Vehicle Trajectory Similarity: Models, Methods, and Applications

RONIEL S. DE SOUSA, University of Ottawa, Canada and Federal University of Minas Gerais, Brazil
AZZEDINE BOUKERCHE, University of Ottawa, Canada
ANTONIO A. F. LOUREIRO, Federal University of Minas Gerais, Brazil

The increasing availability of vehicular trajectory data is at the core of smart mobility solutions. Such data offer us unprecedented information for the development of trajectory data mining-based applications. An essential task of trajectory analysis is the employment of efficient and accurate methods to compare trajectories. This work presents a systematic survey of vehicular trajectory similarity measures and provides a panorama of the research field. First, we show an overview of vehicle trajectory data, including the models and some preprocessing techniques. Then, we give a comprehensive review of methods to compare trajectories and their intrinsic properties. We classify the methods according to the trajectory representation and features such as metricity, computational complexity, and robustness to noise and local time shift. Last, we discuss the applications of vehicular trajectory similarity measures and some open research problems.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Information systems** → *Geographic information systems*;

Additional Key Words and Phrases: Datasets, mobility, trajectory, vehicle

ACM Reference format:
Roniel S. De Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. 2020. Vehicle Trajectory Similarity: Models, Methods, and Applications. *ACM Comput. Surv.* 53, 5, Article 94 (September 2020), 32 pages.
https://doi.org/10.1145/3406096

1 INTRODUCTION

Mobility aspects of society are changing rapidly. As the cities grow and their mobility requirements change, it is necessary to use new technologies and transportation modes to move people and other loads like goods and animals in a smart way, leading to smart transportation solutions [10]. Furthermore, the development of novel Information and Communication Technologies (ICTs) is leading to a rise in the availability of mobility datasets, which are acquired using several data

This work was partially supported by the NSERC CREATE TRANSIT, NSERC DIVA Strategic Research Network, and Canada Research Chairs Program. We thank the research agencies CAPES, CNPq, and Grants No. 15/24494-8 and No. 18/23064-8, São Paulo Research Foundation (FAPESP).
Authors' addresses: R. S. de Sousa, School of Electrical Engineering and Computer Science, University of Ottawa, K1N 6N5, Canada, and Department of Computer Science, Federal University of Minas Gerais, 31270-901, Brazil; emails: roniel.desousa@uottawa.ca, ronisds@dcc.ufmg.br; A. Boukerche, School of Electrical Engineering and Computer Science, University of Ottawa, K1N 6N5, Canada; email: boukerch@site.uottawa.ca; A. A. F. Loureiro, Department of Computer Science, Federal University of Minas Gerais, 31270-901, Brazil; email: loureiro@dcc.ufmg.br.

0360-0300/2020/09-ART94 $15.00
https://doi.org/10.1145/3406096

sources. For instance, people can record their real-world movements by carrying a mobile phone that generates spatial trajectories, logging their travel routes, and posting geotagged photos or making "check-ins" in social networks. Vehicles equipped with location acquisition devices can also generate a massive amount of trajectory data. Currently, the Global Positioning System (GPS), Europe's Galileo, and other Global Navigation Satellite Systems (GNSS) are the leading technologies to provide autonomous geospatial positioning [29].

Vehicle trajectory data offer us crucial information for applications in different research fields (e.g., urban planning, traffic analysis [24, 25], location-based social networks, data mining, and vehicular networks [26]). For instance, the analysis of popular routes is useful for both route recommendations and the city's road development planning. Furthermore, traffic monitoring systems can predict traffic congestion regions using historical data.

A fundamental task of trajectory analysis is the comparison of trajectories. Based on similarity measures, trajectories can be clustered, classified, and retrieved [69]. Currently, there are several methods to calculate the similarity (alternatively, the distance) between two or more trajectories, each one best suited for a particular scenario. For some data analysis tasks, such as clustering, choosing a proper similarity measure plays an important role [8]. Moreover, the representation of a trajectory influences the choice of the most indicated method.

A sequence of raw GPS data is the most common vehicle trajectory representation. Trajectories adopting this representation are called positioning-based trajectories. However, receivers of satellite-based positioning systems introduce errors to the recorded locations. These errors are more common in metropolitan regions due to low satellite visibility and phenomena that affect signal quality, such as strength attenuation and interference [55]. Therefore, preprocessing techniques need to be employed to reconstruct a road-network constrained trajectory [125]. For instance, map-matching is a technique to snap each trajectory sample to the road network, and filling the gaps aims at removing the uncertainty between consecutive trajectory samples. For this representation, there is another set of methods to compare the trajectories. In the remainder of this work, we adopt the terms *GPS-based trajectory* and *positioning-based vehicular trajectory* interchangeable.

The trajectory similarity research field has been in evidence in the last couple of years. However, most of the proposals in this area deal with the similarity of trajectories of objects of any kind (e.g., hurricanes, people, vehicles). Therefore, there is a lack of a systematic review that provides a detailed view of the field addressing the intrinsic characteristics of terrestrial vehicular trajectories, such as the geographic restriction of this type of trajectory to the physical definitions of the road network. To this end, this work presents a survey of the representations, methods, and applications of vehicle trajectory similarity.

We structure the remainder of this work as follows. Section 2 reviews the surveys on vehicle trajectory similarity. As discussed, most of the related work deals with the similarity of trajectories of any object, without focusing on the peculiarities of vehicular trajectories.

Currently, there are many representations of vehicular trajectories. Each representation has a set of characteristics that influences the choice of the most suitable similarity measures to be used with it. Section 3 presents the main representations as well as basic concepts of vehicular trajectories.

Section 4 presents a comprehensive review of the methods to compute the vehicle trajectory similarity. Among other characteristics, we classify the methods according to their representations, computational complexity, and capacity to deal with trajectories of different lengths, sampling rates, and speeds.

Finally, many applications benefit from comparing trajectories, but for each application, some similarity measures are more appropriate than others. Section 5 presents a review of the applications, discusses the trade-offs to consider when selecting a similarity measure, and provides some

Table 1. Surveys on Trajectory Similarity Measures

Reference	Methods covered	Goals
Wang et al. [113]	Methods for GPS-based trajectories: ED, DTW, PDTW, LCSS, ERP, and EDR	Evaluate the effectiveness of the measures and demonstrate their advantages and drawbacks through an experimental study
Magdy et al. [69]	Methods for GPS-based trajectories: ED, DTW, Time Warp edit distance, ERP, EDR, LCSS, Spatial Assembling Distance, Hausdorff distance, and Fréchet distance	Classifies methods concerning their computational cost, whether the measure is metric or not, the capacity to handle trajectories of different lengths, and robustness to deal with noise and local time shifting
Yu Zheng [125]	Methods for GPS-based trajectories: Closest-pair, DTW, LCSS, EDR, and ERP. Methods for road-network constrained trajectories: Minimum Bounding Rectangles distance and Trajectory-Hausdorff distance	Presents a brief discussion on the main methods to calculate the distance between trajectories, or between trajectories and GPS points
Toohey and Duckham [106]	Methods for GPS-based trajectories: LCSS, Fréchet distance, EDR, and ERP	Compares four methods through an experimental study and briefly discuss some applications
Besse et al. [12]	Methods for GPS-based trajectories: DTW, LCSS, EDR, and ERP	Reviews methods focusing on one application (trajectory clustering), and propose a new method called SSPD

selection guidelines. We discuss potential research directions in Section 6 and conclude this work in Section 7.

2 RELATED WORK

The majority of the reviews presented in the past few years on trajectory similarity focus on general GPS-based trajectories and their widely used similarity measures (i.e., ED, DTW, LCSS [111], ERP [21], and EDR [22]). They do not take into account the different set of trajectory representations and similarity measures for the case of vehicle trajectories. Besides that, these surveys do not deliberate about the applications and the suitability of the similarity measures. Table 1 summarizes these studies, their goals, and methods covered.

Wang et al. [113] presented an experimental study aiming at comparing popular trajectory distance functions (ED, DTW, PDTW [52], LCSS, ERP, and EDR). The focus of the study is on similarity measures for GPS-based trajectories. The purpose was to evaluate the effectiveness of the measures and demonstrate their advantages and drawbacks in different circumstances. To do this, they first created a dataset composed of two sets of trajectories. The first set is the *original trajectories*, while the second one, called *transformed trajectories*, is constructed by changing the sampling rates and adding noise or shift to the *original trajectories.* They generated the transformed trajectories in such a way that, if employing a suitable similarity measure, then the similarity between both sets of trajectories should be lower when the amount of transformation is higher, and it should be higher when the amount of transformation is lower. Wang et al. [113] concluded that each similarity measure deals better with a specific set of transformations. For instance, although ED is very sensitive to noise, it is a suitable method for situations where the compared trajectories are similar in terms of sampling interval and amount of point shift. DTW-based methods exhibited results similar to those of ED. Last, LCSS is sensitive to point shift modifications and robust to changes in the sampling rates and outliers.

Magdy et al. [69] presented another comparative study between trajectory similarity measures. However, unlike Wang et al. [113], Magdy et al. [69] focus on the characteristics of each method instead of presenting an experimental study. First, the authors classified each method evaluating if it considers both the temporal and spatial characteristics of the trajectory or just the spatial characteristics. For the Spatio-temporal similarity methods, a further classification level informs if it takes into account the vehicle movement speeds, or if it employs specific time series analysis techniques. Finally, they classified Spatial Similarity methods regard the use of the trajectories spatial data, geometric shape, or movement direction. Besides that, they compared the measures concerning their computational cost, whether the measure is metric or not, can handle trajectories of different lengths, and their robustness to deal with noise and local time-shifting. They compared measures such as ED, DTW, ERP, EDR, LCSS-based approaches, Spatial Assembling Distance, Hausdorff distance, Fréchet distance, and Trajectory Match Algorithm.

Yu Zheng [125] conducted a systematic survey on trajectory data mining to provide a comprehensive view of this research area. On the issue of trajectory similarity, the author addressed the main methods to estimate the similarity (alternatively, the distance) between trajectories. The article summarizes some well-known methods to compute the distance between GPS-based trajectories, such as closest-pair distance, DTW, LCSS-based, EDR, and ERP. Then it describes two measures of distance between trajectory segments, named Minimum Bounding Rectangles (MBR) distance and Trajectory-Hausdorff distance.

Toohey and Duckham [106] compared four trajectory distance functions: LCSS, Fréchet distance, DTW, and EDR. The authors coded the methods in a package using the R programming language and made the package available for free. The comparison between the measures was performed through experiments with a dataset of delivery drivers in the United Kingdom. The results showed a strong correlation between the values of the Fréchet distance and DTW, and between EDR and LCSS.

Besse et al. [12] tackled the issue of how to cluster GPS-based trajectories using the distances between them. They provided a review of four similarity measures (DTW, LCSS, EDR, and ERP). Also, they presented a novel distance function called Symmetrized Segment-Path Distance (SSPD). The authors implemented the five methods in a python package and made it available for free.

This survey differs from the studies mentioned above once our focus is to present and discuss trajectory analysis techniques regarding different trajectory representations. At the same time, the related work typically covers methods for raw GPS-based trajectories. This perspective is fundamental, since any investigation in this area should start with the trajectory representation and, then, proceed in the study of a particular research issue but considering all aspects defined for that representation. Thus, for each representation, we provide a discussion regarding the state of the art of the corresponding methods. Finally, we highlight and discuss various applications that depend on similarity measures. In this way, we advance the understanding of this research area.

3 BASIC CONCEPTS AND REPRESENTATIONS

A trajectory is a trace generated by the movement of some entity (e.g., person or vehicle) over a specific time window. Although the movement is continuous, traces are usually represented by a sample of the object's real movement due to limitations of location positioning devices [113], so it is not always possible to have a trajectory representation that fully captures the movement of the object. In the context of vehicles, the most common representations of trajectories are *GPS-based trajectories* and *Road-network constrained trajectories*. Data representation formats are closely related to distance measures and thus are intended to support similarity search and a particular set of data mining tasks.

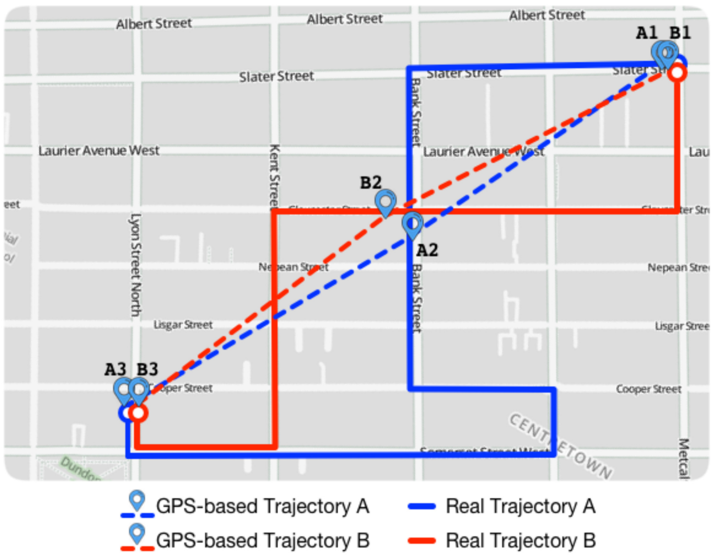


Fig. 1. Two different GPS-based trajectories that look similar because of the gaps in the data.

3.1 GPS-based Trajectories

A GPS-based trajectory, also called a *raw trajectory*, is a trajectory representation directly obtained from the logs generated by GPS-equipped devices. In general, a GPS-based trajectory $T_{gps} = (\rho_1, \rho_2, \ldots, \rho_n)$ is a set of sampling points $\rho_i = (\rho_i.lng, \rho_i.lat, \rho_i.t)$ ordered by timestamps, where $\rho_i.lng$, $\rho_i.lat$, and $\rho_i.t$ represent the longitude, latitude, and timestamp of the sampling point, respectively. Besides that, the trajectory may have other attributes associated with each point, such as the speed and heading of the vehicle.

Most similarity-based applications of vehicle trajectories take into account only the time and location components of the trajectories. For instance, spatiotemporal databases use the time information of each GPS-based trajectory to respond to questions such as *What are the trajectories within a given region* R*, between days* X *and* Y*?* However, for analysis applications such as similarity-based retrieval, we are interested in the amount of time a vehicle took to move from a point to the next one, instead of when precisely each trajectory occurred. Therefore, most of the methods and applications discussed in this work focus on the latter kind of similarity.

A problem of raw trajectories is the trajectory uncertainty, which occurs because positioning-devices record data at discrete time intervals and, therefore, we do not know the real movement of a vehicle between a trajectory point and the following one. For example, when a vehicle is moving at 60 km/h (1 km/min) and reporting its location every 30 s, there is a movement gap of 500 m, which means that the actual position of the vehicle is unknown for most of this time interval. Most trajectory datasets contain such gaps, since location-acquisition devices usually record the positions with a sampling interval of at least 30 s to reduce communication loads and storage costs [9, 15]. On the one hand, some applications need to increase even more the uncertainty of a trajectory to protect the user's privacy. On the other hand, the uncertainty in trajectories hinders various data mining tasks, such as calculating the distance between them. Figure 1 illustrates two different trajectories (blue: A1-A2-A3; red: B1-B2-B3) that look similar if we take into consideration only the points of their GPS-based representations. This misreading happens because of the gaps generated by a low-sampling rate. Thus, many studies attempt to address the uncertainty of trajectories by reducing or removing the gaps, modeling the uncertainty of trajectories, or inferring the actual path of the trajectory. We discuss some of these methods below.

3.1.1 Filling the Gaps. Interpolation is the most straightforward idea to remove the gaps in vehicular trajectories. For instance, assuming that the vehicle moves straight from a sample to another, we can apply linear interpolation [108] to represent the unknown movement. However, in most situations, this technique is ineffective, because it is common for vehicles to follow curved paths. A better option is to use the road network to find the shortest path between consecutive points and thereby improve the interpolation [66]. Other advanced methods produce more accurate results by using historical trajectory data from the same dataset of the analyzed trajectory [9, 15, 101]. Su et al. [101] and Celes et al. [15] proposed an approach composed of a reference system and a calibration method. They use anchor points from historical GPS datasets to construct the reference system, and then the calibration method finds trajectory data in the reference system to fill the gaps. Bedogni et al. [9] proposed a methodology to derive entire trajectories of individual vehicles considering sparse and inaccurate samplings. Using that approach, it is possible to reconstruct a given trajectory.

A further step would be to use the geographic constraints of the road network so as not to insert points outside of the roads. Recent approaches use probabilistic models to verify all possible sequences of modal activities (e.g., acceleration and deceleration) between the trajectory points and then reconstruct a trajectory that has one sample per second [41, 112].

3.1.2 Modeling the Uncertainty of Trajectories. Another approach is to model the uncertainty of trajectories and incorporate it into the trajectory representation. In the past few years, many authors studied Markov chains [31, 79, 88, 122] and other stochastic models [85, 109, 110] to represent better the uncertain positions of moving objects. Pfoser and Jensen [85] proposed a technique that first obtains the positions in-between the sampling points by using interpolation, and then, computes two measurement errors, one about each position in time, and a global worst-case error. They assume that the distribution of the measurement error of GPS points is Gaussian. Besides that, they proposed a trajectory representation that includes the movements as linearly interpolated positions and the parameters of the error distributions associated with the movements. One drawback of this approach is that it uses linear interpolation, which is not accurate to represent vehicle movement.

Trajcevski et al. [110] proposed a trajectory uncertainty model based on ideas similar to the method presented by Pfoser and Jensen [85]. However, their uncertainty model is a cylindrical volume in 3D, in contrast with the model give by Pfoser and Jensen [85], which is an ellipse where the endpoints of the segments represent its foci. However, like the trajectory model of Pfoser and Jenson [85], Trajcevski et al. [110] represent a trajectory as a three-dimensional polyline. In other words, it uses linear interpolation and therefore assumes that the movement of the entity between consecutive sample points follows a straight line at a constant speed. Trajcevski et al. [110] described an uncertainty model based on rectangles intended for scenarios in which we know that the object has a movement restricted to a road network, but its correct lane is unknown. However, they did not examine this method in their paper.

Trajcevski et al. [109] evaluated another model for trajectory uncertainty, which uses the concepts introduced in Reference [43]. In their model, the so-called *beads* bounds the possible whereabouts of a given object between the trajectory sampling points, and the *necklace*, which is a sequence of beads, represents the trajectory. This model has many properties equivalently to the one proposed by Pfoser and Jensen [85]. The model assumes that the only thing known about the moving object in-between two consecutive locations points is that there is a threshold that bounds its maximum speed. Trajcevski et al. [109] demonstrated through experiments that the

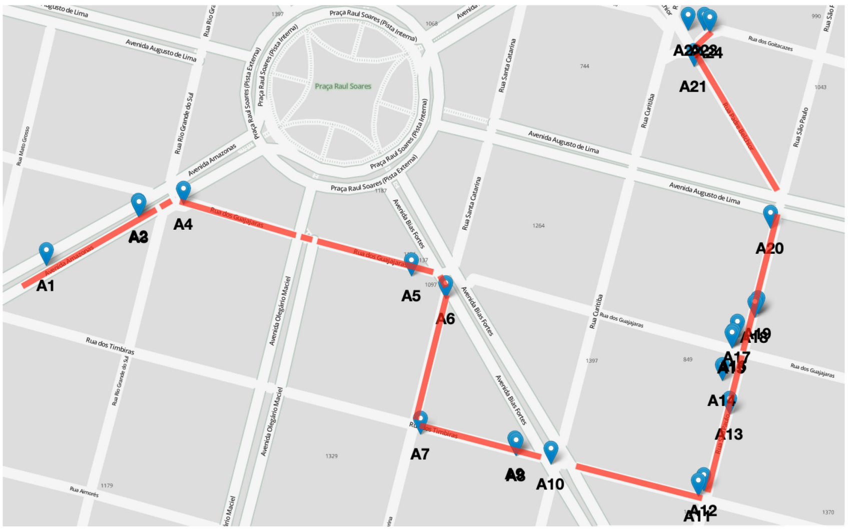


Fig. 2. A GPS-based trajectory (pins on the map) and its representation as a road-network constrained trajectory (red lines). Unlike the GPS-based trajectory, the road-network constrained trajectory does not have positional errors.

beads/necklaces model could be used to improve queries of certain types by using pruning strategies.

Lu et al. [67] proposed to reconstruct a complete trajectory of a mobile object given its partial trajectories recorded in different transportation modes (e.g., subway, car, walk). Their solution, called trajectory splicing, must satisfy some expected requirements: disjoint time overlap of the partial trajectories (i.e., they cannot have a time overlap if they refer to the same object); spatial proximity of a sequence of partial trajectories; and a set of partial trajectories should not be contained in a larger set of trajectories. Trajectory splicing helps to build a complete trajectory when there is uncertainty about the different partial trajectories that might comprise the original one.

3.2 Road-network Constrained Trajectories

Vehicular trajectories have an essential feature that the trajectories of most of the other kinds of moving objects do not have: are physically constrained to the road network. Because of this, we can map vehicular trajectories to the roads respecting the turn restrictions, and thus represent the movement of the vehicle better. We call this type of representation a road-network constrained trajectory. When properly constructed, a road-network constrained trajectory is more concise and precise than the GPS-based trajectory (raw trajectory). Figure 2 illustrates a road-network constrained trajectory in comparison with a GPS-based trajectory.

Road-network constrained trajectories are generally constructed from GPS-based trajectories. However, GPS-based trajectories contain errors such as noise and outliers, as satellite-based positioning systems do not always produce accurate data. These errors mainly happen because of poor satellite visibility in urban regions, which results in the occurrence of phenomena such as reflection and diffraction [55]. Therefore, some preprocessing techniques (e.g., noise filtering and map matching) are employed to reconstruct a road-network constrained trajectory [125].

In recent years, many authors proposed models for road networks, and consequently, for road-network constrained trajectories. For instance, we can use cubic spline interpolation to model the curved shapes of road networks [5]. However, the most common representation of a road-network constrained trajectory, which applies concepts of graph theory, is more straightforward than that.

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be a directed graph that represents the road network $\mathcal{N}$, where the set of nodes $\mathcal{V}$ represents the road intersections, and the set of arcs $\mathcal{E}$ represents the roads (for simplicity, we use the term road as a synonymous of road segment). Each arc $e \in \mathcal{E}$ connects a start node $e.from \in \mathcal{V}$ to an end node $e.to \in \mathcal{V}$, and each node $v \in \mathcal{V}$ is endowed with its position $(v.longitude, v.latitude)$ on Earth. Besides that, the road network graph can have additional information, such as the maximum allowed speed and number of lanes in a road.

A road-network constrained trajectory $\Gamma = (t_1, t_2, \ldots, t_n)$ is a ordered set of connected roads that represents a path traveled by a vehicle. That is, $\forall t_i \in (t_1, t_2, \ldots, t_{n-1})$, $t_i \in \mathcal{E}$, and $t_i.to$ must be equal to $t_{i+1}.from$. Besides that, $t_1.from$ and $t_n.to$ denotes the first and last position of the vehicle, respectively. Additional data can improve this representation. For instance, the trajectory data can also include the offsets that describe the vehicle's initial position at the first point t_1 and final position at the last point t_n, the times of the beginning and end of the trajectory, and the mean speeds of the vehicle in each road.

Another representation focus on the position of the vehicle relative to the route (trajectory) instead of the road segments in the road network [37, 38]. The most practical motivation for this representation is that the trajectory becomes much smaller, because by using it, we need to store new positions only when the vehicle moves to another highway. The former model relates the vehicle locations to the edges, so we need to store a new position at every exit/junction of the road network. One drawback of this model is that it makes difficult the operations, query processing, and optimization strategies, including indexing.

3.3 Other Trajectory Representations

Trajectories datasets are usually large, which can bring challenges to data analysis tasks when using representations like GPS-based trajectories or road-network constrained trajectories. Therefore, in recent years, some authors have proposed dimensionality reduction, hashing, and other codifications to represent trajectories better and process large scale datasets [16, 61, 86, 96].

3.3.1 Dimensionality Reduction. Road-network constrained trajectories have, by default, a lower dimension when compared with GPS-based trajectories. This difference is because the positions of the former are of only one dimension (i.e., the identifier of the road segment), as opposed to the 2D geospatial coordinates of raw trajectories. Instead of assigning random identifiers to the roads, one can use the Hilbert curve [42] to have a notion of space ordering of the road segments, which can facilitate queries and indexing. This is the idea behind the method proposed by Pfoser and Jensen [86] to reduce the dimensionality of trajectories. First, they sort all network edges based on the Hilbert value of the middle point of the edge. Then, they map all edges into sub-internals according to their ordering. The first edge becomes the first sub-interval, which extends a distance that corresponds to its distance in the road network. The second edge then starts where the first ends, and so on. Finally, they map the trajectories from polylines to this new movement space by identifying the edge where the vehicle was and how much of this edge the vehicle traveled. This method can reduce the size of the trajectories as well as the dead space in the indexes. However, a problem not discussed by the authors is how to map the movements of the vehicles to the road network.

3.3.2 Binary-Encoded Trajectories. We usually express locations in real-world road networks by using hierarchical administrative districts, such as city, road name, and road number. This perception of the hierarchy is not taken into account when we use space-filling curves to represent trajectories. Lee et al. [61] proposed a location encoding method that converts the positions into a binary string and has the notion of hierarchy. Their approach consists of three steps. The first step retrieves the address of the vehicle's location and expresses it as a triplet (district, road, and location on the road). Then, the second step converts the tree address fields into three binary strings, and finally, the last step joins the binary strings. They use R-tree [39] to find the closest road segments to each position and get their address, and then, the Z-order curve [87] to encode the address as binary strings. One drawback of this representation is that the trajectory data size can be large, as all positions are encoded and concatenated without any compression. Equation (1) illustrates a 16-bits binary-encoded location consisting of four components, each one of four bits. When using this representation, for each scenario, one needs to carefully design the size of each component of the string so as not to waste memory. For instance, all road segments should identify the same amount of locations, and thus, we need to divide the longer segments into smaller ones so that every segment has similar sizes:

$$\text{Binary-Encoded Location} = \underbrace{0100}_{\text{Country}}\ \underbrace{0111}_{\text{District}}\ \underbrace{1001}_{\text{Road}}\ \underbrace{0010}_{\text{Location on Road}}. \tag{1}$$

3.3.3 Hash-based Trajectories. Geohash is a widely used public domain latitude/longitude geocoding system invented by Gustavo Niemeyer [80]. In summary, a geohash is a hierarchical spatial data system that uses the z-order space-filling curve to map a point to a binary string. It works by recursively dividing the space in half by straight lines, both horizontally and vertically, until the squares bounded by the lines are small enough. The number of divisions performed defines the accuracy of the geohash. Then, one can convert the sequence of bits to an alphanumeric string by using a variant of the base 32 transfer encoding. Among other features, geohashes allow precision control, its prefixes for nearby positions are similar, and one can remove some characters from the end of the geohash to save storage space. Because of its properties, recent proposals use geohash to encode trajectory data [7, 16].

Chapuis and Garbinato [16] proposed a method called geodab that combines hashing and geohashing [80] to represent trajectories in a way that facilitates the processing of dense trajectory datasets. A geodab is constructed from a sequence of points (i.e., from a GPS-based trajectory) as follows. First, it computes a prefix and a suffix for the geodab. The prefix is the geohash of the area that contains all points of the trajectory, and its objective is to distribute different geodabs on the Z-order curve [87] taking into account the sample locations. The suffix is another hash, and it aims to reflect the ordering of the points. Finally, the hash and geohash are concatenated and encoded as a 32 bits hash. The length of the hash can change according to the application requirements. As it includes both hash and geohash, the 32-bits hash that represents the geodab can distinguish the trajectories considering both their paths and points ordering. The authors [16] demonstrated how geodabs and a fingerprinting algorithm, called winnowing [98], can index trajectory datasets. Finally, they presented a scalable method to distribute indexes of trajectories across different clusters.

3.3.4 Application-based Trajectories. For specific scenarios, it is better to design the trajectory representation according to the application. Tiesyte and Jensen [105] proposed a representation that is intended, for example, to compare trajectories of collective transport. Their model assumes that the vehicle locations are pre-defined and only takes into account the temporal component of the trajectory. In other words, a trajectory is a function that receives as input a location and provides as output the time the vehicle took to reach this location. Besides that, this representation allows the construction of predictive similarity measures, that is, similarity measures that assume that past similar trajectories are also similar in the future.

3.3.5 Deep Representation Learning. Representation learning is a process that transforms input data to facilitate analysis tasks such as building predictors. The idea is that the new representation has additional properties and preserves most information from the original data. We call deep representation learning the techniques composed by a set of non-linear transformations [11].

Recently, many studies proposed the application of representation learning for trajectories [35, 64, 119]. For instance, Li et al. [64] introduced a novel deep learning-based approach for trajectory representation. Their method, called t2vec, can perform similarity computations of low-quality data accurately and efficiently. The t2vec uses historical GPS-trajectory information and deep learning techniques to infer and describe the route information of a trajectory. After processing the trajectories to learn their representations, the time complexity to calculate the similarity between them is linear. In their experiments, the t2vec framework showed to be more accurate and efficient than the similarity measures EDR [22], LCSS [111], and EDwP [90]. Trajectory Embedding via road networks (Trembr) [35] is another deep learning-based approach for trajectory representation. The main difference between Trembr and other related methods (e.g., t2vec) is that Trembr incorporates the underlying road network. To do this, they use a map-matching technique to map the sample points of raw trajectories onto the road network, which then facilitates and constrains the learning process.

4 METHODS

Several methods are available to measure the similarity of vehicle trajectories. Each method has a specific set of properties. We present the properties that trajectory similarity measures can have in Section 4.1. Furthermore, the most proper method depends on how we represent trajectories. In Section 4.2, we present a comprehensive review of the methods to compare GPS-based trajectories and, in Section 4.3, we review the similarity measures for road-network constrained trajectories.

4.1 Properties

Metricity. We can classify similarity measures according to whether or not it is a metric. The advantage of being a metric is that the measure can be indexed directly by known distance-based indexing techniques. We call a distance function $D(A, B)$ (associated with a given similarity measure) as a metric if, given trajectories A, B, and C, it satisfies the following conditions:

- Uniqueness: $D(A, B) = 0 \Leftrightarrow A = B$
- Nonnegativity: $D(A, B) \geq 0$
- Symmetry: $D(A, B) = D(B, A)$
- Triangle Inequality: $D(B, C) \leq D(A, B) + D(A, C)$

Efficiency. Trajectory similarity is a crucial component of data mining tasks, and in most cases, it is used to search for trajectories in massive datasets. Thus, it is primordial to have methods of low computational complexity to calculate the similarity between trajectories.

Local time shifting. Vehicles are allowed to move at different speeds, and location-acquisition devices can acquire data at different sampling rates. For instance, two trajectories can follow the same path but at different timestamps. The shifts in the sample acquisition times can even be only in some parts of the trajectories. Thus, vehicle trajectory similarity measures need to be able to take into account local time shifts when computing the distance between trajectories.

Different lengths. Vehicular trajectories can have different lengths. Thus, similarity measures need to be able to handle trajectories that do not have the same number of samples, if they are

Table 2. Main Notations

Symbol	Description
T_i	GPS Trajectory i
$\|T_i\|$	Number of sampling points of trajectory T_i
ρ_i^j	ith point of Trajectory j
$dist(\rho_i, \rho_j)$	Straight-line distance between the given points if they are in Euclidean space, or the greatest circle distance if the points are geographical
$Head(T_i)$	First point of T_i, that is, ρ_1^i
$Rest(T_i)$	Sub-trajectory of T_i without the first point ρ_1^i

GPS-based trajectories, and that have a different number of road segments if they are road-network constrained trajectories.

Robustness to noise and outliers. Most of the noise in GPS-based trajectories come from the fact that satellite-based positioning systems have precision errors. For instance, these errors are more common in metropolitan regions due to low satellite visibility and phenomena that affect signal quality [55]. In addition to noise, anomalies in the sensor collecting the data might introduce outliers into GPS-based trajectories. For methods based mostly on points distance, these outliers can result in similarity measures that are quite different from reality. One solution for this is to apply preprocessing techniques to remove outliers from the trajectories before comparing them. As for the noises, one drawback of most solutions that are robust to this is that they violate the triangle inequality, because they do not take into account the different sub-trajectories equally [111].

Parameter-free. Some methods introduce parameters to compare trajectories, such as a threshold to match similar geospatial points. The problem with approaches like the aforementioned is that the parameters need to be adjusted a priori per each scenario.

Completeness. It compares trajectories as a whole instead of just segments of the trajectories.

4.2 Methods for GPS-based Trajectories

GPS-based vehicular trajectories are a particular case of time series, and thus most of the similarity measures to compare them were initially proposed to compare time series. However, we cannot directly apply similarity measures for time series to compare GPS-based trajectories due to the unique characteristics of the later. For instance, vehicular trajectories usually have two or three dimensions, while most of the measures to compare time series focus on one-dimensional data. Besides that, GPS-based trajectories can have many outliers because of precision errors in satellite-based positioning systems. Another difference is that the different sampling rates, speeds, moving patterns, and regions of vehicle trajectories may introduce local time shifts into it. Thus, we need to adapt the similarity measures for time series before using it to compare vehicular trajectories.

Table 2 compiles the main notations used throughout this work.

4.2.1 Euclidean Distance (ED). Euclidean Distance (alternatively, L_2 Norm) is a well-known function commonly used to calculate the distance between time series. In the context of vehicle trajectories, the euclidean distance between two trajectories of the same length is the average of the distances between ordered pairs of points of the two trajectories. Formally, given trajectories T_1 and T_2, the Euclidean distance between T_1 and T_2 is

$$ED(T_1, T_2) = \frac{\sum_{i=1}^{n} dist\left(\rho_i^1, \rho_i^2\right)}{|T_1|}. \tag{2}$$

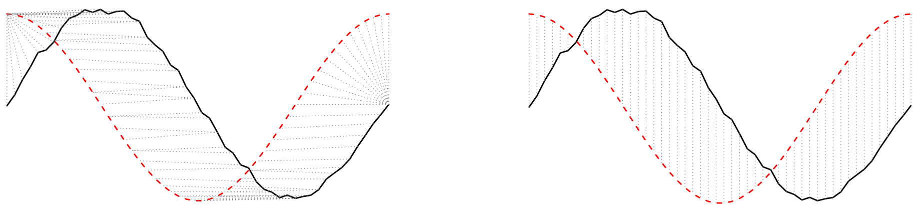

(a) Matching using Dynamic Time Warping. (b) Matching using Euclidean Distance.

Fig. 3. Unlike ED, DTW supports shrinkage or stretching of trajectories in the time axis to provides a better adjustment of the points, and thus represents a more sophisticated distance measure.

The main drawback of ED, besides not being able to compare differently sized trajectories, is that it cannot handle local time-shifting. Instead, it compares trajectory points one by one by using their indexes. As the compared points may be far apart from each other, especially considering trajectories of different speeds, ED cannot handle trajectories of different sizes and struggles with outliers and noise. However, ED is a metric, and so is easily indexable. Furthermore, ED is parameter-free, which makes it a suitable option to compare long trajectories.

4.2.2 Dynamic Time-warping-based Measures (DTW). Like ED, DTW is a distance measure originally proposed to compare time series. DTW finds the match of minimal cost, where each point of the first trajectory is paired to one or more consecutive points of the second trajectory while respecting some restrictions. Moreover, DTW allows the comparison between trajectories of different lengths, and its original definition is also parameter-free. Although DTW is non-metric, many studies in the literature propose approximate and exact indexing techniques to provide a fast search of trajectories [51]. Figure 3 illustrates DTW in comparison with the Euclidean Distance.

Given two vehicular trajectories T_1 and T_2, one can compute its distance using Dynamic Time Warping $DTW(T_1, T_2)$ using Equation (3):

$$DTW(T_1,T_2)=\begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ dist(Head(T_1), Head(T_2)) + \min\begin{cases} DTW(Rest(T_1), T_2) \\ DTW(T_1, Rest(T_2)) \\ DTW(Rest(T_1), Rest(T_2)) \end{cases}, & \text{otherwise.} \end{cases} \tag{3}$$

Warping path is the sequence of steps that computes the value of $DTW(T_1, T_2)$. As this process has optimal substructure and overlapping sub-problems, it is implemented using dynamic programming. The computational complexity of DTW is $O(|T_1| \cdot |T_2|)$, which makes it much slower than other similarity measures, such as the Euclidean Distance. Thus, many studies proposed pruning methods to makes faster the similarity-based queries that use DTW [51, 52, 95].

Keogh and Pazzani [52] proposed the Piecewise Dynamic Time Warping (PDTW) to compute approximate values of the DTW distance efficiently. Their technique reduces the size of the time series by a piecewise aggregate approximation and, consequentially, makes the DTW faster by a constant factor that depends on input data, but typically one or two orders of magnitude. They showed through experiments that PDTW provides accurate approximations of the DTW distance.

FastDTW [95] is an approximation algorithm of DTW that uses a multilevel approach starting from a downsampled version of the time series and iteratively performs the following steps. First, it finds the warping path on the lower resolution grid. Then, it projects the solution on a higher resolution grid and defines a first warp path band that is adjustable with a radius r parameter. The last step of the iteration is to refine the solution by evaluating the higher-level grid limited to this band. Both the time and space complexities of FastDTW are linear.

Mao et al. [70] proposed the Segment-based Dynamic Time Warping (SDTW), which is the integration of three distance measures with the traditional DTW algorithm. Given the points ρ_i^1 of trajectory T_1 and ρ_j^2 of trajectory T_2, the point-segment distance between them is their spatial distance taking into account their neighboring points and is used to reduce the sensitivity of the method to the sampling rate of the trajectories. Given that ρ_i^1 has an earlier timestamp and ρ_j^2 has a later timestamp, the prediction distance between them is the distance between ρ_i^1 and the approximate position of ρ_j^2 at the earlier timestamp. Finally, the segment-segment distance combines an adjustable parameter ω and the spatial, temporal, and angle distances. According to Mao et al. [70], the prediction distance employs the trajectory data features to optimizes the temporal distance. Also, the segment-segment distance is aware of the trajectory shape and thus improve the similarity measure accuracy. Despite its good accuracy, SDTW presents a high time complexity equal to $O(\log(|T_1| + |T_2|) \cdot |T_1| \cdot |T_2|)$.

4.2.3 Edit Distance-based Measures. The concept of *edit distance* was initially proposed to compare strings and then was employed to measure the similarity of time series. In the context of strings, edit distance is the smallest amount of operations (insertion, deletion, and replacement) required to make both strings equals [78]. We call *simple edit distance* (or edit distance) if all operations cost one, and *general edit distance* if the operations have different costs, or if the costs change according to the characters associated with the operation. Finally, if only insertions and deletions at cost one are allowed, we call it the Longest Common Subsequence (LCSS).

To compare trajectories using the LCSS, we usually define a threshold to match points that are close to each other. Vlachos et al. [111] proposed an LCSS-based algorithm to estimate the similarity between trajectories. It is non-metric and relies on two parameters, named δ and ϵ. δ controls how far in time a point of the first trajectory can go to match a point of the second one, and ϵ is the matching threshold. Formally, the LCSS between a pair of trajectories as presented by Vlachos et al. [111] is

$$LCSS_{\delta,\epsilon}(T_1,T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ 1 + LCSS_{\delta,\epsilon}(Rest(T_1), Rest(T_2)), & \text{if } dist(Head(T_1), Head(T_2)) < \epsilon \text{ and} \\ & ||T_1| - |T_2|| \le \delta \\ \max \begin{cases} LCSS_{\delta,\epsilon}(Rest(T_1), T_2) \\ LCSS_{\delta,\epsilon}(T_1, Rest(T_2)) \end{cases}, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, they normalize the LCSS by dividing it by the number of samples of the shortest trajectory. The method proposed by Vlachos et al. is *symmetric*, that is, $LCSS_{\delta,\epsilon}(T_1,T_2)$ is equal to $LCSS_{\delta,\epsilon}(T_2,T_1)$, and has time complexity equal to $O(\delta \cdot (|T_1| + |T_2|))$ (although that of LCSS originally is $O(|T_1| \cdot |T_2|)$).

Edit Distance on Real Sequence (EDR) [22] is a similarity measure with characteristics close to the LCSS. The main difference between LCSS and EDR is that EDR assigns penalties when a point on one trajectory does not match the point on the other trajectory. Formally, EDR between two trajectories is

$$EDR(T_1,T_2) = \begin{cases} |T_2|, & \text{if } |T_1| = 0 \\ |T_1|, & \text{if } |T_2| = 0 \\ \min \begin{cases} EDR(Rest(T_1), Rest(T_2)) + subcost \\ EDR(Rest(T_1), T_2) + 1 \\ EDR(T_1, Rest(T_2)) + 1 \end{cases}, & \text{otherwise,} \end{cases} \tag{5}$$

where $subcost = 0$ if $dist(Head(T_1), Head(T_2)) < \epsilon$ and $subcost = 1$, otherwise.

Like other edit distance-based methods, we can apply dynamic programming to compute EDR and its time complexity is quadratic ($O(|T_1| \cdot |T_2|)$), which makes using it with sequential scan on large databases a slow procedure. Besides that, EDR uses a threshold ϵ to reduce the effects of noise. EDR is not a metric, because the employment of the threshold makes it violates the triangle inequality. Therefore, we cannot use distance-based indexing techniques to improve trajectory search using the EDR distance.

Chen and Ng [21] presented the Edit Distance with Real Penalty (ERP), which is metric and so can cope with some indexing problems of other methods. Besides that, ERP supports local time shifting. The idea of ERP is to use a constant reference point g instead of a threshold ϵ to compute distances. Chen and Ng [21] proved that ERP satisfies the triangle inequality regardless of the value of g. However, they suggested using g equal to zero. Equation (6) defines ERP:

$$ERP(T_1, T_2) = \begin{cases} \sum_{i=1}^{|T_2|} dist(\rho_i^2, g), & \text{if } |T_1| = 0 \\ \sum_{i=1}^{|T_1|} dist(\rho_i^1, g), & \text{if } |T_2| = 0 \\ \min \begin{cases} ERP(Rest(T_1), Rest(T_2)) + dist(Head(T_1), Head(T_2)) \\ ERP(Rest(T_1), T_2) + dist(\rho_1^1, g) \\ ERP(T_1, Rest(T_2)) + dist(\rho_1^2, g) \end{cases}, & \text{otherwise.} \end{cases} \tag{6}$$

ERP has the same computational behavior of EDR, LCSS, and DTW, and thus its time complexity is $O(|T_1| \cdot |T_2|)$. However, unlike the other distance functions, ERP is a metric, so it can be used to prune unnecessary trajectories by applying the triangle inequality.

Time Warp Edit Distance (TWED) [71] is a distance measure presented as an alternative to ERP as it has the same features of ERP. The main difference between them is that TWED employs the samples acquisition times during the computations. TWED introduces two parameters, ν and λ. The first parameter, ν, applies higher penalties to matched points that are farther away from each other concerning their acquisition times, and λ is the penalty for the deletions. Formally, the TWED between two trajectories is

$$TWED_{\lambda,\nu}(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \min \begin{cases} TWED_{\lambda,\nu}(Rest(T_1), T_2) + \Gamma(\rho_1^1) \\ TWED_{\lambda,\nu}(T_1, Rest(T_2)) + \Gamma(\rho_1^2) \\ TWED_{\lambda,\nu}(Rest(T_1), Rest(T_2)) + \Gamma(\rho_1^1, \rho_1^2) \end{cases}, & \text{otherwise,} \end{cases} \tag{7}$$

where

$$\Gamma\left(\rho_i^j\right) = dist\left(\rho_i^j, \rho_{i+1}^j\right) + \nu \cdot \left(\rho_i^j.t - \rho_{i+1}^j.t\right) + \lambda, \tag{8}$$

$$\Gamma\left(\rho_i^1, \rho_j^2\right) = dist\left(\rho_i^1, \rho_j^2\right) + dist\left(\rho_{i+1}^1, \rho_{j+1}^2\right) + \nu \cdot \left(\left|\rho_i^1.t - \rho_j^2.t\right| + \left|\rho_{i+1}^1.t - \rho_{j+1}^2.t\right|\right), \tag{9}$$

and $\rho_{|T_i|+1}^i$ (one index after the last point of the trajectory) represents a point with value zero.

Edit Distance with Projections (EDwP) [90] is a parameter-free distance function aimed at trajectories that have different sampling rates. The first step of EDwP is to convert the GPS-based trajectory into line segments through linear interpolation. Then, given two trajectories T_1 and T_2 that are presented as segment sequences, the EDwP between them is the least expensive set of *insert* and *replace* operations that make them identical. Like most edit distance-based functions, EDwP takes into account local time shift and has a quadratic computational cost. However, EDwP

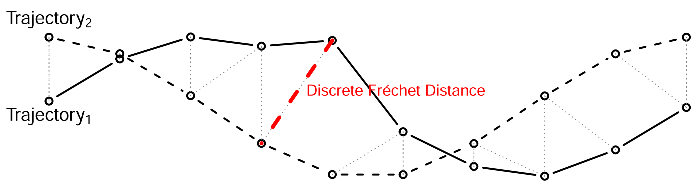


Fig. 4. The bold dashed line represents the discrete Fréchet distance between the trajectories. In this example, the discrete Fréchet distance coincides with the Hausdorff distance.

robust against sampling rate variations. Equation (10) formally defines the EDwP:

$$EDwP(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \min \begin{cases} EDwP(Rest(T_1), Rest(T_2)) + RepCost(T_1, T_2) \\ EDwP(Ins(T_1, T_2), T_2) \\ EDwP(T_1, Ins(T_2, T_1)) \end{cases}, & \text{otherwise,} \end{cases} \tag{10}$$

where

$$RepCost(T_1, T_2) = Rep(T_1.e_1, T_2.e_1) \times Coverage(T_1.e_1, T_2.e_1), \tag{11}$$

$$Rep(e_1, e_2) = dist(e_1.s, e_2.s) + dist(e_1.e, e_2.e), \tag{12}$$

$$Coverage(e_1, e_2) = length(e_1) + length(e_2), \tag{13}$$

and e_i represents a line segment created by the linear interpolation between two consecutive points. $e_i.s$ and $e_i.e$ represent the first and last points, respectively, of the line segment e_i. $T_i.e_1$ represents the first line segment of the trajectory T_i. Finally, $Ins(T_1, T_2)$ is an operation to aid robust matching that returns a modified version of T_1. It works by introducing one extra point between the first two points of T_1. Thus, $Ins(T_1, T_2)$ effectively divides the first line segment of T_1 ($T_1.e_1$) into two segments. The split point is the point on $T_1.e_1$ that is spatially closest to the endpoint of the first line segment of T_2 ($T_2.e_1.e$).

The results of EDwP depend on the length of the trajectories, as it is the sum of every operation. Therefore, in certain situations, it is better to normalize the results of EDwP. To do this, we divide the EDwP by the sum of the lengths of trajectories T_1 and T_2.

4.2.4 Fréchet Distance. Fréchet Distance [33], illustrated in Figure 4, is one of the most popular similarity measures and falls within the category of geometric shape-based measures. A classic example to illustrate Fréchet distance between two trajectories is that of a man walking his dog. The man is traversing a finite curved path while walking his dog on a leash, and the dog is traversing a different one. Both man and dog may have different speeds at different times, but they cannot move backward. The Fréchet distance is the length of the smallest leash sufficient for both to traverse their separate paths. The advantage of the Fréchet distance is that it can compare trajectories that have different sizes and sampling rates. However, the Fréchet distance is not robust to noise and outliers, nor it tackles local time shifting.

The concept of Fréchet distance is for continuous curves, but for some applications (e.g., analysis of vehicle trajectories), it is often useful to use polygonal curves. Alt and Godau [3] presented an exact algorithm based on a parametric search that receives as input two polygonal curves and computes their Fréchet distance in $O(|T_1| \cdot |T_2| \cdot log(|T_1| \cdot |T_2|))$. To address the high computational complexity of the algorithm proposed by Ant and Godau, Eiter and Mannila [30] presented a discrete version of the Fréchet distance for polygonal curves called *coupling distance*. The idea of coupling distance is to look at all possible couplings between the endpoints of the polygonal curves

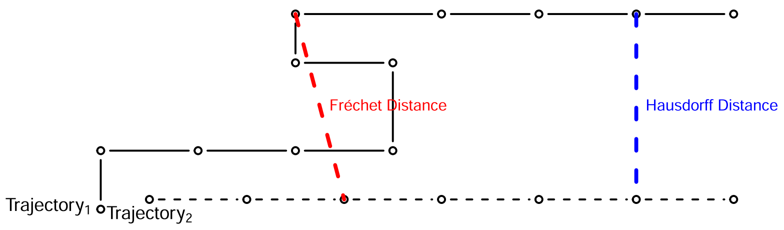


Fig. 5. The difference between Hausdorff distance and Fréchet distance.

line segments. Eiter and Mannila [30] showed that the coupling distance is a good approximation to the Fréchet distance and presented an algorithm that computes it in $O(|T_1| \cdot |T_2|)$.

4.2.5 Hausdorff Distance. Hausdorff distance, also called Pompeiu-Hausdorff distance, is a metric measure similar to Fréchet distance. In the context of GPS-based trajectories, it is the longest of all distances from a location in one trajectory to the nearest location in the other trajectory, and vice versa. The main difference between Fréchet and Hausdorff distances is that the order of points affects the former distance but not later one. In Figure 4, the Hausdorff distance coincides with the discrete Fréchet distance. Figure 5 illustrates the difference between the two distance measures.

Equation (14) defines the Hausdorff distance $Dist_{Hausdorff}$ between two GPS-based trajectories T_1 and T_2. The computational complexity of the Hausdorff distance is $O(|T_1| \cdot |T_2|)$, the same as that of discrete Fréchet distance:

$$Dist_{Hausdorff}(T_1, T_2) = \max \left\{ \max_{\rho_i^1 \in T_1} \min_{\rho_j^2 \in T_2} dist\left(\rho_i^1, \rho_j^2\right), \max_{\rho_j^2 \in T_2} \min_{\rho_i^1 \in T_1} dist\left(\rho_i^1, \rho_j^2\right) \right\}. \tag{14}$$

4.2.6 Other Methods. Other methods are designed by considering the actual applications. Chen et al. [23] proposed a similarity function that combines the characteristics of LCSS, in the sense that it can ignore samples that do not have good match candidates in the other trajectory, and DTW, which can match a trajectory point with more than one point of the other trajectory. It works by matching each point of one trajectory to only one point of the other trajectory. Therefore, Chen et al. [23] designed their method for situations where the first trajectory represents a few query locations, and the object is to verify whether the query locations match with the trajectory instead of verifying whether the trajectory and the query locations are similar in shape. Given two trajectories T_1 and T_2, their similarity measure as proposed by Chen et al. [23] is

$$Sim_o(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \max \begin{cases} e^{-dist(Head(T_1), Head(T_2))} + Sim_o(Rest(T_1), T_2) \\ Sim_o(T_1, Rest(T_2)) \end{cases}, & \text{otherwise.} \end{cases} \tag{15}$$

With the exponential function $e^{-dist(Head(T_1), Head(T_2))}$, points that are closer to each other receive a much higher similarity value compared to those that are far away. In other words, the contribution exponentially decreases as $dist(Head(T_1), Head(T_2))$ linearly increases.

Ta et al. [102] presented a bi-directional similarity (BDS) measure to support solutions of the trajectory similarity join problem. Trajectory similarity join is the task of finding every pair of similar trajectories from different datasets. BDS works by aligning each point of one trajectory to the nearest location on the other trajectory, which is not necessarily a sample of the trajectory, and vice versa. The overall complexity of BDS is $O(|T_1| \cdot |T_2|)$.

Other methods use the shape formed by the line strings of two trajectories to compute the distance between them. For instance, the Locality In-between Polylines (LIP) [84] sums the areas of all polygons formulated between intersection points, which in turn are generated by the overlay

Table 3. Trajectory Similarity Methods for GPS-based Trajectories

Method	Complex.	Metric	≠ len.	Param.-free	Loc. Ti. Shift	Noise
ED	$O(n)$	✓		✓		
DTW	$O(n^2)$		✓	✓	✓	
PDTW [52]	$O(n^2)$		✓		✓	
FastDTW [95]	$O(n)$		✓		✓	
SDTW [70]	$O(n^2 \log n)$		✓		✓	✓
ERP [21]	$O(n^2)$	✓	✓	✓	✓	
EDR [22]	$O(n^2)$		✓		✓	✓
TWED [71]	$O(n^2)$	✓	✓		✓	
LCSS [111]	$O(n^2)$		✓		✓	✓
Fréchet Dis.	$O(n^2 \log n^2)$		✓	✓		
Hausdorff Dis.	$O(n^2)$	✓	✓	✓		
EDwP [90]	$O(n^2)$		✓	✓	✓	✓

of the trajectories in the two-dimensional plane. They use linear interpolation between consecutive points to represent the line segments. Besides that, LIP uses the length of the segments as weights to the areas of the polygons. Equation (16) defines LIP:

$$LIP(T_1, T_2) = \sum_{\forall polygon_i} Area_{polygon_i} \cdot w_i. \tag{16}$$

Pelekis et al. [84] proposed, in addition to LIP, a time-aware distance function called STLIP that takes into account the time factor. In summary, STLIP is a multiple of LIP weighted by a penalty defined by the user that adds the time factor to the distance function. The time complexity of LIP and STLIP is $O(N \cdot log(N))$, where $N = |T_1| + |T_2|$.

NeuTraj [118] is a generic approach based on neural metric learning that accelerates approximate trajectory similarity computations for any measure discussed above. In summary, NeuTraj uses the pair-wise similarities of a sample of a given trajectory database as guidance to a recurrent neural network (RNN) that approximates a generic distance function, such as DTW, Hausdorff, and ED. Once the neural network is constructed, NeuTraj computes the approximate distance between two trajectories in linear time.

Table 3 summarizes the similarity methods for GPS-based trajectories and their properties.

4.3 Methods for Road-network Constrained Trajectories

The most common representation of road-network constrained trajectories is as an ordered set of connected road segments from a given road network. Therefore, some proposals use operations from set theory to measure the similarity between these trajectories [114, 117]. However, instead of using the number of segments in common, we can use the lengths of each segment. For instance, Won et al. [114] presented a dissimilarity measure called Dissimilarity with Length (DSL) to cluster road-network constrained trajectories. The DSL between two trajectories is the sum of the road lengths of their disjoint set, divided by the sum of the lengths of both trajectories. Formally,

$$DSL(T_1, T_2) = \frac{L_d(T_1, T_2)}{L_s(T_1) + L_s(T_2)}, \tag{17}$$

where $L_d(T_i, T_j)$ is the sum of the road lengths of the disjoint set of T_i and T_j, and $L_s(T_i)$ is the total length of the roads of T_i.

A drawback of the method proposed by Won et al. [114] is that it does not identify the similarity of trajectories that are close to each other but do not have segments in common. Besides that, the DSL method does not take into account the trajectories timestamps nor their speeds.

Xia et al. [117] proposed a method based on the Jaccard similarity coefficient [103] to calculate the spatiotemporal similarity between road-network constrained trajectories. Equation (18) defines its spatial component, where $L_c(T_1, T_2)$ is the sum of the road lengths contained in the intersection set of trajectories T_1 and T_2, and $L_s(T_i)$ is the sum of the road lengths of trajectory T_i:

$$SSim(T_1, T_2) = \frac{L_c(T_1, T_2)}{L_s(T_1) + L_s(T_2) - L_c(T_1, T_2)}. \tag{18}$$

They compute the temporal component $TSim(T_i, T_j)$ of the similarity measure by replacing, in Equation (18), the spatial lengths L_s by lifespans of the temporal dimension L_t. The authors argue that the trajectories should be considered as spatiotemporal similar when the spatial and temporal similarities are both high. Thus, the spatiotemporal similarity measure $STSim(T_i, T_j)$ between trajectories is the product of the spatial similarity measure $SSim(T_i, T_j)$ and the temporal similarity measure $TSim(T_i, T_j)$.

The similarity measures presented by Xia et al. [117] and Won et al. [114] are similar and thus have the same drawbacks. First, their methods do not identify the similarity of trajectories that are close to each other and that do not have segments in common. Their methods also do not take into account either the timestamps of the trajectories or the speeds of the vehicles. Finally, both methods have high computational complexity.

For particular applications, such as trajectories searching, we can compute the similarity between road-network-constrained trajectories according to a collection of Points of Interest (POIs) or Times of Interest (TOIs). For instance, one can be interested in trajectories that passed through some places at some time intervals. Hwang et al. [45] proposed a spatial similarity measure and a temporal similarity measure for road-network constrained trajectories, which considers the (S_{id}, d, t) coordinate system instead of the Euclidean space, where S_{id} is a road segment identifier, t is the timestamp of the sampling point, and d is the distance between the current position and the starting location of the road segment. The authors of Reference [45] applied both spatial and temporal similarity measures in a trajectory search technique that uses the spatial similarity to select some trajectories and then employs the temporal similarity measure to improve the selection. They consider two trajectories similar if they pass through the same POIs. According to the authors, the POIs can be, for instance, relevant places defined according to application criteria. Few applications can use their method because of the necessity to define in advance the points of interest. Equation (19) defines the spatial similarity of trajectories T_1 and T_2 using a set of POIs P.

$$Sim_{POI}(T_1, T_2, P) = \begin{cases} 1, & \text{if } \forall p \in P, p \in T_1 \wedge p \in T_2 \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Equations (20) and (21) define the temporal component of the similarity measures. Equation (20) computes the distance considering one point of interest, while Equation (21) considers a set of points of interest:

$$Dist_T(T_1, T_2, p) = |t(T_1, p) - t(T_2, p)|, \tag{20}$$

$$Dist_T(T_1, T_2, P) = \left(\sum_{i=1}^{k} |t(T_1, p_i) - t(T_2, p_i)|^k \right)^{\frac{1}{k}}, \tag{21}$$

where k is the size of the set of points of interest P and $t(T_i, p)$ represents the timestamp of the moment when T_i pass through the point of interest p. If p is neither contained in T_1 nor T_2, then it is assumed a infinity temporal distance.

Hwang et al. [46] extended their previous work [45] by introducing the concept of TOI on road networks. The TOI can represent, for instance, the peak hours where there are the most traffic jams. The temporal similarity between two trajectories T_1 and T_2 using a set of TOIs T is

$$Sim_{TOI}(T_1, T_2, T) = \begin{cases} 1, & \text{if } \forall t \in T, t \in [T_1.s, T_1.e] \wedge t \in [T_2.s, T_2.e] \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where $T_i.s$ and $T_i.e$ represent the times when the trajectory started and ended, respectively.

Hwang et al. [46] also proposed Equation (23) to calculate the trajectories spatial distance, taking into account the set of times of interest TOI:

$$Dist_{TOI}(T_1, T_2, TOI) = \sum_{i=1}^{k} dist(p(T_1, TOI_i), p(T_2, TOI_i)), \tag{23}$$

where $p(T_i, TOI_j)$ denotes the point of T_i in the timestamp TOI_j and k is the size of set TOI.

The measures proposed by Hwang et al. [45, 46] have several drawbacks. First, the fact that the sets POIs and TOIs need be defined in advance by users restricts their usage to few applications and may lead to erroneous conclusions if the POIs and TOIs are not adequately defined. Besides that, the similarity space (1, if the trajectories are similar, and 0 otherwise) only informs if the trajectories are similar or not. However, in most situations, it is better to know how similar the trajectories are.

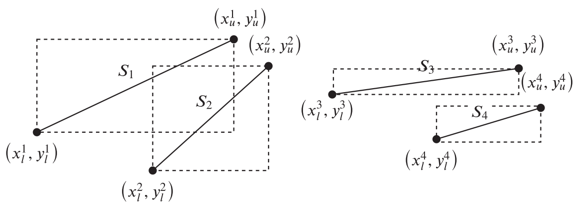
Abraham and Lal [2] presented a spatiotemporal similarity measure to overcome the lack of similarity level notion of the approach proposed by Hwang et al. [45, 46]. Their method, which compares road-network constrained trajectories that are represented by a binary encoding scheme [61], takes into account the hierarchical components of the binary-encoded location. To this, they consider each location's component separately. Besides the binary encoding scheme, they use dimensionality reduction [1] to manage the road network data.

Tiakas et al. [104] proposed another approach to overcome the drawbacks of Hwang' measures. They defined a set of distance functions (metrics). For instance, $D_{network}$ and D_{time} compare trajectories in space and in time, respectively, and D_{total} is the total (combined) distance, which is weighted by two parameters that define the importance of space and time factors. Formally, D_{total} is defined as

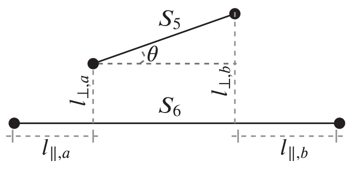
$$D_{total}(T_1, T_2) = D_{network}(T_1, T_2) \cdot W_{network} + D_{time}(T_1, T_2) \cdot W_{time}. \tag{24}$$

Like other similarity measures for road-network constrained trajectories, the method proposed by Tiakas et al. [104] describes the road network as a directed graph. In summary, the network distance $D_{network}$ computes routing distances taking into account the graph of the road network, and the time distance D_{time} computes the time required to travel between two consecutive nodes. The distance measures $D_{network}$, D_{time}, and D_{total} can handle only equally sized trajectories. To overcome this issue, they proposed a decomposition process that splits the trajectories into sub-trajectories of equal size, and then, they index the sub-trajectories by M-trees. Although all distance measures proposed by Tiakas et al. [104] are metrics, the introduction of the $W_{network}$ and W_{time} parameters is a disadvantage of their approach.

Aiming at big trajectory data clustering, Kumar et al. [57, 58] proposed a DTW-based method, called trajDTW, that employs the Dijkstra algorithm to compute road-network constrained trajectory similarities. The trajDTW defines a window parameter w, which is equal to half the size of the shortest trajectory, to avoid overestimation of the real distance. The authors demonstrated in Reference [57] the superiority of the trajDTW over DSL and Hausdorff distance measures. However, to be faster, trajDTW requires the pre-computation of the distances between all pairs of edges in the road network [91].



(a) Minimum Bounding Rectangles distance.



(b) Trajectory-Hausdorff distance.

Fig. 6. The Minimum Bounding Rectangles (MBR) and Trajectory-Hausdorff (TD_{Haus}) distance measures for line segments.

Longest Common Road Segments (LCRS) [124] similarity measure is a version of the LCSS function adapted to road segments. First, LCRS computes the longest common road segments between two road-network restricted trajectories in terms of road segment lengths instead of the number of road segments (Equation (25)). Next, it normalizes the length of these segments using a method similar to Jaccard (Equation (26)). We can compute the LCRS between two trajectories using a dynamic programming algorithm with quadratic time complexity. Also, LCRS shares other features of LCSS, such as being able to compute the similarity between trajectories of different sizes, and not being a metric.

$$LCRS(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ length(Head(T_1)) + LCRS(Rest(T_1), Rest(T_2)), & \text{if } Head(T_1) = Head(T_2) \\ \max \begin{cases} LCRS(Rest(T_1), T_2) \\ LCRS(T_1, Rest(T_2)) \end{cases}, & \text{otherwise}, \end{cases} \tag{25}$$

$$LCRS_{normalized}(T_1, T_2) = \frac{LCRS(T_1, T_2)}{|T_1| + |T_2| - LCRS(T_1, T_2)}. \tag{26}$$

4.3.1 Road Segments Distance. When comparing road-network constrained trajectories, one can consider the difference between the shapes of each road segment of the trajectories. To this end, we can use the concept of Minimum Bounding Rectangles (MBR) or the Trajectory-Hausdorff Distance.

Figure 6(a) illustrates the distance of two trajectory segments using their MBRs. The MBR of a line segment S_i is described by the coordinates of its lower bound (x_l, y_l) and upper bound (x_u, y_u) points. Equation (27) defines the MBR-based distance D_{MBR} between two trajectory segments. In the example, the MBR distance between S_1 and S_2 is 0 and between S_3 and S_4 is $y_l^3 - y_l^4$:

$$D_{MBR}(S_1, S_2) = \sqrt{(\Delta([x_l^1, x_u^1], [x_l^2, x_u^2]))^2 + (\Delta([y_l^1, y_u^1], [y_l^2, y_u^2]))^2}, \tag{27}$$

where $\Delta\left(\left[x_l^1, x_u^1\right], \left[x_l^2, x_u^2\right]\right)$ is the distance between two intervals, defined as

$$\Delta\left(\left[x_l^1, x_u^1\right], \left[x_l^2, x_u^2\right]\right) = \begin{cases} 0, & \text{if } [x_l^1, x_u^1] \cap [x_l^2, x_u^2] \neq \emptyset, \\ x_l^1 - x_u^2, & \text{if } x_l^1 > x_u^2, \\ x_l^2 - x_u^1, & \text{if } x_l^2 > x_u^1. \end{cases} \tag{28}$$

The Trajectory-Hausdorff distance, as proposed by Lee et al. [60], compares trajectory segments by joining together the parallel ($d_{\|}$), angular (d_θ), and perpendicular ($d_\perp$) distances. Lee et al. [60] created their distance function by adapting techniques used in the pattern recognition research field, which in turn were inspired by the Hausdorff distance. Therefore, Zheng called the method of Lee et al. [60] as Trajectory-Hausdorff Distance (TD_{Haus}) in Reference [125]. Formally,

$$TD_{Haus}(S_i, S_j) = \omega_\perp \cdot d_\perp + \omega_\parallel \cdot d_\parallel + \omega_\theta \cdot d_\theta, \tag{29}$$

where $d_\perp = \frac{l_{\perp,a}^2 + l_{\perp,b}^2}{l_{\perp,a} + l_{\perp,b}}$, $d_\parallel = min(l_{\parallel,a}, l_{\parallel,b})$, $d_\theta = |S_j| \cdot \sin\theta$, and the weights $\omega_\perp$, $\omega_\parallel$, and ω_θ are defined according to the application requirements. Figure 6(b) illustrates these components.

4.4 Methods for Other Trajectory Representations

For trajectory representations where the positions of the vehicle are pre-defined, and only the temporal component of the trajectories is of interest, the shrinkage or stretching of trajectories is not allowed, and thus, methods such as DTW are not appropriate. Therefore, Tiesyte and Jensen [105] considered specific distance measures (e.g., modified versions of the LCSS, L_p distance, and weighted L_p distance) that are appropriate for this scenario. For the weighted L_p distance, one can define the weights according to the application. For instance, the weights can be equal, indicating that all parts of the trajectories are evenly important. Another option is to create correlation-based weights when historical data is available. Tiesyte and Jensen [105] used Kendall's tau coefficient τ [50] for determining the weights according to the correlations. Other correlation coefficients, such as Pearson's [92], are also feasible, but the authors opted for Kendall's coefficient based on empirical results.

Deep learning-based trajectory representations, such as the Trembr framework, convert trajectories to low-dimensional feature vectors called trajectory embeddings. We can use these embeddings to calculate the similarity between two trajectories. For instance, Fu and Lee [35] proposed to employ the Euclidean distance between the learned embeddings to measure the distance between trajectories. Their method outperformed other deep learning-based methods. However, it is unclear how much the results of the Trembr framework depend on the quality of the map-matching performed in the preprocessing step, and consequently, on the sampling rate of the raw trajectory.

5 APPLICATIONS

The task of comparing vehicular trajectories is fundamental for trajectory analysis applications such as predictive queries [27], traffic monitoring [40, 53], and behavior mining [115]. These analysis applications are, in turn, enablers of numerous other applications. For instance, driver assistance systems can detect traffic congestion in real time and calculate optimal routes. The public administration can learn about traffic patterns to guide the policies of urban mobility improvement, or use historical data to forecast the development of traffic jams. Automobile insurers or taxi companies might benefit by applying behavior analysis or recognizing anomalous trajectories. Therefore, choosing a proper similarity measure plays a crucial role in enhancing the efficiency and precision of these applications. For example, DTW-based approaches provide accurate results for similarity-based trajectory retrieval, but they are not suitable for trajectory retrieval in large databases. In the following, we present some analysis applications of vehicular trajectory similarity measures and discuss open research problems related to these applications.

5.1 Similarity-based Trajectory Retrieval

Most applications of vehicular trajectory similarity measures derive from the search for related trajectories in large vehicular trajectory databases. For instance, when a vehicle is traveling on a known route, one can identify trajectories similar to the current, partial trajectory, from historical datasets to predict the travel time [49, 63, 65, 105]. As the employed similarity measure directly impacts the effectiveness and efficiency of these kinds of tasks, one needs to choose the most suitable similarity measure carefully. Given a distance function F, we can express the total time T

to evaluate a query as

$$T = \text{I/O time} + \text{number of similarity evaluations} \times \text{complexity of } F. \tag{30}$$

However, in many applications, the cost to solve the function F is much higher than the cost of the other elements, especially now with the increasing availability of main memory [17].

5.1.1 Sequential Search. The most straightforward idea to search for trajectories similar to a given trajectory T is to make a linear scan over all trajectories and pick those with distance to T less than a given threshold [45]. However, using sequential scanning on today's large-scale databases is becoming impractical. Therefore, there is a demand for the development of new indexing techniques that employ fewer similarity computations [18].

Even though the sequential search is not efficient for querying similar trajectories in large databases, it is still used as a last step verification by many similarity-based retrieval techniques, which in turn employ traditional similarity measures. For instance, DTW is a ubiquitous distance measure widely used in sequential searches. However, as it presents a high computational complexity, some proposals aim to optimize the performance of DTW to allows its usage in the searching of trajectories in massive datasets. A standard technique to accelerate DTW-based queries is to apply a lower bound to prune off unlikely candidate trajectories. Rakthanmanon et al. [89] demonstrated that, with careful implementation, DTW is much more efficient than advanced ED-based techniques.

5.1.2 Trajectory Indexing. When the used distance function is a metric, we can apply the triangle inequality property to index trajectories and thus improve the efficiency of trajectories retrieval. One approach is the *pivot-based metric index* [20], which works as follows. First, in a preprocessing step, select the reference trajectories (pivots) $P_1, \ldots, P_k$ from the trajectory dataset and compute their distances to all trajectories in the dataset. Then, given a threshold distance d and a search trajectory Q, compute the distance from Q to each pivot P_i. Now we can, employing the triangle inequality, prune off all other trajectories T_i in which $|dist(Q, P_j) - dist(P_j, T_i)| > d$ guaranteeing no false dismissals. Many indexing structures follow this idea, such as AESA [93], LAESA [74], VP-tree [121], FQ-tree [6], MVP-tree [14], PM-tree [100], Omni-family [107], M-index [81], EP [94], and SPB-tree [19]. For instance, some treelike indexes select the pivots as roots and stores the distances from each root to their children.

Another option to improve the querying of a dense set of trajectories is to store hash-based trajectories instead of traditional spatial-indexing structures and use indexing techniques based on these hashes. For instance, trajectory fingerprinting with geodabs [16], discussed in Section 3.3.3, enables efficient queries on dense trajectory datasets. Although fingerprinting has a probabilistic nature, which makes the index unable to discriminate between true and false positives, the authors of geodabs demonstrated that it has a minimal influence on the accuracy of the results. Like other proposals, a drawback of the geodabs evaluation is that it relied on synthetic datasets due to the lack of public available large and dense trajectory datasets. Thus, an open research issue is to evaluate trajectory indexing techniques with real large-scale trajectory datasets.

5.1.3 Trajectory Similarity Join Problem. A variation of similarity-based retrieval is the trajectory similarity join problem, which intends at finding every pair of trajectories, from two distinct datasets, that have a similarity measure above a given threshold. Without loss of generality, techniques that address the trajectory similarity join problem can also be used to retrieve similar trajectories. Therefore, some studies employ similarity measures to address the trajectory similarity join problem [90, 97, 99, 102, 120, 124]. Besides that, self join (i.e., finding similar trajectories in only one dataset) is another way to visualize the clustering problem. We discuss trajectory clustering in Section 5.2.

Ta et al. [102] introduced the bi-directional similarity metric (BDS, discussed in Section 4.2.6) and signature-based methods to address the similarity join problem. They presented a set of algorithms for sorting, filtering, and indexing trajectories, which reduces the number of trajectory comparisons performed, and thus improves overall processing time. However, experimental results point to scalability issues for processing large databases. Therefore, the employment of more effective trajectory similarity measures that do not decrease the effectiveness of the results is still an open issue.

5.2 Clustering

Clustering is a tool that involves many disciplines for finding and defining groups of entities, called clusters, in datasets [76]. For instance, in the data mining research field, clustering is a mechanism for finding patterns in datasets. In knowledge discovery, it is a tool for updating, correcting, and extending the existing knowledge. In machine learning, it is a tool for prediction [76]. Therefore, vehicular trajectories clustering, which aims at grouping similar trajectories according to specific criteria, is a central application of trajectory similarity measures. Recent surveys [13, 123] presented reviews of trajectory clustering algorithms. Thus, in this section, we briefly discuss similarity-based clustering techniques.

Some naive approaches rely on using trajectory similarity measures and adapting traditional clustering algorithms. For instance, Kim and Mahmassani [53] developed a framework that uses the LCSS distance and the DBSCAN algorithm [32] to cluster GPS-based trajectories. They also proposed the employment of hierarchical agglomerative clustering to identify traffic streams, and then, classify new trajectories.

K-means is a traditional and straightforward clustering algorithm that is still widely used in many areas [47]. The idea of K-means is to find *K* clusters and associate each data point with the nearest cluster, aiming at minimizing the total variance of all data points in each cluster. However, this task is a known NP-hard problem. Therefore, most proposals employ a standard algorithm to find a local optimum. However, Meilă [72] demonstrated that if the clusters are appropriately defined, the probability that K-means converges to the global optimum is high. K-means performs many comparisons between data points. Thus, it is crucial to use similarity measures that are fast to compute, such as FastDTW, when applied to cluster vehicular trajectories [95].

T-OPTICS [77] is a popular density-based trajectory clustering algorithm that employs the euclidean distance and extends the OPTICS algorithm [4] to work with trajectories. Besides the T-OPTICS, the OPTICS algorithm inspired many other studies. Deng et al. [28] proposed an algorithm called Tra-POPTICS aiming at improving both the scalability and computing performance to cluster big trajectory datasets. Tra-POPTICS is inspired by the POPTICS algorithm [83], which in turn is a scalable version of the OPTICS algorithm. One of its modifications is that it employs the similarity measure introduced by Frentzos et al. [34] rather than using the euclidean distance.

As vehicle trajectories can be quite long, some clustering methods aims at grouping common sub-trajectories instead of whole trajectories, which can be useful for many applications. For instance, one may be interested in recovering all vehicles that traveled close to a set of road segments. TRACLUS [60] is a framework composed of two components. In the first component, an algorithm based on the minimum description length (MDL) principle [36] is responsible for partitioning the trajectories into sub-trajectories (line segments). Then, in the second component, a density-based clustering algorithm groups similar sub-trajectories into clusters.

Hu et al. [44] applied the Dirichlet process mixtures to represent, cluster, and find trajectories. Their technique employs the discrete Fourier transform (DFT) to describe sub-trajectories, which, in turn, are then used to train the DPMM. Then, they use the parameters of the tDPMMs to index the trajectory patterns. Unlike most other work, their method automatically estimates a suitable amount of clusters and can cluster new trajectories without retraining. However, their discussion about the method focuses on video-based trajectories, and it is unclear what would be the result of applying it to cluster low sampled GPS-based or road-network constrained trajectories.

Fast-clusiVAT [58] is another trajectory clustering algorithm that automatically estimates a suitable amount of clusters. In summary, Fast-clusiVAT is a modified and faster implementation of the clusiVAT algorithm [56] that employs the trajDTW distance measure. Experimental results showed that fast-clusiVAT outperforms some widely used clustering algorithms and provides a significant speedup over its previous version, clusiVAT, without loss of cluster quality.

Most of the clustering techniques discussed above employ a distance function but do not take into account the turn restrictions of the road networks. Han et al. [40] tackled this issue by proposing the NEAT system to cluster road-network constrained trajectories. Besides addressing road-network constrained trajectories, NEAT also considers the road-network-based proximity. NEAT is a three-phase clustering framework based on the following design guidelines. First, the road intersections represent initial partitioning points, and the indivisible sub-trajectories, called fragments, represent the trajectories. Next, the fragments that also represent road segments are clusters of objects associated with the trajectories. Finally, the last phase groups the fragments by taking into account the turn restrictions of consecutive road segments. Although the NEAT framework does not directly employ a similarity measure, it uses a function based on the Hausdorff distance in the last phase. The advantage of the NEAT framework is that as it uses the information about the road network restrictions and does not depend on computing many slow distance measures, it runs very fast. Experiments have shown that NEAT is faster than the TRACLUS framework [60] by more than three orders of magnitude. However, NEAT depends on a preprocessing stage that solves the map-matching problem to divide the trajectories into fragments.

5.3 Similarity-based Trajectory Prediction

We can use historical trajectory data and similarity measures to predict short- and long-term vehicle trajectories. For instance, short-term mobility prediction is crucial for advanced driver assistance systems and autonomous driving, and with long-term mobility prediction, traffic monitoring systems can forecast traffic congestion and guide drivers to faster routes. Recently, Lefèvre et al. [62] reviewed techniques for vehicle motion prediction, most of which rely on the Markov property. Besides that, other studies have proposed the use of deep neural networks, such as Long Short Term Memory (LTSM) [48] and Convolutional Neural Network [68], for vehicle trajectory prediction.

Some proposals employ similarity measures for prediction by matching the current and partial vehicle trajectory with the motion patterns learned from historical data [82, 91, 116]. Okamoto et al. [82] proposed a technique to predict short-term vehicle-motion that, in addition to the Markov property, applies a modified DTW distance and the notion of conservation of similarity (CoS). The basic idea of the CoS is that if two vehicles behaved similarly for the previous several seconds, they would move similarly in the next several seconds, implying that the similarity between the vehicles is conserved. Their method assumes that some vehicles behave similarly due to restrictive constraints such as traffic rules. However, the high computational complexity of the modified DTW may prevent its use in large databases.

Rathore et al. [91] proposed the *Traj-clusiVAT-based TP* clustering framework. The framework is based on the Markov model and can make long-term and short-term predictions. The Traj-clusiVAT-based TP framework employs the Traj-clusiVAT algorithm, which is an extension of the clusiVAT clustering algorithm. One of the improvements of the Traj-clusiVAT algorithm is that it implements a new method to calculate, for each cluster, a *representative trajectory*, which is then used to distribute new trajectories to an existing cluster. Because of that, the Traj-clusiVAT-based TP framework can handle big trajectory datasets in dense road networks efficiently.

5.4 Trajectory Outlier Detection

Trajectory outlier detection aims at finding trajectories or sub-trajectories that deviate so much of other trajectories of the dataset [73]. It has a broad application base. For instance, removing outliers is a crucial preprocessing step in support of data mining tasks. Also, the occurrence of similar outliers within a brief time window can indicate abnormal events such as traffic accidents.

Recently, Meng et al. [73] make a review of trajectory outlier detection algorithms from three aspects. First, they classified the algorithms that take into account multi-attributes, such as speed, direction, position, and time. The second viewpoint is if the outlier detection algorithm employs a proper distance function to compute the difference between trajectories efficiently and accurately. Finally, they reviewed studies that aim at enhancing prior outlier detection algorithms regarding space complexity and processing time. In this section, we are interested in trajectory outlier detection algorithms that employ distance measures.

Some of the earlier trajectory outlier detection algorithms compare whole trajectories and thus are not able to detect outliers in sub-trajectories [54]. To solve this problem, Lee et al. [59] presented a two-stage framework for trajectory outlier detection. The first stage breaks a trajectory into line segments, and the second stage identifies outliers from the set of line segments. Besides the framework, they developed an outlier detection algorithm for sub-trajectories, called TRAOD, that employs a line segment Hausdorff distance. A drawback of the TRAOD algorithm is that it only involves the spatial information of the trajectory and ignores the temporal information.

Mirge et al. [75] proposed a straightforward technique that employed the Hausdorff distance to detect outliers in GPS-based trajectory databases. In summary, they calculate the pairwise Hausdorff distance of all trajectories in the dataset and cluster them according to their distances. Then, the approach identifies all trajectories of a cluster as outliers if the cluster size is below a given outlier threshold. Due to its high computational complexity, one can use this approach only to detect outliers in small databases. Besides that, it is unclear how we can choose the value of the outlier threshold.

Wu et al. [115] presented a technique called DB-TOD for the detection of trajectory outliers that models human behavior by extending the maximum entropy inverse reinforcement learning model. The main advantage of their approach is the ability to detect potential outliers from partial trajectories. The DB-TOD learns from historical datasets the most probable route choices and then infers the likely cost of each road segment. This approach is efficient, because the historical dataset is accessed only once for the training.

6 FUTURE RESEARCH DIRECTIONS

Even with the tremendous amount of research work in the field of vehicle trajectory data mining, the increasing availability of the so-called “big trajectory data” still presents a large set of challenges to be addressed, even for primary tasks such as trajectory similarity. Throughout this survey, we discussed some open issues that demand additional investigation related to specific methods and applications. In this section, we give an overview of potential research directions, most of which stem from the challenges of the big data era.

Similarity-based trajectory retrieval, discussed in Section 5.1, despite being one of the most basic applications of a similarity measure, surprisingly is still an open issue, at least when dealing with massive datasets. The problem is twofold. First, accurate similarity measures are time demanding. Second, as stated by Rakthanmanon et al. [89], who developed a known suite of optimizations for the DTW-based time series retrieval, "*There are no known techniques to support similarity search of arbitrary lengths once we have datasets in the billions.*" Although the set of optimizations presented by Rakthanmanon et al. [89] shows the best results in the literature, the time required to process massive datasets is still high (several hours). Therefore, we still lack efficient approaches to tackle the similarity-based search of big trajectory data.

Another big challenge regards the trajectory representation. As discussed in Section 3, the most widely used vehicle trajectory representations are the *GPS-based trajectories* and *road-network constrained* trajectories. The advantage of these representations is their simplicity, which facilitates the development of efficient similarity measures and applications. However, these representations lack the level of detail required by many applications. For instance, none of these representations can provide the vehicle's location/speed at any given timestamp. For a GPS-based trajectory, it is possible to apply preprocessing techniques to increase its granularity (sampling rate), but for some situations, this is not enough yet. Moreover, increasing the trajectory sizes also increases the processing time of data analysis tasks. A promising direction is to develop machine learning-based techniques to represent the trajectories better, such as the novel deep learning trajectory representation proposed by Li et al. [64].

Although this work focuses on vehicle trajectory data, emerging smart mobility solutions will need to tackle trajectories from different data sources and modes of transport. Therefore, besides the development of advanced trajectory representations, there is a need for novel data fusion and visualization techniques to address multi-modal and multi-source trajectories. Furthermore, a relevant open research problem in this subject is the impact of these heterogeneous trajectories on the effectiveness of existing similarity measures.

7 CONCLUSION

The increasing availability of vehicular mobility data attracted many researchers in the past couple of years due to its extensive amount of applications and the demand for algorithms to mine big trajectory data. For some data mining tasks (e.g., clustering and retrieval), the development of efficient and accurate similarity measures plays an important role.

This work surveyed the research efforts related to vehicular trajectory similarity measures. We introduced the basic concepts and representations of vehicle trajectory data. For each representation, we discussed its advantages, disadvantages, and preprocessing techniques to overcome some of the drawbacks. Then, we presented a comprehensive review of methods to compare trajectories. In doing so, we classified each method according to the trajectory representation and features such as metricity, computational complexity, and robustness to noise and local time shift. Finally, we presented an overview of applications of trajectory similarity measures and briefly discussed some open research problems.

REFERENCES

[1] Sajimon Abraham and P. Sojan Lal. 2008. Trigger-based security alarming scheme for moving objects on road networks. In *Intelligence and Security Informatics*, Christopher C. Yang, Hsinchun Chen, Michael Chau, Kuiyu Chang, Sheau-Dong Lang, Patrick S. Chen, Raymond Hsieh, Daniel Zeng, Fei-Yue Wang, Kathleen Carley, Wenji Mao, and Justin Zhan (Eds.). Springer, Berlin, 92–101.

[2] Sajimon Abraham and P. Sojan Lal. 2012. Spatio-temporal similarity of network-constrained moving object trajectories using sequence alignment of travel locations. *Transportation Research Part C: Emerging Technologies* 23 (2012), 109–123. DOI: https://doi.org/10.1016/j.trc.2011.12.008 Data Management in Vehicular Networks.

[3] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* 5, 01–02 (1995), 75–91.

[4] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. *SIGMOD Rec.* 28, 2 (June 1999), 49–60. DOI: https://doi.org/10.1145/304181.304187

[5] Kendall Atkinson. 2002. Modelling a road using spline interpolation. *Reports on Computational Mathematics* 145 (2002), 1–17.

[6] Ricardo A. Baeza-Yates, Walter Cunto, Udi Manber, and Sun Wu. 1994. Proximity matching using fixed-queries trees. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM'94)*. Springer-Verlag, Berlin, Heidelberg, 198–212. Retrieved from http://dl.acm.org/citation.cfm?id=647814.738307.

[7] Ya Ban, Rui Wang, Hongjing Liu, Jing Yuan, Hao Luo, Fuli Yang, Ling Yu, and Xinping Xu. 2018. A moving objects index method integrating GeoHash and quadtree. In *Proceedings of the International Conference on Computer Modeling, Simulation and Algorithm (CMSA'18)*. Atlantis Press, Paris, France, 4.

[8] Gustavo E. Batista, Eamonn J. Keogh, Oben Moses Tataw, and Vinícius M. Souza. 2014. CID: An efficient complexity-invariant distance for time series. *Data Min. Knowl. Discov.* 28, 3 (May 2014), 634–669. DOI: https://doi.org/10.1007/s10618-013-0312-3

[9] L. Bedogni, M. Fiore, and C. Glacet. 2018. Temporal reachability in vehicular networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'18)*. IEEE Computer Society, Washington, DC, 81–89. DOI: https://doi.org/10.1109/INFOCOM.2018.8486393

[10] Clara Benevolo, Renata Paola Dameri, and Beatrice D'Auria. 2016. Smart mobility in smart city. In *Empowering Organizations*, Teresina Torre, Alessio Maria Braccini, and Riccardo Spinelli (Eds.). Springer International Publishing, Cham, 13–28.

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (Aug. 2013), 1798–1828. DOI: https://doi.org/10.1109/TPAMI.2013.50

[12] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer. 2016. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Trans. Intell. Transport. Syst.* 17, 11 (Nov. 2016), 3306–3317. DOI: https://doi.org/10.1109/TITS.2016.2547641

[13] Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. 2018. A survey on trajectory clustering analysis. *CoRR* abs/1802.06971 (2018). arXiv:1802.06971 http://arxiv.org/abs/1802.06971

[14] Tolga Bozkaya and Meral Ozsoyoglu. 1997. Distance-based indexing for high-dimensional metric spaces. *SIGMOD Rec.* 26, 2 (June 1997), 357–368. DOI: https://doi.org/10.1145/253262.253345

[15] C. Celes, F. A. Silva, A. Boukerche, R. M. d. C. Andrade, and A. A. F. Loureiro. 2017. Improving VANET simulation with calibrated vehicular mobility traces. *IEEE Trans. Mobile Comput.* 16, 12 (Dec. 2017), 3376–3389. DOI: https://doi.org/10.1109/TMC.2017.2690636

[16] B. Chapuis and B. Garbinato. 2018. Geodabs: Trajectory indexing meets fingerprinting at scale. In *Proceedings of the IEEE 38th International Conference on Distributed Computing Systems (ICDCS'18)*. IEEE Computer Society, Washington, DC, 1086–1095. DOI: https://doi.org/10.1109/ICDCS.2018.00108

[17] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. 2001. Searching in metric spaces. *ACM Comput. Surv.* 33, 3 (Sept. 2001), 273–321. DOI: https://doi.org/10.1145/502807.502808

[18] Lei Chen. 2005. *Similarity Search over Time Series and Trajectory Data*. Ph.D. Dissertation. University of Waterloo, Waterloo, Ontario, Canada.

[19] L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen. 2015. Efficient metric indexing for similarity search. In *Proceedings of the IEEE 31st International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, 591–602. DOI: https://doi.org/10.1109/ICDE.2015.7113317

[20] Lu Chen, Yunjun Gao, Baihua Zheng, Christian S. Jensen, Hanyu Yang, and Keyu Yang. 2017. Pivot-based metric indexing. *Proc. VLDB Endow.* 10, 10 (June 2017), 1058–1069. DOI: https://doi.org/10.14778/3115404.3115411

[21] Lei Chen and Raymond Ng. 2004. On the marriage of Lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*. VLDB Endowment, 792–803. Retrieved from http://dl.acm.org/citation.cfm?id=1316689.1316758.

[22] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. ACM, New York, NY, 491–502. DOI: https://doi.org/10.1145/1066157.1066213

[23] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. 2010. Searching trajectories by locations: An efficiency study. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'10)*. ACM, New York, NY, 255–266. DOI: https://doi.org/10.1145/1807167.1807197

[24] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A.F. Loureiro. 2020. A distributed and low-overhead traffic congestion control protocol for vehicular ad hoc networks. *Comput. Commun.* 159 (2020), 258–270. DOI: https://doi.org/10.1016/j.comcom.2020.05.032

[25] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro. 2019. DisTraC: A distributed and low-overhead protocol for traffic congestion control using vehicular networks. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*. IEEE Computer Society, Washington, DC, 1–6. DOI: https://doi.org/10.1109/ISCC47284.2019.8969603

[26] R. S. de Sousa, F. S. da Costa, A. C. B. Soares, L. F. M. Vieira, and A. A. F. Loureiro. 2018. Geo-SDVN: A geocast protocol for software defined vehicular networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*. IEEE Computer Society, Washington, DC, 1–6. DOI:https://doi.org/10.1109/ICC.2018.8422755

[27] Ke Deng, Kexin Xie, Kevin Zheng, and Xiaofang Zhou. 2011. *Trajectory Indexing and Retrieval*. Springer, New York, NY, 35–60. DOI:https://doi.org/10.1007/978-1-4614-1629-6_2

[28] Ze Deng, Yangyang Hu, Mao Zhu, Xiaohui Huang, and Bo Du. 2015. A scalable and fast OPTICS for clustering trajectory big data. *Cluster Comput.* 18, 2 (June 2015), 549–562. DOI:https://doi.org/10.1007/s10586-014-0413-9

[29] John M. Dow, R. E. Neilan, and C. Rizos. 2009. The international GNSS service in a changing landscape of global navigation satellite systems. *J. Geodesy* 83, 3 (Mar. 2009), 191–198. DOI:https://doi.org/10.1007/s00190-008-0300-3

[30] Thomas Eiter and Heikki Mannila. 1994. *Computing Discrete Fréchet Distance*. Technical Report. Citeseer.

[31] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Zufle. 2012. Querying uncertain spatio-temporal data. In *Proceedings of the IEEE 28th International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, 354–365. DOI:https://doi.org/10.1109/ICDE.2012.94

[32] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 226–231. http://dl.acm.org/citation.cfm?id=3001460.3001507

[33] M. Maurice Fréchet. 1906. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884–1940)* 22, 1 (Dec. 1906), 1–72. DOI:https://doi.org/10.1007/BF03018603

[34] E. Frentzos, K. Gratsias, and Y. Theodoridis. 2007. Index-based most similar trajectory search. In *Proceedings of the IEEE 23rd International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, 816–825. DOI:https://doi.org/10.1109/ICDE.2007.367927

[35] Tao-Yang Fu and Wang-Chien Lee. 2020. Trembr: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.* 11, 1, Article 10 (Feb. 2020), 25 pages. DOI:https://doi.org/10.1145/3361741

[36] Peter D. Grnwald, In Jae Myung, and Mark A. Pitt. 2005. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. MIT Press, Cambridge, MA.

[37] Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. 2000. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.* 25, 1 (Mar. 2000), 1–42. DOI:https://doi.org/10.1145/352958.352963

[38] Ralf Hartmut Güting, Teixeira de Almeida, and Zhiming Ding. 2006. Modeling and querying moving objects in networks. *VLDB J.* 15, 2 (June 2006), 165–190. DOI:https://doi.org/10.1007/s00778-005-0152-x

[39] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.* 14, 2 (June 1984), 47–57. DOI:https://doi.org/10.1145/971697.602266

[40] B. Han, L. Liu, and E. Omiecinski. 2015. Road-network aware trajectory clustering: Integrating locality, flow, and density. *IEEE Trans. Mobile Comput.* 14, 2 (Feb. 2015), 416–429. DOI:https://doi.org/10.1109/TMC.2013.119

[41] P. Hao, K. Boriboonsomsin, G. Wu, and M. J. Barth. 2017. Modal activity-based stochastic model for estimating vehicle trajectories from sparse mobile sensor data. *IEEE Trans. Intell. Transport. Syst.* 18, 3 (Mar. 2017), 701–711. DOI:https://doi.org/10.1109/TITS.2016.2584388

[42] David Hilbert. 1935. *Über die stetige Abbildung einer Linie auf ein Flächenstück*. Springer, Berlin, 1–2. DOI:https://doi.org/10.1007/978-3-662-38452-7_1

[43] Kathleen Hornsby and Max J. Egenhofer. 2002. Modeling moving objects over multiple granularities. *Ann. Math. Artific. Intell.* 36, 1 (Sep. 2002), 177–194. DOI:https://doi.org/10.1023/A:1015812206586

[44] Weiming Hu, Xi Li, Guodong Tian, Stephen Maybank, and Zhongfei Zhang. 2013. An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 5 (May 2013), 1051–1065. DOI:https://doi.org/10.1109/TPAMI.2012.188

[45] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. 2005. Spatio-temporal similarity analysis between trajectories on road networks. In *Proceedings of the 24th International Conference on Perspectives in Conceptual Modeling (ER'05)*. Springer-Verlag, Berlin, 280–289. DOI:https://doi.org/10.1007/11568346_30

[46] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. 2006. Searching for similar trajectories on road networks using spatio-temporal similarity. In *Proceedings of the 10th East European Conference on Advances in Databases and Information Systems (ADBIS'06)*. Springer-Verlag, Berlin, Heidelberg, 282–295. DOI:https://doi.org/10.1007/11827252_22

[47] Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recogn. Lett.* 31, 8 (2010), 651–666. DOI:https://doi.org/10.1016/j.patrec.2009.09.011

[48] H. Jiang, L. Chang, Q. Li, and D. Chen. 2019. Trajectory prediction of vehicles based on deep learning. In *Proceedings of the 4th International Conference on Intelligent Transportation Engineering (ICITE'19)*. IEEE Computer Society, Washington, DC, 190–195. DOI:https://doi.org/10.1109/ICITE.2019.8880168

[49] M. Jiang and T. Zhao. 2019. Vehicle travel time estimation by sparse trajectories. In *Proceedings of the IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC'19)*, Vol. 1. IEEE Computer Society, Washington, DC, 433–442. DOI: https://doi.org/10.1109/COMPSAC.2019.00069

[50] Maurice G. Kendall. 1948. *Rank Correlation Methods.* Griffin, London.

[51] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowl. Info. Syst.* 7, 3 (Mar. 2005), 358–386. DOI: https://doi.org/10.1007/s10115-004-0154-9

[52] Eamonn J. Keogh and Michael J. Pazzani. 2000. Scaling up dynamic time warping for datamining applications. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00).* ACM, New York, NY, 285–289. DOI: https://doi.org/10.1145/347090.347153

[53] Jiwon Kim and Hani S. Mahmassani. 2015. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. *Transport. Res. Procedia* 9 (2015), 164–184. DOI: https://doi.org/10.1016/j.trpro.2015.07.010

[54] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. 2000. Distance-based outliers: Algorithms and applications. *VLDB J.* 8, 3 (Feb. 2000), 237–253. DOI: https://doi.org/10.1007/s007780050006

[55] M. Kubicka, A. Cela, H. Mounier, and S. Niculescu. 2018. Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intell. Transport. Syst. Mag.* 10, 2 (2018), 150–166. DOI: https://doi.org/10.1109/MITS.2018.2806630

[56] D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. C. Bezdek, and T. C. Havens. 2013. clusiVAT: A mixed visual/numerical clustering algorithm for big data. In *Proceedings of the IEEE International Conference on Big Data.* IEEE Computer Society, Washington, DC, 112–117. DOI: https://doi.org/10.1109/BigData.2013.6691561

[57] Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, X. Wang, and C. Leckie. 2015. A scalable framework for clustering vehicle trajectories in a dense road network. In *Proceedings of the ACM SIGKDD International Workshop Urban Computing.* ACM, New York, NY.

[58] D. Kumar, H. Wu, S. Rajasegarar, C. Leckie, S. Krishnaswamy, and M. Palaniswami. 2018. Fast and scalable big data trajectory clustering for understanding urban mobility. *IEEE Trans. Intell. Transport. Syst.* 19, 11 (Nov 2018), 3709–3722. DOI: https://doi.org/10.1109/TITS.2018.2854775

[59] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08).* IEEE Computer Society, Washington, DC, 140–149. DOI: https://doi.org/10.1109/ICDE.2008.4497422

[60] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: A partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'07).* ACM, New York, NY, 593–604. DOI: https://doi.org/10.1145/1247480.1247546

[61] SangYoon Lee, Sanghyun Park, Woo-Cheol Kim, and Dongwon Lee. 2007. An efficient location encoding method for moving objects using hierarchical administrative district and road network. *Info. Sci.* 177, 3 (Feb. 2007), 832–843. DOI: https://doi.org/10.1016/j.ins.2006.07.016

[62] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* 1, 1 (2014), 1. DOI: https://doi.org/10.1186/s40648-014-0001-z

[63] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning travel time distributions with deep generative model. In *The World Wide Web Conference (WWW'19).* Association for Computing Machinery, New York, NY, 1017–1027. DOI: https://doi.org/10.1145/3308558.3313418

[64] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. 2018. Deep representation learning for trajectory similarity computation. In *Proceedings of the IEEE 34th International Conference on Data Engineering (ICDE'18).* IEEE Computer Society, Washington, DC, 617–628. DOI: https://doi.org/10.1109/ICDE.2018.00062

[65] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18).* Association for Computing Machinery, New York, NY, 1695–1704. DOI: https://doi.org/10.1145/3219819.3220033

[66] Siyuan Liu, Ce Liu, Qiong Luo, Lionel M. Ni, and Ramayya Krishnan. 2012. Calibrating large scale vehicle trajectory data. In *Proceedings of the IEEE 13th International Conference on Mobile Data Management (Mdm 2012) (MDM'12).* IEEE Computer Society, Washington, DC, 222–231. DOI: https://doi.org/10.1109/MDM.2012.15

[67] Qiang Lu, Rencai Wang, Bin Yang, and Zhiguang Wang. 2019. Trajectory splicing. *Knowl. Info. Syst.* 62 (2019), 1–34.

[68] J. Lv, Q. Li, Q. Sun, and X. Wang. 2018. T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction. In *Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp'18).* IEEE Computer Society, Washington, DC, 82–89. DOI: https://doi.org/10.1109/BigComp.2018.00021

[69] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy. 2015. Review on trajectory similarity measures. In *Proceedings of the IEEE 7th International Conference on Intelligent Computing and Information Systems (ICICIS'15).* IEEE, Cairo, Egypt, 613–619. DOI: https://doi.org/10.1109/IntelCIS.2015.7397286

[70] Yingchi Mao, Haishi Zhong, Xianjian Xiao, and Xiaofang Li. 2017. A segment-based trajectory similarity measure in the urban transportation systems. *Sensors* 17, 3 (2017), 14. DOI: https://doi.org/10.3390/s17030524

[71] Pierre-François Marteau. 2009. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 2 (Feb. 2009), 306–318. DOI: https://doi.org/10.1109/TPAMI.2008.76

[72] Marina Meilă. 2006. The uniqueness of a good optimum for K-means. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06).* ACM, New York, NY, 625–632. DOI: https://doi.org/10.1145/1143844.1143923

[73] Fanrong Meng, Guan Yuan, Shaoqian Lv, Zhixiao Wang, and Shixiong Xia. 2018. An overview on trajectory outlier detection. *Artific. Intell. Rev.* (Feb. 2018), 7. DOI: https://doi.org/10.1007/s10462-018-9619-1

[74] Luisa Micó, Jose Oncina, and Rafael C. Carrasco. 1996. A fast branch & bound nearest neighbour classifier in metric spaces. *Pattern Recogn. Lett.* 17, 7 (June 1996), 731–739. DOI: https://doi.org/10.1016/0167-8655(96)00032-3

[75] Vaishali Mirge, Kesari Verma, and Shubhrata Gupta. 2017. Outlier detection in vehicle trajectories. *Int. J. Comput. Appl.* 171 (Aug. 2017), 1–6. DOI: https://doi.org/10.5120/ijca2017915139

[76] Boris Mirkin. 2016. *Clustering: A Data Recovery Approach.* Chapman and Hall/CRC, Boca Raton, FL.

[77] Mirco Nanni and Dino Pedreschi. 2006. Time-focused clustering of trajectories of moving objects. *J. Intell. Info. Syst.* 27, 3 (Nov. 2006), 267–289. DOI: https://doi.org/10.1007/s10844-006-9953-7

[78] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surv.* 33, 1 (Mar. 2001), 31–88. DOI: https://doi.org/10.1145/375360.375365

[79] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. 2013. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *Proc. VLDB Endow.* 7, 3 (Nov. 2013), 205–216. DOI: https://doi.org/10.14778/2732232.2732239

[80] Gustavo Niemeyer. 2008. Geohash. Retrieved from https://en.wikipedia.org/wiki/Geohash.

[81] David Novak, Michal Batko, and Pavel Zezula. 2011. Metric index: An efficient and scalable solution for precise and approximate similarity search. *Info. Syst.* 36, 4 (2011), 721–733. DOI: https://doi.org/10.1016/j.is.2010.10.002

[82] K. Okamoto, K. Berntorp, and S. Di Cairano. 2017. Similarity-based vehicle-motion prediction. In *Proceedings of the American Control Conference (ACC'17).* IEEE Computer Society, Washington, DC, 303–308. DOI: https://doi.org/10.23919/ACC.2017.7962970

[83] Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. 2013. Scalable parallel OPTICS data clustering using graph algorithmic techniques. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13).* ACM, New York, NY, Article 49, 12 pages. DOI: https://doi.org/10.1145/2503210.2503255

[84] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. 2007. Similarity search in trajectory databases. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning (TIME'07).* IEEE Computer Society, Washington, DC, 129–140. DOI: https://doi.org/10.1109/TIME.2007.59

[85] Dieter Pfoser and Christian S. Jensen. 1999. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases (SSD'99).* Springer-Verlag, London, UK, 111–132. Retrieved from http://dl.acm.org/citation.cfm?id=647226.719082.

[86] Dieter Pfoser and Christian S. Jensen. 2003. Indexing of network constrained moving objects. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems (GIS'03).* ACM, New York, NY, 25–32. DOI: https://doi.org/10.1145/956676.956680

[87] M. Purss, R. Gibb, F. Samavati, P. Peterson, J. Rogers, J. Ben, and C. Dow. 2017. *Topic 21: Discrete Global Grid Systems Abstract Specification.* Technical Report. Open Geospatial Consortium, Wayland, MA.

[88] Shaojie Qiao, Changjie Tang, Huidong Jin, Teng Long, Shucheng Dai, Yungchang Ku, and Michael Chau. 2010. PutMode: Prediction of uncertain trajectories in moving objects databases. *Appl. Intell.* 33, 3 (Dec. 2010), 370–386. DOI: https://doi.org/10.1007/s10489-009-0173-z

[89] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12).* ACM, New York, NY, 262–270. DOI: https://doi.org/10.1145/2339530.2339576

[90] S. Ranu, Deepak P, A. D. Telang, P. Deshpande, and S. Raghavan. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *Proceedings of the IEEE 31st International Conference on Data Engineering.* IEEE Computer Society, Washington, DC, 999–1010. DOI: https://doi.org/10.1109/ICDE.2015.7113351

[91] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, and James C. Bezdek. 2018. A scalable framework for trajectory prediction. Retrieved from http://arxiv.org/abs/1806.03582.

[92] Joseph Lee Rodgers and W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *Amer. Stat.* 42, 1 (1988), 59–66. DOI: https://doi.org/10.1080/00031305.1988.10475524 arXiv:https://doi.org/10.1080/00031305.1988.10475524

[93] E. V. Ruiz. 1986. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recogn. Lett.* 4, 3 (July 1986), 145–157. DOI: https://doi.org/10.1016/0167-8655(86)90013-9

[94] Guillermo Ruiz, Francisco Santoyo, Edgar Chávez, Karina Figueroa, and Eric Sadit Tellez. 2013. Extreme pivots for faster metric indexes. In *Similarity Search and Applications*, Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula (Eds.). Springer, Berlin, 115–126.

[95] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.* 11, 5 (Oct. 2007), 561–580. Retrieved from http://dl.acm.org/citation.cfm?id=1367985.1367993.

[96] I. Sanchez, Z. M. M. Aye, B. I. P. Rubinstein, and K. Ramamohanarao. 2016. Fast trajectory clustering using Hashing methods. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'16).* IEEE Computer Society, Washington, DC, 3689–3696. DOI: https://doi.org/10.1109/IJCNN.2016.7727674

[97] Swaminathan Sankararaman, Pankaj K. Agarwal, Thomas Mølhave, Jiangwei Pan, and Arnold P. Boedihardjo. 2013. Model-driven matching and segmentation of trajectories. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'13).* ACM, New York, NY, 234–243. DOI: https://doi.org/10.1145/2525314.2525360

[98] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'03).* ACM, New York, NY, 76–85. DOI: https://doi.org/10.1145/872757.872770

[99] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. 2018. Parallel trajectory similarity joins in spatial networks. *VLDB J.* 27, 3 (June 2018), 395–420. DOI: https://doi.org/10.1007/s00778-018-0502-0

[100] Tomás Skopal, Jaroslav Pokornỳ, and Vaclav Snasel. 2004. PM-tree: Pivoting metric tree for similarity search in multimedia databases. In *Proceedings of the Conference on Advances in Databases and Information Systems (ADBIS'04).* Springer-Verlag, Berlin, 16.

[101] Han Su, Kai Zheng, Jiamin Huang, Haozhou Wang, and Xiaofang Zhou. 2015. Calibrating trajectory data for spatio-temporal similarity analysis. *VLDB J.* 24, 1 (Feb. 2015), 93–116. DOI: https://doi.org/10.1007/s00778-014-0365-y

[102] Na Ta, Guoliang Li, Yongqing Xie, Changqi Li, Shuang Hao, and Jianhua Feng. 2017. Signature-based trajectory similarity join. *IEEE Trans. Knowl. Data Eng.* 29, 4 (Apr. 2017), 870–883. DOI: https://doi.org/10.1109/TKDE.2017.2651821

[103] Pang-Ning Tan. 2018. *Introduction to Data Mining*. Pearson Education India.

[104] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. 2009. Searching for similar trajectories in spatial networks. *J. Syst. Softw.* 82, 5 (May 2009), 772–788. DOI: https://doi.org/10.1016/j.jss.2008.11.832

[105] Dalia Tiesyte and Christian S. Jensen. 2008. Similarity-based prediction of travel times for vehicles traveling on known routes. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'08).* ACM, New York, NY, Article 14, 10 pages. DOI: https://doi.org/10.1145/1463434.1463452

[106] Kevin Toohey and Matt Duckham. 2015. Trajectory similarity measures. *SIGSPATIAL Spec.* 7, 1 (May 2015), 43–50. DOI: https://doi.org/10.1145/2782759.2782767

[107] Caetano Traina, Jr., Roberto F. Filho, Agma J. Traina, Marcos R. Vieira, Christos Faloutsos, and Christos Faloutsos. 2007. The omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. *VLDB J.* 16, 4 (Oct. 2007), 483–505. DOI: https://doi.org/10.1007/s00778-005-0178-0

[108] Goce Trajcevski. 2011. *Uncertainty in Spatial Trajectories.* Springer, New York, NY, 63–107. DOI: https://doi.org/10.1007/978-1-4614-1629-6_3

[109] Goce Trajcevski, Alok Choudhary, Ouri Wolfson, Li Ye, and Gang Li. 2010. Uncertain range queries for necklaces. In *Proceedings of the 11th International Conference on Mobile Data Management (MDM'10).* IEEE Computer Society, Washington, DC, 199–208. DOI: https://doi.org/10.1109/MDM.2010.76

[110] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. 2004. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.* 29, 3 (Sept. 2004), 463–507. DOI: https://doi.org/10.1145/1016028.1016030

[111] Michail Vlachos, Dimitrios Gunopoulos, and George Kollios. 2002. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02).* IEEE Computer Society, Washington, DC, 673. Retrieved from http://dl.acm.org/citation.cfm?id=876875.878994.

[112] Hua Wang, Changlong Gu, and Washington Yotto Ochieng. 2019. Vehicle trajectory reconstruction for signalized intersections with low-frequency floating car data. *J. Adv. Transport.* 2019 (2019), 14.

[113] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. 2013. An effectiveness study on trajectory similarity measures. In *Proceedings of the 24th Australasian Database Conference - Volume 137 (ADC'13).* Australian Computer Society, Darlinghurst, Australia, 13–22. Retrieved from http://dl.acm.org/citation.cfm?id=2525416.2525418.

[114] J. Won, S. Kim, J. Baek, and J. Lee. 2009. Trajectory clustering in road network environment. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*. IEEE Computer Society, Nashville, TN, 299–305. DOI: https://doi.org/10.1109/CIDM.2009.4938663

[115] Hao Wu, Weiwei Sun, and Baihua Zheng. 2017. A fast trajectory outlier detection approach via driving behavior modeling. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'17)*. ACM, New York, NY, 837–846. DOI: https://doi.org/10.1145/3132847.3132933

[116] M. Wu, X. Zeng, Y. Lin, and Y. Wen. 2019. Vehicle trajectory prediction models by combining communication data. In *Proceedings of the 11th International Conference on Advanced Computational Intelligence (ICACI'19)*. IEEE Computer Society, Washington, DC, 113–117. DOI: https://doi.org/10.1109/ICACI.2019.8778574

[117] Ying Xia, Guoyin Wang, Xu Zhang, Gyoung Bae Kim, and Hae-Young Bae. 2011. Spatio-temporal similarity measure for network constrained trajectory data. *Int. J. Comput. Intell. Syst.* 4 (2011), 1070–1079.

[118] D. Yao, G. Cong, C. Zhang, and J. Bi. 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE'19)*. IEEE Computer Society, Washington, DC, 1358–1369. DOI: https://doi.org/10.1109/ICDE.2019.00123

[119] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi. 2017. Trajectory clustering via deep representation learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'17)*. IEEE Computer Society, Washington, DC, 3880–3887. DOI: https://doi.org/10.1109/IJCNN.2017.7966345

[120] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2018. Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile. *Data Min. Knowl. Discov.* 32, 1 (Jan. 2018), 83–123. DOI: https://doi.org/10.1007/s10618-017-0519-9

[121] Peter N. Yianilos. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 311–321. Retrieved from http://dl.acm.org/citation.cfm?id=313559.313789.

[122] Ge Yu, Yu Gu, Jianzhong Qiao, Lei Chen, and Chuanfei Xu. 2013. Interval reverse nearest neighbor queries on uncertain data with markov correlations. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE'13)*. IEEE Computer Society, Washington, DC, 170–181. DOI: https://doi.org/10.1109/ICDE.2013.6544823

[123] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. 2017. A review of moving object trajectory clustering algorithms. *Artif. Intell. Rev.* 47, 1 (Jan. 2017), 123–144. DOI: https://doi.org/10.1007/s10462-016-9477-7

[124] H. Yuan and G. Li. 2019. Distributed in-memory trajectory similarity search and join on road network. In *Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE'19)*. IEEE Computer Society, Washington, DC, 1262–1273. DOI: https://doi.org/10.1109/ICDE.2019.00115

[125] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3, Article 29 (May 2015), 41 pages. DOI: https://doi.org/10.1145/2743025

Received September 2019; revised February 2020; accepted June 2020