# Trajectory Data Mining: An Overview

YU ZHENG
Microsoft Research

The advances in location-acquisition and mobile computing techniques have generated massive spatial trajectory data, which represent the mobility of a diversity of moving objects, such as people, vehicles and animals. Many techniques have been proposed for processing, managing and mining trajectory data in the past decade, fostering a broad range of applications. In this article, we conduct a systematic survey on the major research into *trajectory data mining*, providing a panorama of the field as well as the scope of its research topics. Following a roadmap from the derivation of trajectory data, to trajectory data preprocessing, to trajectory data management, and to a variety of mining tasks (such as trajectory pattern mining, outlier detection, and trajectory classification), the survey explores the connections, correlations and differences among these existing techniques. This survey also introduces the methods that transform trajectories into other data formats, such as graphs, matrices, and tensors, to which more data mining and machine learning techniques can be applied. Finally, some public trajectory datasets are presented. This survey can help shape the field of *trajectory data mining*, providing a quick understanding of this field to the community.

H.2.8 [**Database Management**]: Database Applications - *data mining*, *spatial databases and GIS*; I.2.6 [**Artificial Intelligence**]: Learning - *Knowledge acquisition*

General Terms: Algorithms, Measurement, Experimentation

Additional Key Words and Phrases: Spatio-temporal data mining, Trajectory data mining, Trajectory compression, Trajectory Indexing and retrieval, Trajectory pattern mining, Trajectory outlier detection, Trajectory uncertainty, Trajectory classification, Urban computing.

## 1. INTRODUCTION

A *spatial trajectory* is a trace generated by a moving object in geographical spaces, usually represented by a series of chronologically ordered points, e. g. $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, where each point consists of a geospatial coordinate set and a timestamp such as $p = (x, y, t)$.

The advance in location acquisition technologies has generated a myriad of spatial trajectories representing the mobility of various moving objects, such as people, vehicles, and animals. Such trajectories offer us unprecedented information to understand moving objects and locations, fostering a broad range of applications in location-based social networks [140], intelligent transportation systems, and urban computing [142]. The prevalence of these applications in turn calls for systematic research on new computing technologies for discovering knowledge from trajectory data. Under the circumstance, *Trajectory Data Mining* has become an increasingly important research theme, attracting the attention from numerous areas, including computer science, sociology, and geography.

Authors' addresses: Y. Zheng, Microsoft Research, Building 2, No. 5 Danling Street, Haidian District, Beijing 100080, China; email: yuzheng@microsoft.com.

http://dx.doi.org/10.1145/2743025

Intensive and extensive individual research has been done in the field of *trajectory data*

*mining*. However, we are lack of a systematic review that can well shape the field and position existing research. Facing a huge volume of publications, the community is still not very clear about the connections, correlations and difference among these existing techniques. To this end, we conduct a comprehensive survey that thoroughly explores the field of *trajectory data mining*, according to the paradigm shown in Figure1:

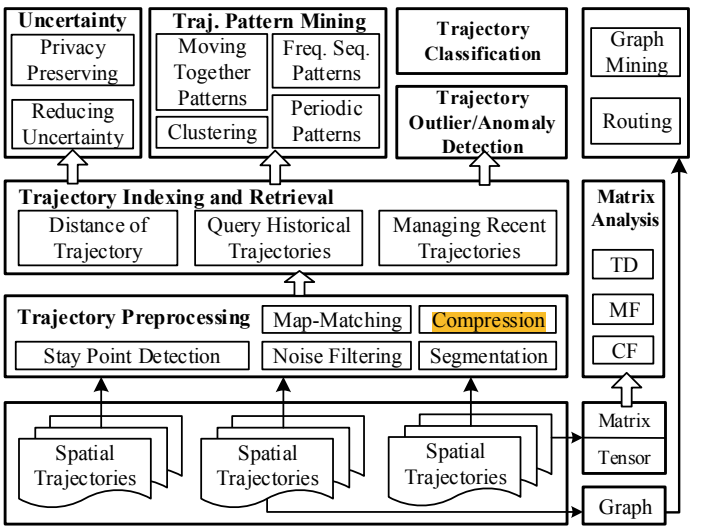



**Figure 1 Paradigm of trajectory data mining**

Firstly, in Section 2, we classify the sources generating trajectory data into four groups, listing a few key applications that trajectory data can enable in each group.

Secondly, before using trajectory data, we need to deal with a number of issues, such as noise filtering, segmentation, and map-matching. This stage is called *trajectory pre-processing*, which is a fundamental step of many trajectory data mining tasks. The goal of *noise filtering* is to remove from a trajectory some noise points that may be caused by the poor signal of location positioning systems (e.g. when traveling in a city canyon). *Trajectory compression* is to compress the size of a trajectory (for the purpose of reducing overhead in communication, processing, and data storage) while maintaining the utility of the trajectory. *A stay point detection* algorithm identifies the location where a moving object has stayed for a while within a certain distance threshold. A stay point could stand for a restaurant or a shopping mall that a user has been to, carrying more semantic meanings than other points in a trajectory. *Trajectory segmentation* divides a trajectory into fragments by time interval, or spatial shape, or semantic meanings, for a further process like clustering and classification. *Map-Matching* aims to project each point of a trajectory onto a corresponding road segment where the point was truly generated. We detail *trajectory pre-processing* in Section 3.

Thirdly, many online applications require instantly mining of trajectory data (e.g. detecting traffic anomalies), calling for effective data management algorithms that can quickly retrieve particular trajectories satisfying certain criteria (such as spatio-temporal constraints) from a big trajectory corpus. There are usually two major types of queries: the nearest neighbors and range queries. The former is also associated with a distance metric, e.g. the distance between two trajectories. Additionally, there are two types (historical and

recent) of trajectories, which need different managing methods. We will introduce *trajectory indexing and retrieval* in Section 4.

Fourthly, based on the first two steps, we can then conduct mining tasks, like trajectory pattern mining, trajectory uncertainty, outlier detection, and classification.

- *Trajectory Uncertainty*: Objects move continuously while their locations can only be updated at discrete times, leaving the location of a moving object between two updates uncertain. To enhance the utility of trajectories, a series of research tried to model and reduce the uncertainty of trajectories. On the contrary, a branch of research aims to protect a user's privacy when the user discloses her trajectories. We review uncertainty of trajectory in Section 5.
- *Trajectory Pattern Mining*: The huge volume of spatial trajectories enables opportunities for analyzing the mobility patterns of moving objects, which can be represented by an individual trajectory containing a certain pattern or a group of trajectories sharing similar patterns. In Section 6, we survey the literature that is concerned with four categories of patterns: moving together patterns, trajectory clustering, periodic patterns, and frequent sequential patterns.
- *Trajectory Classification:* Using supervised learning approaches, we can classify trajectories or segments of a trajectory into some categories, which can be activities (like hiking and dining) or different transportation modes, such as walking and driving. We show examples of trajectory classification in Section 7.
- *Trajectory Outlier Detection:* Different from trajectory patterns that frequently occur in trajectory data, trajectory outliers (a.k.a. anomalies) can be items (a trajectory or a segment of trajectory) that is significantly different from other items in terms of some similarity metric. It can also be events or observations (represented by a collection of trajectories) that do not conform to an expected pattern (e.g. a traffic congestion caused by a car accident). Section 8 introduces outlier/anomaly detection from trajectory data.

Finally, besides studying trajectories in its original form, we can transform trajectories into other formats, such as graph, matrix and tensor (see the right part of Figure 1). The new representations of trajectories expand and diversify the approaches for trajectory data mining, leveraging existing mining techniques, e.g. graph mining, collaborative filtering (CF), matrix factorization (MF), and tensor decomposition (TD). In Section 9, we present representative examples of the transformation.

The contribution of this paper lies in four folds. *First*, the paper presents a framework for trajectory data mining, defining the scope and roadmap for this field. The framework provides a panorama with which people can quickly understand and step into this field. *Second*, individual research works are well positioned, categorized and connected in each layer of this framework. Professionals can easily locate the methods they need to solve a problem, or find the unsolved problems. *Third*, this paper proposes a vision to transfer trajectories into other formats, to which a diversity of existing mining techniques can be applied. This expands the original scope of trajectory data mining, advancing the methodologies and applications of this field. *Fourth*, we collect a list of sources from which people can obtain various public trajectory datasets for research. We also introduce the conferences and journals that are concerned with the research on trajectory data.

## 2. TRAJECTORY DATA

In this section, we classify the derivation of trajectories into four major categories, briefly introducing a few application scenarios in each category. Trajectory data representing human mobility can help build a better social network [6][140][141] and travel recommendation [152][154][156].

1) Mobility of people: People have been recording their real-world movements in the form of spatial trajectories, passively and actively, for a long time.

- Active recording: Travelers log their travel routes with GPS trajectories for the purpose of memorizing a journey and sharing experiences with friends. Bicyclers and joggers record their trails for sports analysis. In Flickr, a series of geo-tagged photos can formulate a spatial trajectory as each photo has a location tag and a timestamp corresponding to where and when the photo was taken. Likewise, the "check-ins" of a user in a location-based social network can be regarded as a trajectory, when sorted chronologically.
- Passive recording: A user carrying a mobile phone unintentionally generates many spatial trajectories represented by a sequence of cell tower IDs with corresponding transition times. Additionally, transaction records of a credit card also indicate the spatial trajectory of the cardholder, as each transaction contains a timestamp and a merchant ID denoting the location where the transaction occurred.

2) Mobility of transportation vehicles: A large number of GPS-equipped vehicles (such as taxis, buses, vessels, and aircrafts) have appeared in our daily life. For instance, many taxis in major cities have been equipped with a GPS sensor, which enables them to report a time-stamped location with a certain frequency. Such reports formulate a large amount of spatial trajectories that can be used for resource allocation [127][129], traffic analysis [104][125], and improving transportation networks [151].

3) Mobility of animals: Biologists have been collecting the moving trajectories of animals like tigers and birds, for the purposes of studying animals' migratory traces, behavior and living situation [51][57].

4) Mobility of natural phenomena: Meteorologists, environmentalists, climatologists and oceanographers are busy collecting the trajectories of some natural phenomena, such as hurricanes, tornados, and ocean currents. These trajectories capture the change of the environment and climate, helping scientists deal with nature disasters and protect the natural environment we live in.

## 3. TRAJECTORY DATA PREPROCESSING

This section introduces four folds of basic techniques that we need to process a trajectory before starting a mining task, consisting of noise filtering, stay point detection, trajectory compression, and trajectory segmentation.

### 3.1 Noise Filtering

Spatial trajectories are never perfectly accurate, due to sensor noise and other factors, such as receiving poor positioning signals in urban canyons. Sometimes, the error is acceptable (e.g. a few GPS points of a vehicle fall out of the road the vehicle was actually driven), which can be fixed by map-matching algorithms (introduced in Section 3.5). In other situations, as shown in Figure 2, the error of a noise point like $p_5$ is too big (e.g. several hundred meters away from its true location) to derive useful information, such as travel speed. So, we need to filter such noise points from trajectories before starting a mining task. Though this problem has not been completely solved, existing methods fall in three major categories:

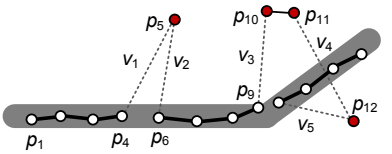



**Figure 2. Noise points in a trajectory**

***Mean (or Median) filter***: For a measured point $\boldsymbol{z}_i$, the estimate of the (unknown) true value is the mean (or median) of $\boldsymbol{z}_i$ and its $n-1$ predecessors in time. The mean (median) filter can be thought of as a sliding window covering $n$ temporally adjacent values of $\boldsymbol{z}_i$. In the example shown in Figure 2, $p_5.\boldsymbol{z} = \sum_{i=1}^{5} p_i.\boldsymbol{z}\,/5$, if we use a mean filter with a sliding window size of 5. Median filter is more robust than the mean filter when handling extreme errors. The mean (median) filters are practical for handling individual noise points like $p_5$ in a trajectory with a dense representation. However, when dealing with multiple consecutive noise points, e.g. $p_{10}, p_{11}$ and $p_{12}$, a larger size of sliding window is needed. This results in a bigger error between the calculated mean (or median) value and a point's true position. When the sampling rate of trajectory is very low (i.e. the distance between two consecutive points could be longer than several hundred meters), the mean and median filters are not good choices anymore.

***Kalman and Particle filters***: The trajectory estimated from the Kalman filter is a tradeoff between the measurements and a motion model. Besides giving estimates that obey the laws of physics, the Kalman filter gives principled estimates of higher order motion states like speed. While the Kalman filter gains efficiency by assuming linear models plus Gaussian noise, the particle filter relaxes these assumptions for a more general, but less efficient, algorithm. A tutorial-like introduction to using the Kalman and Particle filters to fix noisy trajectory points can be found in [53].

The initialization step of the particle filtering is to generate $P$ particles $\boldsymbol{x}_i^{(j)}, j = 1,2,..P$ from the initial distribution. For example, these particles would have zero velocity and be clustered around the initial location measurement with a Gaussian distribution. The second step is "importance sampling," which uses the dynamic model $P(\boldsymbol{x}_i|\boldsymbol{x}_{i-1})$ to probabilistically simulate how the particles change over one time step. The third step computes "importance weights" for all the particles using the measurement model $\omega_i^{(j)} = P(z_i|\hat{\boldsymbol{x}}_i^{(j)})$. Larger importance weights correspond to particles that are better supported by the measurement. The important weights are then normalized so they sum to one. The last step in the loop is the "selection step" when a new set of $P$ particles $\boldsymbol{x}_i^{(j)}$ is selected from the $\hat{\boldsymbol{x}}_i^{(j)}$ proportional to the normalized importance weights $\omega_i^{(j)}$. Finally, we can compute a weight sum by $\hat{\boldsymbol{x}}_i = \sum_{i=1}^{P} \omega_i^{(j)} \hat{\boldsymbol{x}}_i^{(j)}$.

The Kalman and particle filters, model both the measurement noise and the dynamics of the trajectory. However, they depend on the measurement of an initial location. If the first point in a trajectory is noisy, the effectiveness of the two filters drops significantly.

***Heuristics-Based Outlier Detection***: While the above mentioned filters replace a noise measurement in a trajectory with an estimated value, the third category of methods removes noise points directly from a trajectory by using outlier detection algorithms. The noise filtering method, which has been used in T-Drive [123][124][125] and GeoLife [145][153] projects, first calculates the travel speed of each point in a trajectory based on the time interval and distance between a point and its successor (we call this a segment). The segments, such as $p_4 \rightarrow p_5$, $p_5 \rightarrow p_6$, and $p_9 \rightarrow p_{10}$ (illustrated by the dotted lines in Figure 2), with a speed larger than a threshold, e.g. 300km/h, are cut off. Given that the number of

noise points is much smaller than common points, the separated points like $p_5$ and $p_{10}$ can be regarded as outliers. Some distance-based outlier detection can easily find the number of $p_5$'s neighbors within a distance $d$ is smaller than $p$ proportion of the points in the entire trajectory. Likewise, $p_{10}, p_{11}$ and $p_{12}$ can be filtered. While such algorithms can handle the initial error in a trajectory and data sparsity problems, setting the threshold $d$ and $p$ is still based on heuristics.

### 3.2 Stay Point Detection

Spatial points are not equally important in a trajectory. Some points denote locations where people have stayed for a while, such as shopping malls and tourist attractions, or gas stations where a vehicle was refueled. We call this kind of points "*Stay Points*". As shown in Figure 3 A), there are two types of stay points occurring in a trajectory. One is a single point location, e.g. *Stay Point 1*, where a user remains stationary for a while. This situation is very rare, because a user's positioning device usually generates different readings even in the same location. The second type, like *Stay Points 2* shown in Figure 3 A), is more generally observed in trajectories, representing the places where people move around (e.g. as depicted in Figure 3 B) and C)) or remain stationary but with positioning readings shifting around.

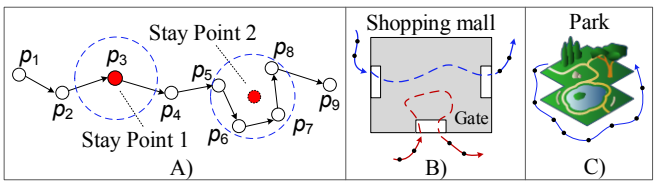



**Figure 3. Stay points in a trajectory**

With such stay points, we can turn a trajectory from a series of time-stamped spatial points $\boldsymbol{P}$ into a sequence of meaningful places $\boldsymbol{S}$,

$$\boldsymbol{P} = p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n, \Rightarrow \boldsymbol{S} = s_1 \xrightarrow{\Delta t_1} s_2 \xrightarrow{\Delta t_2}, \ldots, \xrightarrow{\Delta t_{n-1}} s_n,$$

therefore facilitating a diversity of applications, such as travel recommendations [152] [154], destination prediction [116], taxi recommendation [127][129], and gas consumption estimation[132][133]. On the other hand, in some applications, e.g. estimating the travel time of a path [104] and driving direction suggestion [125], such stay points should be removed from a trajectory during the preprocessing.

Li and Zheng et al. [54] first proposed the stay point detection algorithm. This algorithm first checks if the distance between an anchor point (e.g. $p_5$) and its successors in a trajectory larger than a given threshold (e.g. 100 meter). It then measures the time span between the anchor point and the last successor (i.e. $p_8$) that is within the distance threshold. If the time span is larger than a given threshold, a stay point (characterized by $p_5, p_6$, $p_7$, and $p_8$) is detected; the algorithm starts detection the next stay point from $p_9$. Yuan and Zheng et al. [127][130] improved this stay point detection algorithm based on the idea of density clustering. After finding $p_5$ to $p_8$ is a candidate stay point (using $p_5$ as an anchor point), their algorithm further checks the successor points from $p_6$. For instance, if the distance from $p_9$ to $p_6$ is smaller than the threshold, $p_9$ will be added into the stay point.

### 3.3 Trajectory Compression

Basically, we can record a time-stamped geographical coordinate every second for a moving object. But, this costs a lot of battery power and the overhead for communication, computing and data storage. In addition, many applications do not really need such a precision of location. To address this issue, two categories of trajectory compression strategies (based on the shape of a trajectory) have been proposed, aiming to reduce the size of a trajectory while not to compromise much precision in its new data representation [53]. One is the offline compression (a.k.a. batch mode), which reduces the size of trajectory after the trajectory has been fully generated. The other is online compression, compressing a trajectory instantly as an object travels.

***Distance Metric***: Besides the two strategies, there are two distance metrics to measure the error of a compression: *Perpendicular Euclidean Distance* and *Time Synchronized Euclidean Distance*. As illustrated in Figure 4, supposing we compress a trajectory with 12 points into a representation of three points (i.e. $p_1, p_7$, and $p_{12}$), the two distance metrics are the summation of the lengths of the segments connecting $p_i$ and $p_i'$, in Figure 4 A) and B), respectively. The latter distance assumes a constant speed traveling between $p_1$ and $p_7$, calculating the projection of each original point on $\overline{p_1 p_7}$ by time intervals.

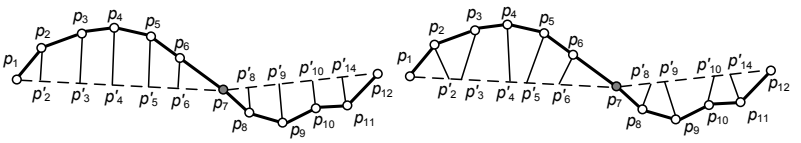



**Figure 4. Distance metric measuring the compression error**

***Offline Compression***: Given a trajectory that consists of a full series of time-stamped points, a batched compression algorithm aims to generate an approximated trajectory by discarding some points with a negligible error from the original trajectory. This is similar to the line simplification problem, which has been studied in the computer graphics and cartography research communities [68].

A well-known algorithm, called Douglas-Peucker [28], is used to approximate the original trajectory. As demonstrated in Figure 5 A), the idea of Douglas-Peucker is to replace the original trajectory by an approximate line segment, e.g. $\overline{p_1 p_{12}}$. If the replacement does not meet the specified error requirement (Perpendicular Euclidean Distance is used in this example), it recursively partitions the original problem into two sub-problems by selecting the point contributing the biggest error as the splitting point (e.g. $p_4$). This process continues until the error between the approximation and the original trajectory is below a specified error. The complexity of the original Douglas-Peucker algorithm is $O(N^2)$, where $N$ is the number of points in a trajectory. Its improvement achieves $O(NlogN)$ [39]. To ensure that the approximated trajectory is optimal, Bellman's algorithm [7] employs a dynamic programming technique with a complexity of $O(N^3)$.

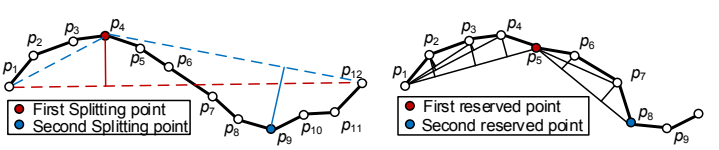



**Figure 5. Illustration of Douglas-Peucker algorithm**

***Online Data Reduction***: As many applications require to transmit trajectory data in a timely fashion, a series of online trajectory compression techniques have been proposed to determine whether a newly acquired spatial point should be retained in a trajectory. There are two major categories of online compression methods. One is the window-based algorithms, such as the *Sliding Window* algorithm [46] and *Open Window* algorithm [67]. The other is based on speed and direction of a moving object.

The idea of the *Sliding Window* algorithm is to fit the spatial points in a growing sliding window with a valid line segment and continue to grow the sliding window until the approximation error exceeds some error bound. As illustrated in Fig 5 B), $p_5$ will be first reserved as the error for $p_3$ exceeds the threshold. Then, the algorithm starts from $p_5$ and reserve $p_8$. Other points are negligible. Different from the *Sliding Window* algorithm, the *Open Window* algorithm [67] applies the heuristic of the Douglas-Peucker algorithm to choose the point with the maximum error in the window (e.g. $p_3$ in Figure 5 B) to approximate the trajectory segment. This point is then used as a new anchor point to approximate its successors.

Another category of algorithms consider speed and directions as key factors when doing online trajectory compression. For instance, Potamias et al. [84] use a safe area, derived from the last two locations and a given threshold, to determine whether a newly acquired point contains important information. If the new data point is located within the safe area, then this location point is considered as redundant and thus can be discarded; otherwise, it is included in the approximated trajectory.

***Compression with Semantic Meaning***: A series of research [87][17] aims to keep the semantic meanings of a trajectory, when compressing the trajectory. For instance, in a location-based social network [140], some special points where a user stayed, took photos, or changed direction greatly, would be more significant than other points in presenting semantic meanings of a trajectory. Chen et al. [17] proposed a trajectory simplification algorithm (TS), which considers both the shape skeleton and the aforementioned special points. TS first divides a trajectory into walking and non-walking segments using a trajectory segmentation algorithm [147] (see Section 3.4). A point is weighted by its heading change degree and the distance to its neighbors.

Another branch of research [45][90] considers trajectory compression with the constraints of transportation networks. For example, we can reduce the redundant points on the same road segment. We can even discard all the newly acquired points after an anchor point, as long as the moving object is traveling on the shortest path from the anchor point to its current location. This branch of work usually needs the support of map-matching algorithms (refer to Section 3.5). In 2014, PRESS [90] was proposed to separate the spatial representation of a trajectory from its temporal representation. PRESS consists of a hybrid spatial compression algorithm and an error bounded temporal compression algorithm, compressing the spatial and temporal information of trajectories respectively. The spatial compression combines frequent sequential pattern mining techniques with Huffman Coding to reduce the size of a trajectory; i.e. a frequently traveled path can be represented by a shorter code, therefore saving storages.

## 3.4 Trajectory Segmentation

In many scenarios, such as trajectories clustering and classification, we need to divide a trajectory into segments for a further process. The segmentation does not only reduce the computational complexity but also enable us to mine richer knowledge, such as sub-trajectory patterns, beyond what we can learn from an entire trajectory. In general, there

are three types of segmentation methods.

The first category is based on *time interval*. For example, as illustrated in Figure 6 A), if the time interval between two consecutive sampling points is larger than a given threshold, a trajectory is divided into two parts at the two points, i.e. $p_1 \rightarrow p_2$ and $p_3 \rightarrow \cdots \rightarrow p_9$. Sometimes, we can divid a trajectory into segments of the same time length.

The second category of methods is based on the *shape of a trajectory*. For example, as demonstrated in Figure 6 B), we can partition a trajectory by the turning points with heading direction changing over a threshold. Alternative, we can employ the line simplification algorithms, such as Douglas-Peucker algorithm, to identify the key points maintaining a trajectory's shape, as depicted in Figure 6 C). The trajectory is then partitioned into segments by these key points. Similarly, Lee et al [51] proposed to partition a trajectory by using the concept of Minimal Description Language (MDL), which is comprised of two components: $L(H)$ and $L(D|H)$. $L(H)$ is the length, in bits, of the description of the hypothesis $H$; and $L(D|H)$ is the length, in bits, of the description of the data when encoded with the help of the hypothesis. The best hypothesis $H$ to explain $D$ is the one that minimizes the sum of $L(H)$ and $L(D|H)$. More specifically, they use $L(H)$ to denote the total length of partitioned segments (like $\overline{p_1p_7}$ and $\overline{p_1p_9}$), while let $L(D|H)$ to represent the total (perpendicular and angle) distance between the original trajectory and the new partitioned segments. Using an approximation algorithm, they find a list of characteristic points that minimize $L(H) + L(D|H)$ from a trajectory. The trajectory is partitioned into segments by these characteristic points.

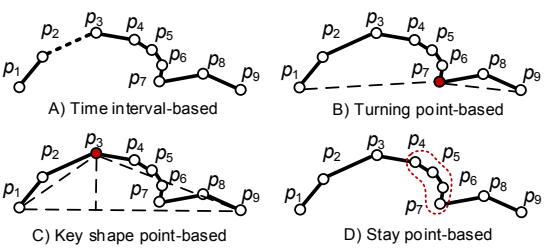



**Figure 6. Methods of trajectory segmentation**

The third category of methods is based on the *semantic meanings* of points in a trajectory. As illustrated in Figure 6 D), a trajectory can be divided into segments, i.e. $p_1 \rightarrow p_2 \rightarrow p_3$ and $p_8 \rightarrow p_9$, based on the stay points it contains. Whether we should keep the stay points in the divided results depends on applications. For example, in a task of travel speed estimation, we should remove the stay points (from a taxi's trajectory) where a taxi was parked to wait for passengers [125]. On the contrary, to estimate the similarity between two users [52], we can only focus on the sequences of stay points, while skipping other raw trajectory points between two consecutive stay points.

Another semantic meaning-based trajectory segmentation is to divide a trajectory into segments of different transportation modes, such as driving, taking a bus, and walking. For example, Zheng et al. [144][146][149] proposed a walk-based segmentation method. The key insight is that people have to walk through the transition between two different transportation modes. Consequently, we can first distinguish *Walk Points* from *non-Walk Points* in a trajectory, based on a point's speed $(p.v)$ and acceleration $(p.a)$. The trajectory can then be divided into alternate *Walk Segments* and *non-Walk Segments*, as illustrated in Figure 7 A). In reality, however, as shown in Figure 7 B), a few points from *non-Walk Segments* may be detected as possible *Walk Points*, e.g., when a bus moves slowly in traffic

congestion. On the other hand, due to the locative error, a few points from *Walk Segments* might exceed the upper bound of travel speed ($v_t$), therefore being recognized as *non-Walk Points*. To address this issue, a segment is merged into its backward segment, if the distance or time span of the segment is less than a threshold. After that, a segment is regarded as a *Certain Segment* if its length exceeds a threshold, as presented in Figure 7 C). Otherwise, it is deemed as an *Uncertain Segment*. As common users do not frequently change their transportation modes within a short distance, *Uncertain Segments* are merged into one *non-Walk Segment* if the number of consecutive *Uncertain Segments* exceeds a certain threshold (3 in this example). Later, features are extracted from each segment to determine its exact mode.

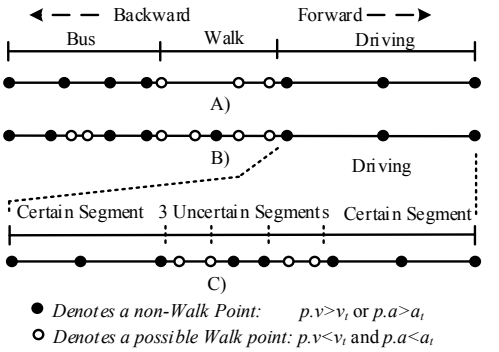



**Figure 7. Change point-based segmentation method**

## 3.5 Map Matching

Map matching is a process to convert a sequence of raw latitude/longitude coordinates to a sequence of road segments. Knowledge of which road a vehicle was/is on is important for assessing traffic flow, guiding the vehicle's navigation, predicting where the vehicle is going, and detecting the most frequent travel path between an origin and a destination, etc. Map matching is not an easy problem, given parallel roads, overpasses, and spurs [49]. There are two approaches to classify map matching methods, based on the additional information used, or the range of sampling points considered in a trajectory.

***According to the additional information used***, map matching algorithms can be categorized into four groups: *geometric* [36], *topological* [22][118], *probabilistic* [75] [83] [86] and other *advanced* techniques [63][73][126]. *Geometric* map matching algorithms consider the shape of individual links in a road network, for example, matching a GPS point to the nearest road. *Topological* algorithms pay attention to the connectivity of a road network. Representative algorithms are those that use the Fréchet distance to measure the fit between a GPS sequence and candidate road sequence [9]. To deal with noisy and low-sampling rate trajectories, *probabilistic* algorithms [75][83][86] make explicit provisions for GPS noise and consider multiple possible paths through the road network to find the best one. More *advanced* map matching algorithms have emerged recently that embrace both the topology of the road network and the noise in trajectory data, exemplified by [63] [73][126]. These algorithms find a sequence of road segments that simultaneously come close to the noisy trajectory data and form a reasonable route through the road network.

***According to the range of sampling points*** considered, map matching algorithms can be classified into two categories: *local/incremental* and *global* methods. The *local/ incremental* algorithms [26][16] follow a greedy strategy of sequentially extending the

solution from an already matched portion. These methods try to find a local optimal point based on the distance and orientation similarity. *Local/incremental* methods run very efficiently, often adopted in online applications. However, when the sampling rate of a trajectory is low, the matching accuracy degrades. Instead, *Global* algorithms [4][9] aim to match an entire trajectory with a road network, e.g. considering the predecessors and successors of a point. *Global* algorithms are more accurate, but less efficient, than *local* methods, usually applied to offline tasks (e.g. mining frequent trajectory patterns), where entire trajectories have already been generated.

Advanced algorithms [63][73] [126] embrace local and global information (or geometric, topological, and probability) to deal with the mapping of a low-sampling-rate trajectory. As shown in Figure 8 A), the algorithm proposed in [54] first finds the local candidate road segments that is within a circle distance to each point in a trajectory. For instance, road segments $e_i^1$,$e_i^2$ and $e_i^3$ are within the circle distance to $p_i$, and $c_i^1$,$c_i^2$ and $c_i^3$ are the candidate points on these road segments. The distance between $p_i$ and a candidate point $dist(c_i^j,\ p_i)$ indicates the probability $N(c_i^j)$ that $p_i$ can be matched to the candidate point. This *probability* can be regarded as the *local* and *geometric* information, which is modeled by a normal distribution:

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{dist(c_i^j, p_i)^2}{2\sigma^2}}.$$

The algorithm also considers the transition probability between the candidate points of each two consecutive trajectory points. For example, as depicted in Figure 8 B), $c_i^2$ is more likely to be the true match of $p_i$, considering $p_{i-1}$ and $p_{i+1}$. The transition probability between two candidate points is denoted by the ratio between their Euclidian distance and the road network distance. The transition is actually based on the *topologic* information of a road network. Finally, as shown in Figure 8 C), combining the local and transition probabilities, the map matching algorithm finds a path (on a candidate graph) that maximizes the *global* probability of matching. The idea is similar to Hidden Markov Model where emission and transition probabilities are considered to find the most possible sequence of status given a sequence of observations [73].

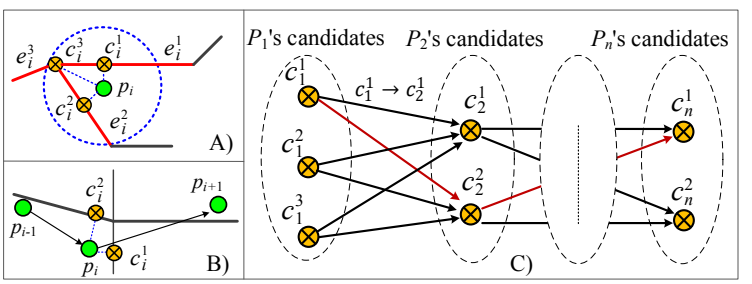



**Figure 8. An advanced map-matching algorithm**

## 4. TRAJECTORY DATA MANAGEMENT

Mining massive trajectories is very time consuming, as we need to access different samples of the trajectories or different parts of a trajectory many times. This calls for effective data management techniques that can quickly retrieve the trajectories (or parts of a trajectory) needed. Different from moving object databases that are concerned with the current location of a moving object, the trajectory data management introduced in this section deals

with the traveling history of a moving object. A more comprehensive survey on trajectory data manage can be found in [27].

### 4.1 Trajectory Indexing and Retrieval

There are two major types of queries: K-Nearest Neighbor (KNN) queries and Range queries, as depicted in Figure 9.

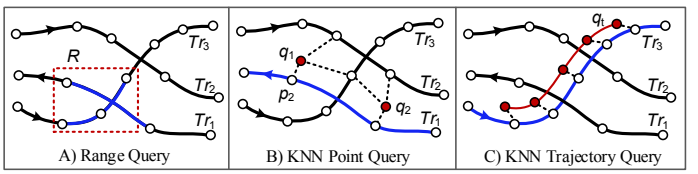



**Figure 9. Two categories of queries for trajectory data**

***Range queries*** retrieve the trajectories falling into (or intersecting) a spatial (or spatio-temporal) range. For example, as shown in Figure 9 A), a range query can help us retrieve the trajectories of vehicles passing a given rectangular region *R* between 2pm-4pm in the past month. The retrieved trajectories (or segments) can then be used to derive features, such as the travel speed and traffic flow, for data mining tasks like classification and prediction. There are three approaches to answering such kind of spatio-temporal range queries.

The *first* approach regards the time as the third dimension besides the 2D geographical space, building a 3D-Rtree based on trajectories, as depicted in Fig 10 A). A spaito-temporal range query is then formulated as a 3D query box. So, answering such a query means finding the nodes on a 3D-Rtree within the 3D query box. The 3D-Rtree works well for indexing trajectories generated in the near recent (e.g. in the past few hours). When the time span of the trajectories to be indexed last for a long period (i.e. more segments of newly generated trajectories will be inserted into a 3D-Rtree index), however, the overlap among 3D boxes bounding segments of different trajectories occurs more often. This results in a frequent update of indexing structure and a significant increase of node accesses when retrieving a trajectory. Though ST-R-tree and TB-tree [82] have been proposed to address this issue, the overlap among different 3D boxes still keeps on increasing as time goes by.

The *second* approach divides a time period into multiple time intervals, building an individual spatial index like R-tree for the trajectories generated in each interval. The part of indexing structure that does not change over time is shared by two time slots. Representative indexing structures are multiple version R-tree, such as Rt-Tree [110], HR-Tree [92] and H+R-Tree [93], as illustrated in Figure 10 B). Given a spatio-temporal range query, such an index first finds the time slots falling in the temporal range and then retrieve the trajectories intersecting the spatial query range from each spatial index of these time slots.

The third approach partition a geographical space into grids and then build a temporal index for the trajectories falling in each grid. As shown in Figure 10 C), CSE-tree [103] divides a trajectory into several segments by the grids. Each segment falling in a grid is represented by a 2D point whose coordinates are the starting time and ending time of the segment. These points are then indexed by a hybrid B+tree. When retrieving trajectories satisfying a spatio-temporal query, CSE-tree first finds the grids intersecting the spatial range of the query and then searches the hybrid B+tree of these grids for the segments of

trajectories falling in the temporal range of the query. Finally, CSE-tree merges the IDs of trajectory segments (and their starting and ending times) retrieved from different grids.

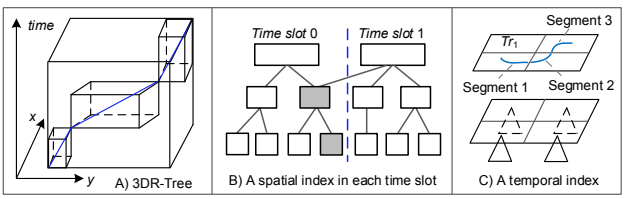



**Figure 10. Three approaches answering range queries**

***KNN queries*** retrieve the top *K* trajectories with the minimum aggregate distance to a few points (entitled the KNN point query [21][94][95]) or a specific trajectory (entitled the KNN trajectory query [117][3]).

As depicted in Figure 9 B), an example of the KNN point query is to retrieve the trajectories of vehicles that are close to two given restaurants (e.g. $q_1$ and $q_2$). Sometimes, the order between the query points is also considered [21], e.g. finding the top-$k$ nearest trajectories first passing $q_1$ and then $q_2$. Without the order, $Tr_1$ is the nearest trajectory to the two points. However, $Tr_2$ becomes the nearest after considering the order. The KNN point queries concern more about whether a trajectory provides a good connection to query locations rather than whether the trajectory is similar to the query in shape. Additionally, the number of query points is usually very small and can be far away from each other in applications. As a result, we cannot connect these query points sequentially to formulate a trajectory and then call the solution designed for the KNN trajectory query to solve it.

As illustrated in Figure 9 C), a KNN trajectory query can find the GPS logs of people traveling through a specific route. To answer such a query, the first step is to define a similarity/distance function between two trajectories. Then efficient query processing algorithms are designed to address the problem of searching over a large set of candidate trajectories. Sometimes, we need to retrieve the trajectories of vehicles traversing a specific path. There are two ways to achieve the goal.

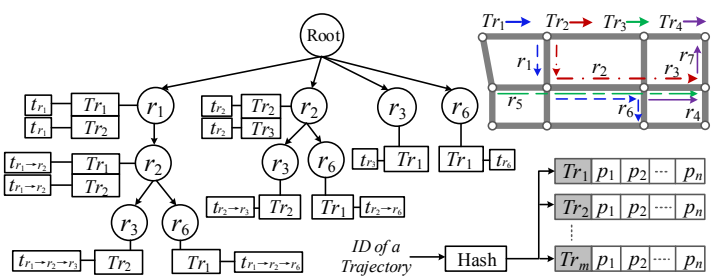



**Figure 11. Suffix-tree-like index for maintaining trajectories**

One is to regard a path on a road network as a trajectory and use the KNN trajectory query to detect the trajectories that are close to the path. The other way is first to convert a trajectory into a sequence of road segments by using a map matching algorithm. Some indexing structures are then built to manage the relationship between paths and the trajectories passing them. Figure 11 presents a suffix tree-based indexing structure [104] that manages the four trajectories $Tr_1$, $Tr_2$, $Tr_3$, and $Tr_4$ traversing a road network. Here, each node in the indexing tree stands for a road segment; each path on the tree corresponds to a route on the road network. Each node stores the IDs and travel times of the trajectories

that traverse the path from the root to the node. For example, $t_{r_1 \to r_2 \to r_3}$ stands for the time for traveling path $r_1 \to r_2 \to r_3$. By searching for the tree, we can easily get the IDs of trajectories passing a path and retrieve the points of each trajectory through a hash table (as shown in the bottom-right part of Figure 11). The detailed content of each trajectory can be stored on disk if the memory is not big enough. Because the size of the index grows quickly as the number of trajectories increases, such index is only suitable for managing trajectories generated recently.

### 4.2 Distance/Similarity of Trajectories

When answering KNN queries or clustering trajectories, we need to calculate the distance (alternatively we can say similarity) between a trajectory and a few points, or the distance between two trajectories.

***The distance between a point $q$ and a trajectory*** $A$ is usually measured by the distance from $q$ to its nearest point in $A$, denoted as $D(q, A) = min_{p \in A} D(p, q)$; e.g. $q_1$ and $p_2$ shown in Figure 9 B). An approach extending the distance from a single point $q$ to multiple query points $Q$ is $D(Q, A) = \sum_{q \in Q} e^{D(q,A)}$, or $S(Q, A) = \sum_{q \in Q} e^{-D(q,A)}$, written in a similarity fashion. The intuition of using the exponential function is to assign a larger contribution to a closer matched pair of points while giving much lower value to those far-away pairs. Chen et al. [21] define the Best Connect Distance which can measure the distance between a trajectory and a few points with or without an order.

***The Distance between two trajectories*** is usually measured by some kind of aggregation of distances between trajectory points. Closest-Pair Distance uses the minimal distance between the points in two trajectories $(A, B)$ to represent the similarity of trajectories, i.e. $CPD(A, B) = min_{p \in A, p' \in B} D(p, p')$. Assuming that two trajectories are of the same length, Sum-of-Pairs Distance uses the sum of corresponding points from the two trajectories to denote the distance, i.e. $SPD(A, B) = \sum_{i=1}^{n} D(p_i, p'_i)$. As the assumption may not hold in reality, Dynamic Time Wrapping (DTW) distance was proposed to allow 'repeating' some points as many times as needed in order to get the best alignment [3]. As some noise points from a trajectory may cause a big distance between trajectories, the concept of the Longest Common Sub-Sequence (LCSS) is employed to address this issue. The LCSS-based Distance allows to skip some noise points when calculating the distance of trajectories, using a threshold $\delta$ to control how far in time we can go in order to match one point from a trajectory to a point in another trajectory. Another threshold $\varepsilon$ is used to determine whether two points (from two different trajectories) are matched. Chen et al. [18] proposed the EDR distance, which is similar to LCSS in using a threshold $\varepsilon$ to determine a match, while assign penalties to the gaps between two matched sub-trajectories. In [19], Chen et al. also proposed the ERP distance aiming to combine the merits of DTW and EDR, by using a constant reference point for computing distance. Note that none of DTW, LCSS and EDR is not a metric distance function, as they do not satisfy the triangle inequality. ERP is metric, thus can be used to prune unnecessary trajectories [18].

Basically, LCSS and Edit Distance were proposed for matching strings. When used to match two trajectories, there is a threshold $\varepsilon$ need to set; this is not easy. K-BCT [21] is a parameter-free similarity metric for trajectories, combining the merits of DTW and LCSS. During the matching process, K-BCT can repeat some trajectory points and skip un-matched trajectory points including outliers.

***The distance between two trajectory segments***: A distance measure for trajectory segments is based on the Minimum Bounding Rectangles (MBR) of segments [43]. As

demonstrated in Figure 12 A), the MBRs of two segments $(L_1, L_2)$ are $(B_1, B_2)$, each of which is described by the coordinates of the low bound point $(x_l, y_l)$ and upper bound point $(x_u, y_u)$. The MBR-Based distance $D_{min}(B_1, B_2)$ is defined as the minimum distance between any two points from $(B_1, B_2)$, calculated as:

$$\sqrt{(\Delta([x_l, x_u], [x'_l, x'_u]))^2 + (\Delta([y_l, y_u], [y'_l, y'_u]))^2};$$

Where the distance between two intervals is defined as:

$$\Delta([x_l, x_u], [x'_l, x'_u]) = \begin{cases} 0 & [x_l, x_u] \cap [x'_l, x'_u] \neq \emptyset \\ x'_l - x_u & x'_l > x_u \\ x_l - x'_u & x_l > x'_u \end{cases};$$

In the two examples shown in Figure 12 A), the distance between $L_1$ and $L_2$ is 0 and $y'_l - y_u$, respectively.

As depicted in Figure 12 B), Lee et al. [51] proposed a distance function, entitled Trajectory-Hausdorff Distance ($D_{Haus}$), which is a weighted sum of three terms: 1) The aggregate perpendicular distance ($d_\perp$) that measures the separation between two trajectories; 2) The aggregate parallel distance ($d_{//}$) that captures the difference in length between two trajectories; 3) The angular distance ($d_\theta$) that reflects the orientation difference between two trajectories. Formally,

$$D_{Haus} = w_1 d_\perp + w_2 d_{//} + w_3 d_\theta,$$

where $d_\perp = \frac{d_{\perp,a}{}^2 + d_{\perp,b}{}^2}{d_{\perp,a} + d_{\perp,b}}$, $d_{//} = \min(d_{//,a}, d_{//,b})$, $d_\theta = ||L_2|| \cdot sin\theta$, and $w_1$, $w_2$, and $w_3$ are weights depending on applications.

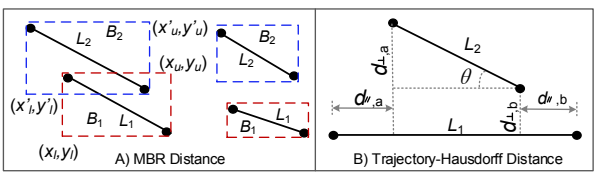



**Figure 12. Distance metrics for trajectory segments**

## 5. UNCERTAINTY IN A TRAJECTORY

As the location of a moving object is recorded at a certain time interval, the trajectory data we obtain is usually a sample of the object's true movement. On one hand, the movement of an object between two consecutive sampling points becomes unknown (or called uncertain). To this end, we expect to reduce the uncertainty of a trajectory. On the other hand, in some applications, to protect a user's privacy that could be leaked from her trajectories, we need to make a trajectory even more uncertain.

### 5.1 Reducing Uncertainty from Trajectory Data

Many trajectories have been recorded with a very low sampling rate, leading to an object's movement between sampling points uncertain; we call them uncertain trajectories. For instance, as shown in Figure 13 A), the GPS coordinates of a taxi $(p_1, p_2, p_3)$ were recorded every few minutes to reduce communication loads, resulting in multiple possible paths between two consecutive sampling points. As illustrated in Figure 13 B), people's check-in records in a location-based social networking service like FourSqure can be regarded as

trajectories if we connect them chronologically. As people do not check in very often, the time interval (and distance) between two consecutive check-ins may be hours (and several kilometers). Consequently, we have no idea how a user traveled between two check-ins. As demonstrated in Figure 13 C), to save energy, the GPS logger installed on a migratory bird can only send a location record every half day. As a result, the path that a bird flied over two particular locations is quite uncertain.

*5.1.1 Modeling Uncertainty of a Trajectory for Queries* Several models of uncertainty paired with appropriate query evaluation techniques [81][23] have been proposed for moving object databases to answer queries, e.g. "is it possible for an object to intersect a query window". As illustrated in Fig 13 B), we do not know whether the trajectory formulated by the three blue check-ins should be retrieved or not by the range query $R$, without modeling the uncertainty of the trajectory. Many of these techniques aim at providing conservative bounds for the positions of uncertain objects between two sampling points. This is usually achieved by employing geometric objects, such as cylinders [101][100] or beads [99], as trajectory approximations. These models concern more little about data mining, therefor are not the focus of our paper. Recent approaches use independent probability density functions at each point of time [24], or stochastic processes [85][109][30][74] (e.g. Markov chains), to better model the uncertain positions of an object and answer different queries.

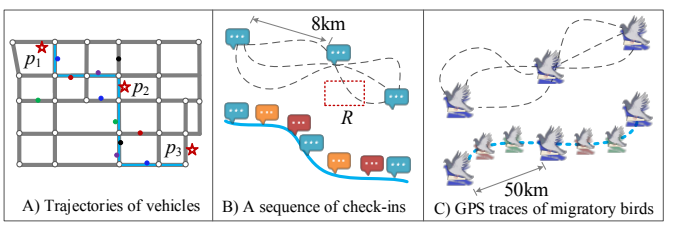



**Figure 13. Examples of uncertain trajectories**

*5.1.2 Path Inference from Uncertain Trajectories* Different from the aforementioned models aiming at the retrieval of existing trajectories by different queries, a new series of techniques infers (or say 'constructs') the most likely $k$-route(s) that a moving object could travel (i.e. the missing sub-trajectory) between a few sample points based on a bunch of uncertain trajectories. The major insight is that trajectories sharing (or partially sharing) the same/similar routes can often supplement each other to make themselves more complete. In other words, it's possible to interpolate an uncertain trajectory by cross-referring other trajectories on (or partially on) the same/similar route, i.e. "uncertain + uncertain → certain". For example, given the uncertain trajectories of many taxicabs (marked by different colored points in Figure 13 A)), we could infer that the blue path is the most likely route traversing $(p_1, p_2, p_3)$. Likewise, based on check-in data of many users, as depicted in Fig 13 B), we could find the blue curve the most possible travel path between the three blue check-ins. Similarly, given the uncertain GPS traces of many birds, we can identify the path that birds fly over a few locations. Reducing the uncertainty of trajectories can support scientific studies and enable many applications, such as travel recommendation and traffic management. There are two categories of methods to complement an uncertain trajectory:

One is designed for the trajectories generated in a road network setting [134]. What set this category of methods apart from map-matching algorithms lies in two aspects. First, the methods for reducing the uncertainty of trajectories leverage the data from many other

trajectories, while map-matching algorithms only use the geometric information from a single trajectory and the topological information of road networks. Second, the sampling rate of trajectories handled by the uncertainty methods can be very low, e.g. more than 10 minutes. This seems nearly impossible for a map matching algorithm.

The other is for a free space, where moving objects (like flying birds or people hiking a mountain) do not follow paths in road networks [105], as illustrated in Figure 13 B) and C). The major challenges are two folds. One is to determine those trajectories which may be relevant to a series of query points. The other is to construct a route that can approximate a bunch of relevant trajectories. As shown in Figure 14 A), the method proposed in [105] first partitions a geo-space into uniform grids (the size of a grid depends on the required inference accuracy) and then maps trajectories onto these grids. Some grids can be connected to form a region if the trajectories passing them satisfy one of the following two rules: 1): *If* the starting points $(p_1^1, p_1^2)$ of two trajectory segments are located in two grids $(g_1, g_2)$ that are geospatial neighbors, and the ending points $(p_2^1, p_2^2)$ of the two segments are located in the same grid, and the travel times $(\Delta t_1, \Delta t_2)$ of the two segments are similar, *then* the two grids $(g_1, g_2)$ can be connected. 2): *If* the starting points $(p_2^1, p_2^2)$ are located in the same grid, and ending points $(p_3^1, p_3^2)$ fall in the grids $(g_4, g_5)$ that are neighbors, travel times $(\Delta t_1', \Delta t_2')$ of the two segments are similar, then grids $(g_4, g_5)$ can be connected.

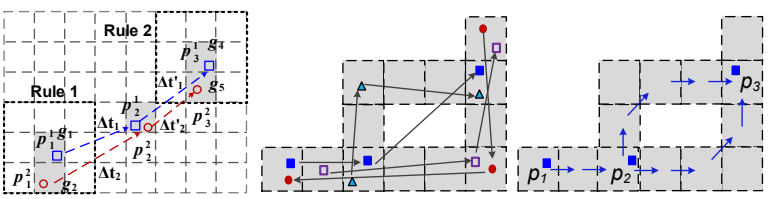



**Figure 14. The most-likely route based on uncertain trajectories**

After turning disjoint grids into connected region(s), as demonstrated in Figure 14 B), we can build a routable graph where a node is a grid. The direction and travel time between two adjacent grids in the graph is inferred based on the trajectories passing the two grids. Finally, as depicted in Figure 14 C), given three query points, we can find the most likely route on the graph based on a routing algorithm. To find a more detailed path, a regression can be performed over the trajectories passing the identified route.

Su et al. [91] proposed an anchor-based calibration system that aligns trajectories to a set of fixed anchor points. The approach considers the spatial relationship between anchor points and trajectories. It also trains inference models from historical trajectories to improve the calibration.

## 5.2 Privacy of Trajectory Data

Instead of making a trajectory more certain, a series of techniques aim to protect a user from the privacy leak caused by the disclosure of the user's trajectories [1][111][25]. This kind of technology tries to blur a user's location, while ensuring the quality of a service or the utility of the trajectory data. There are two major scenarios that we need to protect a user's trajectory data from the privacy leak.

One is in real-time continuous location-based services, e.g. tell me the traffic conditions that is 1km around me. In this scenario, a user may not want to exactly disclose her current location when using a service. Different from the simple location privacy, the spatio-

temporal correlation between consecutive samples in a trajectory may help infer the exact location of a user. Techniques trying to protect the privacy leak in this scenario include, spatial cloaking [69], mix-zones [8], Path confusion [40], Euler histogram-based on short IDs [108], dummy trajectories [47], etc.

The second is the publication of historical trajectories. Collecting many trajectories of an individual may allow attackers to infer her home and work places, therefore identifying who the individual is. Major techniques for protecting users' privacy in such scenario include, clustering-based [1], generalization-based [72], suppression-based [98], and grid-based approach [35]. A comprehensive survey on trajectory privacy can be found in [25].

## 6. TRAJECTORY PATTERN MINING

In this section, we study four major categories of patterns that can be discovered from a single trajectory or a group of trajectories. They are moving together patterns, trajectory clustering, sequential patterns, and periodic patterns.

### 6.1 Moving Together Patterns

A branch of research is to discover a group of objects that move together for a certain time period, such as *flock* [37][38], *convoy* [42][44], *swarm* [55], *traveling companion* [96][97], and *gathering* [135][136]. These patterns can help the study of species' migration, military surveillance, and traffic event detection, etc. These patterns can be differentiated between each other based on the following factors: the shape or density of a group, the number of objects in a group, and the duration of a pattern.

Specifically, a *flock* is a group of objects that travel together within a disc of some user-specified size for at least $k$ consecutive timestamps. A major concern with *flock* is the pre-defined circular shape, which may not well describe the shape of a group in reality, therefore may result in the so-called lossy-flock problem. To avoid rigid restrictions on the size and shape of a moving group, the *convoy* is proposed to capture generic trajectory pattern of any shape by employing the density-based clustering. Instead of using a disc, a *convoy* requires a group of objects to be density-connected during $k$ consecutive time points. While both *flock* and *convoy* have a strict requirement on consecutive time period, Li et al. [55] proposed a more general type of trajectory pattern, called *swarm*, which is a cluster of objects lasting for at least $k$ (possibly non-consecutive) timestamps. While *convoy* and *swarm* need to load entire trajectories into memory for a pattern mining, the *traveling companion* [96] uses a data structure (called *traveling buddy*) to continuously find *convoy/swarm*-like patterns from trajectories that is being streamed into a system. So, the *traveling companion* patterns can be regarded as an online (and incremental) detection fashion of *convoy* and *swarm*.

To detect some incidents, such as celebrations and parades, in which objects join in and leave an event frequently, the *gathering* pattern [135] [136] further looses the constraints of the aforementioned patterns by allowing the membership of a group to evolve gradually. Each cluster of a *gathering* should contain at least $m_p$ participators, which are the objects appearing in at least $k_p$ clusters of this gathering. As the *gathering* pattern is used to detect events, it also requires the geometric property (like location and shape) of a detected pattern relative stable.

Figure 15 A) illustrates these patterns. If set the requirement of timestamps $k = 2$, a group $< o_2, o_3, o_4 >$ is a *flock* from $t_1$ to $t_3$. Though $o_5$ is a companion of the group, it cannot be included due to the fixed size of the disc employed by the *flock* definition. On

the other hand, a *convoy* can include $o_5$ into the group, since $< o_2, o_3, o_4, o_5 >$ is density-based connected from $t_1$ to $t_3$. The five objects also form a *swarm* during the non-consecutive time period $t_1$ and $t_3$. As demonstrated in Figure 15 B), if we set $k_p$=2 and $m_p$=3, then $< C_1, C_2, C_4 >$ is a gathering. $< C_1, C_3, C_5 >$ is not a gathering as $C_5$ is too far away from $C_2$ and $C_3$.

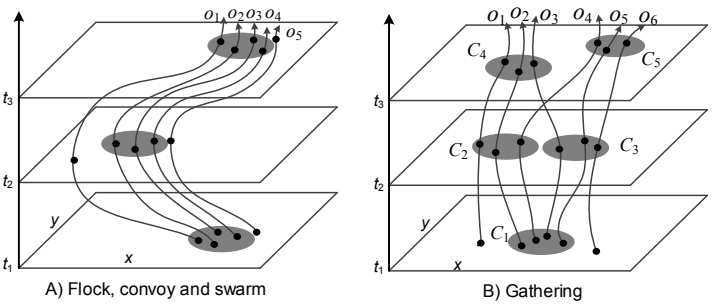



**Figure 15. Examples of moving together patterns**

The aforementioned pattern mining algorithms usually use a density-based distance metric (in a Euclidian Space) to find a cluster of moving objects. Christian et al. [41] extend the distance metric by considering semantic factors, such as heading directions and speed, of a moving object.

## 6.2 Trajectory Clustering

To find representative paths or common trends shared by different moving objects, we usually need to group similar trajectories into clusters. A general clustering approach is to represent a trajectory with a feature vector, denoting the similarity between two trajectories by the distance between their feature vectors. However, it is not easy to generate a feature vector with a uniform length for different trajectories, as different trajectories contain different and complex properties, such as length, shape, sampling rate, number of points and their orders. In addition, it is difficult to encode the sequential and spatial properties of points in a trajectory into its feature vector.

Given the challenges mentioned above, a series of technique works have been done. Since the distance metrics between trajectories have been introduced in Section 4.2, we hereafter focus on the clustering methods proposed for trajectories. Note that the clustering methods discussed in this section are dedicated for trajectories in free spaces (i.e. without a road network constraint). Though there are a few publications (e.g. [48]) discussing the trajectory clustering in a road network setting, this problem can actually be solved by the combination of map-matching and graph clustering algorithms. That is, we can first use map-matching algorithms to project trajectories onto a road network and then employ graph clustering algorithms to find a subgraph (i.e. a collection of roads) on the road network.

相关

Gaffney et al. [13][33] proposed to group similar trajectories into clusters by using a regression mixture model and the EM algorithm. This algorithm clusters trajectories with respect to the overall distance between two entire trajectories. However, moving objects rarely travel together for an entire path in the real world. To this end, Lee et al. [51] proposed to partition trajectories into line segments and to build groups of close trajectory segments using the Trajectory-Hausdorff Distance, as illustrated in Figure 16 A). A representative path is later found for each clusters of segments. Since trajectory data are

often received incrementally, Li et al. [56] further proposed an incremental clustering algorithm, aiming to reduce the computational cost and storage of received trajectories. Both Lee [51] and Li [56] adopted a *Micro-and-Macro-clustering framework*, which was proposed by Aggarwal et al. [2] to cluster data streams. That is, their methods first find mirco-clusters of trajectory segments (as demonstrated in Figure 16 B), and then group micro-clusters into macro-clusters (as shown in Figure 16 C). A major insight of Li's work [56] is that new data will only affect the local area where the new data were received rather than the far-away areas.

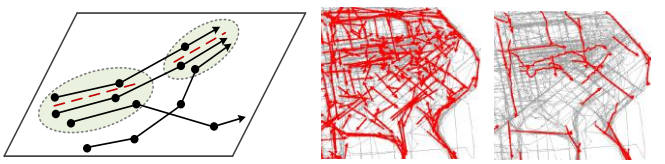



**Figure 16. Trajectory clustering based on partial segments** [56]

## 6.3 Mining Sequential Patterns from Trajectories

A branch of research is to find the *sequential patterns* from a single trajectory or multiple trajectories. Here, a sequential pattern means a certain number of moving objects traveling a common sequence of locations in a similar time interval. The locations in a travel sequence do not have to be consecutive. For instance, two trajectories $A$ and $B$,

$$A: l_1 \xrightarrow{1.5h} l_2 \xrightarrow{1h} l_7 \xrightarrow{1.2h} l_4. \quad B: l_1 \xrightarrow{1.2h} l_2 \xrightarrow{2h} l_4,$$

share a common sequence $l_1 \rightarrow l_2 \rightarrow l_4$, as the visiting orders and travel times are similar (though $l_2$ and $l_4$ is not consecutive in trajectory $A$). When the occurrence of such a common sequence in a corpus, usually called support, exceeds a threshold, a *sequential trajectory pattern* is detected. Finding such kind of patterns can benefit travel recommendation [154][34], life pattern understanding [116], next location prediction [71], estimating user similarity [107][54], and trajectory compression [90].

To detect the sequential patterns from trajectories, we first need to define a (common) location in a sequence. Ideally, in trajectory data, like user check-in sequences from a social networking service, each location is tagged with a unique identity (such as the name of a restaurant). If two locations share the same identity, they are common. In many GPS trajectories, however, each point is characterized by a pair of GPS coordinates, which do not repeat themselves exactly in every pattern instance. This makes the points from two different trajectories are not directly comparable. In addition, a GPS trajectory may consist of thousands of points. Without handled properly, these points will result in a huge computational cost.

*6.3.1 Sequential Pattern Mining in a Free Space* ***Line-simplification-based methods***: An early solution aiming to deal with the aforementioned issues was proposed in 2005 [11]. The solution first identifies key points shaping a trajectory, by using a line simplification algorithm like DP [28]. It then groups the fragments of a trajectory that are close to each simplified line segment so as to count the support of each line segment. The travel time between two points in a trajectory is not considered.

***Clustering-based methods***: Recently, a more general way to solve the above-mentioned problems is to cluster points from different trajectories into regions of interests. A point from a trajectory is then represented by the cluster ID the point belongs to. As a consequence, a trajectory is re-formed as a sequence of cluster IDs, which are comparable

among different trajectories. For example, as shown in Figure 17 A), the three trajectories can be represented as:

$$Tr_1: l_1 \xrightarrow{\Delta t_3} l_3\ ,\ Tr_2: l_1 \xrightarrow{\Delta t_1} l_2 \xrightarrow{\Delta t_2} l_3,\ Tr_3: l_1 \xrightarrow{\Delta t'_1} l_2 \xrightarrow{\Delta t'_2} l_3,$$

where $l_1$, $l_3$ and $l_3$ are clusters of points. After the transformation, we can mine the sequential patterns from these sequences by using existing sequential pattern mining algorithms, such as PrefixSpan [80] and CloseSpan [112], with time constraints. In this example, setting the support threshold to 3, we can find $l_1 \rightarrow l_3$ is a sequential pattern if

$$\frac{|\Delta t_3 - (\Delta t_1 + \Delta t_2)|}{\max(\Delta t_3, \Delta t_1 + \Delta t_2)} < \rho, \text{ and } \frac{|\Delta t_3 - (\Delta t'_1 + \Delta t'_2)|}{\max(\Delta t_3,\ \Delta t'_1 + \Delta t'_2)} < \rho$$

where $\rho$ is a ratio threshould guaranteeing that two travel times are similar. Likewise, setting the threshold of support to 2, $l_1 \rightarrow l_2 \rightarrow l_3$ is a sequential pattern, if $\Delta t_1$ is similar to $\Delta t'_1$ and $\Delta t_2$ is similar to $\Delta t'_2$. Towards this direction, Giannotti et al. [34] divide a city into uniform grids, grouping these grids into regions of interests based on the density of GPS points fallen in each grid. An Apriori-like algorithm is then proposed to detect the sequential patterns of the region of interests.

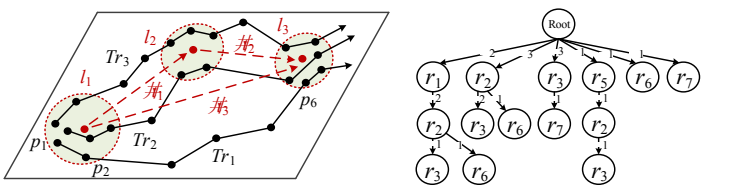



**Figure 17. Sequential pattern mining in trajectory data**

With respect to the applications caring more about the semantic meaning of a location, we can first detect stay points from each trajectory, turning a trajectory into a sequence of stay points (see Section 3.2). Later, we can cluster these stay points to formulate regions of interests and use the cluster ID that a stay point belongs to represent a trajectory. Following this strategy, Ye and Zheng et al. [116] proposed to mine life patterns from an individual's GPS trajectories. Xiao et al. [106][107] proposed a graph-based sequence matching algorithm to find the sequential pattern shared by two users' trajectories. These patterns are then used to estimate the similarity between two users.

*6.3.2 Sequential Pattern Mining in a Road Network* When the sequential pattern mining problem is applied to a road network setting, we can first map each trajectory onto a road network by using map-matching algorithms. A trajectory is then represented by a sequence of road segment IDs, which can be regarded as strings. As a result, some sequential pattern mining algorithms, such as LCSS and Suffix Tree, designed for strings can be adapted to finding sequential trajectory patterns. Figure 17 B) presents a suffix tree that represents the four trajectories depicted in Figure 11. Here, a node is a road segment, and the path from the root to a node corresponds to a suffix of the string representing a trajectory. For example, $Tr_1$ is represented by a string $r_1 \rightarrow r_2 \rightarrow r_6$, where $r_2 \rightarrow r_6$ and $r_6$ are suffixes of the string. The number associated with each link denotes the number of trajectories traversing the path, i.e. the support of the string pattern. For instance, there are two trajectories ($Tr_1$ and $Tr_2$) traversing $r_1 \rightarrow r_2$ and one trajectory traversing $r_1 \rightarrow r_2 \rightarrow r_6$. After building such a suffix tree, we can find the frequent patterns (i.e. the paths on the tree) with a support greater than a given threshold, with a complexity of $O(n)$. Note that the size of a suffix tree can be much bigger than the original trajectories. So, when the size

of a trajectory dataset is very large, we need to set a constraint on the depth of its suffix tree. Additionally, the sequential patterns derived from the suffix tree have to be consecutive. Though the temporal constraint is not explicitly considered, two objects' travel times on the same path should be similar, given the speed constraint of a path.

Towards this direction, Song et al. [90] use suffix tree to detect frequent trajectory patterns, which are then leveraged to compress trajectories in conjunction with Huffman Encoding. Wang and Zheng et al. [104] employ suffix tree to find frequent trajectory patterns, which are used to reduce the candidates of combination of sub-trajectories when estimating the travel time of a query path.

### 6.4 Mining Periodical Patterns from Trajectories

Moving objects usually have periodic activity patterns. For example, people go shopping every month and animals migrate yearly from one place to another. Such periodic behaviors provide an insightful and concise explanation over a long moving history, helping compress trajectory data and predict the future movement of a moving object.

Periodic pattern mining has been studied extensively for time series data. For example, Yang et al. tried to discover asynchronous patterns [113], surprising periodic patterns [114], and patterns with gap penalties [115], from (categorical) time series. Due to the fuzziness of spatial locations, existing methods designed for time series data are not directly applicable to trajectories. To this end, Cao et al. [12] proposed an efficient algorithm for retrieving maximal periodic patterns from trajectories. This algorithm follows a paradigm that is similar to *frequent* pattern mining, where a (global) minimum support threshold is needed. In the real world, however, periodic behaviors could be more complicated, involving multiple interleaving periods, partial time span, and spatiotemporal noises and outliers.

To deal with these issues, Li et al. [57] proposed a two-stage detection method for trajectory data. In the first stage, the method detects a few reference spots, where a moving object has visited frequently, by using a density-based clustering algorithm, such as KDE. The trajectory of a moving object is then transformed into several binary time series, each of which indicates the "in" (1) and "out" (0) status of the moving object at a reference spot. Through applying Fourier Transform and autocorrelation methods to each time series, the values of periods at each reference spot can be calculated. The second stage summarizes the periodic behaviors from partial movement sequences by using a hierarchical clustering algorithm. In 2012, Li et al. [58] further extend the research [57] to mining periodic patterns from incomplete and sparse data sources.

## 7. TRAJECTORY CLASSIFICATION

Trajectory classification aims to differentiate between trajectories (or its segments) of different status, such as motions, transportation modes, and human activities. Tagging a raw trajectory (or its segment) with a semantic label raises the value of trajectories to the next level, which can facilitate many applications, such as trip recommendation, life experiences sharing, and context-aware computing.

In general, trajectory classification is comprised of three major steps: 1) Divide a trajectory into segments using segmentation methods. Sometimes, each single point is regarded as a minimum inference unit. 2) Extract features from each segment (or point). 3) Build a model to classify each segment (or point). As a trajectory is essentially a sequence, we can leverage existing sequence inference models, such as Dynamic Bayesian Network (DBN), HMM and Conditional Random Field (CRF), which incorporate the information

from local points (or segments) and the sequential patterns between adjacent points (or segments).

Using a sequence of 802.11 radio signals, LOCADIO [50] employs a Hidden Markov Model to classify the motion of a device into two status: Still and Moving. Based on a trajectory of GSM signals, Timothy et al. [102] attempted to classify the mobility of a user into three status, consisting of stationary, walking and driving. Zhu et al. [158] aim to infer the status of a taxi, consisting of *Occupied*, *Non-occupied*, and *Parked*, according to its GPS trajectories. They first seek the possible *Parked* places in a trajectory, using a stay point-based detection method. A taxi trajectory is then partitioned into segments by these *Parked* places (refer to Figure 6 D) for an example). For each segment, they extract a set of features incorporating the knowledge of a single trajectory, historical trajectories of multiple taxis, and geographic data like road networks and POIs. After that, a two-phase inference method is proposed to classify the status of a segment into either *Occupied* or *Non-occupied*. The method first uses the identified features to train a local probabilistic classifier and then globally considers travel patterns via a Hidden Semi-Markov Model.

Zheng et al. [149][147]classify a user's trajectory by transportation modes, which is comprised of *Driving*, *Biking*, *Bus*, and *Walking*. As people usually change transportation modes in a single trip, a trajectory is first partitioned into segments based on the *Walk*-based segmentation method (refer to Figure 7 for details). A set of features, such as the heading change rate, stop rate and velocity change rate, are extracted, being fed into a Decision Tree Classifier. Based on the inference results, a graph-based post-processing step is conducted to fix the possibly wrong inference, considering the transition probability between different transportation modes at different places.

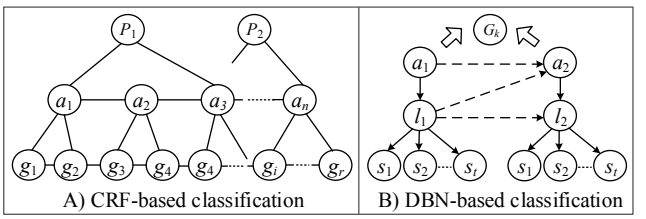



**Figure 18. Trajectory classification for activity recognition**

Lin et al. [59][79] proposed a hierarchical inference model for location-based activity recognition and significant place discovery, as shown in Figure 18 A). A GPS trajectory is first divided into 10-meter segments, each of which is then projected onto corresponding street patches by using a CRF-based map matching algorithm. Based on the features extracted from these street patches, the model classifies a sequence of GPS points into a sequence of activities like $a_1, a_2, \ldots, a_n$ (such as *Walk*, *Driving* and *Sleep*) and identifies a person's significant places like $P_1$ and $P_2$ (e.g. home, work, and bus stops), simultaneously. Yin et al. [119] proposed a DBN-based inference model to infer a user's activities as well as high level goals, according to a sequence of WiFi signals. Figure 18 B) presents the structure of the DBN, where the bottom layer contains the input of raw WiFi signals; the second layer is a list of locations where these signals are received; the top level corresponds to user activities. Finally, the high-level goal is inferred based on the sequence of inferred activities.

## 8. ANOMALIES DETECTION FROM TRAJECTORIES

Trajectory outliers (a.k.a. anomalies) can be items (e.g. a trajectory or a segment of trajectory) that is significantly different from other items in terms of some similarity metric. It can also be events or observations (represented by a collection of trajectories) that do not conform to an expected pattern (e.g. a traffic congestion caused by a car accident). A survey on general anomaly detection methods can be found in [14].

### 8.1 Detecting Outlier Trajectories

An outlier trajectory is a trajectory or a part of a trajectory that is significantly different from others in a corpus in terms of a distance metric, such as shape and travel time. The outlier trajectories could be a taxi driver's malicious detour [61][131] or unexpected road changes (due to traffic accidents or constructions). It can also remind people when traveling on a wrong path.

A general idea is to leverage existing trajectory clustering or frequent pattern mining methods. If a trajectory (or a segment) cannot be accommodated in any (density-based) clusters, or not frequent, it may be an outlier. Lee et al. [52] proposed a partition-and-detection framework to find anomalous segments of trajectories from a trajectory data set. This method can be an extension of the trajectory clustering proposed in [51].

### 8.2 Identifying Anomalous Events by Trajectories

Another direction is to detect traffic anomalies (rather than trajectory itself) by using many trajectories. The traffic anomalies could be caused by accidents, controls, protests, sports, celebrations, disasters and other events.

Liu et al. [62] partition a city into disjointed regions with major roads and glean the anomalous links between two regions according to the trajectories of vehicles traveling between the two regions. They divide a day into time bins and identify for each link three features: the number of vehicles traveling a link in a time bin, proportion of these vehicles among all vehicles entering the destination region, and that departing from the origin region. The three features of a time bin were respectively compared with those in the equivalent time bins of previous days to calculate the minimum distort of each feature. Then, the link of the time bin can be represented in a 3-dimension space, with each dimension denoting the minimum distort of a feature. Later, the Mahalanobis distance is used to measure the extreme points (in the 3D space), which are regarded as outliers. Following the aforementioned research, Sanjay et al. [15] proposed a two-step mining and optimization framework to detect traffic anomalies between two regions and explain an anomaly with the traffic flows passing the two regions (see Section 10 for details).

Pan and Zheng et al. [76] identify traffic anomalies according to drivers' routing behavior on an urban road network. Here, a detected anomaly is represented by a sub-graph of a road network where drivers' routing behaviors significantly differ from their original patterns. They then tried to describe the detected anomaly by mining representative terms from the social media that people have posted when the anomaly was happening.

Pang et al. [77][78] adapt likelihood ratio test, which have previously been used in epidemiological studies, to describe traffic patterns. They partitioned a city into uniform grids and count the number of vehicles arriving in a grid over a time period. The objective is to identify contiguous set of cells and time intervals which have the largest statistically significant departure from expected behavior (i.e., the number of vehicles). The regions whose log-likelihood ratio statistic value drops in the tail of $\chi^2$ distribution are likely to be anomalous [14].

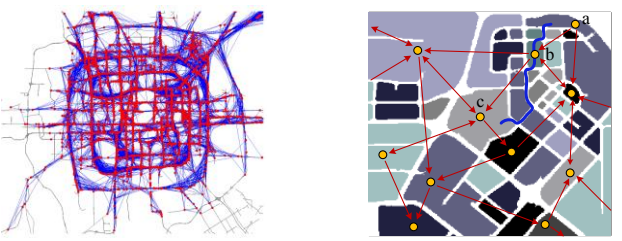
# 9. TRANSFER TRAJECTORY TO OTHER REPRESENATIONS

## 9.1 From Trajectory to Graph

Trajectories can be transformed into other data structures, besides being processed in its original form. This enriches the methodologies that can be used to discover knowledge from trajectories. Turning trajectories into graphs is one of the representative types of transformation. When conducting such a transformation, the main effort is to define what a node and an edge is in the transformed graph. The methods for transforming trajectories into a graph differentiate between one another, depending on whether a road network is involved in the transformation.

*9.1.1 In a Road Network Setting* A road network is essentially a directed graph, where a node is an intersection and an edge denotes a road segment. Consequently, the most intuitive approach to turning trajectories into a graph is to project trajectories onto a road network. We can then calculate some weights, such as speed and traffic volume, for the edges based on the projected trajectories. Later, given the weighted graph, we can find the most-likely route (traveled by people) between a few query points [134], identify the most popular route between a source and a destination [64], detect traffic anomalies [76], and update maps automatically.

The second approach is to build a *landmark* graph. For example, Yuan and Zheng et al. [125][124] proposed an intelligent driving direction system, entitled T-Drive, based on the GPS trajectories generated by a large number of taxicabs. After the map matching process, T-Drive regards the top $k$ road segments frequently traversed by taxicabs as *landmark* nodes (i.e. the red points shown in Figure 19 A). The trajectories traversing two *landmarks* consecutively are aggregated into a *landmark* edge (denoted by a blue line), being used to estimate the travel time between two *landmarks*. A two-stage routing algorithm is proposed to find the fastest driving path. The algorithm first searches the *landmark graph* for a rough route (represented by a sequence of *landmarks*), and then finds a detailed route connecting consecutive *landmarks* on the original road network.




A) A landmark graph B) A region graph

**Figure 19. Transforming trajectories into graphs**

The third approach is to build a *region graph*, where a node denotes a region and an edge stands for the aggregation of commutes between the two regions. For instance, as illustrated in Figure 19. B), using an image segmentation-based algorithm [128], Zheng et al. [151] partition a city into regions by major roads so as to detect the underlying problems in a city's road network. A region bounded by major roads is then represented by a node, and two regions are connected with an edge if there are a certain number of commutes between them. After the transformation, they glean the region pairs (i.e. edges) that are not well connected, i.e. with a huge traffic volume, a slow travel speed, and a long detour between them, using a skyline algorithm. The region graphs are also employed to detect

traffic anomalies [62][15] and urban functional regions [122][130].

*9.1.2 In Free Spaces* Another branch of research transfers trajectories into a graph without using a road network, according to two major steps: 1) Identify key locations as vertexes from raw trajectories by using clustering methods; 2) Connect the vertexes to formulate a routable graph based on trajectories passing two locations.

***Travel recommendation***: Zheng et al. [155][152] proposed to find the interesting locations and travel sequences from trajectories generated by many people. In the method, they first detect stay points from each trajectory and then cluster the stay points from different people into locations, as shown in Figure 20 A). Based on these locations and raw trajectories, they build a user-location bipartite graph as illustrated in Figure 20 B), as well as a routable graph between locations, as depicted in Figure 20 C).

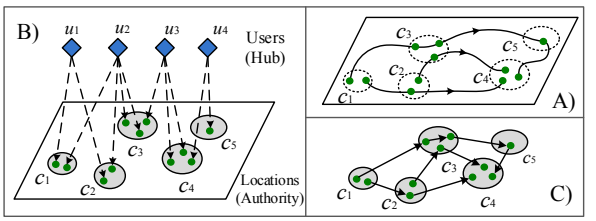



**Figure 20. mining interesting locations and travel sequences**

In the bipartite graph, a user and a location are regarded as two different types of nodes. An edge is built between a user node and a location node if the user has visited the location. A HITS (Hypertext Induced Topic Search)-based model is then employed to infer the interest level of a location (i.e. the authority score) and the travel knowledge of a user (i.e. the hub score). According to the inferred scores, we can identify the top-k most interesting locations and travel experts in a city. Jie et al. [5] apply the similar idea in a collaborative filtering framework to conduct the travel recommendation, concerned with a user's preferences, social environment and current location.

In the location graph, as shown in Figure 20 C), an edge denotes the aggregation of raw trajectories traveling through two locations. To calculate the importance (or the representativeness) of an edge in this graph, three factors are considered: 1) The authority score of the source location (of the edge) weighted by the probability of people's moving out by this edge; 2) The authority score of a destination location (in the edge) weighted by the probability of people's moving in by this edge; 3) The hub scores of the users who have traveled this edge. The score of a path is calculated by summing up the score of the edges the path contains.

Inspired by [152], a series of research was conducted to identify the popular routes from massive trajectories since 2010. Specifically, Yoon et al. [120][121] suggest the best travel route, consisting of a sequence of locations with a typical stay time interval at each location, to a user, given the user's source and destination as well as the time period the user has. Chen et al. [20] identify turning points from each raw trajectory, clustering these turning points into groups. These clusters are then used as vertexes to build a transfer network. Afterwards, the probability that people would travel from one vertex to another is calculated based on the counts of trajectories passing the two vertexes. Finally, given a source and a destination, the path with the maximum production of probabilities is found in the transfer network as the most popular route. However, the proposed method is not applicable to low-sampling-rate trajectories. To this end, Lin and Zheng et al. [105] divide

a geographical space into uniform grids and then construct routable graph based on the grids and raw trajectories. Refer to Section 5.1.2 for details.

Another branch of research is to detect the community of places based on the graph that is learned from trajectories, using some community discovery methods. A community of places is a cluster of locations with denser connections between locations in the cluster than between clusters. For example, Rinzivillo et al. [88] aim to find the borders of human mobility at the lower spatial resolution of municipalities or counties. They mapped vehicle GPS tracks onto regions to formulate a complex network in Pisa. A community discovery algorithm, namely Infomap, was then used to partition the network into non-overlapped sub-graphs. More semantic meanings of a trajectory, such as a user's travel speed and experiences, have been considered in [60][156] to estimate the strength of interaction between two locations.

***Estimating user similarity***: Another series of research transfers users' trajectories into hierarchical graphs so as to compute the similarity between different users. This is a foundation of many social applications, such as friend recommendation and community discovery.

As illustrated in Figure 21, Zheng et al. [154] deposit together the stay points detected from different users' trajectories, clustering them divisively by using a density-based clustering algorithm iteratively. As a result, a tree-based hierarchy is built, where a node on a higher level is a coarse-grained cluster (of stay points) and the nodes on a lower level are fine-grained clusters. The hierarchy is shared by different users as it is derived from all users' stay points. By projecting a user's trajectories onto this shared hierarchy, an individual hierarchical graph can be constructed for a user. As demonstrated in the bottom left and bottom right part of Figure 21, two users' location histories are transformed from a collection of trajectories (which are not comparable between one another) to two individual graphs with common nodes. By matching the two graphs, common sequences of clusters are found on each level of the graphs. For example, $c_{32} \rightarrow c_{31} \rightarrow c_{34}$ is a common sequence shared by the two users on the third level. Considering the popularity of a cluster in a common sequence, the length and the level (on the hierarchy) of the common sequences, a similarity score is calculated for a pair of users.

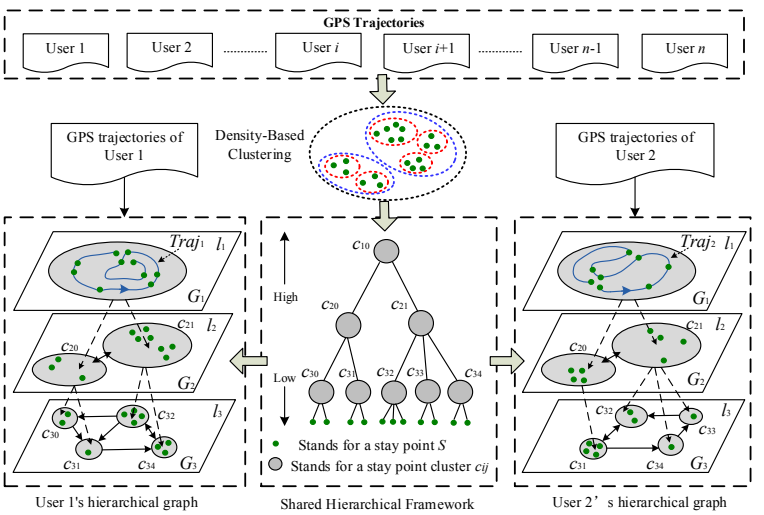



**Figure 21. Hierarchical graph-based user similarity estimation**

Xiao et al. [107] extend the similarity computing from physical locations to a semantic space, aiming to facilitate the similarity estimation between users living in different cities or countries. A stay point detected from a trajectory is represented by the distribution of POIs (across different categories) within the scope of the stay point. The stay points from different users are then clustered into a hierarchy according to their distributions on different POI categories, in a similar way to that of Figure 21.

### 9.2 From Trajectory to Matrix

Another form that we can transform trajectories into is a matrix. Using existing techniques, such as Collaborative Filtering and matrix factorization, a matrix can help complement missing observations. A matrix can also be used as an input to identify anomalies. The key of the transformation lies in three aspects: 1) What does a row mean; 2) what is a column; and 3) what does an entry denote?

***Travel recommendation***: Zheng et al. [155][154] transform users' GPS trajectory into a user-location matrix, where a row stands for a user and a column denotes a location (such as a cluster shown in Figure 21). The value of an entry means the number of visits of a user to a location. The matrix is very sparse, as a user can visit a very few locations. A collaborative filtering model is then applied to the matrix to predict a user's interests in an unvisited location.

Zheng et al. [138] proposed a coupled matrix factorization method to enable location-activity recommendation, using activity-tagged GPS trajectories. As illustrated in Figure 22, a location-activity matrix $X$ is built, with a row stands for a venue (e.g. a cluster of GPS points) and a column representing a user-labeled activity (like shopping and dinning). An entry in matrix $X$ denotes the frequency of an activity that has been observed in users' labels in a particular location. Intuitively, this is a sparse matrix. A simple method to fill the missing entries is to decompose a matrix into the production of two low-rank matrices ($U$ and $V$) based on non-zero entries. After that, the missing entries can be filled by $X = UV^T$. Once this location-activity matrix is completely filled, given an activity, the top $k$ locations, with a relatively high frequency from the column that corresponds to that activity, can be recommended. So does the activity recommendation for a location. To make a better recommendation, two context matrices, consisting of a location-feature matrix $Y$ and an activity-activity matrix $Z$, are built based on additional data sources. The main idea is to propagate the information among $X$, $Y$ and $Z$ by requiring them to share low-rank matrices $U$ and $V$ in a collective matrix factorization model.

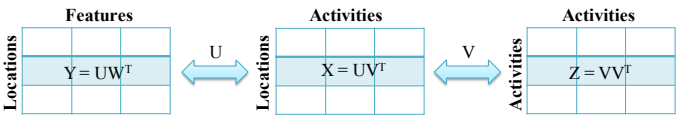



**Figure 22. Matrix factorization for recommendation**

***Traffic condition estimation***: Shang and Zheng et al. [89] proposed a coupled-matrix factorization method to instantly estimate the travel speed on each road segment throughout an entire city, based on the GPS trajectory of a sample of vehicles (such as taxicabs). As shown in Figure 23 A), after map matching the GPS trajectories onto a road network, they formulate a matrix $M'_r$ with a row denoting a time slot (e.g., 2pm-2:10pm) and a column standing for a road segment. Each entry in $M'_r$ contains the travel speed on a particular road segment and in a particular time slot, calculated based on the recently received GPS trajectories. The goal is to fill the missing values in row $t_j$, which corresponds to the current

time slot. Though we can achieve the goal by solely applying matrix factorization to $M'_r$, the accuracy of the inference is not very high as the majority of road segments are not covered by trajectories.

To address this issue, four context matrices ($M_r$, $M_G$, $M'_G$ and $Z$) are built. Specifically, $M_r$ stands for the historical traffic patterns on road segments. While the rows and columns of $M_r$ have the same meaning as $M'_r$, an entry of $M_r$ denotes the average travel speed derived from the historical data over a long period. The difference between the two corresponding entries from $M'_r$ and $M_r$ indicates the deviation of current traffic situation (on a road segment) from its average patterns. As depicted in Figure 23 B), $Z$ contains the physical features of a road segment, such as the shape of a road, number of lanes, speed constraint, and the distribution of surrounding POIs. The general assumption is that two road segments with similar geographical properties could have similar traffic conditions at the same time of day. To capture the high-level traffic conditions, as demonstrated in Figure 23 C), a city is divided into uniform grids. By projecting the recently received GPS trajectories into these grids, a matrix $M'_G$ is built, with a column standing for a grid and a row denoting a time slot; an entry of $M'_G$ means the number of vehicles traveling in a particular grid and at a particular time slot. Likewise, by projecting the historical trajectories over a long period into the grids, a similar $M_G$ is built, with each entry means the average number of vehicles traveling in a particular grid and at a particular time slot. So, $M'_G$ denotes the real-time high-level traffic conditions in a city and $M_G$ indicates the historical high-level traffic patterns. The difference between the same entries of the two matrices suggests the deviation of current high-level traffic conditions from their historical averages. By combining these matrices, i.e. $X = M'_r || M_r$ and $Y = M'_G || M_G$, a coupled matrix factorization is applied to $X$, $Y$, and $Z$, with the objective function as follows.

$$L(T,R,G,F) = \frac{1}{2}||Y - T(G;G)^T||^2 + \frac{\lambda_1}{2}||X - T(R;R)^T||^2$$
$$+\frac{\lambda_2}{2}||Z - RF^T||^2 + \frac{\lambda_3}{2}(||T||^2 + \left|||R|\right|^2 + ||G||^2 + ||F||^2)$$

where $\|\cdot\|$ denotes the Frobenius norm. The first three terms in the objective function control the loss in matrix factorization, and the last term is a regularization of penalty to prevent over-fitting.

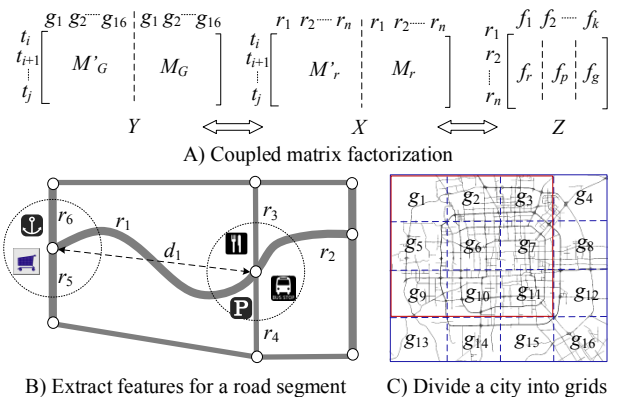
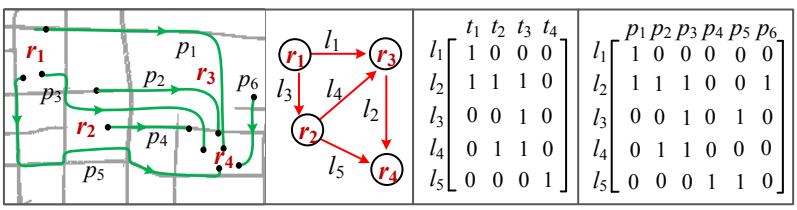



**Figure 23. Estimate traffic conditions based on trajectories**

***Diagnosing traffic anomalies***: Chawla et al. [15] aim to identify the traffic flows that cause an anomaly between two regions. In the methodology, they first partition a city into regions by major roads, building a region graph based on trajectories of taxicabs, as

illustrated in Figure 24 A). A trajectory is then represented by a path on the graph, i.e. a sequence of links between regions, as shown in Fig 24. B). Two matrices are built based on the trajectories and graph. One is a link-traffic matrix $L$, as shown in Fig 24 C), where a row is a link and a column corresponds to a time interval. An entry of $L$ denotes the number of vehicles traversing a particular link at a specific time interval. The other is a link-path matrix $A$, with a row standing for a link and column denoting a path. An entry of $A$ is set to 1 if a particular link is contained in a particular path. Given matrix $L$, they first use a PCA (Principal Component Analysis) algorithm to detect some anomalous links, which were represented by a column vector $b$ with 1 denoting an anomaly detected on the link. Then, the relationship between anomalous links and paths was captured by solving the equation, $Ax = b$, where $x$ is a column vector denoting which paths contribute to the emergency of these anomalies shown in $b$. Using $L_1$ optimization techniques, $x$ can be inferred.




A) From trajectories to path B) Links C) Link traffic matrix D) Link-path matrix

**Figure 24. From trajectories to matrices for detecting anomalies**

## 9.3 From Trajectory to Tensor

A nature extension of the matrix-based transformation is turning trajectories into a (3D) tensor, where the third dimension is added to a matrix so as to accommodate additional information. The goal of the transformation is usually to fill the missing entries (in a tensor) or find the correlation between two objects, like two road segments or gas stations. A common approach to solving this problem is to decompose a tensor into the multiplication of a few (low-rank) matrices and a core tensor (or just a few vectors), based on the tensor's non-zero entries. When a tensor is very sparse, in order to achieve a better performance, the tensor is usually decomposed with other (context) matrices in a framework of collaborative filtering.

Zheng et al. [139][137] extend the generic location-activity research [138] into a personalized one, by adding a user dimension into the original location-activity matrix. As shown in Figure 25, a user-location-activity tensor $A$ is built, with an entry denoting the times that a particular user has performed a particular activity in a particular location. If we can infer the value of every entry, personalized recommendation can be enabled. However, tensor $A$ is very sparse as a user usually visits a few places. Thus, a simple tensor completion method cannot fill its missing entries very well. To address this issue, four context matrices are built based on additional data sources, such as road network and POI datasets, which are not sparse. In addition, these matrices share some dimension with tensor $A$. For instance, tensor $A$ share the *user* dimension with matrix $B$ and the *location* dimension with matrix $E$. Consequently, the knowledge from these matrices can be transferred into the tensor to help completing tensor $A$.

Wang and Zheng et al. [104] proposed a coupled tensor-decomposition-based method to instantly estimate the travel time of a path, based on a sample of vehicles' GPS trajectories.

To model the traffic conditions of the current time slot, they construct a tensor $\mathcal{A}_r \in \mathbb{R}^{N\times M\times L}$, with the three dimensions standing for road segments, drivers and time slots, respectively, based on the GPS trajectories received in the most recent $L$ time slots and the road network data. As shown in Figure 26, an entry $\mathcal{A}_r(i,j,k) = c$ denotes the $i$th road segment is traveled by the $j$th driver with a time cost $c$ in time slot $k$ (e.g., 2-2:30pm). The last time slot denotes the present time slot, combined with the $L$-1 time slots right before it to formulate the tensor. Clearly, the tensor is very sparse as a driver can only travel a few road segments in a short time period. If the missing entries can be inferred based on the values of non-zero entries, we can obtain the travel time of any driver on any road segment in the present time slot.

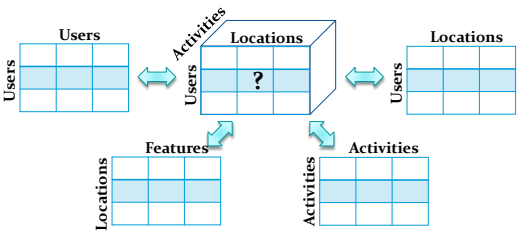



**Figure 25. Recommendation based on trajectories and tensors.**

To this end, another tensor $\mathcal{A}_h$ is built based on the historical trajectories over a long period of time (e.g. one month). $\mathcal{A}_h$ has the same structure as $\mathcal{A}_r$, while an entry $\mathcal{A}_h(i,j,k) = c'$ denotes the $j$th driver's average travel time on the $i$th road segment in time slot $k$ in the history. Intrinsically, $\mathcal{A}_h$ is much denser than $\mathcal{A}_r$, denoting the historical traffic patterns and drivers' behavior on an entire road network. Besides, two context matrices ($X$ and $Y$) are built to help supplement the missing entries of $\mathcal{A}_r$. Matrix $X$ (consisting of $X_r$ and $X_h$) represents the correlation between different time slots in terms of the coarse-grained traffic conditions. This is similar to its correspondence shown Figure 23 C). An entry of $X_r$ denotes the number of vehicles traversing a particular grid in a particular time slot. A row of $X_r$ represents coarse-grained traffic conditions in a city at a particular time slot. Consequently, the similarity of two different rows indicates the correlation of traffic flows between two time slots. $X_h$ has the same structure as $X_r$, storing the historical average number of vehicles traversing a grid from $t_i$ to $t_j$. Matrix $Y$ stores each road segment's geographical features, which are similar to that of Matrix $Z$ shown in Figure 23 A). Later, they decompose $\mathcal{A} = \mathcal{A}_r || \mathcal{A}_h$ with matrices $X$ and $Y$ collaboratively, by optimizing the following objective function.

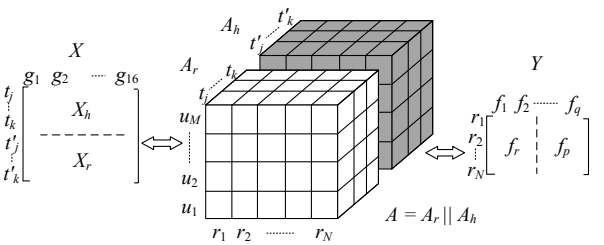



**Figure 26. Travel time estimation using tensor decomposition.**

$$\mathcal{L}(S,R,U,T,F,G) = \frac{1}{2}\|\mathcal{A} - S\times_R R\times_U U\times_T T\|^2 + \frac{\lambda_1}{2}\|X - TG\|^2$$
$$+\frac{\lambda_2}{2}\|Y - RF\|^2 + \frac{\lambda_3}{2}(\|S\|^2 + \|R\|^2 + \|U\|^2 + \|T\|^2 + \|F\|^2 + \|G\|^2);$$

A similar idea was employed by Zhang et al. [132][133] to estimate the queuing time in each gas station throughout a city. The queuing time is further used to estimate the number of vehicles that are being refueled. Specifically, the refueling event are first detected from a taxicab's GPS trajectories based on a stay point-based inference method. Then, as shown in Figure 27, a three dimension tensor $F$ is built, with the first dimension denoting gas stations, second one standing for time of day, and the third for the day of the week. An entry means the average waiting time (detected from taxi trajectories) at a station in a particular day of the week and at a particular time interval. This tensor is intrinsically sparse as we cannot guarantee to have a taxicab being refueled in each station anytime. A context matrix is built, incorporating the geographical features of a station. Intuitively, two gas stations with the similar surrounding environment (including road networks and POIs) and traffic flow could have the similar refueling pattern. The coupled tensor decomposition method mentioned in pervious examples is then applied to the tensor and matrix, filling the missing value in $F$.

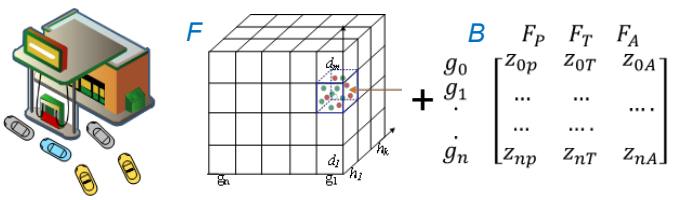



**Figure 27. Estimate the refueling behavior in a gas station**

## 10. MISCELLANEOUS

### 10.1 Public Trajectory Datasets

Collecting data is always the first priority of trajectory data mining. Thanks to researchers in this field, there are quite a few real trajectory datasets that are publicly available:

- *GeoLife Trajectory Dataset* [159]: a GPS trajectory dataset from Microsoft Research GeoLife project [153], collected by 182 users from April 2007 to August 2012. The dataset has been used to estimate the similarity between users [54], which enables friend and location recommendations [154][155]. It was also used by Chen et al. [21] for studying the problem of finding the nearest trajectory to a sequence of query points.
- *T-Drive Taxi Trajectories* [160]*:* A sample of trajectories from Microsoft Research T-Drive project [123], generated by over 10,000 taxicabs in a week of 2008 in Beijing. The full dataset was used to suggest the practically fastest driving directions to normal drivers [124][125], recommend passenger-pickup location for taxi drivers [127][130], enable dynamic taxi ridesharing [65][66], glean the problematic design in a city's transportation network [151], and identify urban functional regions [122].
- *GPS Trajectory with Transportation Labels* [161]*:* Each trajectory has a set of transportation mode labels, such as driving, taking a bus, riding a bike and walking. The dataset can be used to evaluate trajectory classification and activity recognition [149][146].
- *Check-in Data from Location-based Social Networks* [162]*:* The dataset consists of the check-in data generated by over 49,000 users in New York City and 31,000 users in Los Angeles as well as the social structure of the users. Each check-in includes a venue ID, the category of the venue, a timestamp, and a user ID. As the check-in data of a

user can be regarded as a low-sampling-rate trajectory, this dataset has been used to study the uncertainty of trajectories [105] and evaluate location recommendation [5].

- *Hurricane trajectories* [163]: This dataset is provided by the National Hurricane Service (NHS), containing 1,740 trajectories of Atlantic Hurricanes (formally defined as tropical cyclone) from 1851 to 2012. NHS also provides annotations of typical hurricane tracks for each month throughout the annual hurricane season that spans from June to November. The dataset can be used to test trajectory clustering and uncertainty.
- *The Greek truck trajectories* [164]: This dataset contains 1,100 trajectories from 50 different trucks delivering concrete around Athens, Greece. It was used to evaluate trajectory pattern mining task in [34].
- *Movebank animal tracking data* [165]: Movebank is a free, online database of animal tracking data, helping animal tracking researchers to manage, share, protect, analyze, and archive their data.

When needing massive trajectories to test the efficiency of a method, we can generate synthetic trajectories based on traffic generators, e.g. BerlinMod [29] and Thomas-Brinkhoff [10]. There is also a web-based interface, called MNTG [70], which supports the two traffic generators to work on any arbitrary road networks. Ma et al. [66] build a taxi ride request simulator based on the pickup and drop-off points of the real taxi trajectories generated in Beijing. The simulator is used to test the efficiency of a taxi ride sharing service.

### 10.2 Conferences and Journals Concerning Trajectories

Research about trajectory data mining has a wide presence at the following venues:

- General data mining conferences: KDD, ICDM, SDM, PAKDD, and ICML-PKDD.
- General database conferences: ICDE, VLDB, SIGMOD, EDBT and DASFAA.
- General artificial intelligence conferences: IJCAI and AAAI.
- Spatial-data-focusing conferences: ACM SIGSPATIAL GIS, SSTD and MDM.
- Application-driven conferences and workshops: International Conference on Ubiquitous Computing, and the International Workshop on Urban Computing [146].
- Journals and Transactions: IEEE TKDE, ACM TKDD, ACM TIST, VLDB, Data Mining and Knowledge Discovery, KAIS, DKE, Journal on Personal and Ubiquitous Computing. Besides the journals in computer science area, there are many journals in other disciplinary, such as Transportation Research Part C, IEEE Transaction on Intelligent Transportation Systems, and Transportation Record.

### 10.3 Potential Future Direction

In the big data era, a data mining task needs to harness a diversity of data. This is calling for new technology that can unlock the power of knowledge from multiple data sources. Under such a circumstance, how to mine trajectory data together with other data sources is a new challenge. There two approaches towards this goal.

One is to combine trajectories with other data sources to fulfill a data mining task. For example, Zheng et al. [143][148] infer the fine-grained air quality, using trajectories of vehicles, POIs and meteorological data. Fu et al. [31][32] combine human mobility data represented by trajectories with social media and geographical data to rank the potential value of real estates. Yuan et al. [122][130] aim to identify the functional regions in a city based on taxi trajectories, road network data and POIs. Zheng et al. [150] diagnoses the urban noise, using check-in data, traffic, and 311 complaints. The other approach is to use

other sources to enrich a trajectory. For instance, leveraging POIs and road network data, Wang et al. [104] better estimates the travel time of a path based on sparse trajectories.

The new challenge calls for 1) data management techniques that can organize multi-modal data for an efficient retrieval and mining; 2) the cross-domain machine learning methods that can unlock the power of knowledge that cannot be discovered from a single data source; and 3) advanced visualization techniques that can suggest the insights across different sources.

## 11. CONCLUSION

The wide availability of trajectory data has fostered a diversity of applications, calling for algorithms that can discover knowledge from the data effectively and efficiently. This paper surveys the techniques concerned with different stages of trajectory data mining, recapping them by categories and exploring the differences between one another. This paper also suggests the approaches of transforming raw trajectories into other data structures, to which more existing data mining techniques can be applied. This paper provides an overview on how to unlock the power of knowledge from trajectories, for researchers and professionals from not only computer sciences but also a broader range of communities dealing with trajectories. At the end this paper, a list of public trajectory datasets have been given and a few future directions have been suggested.

## ACKNOWLEDGMENT

A small portion of the content of this article is derived from a few chapters of a book, entitled Computing with Spatial Trajectories [157], which was co-edited by Xiaofang Zhou and me. I appreciate the chapter authors of this book: C.-Y. Chow, K. Deng, C. S. Jensen, H. Jeung, J. Krumm, W.-C. Lee, M. F. Mokbel, K. Xie, K. Zheng, G. Trajcevski, Q. Yang, M. L. Yiu, V. W. Zheng, and Y. Zhu.

## REFERENCES

[1] O. Abul, F. Bonchi, and M. Nanni. 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th IEEE International Conference on Data Engineering*. IEEE, 376-385.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases*. VLDB Endowment 29, 81-92.

[3] R. Agrawal, C. Faloutsos, and A. Swami. 1993. *Efficient similarity search in sequence databases*. Springer, 69-84.

[4] H. Alt, A. Efrat, G. Rote, and C. Wenk. 2003. Matching Planar Maps. *Journal of Algorithms* 49, 2, 262–283.

[5] J. Bao, Y. Zheng, and M. F. Mokbel. 2012. Location-based and Preference-Aware Recommendation Using Sparse Geo-Social Networking Data. In *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 199-208.

[6] J. Bao, Y. Zheng, D. Wilkie, M. F. Mokbel. 2015. A Survey on Recommendations in Location-based Social Networks. GeoInformatica 19, 3, 525-565.

[7] R. Bellman. 1961. On the Approximation of Curves by Line Segments Using Dynamic Programming. *Communications of the ACM* 4, 6, 284.

[8] A. R. Beresford, and F. Stajano. 2003. Location privacy in pervasive computing. *IEEE Pervasive Computing* 2, 1, 46–55.

[9] S. Brakatsouls, D. Pfoser, R. Salas and C. Wenk. 2005. On Map-Matching Vehicle Tracking Data. In *Proceedings of the 31st International Conference on Very Large Data Bases*. VLDB Endowment, 853-864.

[10] T. Brinkhoff and O. Str, 2002. A Framework for Generating Network-Based Moving Objects. *Geoinformatica*, 6, 2, 153-180.

[11] H. Cao, N. Mamoulis, and D. W. Cheung. 2005. Mining frequent spatio-temporal sequential patterns. In *Proceeding of the 5th IEEE International Conference on Data Mining*. IEEE, 82–89.

[12] H. Cao, N. Mamoulis, and D.W. Cheung. 2007. Discovery of periodic patterns in spatiotemporal sequences. *IEEE Transactions on Knowledge and Data Engineering* 19, 4, 453-467.

[13] I. V. Cadez, S. Gaffney, and P. Smyth. 2000. A general probabilistic framework for clustering individuals and objects. In *Proceedings of the 6th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 140-149.

[14] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly detection: a survey. *ACM Computing Surveys* 41, 3, 1–58.

[15] S. Chawla, Y. Zheng, and J. Hu. 2012. Inferring the root cause in road traffic anomalies. In *Proceedings of the 12th IEEE International Conference on Data Mining*. IEEE, 141-150.

[16] S. S. Chawathe. 2007. Segment-based Map Matching. *IEEE Symposium Intelligent Vehicles*. IEEE, 1190-1197.

[17] Y. Chen, K. Jiang, Y. Zheng, C. Li, and N. Yu. 2009. Trajectory Simplification Method for Location-Based Social Networking Services. In *Proceedings of ACM SIGSPATIAL Workshop Location-Based Social Networking Services*. ACM, 33-40.

[18] L. Chen, and R. Ng. 2004. On the marriage of lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases*. VLDB Endowment, 792–803.

[19] L. Chen, M.T. Ozsu, and V. Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data*. ACM, 491-502.

[20] Z. Chen, H. T. Shen, and X. Zhou. 2011. Discovering popular routes from trajectories. In *Proceedings of the 27th IEEE International Conference on Data Engineering*. IEEE, 900–911.

[21] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng and X. Xie. 2010. Searching trajectories by locations – an efficient study. In *Proceedings of the 29th ACM SIGMOD International Conference on Management of Data*. ACM, 255-266.

[22] W. Chen, M. Yu, Z. Li and Y. Chen. 2003. Integrated Vehicle Navigation System for Urban Applications. In *Proceedings of International Conference Global Navigation Satellite System*. CGNS, 15-22.

[23] R. Cheng, J. Chen, M.F. Mokbel, and C.Y. Chow. 2008. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *Proceedings of IEEE 24th Conference on Data Engineering*. IEEE, 973-982.

[24] R. Cheng, D.V. Kalashnikov, and S. Prabhakar. 2004. Querying imprecise data in moving objects environments. *IEEE Transaction on Knowledge and Data Engineering* 16, 9.

[25] C.Y. Chow and M.F. Mokbel. 2011. Privacy of Spatial Trajectories. *Computing with Spatial Trajectories*, Y. Zheng and X. Zhou eds. Springer, 109-141.

[26] A. Civilis, C. S. Jensen, J. Nenortaite, and S. Pakalnis. 2005. Techniques for Efficient Road-network-based Tracking of Moving Objects. *IEEE Transactions on Knowledge and Date Engineering* 17, 5, 698-711.

[27] K. Deng, K. Xie, K. Zheng and X. Zhou. 2011. Trajectory Indexing and Retrieval. *Computing with Spatial Trajectories*. Y. Zheng and X. Zhou eds. Springer, 35-60.

[28] D. Douglas and T. Peucker. 1973. Algorithms for the Reduction of the Number of Points Required to Represent a Line or its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10, 2, 112–122.

[29] C. Duntgen, T. Behr and R. H. Guting. 2009. BerlinMOD: A Benchmark for Moving Object Databases. The VLDB Journal, 18, 6, 1335-1368.

[30] T. Emrich, H.P. Kriegel, N. Mamoulis, M. Renz, and A. Z¨ufle. 2012. Querying uncertain spatio-temporal data. In *Proceedings of the 28th IEEE Conference on Data Engineering*. IEEE, 354–365.

[31] Y. Fu, Y. Ge, Y. Zheng, Z. Yao, Y. Liu, H. Xiong, N. J. Yuan. 2014. Sparse Real Estate Ranking with Online User Reviews and Offline Moving Behaviors. In *the Proceeding of the 14th IEEE International Conference on Data Mining*. IEEE, 120-129.

[32] Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng. 2014. Exploiting Geographic Dependencies for Real Estate Appraisal: A Mutual Perspective of Ranking and Clustering. In *Proceeding of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 1047-1056.

[33] S. Gaffney and P. Smyth. 1999. Trajectory clustering with mixtures of regression models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 63-7.

[34] F. Giannotti, M. Nanni, D.Pedreschi, and F. Pinelli. 2007. Trajectory Pattern Mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 330-339.

[35] G. Gid´ofalvi, X. Huang, and T.B. Pedersen. 2007. Privacy-preserving data mining on moving object trajectories. In *Proceedings of the 8th IEEE International Conference on Mobile Data Management*. IEEE, 60-68.

[36] J. S. Greenfeld. 2002. Matching GPS Observations to Locations on a Digital Map. In *Proceedings of the 81st Annual*

*Meeting of the Transportaion Research Board*. 576–582.

[37] J. Gudmundsson and M.V. Kreveld. 2006. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*. ACM, 35-42.

[38] J. Gudmundsson, M.V. Kreveld, and B. Speckmann. 2004. Efficient detection of motion patterns in spatio-temporal data sets. In *Proceedings of the 12th Annual ACM International Symposium on Advances in Geographic Information Systems*. ACM, 250–257.

[39] J. Hershberger and J. Snoeyink. 1992. Speeding up the Douglas-Peucker Line simplification Algorithm. In *Proceedings of International Symposium on Spatial Data Handling*. 134–143.

[40] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. 2010. Achieving guaranteed anonymity in GPS traces via uncertainty-aware path cloaking. *IEEE Transactions on Mobile Computing* 9, 8, 1089–1107.

[41] C. S. Jensen, D. Lin, and B. C. Ooi. 2007. Continuous clustering of moving objects. *IEEE Transaction on Knowledge and Data Engineering* 19, 9, 1161–1174.

[42] H. Jeung, H. Shen, and X. Zhou. 2008. Convoy queries in spatio-temporal databases. In *Proceedings of the 24th IEEE International Conference on Data Engineering*. IEEE, 1457–1459.

[43] H. Jeung, M. L. Yiu, and C. S. Jensen. 2011. Trajectory Pattern Mining," *Computing with Spatial Trajectories*. Y. Zheng and X. Zhou eds. Springer, 143-177.

[44] H. Jeung, M. Yiu, X. Zhou, C. Jensen, and H. Shen. 2008. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment* 1, 1, 1068–1080.

[45] G. Kellaris, N. Pelekis, and Y. Theodoridis. 2009. Trajectory compression under network constraints. In *Proceedings of International Symposiums on Advances in Spatial and Temporal Databases*. 392-398.

[46] E.J. Keogh, S. Chu, D. Hart, and M.J. Pazzani. 2001. An On-Line Algorithm for Segmenting Time Series. In *Proceedings of IEEE International Conference on Data Mining*. IEEE, 289–296.

[47] H. Kido, Y. Yanagisawa, and T. Satoh. 2005. An anonymous communication technique using dummies for location-based services. In *Proceedings of the 3rd International Conference on Pervasive Services*. IEEE, 88-97.

[48] A. Kharrat, I. S. Popa, K. Zeitouni, and S. Faiz. 2008. Clustering Algorithm for Network Constraint Trajectories. *Headway in Spatial Data Handling*. 631-647.

[49] J. Krumm. 2011. "Trajectory Analysis for Driving". *Computing with Spatial Trajectories*, Y. Zheng and X. Zhou eds. Springer, 213-241.

[50] J. Krumm, and E. Horvitz. 2004. LOCADIO: Inferring Motion and Location from Wi-Fi Signal Strengths. In *Proceedings of International Conference on Mobile and Ubiquitous Systems*. IEEE, 4-13.

[51] J. G. Lee, J. Han, and K. Y. Whang. 2007. Trajectory clustering: A partition-and-group framework. In *Proceedings of ACM SIGMOD Conference on Management of Data*. ACM, 593-604.

[52] J. Lee, J. Han, and X. Li. 2008. Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Proceedings of the 24th IEEE Conference on Data Engineering*. IEEE, 140-149.

[53] W.-C. Lee, J. Krumm. 2011. Trajectory Preprocessing. *Computing with Spatial Trajectories*, Y. Zheng and X. Zhou eds., pp.1-31, Springer.

[54] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and M. Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th Annual ACM International Symposium on Advances in Geographic Information Systems*. ACM, 34.

[55] Z. Li, B. Ding, J. Han, and R. Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment* 3, 1-2, 723–734.

[56] Z. Li, J. Lee, X.Li, and J. Han. 2010. Incremental Clustering for Trajectories. *Database Systems for Advanced Applications*. 32-46.

[57] Z. Li, B. Ding, J. Han, R.Kays, and P. Nye. 2010. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1099-1108.

[58] Z. Li, J. Wang and J. Han. 2012. Mining event periodicity from incomplete observations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 444-452.

[59] L. Liao, D. Fox, and H. Kautz. 2004. Learning and Inferring Transportation Routines. In *Proceedings of National Conference on Artificial Intelligence*. 348-353.

[60] S. Liu, K. Jayarajah, A. Misra, R. Krishnan. 2013. TODMIS: Mining Communities from Trajectories. In *Proceedings of*

*the 22nd ACM CIKM International Conference on Information and Knowledge Management*. ACM, 2109-2118.

[61] S. Liu, L. Ni, R. Krishnan. 2014. Fraud Detection from Taxis' Driving Behaviors. *IEEE Transactions on Vehicular Technology*. 63, 1. 464 – 472.

[62] W. Liu, Y. Zheng, S. Chawla, J. Yuan and X. Xie. 2011. Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1010-1018.

[63] Y. Lou, C. Zhang, Y. Zheng, X. Xie, et al. 2009. Map-Matching for Low-Sampling-Rate GPS Trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Geographical Information Systems*. ACM, 352-361.

[64] W. Luo, H. Tan, L. Chen, and M. N. Lionel. 2013. Finding time period-based most frequent path in big trajectory data. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. ACM, 713-724.

[65] S. Ma, Y. Zheng and O. Wolfson. 2013. T-Share: A Large-Scale Dynamic Taxi Ridesharing Service. In *Proceedings of the 29th IEEE International Conference on Data Engineering*, IEEE, 410-421.

[66] S. Ma, Y. Zheng and O. Wolfson. 2015. Real-Time City-Scale Taxi Ridesharing. *IEEE Transactions on Knowledge and Data Engineering*, issue 99. DOI: http://doi.ieeecomputersociety.org/10.1109/TKDE.2014.2334313.

[67] N. Maratnia and R.A.D. By. 2004. Spatio-Temporal Compression Techniques for Moving Point Objects. In *Proceedings of the 9th International Conference on Extending Database Technology*. 765–782.

[68] R.B. McMaster. 1986. A statistical Analysis of Mathematical Measures of Linear Simplification. *The American Cartographer* 13, 2, 103-116.

[69] M. F. Mokbel, C.Y. Chow, and W. G. Aref. 2007. The new Casper: Query procesing for location services without compromising privacy. In *Proceedings of the 23th IEEE Conference on Data Engineering*. IEEE, 1499-1500.

[70] M. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, and S. Yackel. 2014. A Demonstration of MNTG - A Web-based Road Network Traffic Generator. In *Proceeding of the 30th IEEE International Conference on Data Engineering*, IEEE, 1246 - 1249.

[71] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti. 2009. WhereNext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 637-646.

[72] M.E. Nergiz, M. Atzori, Y. Saygin, and B. Guc. 2009. Towards trajectory anonymization: A generalization-based approach. *Transactions on Data Privacy* 2, 1, 47–75,

[73] P. Newson, J. Krumm. 2009. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Geographical Information Systems*. ACM, 336–343.

[74] J. Niedermayer, A. Zufle, T. Emrich, M. Renz, N. Mamouliso, L. Chen, and H. Kriegel. 2014. Probabilistic Nearest Neighbor Queries on Uncertain Moving Object Trajectories. *Proceedings of the VLDB Endowment* 7, 3, 205-216.

[75] W. Y. Ochieng, M. A. Quddus and R. B. Noland. 2004. Map-matching in Complex Urban Road Networks. *Brazilian Journal of Cartography* 55, 2, 1-18.

[76] B. Pan, Y. Zheng, D. Wilkie and C. Shahabi. 2013. Crowd Sensing of Traffic Anomalies based on Human Mobility and Social Media. In *Proceedings of the 21st Annual ACM International Conference on Advances in Geographic Information Systems*. ACM, 334-343.

[77] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng. 2011. On Mining Anomalous Patterns in Road Traffic Streams. In *Proceedings of International Conference on Advanced Data Mining and Applications*. 237-251.

[78] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng. 2013. On Detection of Emerging Anomalous Traffic Patterns Using GPS Data. *Data & Knowledge Engineering*, 357-373.

[79] D.J. Patterson, L. Liao, D. Fox and H. Kaut. 2003. Inferring High-Level Behavior from Low-Level Sensors. In *Proceedings of the 5th International Conference on Ubiquitous Computing*. ACM, 73-89.

[80] J. Pei, J. Han, B. Mortazavi-Asl, and H. Pinto. 2011. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings of the 29th IEEE Conference on Data Engineering*. IEEE, 0215-0215.

[81] D. Pfoser, and C. S. Jensen. 1999. Capturing the uncertainty of moving objects representation. In *Proceedings of International Symposium on Advances in Spatial Databases*. 111-131.

[82] D. Pfoster, C. S. Jensen, and Y. Theodoridis. 2000. Novel approaches to the indexing of moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases*. VLDB Endowment, 395-406.

[83] O. Pink and B. Hummel. 2008. A Statistical Approach to Map Matching Using Road Network Geometry, Topology and Vehicular Motion Constraints. In *Proceedings of the 11th International IEEE Conference on Intelligent Transportation*

*Systems*. IEEE, 862-867.

[84] M. Potamias, K. Patroumpas, and T. Sellis. 2006. Sampling Trajectory Streams with Spatio-Temporal Criteria. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*. IEEE, 275–284.

[85] S. Qiao, C. Tang, H. Jin, T. Long, S. Dai, Y. Ku, and M. Chau. 2010. Putmode: prediction of uncertain trajectories in moving objects databases. *Applied Intelligence* 33, 3, 370–386.

[86] M. A. Quddus, W. Y. Ochieng and R. B. Noland. 2006. A High Accuracy Fuzzy Logic-based Map-Matching Algorithm for Road Transport. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations* 10, 3, 103-115.

[87] K. Richter, F. Schmid, and P. Laube. 2012. Semantic trajectory compression: Representing urban movement in a nutshell. *Journal of Spatial Information Science*, 3–30.

[88] S. Rinzivillo, S. Mainardi, F. Pezzoni, M. Coscia, D. Pedreschi, F. Giannotti. 2012. Discovering the Geographical Borders of Human Mobility. *Künstl Intell*, 26, 3, 253–260.

[89] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu. 2014. Inferring Gas Consumption and Pollution Emission of Vehicles throughout a City. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1027-1036.

[90] R. Song, W. Sun, B. Zheng, and Y. Zheng. 2014. PRESS: A Novel Framework of Trajectory Compression in Road Networks. *Proceedings of the VLDB Endowment* 7, 9, 661-672.

[91] H. Su, K. Zheng, H. Wang, J. Huang and X. Zhou. 2013. Calibrating Trajectory Data for Similarity-based Analysis. In *Proceedings of the 39th International Conference on Very Large Data Bases*. VLDB Endowment, 833-844.

[92] Y. Tao, and D. Papadias. 2001. Efficient historical R-trees. *Scientific and Statistical Database Management*, 223-232.

[93] Y. Tao, and D. Papadias. 2001. Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In *Proceedings of the 27th International Conference on Very Large Data Bases*. VLDB Endowment, 431-440.

[94] Y. Tao, D. Papadias and Q. Shen. 2002. Continuous nearest neighbour search. In *Proceedings of the 28th International Conference on Very Large Data Bases*. VLDB Endowment, 287-298.

[95] L. A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu and J. Han. 2011. Retrieving k-Nearest Neighboring Trajectories by a Set of Point Locations. In *Proceedings of the 12th Symposium on Spatial and Temporal Databases*. Springer, 223-241.

[96] L.A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C. Hung, and W. Peng. 2012. Discovery of Traveling Companions from Streaming Trajectories. In *Proceedings of the 28th IEEE International Conference on Data Engineering*. IEEE, 186 - 197.

[97] L.A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, W. Peng, and T. L. Porta. 2012. A Framework of Traveling Companion Discovery on Trajectory Data Streams. *ACM Transactions on Intelligent Systems and Technology* 5, 1.

[98] M. Terrovitis, and N. Mamoulis. 2008. Privacy preservation in the publication of trajectories. In *Proceedings of the 9th IEEE International Conference on Mobile Data Management*. IEEE, 65–72.

[99] G. Trajcevski, A. N. Choudhary, O. Wolfson, L. Ye, and G. Li. 2010. Uncertain range queries for necklaces. In *Proceedings of the 11th IEEE International Conference on Mobile Data Management*. IEEE, 199–208.

[100] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. F. Cruz. 2009. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM, 874–885.

[101] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. 2004. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems* 29, 3, 463–507.

[102] S. Timothy, A. Varshavsky, A. Lamarca, M. Y. Chen, and T. Chounhury. 2006. Mobility detection using everyday GSM traces. In *Proceedings of the 8th International Conference on Ubiquitous Computing*. ACM, 212-224.

[103] L. Wang, Y. Zheng, X. Xie, and W. Ma. 2008. A Flexible Spatio-Temporal Indexing Scheme for Large-Scale GPS Track Retrieval," In *Proceedings of the 8th IEEE International Conference on Mobile Data Management*. IEEE, 1-8.

[104] Y. Wang, Y. Zheng, and Y. Xue. 2014. Travel Time Estimation of a Path using Sparse Trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 25-34.

[105] L. Wei, Y. Zheng, and W. Peng. 2012. Constructing Popular Routes from Uncertain Trajectories. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 195-203.

[106] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. 2010. Finding Similar Users Using Category-Based Location History. In *Proceedings of the 18th Annual ACM International Conference on Advances in Geographic Information Systems*. ACM, 442-445.

[107] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. 2014. Inferring Social Ties between Users with Human Location History. *Journal*

*of Ambient Intelligence and Humanized Computing* 5, 1, 3-19.

[108] H. Xie, L. Kulik, and E. Tanin. 2010. Privacy-aware traffic monitoring. *IEEE Transactions on Intelligent Transportation Systems* 11, 1, 61–70.

[109] C. Xu, Y. Gu, L. Chen, J. Qiao, and G. Yu. 2013. Interval reverse nearest neighbor queries on uncertain data with markov correlations. In *Proceedings of the 29th IEEE Conference on Data Engineering*. IEEE, 170-181.

[110] X. Xu, J. Han, and W. Lu. 1990. RT-tree: An improved R-Tree indexing structure for temporal spatial databases. In *Proceedings of International Symposium on Spatial Data Handling*. 1040-1049.

[111] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. 2013. Destination Prediction by Sub-Trajectory Synthesis and Privacy Protection against Such Prediction. In *Proceedings of IEEE 29th Conference on Data Engineering*. IEEE, 254-265.

[112] X. Yan, J. Han, and R. Afshar. 2003. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In *Proceedings of the 3rd SIAM International Conference on Data Mining*. IEEE, 166-177.

[113] J. Yang, W. Wang, and P.S. Yu. 2003. Mining asynchronous periodic patterns in time series data. *IEEE Transaction on Knowledge and Data Engineering* 15, 3, 613-628.

[114] J. Yang, W. Wang, and S. Y. Philip. 2001. Infominer: Mining surprising periodic patterns. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 395-400.

[115] J. Yang, W. Wang, and P. S. Yu. 2002. Infominer+: Mining partial periodic patterns with gap penalties. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 725-728.

[116] Y. Ye, Y. Zheng, Y. Chen, J. Feng, and X. Xie. 2009. Mining Individual Life Pattern Based on Location History. In *Proceedings of the 10th IEEE International Conference on Mobile Data Management*. IEEE, 1-10.

[117] B.K. Yi, H. Jagadish, and C. Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th IEEE International Conference on Data Engineering*. IEEE, 201-208.

[118] H. B. Yin and O. Wolfson. 2004. A Weight-based Map Matching Method in Moving Objects Databases. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*. IEEE, 437-410.

[119] J. Yin, X. Chai, and Q. Yang. 2004. High-level goal recognition in a wireless Lan. In *Proceedings of National Conference on Artificial Intelligence*. AAAI, 578–584.

[120] H. Yoon, Y. Zheng, X. Xie, and W. Woo. 2012. Social Itinerary Recommendation from User-generated Digital Trails. *Journal on Personal and Ubiquitous Computing* 16, 5, 469-484.

[121] H. Yoon, Y. Zheng, X. Xie, and W. Woo. 2011. Smart Itinerary Recommendation based on User-Generated GPS Trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Intelligence and Computing*. ACM, 19-34.

[122] J. Yuan, Y. Zheng, X. Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 186-194.

[123] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. 2010. T-Drive: Driving Directions Based on Taxi Trajectories. In *Proceedings of the 18th Annual ACM International Conference on Advances in Geographic Information Systems*. ACM, 99-108.

[124] J. Yuan, Y. Zheng, X. Xie, and G. Sun. 2011. Driving with Knowledge from the Physical World. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 316-324.

[125] J. Yuan, Y. Zheng, X. Xie, and G. Sun. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Transaction on Knowledge and Data Engineering* 25, 1, 220-232.

[126] J. Yuan, Y. Zheng, C. Zhang，X. Xie，G. Sun. 2010. An Interactive-Voting based Map Matching Algorithm. In *Proceedings of the 11th IEEE International Conference on Mobile Data Management*. IEEE, 43-52.

[127] J. Yuan, Y. Zheng, L. Zhang, X. Xie, G. Sun. 2011. Where to Find My Next Passenger? In *Proceedings of the 13th International Conference on Ubiquitous Computing*. ACM, 109-118.

[128] N. J. Yuan, Y. Zheng, and X. Xie. 2012. Segmentation of Urban Areas Using Road Networks. Technical Report MSR-TR-2012-65.

[129] N. J. Yuan, Y. Zheng, L. Zhang, X. Xie. 2013. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE Transaction on Knowledge and Data Engineering* 25, 10, 2390-2403.

[130] N. J. Yuan, Y. Zheng, X. Xie, Y. Wang, K. Zheng and H. Xiong. 2015. Discovering Urban Functional Zones Using Latent Activity Trajectories, *IEEE Transactions on Knowledge and Data Engineering*, 27, 3, 1041-4347.

[131] D. Zhang, N. Li, Z. Zhou, C. Chen, L. Sun, and S. Li. 2011. iBAT: detecting anomalous taxi trajectories from GPS traces.

In *Proceedings of the 13th International Conference on Ubiquitous Computing*. ACM, 99-108.

[132] F. Zhang, D. Wilkie, Y. Zheng, X. Xie. 2013. Sensing the Pulse of Urban Refueling Behavior. In *Proceedings of the 15th International Conference on Ubiquitous Computing*. ACM, 13-22.

[133] F. Zhang, N. J. Yuan, D. Wilkie, Y. Zheng, X. Xie. 2015. Sensing the Pulse of Urban Refueling Behavior: A Perspective from Taxi Mobility. *ACM Transactions on Intelligent Systems and Technology,* 6, 3.

[134] K. Zheng, Y. Zheng, X. Xie, X. Zhou. 2012. Reducing Uncertainty of Low-Sampling-Rate Trajectories. In *Proceedings of the 28th IEEE International Conference on Data Engineering*. IEEE, 1144 – 1155.

[135] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang. 2013. On Discovery of Gathering Patterns from Trajectories. In *Proceedings of the 28th IEEE International Conference on Data Engineering*. IEEE, 242-253.

[136] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, X. Zhou. 2014. Online Discovery of Gathering Patterns over Trajectories. *IEEE Transaction on Knowledge and Data Engineering* 26, 8, 1974 – 1988.

[137] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, Q. Yang. 2010. Collaborative Filtering Meets Mobile Recommendation: A User-centered Approach. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI, 236-241.

[138] V. W. Zheng, Y. Zheng, X. Xie, Q. Yang. 2010. Collaborative Location and Activity Recommendations with GPS History Data. In *Proceedings of the 19th International Conference on World Wide Web.* ACM, 1029-1038.

[139] V. W. Zheng, Y. Zheng, X. Xie, Q. Yang. 2012. Towards Mobile Intelligence: Learning from GPS History Data for Collaborative Recommendation. *Artificial Intelligence*, 184-185, 17-37.

[140] Y. Zheng. 2011. Location-Based Social Networks: Users. *Computing with Spatial Trajectories*, Y. Zheng and X. Zhou, eds. Springer, 243-276.

[141] Y. Zheng. 2012. Tutorial on Location-Based Social Networks. In *Proceedings of the 21st International Conference on World Wide Web.* ACM.

[142] Y. Zheng, L. Capra, O. Wolfson, H. Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM Transactions on Intelligent Systems and Technology* 5, 3, 38-55.

[143] Y. Zheng, X. Chen, Q. Jin, Y. Chen, X. Qu, X. Liu, E. Chang, W. Ma, Y. Rui, W. Sun. 2014. A Cloud-Based Knowledge Discovery System for Monitoring Fine-Grained Air Quality. MSR-TR-2014-40.

[144] Y. Zheng, Y. Chen, Q. Li, X. Xie, W.-Y. Ma. 2010. Understanding transportation modes based on GPS data for Web applications," *ACM Transactions on the Web* 4, 1, 1-36.

[145] Y. Zheng, Y. Chen, X. Xie, W.-Y. Ma. 2009. GeoLife2.0: A Location-Based Social Networking Service. In *Proceedings of the 10th IEEE International Conference on Mobile Data Management.* IEEE, 357-358.

[146] Y. Zheng, S. E. Koonin, O. E. Wolfson. Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing. ACM.

[147] Y. Zheng, Q. Li, Y. Chen, X. Xie. 2008. Understanding Mobility Based on GPS Data. In *Proceedings of the 11th International Conference on Ubiquitous Computing.* ACM, 312-321.

[148] Y. Zheng, F. Liu, H.P. Hsieh. 2013. U-Air: When Urban Air Quality Inference Meets Big Data. In *Proceedings of the* 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 1436-1444.

[149] Y. Zheng, L. Liu, L. Wang, X. Xie. 2008. Learning Transportation Mode from Raw GPS Data for Geographic Application on the Web. In *Proceedings of the 17th International Conference on World Wild Web.* ACM, 247-256.

[150] Y. Zheng, T. Liu, Y. Wang, Y. Liu, Y. Zhu, E. Chang. 2014. Diagnosing New York City's Noises with Ubiquitous Data. In *Proceedings of the 16th ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 715-725.

[151] Y. Zheng, Y. Liu, J. Yuan, X. Xie. 2011. Urban Computing with Taxicabs. In *Proceedings of the 13th International Conference on Ubiquitous Computing.* ACM, 89-98.

[152] Y. Zheng, X. Xie. 2011. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology* 2, 1, 2-19.

[153] Y. Zheng, X. Xie, W.-Y. Ma. 2010. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. *IEEE Data Engineering Bulletin* 33, 2, 32-40.

[154] Y. Zheng, L. Zhang, Z. Ma, X. Xie, W.-Y. Ma. 2011. Recommending friends and locations based on individual location history. *ACM Transaction on the Web* 5, 1, 5-44.

[155] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web.* ACM, 791-800.

[156] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma. 2009. Mining Correlation between Locations Using Human Location History. In

*Proceedings of the 17th Annual ACM International Conference on Advances in Geographic Information Systems*. ACM, 352-361.

[157] Y. Zheng and X. Zhou. 2011. Computing with Spatial Trajectories. Springer.

[158] Y. Zhu, Y. Zheng, L. Zhang, D. Santani, X. Xie, Q. Yang. 2011. Inferring Taxi Status Using GPS Trajectories. Technical Report MSR-TR-2011-144.

[159] GeoLife Data: http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/default.aspx

[160] T-Drive Data: http://research.microsoft.com/apps/pubs/?id=152883

[161] Trajectory with transportation modes: http://research.microsoft.com/apps/pubs/?id=141896

[162] User check-in data: https://www.dropbox.com/s/4nwb7zpsj25ibyh/check-in%20data.zip

[163] Hurricane trajectory (HURDAT): http://www.nhc.noaa.gov/data/hurdat

[164] The Greek Trucks Dataset," http://www.chorochronos.org

[165] Movebank data: https://www.movebank.org/

**Yu Zheng** is a lead researcher from Microsoft Research and a Chair Professor at Shanghai Jiao Tong University. His research interests include big data analytics, spatio-temporal data mining, and ubiquitous computing. Currently, he is leading the research on urban computing in Microsoft Research, passionate about using big data to tackle urban challenges. He has published over 50 referred papers as a leading author at prestigious conferences and journals, such as *KDD, VLDB, UbiComp, and IEEE TKDE*. He received five best paper awards from *ICDE'13* and *ACM SIGSPATIAL'10*, etc. Zheng is currently a member of Editorial Advisory Board of *IEEE Spectrum*, an associate editor of *GeoInformatica* and *IEEE Transaction on Big Data*. He has served as chair on over 10 prestigious international conferences, e.g. the program co-chair of *ICDE 2014* (Industrial Track) and *UIC 2014*. He has been invited to give over 10 keynote speeches at international conferences and forums (e.g. *IE 2014* and *APEC 2014 Smart City Forum*). He is an IEEE senior member and ACM senior member. In 2013, he was named one of the *Top Innovators under 35* by *MIT Technology Review* (*TR35*) and featured by *Time* Magazine for his research on urban computing. In 2014, he was named one of the *Top 40 Business Elites under 40 in China* by *Fortune* Magazine, because of the business impact of urban computing he has been advocating. Zheng is also an Adjunct Professor at Hong Kong Polytechnic University and an Affiliate Professor at Southwest Jiaotong University.