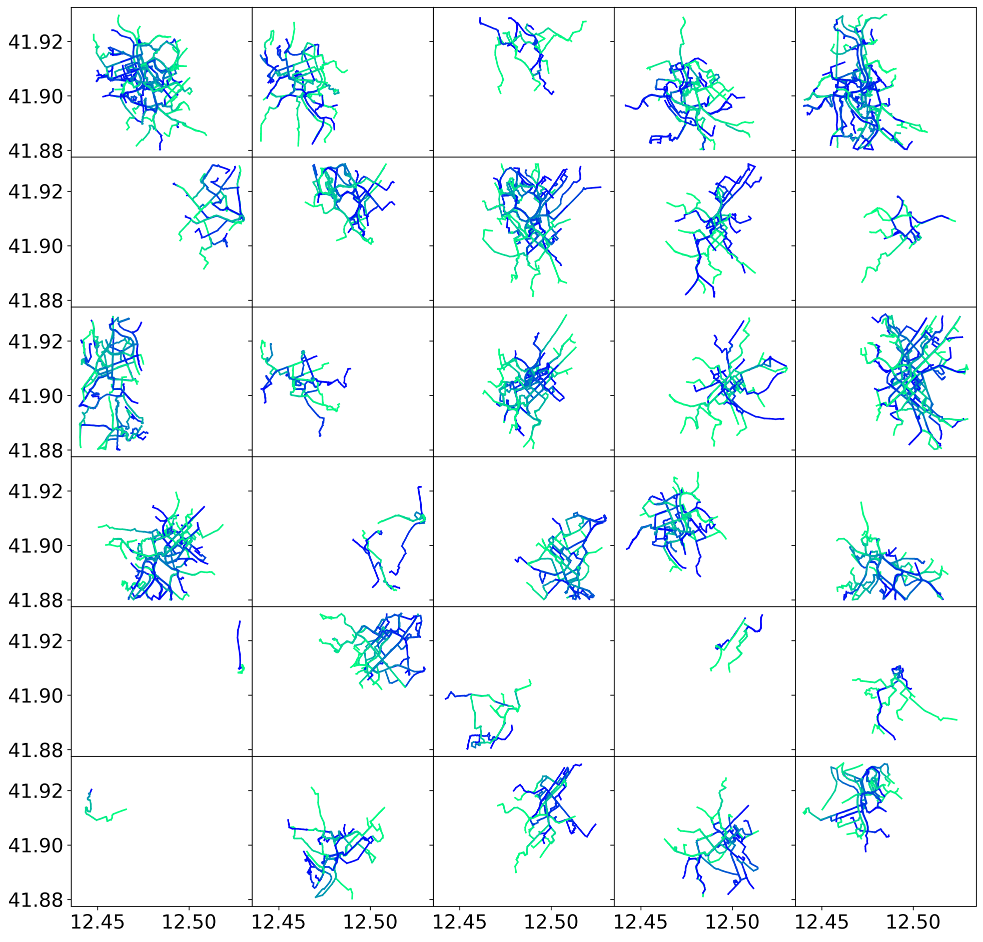


Figure B.5: Clustering output over Rome generated from DTW distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

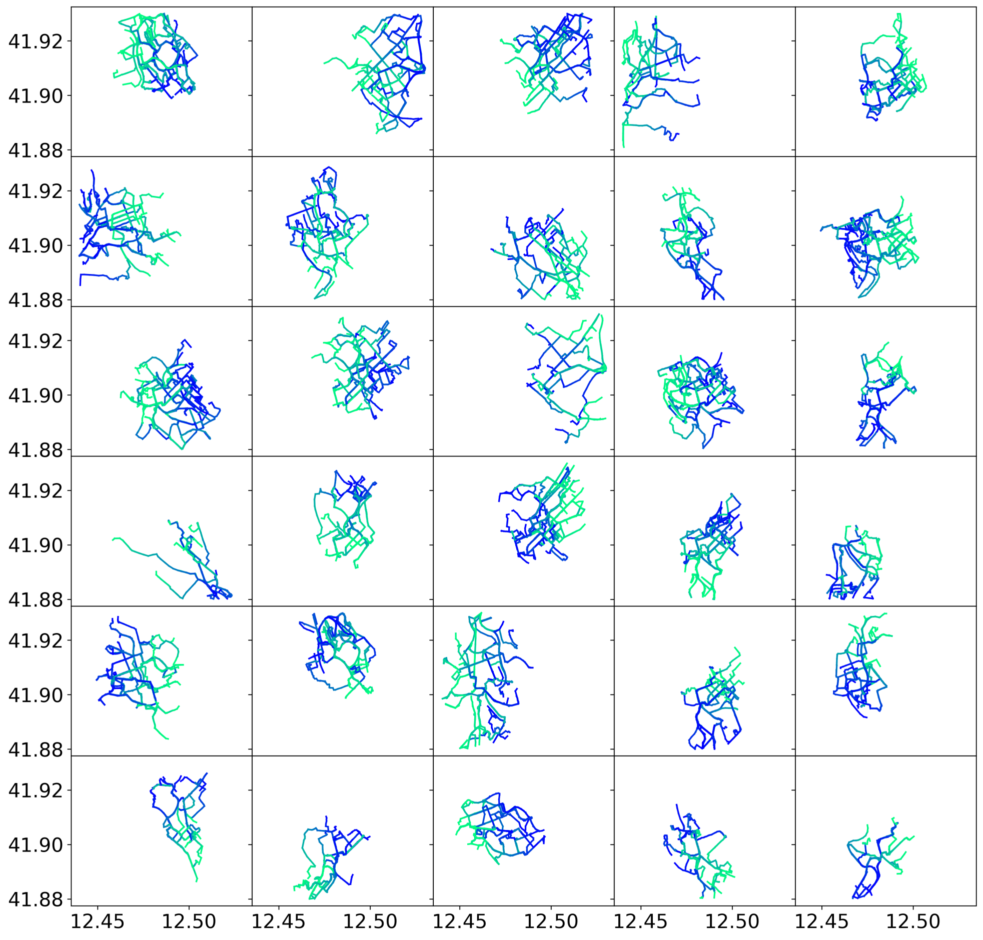


Figure B.6: Clustering output over Rome generated from Fréchet distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

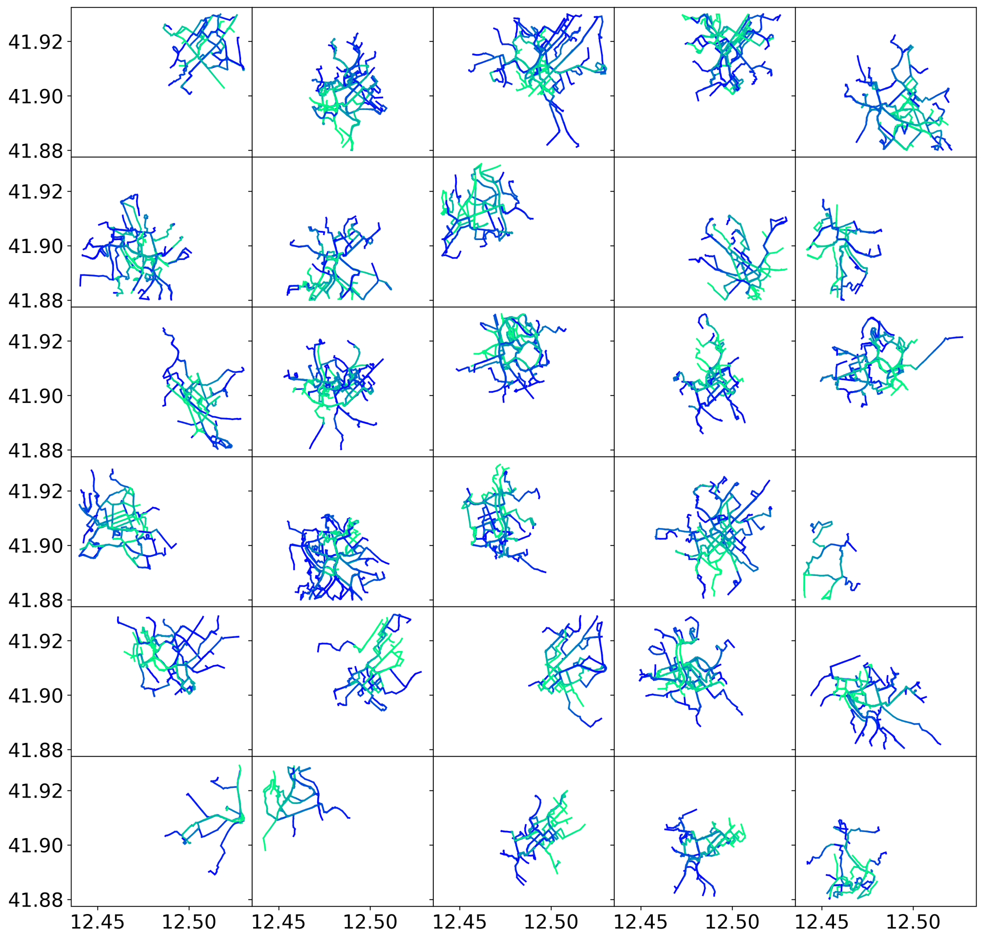


Figure B.7: Clustering output over Rome generated from Grid distance. The trajectories' colour gradient displays the movement direction from light to dark blue.

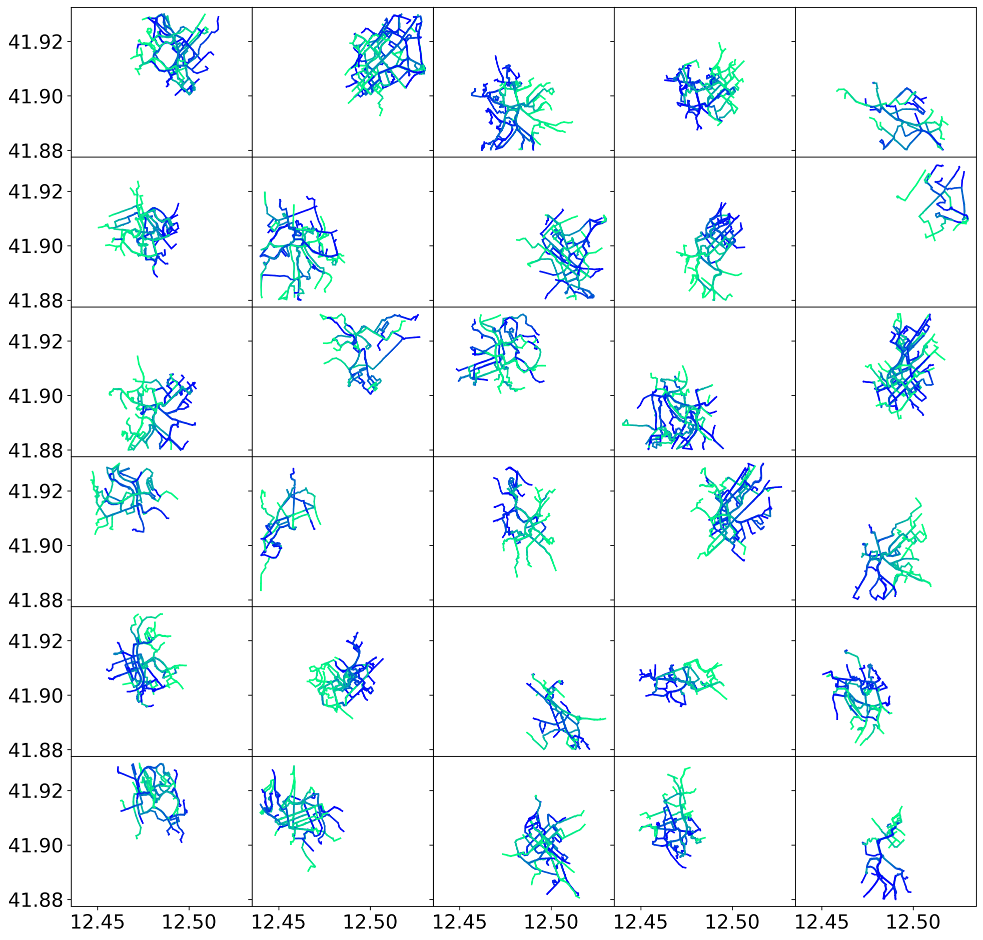


Figure B.8: Clustering output over Rome generated from Disk distance. The trajectories' colour gradient displays the movement direction from light to dark blue.