A Survey on Trajectory Data Management, Analytics, and Learning

SHENG WANG, New York University, United States
ZHIFENG BAO and J. SHANE CULPEPPER, RMIT University, Australia
GAO CONG, Nanyang Technological University, Singapore

Recent advances in sensor and mobile devices have enabled an unprecedented increase in the availability and collection of urban trajectory data, thus increasing the demand for more efficient ways to manage and analyze the data being produced. In this survey, we comprehensively review recent research trends in trajectory data management, ranging from trajectory pre-processing, storage, common trajectory analytic tools, such as querying spatial-only and spatial-textual trajectory data, and trajectory clustering. We also explore four closely related analytical tasks commonly used with trajectory data in interactive or real-time processing. Deep trajectory learning is also reviewed for the first time. Finally, we outline the essential qualities that a trajectory data management system should possess to maximize flexibility.

CCS Concepts: • **Information systems** → *Data management systems*;

Additional Key Words and Phrases: Trajectory, storage system, similarity search, urban analytics, deep learning

ACM Reference format:
Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2021. A Survey on Trajectory Data Management, Analytics, and Learning. *ACM Comput. Surv.* 54, 2, Article 39 (March 2021), 36 pages.
https://doi.org/10.1145/3440207

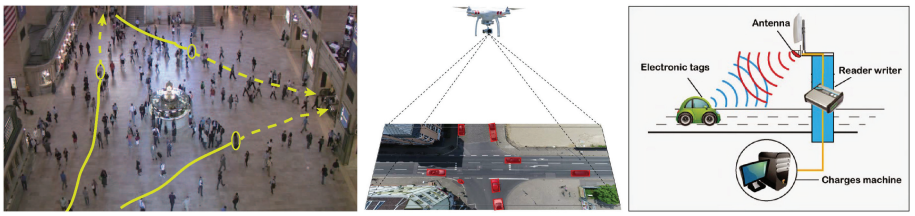
1 INTRODUCTION

In the late 1990s, Global Positional System (GPS) satellites began transmitting two additional signals to be used for civilian (non-military) applications to improve aircraft safety.[1] Demand for GPS-based navigation has grown steadily ever since, and privately owned vehicles are now the dominant consumers of this technology. An increased reliance on GPS-equipped smartphones has

[1] https://www.pcworld.com/article/2000276/a-brief-history-of-gps.html.

Zhifeng Bao is supported in part by ARC DP200102611, DP180102050, and a Google Faculty Award. J. Shane Culpepper is supported in part by ARC DP190101113. Gao Cong is supported in part by a MOE Tier-2 grant MOE2019-T2-2-181, and a MOE Tier-1 grant RG114/19.
Authors' addresses: S. Wang, New York University, United States; email: swang@nyu.edu; Z. Bao and J. S. Culpepper, RMIT University, Australia; emails: {zhifeng.bao, shane.culpepper}@rmit.edu.au; G. Cong, Nanyang Technological University, Singapore; email: gaocong@ntu.edu.sg.

0360-0300/2021/03-ART39 $15.00
https://doi.org/10.1145/3440207



(a) Human trajectories in surveillance [208] (b) Monitoring cars by UAV [23] (c) Car localization by RFID [3]

Fig. 1. New localization techniques that can capture trajectory data more proactively than using GPS.

also led to a rise in the use of location-based services (LBS), such as ride-sharing and social-network check-ins.

In addition to GPS data, other sensor devices such as traffic surveillance cameras, unmanned aerial vehicles (UAV),[2] and Radio-frequency identification (RFID) can also collect location data without deploying battery-dependent receiving devices to track objects, as shown in Figure 1. Such devices can accurately report precise locations and track objects proactively and continuously.

1.1 Trajectory Data

Devices that track moving objects often generate a series of ordered points representing a *trajectory*. More formally, a trajectory T is composed of two or more *spatial-only* points and defined as:

Definition 1 (Point). A point $p = \{x, y, t\}$ records the latitude x, the longitude y at timestamp t.

Definition 2 (Trajectory). A trajectory T is a sequence of points $\{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$.

The number of points in a trajectory T is denoted as the *length* of the trajectory, and the *sampling rate* is the number of samples per second (or other time units). Additional information γ can be included with each point p in a trajectory T generated from the location-based service.[3] For example, textual data can be integrated into a trajectory from social network check-in data or travel blogs and have been referred to as *spatial-textual trajectories* [19, 87], *semantic trajectories* [193], or *symbolic trajectories* [73]. This survey will focus primarily on these two closely related forms of trajectory data as solutions and applications in recent research work commonly overlap.

Survey Scope. We focus primarily on urban trajectories. Other studies of domain specific data such as aircraft [16] are not covered in this work. Note that the main difference between a moving object [135] and a trajectory is the distinction between a pointwise sequence and a continuous projection, and each of them often employs fundamentally different storage and algorithmic solutions.

Public Trajectory Datasets. Table 1 shows several existing urban trajectory datasets, which can be broadly divided into three groups: humans, vehicles (car, truck, train, bus, tram, etc.), others (animals and hurricanes). We can observe that human derived trajectory data (including vehicles) are a common source of trajectory data and are currently the largest source of trajectory data. For example, 1.1 billion (seventh row) individual taxi trips were recorded in New York City from January 2009 through June 2015. Other datasets including the movement of animals and hurricanes

[2] https://www.gpsworld.com/in-mit-project-drones-map-without-gps/.
[3] For spatial-only trajectories, we can set $\gamma = \emptyset$. Note that γ can also be attached to the whole trajectory directly.

Table 1. Publicly Available Trajectory Datasets

Categorization	Type	Exemplary Datasets	#Trajectories	Applications	
Human	GPS tracking	GeoLife [206]	17,621	Human mobility [194]	Urban area
	Check-in	Foursquare [19] Gowalla [39]	104,478	Tourism planning [165] Mobility prediction [58]	
	Online sharing	OpenStreetMap [9]	8.7 million	Commuting analytics	
	Video surveillance	Grand central station [208]	20,000	Crowd behavior	
	COVID-19	KCDC [5], JHU [8]	<100	Disease control	
Vehicle	Taxi trips	T-drive [192], Porto [2]	1.7 million	Traffic monitoring	
	Taxi trip-requests	NYC [10], Chicago [10]	1.1 billion	Ride sharing	
	Traffic cameras	NGSIM [4]	—	Traffic simulation	
	Drones	HighD [90], inD [23]	110,000	Traffic prediction	
	Self-driving	Argoverse [28], ApolloScape [118]	300,000	Trajectory forecasting	
	Trucks	Greece Trucks [6]	1100	Pattern mining	
Others	Hurricanes	Atlantic hurricanes [7]	1740	Disaster detection	
	Animals	Zebranet [7], Movebank [1]	33	Animal behavior [92]	

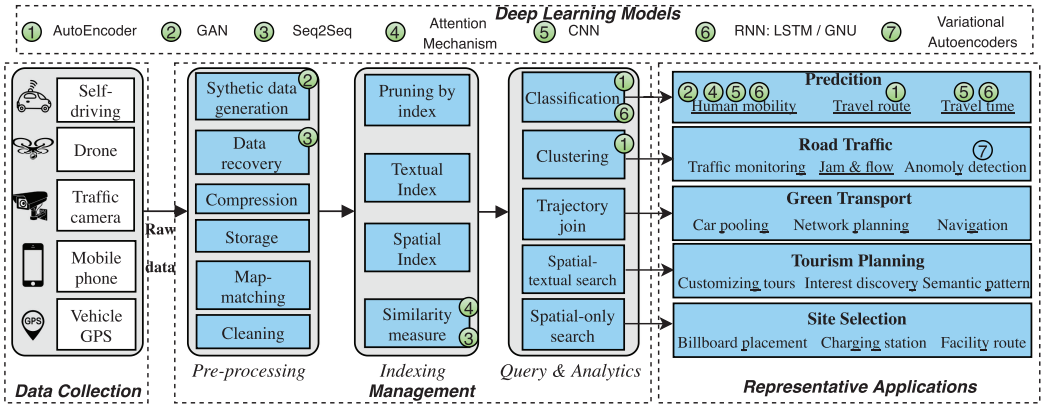


Fig. 2. Overview of trajectory data collection, management, representative applications, and widely used deep learning models (indicated by the green labels, and the full names of models can be found in Section 7).

have relatively few available trajectories, making such collections much more tractable to process and analyze.

1.2 An Overview

Figure 2 provides an overview of trajectory data management, analytics, and learning. Briefly, a trajectory data management and analytic system has several fundamental components:

- **Cleaning:** Common techniques to clean common sensor errors in raw trajectories and normalize sampling rates.
- **Storage:** Open-source, commercial, and other commonly deployed storage formats that are used to represent trajectory data.
- **Similarity Measures:** The most commonly used measures for top-k similarity search, similarity joins, and clustering.
- **Indexing:** State-of-the-art solutions for indexing spatial and spatial-textual trajectory data, as well as effective pruning solutions to improve performance.

Table 2. Survey Contributions Relative to Prior Trajectory Surveys

Survey	Datasets	Commercial systems	Similarity measures	Pruning mechanisms	Indexing	Processing pipeline decomposition	Deep learning
2011 [207]	✗	✗	Spatial	Spatial	Spatial	✗	✗
2013 [127]	✗	✗	✗	✗	✗	✗	✗
2015 [204]	Only GPS	✗	Spatial	✗	Spatial	Mining	✗
2016 [59]	✗	✗	✗	✗	✗	✗	✗
2018 [133]	✗	✗	✗	✗	Spatial	✗	✗
This survey	✓	✓	✓	✓	✓	✓	✓

- **Query and Analytics:** The most common trajectory queries, including range queries, top-k similarity search, and trajectory joins.
- **Upstream Urban Applications:** The four most common offline urban applications of trajectory data with problem formulations and solutions.

1.3 New Expository Contributions Found from This Survey

Managing large-scale trajectory data has many important challenges in efficiency and scalability due to its size and diversity. A survey covering state-of-the-art and open problems in each stage, from data pre-processing all the way to upper-stream applications, is valuable to both research and industrial communities. While several surveys on trajectory data exist, none review trajectory data management and applications as outlined in Figure 2. Prior surveys have covered trajectory data mining [204], semantic trajectory modeling [127], and related applications [133]. Table 2 compares and contrasts prior surveys with our own, highlighting the new areas covered. A couple of surveys from closely related areas that do partially overlap with the content of this survey include: trajectory similarity measures [146], trajectory privacy [72], and trajectory classification [22]. In these instances, this survey includes more recent work from these important areas. The closest related review work is the textbook *Computing with Spatial Trajectories* [207], published in 2011. Current techniques have advanced significantly since the early 2010s, including the quickly growing area of spatial-textual trajectories, which were not covered previously. The key contributions of this survey can be summarized as follows:

- **Scalability and Storage.** This survey comprehensively reviews trajectory storage, which is commonly required in scalable offline analytics.
- **Processing Pipeline Decomposition.** We provide insights into the connections among all components in the entire data processing pipeline. For the backend, we compare trajectory storage systems (academic and commercial); for middleware, we compare similarity measures, queries, and index structures; for upstream trajectory applications, we propose a taxonomy for key operators used across multiple tasks.
- **Representative Applications.** Based on a comprehensive review of recent research in SIGMOD, PVLDB, ICDE, KDD, SIGSPATIAL, VLDBJ, TKDE, IJGIS, and TITS, we focus on urban applications requiring real-time responses or timely decision making, including (1) road traffic, (2) green transport, (3) tourism planning, and (4) site selection. These applications depend heavily on downstream analytics tools. For example, clustering is commonly used to generate candidate routes and to generate a fixed number of results for traffic monitoring.

Table 3. An Overview of Existing Trajectory Database Systems

System	Data	Indexing	Range	*k*NN	Similarity Measure
TrajStore [44]	Trajectory	Quad-tree	✓	✗	✗
SharkDB [157]	Trajectory	Frame structure	✓	✓	Point-to-trajectory
UITraMan [53]	Trajectory	R-tree	✓	✓	Point-to-trajectory
TrajMesa [97]	Trajectory	Z-order curve	✓	✓	Point-to-trajectory
DITA [141]	Trajectory	R-tree	✓	✓	Trajectory-to-trajectory
Torch [166]	Trajectory	Grid-index	✓	✓	Trajectory-to-trajectory
Oracle	LineString	R-tree	✓	✓	Point-to-point
PostGIS	LineString	R-tree	✓	✓	Point-to-point
SQL Server	LineString	Grid-index	✓	✓	Point-to-point
GeoMesa	LineString	Z-order curve	✓	✓	Point-to-point
Tile38	LineString	R-tree	✓	✓	Point-to-point
GeoSpark [189]	LineString	R-tree	✓	✓	Point-to-point
SpatialSpark [188]	LineString	R-tree	✓	✗	✗

- **Deep Trajectory Learning.** New trends on using deep learning for various trajectory data analytics tasks, including trajectory recovery and generation, trajectory representation and similarity search, clustering, classification, anomaly detection, and trajectory prediction.
- **Perspectives on Future Trajectory Data Management Challenges.** We conclude with a list of key challenges and associated open problems for researchers working in the area.

1.4 Outline of This Survey

The remainder of the survey is organized as follows: Section 2 presents techniques for trajectory data storage and pre-processing. Section 3 compares common trajectory similarity measures. Section 4 illustrates queries and indexing techniques for spatial-only trajectory data and spatial-textual trajectory data, respectively. Section 5 reviews trajectory clustering from a data management perspective. Section 6 presents a taxonomy of four application types commonly encountered in trajectory data management. Section 7 introduces the latest progress in deep trajectory learning. Section 8 discusses the future of trajectory data system management. Section 9 concludes the article.

2 MANAGEMENT SYSTEMS AND PRE-PROCESSING

2.1 Trajectory Representation and Storage

A list of coordinates denoted by two floats (e.g., "−37.807302, 144.963242") are the traditional representation of a spatial point location. Two or more such points can be used to represent a trajectory for storage, search, and analytics, and this is the most common method in both research and industry. Table 3 shows several representative research systems specifically designed for trajectory data, as well as several commercial systems that were not necessarily designed for only trajectory data but can store and manipulate trajectory data through module extensions. Since commercial systems are designed to handle many different types of data, they can incur additional cost overheads.

In the academic field, several trajectory storage systems have been built to manage trajectory data [44, 53, 141, 157, 166, 183] and store trajectories as a set of points. These trajectory management systems [44, 157, 183] only support a single storage format, and often only a single query type such as a range query (finding trajectories in a rectangle). More recently, distributed systems

[53, 97, 141] have been developed to store point-based trajectory datasets scalably and can perform more advanced queries such as *k*NN search over trajectories.

Existing commercial or open-source systems (e.g., a general database such as Oracle Spatial,[4] SQL Server,[5] and MySQL[6] or geospatial databases such as ArcGIS,[7] Tile38,[8] PostGIS,[9] and GeoMesa[10]) can also be extended for trajectory management. For example, *LineString* defined a portable data-interchange format-GeoJSON[11] that is a geographical representation for an array of point coordinates. A trajectory can be stored as a *LineString* using this format. *GPX* (the GPS Exchange Format) [61] is another format used by OpenStreetMap for routing and tracking. Figures 10 and 11 in Appendix [162] show an example of these two formats. However, operator support for LineString and GPX data is limited, e.g., return the number of points, the starting the ending point, or the *i*th point. Range and ***k*NN** queries are supported over all points but not the trajectories themselves.

SpatialSpark [188] and GeoSpark [189] are two representative open-source systems from the research field of spatial databases, which also support the storage and querying of LineString. Range and *k*NN queries are supported for the point data but not trajectories. Nevertheless, these spatial database systems could be extended to support any number of trajectory search operators in the future, since trajectory data are extension point data. In Section 4.2, we will review indexing techniques for point data in more detail.

2.2 Trajectory Pre-processing

Trajectory Cleaning. Since trajectory data are often collected with GPS devices that have an average user range error of 7.8 m (25.6 ft), with 95% probability, trajectory point data are *noisy* [204]. Moreover, the sampling rate of GPS devices varies by application. For example, two vehicles following an identical path can produce trajectories that contain a different number of sampled points. These factors can directly affect distance or similarity computations and degrade result quality. Hence, when processing raw trajectory datasets, data cleaning is not only beneficial but also in certain circumstances a requirement. *Data segmentation* [26], *calibration* [147], and *enrichment* [13] are the three most common cleaning techniques for trajectory data. Specifically, segmenting data splits a long trajectory into several short trajectories. For example, analysis of a single taxi "trip" makes more sense than analyzing the movements of the taxi for an entire day. Buchin et al. [26] addressed the problem of segmenting a trajectory based on spatiotemporal criteria, including location, heading, speed, velocity, curvature, sinuosity, curviness, and shape.

Su et al. [147] proposed a calibration method that transforms a heterogeneous trajectory dataset to one with (almost) unified sampling strategies, such that the similarity between trajectories can be computed more accurately. For noisy points, calibration identifies outliers and adds statistical correction instead of filtering out these points directly. Such calibration also plays an important role in precise similarity computation [132], which we will introduce later. Liu et al. [109] further considered the constraint of road network topology, geometry information, and historical information to re-calibrate noisy data points. For trajectories of sparse data points, Alvares et al. [13] proposed that the data can be enriched with additional points and semantic information about

[4] https://www.oracle.com/technetwork/database/options/spatialandgraph/overview/index.html.
[5] https://docs.microsoft.com/en-us/sql/relational-databases/spatial/spatial-data-sql-server.
[6] https://dev.mysql.com/doc/refman/5.7/en/gis-linestring-property-functions.html.
[7] http://www.arcgis.com/index.html.
[8] https://github.com/tidwall/tile38.
[9] https://postgis.net/.
[10] https://www.geomesa.org/.
[11] https://en.wikipedia.org/wiki/GeoJSON.

the types of visited places. Conversely, trajectory simplification [196] can be applied to remove redundant points.

Trajectory Compression. Existing trajectory compression techniques can be divided into two groups: simplification based and road network based. Excluding extra points is a common space reduction method when the sampling rate is high and is commonly referred to as *trajectory simplification* [196] and also used in trajectory cleaning as discussed above. Removing points can reduce size but must be used carefully as it can also reduce the resolution when analyzing the data. To alleviate this problem, an error ratio can be applied to bound the loss for specific computations and data processing operations that are predefined before data simplification [113].

Alternatively, a road network can be used to enable better compression [144] with little or no reduction in quality, for certain types of data. Each trajectory is projected onto a road network as a sequence of road segments. Next, each road segment is uniquely encoded using *Huffman coding* [43]. Each trajectory is succinctly represented as a concatenation of the codewords and is significantly more effective than attempting to compress the raw floating point value pairs (latitude and longitude). Further, string compression techniques [184] can be used directly on the trajectory data, and any temporal information can also be succinctly stored using these techniques.

Map-matching. Using a road network, map-matching [114, 124] projects a raw trajectory onto a real path and supports cleaning and compression. A road network is modeled as a weighted graph [43], where a road segment is an edge, and its weight represents the length of this road. Consequently, a path mapped from a raw trajectory is a set of connected edges in a road network.

Definition 3 (Road Network). A road network is a directed graph $G = (V, E)$, where V is a set of vertices v representing the intersections and terminal points of the road segments, E is a set of edges e representing road segments, and each vertex has a unique id allocated from 1 to $|V|$.

Definition 4 (Mapped Trajectory). Given a raw trajectory T and a road network G, we map T to a road network path P, which is composed of a set of connected edges in G, such that $T : e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_m$, and denoted as T containing a series of edge IDs.

Map-matching techniques should consider both effectiveness and efficiency. Efficient algorithms enable us to find the nearest road segments for each point in the trajectory and are required to connect the candidate road segment to the best path. Uncertainty is often the biggest hurdle and is a reflection of parameter selection, which would reduce the number of potential candidate paths during shortest path search in G. For effectiveness, mapping a trajectory to the true traversal path of a vehicle when errors in the data can be substantial is a key limiting factor.

To measure effectiveness, a ground truth is required and is a problem in its own right. There are three common ways to generate a ground truth for the map-matching problem: (1) using real vehicles equipped with GPS devices on the road [124], (2) human adjudication [114], and (3) simulated GPS sampling [83], which is the least costly approach.

3 TRAJECTORY SIMILARITY MEASURES

This section will introduce the most widely used similarity measures to compare two trajectories. Figure 3 presents two different types of query-to-trajectory similarity measures. Labels on each dashed line denote the distance from a query point to a point in the trajectory.

3.1 Point-to-Trajectory

A k Best Connected Trajectories (kBCT) search [38, 131, 149] is arguably the most commonly employed formulation of point-to-trajectory similarity search. To compute the result, each query

(a) Points-to-trajectory (b) Trajectory-to-trajectory

Fig. 3. Two types of similarity measures: (a) a query is a set of points; (b) a query is a trajectory.

Table 4. An Overview of Existing Trajectory-to-trajectory Similarity Measures

Type	Measures	Complexity	Metric	No Parameter	Robustness		
					GPS Error	Sampling Rate	Point Shift
Curve	Hausdorff [134]	$O(n^2)$	✓	✓	✗	✗	✗
	DFD [54]	$O(n^2)$	✓	✓	✗	✗	✗
Real	DTW [187]	$O(n^2)$	✗	✓	✗	✗	✗
	ERP [33]	$O(n^2)$	✓	✗	✗	✗	✗
Edit	EDR [34]	$O(n^2)$	✗	✗	✓	✗	✗
	LCSS [154]	$O(n^2)$	✗	✗	✓	✗	✗
Temporal	DISSIM [62]	$O(n^2)$	✗	✗	✗	✓	✗
	EDwP [132]	$O(n^2)$	✗	✓	✗	✓	✗
Segment	LORS [166], LCRS [190]	$O(n^2)$	✗	✓	✓	✓	✓
	EBD [164]	$O(nlog(n))$	✓	✓	✓	✓	✓

point $q \in Q$ is paired with the closest point $p \in T$, and the sum of the distances for all pairs are aggregated to generate the final distance (similarity) score: $d_{kBCT}(Q,T) = \sum_{q \in Q} \min_{p \in T} d(q,p)$.

For example, applying $d_{kBCT}(Q,T)$ to Figure 3 yields $d_{kBCT}(Q_1,T_1) = 1 + 1.7 + 2.2 = 4.9$, computed as (a) is the sum of the distance between every query point in Q to its nearest neighbor in T (the three dotted lines). When there is only a single query point, the problem reduces to a ***k*NN** query [53, 157]. Since this solution requires every point in every trajectory in a collection to be considered to locate the the nearest point for all $q \in Q$, the complexity of a brute force solution is quadratic. Note that this distance measure can only be computed when the query Q is a set of points and does not obey the symmetry rule, i.e., $d(Q,T) \neq d(T,Q)$.

An alternative to summing the distances between all nearest neighbor pairs is another well-known measure called the closest-pair distance (CPD) [204], which minimizes distance as follows: $d_{CPD}(Q,T) = \min_{q \in Q} \min_{p \in T} d(q,p)$. Then $d_{CPD}(Q_1,T_1) = 1$ in Figure 3. When compared with d_{kBCT}, d_{CPD} is more robust when erroneous points exist in a trajectory.

3.2 Trajectory-to-Trajectory

One weakness in point-to-trajectory measures is that they do not capture the true ordering of points. However, many applications of trajectory search expect an ordering constraint to hold. That is, the similarity measure should satisfy a *local time shifting* [34] constraint.

3.2.1 Pointwise Measures. We now provide a taxonomy of similarity measures that can be used to compare and contrast each of them. As shown in Table 4, we broadly divide the most commonly used similarity measures into five categories: curve based, real-distance, edit-distance, temporal-aware, and segment based. Each of these will now be described in more detail, and the example shown in Figure 3(b) will be used to more concretely illustrate important properties for each measure.

Since a trajectory can also be viewed as a *geometric curve*, Hausdorff distance [134] measures the separation between two subsets of a metric space adversarially, without imposing an ordering

constraint. Informally, it is the maximum of the distances between each point in a set Q to the *nearest* point in the reference set (trajectory T): $d_{Hau}(Q,T) = \max_{q\in Q} \min_{p\in T} d(p,q)$.[12]

Discrete Fréchet Distance (DFD) [54] extends the Hausdorff distance to account for location and ordering of the points along the curve, as shown in Equation (1), where p and q are the head (first) point of T and Q, respectively, and T_h and Q_h are the sub-trajectory excluding the head p and q. Accordingly, using the example in Figure 3(b), d_{Hau} and d_{DFD} will return the same value 3.1, since the matching shown in the figure not only finds the nearest neighbor for each point but also obeys local time shifting (the order constraint discussed above). If any two points in T_5 swap locations, then d_{Hau} does not change while d_{DFD} could,

$$d_{DFD}(Q,T) = \begin{cases} d(p,q), \text{ if } Q = \{q\} \textit{ and } T = \{p\} \\ \infty, \text{ if } Q = \emptyset \textit{ or } T = \emptyset \\ \max\{d(q,p), \min\{d_{DFD}(Q_h,T_h),\ d_{DFD}(Q,T_h), d_{DFD}(Q_h,T)\}\}, \text{ otherwise} \end{cases}. \quad (1)$$

Since a trajectory can be also viewed as a time series, many similarity measures originally designed for time-series search can be leveraged in trajectory related problems [204]. Time-series measures that have been applied to trajectory data include Dynamic Time Warping (DTW) [187], Longest Common Subsequence (LCSS) [154], Edit Distance for Real sequences (EDR) [34], and Edit distance with Real Penalty (ERP) [33]. DTW computes the distance based on the sum of minimum distances, instead of choosing the maximum as in Discrete Fréchet Distance, as shown in Equation (2). To compute DTW for Figure 3(b), every possible pairing of points in T_5 and Q_3 are compared in order, and the sum of the minimum of each pairing $d_{DTW}(Q_3,T_5) = 1.2 + 3 + 1 + 2.2 + 3.1 = 10.5$,

$$d_{DTW}(Q,T) = \begin{cases} d(p,q), \text{ if } Q = \{q\} \textit{ and } T = \{p\} \\ \infty, \text{ if } Q = \emptyset \textit{ or } T = \emptyset \\ d(q,p) + \min\{d_{DTW}(Q_h,T_h),\ d_{DTW}(Q,T_h),\ d_{DTW}(Q_h,T)\}, \text{ otherwise} \end{cases}. \quad (2)$$

Instead of computing the real distance, computing an edit distance (0 or 1) can be more robust for noisy data as outliers can heavily impact DTW-based comparisons. To judge whether two points are matched, edit-distance-based measures set a range threshold τ, i.e., $match(p,q) = true$, if $|p_x - q_x| \le \tau$ and $|p_y - q_y| \le \tau$, where $||$ denotes the absolute value and p_x and p_y denote the latitude and longitude of point p, respectively. Now LCSS and EDR can be defined as shown in Equation (3) and Figure 4. The main difference between LCSS and EDR is that LCSS measures the similarity between two trajectories, while EDR measures the *dissimilarity*. For example, consider $\tau = 0.5$ for each of these in Figure 3(b). The result is no matching point for $d_{LCSS}(Q_3,T_5) = 0$ and $d_{EDR}(Q_3,T_5) = 4$. If $\tau = 5$, then every point can be matched, and $d_{LCSS}(Q_3,T_5) = 4$ and $d_{EDR}(Q_3,T_5) = 0$. Hence, both measures are sensitive to the hyperparameter τ,

$$d_{LCSS}(Q,T) = \begin{cases} 0, \text{ if } Q = \emptyset \text{ or } T = \emptyset \\ 1 + d_{LCSS}(Q_h,T_h), \text{ if } match(p,q) = true \\ \max\{d_{LCSS}(Q,T_h),\ d_{LCSS}(Q_h,T)\}, \text{ otherwise} \end{cases}, \quad (3)$$

$$d_{EDR}(Q,T) = \begin{cases} |Q| \textit{ or } |T|, \text{ if } T = \emptyset \textit{ or } Q = \emptyset \\ \min\{d_{EDR}(Q_h,T_h), d_{EDR}(Q,T_h) + 1,\ d_{EDR}(Q_h,T) + 1\}, \text{ if } match(p,q) = false \\ \min\{1 + d_{EDR}(Q_h,T_h), d_{EDR}(Q,T_h) + 1,\ d_{EDR}(Q_h,T) + 1\}, \text{ otherwise} \end{cases}. \quad (4)$$

[12]Note that $d_{Hau}(Q,T)$ is the directed distance from Q to T. A more general definition that obeys the symmetry and triangle inequality would be: $\max\{d_{Hau}(Q,T), d_{Hau}(T,Q)\}$, and is metric as shown in Table 4.

Since these measures (DTW, LCSS, and EDR) do not obey the triangle inequality, which is an essential requirement for *metric* space pruning, all of the time-series similarity measures are *non-metric* except for ERP [33]. In ERP, a point g can be any fixed point in a metric space and is used as the reference origin point $g = \{0, 0\}$. Changing g can also change distance scores similarly to LCSS, which can result in ***k*NN**-based algorithms being non-deterministic,

$$d_{ERP}(Q,T) = \begin{cases} \sum_{p \in T} d(g,p), & \text{if } Q = \emptyset \\ \sum_{q \in Q} d(q,g), & \text{if } T = \emptyset \\ \min\{d_{ERP}(Q_h,T_h) + d(q,p), d_{ERP}(Q,T_h) + d(g,p), d_{ERP}(Q_h,T) + d(q,g)\}, & \text{otherwise} \end{cases}. \tag{5}$$

3.2.2 Temporal-aware Pointwise Similarity. The aforementioned six measures are the representative of commonly deployed similarity measures for spatial-only trajectory applications. In addition to spatial information, temporal information is also an important factor in accurate sample rate calibration.[13] Frentzos et al. [62] proposed a measure called DISSIM to compute the dissimilarity: $d_{DISSIM} = \int_{t_1}^{t_n} d(Q_t, T_t)dt$. Here, Q_t and T_t denote the points of Q and T at timestamp t in the range $[t_1, t_n]$. The core idea is to integrate time w.r.t. Euclidean distance for trajectories occurring in the same period. DISSIM can resolve various sampling rate problems commonly encountered through integration, but can also be computationally expensive. To alleviate this issue, EDwP [132] calibrates trajectory manipulations by adding or removing points to align the sampling rate between any two trajectories, and computes similarity as described for EDR.

3.2.3 Segment-based Similarity. Temporal-aware pointwise methods can compute similarity in much richer trajectory data, but results often have precision issues unless sample-rate calibration is applied. Another alternative is to convert trajectories to segments. This approach has been shown to reduce sample mismatch effects as well as reducing the complexity of the similarity computations applied. Tiakas et al. [151] was among the first to propose this solution for road networks. The approach uses the sum of the distances between nodes in the road network that contribute to the final distance, but the trajectory pairs must have the same length to guarantee point-to-point matching. To circumvent the length constraint, Mao et al. [120] proposed a related solution that used DTW after converting pointwise trajectories to paths on a road network. In both approaches, the similarity is still computed based on the end-points of road segments, so map matching is only applied once to clean trajectory data and to align sampling rates, but it still needs to compute the Euclidean distance between nodes that can be computationally expensive even is small datasets.

To reduce the costs in when computing the distances, Wang et al. [166] proposed the use of *longest overlapped road segments* (LORS). This measure was inspired by LCSS and adapted to leverage the properties inherent in map-matched data. LORS does not compute the Euclidean distance between points to determine if they adhere to a threshold distance constraint. Instead, overlapping segments between trajectories are identified first and then reused to compute the similarities,

$$d_{LORS}(Q,T) = \begin{cases} 0, & \text{if } Q \text{ or } T \text{ is empty} \\ |e_{1m}| + d_{LORS}(Q_h,T_h), & \text{if } e_{1m} = e_{2x} \\ \max(d_{LORS}(Q_h,T), d_{LORS}(Q,T_h)), & \text{otherwise} \end{cases}. \tag{6}$$

Here, the inputs are $Q = (e_{11}, e_{12}, \ldots, e_{1m})$ and $T = (e_{21}, e_{22}, \ldots, e_{2x})$, where e_{1m} are the end edge of trajectory Q, and $|e_{1m}|$ is the travel length of a graph edge e_{1m}. Yuan and Li [190] further

[13] Actually all aforementioned spatial-only measures can be easily extended to handle the spatio-temporal case, by performing two separate distance calculations on the spatial and temporal features and then combining the score by using a balancing factor α.

extended and normalized LORS to account for trajectory length effects in $|Q|$ and $|T|$. The resulting measure LCRS can be defined as: $d_{LCRS}(Q,T) = \frac{d_{LORS}(Q,T)}{|Q|+|T|-d_{LORS}(Q,T)}$. By modeling network-constrained trajectories as strings in a similar manner, Koide et al. [89] proposed a generalized metric called *weighted edit distance* (WED), which supports user-defined cost functions that can be used with several important similarity functions such as ERP and EDR.

Both LORS, LCRS, and WED satisfy the local time shift constraint and have a quadratic complexity using dynamic programming. Wang et al. [164] later simplified the computational costs of LORS further. The key insight was to exploit ordered integer intersection (through finger search), where ordered integers represented unique IDs assigned to road segments during map-matching. The resulting method called *edge-based distance* (EBD): $d_{EBD}(Q,T) = \max(|Q|,|T|) - |Q \cap T|$, was shown to have comparable precision to LORS and was also highly scalable in practice.

3.3 Complexity and Robustness

Table 4 compares characteristics of similarity measures based on four different properties: complexity, metricity, parameter independence, and robustness. Since parameters such as τ in LCSS and EDR can be easily observed from their definitions, we will elaborate the other three characteristics next.

Complexity. Many trajectory-based similarity measures were designed to allow search to be more resilient to variance produced by local time shifting. However, computing optimal distances with ordering constraints generally requires dynamic programming solutions with quadratic complexity ($O(n^2)$) as well as the associated space overheads.

Metricity. Fully metric compliance is often crucial for a similarity measure, as obeying the triangle inequality provides both theoretical and practical advantages. Here, we use an example to illustrate the concept. Consider the computation of the well-known metric similarity measure, Euclidean distance. Given a query point q with p_1 and p_2 as the two next candidates for a ***k*NN** query, $d(q,p_1)$ is computed. Before distance is computed for p_2, the lower and upper bound of $d(q,p_2)$ can be estimated based on $d(q,p_1)$ and $d(p_1,p_2)$ as $|d(q,p_1) - d(p_1,p_2)| < d(q,p_2) < d(q,p_1) + d(p_1,p_2)$, where $d(p_1,p_2)$ can be pre-computed off-line for any query. The lower bound can be compared with $d(q,p')$, where p' is currently the best nearest neighbor candidate. If the bound is greater than the current best result, then we can discard p_2 directly as it can never replace p'.

Robustness. In addition to complexity and metricity, many similarity measures are sensitive to noise, sampling rate, and point shifting (points are shifted on the path where the trajectory lies). The existence of noise due to GPS errors or other common sensor data collection approaches result in more (or fewer) sampled points, point shifting, and other quality issue. Wang et al. [156] systematically compared the robustness of six common similarity measures in a real-world trajectory dataset, and a complete version [146] was published later with additional point-based measures. However, the most recent segment-based measures [166, 190] were not covered.

Figure 4 illustrates several common robustness problems encountered in trajectory data. T_1 and T_2 are two vehicle trajectories in a road network that overlap. After sampling, a set of points represents each one, with perfect data resulting in Figure 4(b). In reality, sampling rates can vary, resulting in a point shift (Figure 4(c)). GPS errors (Figure 4(d)) can also occur in sampled points that are not on the same road, and inconsistent sampling rates (Figure 4(e)) lead to a different number of points being produced for the two trajectories. Each of the above issues can lead to distance scores greater than those found in error free data, and can even result in incorrect solutions for certain query types. Robust similarity measures are therefore highly desirable.

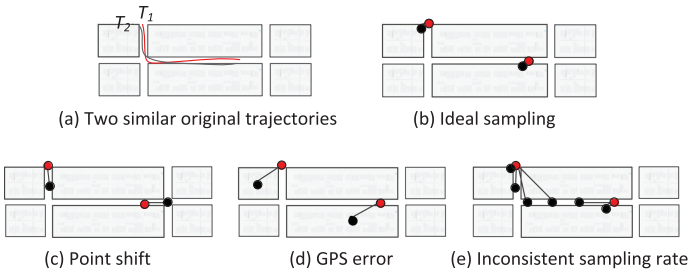


Fig. 4. A similarity measure that is sensitive to point shift, error, and sampling rate.

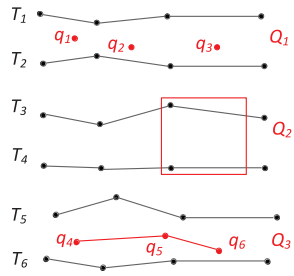


Fig. 5. Three kinds of spatial-only trajectory queries.

Table 5. An Overview of Common Operations Applied to Trajectory and Point Data

Data	Type	Query	Input	Output	Measure	Index
Spatial-only	Basic	Range [91, 136]	Range	Trajectories	N/A	R-tree
		Path [88, 91]	Path	Trajectories	N/A	R-tree
	*k*NN	***k*NN** [20, 74]	Point	Points	Euclidean	R-tree
		*k*NNT [34, 141, 154]	Trajectory	Trajectories	LCSS, DTW	R-tree
		***k*BCT** [38, 149]	Points	Trajectories	Aggregate	R-tree
	Reverse *k*NN	**R*k*NN** [150]	Point	Points	Euclidean	R-tree
		R*k*NNT [163]	Trajectory	Trajectories	CPD	N/A
	Clustering	Density [95]	Thresholds	Paths	N/A	N/A
		Partition [164]	k	Trajectory	EBD	Inverted index
	Join	Similarity join [148, 190]	Threshold	Pairs	Normalized CPD	Signature
		***k*NN** join [57]	k	Pairs	CPD	Grid-index
Spatial-textual	Top-k	**KS** [25, 52]	Keywords	Points	TF-IDF	Inverted list
		T*k*SK [105, 195]	Point, keywords	Points	Aggregate	IR-tree
		T*k*STT [140, 202]	Points, keywords	Trajectories	Aggregate	Grid-index, Inverted list

4 TRAJECTORY SEARCH AND JOIN

4.1 Spatial-only Trajectory Search

Figure 5 shows three exemplar queries Q_1, Q_2, Q_3 for the spatial-only trajectories T_1 to T_6. Given three points (q_1, q_2, q_3), Q_1 finds the best connected trajectory. Among all six trajectories, T_1 and T_2 are both possible solutions. For a pointwise similarity search as shown in Q_1, the similarity measure used should ideally be able to reliably distinguish between the two candidates. Given a target range (red box), Q_2 finds all trajectories covered (partially or fully). Valid results are T_3 and T_4, and cannot be computed using any of the similarity measures shown. In comparison, if Q_3 is the query for the top-k search, then a similarity measure must to be defined that can capture all of the properties of Q_3, including points and ordering.

Table 5 compares and contrasts several common trajectory queries by input, output, similarity measure, and most appropriate index. The "Index" column shows the preferred indexing approach. Many other alternatives exist. A more comprehensive analysis and discussion of indexing solutions is in Section 4.2. Query types range from basic trajectory search, top-k trajectory similarity search

(spatial-only and spatial-textual), to more complex operations such as reverse k nearest neighbors search, and trajectory clustering. We now discuss each of these query operations in detail along with several indexing techniques that have been proposed to accelerate performance.

4.1.1 Basic Trajectory Search. Basic trajectory search includes three classic query formulations. The most basic is a Range Query (RQ) [91, 134, 136, 144], which finds all (sub-) trajectories located in a spatial or temporal region. Range queries has many applications in traffic monitoring such as returning all vehicles at a road intersection. The second is a *Path Query* (PQ), which retrieves the trajectories that contain any edge of the given path query. The third is a *Strict Path Query* (SPQ) [88, 91, 136], which finds all trajectories that traverse an entire path from beginning to end. Interestingly, path queries and strict path queries share many common properties with disjunctive (OR) and conjunctive (AND) Boolean queries [119].

Definition 5 (Range Query). Given a trajectory database $D = \{T_1, \ldots, T_{|D|}\}$ and query rectangular region Q_r, a range query retrieves the trajectories: $\text{RQ}(Q_r) = \{T \in D | \exists p_i \in T(p_i \in Q_r)\}$.

Definition 6 (Path Query). Given a Q_p that is a path in G, a path query retrieves the trajectories T that pass through at least one edge e_j of Q_p: $\text{PQ}(Q_p) = \{T \in D | \exists e_i \in T, e_j \in Q_p(e_i = e_j)\}$.

Definition 7 (Strict Path Query). Given a Q_p, a strict path query retrieves the trajectories whose edges can all be found in Q_p: $\text{SPQ}(Q_p) = \{T \in D | \exists i, j(T_{ij} = Q_p)\}$, where $T_{ij} = \{e_i, e_{i+1}, \ldots, e_j\}$ is the sub-trajectory of T.

4.1.2 k Nearest Neighbors Query. To find a relevant subset from a large set of objects for a given query object, k nearest neighbor queries have been widely applied in spatial databases. For trajectories, the query can be either a trajectory or a set of points.

Definition 8 (k Nearest Neighbors Query over Trajectories). Given a trajectory database $D = \{T_1, \ldots, T_{|D|}\}$ and query $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$, a k Nearest Neighbors Query ($k\mathbf{NN}(Q)$) retrieves a set $D_s \subseteq D$ with k trajectories such that: $\forall T \in D_s, \forall T' \in D - D_s, d(Q, T) < d(Q, T')$.

Search by Trajectory. Given a query trajectory Q, a k Nearest Neighbor Trajectories Query aims to find the k most similar/nearest trajectories to Q, based on a given trajectory similarity measure [166, 182]. Such a query can be used to find the most similar trips in traffic flow analysis. For the special case when the trajectory is a single point, which is also known as ***k*NN** search (we use *kNNT* to denote trajectory search), with the default similarity being Euclidean distance.

Search by Points. There has also been previous work targeting search on spatial-only trajectory data where the input query is a set of points [38, 131, 149]. Chen et al. [38] initially proposed the problem of querying over a set of points with spatial-only trajectories. They proposed incremental expansion using an R-tree to prune candidate points from consideration. They referred to the approach *Incremental k Nearest Neighbors* (IKNN). To optimize the IKNN algorithm, Tang et al. [149] devised a qualifier expectation measure that ranks partially matched candidate trajectories. The approach accelerates query processing significantly when non-uniform trajectory distributions and/or outlier query locations exist in the solution space.

4.1.3 Reverse k Nearest Neighbor Query. Instead of searching the most relevant objects to a query object, *Reverse k Nearest Neighbors* (RkNN) queries attempt to locate objects that will take the query as one of their k nearest neighbors. Formally, **R*k*NN** is defined as below:

Definition 9 (Reverse k Nearest Neighbor). Given a set of points (or trajectories) D and a query point (or trajectory) Q, a $\mathbf{R}k\mathbf{NN}(Q)$ retrieves all objects $T \in D$ that take Q as kNN, i.e., $\forall T$, $Q \in k\mathbf{NN}(T)$.

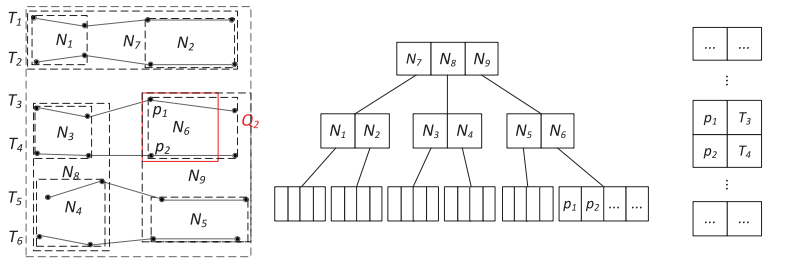


Fig. 6. Indexing spatial-only trajectories using an R-tree and a mapping table.

R*k*NN queries for spatial point data have attracted considerable attention in the research community [150] as it can be used to solve a wide variety of industry-relevant problems. An **R*k*NN** query aims to identify all (spatial) objects that have a query location as a k nearest neighbor. Important applications of the **R*k*NN** query include resource allocation [163] and profile-based marketing [199]. For example, **R*k*NN** queries can be used to estimate the number of customers a new restaurant would attract based on location, and was initially referred to as a *bichromatic* **R*k*NN** based on the dual-indexing solution proposed to solve the problem [150].

Instead of querying a set of static points, Cheema et al. [29] proposed the *continuous reverse nearest neighbors* query to monitor a moving object. The goal is to find all static points that contain the moving object as a k nearest neighbor. This approach targets a single point rather than a commuter trajectory of multiple-points, which was later solved in **R*k*NNT** [163].

Reverse Trajectory Search. The Reverse k Nearest Neighbor Query over Trajectories (**R*k*NNT** [163]), and can be defined as follows. Given a trajectory dataset $\mathcal{D}_T$ and a set of routes $\mathcal{D}_R$ (also defined as trajectory) and a candidate point set $Q = (o_1, o_2, \ldots, o_m)$ as a query, **R*k*NNT** returns all the trajectories in $\mathcal{D}_T$ that will take Q as k nearest routes using a point-to-trajectory similarity measure. The main application of **R*k*NNT** is to estimate the capacity of a new bus route and can be further used to plan a route with a maximum capacity between a source and destination, which was defined as a **MaxR*k*NNT** in Reference [163]. The main challenge in solving **R*k*NNT** is how to prune the trajectories that cannot be in the results without explicitly accessing the whole dataset $\mathcal{D}_T$. All of the pruning methods commonly used for **R*k*NN** may work for trajectories. For example *half-space* pruning [150], which can prune an area by drawing a perpendicular bisector between a query point and a data point will not work. Wang et al. [163] proposed the use of an R-tree of the trajectory routes (an example can be seen in Figure 6). By drawing bisectors between the route points and the query, an area can be found where all trajectory points contained in a region can not have the nearest route for the query.

4.2 Spatial Indexing Algorithms

Trajectory indexing is commonly applied to improve efficiency in trajectory related search problems. Trajectory data are processed off-line to improve scalability and prune the search space. Existing approaches to trajectory indexing have two components: indexable point data and mapping tables.

Point Indexing. Space-efficient index representations and processing frameworks are crucial for trajectory data, and the majority of trajectory search solutions [38] rely on an R-tree [74], which store all points from the raw trajectories, and are historically the dominant approach deployed for spatial computing applications. As shown in Figure 6, trajectories are decomposed to points first,

then an R-tree is used to index each point. A mapping table identifies the trajectory that contains each point. Since trajectory datasets such as T-drive [192] can easily contain millions of points, the R-tree must manage an enormous number of maximum bounding rectangles (MBR), which have prohibitive memory costs in practice. So, traditional methods employed for spatial pruning can be infeasible on raw trajectory data. Simpler Grid-index solutions can sometimes be more appropriate in such scenarios [168, 202]. Nevertheless, the core problem is compounded by the fact that many of the similarity measures proposed for trajectory search are non-metric.

Mapping Table. A mapping table is used to map points to trajectories [38, 202]. After searching for a point, the mapping table identifies the trajectory containing the point. Ensure that the mapping is unambiguous, every point has a unique identifier ranging from $[1, |\mathcal{D}.P|]$, each trajectory has a unique identifier ranging from $[1, |\mathcal{D}|]$, $|\mathcal{D}.P|$ and $|\mathcal{D}|$ are the number of points and trajectories. With the mapping table in Figure 6(c), we can know that p_1 is a point of T_3 and p_2 belongs to T_4.

Pruning Mechanism. For range queries, most existing approaches employ MBR-based pruning, as MBR intersection with a query trajectory can be used to effectively and efficiently prune the search space. If a node intersects with a query range, then the object covered can be removed from consideration. For example, the nodes N_7 and N_8 in Figure 6(a) do not intersect with N_2, so they can be eliminated from consideration, leaving only N_9. For ***k*NN** trajectory search, there are two common pruning techniques: *early termination* and *early abandoning*. Early termination is initially proposed in the threshold algorithm [56]. Given an index, the algorithm scans a sorted list, and early terminates when all remaining items exceed some pre-computed upper bound. It is similar to pruning in ***k*NN** search with an R-tree where the minimum distance between the query and a node in the index can be used to determine if the distance for the items contained must be computed. In contrast to early termination that can prune many items in a single operation, early abandoning must consider each item but exploits a bound to minimize the number of expensive computations applied during processing. This technique is commonly used to accelerate time-series similarity search and ***k*NN** search [154]. By estimating the upper bound of the similarity score and comparing it to the kth item in a max-heap (for top-k search for example), an algorithm can determine if a full similarity computation must be made for the item under consideration.

4.3 Trajectory Join

Definition 10 (Trajectory Join). Given two sets of trajectories S_1 and S_2, and a similarity threshold ϵ (or k in ***k*NN** search), a trajectory join operation will return all pairs of trajectories T_i and T_j from S_1 and S_2 with a similarity that exceeds ϵ (or $T_i \in k\mathbf{NN}(T_j)$).

The main applications of trajectory joins are in trajectory data cleaning, near-duplicate detection, and carpooling. For example, given a driver trajectory database and a rider trajectory database, a trajectory join will match riders with similar trajectories to drivers. Trajectory join operations can be divided into two categories: trajectory similarity joins [148] and trajectory k Nearest Neighbors joins [57]. A simple baseline is to conduct a similarity search or ***k*NN** search for each trajectory. Given the quadratic complexity of this solution, scaling can be problematic.

To reduce the complexity, many different indexing techniques can be used. Bakalov and Tsotras [17] considered the moving object trajectory similarity join, which is a pointwise join in a specific timestamp, and similarity is computed between points and not trajectories. Ta et al. [148] proposed a signature-based solution to filter trajectory pairs before distance computation, where the signature is built for each trajectory to help prune, based on its crossed and neighbor grids. Shang et al. [139] explored the spatial-temporal trajectory similarity join problem for road networks. A

two-step algorithm is applied to each trajectory: expansion and verification, similar to trajectory similarity search in Section 4.

Distributed Trajectory Similarity Join. Distributed computing is often applied to solve the trajectory join problem due to the high computational costs of the distance computation. Fang et al. [57] proposed a distributed trajectory ***k*NN** join method using MapReduce. Their solution used Hadoop, and the algorithm can also run on a single machine. Shang et al. [141] computed distributed trajectory similarity joins using Spark, where a global index is deployed over multiple computer nodes and used to prune the search space. Yuan and Li [190] also explored the distributed trajectory similarity join problem using similar techniques to Shang et al. [141] but applied the similarity measure LCRS described in Section 3.2.3 to compute.

4.4 Spatial-Textual Trajectory Search

Spatial-textual trajectory search incorporates text and keywords to the data, and relevant trajectories have more than one dimension of similarity that must be considered. For example, the keywords "*seafood*," "*coffee*," and "*swimming*" might be contained in a query and a trajectory.

4.4.1 Top-k Spatial-Textual Trajectory Search. **Similarity Measures.** In contrast to spatial-only similarity measures, spatial-textual measures for an activity trajectory depend on both spatial and textual components. Linear combinations of two distinct similarity measures is a common solution for spatial-textual data [41, 195]. The spatial distance is combined with TF·IDF similarity for the text, with a user-defined weight between the two to balance importance based on the target application. Such a measure was also extended by Shang et al. [140], where the spatial-distance is computed in a manner similar to the ***k*BCT**. The textual similarity was then computed using simple keyword intersection. Since the text relevance is computed as a part in the final score, it is more flexible in real applications. Conversely, a conjunctive approach [42, 202] tightens the constraint, requiring that the result trajectory should contain all of the query keywords, and results are ordered by spatial distance.

Pointwise Search with Keywords. Several recent papers have explored the problem of spatial-textual trajectory search in a range of different scenarios. Cong et al. [42] proposed a sub-trajectory-based solution to find the minimum travel distance for a single query point using a Boolean keyword match. Shang et al. [140] presented a disjunctive solution for multiple points, where the distance is measured between points, but keywords are assigned to an entire trajectory instead of each point.

Zheng et al. [202] attached the keywords to a specific point in the trajectory to allow finer-grained matching of textual information. However, their work only supported conjunctive text matching, so results must contain all query keywords, simplifying the more general keyword search problem. To handle the case when users do not have a preferred locations or exact text matching should not be guaranteed, Zheng et al. [201] proposed an approximate query solution over trajectories that use a set of keywords, and the similarity is measured as the travel distance, just as Cong et al. [42] did. A special case of this problem is when the query contains only keywords, and is known as a keyword-only search (KS) [25]. Another special case is when the trajectory data and query are a single point and is known as Top-*k* Spatial Keyword Search (T*k*SK) [195].

4.4.2 Boolean Range and Top-k Query. Hariharan et al. [78] utilized the Boolean range queries to find all objects containing query keywords, all of which must be located within a bounded range. Boolean Top-*k* queries [50] find the *k* nearest objects using conjunctive Boolean constraints on the text, i.e., it conducts a ***k*NN** search over all the points that contain all the keywords from query. Han et al. [77] investigated Boolean range queries on trajectories and considered spatial, temporal, and

textual information in the solution, and an index based on an octree and inverted index is proposed to answer their proposed query.

4.4.3 Pruning with an Inverted Index. Inverted indexes [210] are widely used to efficiently manage text information for spatial-textual problems. Related algorithms and data structures for inverted indexes have benefited from many years of practical improvements to support web search and related applications, making them an obvious candidate to manage textual components in heterogeneous data collections. A unique object identifier is mapped to every keyword contained, and an inverted list of all objects containing each keyword can only represent existence, or contain auxiliary data such as the frequency of the keyword in the object text. The auxiliary data can also be a pre-computed similarity score or even offsets for phrase reconstruction. Most prior work [195] attaches inverted lists to nodes in an existing tree-based spatial index (a Grid-index for example) to support spatial-textual similarity computations during tree traversal.

5 TRAJECTORY CLUSTERING AND CLASSIFICATION

5.1 Clustering Large-scale Trajectories

Clustering similar trajectories to produce representative exemplars can be a powerful visualization tool to track the mobility of vehicles and humans. It has been investigated in many different applications, such as spatial databases [12, 82, 95], data mining [129], transportation [181], and visualization [60]. Clustering is most often applied to spatial-only trajectories, with prior work on spatial-textual trajectory clustering being relatively rare. Trajectory clustering can be broadly divided into two categories: Partition-based [27, 60, 68, 82, 129, 186] and Density-based [12, 15, 75, 95, 104]. Both the partition and the density-based trajectory clustering require extensive similarity computations, with the only distinction being if it is computed for whole trajectories or using only sub-trajectories.

5.1.1 Partition-based Clustering. Given a set of trajectories, partition-based clustering divides trajectories into a limited number (k) of groups (clusters). The similarity measure (Section 3) and parameter k are selected *a priori* depending on the use case.

Definition 11 (Partition-based Trajectory Clustering). Given a set of trajectories $\{T_1, T_2, \ldots, T_n\}$, partition-based clustering aims to partition n trajectories into k ($k \leq n$) clusters $S = \{S_1, S_2, \ldots, S_k\}$ by minimizing the objective function: $O = arg\,min_S \sum_{j=1}^{k} \sum_{T_i \in S_j} d(T_i, \mu_j)$.

Here, each cluster S_j has a centroid trajectory or path μ_j, and $d(T_i, \mu_j)$ is a similarity measure. Many different similarity measures such as the ones introduced in Section 3 have been used for trajectory clustering. For partition-based clustering, the most appropriate similarity measure used in the application scenario can vary widely (e.g., vehicle [27, 60, 82, 129], soccer player [68], cellular [181], and large vessels [186]). Parameterization can also be an important hurdle as well as several approaches need to optimize multiple parameters in addition to k in order to produce high quality results. As an extension of k-**means**, which is NP-hard when computing an exact solution, is partition-based trajectory clustering, which can also be computationally intractable for large collections when certain similarity measures are required. So, often the only solution is to find ways to reduce the number of trajectories compared through similarity thresholding [129] or through approximation ratios if an exact algorithm, which cannot be achieved in polynomial time [27].

One possible approach is to use k-**paths** clustering [164], which is an extension of k-**means** for trajectories on a road network. The key idea in k-**paths** is to use a quasilinear similarity measure-EBD and prune the search space based with an inverted index and exploit the metric properties of EBD. As we described in Section 3, EBD computes similarity by performing a set intersection over edges, which substantially improves the performance of pairwise similarity computation, and

also maintains comparable precision to common measures that have a quadratic computational complexity. Then, an indexing technique based on an edge inverted index and a tree structure for metric similarity measure is used to further reduce the number of similarity computations in the assignment and refinement phases.

5.1.2 Density-based Clustering. Density-based trajectory clustering first finds dense "segments" and then connects these segments to produce representative routes. However, the most appropriate approach to identifying dense segments highly dependent on properties of the data. TRACLUS [95] is the most cited trajectory clustering algorithm that solves this problem in two steps, each trajectory is partitioned into a set of line segments (sub-trajectory), and then similar line segments are grouped together into clusters. A distance threshold τ is the most common parameter used in density-based clustering solutions, which specifies the neighborhood radius.

Based on TRACLUS [95], Li et al. [104] proposed an incremental clustering framework to allow new trajectories to be added to a database, and a new parameter is further introduced for a new step-*generating micro-clusters before the final clusters*. Similarly, Agarwal et al. [12] reduced the clustering problem to finding a frequent path in a graph, which allowed standard graph traversal techniques to be applied to identify *pathlets* that could be used to represent each cluster with common sub-trajectories. Each cluster's pathlet is defined as a sequence of points that is not necessarily a subsequence of an input trajectory. The main difference with TRACLUS is that the pathlets are the result of optimizing a single objective function to best represent the trajectories, while TRACLUS generates a set of common segments through density-based clustering, then connects them to form several common paths to represent the trajectories. Unfortunately, finding pathlets has been proved to be an NP-hard problem, so exact solutions are intractable.

5.2 Trajectory Classification

For urban data, there are mainly two types of trajectory classification problems: (1) similarity-based classification [94] and (2) transportation modes and activities classification [45, 205], which was reviewed recently in Reference [204]. Next, we will mainly focus on the similarity-based classification, and also introduce a new classification task for travel-based inferences [30, 67, 123, 161].

5.2.1 General Similarity-based Classification. Several studies have investigated trajectory classification problems, which also require extensive similarity computations [94, 128, 142]. The main distinction in trajectory clustering is that classification assigns a label to an individual trajectory based on its features, while clustering is conducted over all items in a dataset. Specifically, Lee et al. [94] proposed a feature generation framework for trajectory classification, where two types of clustering, region based and trajectory based [95], are used to generate features for traditional classifiers such as decision tree or an SVM. Sharma et al. [142] proposed a nearest neighbor classification for trajectory data by computing distance with trajectory sampled and then choosing the nearest trajectory as the label. Andres et al. [14] found relevant sub-trajectories as features for robust classification, where the distance is computed between two equal-length sub-trajectories.

5.2.2 Trip Purpose Classification. Inferring the purpose of a trip has potential applications for improving urban planning and governance [161], but it is normally conducted by through manual surveys. With the proliferation of trajectory data availability, trip purposes can be also classified. Gong et al. [67] categorized the spatio-temporal characteristics of nine daily activity types based on inference results, including their temporal regularities, spatial dynamics, and distributions of trip lengths and directions. Wang et al. [161] proposed a probabilistic framework for inducing trip ordering in massive taxi GPS trajectories collections. The key idea is to augment the origin and destination with neighbor points of interest (POIs) and identify POI links based on time periods. Then

Table 6. A Summary of Trajectory Applications, Where "P2P" Denotes the Point-to-point Distance, and "P2T" Denotes the Point-to-trajectory Distance

Type	Work	Map matching	Storage	Measure	Representative query	Clustering	Indexing
Road Traffic	Monitoring [166, 170, 182]	✓	Segment	—	Range query	✓	✓
	Jam & Flow [48, 85, 203]	✓	Segment	EBD	✗	✓	✓
	Anomaly [32, 172]	✗	Point	EDR	✗	✓	✓
Green Transport	Network design [31, 111, 130]	✗	Point	P2P	Constrained path search	✓	✓
	Navigation [47, 70]	✓	Segment	✗	Shortest path query	✗	✓
	Carpooling [79, 81]	✗	Segment	P2P	Join	✓	✓
Tourism Planning	Trip search [165, 167]	✗	Point	P2T	TkSTT	✓	✓
	Customized trip [37, 169]	✗	Point	P2P	Frequent path	✓	✓
	Interest Discovery [126, 169]	✗	Point	P2P P2T	Multiple queries	✓	✓
Site Selection	Billboard [199, 200]	✗	Point	P2T	Set cover	✗	✓
	Charging station [102, 106]	✗	Point	✗	✗	✗	✓
	Facility route [18, 96, 163]	✓	Segment	P2T	Shortest path query	✓	✓

the trip intents can be explained semantically. Nair et al. [123] learned a model to automatically infer the purpose of a cycling trip using personal data from cyclists, GPS trajectories, and a variety of built-in and social environment features extracted from open datasets characterizing the streets.

6 TRAJECTORY-BOOSTED APPLICATIONS IN URBAN PLANNING

Large-scale trajectory data are primarily collected in urban areas, making it a valuable tool for applications in real-time smart cities and timely decision making. We will focus on four categories of applications that can benefit from trajectory data. Table 6 provides a summary of representative work in each of these broad areas. To intuitively illustrate the connections between each of these, we also provide several examples in Figure 12. Other examples can be found in the Appendix of Reference [162].

6.1 Applications in Road Traffic

Cities and government agencies are often a valuable source of trajectory data. There has been an increased emphasis on collecting data from traffic signals and other related sensors for internal auditing purposes, and regulatory requirements often require the data collected to be made publicly available, since it was gathered using tax income. Such data can be used for traffic monitoring [170], anomaly detection [172], and traffic jam and flow analysis [175].

An Overview. Figure 7 shows the pipeline of three representative trajectory analytical tasks for road traffic. As a common operator, mapping trajectories onto road networks not only cleans the data but also enables traffic jam and flow analysis in real time.

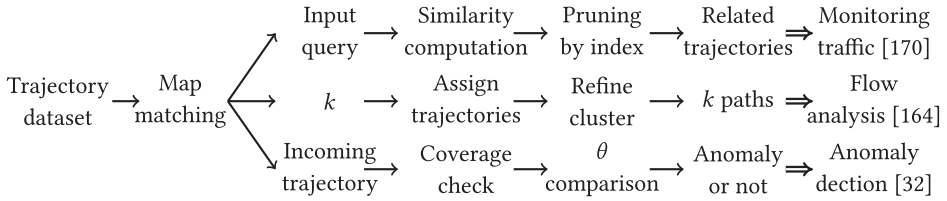


Fig. 7. Pipeline decomposition of three types of work on road traffic.

6.1.1 Traffic Monitoring. Since trajectories record the trace of vehicles in the road, many visualization systems [35] were developed to observe and monitor past or real-time traffic trends and movement patterns based on trajectory data. Users can interactively explore the traffic condition in a specific area or road, and further control the traffic if necessary. As visualizing an entire city-wide trajectory dataset on a single screen can be unreadable, selecting an area and identifying the trajectories covered is a useful application of a range query, which was described in Section 4. In addition to a range query, a wide variety of *trajectory search* queries [34, 49, 88, 91, 134, 136] have been proposed over the years to support various applications. To monitor all vehicles that use *a Main Street*, a *path query* [91] can be issued. Further, a *strict path query* identifies every vehicle that traverses all of *a Main Street*. Wang et al. [170] integrated all the above queries into a real-time traffic monitoring system, where user can interactively conduct queries and get results in real time.

6.1.2 Traffic Jam and Flow Analytics. Live traffic monitoring enables analysts to track usage of specific roads or areas in real time, and can be used to identify traffic jams to notify commuters through digital traffic signs or GPS applications. After mapping trajectory data to a road network, Wang et al. [175] visually inspected traffic data and verified problem areas based on speed limit information of the streets under consideration.

Instead of inspecting individual road segments to identify traffic jams, traffic flow analysis does not rely on domain-specific features such as speed limits. The goal is to discover important movement patterns of drivers such as repeatedly using a common route composed of several different road segments. Several recent papers have proposed concept definitions that capture essential properties of traffic flow analysis based on trajectories. Gudmundsson and Van Kreveld [69] defined the concept of a *flock*, which is a common sub-trajectory covered by a set of trajectories with at least k circles with the same-radius. Jeung et al. [85] extended the concept of a *convoy*, where the circles can have various radii. Li et al. [103] proposed the concept of a *swarm* to solve the problem that a sub-trajectory has to be a continuous set of points in the corresponding trajectory. Rather than a single sub-trajectory, Zheng et al. [203] proposed the concept of *gathering*, which returns multiple sub-trajectories. To further avoid specifying multiple hyper-parameters, e.g., the number of circles, k-**paths** trajectory clustering will return k most representative trajectories (real paths on road network) [164], which can be used in the traffic flow analysis.

6.1.3 Anomaly Trajectory Detection. Anomaly or outlier trajectories in a dataset can be defined as a trajectory that falls outside of a predefined confidence interval w.r.t. the entire collection distribution. It has be applied in applications such as identifying criminal behavior in a population [143] and taxi driver fraud detection [172]. A combination of similarity and clustering can be used to solve this problem. Computing similarity between trajectories can be used to identify trajectories that are the most dissimilar in a dataset, and clustering can be used to aggregate common trends that might not be easily identifiable otherwise. For example, given an incoming trajectory T between a source s and a destination d, anomaly detection was defined by Chen et al. [32] as follows: (1) the anomaly score of T based on the proportion of existing trajectories in D passing through s

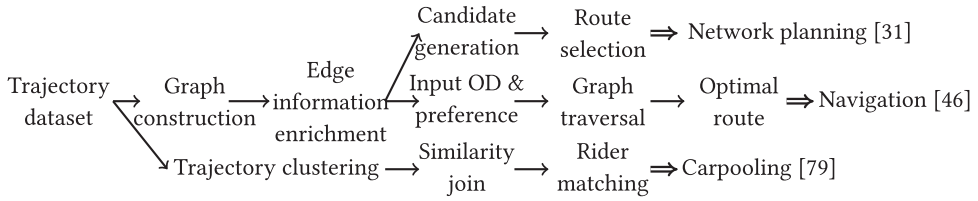


Fig. 8. Pipeline decomposition of work on green transport.

and d and covering T completely and (2) a predefined threshold θ to determine if T is anomalous or not. The similarity measures are based on two types of information: (1) sub-trajectories [93] and (2) entire trajectories [32, 172]. Lee et al. [93] detected outliers using line segments, i.e., a trajectory was partitioned into a set of line segments, and then outlying line segments were used to find trajectory outliers. Wang et al. [172] utilized an edit-based similarity measure to detect anomalies in a hierarchical cluster arrangement.

6.2 Applications in Green Transport

Public transportation networks such as subways, bus routes, and gas stations provide more choices for green commuting. Trajectory data are commonly applied in green transport (also known as "sustainable transport" [137]) applications to optimize network design, conduct personalized and adaptive navigation, and carpooling. Note that this survey focuses on algorithms and complexity, and we also refer the interested readers to a review that covers this problem in detail [121] for a more comprehensive view from the perspective of road transportation agencies.

An Overview. Based on these examples, we observe that a critical task in road networks is trip planning or finding common routes in a graph for a passenger or a driver, as shown in Figure 8. Complex route planning problems typically maximize a capacity or minimize a travel time. When viewed this way, many of the problems encountered are NP-hard, reducible to the traveling salesmen problem, and solved using a greedy algorithm.

6.2.1 Transit Network Planning. By mining taxi data, Chen et al. [31] approximated night-time bus route planning by first clustering all points in taxi trajectories to determine "hot spots" that could be bus stops, and then created bus routes based on the connectivity between two stops. Toole et al. [152] used census records and mobile phone location information to estimate demand in transit routes. Historical traffic data [11], smart cards [198], sensors [117], and cellular data [130] can provide more comprehensive demand data and be used to further improve data-driven transit network design. Pinelli et al. [130] derived frequent patterns of movement from trajectories by computing distance and the flow gain and then merging them to generate a network for the whole city. A common shortcoming of these methods is that they are building a new network and are not applicable for most cities that already have bus networks. Based on human mobility patterns extracted from taxi data and smart card data, Liu et al. [111] proposed a localized transportation choice model, which predicted travel demand for different bus routes by taking into account both existing bus network and taxi routes.

6.2.2 Adaptive Navigation. Travel distance-based shortest path search between a source and destination is widely used in navigation services. It can be applied adaptively using historical trajectory data from drivers. With precise map matching techniques, discovering trajectory paths in data can enrich models for a road network, e.g., dynamic time cost of each road [47], driver preferences [70]. Using a dynamic travel time cost for each road segment, travel time estimation [47] and fastest path search [171] is more realistic, capturing various traffic conditions, and

potentially leading to alternative shortest paths when all aspects are considered. Guo et al. [70] proposed learning to route with sparse trajectory sets, by constructing a region graph of transfer routing preferences. In the graph, the nodes are equal-size grids, and the edges represent two grids crossed by more than a fixed number of trajectories. Discovering preferences from trajectories can also be used to support personalized route recommendation for green vehicle routing [21]. Dai et al. [46] developed techniques to support efficient trajectory subset selection for following using driver preferences (travel distance, travel time, and consumed fuel) and the source, destination, and departure time specified by the driver. Delling et al. [51] proposed a framework to generate personalized driving directions from trajectories, where the path preferences were composed of several features, such as avoiding tolls, U-turns, and highway versus surface streets.

6.2.3 Carpooling. There are two common approaches to ridesharing problems that rely heavily on trajectory data. The first one is grouping passengers and drivers based on their historical trajectory data. The underlying goal is to keep all seats in all vehicles occupied and is essentially a type of bin packing problem. For example, Hsieh [81] devised a carpooling system to match passengers and drivers based on their travel trajectories, and the similarity measure was defined as the sum of the distance from the pickup and drop off points to the nearest point to a driver trajectory. The other approach is to detect frequent routes using only rider trajectories. He et al. [79] proposed a carpooling system that generates an efficient route for dynamic ridesharing by mining the frequent routes taken by participating riders. Hong et al. [80] performed rider matching by clustering trajectories and generated a simulation model of ridesharing behaviors to illustrate the potential impact. Similarly to other road traffic applications, similarity computations are performed between passenger and driver trajectories, clustering is then used to detect the frequent routes shared by multiple passengers in the trajectory-driven carpooling.

6.3 Applications in Tourism Planning

The tourism market is an enormously valuable commodity in many countries. For example, this market in Britain is expected to be worth more than £257 billion by 2025 and represents 11% of the total GDP of the UK.[14] From a consumer perspective, historical trip data can also improve overall satisfaction and provide tools to discover more personalized opportunities that align with individual preferences.

6.3.1 Customizing Tours. Searching for similar trips related to a set of given points with keywords [165, 201, 202] or a range [77] can enable more effective zero knowledge querying capabilities, as we described in Section 4.4. In addition to spatial proximity and text relevance, other factors can be considered, such as photos [115]. Lu et al. [115] leveraged existing travel clues recovered from 20 million geo-tagged photos to suggest customized travel route plans according to user preferences. Wei et al. [176] developed a framework to retrieve the top-k trajectories that pass through popular regions. Rather deriving search results using only existing trips, generating new trip plans without keywords or locations is also possible. Chen et al. [37] discovered popular routes from spatial-only trajectories by constructing a transfer network from trajectory data in two steps. First, hot areas are detected as nodes in the network, and then edges are added based on connectivity properties in the entire trajectory collection. Then, a network flow algorithm was proposed to discover the most popular route from the transfer network. Wang et al. [169] developed an interactive trip planning system called TISP, which enabled interesting attractions to be discovered, and used to dynamically modify recommendations using click-based feedback from POIs displayed on the map.

[14] https://www.visitbritain.org/visitor-economy-facts.

6.3.2 User Interest Discovery. When a trajectory is enriched with text, it is possible to discover more useful patterns by leveraging semantic information as well as human interaction. Interest discovery is a valuable tool to improve the effectiveness of recommendations for restaurants, attractions, or public event to tourists. The three most common categories of interests discovery using trajectory data for tourism are (1) POI [66], (2) region of interest (ROI) [84, 153], and (3) interactive discovery [24, 167]. Palma et al. [126] discovered interesting locations from trajectories using a spatio-temporal clustering method, based on the speed of single trajectories. Uddin et al. [153] presented a generalized ROI definition for trajectories that is parameter independent, and an efficient index over the segments by speed was proposed to find the ROIs without scanning the whole dataset. Brilhante et al. [24] utilized Wikipedia content, trajectories over georeferenced Flickr photos, and human feedback to discover a "budget-constrained sightseeing tour" using a tourist's preferences and available time. Wang et al. [167] proposed a unified index, composed of inverted index and grid index, to find POIs, including attractions, hotels, and restaurants, to achieve fast performance for real-time response.

6.3.3 Semantic Pattern Mining. Generally, semantic pattern mining aims to mine the frequent movement with descriptive text information, which is more comprehensive than a spatial-only route. Zhang et al. [193] discovered fine-grained sequential patterns that satisfy spatial compactness, semantic consistency and temporal continuity simultaneously from semantic trajectories. The algorithm first groups all the places by category and retrieves a set of coarse patterns from the database, then splits a coarse pattern in a top-down manner. Instead of two steps in Reference [193], Kim et al. [87] presented a latent topic-based clustering algorithm to discover semantic patterns in the trajectories of geo-tagged text messages. However, the above method can only work over trajectories with texts. Choi et al. [40] studied a regional semantic trajectory pattern mining problem, aiming at identifying all the regional sequential patterns in semantic trajectories. Semantic pattern mining was not covered in previous surveys [127, 201]. Trajectory pattern mining was grouped into four categories by Zheng et al. [201]: (1) co-movement pattern that has been described in the traffic flow analytics, e.g., the convoy, flock, group, gathering; (2) trajectory clustering (check Section 5); (3) sequential patterns that indicate a certain number of moving objects traveling a common sequence of locations in a similar time interval; and (4) periodical patterns that indicate periodic behaviors for future movement prediction.

6.4 Applications in Site Selection

As a core decision-making tool, trajectory-driven site selection has been a crucial factor in increasing business profit and public service quality. Using collected trajectory data to estimate the influence of selected sites for drivers or passengers can be applied to problems such as charging station placement [102], billboard placement [71], and facility route design [163].

Definition 12 (Constrained Site Selection). Given a set of trajectories D, a set of facilities F, a constraint value C, and an influence model (IM), the aim of constrained site selection is to find a subset $S \subset F$ to that satisfies an objective function: $O = arg\,max_{cost(S)<C} IM(S, D)$.

Different objective functions can be defined based on the exact scenario. In Table 7, we compare recent work in this area in terms of constraints, input data, map matchability, objective function, NP-hardness reduction, acceleration strategies, and approximation guarantees.

An Overview. Figure 9 illustrates common connections between various site selection problems. Site selection definitions can change based on a specific problem scenario and may cover a singe dataset or multiple datasets. The general problem is NP-Hard by reduction to the set cover problem, and even approximate solutions tend to scale poorly. Bounded or greedy algorithms can be used

Table 7. Recent Work on Trajectory-driven Site Selection

	Work	Constraint	Extra Input Data besides Trajectory	Road Network	The Objective Function	The Reduced NP Problem	Accelerating Strategies	Guarantee
6.4.1	[102]	k stations	Existing stations, #charging points	✓	Travel time, waiting time	Integer programming	LP-rounding	N/A
	[76]	k stations	+Battery performance	✓	+installing cost, charging cost, waiting time	–	Evolution algorithm	–
	[106]	For the whole city	Petroleum stations, POIs, real-estate	✓	Revenue, queueing time	Bilevel optimization	Alternating framework	local minima
6.4.2	[71]	k billboards	Bus advertisement, bus station audiences	✗	Influence	Set-cover	Index, expansion, upper bound	$1-1/e$
	[107]	Budget or k location	Traffic volume, speed	✗	Coverage value	Maximum coverage	Inverted index, greedy heuristic	$1-1/e$
	[199]	Budget	Billboards' price	✗	One-time impression	Set cover problem	Bound	$1-1/e$
	[200]	Budget	Billboards' price	✗	logistic influence	biclique detection	Branch-and-bound	$\frac{\theta}{2}(1-1/e)$
	[159]	k	Advertisement topics, traffic conditions, mobility transition	✓	Influence spread	Weighted maximum coverage	Divide-and-conquer	N/A
6.4.3	[18]	Budget, construction cost, utilization	Bike trajectories	✓	beneficial score	0-1 knapsack problem	Greedy network expansion, inverted index	N/A
	[163]	Travel distance	Node capacity	✓	Route capacity	Constrained shortest path search	Divide-and-conquer	Exact
	[96]	Source, destination	Parking edges	✓	On-road travel time	Shortest path search	Incremental expansion	Exact
6.4.4	[160]	$k=1$	Probability threshold	✗	Cumulative influence probability	–	Filter-and-validate	Exact
	[101]	k	–	✓	#covered unique trajectories	Max-k-cover	Group pruning	$1-1/e$
	[122]	k	Existing facilities	✓	User inconvenience	k-center	Best-first	$1-1/e$

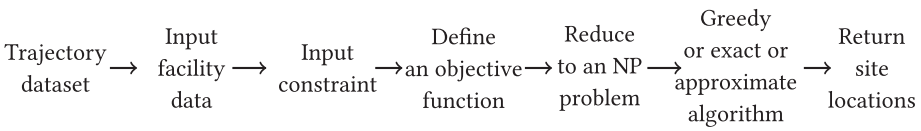


Fig. 9. A common pipeline decomposition of trajectory-driven site selection.

to accelerate the processing along with expansion-based methods, such as estimating the bound for all remaining set candidates by incrementally adding a single facility and comparing it against current result to determine whether expansion can be safely terminated.

6.4.1 Charging Station Deployment. Given the increasing popularity of electric vehicles, building more charging stations has become a crucial problem. Trajectory-driven charging station deployment aims to reduce the detour distance required for charging. Li et al. [102] developed an optimal charging station deployment framework that uses historical electric taxi trajectory data, road map data, and existing charging station information as input. Then, it performs charging station placement and charging point assignment (each charging station has multiple changing points). The objective function is designed to minimize the average time to the nearest

charging station, and the average waiting time for an available charging point, respectively, which can be reduced to the *integer linear programming* problem [138] that is NP-hard. Liu et al. [106] aimed to achieve two goals: (i) the overall revenue generated by the EVCs is maximized, subject to (ii) the overall driver discomfort (e.g., queueing time) for EV charging is minimized. Hence, the charging station deployment is defined as a multiple-objective optimization problem.

6.4.2 Billboard Placement. Billboard placement aims to find a limited number of billboards that maximize the influence on passengers and further increase profits. Recently, several studies [71, 107, 159, 199, 200] have investigated trajectory-driven billboard placement. Guo et al. [71] proposed the top-k trajectory influence maximization problem, which aims to find k trajectories for deploying billboards on buses to maximize the expected influence based on the audience. The model of Liu et al. [107] was based on traffic volume. More specifically, the algorithm counts how many trajectories traverse an edge or vertex containing a billboard, and then identifies the k most frequent for billboard placement. Zhang et al. [199] applied a model on range and one-time impressions, constrained by total budget available for billboard placement. Zhang et al. [200] proposed a logistic influence model that solves a key shortcoming in approaches that depend on one-time impressions [199], which did not consider the relationship between the influence effect and the impression counts for a single user. Wang et al. [159] placed billboards in a road network, and applied a divide-and-conquer strategy to accelerate the processing. As we conclude in Table 7, billboard placement problem definition always have a budget, and it is also general for other site selection problem in reality as public resource allocation also needs to be considered in the budget.

6.4.3 Facility Route Planning. Instead of pointwise candidate set selection, a facility set can also be routes covering multiple candidates, such as planning bike lanes and route search. Bao et al. [18] designed bike lanes based on bike trajectories, where the constraint is a budget and the number of connected components. A greedy network expansion algorithm was proposed to iteratively construct new lanes to reduce the number of connected components until the budget is met. Wang et al. [163] proposed the use of MaxRkNNT for planning bus routes between a given source and destination using a route capacity estimation query called **R*k*NNT**. The key constraint is travel distance, and the objective function maximizes the estimated capacity of the routes. Using transit trajectories, Wu et al. [179] defined various objective functions for passenger preferences, and evaluated new transit routes based on multiple factors on a real-world transport network, including monetary cost, time cost, number of transfers, number of choices, and transit mode.

6.4.4 General Site Selection. There are several general site selection studies that are not limited by any one scenario. Instead of setting a cost budget, it may be more desirable to set a parameter k to choose a set of facilities from the candidate set. Specifically, Wang et al. [160] aimed to identify the optimal location ($k = 1$) that can influence the maximum number of moving objects by using the probabilistic influence. An exact algorithm based on filtering-verification is proposed to prune many inferior candidate locations prior to computing the influence. Li et al. [101] identified the k most influential locations traversed by the maximum number of unique trajectories in a given spatial region with an efficient algorithm to find the location set using a greedy heuristic, in cases where k and the spatial areas are large. Mitra et al. [122] solved the trajectory-aware inconvenience-minimizing placement of k facility services that was proved to be NP-hard [27]. The inconvenience is defined as the extra distance traveled to access the nearest facility location.

7 EMERGING TRENDS ON DEEP TRAJECTORY LEARNING

Deep learning has been successfully used for trajectory data analytics and applications for the last several years, which is attracting increasing interest. We group these methods into several

categories, and discuss the associated research problems in detail in this section. We then cover the emerging trends in deep learning for trajectory data.

7.1 Trajectory Pre-processing

Trajectory Data Generation. Generating synthetic yet realistic location trajectories plays a fundamental role in privacy-aware analysis on trajectory data. A generative model was proposed for generating trajectories [125] based on a set of training trajectories. The proposed solution uses generative adversarial networks (GAN) to produce data points in a metric space, which are then transformed to a sequential trajectory. Experimental results show that the model is able to capture the correlation between visited locations in a trajectory and learn the common semantic/geographic mobility patterns from the training trajectories. The idea of using a GAN to generate synthetic trajectories that can preserve the summary properties of real trajectory data for data privacy was introduced by Liu et al. [110].

Trajectory Data Recovery. As many trajectories are recorded at a low sampling rate, the low-sampled trajectories cannot capture the correct routes of the objects. This problem has been studied in two settings previously, either with road networks using map matching [114], or without road networks [147], as discussed in Section 2.2. Wang et al. [158] aimed to recover the trajectory between two consecutive sampled points of "low-sampled" trajectory data. Using a seq2seq model, the proposed method uses spatial and temporal attention to capture spatiotemporal correlations and integrates a calibration component using a Kalman filter in order to reduce the uncertainty.

7.2 Trajectory Representation and Similarity Search

Representation learning of trajectories aims to represent trajectories as vectors of a fixed dimension. Then the similarity of two trajectories can be computed based on the Euclidean distance of their vector representations, which reduces the complexity of similarity computation from $O(n^2)$ to $O(n)$, where n is the trajectory length. Li et al. [100] proposed a seq2seq-based model to learn trajectory representations, where the spatial proximity is taken into account in the design of the loss function. The trajectory similarity based on the learned representations is robust to non-uniform, low sampling rates, and noisy sample points. Yao et al. [185] proposed the use of a deep metric learning framework to approximate any existing trajectory measure, and is capable of computing similarity for a given trajectory pair in linear time. The basic idea is to sample a number of seed trajectories from the given database and then use their pairwise similarities as guide to approximate the similarity function with a neural metric learning framework. The proposed solution adopts a new spatial attention memory module that augments existing recurrent neural networks (RNN) for trajectory encoding. Fu and Lee [63] proposed to exploit the road networks to learn the trajectory representation based on an encoder decoder model. Deep trajectory representation was also extended to multi-trajectory scenarios, such as trajectories encountered in sporting events with several football players [173]. Wang et al. [174] applied deep reinforcement learning to enable sub-trajectory similarity search, by splitting every trajectory into a set of sub-trajectories that can be candidate solutions for a query trajectory. By learning an optimal splitting policy, the sub-trajectory similarity search is more efficient than heuristics-based methods.

7.3 Deep Trajectory Clustering, Classification, and Anomaly Detection

Yao et al. [186] transformed trajectories into feature sequences that capture object movements, and then applied an autoencoder framework to learn fixed length deep representations of trajectories for clustering. Song et al. [145] proposed a model named DeepTransport to predict the transportation mode such as walk, taking trains, taking buses, etc., from a set of individual peoples GPS trajectories. Long short-term memory (LSTM) is used to constructed DeepTransport to

predict a user's transportation mode. Endo et al. [55] adopted stacked denoising autoencoder to automatically extract features for the transportation mode classification problem. Gao et al. [65] considered a different trajectory classification problem, namely Trajectory-User Linking (TUL), which aims to link trajectories to users who generate them in the location-based social networks. An RNN-based semi-supervised model was proposed to address the TUL problem. Liu et al. [112] proposed a deep generative model, namely a Gaussian Mixture Variational Sequence AutoEncoder, for online anomalous trajectory detection based on Variational autoencoding.

7.4 Trajectory Prediction for Human Mobility Analytics

RNN are widely used for trajectory prediction, and most of the work reviewed also learns user representations from trajectories to capture user preferences in addition to the spatio-temporal contexts. Liu et al. [108] proposed Spatial-Temporal RNN (ST-RNN) for the next location prediction of a given user using an RNN. An ST-RNN model local temporal contexts, periodical temporal contexts, and geographical contexts to learn the representation of users under specific contexts. Zhou et al. [209] adopted a seq2seq encoder-decoder framework consisting of two encoders and two decoders to predict future trajectories. This method does not explicitly learn user representations. Wu et al. [180] designed two RNN models to model trajectories that consider road network constraints on trajectories, and can be used to predict the destination of a trajectory.

DeepMove [58] is based on a multi-modal embedding RNN to capture the complex sequential transitions by jointly embedding multiple factors such as time, user, and location. DeepMove also applies attention mechanism to capture the periodical effect of mobility. Chen et al. [36] proposed a context-aware deep model called DeepJMT for jointly performing mobility prediction (to know where) and time prediction (to know when). DeepJMT captures the user's mobility regularities and temporal patterns using RNN, captures spatial context, periodicity context and social and temporal context using various mechanisms, e.g., co-attention mechanism to capture, and make time prediction using temporal point process.

A convolutional neural network (CNN) can also be used for trajectory prediction. Karatzoglou et al. [86] proposed a CNN-based approach for representing semantic trajectories and predicting future locations. The semantic trajectories are represented as a matrix of semantic meanings and trajectory IDs. A CNN is applied to the matrix to learn the latent features for next visited semantic location prediction. Gao et al. [64] developed a deep generative model called Variational Attention-based Next POI prediction, which simultaneously learns implicit relationships between users and POIs, and captures sequential patterns of user check-in behavior for next POI prediction. A CNN is used to capture long term and structural dependency among user check-ins, achieving comparable learning ability with the state-of-the-art RNN-based methods, while significantly improving the learning efficiency. Lv et al. [116] proposed to model trajectories as two-dimensional images, and employed CNN for trajectory destination prediction.

7.5 Travel Time and Route Estimation

Wang et al. [155] aimed to estimate the travel time of a path from the mobility trajectory data. Their approach used a CNN and an RNN to capture the features of a path, and employed a multi-task learning component to estimate the travel time. Zhang et al. [197] developed a bidirectional LSTM-based deep model, called DeepTravel, to estimate the travel time of a path from the historical trajectories. Li et al. [99] proposed a deep generative model, DeepGTT, to learn the travel time distribution for a route by conditioning on the real-time traffic, which is captured by the trajectory data. Li et al. [98] proposed a deep probabilistic model, DeepST, which unifies three key explanatory factors, the past traveled trajectories, the impact of destination, and real-time traffic for the route decision of a pair of source and destination. Yuan et al. [191] estimated the travel time of an

origin-destination pair at a certain departure time, and proposed a neural network–based prediction model. This model exploits the fact that for a past OD trip its travel time is usually affiliated with the trajectory it travels along, whereas it does not exist during prediction.

8 FUTURE RESEARCH DIRECTIONS AND OPEN ISSUES

Trajectory data have inspired many important research problems and applications in both academia and industry. However, open-source and commercial systems that can support the entire pipeline of trajectory data management and analytics are still non-existent. To better cater to future applications, a unified trajectory data management system would be an important contribution to the community. Desirata of such a system include the following:

- **Data Cleaning:** Map matching, re-sampling, and calibration are fundamental building blocks in efficient and effective analytics. However, current solutions need to input a road network dataset given by users manually, and also need to input several data-dependent parameters. An automated data cleaning pipeline for trajectory data is highly desirable as manual data cleaning is time-consuming and often not reproducible.
- **Trajectory Data Repositories:** Cleaning trajectory is computationally expensive. Publicly available data repositories of raw and cleaned data would not only improve reproducibility, it would also stimulate new research in the area. A standard data format for trajectory data is also not currently defined. Even though LineString and GPX can be used, their use is limited. Labeled datasets for similarity search and classification problems are also non-existent, hampering progress on this important problem.
- **Data Integration and Operator Support:** In more advanced data analytics applications, trajectory data are often heterogeneous. Data integration between trajectory databases and other databases, including spatial databases (point data) and graph databases (road networks) would be a valuable contribution. Trajectory data from multiple devices, including cameras, UAV, and RFID can also be integrated to build more comprehensive profiles of a city. Unstructured metadata from various sources also plays a vital role when apply trajectory data in real-world application. These data might include speed limits, transit timetables, and public service opening hours. While this information is used commonly in commercial applications, the cost of using proprietary data sources is often too high in the research community.
- **Performance Benchmarks:** A trajectory performance benchmark similar to the TPC benchmark[15] would greatly improve the quality of empirical comparisons of new algorithms in research papers. There are many different distance measures and indexing data structures being applied to trajectory search with little understanding of their true performance characteristics. For efficiency, pruning power and I/O are important factors in total running time; for the effectiveness, no ground truth means that search quality cannot truly be measured.
- **Parameterization:** Parameter selection plays a critical role in system performance of trajectory analytics. Parameter-free algorithms are highly desirable, but not always possible. Automated parameter selection is an essential requirement for automatic databases, and for intelligent trajectory analytics. It would also be interesting to investigate how to automate bounding approximation ratios can be achieved during algorithm design.
- **Deep Trajectory Learning:** Deep learning has made some progress in synthetic data generation, representation learning, and mobility prediction. More advanced tasks, including

[15] http://www.tpc.org/information/benchmarks.asp.

query optimization and learned index, can be investigated to improve the query performance. Another promising topic can be the online deep trajectory learning to meet the demand for timely decision making over dynamic trajectory data.

- **Self-Driving Trajectory:** Self-driving [28, 118] will be one of the primary sources of trajectories in the near future. Different from the typical long-term trajectories being currently analyzed in road-network that is the focus in this survey, self-driving trajectories will be short-term lane-level traces with high sampling rates, e.g., each trajectory only lasts for five seconds [28]. Due to the requirement of real-time response for safe driving, managing and analyzing such trajectories will be crucial and challenging, especially in a streaming scenario.
- **Trajectory Analysis for Public Health:** Human movement data recorded by mobile phones has been analyzed to control Malaria in Africa [177] and dengue epidemics in Pakistan [178]. With mass outbreaks of diseases such as COVID-19 [5, 8] in dense urban areas, collecting and analyzing the trajectories of infected people and building real-time warning mechanisms will play an important role in disease traceability, disease control, and emergency response.

9 CONCLUSIONS

In this survey, we have reviewed recent progress in trajectory data management and learning. We first presented an overview of trajectory data management and urban applications. Then, we categorized the problems based on components and operators shared across the tasks. Similarity measures, top-k similarity search, and fast clustering are all widely used for many different problems and scenarios, and were the focus of this study. Finally, we reviewed important research advances on four common analytics applications of trajectories for real-time smart cities. New applications emerge daily that leverage trajectory data, such as deep trajectory learning, and we hope this survey can provide readers an overview of the landscape of trajectory data management and applications. Perspectives on how to choose the most appropriate solution for pre-processing, storage, search, and advanced analytics to achieve high efficiency and effectiveness can be applied in many different problem domains beyond road networks.

REFERENCES

[1] 2013. Movebank. Retrieved from https://www.movebank.org/.
[2] 2015. Taxi Service Trajectory Prediction Challenge 2015. Retrieved from http://www.geolink.pt/ecmlpkdd2015-challenge/.
[3] 2017. How to Track Vehicles with RFID. Retrieved from https://www.asiarfid.com/rfid-journal/how-to-track-vehicles-with-rfid.html.
[4] 2018. Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data. Retrieved from https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories.
[5] 2019. Data Science for COVID-19 (DS4C). Retrieved from https://www.kaggle.com/kimjihoo/coronavirusdataset.
[6] 2019. Greece Trucks. Retrieved from http://www.chorochronos.org/.
[7] 2019. NHC Data Archive. Retrieved from https://www.nhc.noaa.gov/.
[8] 2019. Novel Corona Virus 2019 Dataset. Retrieved from https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset.
[9] 2019. OpenStreetMap: Public GPS traces. Retrieved from https://www.openstreetmap.org/traces.
[10] 2019. TLC Trip Record Data. Retrieved from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[11] Afshin Abadi, Tooraj Rajabioun, and Petros A. Ioannou. 2015. Traffic flow prediction for road transportation networks with limited traffic data. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 653–662.
[12] Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. 2018. Subtrajectory clustering: Models and algorithms. In *PODS*. 75–87.
[13] Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Antonio Fernandes de Macedo, Bart Moelans, and Alejandro Vaisman. 2007. A model for enriching trajectories with semantic geographical information. In *GIS*.

[14] Carlos Andres, Luis Otávio Alvares, Willian Zalewski, Vania Bogorny, and Luis Otavio Alvares. 2018. MOVELETS: Exploring relevant subtrajectories for robust trajectory classification. In *SAC*. 849–856.
[15] Gennady Andrienko, Natalia Andrienko, Salvatore Rinzivillo, Mirco Nanni, Dino Pedreschi, and Fosca Giannotti. 2009. Interactive visual clustering of large collections of trajectories. In *VAST*. 273–274.
[16] Samet Ayhan and Hanan Samet. 2016. Aircraft trajectory prediction made easy with predictive analytics. In *KDD*. 21–30.
[17] Petko Bakalov and Vassilis J. Tsotras. 2008. Continuous spatiotemporal trajectory joins. In *GSN*. 109–128.
[18] Jie Bao, Tianfu He, Sijie Ruan, and Yu Zheng. 2017. Planning bike lanes based on sharing-bikes' trajectories. In *KDD*. 1377–1386.
[19] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *SIGSPATIAL*. 199–208.
[20] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*. 322–331.
[21] Tolga Bektacs, Emrah Demir, and Gilbert Laporte. 2016. Green vehicle routing. In *Green Transportation Logistics*. Springer, 243–265.
[22] Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. 2019. Trajectory data classification: A review. *ACM Trans. Intell. Syst. Technol.* 10, 4 (2019), 1–34.
[23] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. 2019. *The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections*. Technical Report.
[24] Igo Ramalho Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2015. On planning sightseeing tours with TripBuilder. *Inf. Process. Manage.* 51, 2 (2015), 1–15.
[25] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *CIKM*. 426–434.
[26] Maike Buchin, Anne Driemel, Marc Van Kreveld, and Vera Sacristan. 2011. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spatial Inf. Sci.* 3, 3 (2011), 33–63.
[27] T.-h. Hubert Chan, Arnaud Guerquin, and Mauro Sozio. 2018. Fully dynamic k-center clustering. In *WWW*. 579–587.
[28] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. 2019. Argoverse: 3D tracking and forecasting with rich maps. In *CVPR*. 8748–8757.
[29] Muhammad Aamir Cheema, Wenjie Zhang, Xuemin Lin, Ying Zhang, and Xuefei Li. 2012. Continuous reverse k nearest neighbors queries in Euclidean space and in spatial networks. *VLDB J.* 21, 1 (2012), 69–95.
[30] Chao Chen, Chengwu Liao, Xuefeng Xie, Yasha Wang, and Junfeng Zhao. 2019. Trip2Vec: A deep embedding approach for clustering and profiling taxi trip purposes. *Pers. Ubiq. Comput.* 23, 1 (2019), 53–66.
[31] Chao Chen, Daqing Zhang, Nan Li, and Zhi Hua Zhou. 2014. B-planner: Planning bidirectional night bus routes using large-scale taxi GPS traces. *IEEE Trans. Intell. Transport. Syst.* 15, 4 (2014), 1451–1465.
[32] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, and Shijian Li. 2011. Real-time detection of anomalous taxi trajectories from GPS traces. In *MobiQuitous*. 63–74.
[33] Lei Chen and Raymond Ng. 2004. On the marriage of Lp-norms and edit distance. In *VLDB*. 792–803.
[34] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *SIGMOD*. 491–502.
[35] Siming Chen, Xiaoru Yuan, Zhenhuang Wang, Cong Guo, Jie Liang, Zuchao Wang, and Jiawan Zhang. 2015. Interactive visual discovering of movement patterns from sparsely. *IEEE Trans. Vis. Comput. Graph.* 22, 1 (2015), 1–1.
[36] Yile Chen, Cheng Long, Gao Cong, and Chengliang Li. 2020. Context-aware deep model for joint mobility and time prediction. In *WSDM*. 106–114.
[37] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *ICDE*. 900–911.
[38] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. 2010. Searching trajectories by locations-an efficiency study. In *SIGMOD*. 255–266.
[39] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *KDD*. 1082–1090.
[40] Dong-wan Choi, Jian Pei, and Thomas Heinis. 2017. Efficient mining of regional movement patterns in semantic trajectories. *Proc. VLDB* 10, 13 (2017), 2073–2084.
[41] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proc. VLDB* 2, 1 (2009), 337–348.
[42] Gao Cong, Hua Lu, Beng Chin Ooi, Dongxiang Zhang, and Meihui Zhang. 2012. Efficient spatial keyword search in trajectory databases. *arXiv:1205.2880*. Retrieved from https://arxiv.org/abs/1205.2880.

[43] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press.
[44] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. 2010. TrajStore: An adaptive storage system for very large trajectory data sets. In *ICDE*. 109–120.
[45] Sina Dabiri, Chang-Tien Lu, Kevin Heaslip, and Chandan K. Reddy. 2020. Semi-supervised deep learning approach for transportation mode identification using GPS trajectory data. *IEEE Trans. Knowl. Data Eng.* 32, 5 (2020), 1010–1023.
[46] Jian Dai, Bin Yang, Chenjuan Guo, and Zhiming Ding. 2015. Personalized route recommendation using big trajectory data. In *ICDE*. 543–554.
[47] Jian Dai, Bin Yang, and Christian S. Jensen. 2016. Path cost distribution estimation using trajectory data. *Proc. VLDB* 10, 3 (2016), 85–96.
[48] Eleonora D'Andrea and Francesco Marcelloni. 2017. Detection of traffic congestion and incidents from GPS trace analysis. *Expert Syst. Appl.* 73 (2017), 43–56.
[49] Victor Teixeira de Almeida and Ralf Hartmut Güting. 2005. Indexing the trajectories of moving objects in networks. *GeoInformatica* 9, 1 (2005), 33–60.
[50] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *ICDE*. 656–665.
[51] Daniel Delling, Andrew V. Goldberg, Moises Goldszmidt, John Krumm, Kunal Talwar, and Renato F. Werneck. 2016. Navigation made personal: Inferring driving preferences from GPS traces. In *SIGSPATIAL*. 1–9.
[52] Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *SIGIR*. 993–1002.
[53] Xin Ding, Lu Chen, Yunjun Gao, Christian S. Jensen, and Hujun Bao. 2018. UlTraMan: A unified platform for big trajectory data management and analytics. *Proc. VLDB* 11, 7 (2018), 787–799.
[54] Thomas Eiter and Heikki Mannila. 1994. *Computing Discrete Fréchet Distance*. Technical Report. CD–TR 94/64 pages.
[55] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Jotaro Ikedo. 2016. Classifying spatial trajectories using representation learning. *Int. J. Data Sci. Anal.* 2, 3 (2016), 107–117.
[56] Ronald Fagin, Amnon Lotem, and Moni Naor. 2003. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.* 66, 4 (2003), 614–656.
[57] Yixiang Fang, Reynold Cheng, Wenbin Tang, Silviu Maniu, and Xuan Yang. 2016. Scalable algorithms for nearest-neighbor joins on big trajectory data. *IEEE Trans. Knowl. Data Eng.* 28, 3 (2016), 785–800.
[58] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *WWW*. 1459–1468.
[59] Zhenni Feng and Yanmin Zhu. 2016. A survey on trajectory data mining: Techniques and applications. *IEEE Access* 4 (2016), 2056–2067.
[60] Nivan Ferreira, James T. Klosowski, Carlos E. Scheidegger, and Cláudio T. Silva. 2013. Vector field k-means: Clustering trajectories by fitting multiple vector fields. *Comput. Graph. Forum* 32, 3 (2013), 201–210.
[61] Dan Foster. 2004. GPX the GPS Exchange Format. Retrieved from http://www.topografix.com/gpx.asp.
[62] Elias Frentzos, Kostas Gratsias, and Yannis Theodoridis. 2007. Index-based most similar trajectory search. In *ICDE*. 816–825.
[63] Tao Yang Fu and Wang Chien Lee. 2020. TremBR: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.* 11, 1 (2020), 1–25.
[64] Qiang Gao, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. Predicting human mobility via variational attention. In *WWW*. 2750–2756.
[65] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying human mobility via trajectory embeddings. In *IJCAI*. 1689–1695.
[66] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. Trajectory pattern mining. In *KDD*. 330–339.
[67] Li Gong, Xi Liu, Lun Wu, and Yu Liu. 2016. Inferring trip purposes and uncovering travel patterns from taxi trajectory data. *Cartogr. Geogr. Inf. Sci.* 43, 2 (2016), 103–114.
[68] Joachim Gudmundsson and Nacho Valladares. 2012. A GPU approach to subtrajectory clustering using the Fréchet distance. In *SIGSPATIAL*. 259–268.
[69] Joachim Gudmundsson and Marc Van Kreveld. 2006. Computing longest duration flocks in trajectory data. In *GIS*. 35–42.
[70] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian Jensen. 2018. Learning to route with sparse trajectory sets. In *ICDE*. IEEE, 1085–1096.
[71] Long Guo, Dongxiang Zhang, Gao Cong, Wei Wu, and Kian Lee Tan. 2017. Influence maximization in trajectory databases. *IEEE Trans. Knowl. Data Eng.* 29, 3 (2017), 627–641.
[72] Mingming Guo, Xinyu Jin, Niki Pissinou, Sebastian Zanlongo, Bogdan Carbunar, and S. S. Iyengar. 2015. In-network trajectory privacy preservation. *ACM Comput. Surv.* 48, 23 (2015).

[73] Ralf Hartmut Güting, Fabio Valdés, and Maria Luisa Damiani. 2015. Symbolic trajectories. *ACM Trans. Spatial Algor. Syst.* 1, 2 (2015), 7:1–7:51.
[74] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD Rec.* 14, 2 (1984), 47–57.
[75] Binh Han, Ling Liu, and Edward Omiecinski. 2015. Road-network aware trajectory clustering: Integrating locality, flow, and density. *IEEE Trans. Mobile Comput.* 14, 2 (2015), 416–429.
[76] Daehee Han, Yongjun Ahn, Sunkyu Park, and Hwasoo Yeo. 2016. Trajectory-interception based method for electric vehicle taxi charging station problem with real taxi data. *Int. J. Sust. Transport.* 10, 8 (2016), 671–682.
[77] Yuxing Han, Liping Wang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. 2015. Spatial keyword range search on trajectories. In *DASFAA*. 223–240.
[78] Ramaswamy Hariharan, Bijit Hore, Li Chen, and Sharad Mehrotra. 2007. Processing spatial-keyword (SK) queries in Geographic Information Retrieval (GIR) systems. In *SSDBM*. 16–25.
[79] Wen He, Kai Hwang, and Deyi Li. 2014. Intelligent carpool routing for urban ridesharing by mining GPS trajectories. *IEEE Trans. Intell. Transport. Syst.* 15, 5 (2014), 2286–2296.
[80] Zihan Hong, Ying Chen, Hani S. Mahmassani, and Shuang Xu. 2017. Commuter ride-sharing using topology-based vehicle trajectory clustering: Methodology, application and impact evaluation. *Transport. Res. C: Emerg. Technol.* 85 (Oct. 2017), 573–590.
[81] Fu Shiung Hsieh. 2017. Car pooling based on trajectories of drivers and requirements of passengers. In *AINA*. IEEE, 972–978.
[82] Chih Chieh Hung, Wen Chih Peng, and Wang Chien Lee. 2015. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *VLDB J.* 24, 2 (2015), 169–192.
[83] G. R. Jagadeesh, T. Srikanthan, and X. D. Zhang. 2004. A map matching method for GPS based real-time vehicle location. *J. Nav.* 57, 3 (2004), 429–440.
[84] Priit Jarv, Tanel Tammet, and Marten Tall. 2018. Hierarchical regions of interest. In *MDM*. 86–95.
[85] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. 2008. Discovery of convoys in trajectory databases. *Proc. VLDB* 1, 1 (2008), 1068–1080.
[86] Antonios Karatzoglou, Nikolai Schnell, and Michael Beigl. 2018. A convolutional neural network approach for modeling semantic trajectories and predicting future locations antonios. In *ICANN*. 61–72.
[87] Younghoon Kim, Jiawei Han, and Cangzhou Yuan. 2015. TOPTRAC: Topical trajectory pattern mining. In *KDD*. 587–596.
[88] Satoshi Koide, Yukihiro Tadokoro, and Takayoshi Yoshimura. 2015. SNT-index: Spatio-temporal index for vehicular trajectories on a road network based on substring matching. In *UrbanGIS@SIGSPATIAL*. 1–8.
[89] Satoshi Koide, Chuan Xiao, and Yoshiharu Ishikawa. 2020. Fast subtrajectory similarity search in road networks under weighted edit distance constraints. *Proc. VLDB* 13, 11 (2020), 2188–2201.
[90] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *ITSC*.
[91] Benjamin Krogh and Christian S. Jensen. 2016. Efficient in-memory indexing of network-constrained trajectories. In *SIGSPATIAL*. 17:1–17:10.
[92] Scott LaPoint, Paul Gallery, Martin Wikelski, and Roland Kays. 2013. Animal behavior, cost-based corridor models, and real corridors. *Landscape Ecol.* 28, 8 (2013), 1615–1630.
[93] Jae Gil Lee, Jiawei Han, and Xiaolei Li. 2008. Trajectory outlier detection: A partition-and-detect framework. In *ICDE*. 140–149.
[94] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. 2008. TraClass: Trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB* 1, 1 (2008), 1081–1094.
[95] Jae-gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: A partition-and-group framework. In *SIGMOD*. 593–604.
[96] Lei Li, Kai Zheng, Sibo Wang, Wen Hua, and Xiaofang Zhou. 2018. Go slow to go fast: Minimal on-road time route scheduling with parking facilities using historical trajectory. *VLDB J.* 27, 3 (2018), 321–345.
[97] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Yuan Sui, Jie Bao, and Yu Zheng. 2020. TrajMesa: A distributed nosql storage engine for big trajectory data. In *ICDE*. 2002–2005.
[98] Xiucheng Li, Gao Cong, and Yun Cheng. 2020. Spatial transition learning on road networks with deep probabilistic models. In *ICDE*. 349–360.
[99] Xiucheng Li, Aixin Sun, Gao Cong, and Yun Cheng. 2019. Learning travel time distributions with deep generative model. In *WWW*. 1017–1027.
[100] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*. 617–628.

[101] Yuhong Li, Jie Bao, Yanhua Li, Yingcai Wu, Zhiguo Gong, and Yu Zheng. 2016. Mining the most influential k -location set from massive trajectories. *IEEE Trans. Big Data* 4, 4 (2016), 556–570.
[102] Yanhua Li, Jun Luo, Chi Yin Chow, Kam Lam Chan, Ye Ding, and Fan Zhang. 2015. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*. 1376–1387.
[103] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *Proc. VLDB* 3, 1 (2010), 723–734.
[104] Zhenhui Li, Jae-Gil Lee, Xiaolei Li, and Jiawei Han. 2010. Incremental clustering for trajectories. In *DASFAA*. 32–46.
[105] Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang Chien Lee, Dik Lee, and Xufa Wang. 2011. IR-tree: An efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.* 23, 4 (2011), 585–599.
[106] Chen Liu, Ke Deng, Chaojie Li, Jianxin Li, Yanhua Li, and Jun Luo. 2016. The optimal distribution of electric-vehicle chargers across a city. In *ICDM*. 261–270.
[107] Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. 2017. SmartAdP: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *IEEE Trans. Vis. Comput. Graph.* 23, 1 (2017), 1–10.
[108] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*. 194–200.
[109] Siyuan Liu, Ce Liu, Qiong Luo, Lionel M. Ni, and Ramayya Krishnan. 2012. Calibrating large scale vehicle trajectory data. In *MDM*. 222–231.
[110] Xi Liu, Hanzhou Chen, and Clio Andris. 2018. trajGANs: Using generative adversarial networks for geo-privacy protection of trajectory data (Vision paper). In *Location Privacy and Security Workshop*. 1–7.
[111] Yanchi Liu, Chuanren Liu, Nicholas Jing Yuan, Lian Duan, Yanjie Fu, Hui Xiong, Songhua Xu, and Junjie Wu. 2016. Intelligent bus routing with heterogeneous human mobility patterns. *Knowl. Inf. Syst.* 50, 2 (2016), 383–415.
[112] Yiding Liu, Kaiqi Zhao, Gao Cong, and Zhifeng Bao. 2020. Online anomalous trajectory detection with deep generative sequence modeling. In *ICDE*. 949–960.
[113] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. 2013. Direction-preserving trajectory simplification. *Proc. VLDB* 6, 10 (2013), 949–960.
[114] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *GIS*. 352–361.
[115] Xin Lu, Changhu Wang, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. 2010. Photo2Trip: Generating travel routes from geo-tagged photos for trip planning. In *MM*. 143–152.
[116] Jianming Lv, Qinghui Sun, Qing Li, and Luis Moreira-Matias. 2020. Multi-scale and multi-scope convolutional neural networks for destination prediction of trajectories. *IEEE Trans. Intell. Transport. Syst.* 21, 8 (2020), 3184–3195.
[117] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei Yue Wang. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 865–873.
[118] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. 2019. TrafficPredict: Trajectory prediction for heterogeneous traffic-agents. In *AAAI*. 6120–6127.
[119] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press.
[120] 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).
[121] Nikola Markovic, Przemylaw Sekula, Zachary Vander Laan, Gennady Andrienko, and Natalia Andrienko. 2019. Applications of trajectory data from the perspective of a road transportation agency: Literature review and maryland case study. *IEEE Trans. Intell. Transport. Syst.* 20, 5 (2019), 1858–1869.
[122] Shubhadip Mitra, Priya Saraf, and Arnab Bhattacharya. 2021. TIPS: Mining top-k locations to minimize user-inconvenience for trajectory-aware services. *IEEE Trans. Knowl. Data Eng.* 33, 3 (2021).
[123] Suraj Nair, Kiran Javkar, Jiahui Wu, and Vanessa Frias-Martinez. 2019. Understanding cycling trip purpose and route choice using GPS traces and open data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–26.
[124] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL*. 336–343.
[125] Kun Ouyang, Reza Shokri, David S. Rosenblum, and Wenzhuo Yang. 2018. A non-parametric generative model for human trajectories. In *IJCAI*. 3812–3817.
[126] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. 2008. A clustering-based approach for discovering interesting places in trajectories. In *SAC*. 863–868.
[127] Christine Parent, Nikos Pelekis, Yannis Theodoridis, Zhixian Yan, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, and Jose Macedo. 2013. Semantic trajectories modeling and analysis. *Comput. Surv.* 45, 4 (2013), 1–32.

[128] Dhaval Patel, Chang Sheng, Wynne Hsu, and Mong Li Lee. 2012. Incorporating duration information for trajectory classification. In *ICDE*. 1132–1143.
[129] Nikos Pelekis, Ioannis Kopanakis, Evangelos E. Kotsifakos, Elias Frentzos, and Yannis Theodoridis. 2009. Clustering trajectories of moving objects in an uncertain world. In *ICDM*. 417–427.
[130] Fabio Pinelli, Rahul Nair, Francesco Calabrese, Michele Berlingerio, Giusy Di Lorenzo, and Marco Luca Sbodio. 2016. Data-driven transit network design from mobile phone trajectories. *IEEE Trans. Intell. Transport. Syst.* 17, 6 (2016), 1724–1733.
[131] Shuyao Qi, Panagiotis Bouros, Dimitris Sacharidis, and Nikos Mamoulis. 2015. Efficient point-based trajectory search. In *SSTD*. 179–196.
[132] Sayan Ranu, P. Deepak, Aditya D. Telang, Prasad Deshpande, and Sriram Raghavan. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *ICDE*. 999–1010.
[133] Keven Richly. 2018. A survey on trajectory data management for hybrid transactional and analytical workloads. In *BigData*. 562–569.
[134] Gook-pil Roh, Jong-won Roh, Seung-won Hwang, and Byoung-kee Yi. 2011. Supporting pattern-matching queries over trajectories on road networks. *IEEE Trans. Knowl. Data Eng.* 23, 11 (2011), 1753–1758.
[135] Simonas Šaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. 2000. Indexing the positions of continuously moving objects. In *SIGMOD*. 331–342.
[136] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. 2011. Indexing in-network trajectory flows. *VLDB J.* 20, 5 (2011), 643–669.
[137] Preston L. Schiller and Jeffrey R. Kenworthy. 2017. *An Introduction to Sustainable Transportation: Policy, Planning and Implementation*. Routledge.
[138] Alexander Schrijver. 1998. *Theory of Linear and Integer Programming*. John Wiley & Sons.
[139] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. 2017. Trajectory similarity join in spatial networks. *Proc. VLDB* 10, 11 (2017), 1178–1189.
[140] Shuo Shang, Ruogu Ding, Bo Yuan, Kexin Xie, Kai Zheng, and Panos Kalnis. 2012. User oriented trajectory search for trip recommendation. In *EDBT*. 156–167.
[141] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. Dita: Distributed in-memory trajectory analytics. In *SIGMOD*. 725–740.
[142] Lokesh K. Sharma, Om Prakash Vyas, Simon Schieder, and Ajaya K. Akasapu. 2010. Nearest neighbour classification for trajectory data. In *ICT*. 180–185.
[143] Minxin Shen, Duen Ren Liu, and Shi Han Shann. 2015. Outlier detection from vehicle trajectories to discover roaming events. *Inf. Sci.* 294 (2015), 242–254.
[144] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. 2014. PRESS: A novel framework of trajectory compression in road networks. *Proc. VLDB* 7, 9 (2014), 661–672.
[145] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. 2016. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *IJCAI*. 2618–2624.
[146] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29 (2020), 3–32.
[147] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. 2013. Calibrating trajectory data for similarity-based analysis. In *SIGMOD*. 833–844.
[148] 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 (2017), 870–883.
[149] Lu An Tang, Yu Zheng, Xing Xie, Jing Yuan, Xiao Yu, and Jiawei Han. 2011. Retrieving k-nearest neighboring trajectories by a set of point locations. In *SSTD*. 223–241.
[150] Yufei Tao, D. Papadias, and Xiang Lian. 2004. Reverse kNN search in arbitrary dimensionality. In *VLDB*. 744–755.
[151] Eleftherios Tiakas, Apostolos Papadopoulos, Alexandros Yannis Manolopoulos Nanopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. 2009. Searching for similar trajectories in spatial networks. *J. SystSoftw.* 82, 5 (2009), 772–788.
[152] Jameson L. Toole, Serdar Colak, Bradley Sturt, Lauren P. Alexander, Alexandre Evsukoff, and Marta C. González. 2015. The path most traveled: Travel demand estimation using big data resources. *Transport. Res. C: Emerg. Technol.* 58, Part B (2015), 162–177.
[153] Md Reaz Uddin, Chinya Ravishankar, and Vassilis J. Tsotras. 2011. Finding regions of interest from trajectory data. In *MDM*. 39–48.
[154] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 2002. Discovering similar multidimensional trajectories. In *ICDE*. 673–684.
[155] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *AAAI*. 2500–2507.

[156] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. 2013. An effectiveness study on trajectory similarity measures. In *ADC*. 13–22.
[157] Haozhou Wang, Kai Zheng, Jiajie Xu, Bolong Zheng, and Xiaofang Zhou. 2014. SharkDB: An in-memory column-oriented trajectory storage. In *CIKM*. 1409–1418.
[158] Jingyuan Wang, Ning Wu, Xinxi Lu, Xin Zhao, and Kai Feng. 2021. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE Trans. Knowl. Data Eng.* 33, 3 (2021).
[159] Liang Wang, Zhiwen Yu, Dingqi Yang, Huadong Ma, and Hao Sheng. 2019. Efficiently targeted billboard advertising using crowdsensing vehicle trajectory data. *IEEE Trans. Industr. Inf.* 16, 2 (2019), 1058–1066.
[160] Meng Wang, Hui Li, Jiangtao Cui, Ke Deng, Sourav S. Bhowmick, and Zhenhua Dong. 2016. Pinocchio: Probabilistic influence-based location selection over moving objects. *IEEE Trans. Know. Data Eng.* 28, 11 (2016), 3068–3082.
[161] Pengfei Wang, Guannan Liu, Yanjie Fu, Yuanchun Zhou, and Jianhui Li. 2017. Spotting trip purposes from taxi trajectories: A general probabilistic model. *ACM Trans. Intell. Syst. Technol.* 9, 3 (2017).
[162] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2020. *A Survey on Trajectory Data Management Analytics and Learning*. Technical Report. arxiv:2003.11547. Retrieved from http://arxiv.org/abs/2003.11547.
[163] Sheng Wang, Zhifeng Bao, J Shane Culpepper, Timos Sellis, and Gao Cong. 2018. Reverse k nearest neighbor search over trajectories. *IEEE Trans. Knowl. Data Eng.* 30, 4 (2018), 757–771.
[164] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, and Xiaolin Qin. 2019. Fast large-scale trajectory clustering. *Proc. VLDB* 13, 1 (2019), 29–42.
[165] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, Mark Sanderson, and Xiaolin Qin. 2017. Answering top-k exemplar trajectory queries. In *ICDE*. 597–608.
[166] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Zizhe Xie, Qizhi Liu, and Xiaolin Qin. 2018. Torch: A search engine for trajectory data. In *SIGIR*. 535–544.
[167] Sheng Wang, Zhifeng Bao, Shixun Huang, and Rui Zhang. 2018. A unified processing paradigm for interactive location-based web search. In *WSDM*. 601–609.
[168] Shuang Wang and Hakan Ferhatosmanoglu. 2021. PPQ-trajectory: Spatio-temporal quantization for querying in large trajectory repositories. *Proc. VLDB* 14, 2 (2021), 215–227.
[169] Sheng Wang, Mingzhao Li, Yipeng Zhang, Zhifeng Bao, David Alexander Tedjopurnomo, and Xiaolin Qin. 2018. Trip planning by an integrated search paradigm. In *SIGMOD*. 1673–1676.
[170] Sheng Wang, Yunzhuang Shen, Zhifeng Bao, and Xiaolin Qin. 2019. Intelligent traffic analytics: From monitoring to controlling. In *WSDM*. 778–781.
[171] Yong Wang, Guoliang Li, and Nan Tang. 2019. Querying shortest paths on time dependent road networks. *PVLDB* 12, 11 (2019), 1249–1261.
[172] Yulong Wang, Kun Qin, Yixiang Chen, and Pengxiang Zhao. 2018. Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi GPS data. *ISPRS Int. J. Geo-Inf* 7, 1 (2018), 25.
[173] Zheng Wang, Cheng Long, Gao Cong, and Ce Ju. 2019. Effective and efficient sports play retrieval with deep representation learning. In *KDD*. 499–509.
[174] Zheng Wang, Cheng Long, Gao Cong, and Yiding Liu. 2020. Efficient and effective similar subtrajectory search with deep reinforcement learning. *Proc. VLDB* 13, 11 (2020), 2312–2325.
[175] Zuchao Wang, Min Lu, Xiaoru Yuan, Junping Zhang, and Huub Van De Wetering. 2013. Visual traffic jam analysis based on trajectory data. *IEEE Trans. Vis. Comput. Graph.* 19, 12 (2013), 2159–2168.
[176] Ling-Yin Wei, Wen-Chih Peng, and Wang-Chien Lee. 2013. Exploring pattern-aware travel routes for trajectory search. *ACM Trans. Intell. Syst. Technol.* 4, 3 (2013), 1.
[177] Amy Wesolowski, Nathan Eagle, Andrew J. Tatem, David L. Smith, Abdisalan M. Noor, Robert W. Snow, and Caroline O. Buckee. 2012. Quantifying the impact of human mobility on Malaria. *Science* 338, 6104 (Oct. 2012), 267–270.
[178] Amy Wesolowski, Taimur Qureshi, Maciej F. Boni, Pål Roe Sundsøy, Michael A. Johansson, Syed Basit Rasheed, Kenth Engø-Monsen, Caroline O. Buckee, and Burton H. Singer. 2015. Impact of human mobility on the emergence of dengue epidemics in Pakistan. *Proc. Natl. Acad. Sci. U.S.A.* 112, 38 (2015), 11887–11892.
[179] Guojun Wu, Yanhua Li, Jie Bao, Yu Zheng, Jieping Ye, and Jun Luo. 2018. Human-centric urban transit evaluation and planning. In *ICDM*. 547–556.
[180] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. In *IJCAI*. 3083–3090.
[181] Yanbo Wu, Hong Shen, and Quan Z. Sheng. 2015. A cloud-friendly RFID trajectory clustering algorithm in uncertain environments. *IEEE Trans. Parallel Distrib. Syst.* 26, 8 (2015), 2075–2088.
[182] Dong Xie, Feifei Li, and Jeff M. Phillips. 2017. Distributed trajectory similarity search. *Proc. VLDB* 10, 11 (2017), 1478–1489.
[183] Xike Xie, Benjin Mei, Jinchuan Chen, Xiaoyong Du, and Christian S. Jensen. 2016. Elite: An elastic infrastructure for big spatiotemporal trajectories. *VLDB J.* 25, 4 (2016), 1–21.

[184] Xiaochun Yang, Bin Wang, Kai Yang, Chengfei Liu, and Baihua Zheng. 2018. A novel representation and compression for queries on trajectories in road networks. *IEEE Trans. Knowl. Data Eng.* 30, 4 (2018), 613–629.
[185] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *ICDE*. 1358–1369.
[186] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2017. Trajectory clustering via deep representation learning. In *IJCNN*. 3880–3887.
[187] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *ICDE*. 201–208.
[188] Simin You, Jianting Zhang, and Le Gruenwald. 2015. Large-scale spatial join query processing in Cloud. In *ICDE*. 34–41.
[189] Jia Yu, Jinxuan Wu, and Sarwat Mohamed. 2015. Geospark: A cluster computing framework for processing large-scale spatial data. In *SIGSPATIAL*. 4–7.
[190] Haitao Yuan and Guoliang Li. 2019. Distributed in-memory trajectory similarity search and join on road network. In *ICDE*. 1262–1273.
[191] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective travel time estimation: When historical trajectories over road networks matter. In *SIGMOD*. 2135–2149.
[192] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: Driving directions based on taxi trajectories. In *GIS*. 99–108.
[193] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas La Porta. 2014. Splitter: Mining fine-grained sequential patterns in semantic trajectories. *Proc. VLDB* 7, 3 (2014), 769–780.
[194] Chao Zhang, Keyang Zhang, Quan Yuan, Luming Zhang, Tim Hanratty, and Jiawei Han. 2016. Gmove: Group-level mobility modeling using geo-tagged social media. In *KDD*. 1305–1314.
[195] Dongxiang Zhang, Chee-Yong Chan, and Kian-Lee Tan. 2014. Processing spatial keyword query as a top-k aggregation query. In *SIGIR*. 355–364.
[196] Dongxiang Zhang, Mengting Ding, Dingyu Yang, Yi Liu, Ju Fan, and Heng Tao Shen. 2018. Trajectory simplification: An experimental study and quality analysis. *Proc. VLDB* 11, 9 (2018), 934–946.
[197] Hanyuan Zhang, Hao Wu, Weiwei Sun, and Baihua Zheng. 2018. DEEPTRAVEL: A neural network based travel time estimation model with auxiliary supervision. In *IJCAI*. 3655–3661.
[198] Jun Zhang, Dayong Shen, Lai Tu, Fan Zhang, Chengzhong Xu, Yi Wang, Chen Tian, Xiangyang Li, Benxiong Huang, and Zhengxi Li. 2017. A real-time passenger flow estimation and prediction method for urban bus transit systems. *IEEE Trans. Intell. Transport. Syst.* 18, 11 (2017), 3168–3178.
[199] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. 2018. Trajectory-driven influential billboard placement. In *KDD*. 2748–2757.
[200] Yipeng Zhang, Yuchen Li, Zhifeng Bao, Songsong Mo, and Ping Zhang. 2019. Optimizing impression counts for outdoor advertising. In *KDD*. 1205–1215.
[201] Bolong Zheng, Nicholas Jing Yuan, Kai Zheng, Xing Xie, Shazia Sadiq, and Xiaofang Zhou. 2015. Approximate keyword search in semantic trajectory database. In *ICDE*. 975–986.
[202] Kai Zheng, Shuo Shang, Nicholas Jing Yuan, and Yi Yang. 2013. Towards efficient search for activity trajectories. In *ICDE*. 230–241.
[203] Kai Zheng, Yu Zheng, Nicholas J. Yuan, S. Shang, and Xiaofang Zhou. 2014. Online discovery of gathering patterns over trajectories. *IEEE Trans. Knowl. Data Eng.* 26, 8 (2014), 1974–1988.
[204] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 1–41.
[205] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation modes from raw GPS data. In *WWW*. 247–256.
[206] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*. 791–800.
[207] Yu Zheng and Xiaofang Zhou. 2011. *Computing with Spatial Trajectories*. Springer.
[208] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. 2012. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *CVPR*. 2871–2878.
[209] Fan Zhou, Xiaoli Yue, Goce Trajcevski, Ting Zhong, and Kunpeng Zhang. 2019. Context-aware trajectory embedding and human mobility inference. In *WWW*. 3469–3475.
[210] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *Comput. Surv.* 38, 2 (2006), 1–56.

Received March 2020; revised November 2020; accepted November 2020