Knowledge-Based Systems 174 (2019) 177–193

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Semantic trajectory compression via multi-resolution synchronization-based clustering

Chongming Gao, Yi Zhao, Ruizhi Wu, Qinli Yang, Junming Shao *

Data Mining Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

ARTICLE INFO

Article history:
Received 6 May 2018
Received in revised form 5 March 2019
Accepted 7 March 2019
Available online 11 March 2019

Keywords:
Trajectory compression
Semantic enrichment
Clustering

ABSTRACT

With the pervasive use of location-aware devices and rapid development of location sensing technology, trajectory data has been generated in diverse fields. How to store and manage these large amounts of data is a non-trivial task and thus trajectory compression has gained increasing attention in recent years. As traditional compression algorithms often treat trajectories as sequences of lines in geometric space, the global statistics and the semantics embedded in trajectories are not well considered. In this paper, inspired by the powerful concept of synchronization, we introduce a new semantic trajectory compression approach to yield multi-resolution trajectory abstractions with semantic enrichment. The basic idea is to introduce a multi-resolution synchronization-based clustering model to produce semantic regions of interest (ROIs) in a hierarchical way. Specifically, by imposing constraints on points with semantic information in the interaction model, all neighboring points with similar semantics will group together automatically. Afterwards, each trajectory is compressed as a sequence of semantic ROIs and is further represented as a hierarchical ROI network. Moreover, we further extend our model on the data stream setting. The extensive experiments on synthetic data and four real-world data sets demonstrate the effectiveness and efficiency of our proposed model.

1. Introduction

In recent years, more and more trajectory data has been generated in diverse fields. The massive amount of information often contains a variety of patterns, and plays an important role in a wide range of applications, such as location recommendation, onboard diagnostics and movement behavior analysis. However, how to store and manage these massive amounts of data is a non-trivial task. To tackle this, trajectory compression has attracted a lot of attention in recent years. Up to now, many trajectory compression algorithms have been proposed, which can be broadly divided into two categories: traditional compression in free space [1–5] and compression with contextual constraints [6–12]. The philosophy of the former class is to use fewer lines or paths to replace the original lines or paths of every trajectory to reduce the information redundancy. Although those methods often have a good compression ratio, the real-world context often cannot be well preserved. For instance, in route-based recommender systems, the recommended route should be constrained by actual roads, instead of simple unconstrained lines. Meanwhile, the compressed points (or lines) should cover the important places of interest (POIs) of cities, e.g., museums, parks, monuments, attractions, restaurants, accommodations and shops, instead of only taking account of the points extracted from raw trajectories [13].

To tackle this issue, a lot of semantic trajectory compression algorithms have been proposed in recent years. Semantic trajectory is a concept that the raw trajectory is enriched with related contextual data [14,15]. The examples include networks extracted from urban geographic infrastructure elements such as streets, points of interest, or activities beyond pure geometry, such as walking, non-walking segments [9] and stay points [12]. Therefore, the basic idea of compressing semantic trajectories is to consider the trajectory with enriched contextual information, and then apply compression techniques to it [13]. However, the performance of these methods is highly dependent on the number of accessible priors. E.g., if the roadmap or POI network is sparse or has a lot of missing values, the representation of compressed trajectories would be largely influenced.

In this paper, we introduce a multi-resolution synchronization-based clustering model, CascadeSync, to provide a new perspective to compress trajectory and preserve the semantic information in compressed trajectories. Moreover, we introduce a simple yet effective way for trajectory compression and retrieval in a stream setting. To evaluate the performance of the proposed method, we conduct experiments on both synthetic data and four

* Corresponding author.
E-mail addresses: chongming.gao@std.uestc.edu.cn (C. Gao), yizhao@std.uestc.edu.cn (Y. Zhao), ruizhi_wu@std.uestc.edu.cn (R. Wu), qinli.yang@uestc.edu.cn (Q. Yang), junmshao@uestc.edu.cn (J. Shao).
URL: https://dm.uestc.edu.cn (J. Shao).

https://doi.org/10.1016/j.knosys.2019.03.006
0950-7051/

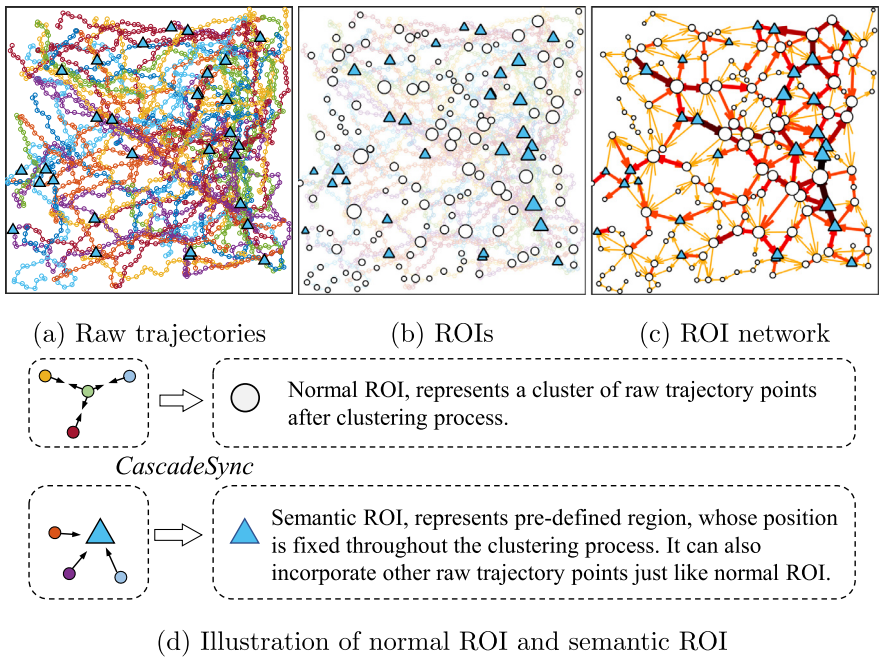


Fig. 1. Illustration of semantic trajectory compression via *CascadeSync*. (a) The 80 trajectories with raw GPS points, and the 30 blue triangles indicate the GPS points with semantic information. (b) The resulting ROIs via constrained synchronization-based clustering. Here each white circle indicates one cluster (i.e., normal ROI), and each blue triangle means one semantic ROI. (c) Trajectory data is finally represented as an ROI network. (d) The normal ROI and semantic ROI.

real-world data sets, and compare the performance to several baselines. We will see that our model has several desirable properties, but let us first illustrate its basic idea.

1.1. Basic idea

Unlike traditional methods which extract key points in each trajectory independently, CascadeSync is a model that aims to find important regions that allow representing all raw trajectory points (i.e., global structure information). These regions of interest, denoted as ROIs, are formed by aggregating surrounding data points based on a synchronization-based clustering model. Due to the powerful concept of synchronization, these GPS points in trajectory data can be clustered from fine-grained to coarse-grained levels. With the derived ROIs, the trajectory data can be represented as a set of multi-resolution ROI-based networks. In addition, if some domain knowledge (e.g., the contextual information for some GPS points) is available, it can be further integrated into the synchronization-based clustering process. For example, assume that there are a bunch of points with semantic information, such as landmark building, road intersections or places where big events occurred. The positions of those important points or regions will be fixed during the synchronization-based clustering process, and they attract neighboring points or regions to group together to form semantic clusters, which are referred to as semantic ROIs. In this way, both global structure information of GPS points and semantic information are well integrated into our compression result.

For illustration, Fig. 1 shows a toy example on a synthetic data set with 80 trajectories. For all these trajectories, suppose that there are thirty points with semantic information, which are indicated as blue triangles in Fig. 1(a). For compression, during the clustering process, these points with semantic information are fixed, and all points (including the fixed points) mutually interacted until all similar points group (i.e., synchronize) together finally. Since the points with semantic information are fixed, during the synchronization process, they attract its neighboring GPS points to group together and thus form semantic regions of interest. For other GPS points, if the neighboring points have no fixed points with semantic information, they are naturally grouped together to form ROIs (without semantic information) (Fig. 1(b)). Fig. 1(d) further illustrates the two resulting ROIs: normal ROI and semantic ROI. Afterwards, with these ROIs, each trajectory can be represented as a sequence of ROIs, and the whole trajectory data is further compressed as an ROI network (Fig. 1(c)). With the ROI network, the global trajectory moving pattern can be well preserved. In addition, due to the desirable properties of synchronization-based clustering, these ROIs, can be further compressed into a higher level and yield a more compact trajectory representation. Building upon the multi-resolution ROI network representation, the global statistics of trajectory data can be extracted and the downstream trajectory mining tasks can be facilitated (e.g., trajectory clustering, classification, pattern mining and outlier detection [13]).

1.2. Contributions

The main contributions of this work are summarized as follows.

- **An intuitive Semantic Trajectory Compression Model.** We propose a semantic trajectory compression model by considering both global trajectory structure information and available contextual information. This method provides a new perspective to compress trajectories with semantics.
- **Multi-resolution Synchronization-based Representation.** Inspired by the powerful concept of synchronization, CascadeSync allows offering a promising way to compress trajectory data. More importantly, it supports a multi-resolution compression and the semantic information can be naturally preserved.
- **Online Trajectory Compression.** An online model is further proposed to support efficient compression and retrieval on trajectory streams, where two operations: *Merge* and *Split* are defined on ROI networks.

- **High Performance.** The empirical experiments show that our proposed method has a strong representative capability and is thousands of times faster than the baseline algorithms. Besides, the desirable semantic information is also well preserved, and a global profile of urban traffic is established.

The rest of paper is organized as follows. Section 2 discusses related work. Section 3 gives some notations and the problem statement. Section 4 elaborates CascadeSync algorithm. Section 5 extends the proposed compression algorithm to stream data. Section 6 presents our extensive experiments. Finally, we conclude our work and discuss future works in Section 7.

2. Related work

According to the comprehensive review of Renso et al. [13], the objectives of trajectory compressions are: (1) to reduce the size of a data set, (2) to ensure that the reduced data set can be computed efficiently, and (3) to ensure that a trajectory from the reduced data set should not deviate from the original one by more than a given threshold. During the past decade, many trajectory compression algorithms have been proposed. In the following, we will give some major works in the existing literature. In addition, we also discuss the works of synchronization-based clustering.

2.1. Traditional trajectory compression

As a reaction to increasing volumes of trajectory data, a lot of trajectory compression technologies have been developed. Sun et al. [16] give a comprehensive overview of trajectory compression algorithms. The early work treats trajectories as sequences of straight lines, so that the compression is cast to a line simplification problem in pure geometric space. Douglas–Peucker algorithm [1] is a classical model, which recursively selects the point whose perpendicular distance is greater than a given error bound until all reserved points meet the condition. Some works [17,18] replace the metric from spatial error to Synchronized Euclidean Distance (SED) to consider the temporal information. For example, Meratnia and Rolf propose the Top-Down Time Ratio (TD-TR) and Open Window Time Ratio (OPW-TR) algorithms with SED metric. Initially, it defines a line segment between the first and the third data point. If the SED from each internal point to the segment is not greater than a given threshold, the algorithm moves one position forward in the sequence. When the threshold is exceeded, the data point that causes the threshold excess is defined as the end position of the current segment, and use the next point as a new start. The process will terminate until all points are handled.

Moreover, Potamias et al. [18] propose two algorithms named Thresholds and STTrace, which are appropriate for online trajectory data compression. The algorithms use the coordinates, speed, and orientation of the current position to calculate a safe area where the next position might be located. If the next incoming position lies in the calculated safe area, it can be ignored. Lee et al. [19] and Soares et al. [20] utilize the concept of minimal description length (MDL) to guide the compression process, which aims to find an optimal trade-off between conciseness and preciseness. The Spatial QUallty Simplification Heuristic (SQUISH) is a method based on the priority queue data structure [21]. Its basic idea is to prioritize the most important points in a trajectory stream. It uses local optimization to select the best subset of points and permanently remove redundant or insignificant points from the original GPS trajectory. Afterwards, Muckell et al. [3] present a new version of SQUISH, called SQUISH-E (Spatial QUality Simplification Heuristic-Extended), which has the flexibility of tuning compression ratio and error.

In addition, some methods are proposed to compress trajectory in an online setting. For instance, Dead Reckoning algorithm estimates the successor point on the go with the current point and its velocity [2]. Liu et al. [4] present the Bounded Quadrant System (BQS) to select points by calculating distance between the new point and a special line maintained online. Lin et al. [5] propose a one-pass strategy that process each point in a trajectory only once to pursue an error bound. However, those online algorithms only output a reduced trajectory data set, which still do not fill the gap between the disordered raw data and the time-aware queries for locations and trajectories. More importantly, these methods are not applicable when introducing constraints such as roadmaps or POI networks in the real world. Although those algorithms can be served as downsampling methods for portable mobile devices to reduce the transmitting cost, they are not suitable for establishing a holistic urban movement profile.

2.2. Semantic trajectory compression

As mentioned by Parent et al. [14], most current application analyses require additional contextual information in the application context. For example, interpreting trajectories of persons within a city requires some knowledge about the features of the city (e.g., map, places of interest). Building upon the city information, spatiotemporal coordinates can be replaced with street and crossing names, or with names of places of interest, such as shops, restaurants, and museums. As a result, the raw trajectories can be replaced by a representation with enriched contextual semantics of the city.

To date, many approaches are proposed to compress trajectories with a variety of contextual information. Schmid et al. [6,7] introduce a novel method called Semantic Trajectory Compression (STC) to replace raw trajectory data with the nodes and edges in road networks enriched with urban transportation information, by using the techniques like navigation research (map matching, place identification) and spatial cognition (wayfinding, generation of directions). However, STC relies on the availability of transport network data for compression, which does not cover every type of human movement (e.g., hiking in national parks). Following the same idea, Liu et al. [8] further propose velocity-based and beacon-based trajectory symbolization, to represent all trajectories using fewer reference points at the same time. Some methods also consider the behavior patterns contained in trajectories. For example, Chen et al. [9] and Zheng et al. [22] partition a trajectory into walking and non-walking segments, using speed, acceleration, and speed change rate, to maintain both the skeleton and semantic meanings of trajectories. Afterwards, they refine the non-walking segments into segments characterized by other transportation modes: bicycle, bus, and driving. They use a combination of techniques, from supervised learning to decision tree inference, and add a postprocessing step to improve the accuracy of the segmentation. Moreover, some works argue that the direction of moving object is important to be preserved [23,24]. Instead of only preserving the positions of trajectories, they argue that the direction information of trajectories is more important for characterizing the structure of trajectory. Another notable work is proposed by Song et al. [10], which presents a framework *PRESS* to split trajectory representation into spatial representation and temporal representation, and then a hybrid spatial compression (HSC) algorithm and an error-bounded temporal compression (BTC) algorithm are proposed, respectively. Besides, there are lots of works emphasizing every point in trajectories should be projected onto roads when mining the trajectory of vehicles generated in urban network, a.k.a. Map-matching [25–28]. For more details, please refer to the excellent surveys [13,29,30]. However, for most existing approaches, they

assume that the semantic information is completely available as prior knowledge. Once the semantic prior knowledge is missing, for example, the roadmap is incomplete, the performance will be greatly affected.

2.3. Synchronization-based clustering

Our work is also highly related to synchronization-based data mining [31]. Synchronization phenomena is prevalent in physical, biological, chemical, and social systems. It comes from the example of typical synchronous flashing of swarms of fireflies in South Asia forests [32]. In the beginning, some fireflies start emitting flashes of light incoherently, but after some time all the fireflies are flashing with the common rhythm with mutual influence. Currently, many synchronization-based models have been proposed in diverse fields [31,33–41]. The extensive Kuramoto model [42,43] is one of the most successful approaches to explore synchronization phenomena. Seliger et al. [33] discuss mechanisms of learning and plasticity in networks of phase oscillators through a generalized Kuramoto model. Arenas et al. [34] apply the Kuramoto model for network analysis, and study the relationship between topological scales and dynamic time scales in complex networks. This analysis provides a useful connection between synchronization dynamics, network topology, and spectral graph analysis. In bioinformatics, Kim et al. [44] propose a strategy to find groups of genes by analyzing the cell cycle specific gene expression with a modified Kuramoto model. Similarly, the co-clusters with correlated genes and conditions are yielded based on two-sided interaction model [40]. Recently, many synchronization-based clustering algorithms [31,35,37,39,45] have been proposed by reformulating the Kuramoto model. For example, Shao et al. [31] propose a new synchronization-based clustering algorithm by introducing local synchronization and minimum description length principle. In contrast to other algorithms, synchronization-based clustering algorithms have many benefits, which allow identifying high-quality clusters and are robust to noisy objects or outliers.

The GPS-based trajectory data is represented by a set of points. It is not a good way to mine trajectory data on these points directly, for different trajectories often have varying lengths and different sampling rates. One intuitive way is to group all these points into many regions with semantics. However, traditional clustering methods are not suitable for this task. For instance, k-means style clustering algorithms need to select the cluster number k explicitly, and the resulting clusters are not evenly distributed, hence the representative error cannot be bounded in ζ. Comparing to k-means based clustering algorithms, synchronization clustering has the advantage that it can automatically produce clusters with a meaningful number. Specifically, the number of clusters is affected by the interaction range ϵ: the number of clusters will decrease with the increase of ϵ. Besides, determining an interaction range is far more intuitive than giving a correct number of clusters. The former can control the clustering process, which motivates us to use ϵ as an indicator for the error bound ζ in a certain situation, i.e., when almost all the points have the representative error smaller than ζ. In contrast, if we use k-means as clustering method for trajectory compression, we can only perform the clustering globally, which is not conducive to controlling the compression process. Another branch of clustering models is those based on DBSCAN [46] or OPTICS [47], which search clusters by looking for the dense area. However, the result clusters are in arbitrary shape, which may violate the real-world region constrains and may lead to meaningless clusters in our compression situation. For example, one main road in the city might be detected as a single cluster, which is not appropriate for trajectory compression. By contrast, synchronization-based clustering approaches tend to produce evenly distributed clusters, which will render meaningful clusters as the compression point on the map. Therefore, the trajectory can be intuitively and reasonably presented by those cluster centers.

Furthermore, synchronization-based clustering algorithms provide a more intuitive way to compress trajectory points since all these points are synchronized together driven by local interaction. Unlike traditional clustering algorithms, synchronization-based clustering approaches view each data object as a phase oscillator and simulate the dynamical behaviors of the objects over time. Through the interaction with similar objects, the phase of an object gradually aligns with its adjacent objects, resulting in a non-linear object movement driven by the local cluster structure. Finally, the objects in a cluster are synchronized together and have the same phase. Therefore, synchronization-based clusters can identify clusters driven by the local data structure, and more importantly, the global data structure can be well-preserved with synchronized objects.

Motivated by those works, in this paper, we view trajectory compression from a dynamic perspective and extract multi-resolution trajectory data abstractions based on synchronization principle. To the best of our knowledge, we are the first to apply the concept of synchronization for data compression.

Table 1
Notations.

Notation	Description
$\mathcal{T}$	Trajectory dataset, each $T_i \in \mathcal{T}$ is a trajectory, defined in (1).
$\mathcal{P}$	GPS point set, each $P_i \in \mathcal{P}$ is a GPS point.
$\mathcal{ROI}$	ROI set, each $ROI_i \in \mathcal{ROI}$ is a node on ROI network.
$\mathcal{E}$	Edge set, each $e_i \in \mathcal{E}$ is an edge of ROI network.
ζ	The representative error bound, defined in (2).
ϵ	The radius of the interaction range, defined in (3).

3. Preliminary

In this section, we will introduce the semantic trajectory compression problem. Before that, we first give some notations used in the following sections, which are listed in Table 1.

3.1. Trajectory and trajectory compression

The most common data format of trajectories, usually collected by GPS devices, are the temporal sequence of latitude/longitude coordinates as follows.

$$T = \{\langle s_1, s_2, \ldots, s_n\rangle \,|s_i = (P_i, t_i)\}, \tag{1}$$

where the $P_i = (x_i, y_i)$ is a (*latitude, longitude*) pair representing a GPS coordinate, representing a sample point on the map. t_i is the time stamp of a point P_i, and n is the length of the trajectory T.

The traditional trajectory compression problem is to use fewer points to replace the original points of each trajectory while most trajectory information needs to be preserved. Given a trajectory T, the compressed trajectory is denoted as $T' = \{\langle P'_1, P'_2, \ldots, P'_m\rangle; , m \leq n\}$. In order to make a compromise between compression ratio and representative error, we introduce the representative error bound to guide the compression.

Definition 3.1 (*Representative Error Bound*)**.** Given a trajectory T and a compression algorithm $\mathcal{A}$ that produces the corresponding compressed trajectory T', the algorithm $\mathcal{A}$ is bounded by the error bound ζ, if for each $P_i \in T$, there exists a point $P'_j \in T'$ with:

$$distance\left(P_i, \mathcal{L}(P'_j, P'_{j+1})\right) \leq \zeta, \tag{2}$$

where $\mathcal{L}(P'_j, P'_{j+1})$ is the straight line that passes the two points (P'_j, P'_{j+1}).

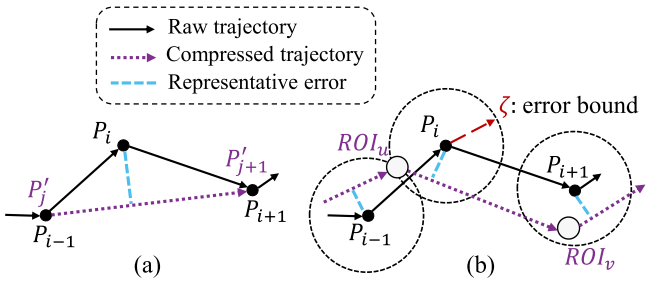


Fig. 2. Illustration of the representative error and the error bound. (a) Traditional representation, where the solid line sequence $\langle P_{i-1}, P_i, P_{i+1}\rangle$ is the original trajectory, and the dash line sequence $\langle P_{i-1}, P_{i+1}\rangle$ is the compressed trajectory. (b) Representation using ROI, where sequence $\langle ROI_u, ROI_v\rangle$ is the compressed trajectory. There is at least one ROI existing in ζ-radius circle of each raw point, so that Eq. (2) can be satisfied).

The distance between the point P_i and $\mathcal{L}$ is shown as dash line with light blue in Fig. 2(a). Note that most trajectory compression algorithms are proposed to be bounded by a given threshold ζ.

3.2. Semantic region of interest

Precisely modeling location identities in trajectories data is of significant importance in location-based service. In many applications, locations represented by GPS coordinates are not adequate to support downstream mining tasks. To extract patterns from human mobility, we require the locations representing high-level activities to describe the profiles of moving objects. After being enriched with domain knowledge, a trajectory can be transformed to sequences such as residence, bus station, and school. As a result, the queries referring to the semantic meanings such as "return objects that stop at the building for half an hour and pass the bus station around eight o'clock" are naturally supported.

Up to now, there are many algorithms devoted to automatically detecting and inferring the significant places for trajectory pattern mining [15,48,49], recommendation [50–52] and route classification [53,54]. However, the basic idea of these works is to simply split the map into multiple grids or group the raw GPS points to clusters, which fail to associate the real-world semantics with the detected places. As mentioned in [7], the semantics should be explicitly defined as a priori. In their work, important urban infrastructures such as hotels, touristic places, and major intersections are defined as nodes and all nodes are further form a network. We refer to those nodes as semantic ROIs. By clustering and map-matching, all raw GPS points are associated with those semantic ROIs.

However, it is noteworthy that the capability of representation and performance of compression highly depend on the quality of predetermined semantic ROIs. When semantic ROIs are insufficient to cover each corner of the city, i.e., if the prior knowledge is sparse (or missing), a profound representative error would be introduced into the compressed results. That is the motivation of our work, where we aim to introduce a new perspective to compress trajectory with limited semantic information.

3.3. Semantic trajectory compression with ROIs

As aforementioned, trajectory compression should make use of real-world semantic information, such as roadmap and POI network, to yield more meaningful compression results. Without loss of generality, the semantic information can be expressed by the ROIs, which are a set of fixed regions on the map with specific semantics. The task of semantic trajectory compression is to find an algorithm $\mathcal{A}$, which is error bounded by ζ. For each raw trajectory $T \in \mathcal{T}$, apply $\mathcal{A}$ to generate a sequence of ROIs to represent T as $T' = \{\langle ROI_1, ROI_2, \ldots, ROI_m\rangle, m \leq n\}$.

In order to satisfy the error bound condition defined in (2), the representation radius, i.e., the distance between point P_i and corresponding ROI, should be bounded in ζ. In other words, for each point there should be at least one ROI in its ζ-radius circle. Fig. 2(b) illustrates a trajectory segment $\langle P_i - 1, P_i, P_i + 1\rangle$ and its compressed version $\langle ROI_u, ROI_v\rangle$.

4. CascadeSync: A multi-resolution clustering model for trajectory compression

In this section, we present CascadeSync, a new semantic trajectory compression framework built upon multi-resolution constrained synchronization-based clustering.

4.1. Region of interest detection via synchronization-based clustering

Typically, a synchronization-based clustering algorithm needs three definitions to simulate a dynamic clustering process: First, a parameter ϵ specifying the interaction range among objects, second, the interaction model for clustering, and finally, a stopping criterion to terminate dynamic clustering. Our approach follows and extends the synchronization-based clustering underlying the algorithm Sync, presented and discussed in full detail in [55].

Definition 4.1 (ϵ-*Range Neighborhood*)**.** Given a GPS data set $\mathcal{P} \subset \Re^n$, the ϵ-range neighborhood of a GPS point $p \in \mathcal{P}$, denoted as $N_\epsilon(p)$, is defined as:

$$N_\epsilon(p) = \{q \big| dist(p, q) \leq \epsilon\}, \tag{3}$$

where $dist(p, q)$ is a metric distance function. Euclidean distance is used in this study.

Definition 4.2 (*Interaction Model*)**.** Let p be a GPS point on the map. With an ϵ-range neighborhood interaction, the dynamics of the value of the point p is defined as:

$$p(t+1) = p(t) + \frac{1}{|N_\epsilon(p)|} \cdot \sum_{q \in N_\epsilon(p)} \sin(q(t) - p(t)), \tag{4}$$

where $sin(x)$ is the coupling function, applying to every dimension of vector x. $p(t+1)$ is the renewal position of $p(t)$ during the dynamic clustering, $t \in \{0, \ldots, T\}$ denotes the iteration step. Note all dimensions are normalized to $[0, \pi/2]$.

Definition 4.3 (*Cluster Order Parameter*)**.** The cluster order parameter r is used to terminate the dynamic clustering by investigating the degree of local synchronization, which is defined as:

$$r(t) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|N_\epsilon(p(t))|} \sum_{q \in N_\epsilon(p)} \mathbf{e}^{-\|q(t) - p(t)\|} \tag{5}$$

The dynamic clustering terminates when $r(t)$ converges, which indicates local phase coherence. At this moment, all cluster points have the same location.

For synchronization-based dynamic clustering, each point is viewed as a phase oscillator and has its own phase (feature vector) at the beginning. As time evolves, each point interacts with its ϵ-range neighborhood according to the interaction model (Eq. (4)). As illustrated in Fig. 3(a), each point interacts with its neighborhood points, and finally all locally similar points are synchronized together and form clusters. For example, the point p_i^l is influenced by its neighbors and finally forms ROI_y^l together with other points in the same region. Meanwhile, since potential

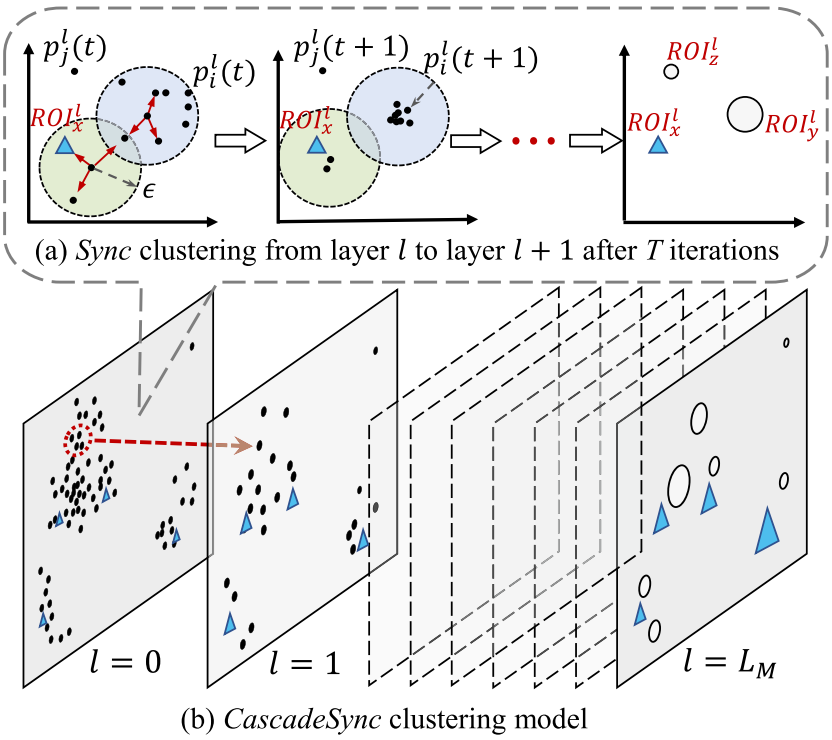


Fig. 3. Illustration of CascadeSync algorithm. The blue triangles are semantic ROIs, which stay fixed throughout the interaction process. The dots on layer $l=0$ are raw GPS points, and the circles on layer $l \geq 1$ are ROIs generated by points on the last layer.

outliers do not interact with other points during the dynamic clustering, they maintain their original values and thus are easily identified. These outliers are also treated as ROIs since they are important to describe some distinctive trajectories. For instance, p_j^l has no near point to interact with, and it is finally represented as ROI_z^l.

The salient feature of synchronization-based clustering is its dynamic property. During the process of interactions, the value of each dimension of a given point changes in a non-linear way driven by the local data structure, and finally the feature vectors of points in a cluster will become the same. More importantly, the derived synchronized ROIs can be viewed as new points to form a new data set. The new data set well preserves the original data structure. Therefore, the powerful concept of synchronization supports a natural hierarchical clustering.

4.2. Multi-resolution network modeling with semantic enrichment

It is important to note that with derived ROIs, the new data set can be further compressed with synchronization-based clustering at a higher level. Therefore, for our trajectory data, we extend Sync to multi-resolution data representation, which is called CascadeSync. The basic idea is quite intuitive. For the first step, we cluster all trajectory data points with a small interaction range ϵ, which usually results in a large number of small-size clusters. Since points in the same cluster have synchronized together, we can use the synchronized point to characterize the whole cluster objects. As a result, a new data set including all these synchronized points can be generated, and cluster again with a much larger interaction range ϵ. However, for a new data set, since each point in the new data set characterizes the different number of points in the previous layer of data, the interaction model should be reformulated to consider the weight of each point, which is defined as follows.

$$p^l(t+1) = p^l(t) + \frac{1}{\sum_{q^l \in N_{\epsilon^l}(p^l)} w_{q^l}} \cdot \sum_{q^l \in N_{\epsilon^l}(p^l)} w_{q^l} \sin(q^l(t) - p^l(t)), \quad (6)$$

where the weight w_{q^l} is the number of points that are represented by the synchronized point q^l on the layer $l-1$. And ϵ^l is the interaction range of layer l, which is manually set by the user according to the application at hand. In the study, we initialize ϵ to be $0.005 \times (L+W)/2$, and add ϵ by $0.005 \times (L+W)/2$ with the increase of number of layer l, where L and W are the length and width of the map.

For illustration, Fig. 3(b) gives a toy example. In fact, CascadeSync not only supports a hierarchical data representation, but also speeds up the synchronization process. The reason is that with the small interaction range, CascadeSync is easier to converge. Although more clusters are generated, they are viewed as new objects and can be further clustered efficiently.

Semantic Enrichment. In previous sections, we focus on the synchronization-based trajectory compression. However, as mentioned in the introduction section, the semantic information is an important property, which needs to be considered for trajectory compression. In real-world scenarios, the semantic information is usually embodied by a collection of fixed points or regions on the map, such as the intersections and turning points on the roadmap, which are drawn as light blue triangles in Fig. 3.

There is an intuitive way to plug those predefined semantic ROIs in our CascadeSync model. Specifically, we make the positions of those ROIs be fixed during the interaction process. Since each point interacts with its neighboring points, and thus all surrounding GPS points will move to these fixed points to form semantic ROIs. If there are no fixed points for some GPS points, near GPS points will synchronize together to form normal ROIs. After all raw GPS points are clustered into normal ROIs or semantic ROIs, each trajectory is represented as a new path with these resulting ROIs. To further explore the statistical information of trajectories contained in ROIs, we define the hierarchical ROI network.

Definition 4.4 (*Hierarchical ROI Network*)**.** Given a trajectory data set, the ROI network is represented as a multi-layer graph $G(\mathcal{V}, \mathcal{E})$

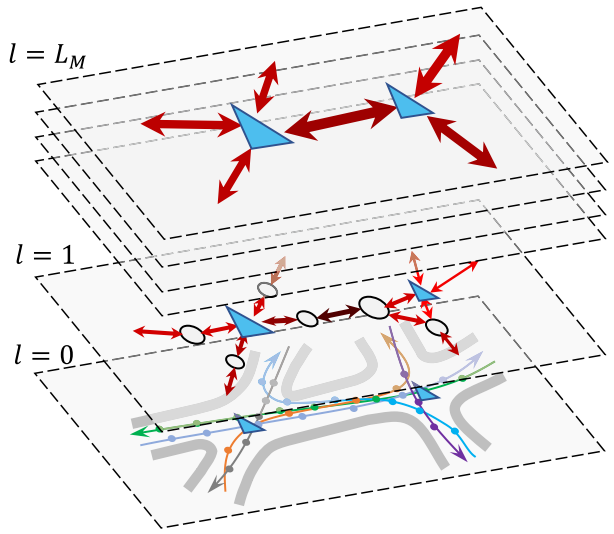


Fig. 4. Illustration of hierarchical ROI network, where the size of ROI is proportional to the point weight. The width and color shade of edges are proportional to the number of trajectories passing through the two ending ROIs.

Algorithm 1: Semantic Trajectory Compression via CascadeSync

Input : Trajectory dataset $\mathcal{T}$; Semantic ROI set $\mathcal{ROI}_s$
Output : Hierarchical ROI network;
Parameter: Initial value ϵ_0; Incremental change $\triangle\epsilon$; Maximal layer $\mathcal{L}_M$

$\mathcal{P}_0 = \mathcal{T} \bigcup \mathcal{ROI}_s$; ▷ The union of raw GPS points set and semantic ROI set.
$\mathcal{P}_0$ = Norm ($\mathcal{P}_0$); ▷ Normalized each dimension to $[0, \pi/2]$.
$layer = 0; \epsilon = \epsilon_0$;
while $layer \leq \mathcal{L}_M$ **do**
  $\mathcal{P}_{layer+1}$ = Sync($\mathcal{P}_{layer}, \mathcal{ROI}_s, \epsilon$); ▷ Fig. 3
  $layer = layer + 1; \epsilon = \epsilon + \triangle\epsilon$;
end
regions of interest set: $\mathcal{ROI} = \mathcal{P}_{layer}$;
ROI network edge set $\mathcal{E}$ = ROINetworkConstructor ($\mathcal{ROI}, \mathcal{T}$); ▷ Fig. 4, 5
Function Sync($\mathcal{P}, \mathcal{ROI}_s, \epsilon$):
  $t = 0$;
  while *TRUE* **do**
    foreach *point* $p_i(t) \in \mathcal{P}$ AND $p_i(t) \notin \mathcal{ROI}_s$ **do**
      Search its ϵ-range neighbors $N_\epsilon(p_i(t))$ with Eq. (3);
      foreach *neighbors* $q(t) \in N_\epsilon(p_i(t))$ **do**
        Compute $p_i(t+1)$ with Eq. (4);
      end
    end
    Compute the cluster order parameter $r(t)$ with Eq. (5);
    if $r(t)$ converges **then**
      return $\{p_1(t), \cdots, p_N(t)\}$;
    end
    $t = t + 1$;
  end
end

with every layer being $G^l(\mathcal{V}^l, \mathcal{E}^l)$, where the node set $\mathcal{V}^l$ contains all ROIs on the layer l. $\mathcal{E}^l$ is edge set of layer l. And there is an edge $e^l_{uv} \in \mathcal{E}^l$ connecting nodes (ROI^l_u, ROI^l_v) if there exists any GPS coordinate pair (P_i, P_j) as a trajectory segment, with P_i represented by ROI^l_u and P_j represented by ROI^l_v on layer l.

We illustrate a hierarchical ROI network in Fig. 4, where the trajectory data is visualized from fine-grained to rough-grained level. The ROI network provides a compact representation to characterize a trajectory. Each trajectory could be expressed as a temporal sequence of ROIs on a given ROI network. In addition, from each ROI, all trajectories that passed this region are recorded. Therefore, a wide range of statistical information, e.g., visiting time distribution, staying time distribution and moving direction, can be extracted from the ROI. For instance, we illustrate a toy example in Fig. 5, in which the raw data and one layer of the ROI network are shown. In Fig. 5(a), a specific trajectory T_{33} is selected, from one layer of ROI network shown in Fig. 5(b), T_{33} is compressed and represented by four ROIs $\langle ROI_1, ROI_2, ROI_4, ROI_5\rangle$. Besides, the basic statistical information of ROI_2 is shown. The traffic patterns of this region can be evaluated from the passing time distribution. Furthermore, some downstream mining tasks, such as trajectory retrieval and frequent pattern mining, could work on such representation directly.

4.3. Time complexity analysis

The time complexity of CascadeSync is $\mathcal{O}(L \times T \times N_l \log N_l)$, T is the time steps in each round and L is number of layers in CascadeSync. Usually, L is small with $L \leq 20$ in practice. N_l is the number of points, which decreases exponentially over the layer l. Finally, the pseudocode of our approach is given in Algorithm 1.

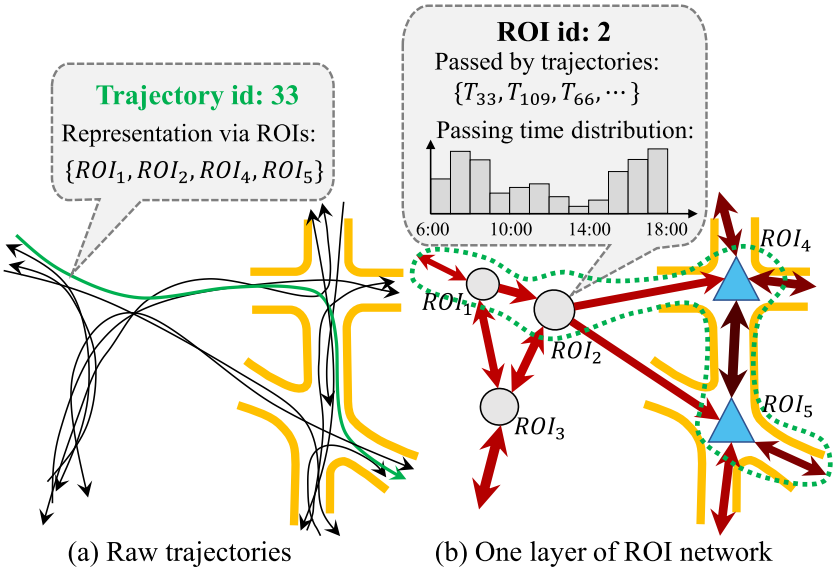


Fig. 5. The ROI network in an incomplete roadmap. Here a specific trajectory T_{33} is selected and compressed to four ROIs: $\langle ROI_1, ROI_2, ROI_4, ROI_5\rangle$. The green dash line encloses the four ROIs on the ROI network, and the basic statistics of ROI_2 is illustrated.

5. Trajectory stream compression and retrieval

Since trajectory data is constantly generated and collected in overwhelming speed in the real world, it is unrealistic to represent and store all trajectories in one ROI network. Here, we extend our model on the stream setting, which aims to compress and store the trajectories over time with a simple strategy.

5.1. Trajectory stream compression

The trajectories of human and vehicles are collected continuously. In order to store and analyze the huge amount of trajectory data, we divide the whole data set into several chunks over time, e.g., a week or a month per chunk. Fig. 6 illustrates the ROI networks generated by CascadeSync in three periods. The partitioning strategy is simple: first, we select an appropriate window size for time partition, which is dependent on the real world applications. For each window, we apply CascadeSync to cluster all GPS points in this time window. For example, the GPS points of trajectory T_1, T_2 and part of T_3 and T_4 are collected and fed in CascadeSync algorithm to yield a hierarchical ROI network G_1. The rest GPS points of T_3 and T_4 are generated in the next period and thus represented by G_2 and G_3, respectively.

To keep the integrity of the trajectory and support trajectory visualization, separated ROI networks should be capable of re-assembling appropriately. To this end, we define two operations, *Merge* and *Split*, on the ROI networks to support the mixture and split of data.

ROI Networks Aggregation. When the time range for a given query spans more than one time window, we need to merge all corresponding ROI networks together. Here we introduce an operation on ROI networks, called *Merge*, which aims at integrating two ROI networks into a single ROI network. Fig. 7 illustrates how it works. For each layer L_i, the initial points, i.e., original GPS points or ROIs generated at layer L_{i-1}, are mixed together from two sources, which is shown in Fig. 7(a). The ROIs distributed in a small region (with a radius of ϵ) should be merged into one ROI, as they are located in the same region. For example, as the red points and blue points are mixed at layer L_{i-1}, the representative region ROI_A and ROI_B should be merged to yield ROI_C at layer L_i. Naturally, the operation can be implemented by applying synchronization-based clustering on the layer, as shown in Fig. 7(b). After applying the *Merge* operation on all layers, the two ROI networks are merged into one ROI network, and this procedure can be repeated to handle multiple ROI networks.

However, there is an inevitable error introduced in the merging process. This is the regret term in online learning which measures the difference between the loss of online model and the offline model. We will analyze it in the experimental section.

ROI Network Decomposition. Similar to the *Merge* operation, an ROI network should be capable of breaking up into several ROI networks, when the application requires a fine-grained temporal range. So we introduce another operation: *Split* on ROI network. The strategy of splitting an ROI network is relatively simple: For layer L_i, some points (light blue points in 8(a)) need to be removed from data on existing ROI network. Hence, we remove them directly and update the indexing of network structure so that the generated ROI will no longer contain those points (8(b)). At first glance, the removal of points should result in the change of the positions of ROIs, but we choose to keep the original ROI fixed. It is because that any adjustment of positions would yield a new representative error, besides, the amount of removal points is relatively small comparing to original data points. And experiments show the operation *Split* can keep the representative error stable.

Algorithm 2: Spatiotemporal Query in Trajectory Stream

Input : Derived ROI networks in all time periods $\mathcal{G} = \{G_1, G_2, \cdots, G_T\}$;
Query $\mathcal{Q} = \{(P_1, \triangle t_1), (P_2, \triangle t_2), \cdots, (P_n, \triangle t_n)\}$;
Output : Trajectory set $\mathcal{T}_q$;

```
foreach point P_i ∈ Q do
    Identify the ROI network set G_i;
    foreach ROI network G_k ∈ G_i do
        Get the trajectory set T_k^i;
    end
    T^i = Union(T_1^i, ..., T_K^i);
end
T_q = Intersect(T^1, T^2, ..., T^n);
```

5.2. Trajectory stream retrieval

The basic requirement for the trajectory database is to support trajectory retrieval effectively so that we are able to conduct the complex analysis on trajectory data. According to the work of Deng et al. [56], there are three kinds of relationship in trajectory queries: (1) Trajectories and points. For example, find all trajectories within 500 m of a gas station between 9:00*pm*-9:30*pm*. (2) Trajectories and regions, e.g., find the region which is passed by the specific trajectories between 9:00*pm*-9:30*pm*. (3) Trajectories and trajectories, e.g., find travelers who may take a similar path in the coming 30 min.

Here we demonstrate our model naturally supports the first two types of query mentioned above, i.e., given at least one point or region, the trajectories near to the point(s)/region(s) in specific time could be easily retrieved. Formally, we define the spatiotemporal query on a trajectory stream as follows.

Definition 5.1 (*Spatiotemporal Query*)**.** For each time, a spatiotemporal query $\mathcal{Q} = \{(P_1, \triangle t_1), (P_2, \triangle t_2), \ldots, (P_n, \triangle t_n)\}$, which consists of a set of distinct points and their time ranges, is to ask for the set of trajectories $\mathcal{T}_q$ that pass or surround all the points in query $\mathcal{Q}$ during the required period. $P_i = (x_i, y_i)$ is a two-dimensional data point or the center of a given region, and $\triangle t_i = (t_i^s, t_i^e)$ denotes the starting time t_i^s and ending time t_i^e for trajectory passing nearby the point P_i, respectively.

Assume a set of ROI networks $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ have been generated for all continuously time windows. The spatiotemporal query can be readily done by applying the following three steps:

Step 1: For each point $P_i \in \mathcal{Q}$, identify the ROI network set $\mathcal{G}_i$ that contains the time range $\triangle t_i = (t_i^s, t_i^e)$ for the query.

Step 2: For each network $G \in \mathcal{G}_i$ generated in Step 1, extract all trajectories that pass through the point P_i, which is denoted as $\mathcal{T}_k^i$. Therefore, take the union operation to get all trajectories on all networks as $\mathcal{T}^i = \texttt{Union}(\mathcal{T}_1^i, \ldots, \mathcal{T}_K^i)$.

Step 3: To obtain the final results under all query conditions in $\mathcal{Q}$, perform the interaction operation on the retrieval results from Step 2, i.e., $\mathcal{T}_q = \texttt{Intersect}(\mathcal{T}^1, \mathcal{T}^2, \ldots, \mathcal{T}^n)$.

The time complexity for the query is $\mathcal{O}(nT)$, where n is the number of points/regions in one query, and T is the number of trajectories. With the hash technique, the process can be accelerated further. The pseudocode is shown in Algorithm 2.

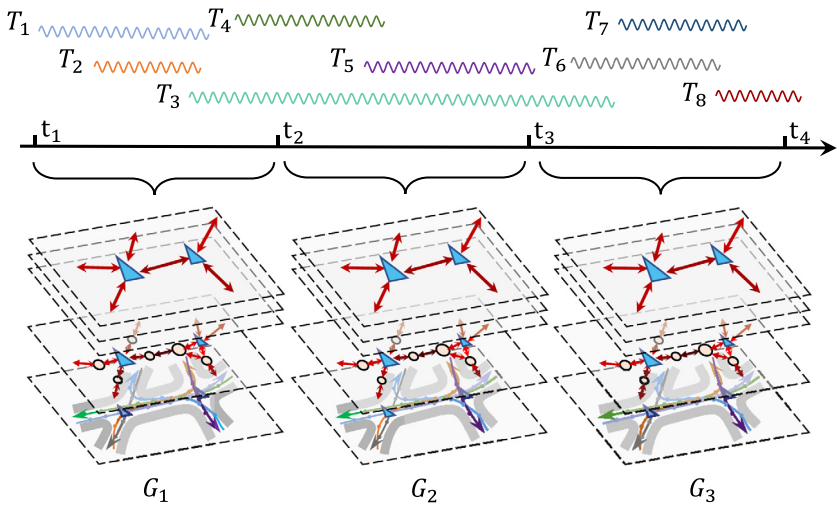
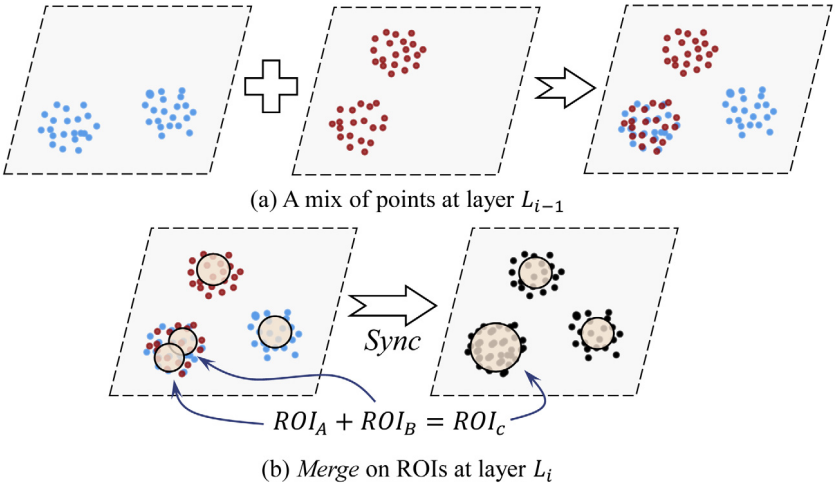


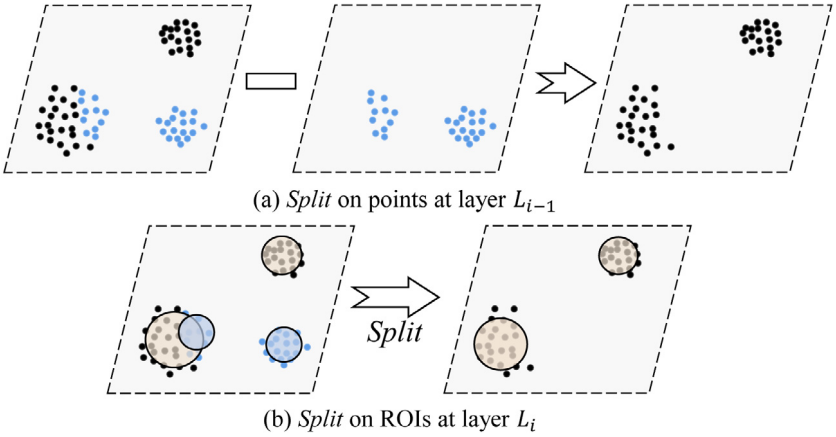
Fig. 6. The maintenance of ROI networks in trajectory streams.



(a) A mix of points at layer L_{i-1}

(b) *Merge* on ROIs at layer L_i

Fig. 7. The merge of ROIs on two hierarchical ROI networks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) *Split* on points at layer L_{i-1}

(b) *Split* on ROIs at layer L_i

Fig. 8. ROI split on a hierarchical ROI network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Experiment

In this section, we perform experiments to evaluate the performance of our algorithm CascadeSync. To reveal the robust property of CascadeSync in different semantic conditions, we design experiments on data sets with and without predefined semantic ROIs, respectively. We also conduct experiments to prove the desirable properties of CascadeSync and also compare

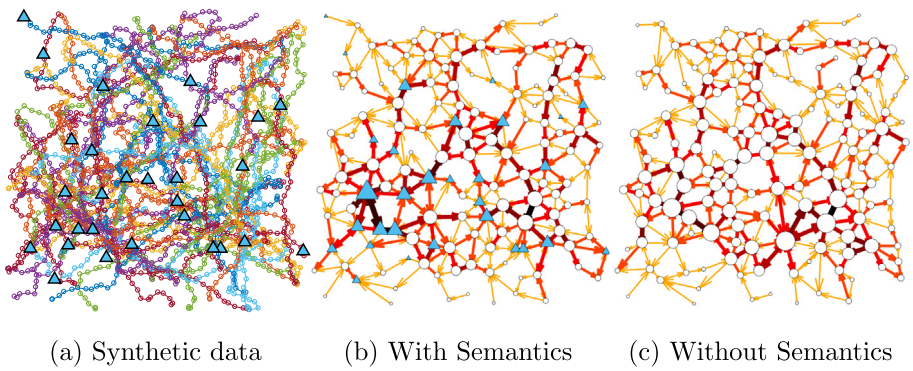


Fig. 9. Trajectory compression on 100 synthetic trajectories. (a) Raw data set. (b) The resulting one-layer ROI network ($\epsilon = 5$ m) with 30 semantic points. (c) The resulting one-layer ROI network ($\epsilon = 5$ m) without 30 semantic points. Here the size of ROI is proportional to the point weight. The width and color shade of edges are proportional to the number of trajectories passing through the two ending ROIs.

the performance of our algorithm with some baseline algorithms. Moreover, the experiments of the spatiotemporal query are also given to show the effectiveness of our model in the trajectory retrieval task.

6.1. Experimental setup

6.1.1. Synthetic data

To prove the concept, we generate random trajectories in a rectangle of 100 × 100 m with the random starting point and ending point, using a probabilistic path planning algorithm [57], which is implemented in MATLAB 2016b as `robotics.PRM` class. Moreover, we randomly select 30 points as semantic ROIs. Fig. 9(a) shows the generated data set with 100 random trajectories, where the triangles with light blue are selected as semantic ROIs.

6.1.2. Real-world data

We evaluate our proposed method on four trajectory data sets: Geolife,[1] T-drive,[2] Atlantic Hurricanes[3] and Migration of Argentine Barn Swallows.[4] Geolife is a trajectory data set collected from daily human life by Zheng et al. [58–60]. T-drive contains trajectories generated by urban taxi, which is collected by Yuan et al. [61,62]. Both Geolife and T-Drive are trajectory data set in Beijing, and the substantial distinction between them lies in the different sampling rates. About 91 percent of trajectories from Geolife are logged in a dense representation, e.g., every 1–5 s or every 5–10 m per point, while trajectories from T-Drive are sampled in a very low frequency, like 2-5 min per point. Atlantic Hurricanes, collected by NHC (National Hurricane Center), records the track of Atlantic hurricane in 1851–2016. And Argentine Barn Swallows records the tracks of migrating birds in South America. All data sets contain the sampling GPS points described by the latitude, longitude and time stamp. We summarize the statistics of the four data sets in Table 2.

6.1.3. Semantic enrichment

To generate the ground truth of semantic information, we downloaded the urban roadmap of Beijing from OpenStreetMap.[5] The roadmap is expressed by nodes and edges. We randomly select 1000 intersections, i.e., the nodes with degree greater than two, as the fixed semantic ROIs on the T-Drive and Geolife data set. Here we do not introduce semantic ROI on Hurricane and Migration data since it is meaningless to define the semantic point or region on the overland area.

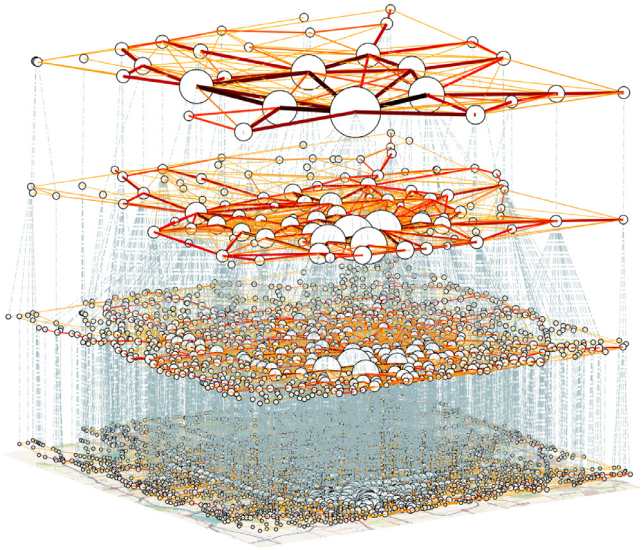


Fig. 10. Hierarchical ROI network generated from raw trajectories on Geolife data. Here ϵ is set as (200 m, 500 m, 1000 m, 3000 m) from bottom layer to top layer, respectively.

6.1.4. Evaluation metrics

Generally, traditional trajectory compression algorithms are evaluated by the compression ratio, which is defined as follows.

$$\text{Compression ratio} = 1 - \sum_i \frac{|T_i'|}{|T_i|} \times 100\%, \tag{7}$$

where each $T_i \in \mathcal{T}'$ is one trajectory of raw trajectory set $\mathcal{T}'$, and T_i' is its compressed representation. $|T_i|$ denotes the number of points or regions of the trajectory T_i. However, the performance of our model cannot be fully evaluated by this metric since there is a trade-off between compression ratio and semantic information preservation. To measure the semantic information preservation, we also compute the representative error η, which is defined as the distance between a true point p and its compressed point p'. The mean value and standard deviation of all points can be recorded. Besides, we also compare the runtime of all algorithms on the four real-world data sets.

[1] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.
[2] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/.
[3] http://www.nhc.noaa.gov/data/hurdat/.
[4] https://www.datarepository.movebank.org/handle/10255/move.655.
[5] https://www.openstreetmap.org.

Table 2
Statistics of four well-known real-world data sets.

Data set	#Point	#Trajectory	[Longitude, Latitude range]	[Width × Length] (m)
Geolife	24,876,978	18,670	[116.194, 116.553, 39.751, 40.033]	[31, 024 × 31, 368]
T-Drive	6,969,481	8768	[116.194, 116.553, 39.751, 40.033]	[31, 024 × 31, 368]
Hurricane	49,682	1830	[−113.708, 44.713, 5.221, 74.725]	[7.728 × 10.860] × 10^6
Migration	4544	9	[−69.75, −33.80, −39.61, 16.80]	[6.272 × 3.915] × 10^6

6.1.5. The selection of comparison methods

The performance of semantic trajectory compression methods is not convenient to compare directly, because they are proposed to enrich semantic information to the new representations of trajectory data. To demonstrate the effectiveness and efficiency of CascadeSync, we use five well-known trajectory compression algorithms as baselines.

Douglas–Peucker [1]: is the most notable line simplification algorithm, which uses a greedy strategy to examine all points between the first and last point until the maximum spatial deviation is within the error bound ζ. The worst-case runtime is $\mathcal{O}(n^2)$.

Douglas–Peucker-SED [17,18]: is a transformation of original Douglas–Peucker algorithm. It takes full consideration of spatiotemporal characteristics by replacing perpendicular distance with Synchronized Euclidean Distance (SED), which measures the distance between two points at identical time stamps. The worst-case runtime complexity is also $\mathcal{O}(n^2)$.

Dead Reckoning [2,21]: is an online compression model, which estimates every successor position through the current position and velocity. By computing the deviation between the estimated position and the true position, the point can be left off from the compressed trajectory if the deviation is less than the error bound. The time complexity of Dead Reckoning is $\mathcal{O}(n)$.

Squish [3,21]: compresses each trajectory by removing points of the lowest priority from the priority queue until it achieves the target compression ratio or the error bound. The time complexity of Squish is $\mathcal{O}(\log n)$.

Traclus-MDL [19]: is a compression technique that can only achieve constant compression ratio. It uses minimum description length (MDL) [63] to jointly model the complexity and representative capacity of compressed trajectories so as to derive the optimal characteristic points as compression results. The time complexity of this model is $\mathcal{O}(n)$.

We compare the compression ratio with these baseline algorithms, and the compression time of all models is listed. Furthermore, the evaluation of enriched semantic information in the new trajectory representations of CascadeSync is further given.

6.2. Proof of concepts

6.2.1. Visualization of ROI network

For illustration, we first visualize the ROI networks generated on different data sets. For the sake of conciseness, we only visualize the multi-layer ROI network of Geolife data in Figs. 10 and 11, which are the results with and without using 1000 fixed intersections in CascadeSync, respectively. Besides, one layer on the hierarchical ROI network of all data sets is illustrated in Figs. 9 and 15. From those figures, we can see our model is intuitive and effective.

6.2.2. Representative error analysis

Unlike traditional methods that compress each trajectory independently, our synchronization-based clustering model CascadeSync compresses trajectory globally, and the representative error can be bound by ζ with high probability. Here we start with the experiment to explore how well the trajectories can be represented by ROIs, as well as the relationship between the error bound ζ and the interaction range ϵ.

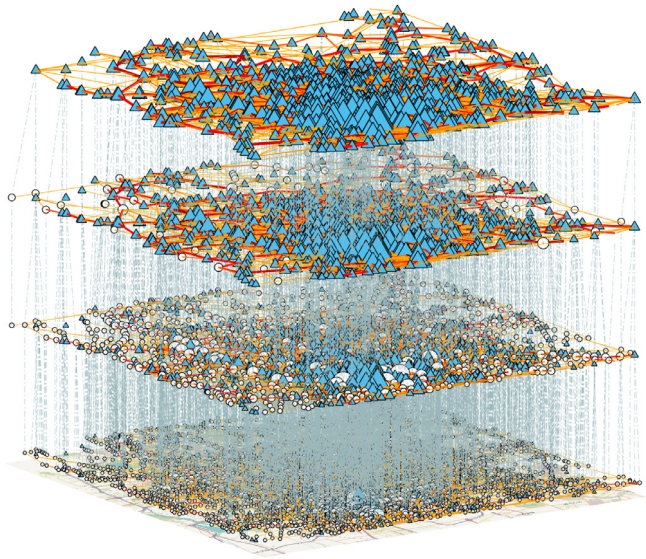


Fig. 11. Hierarchical ROI network generated from raw trajectories with additional 1000 road intersections (which are represented by blue triangles) on Geolife data. Here ϵ is set as (200 m, 500 m, 1000 m, 3000 m) from bottom layer to top layer, respectively.

Without loss of generality, we conduct this experiment on Geolife data set. By applying CascadeSync model, 24,876,978 GPS points are represented as ROIs from fine-grained to coarse-grained resolution. For illustration, the interaction range ϵ varies evenly from 200 m to 2000 m for ten times, which is around 6/1000 to 60/1000 of the length of the map. The average representative error of all points can be calculated at different levels by increasing the interaction range of each layer in CascadeSync. The results are illustrated as box plot in Fig. 12. The horizontal red line in box is the median value, the upper boundary of blue box is the third quartile points, the upper end of black dash line indicates the maximal value, and the red line beyond the maximal value are considered as outliers.

CascadeSync without Semantic ROIs. As for the experiment without predefined semantic ROIs, the result is plotted in Fig. 12(a). We can observe that the maximum is proportional to the interaction range ϵ, and is approximately equal to ϵ. Actually, there are 97.64% points whose representative errors are smaller than ϵ. In other words, if we set the error bound $\zeta = \epsilon$, the result of CascadeSync without semantic ROIs would be bounded by ζ with probability greater than 0.9764.

CascadeSync with Semantic ROIs. By introducing intersections as fixed semantic ROIs, the result is plotted in Fig. 12(b). Different from the previous one, the representative error increases when the interaction range ϵ increases gradually until it reaches 800 m, and then it remains constant with the increase of ϵ. In this experiment, if we set error bound $\zeta = \epsilon$, the result would be bounded by ζ, with probability greater than 0.9971.

Until now, we have shown that our method CascadeSync is bounded by the error bound $\zeta = \epsilon$, with very high probabilities. Moreover, by introducing predefined semantic ROIs, the representative error could be further reduced. The reason is quite

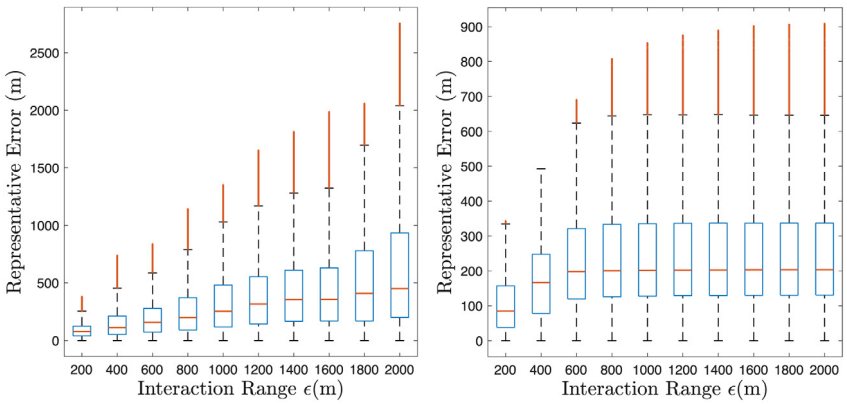


Fig. 12. Relationship between the representative error and the interaction range. The horizontal red line in the box is the median value, the upper boundary of blue box is the third quartile points, the upper end of black dash line indicates the maximal value, and the red line beyond the maximal value is considered as the outliers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

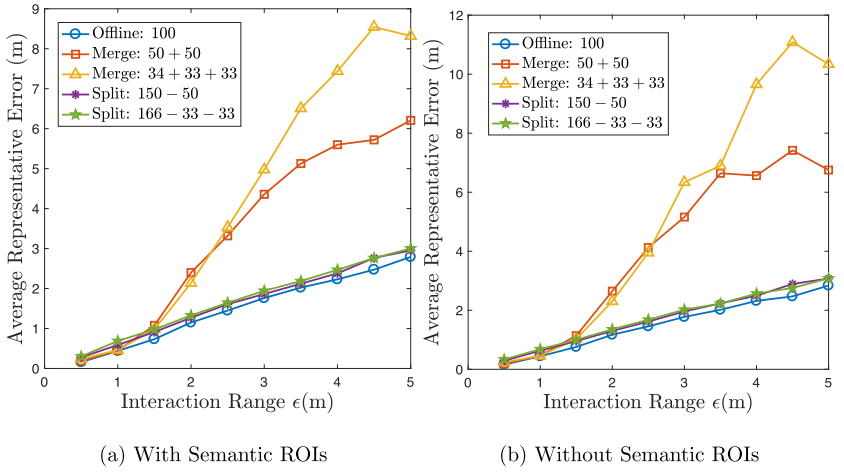


Fig. 13. Average representative error generated by offline/online compression.

intuitive: more semantic information will lead to more dense representative ROIs. However, the compression ratio will also increase, and we will show this in the next experiment.

6.2.3. ROI Network online merge/split analysis

As the amount of trajectory grows, it is impractical to represent all trajectory in a single ROI network. We have developed a window-based method to break the trajectory data depending on data recency. However, if there is a request to present or visualize data with a given specific time period, which the existing window size fails to satisfy, the operation *Merge/Split* would be applied. The online operations on the ROI networks reduce the runtime which is needed by constructing an ROI network from scratch, although the additional error will be introduced. Now we conduct experiments to evaluate the error introduced by the online *Merge/Split* operation(s).

200 random trajectories and 30 fixed semantic ROI are generated with aforementioned synthetic data generation method. 100 trajectories from the 200 trajectories are randomly selected as bases. We design five schemes to generate an ROI network by applying CascadeSync model: (1) ROI network is generated by the 100 bases in one attempt (the offline version); (2) merge two ROI networks which are generated by first half and second half of the bases individually (i.e., 100 trajectories using (50 + 50) merge operation); (3) merge three ROI networks that are generated from three parts of bases (i.e., 100 trajectories using (34 + 33 + 33) merge operation); (4) split an ROI network that is generated from 50 non-base trajectories from an ROI network and make the rest ROI network contains exactly the 100 base trajectories (i.e., 100 trajectories using (150 - 50) split operation); (5) split two ROI networks and make the result ROI network contains exactly the 100 base trajectories (i.e., 100 trajectories using (166 - 33 - 33) split operations).

Fig. 13 illustrates the representative errors with the five schemes. From the figure, we can observe that the operation *Split* always remains the representative error as the same level as the offline version, which means *Split* would not introduce additional error and thus be reliable for various applications. However, the operation *Merge* will introduce additional error as the interaction range ϵ exceeds 1.5 m (15/1000 of the length of the map), and it grows with the increase of ϵ. Another observation is the more times the *Merge* conducted, the larger the additional error introduced since more trajectories are summarized at a higher level. Therefore, the *Merge* operation is only suitable for the applications which are not sensitive to the representative error such as visualization.

Furthermore, to get a fine-grained view of how *Merge* and *Split* affect the representative error. We merge the ROI network with 50 trajectories by adding one single trajectory per time, until all 100 base trajectories are added in an existing ROI network. Similarly, we split the ROI network with 150 trajectories by removing one single trajectory per time until the existing ROI network

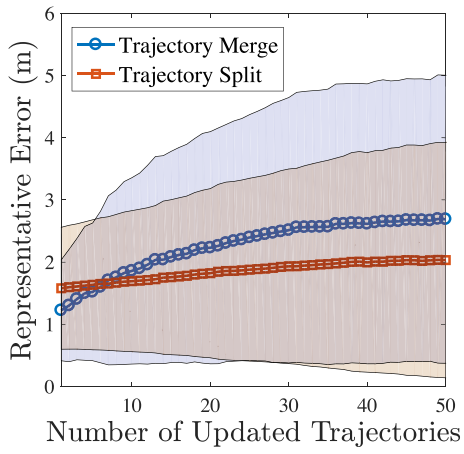


Fig. 14. The relationship between representative error and the number of merged/split trajectories on one layer of ROI network with the interaction range $\epsilon = 2.5$ m. The mean values and area bounded by one standard deviation are shown.

Table 3
Runtime records on the real-world data sets. The result is measured in second (s).

	DP	DP-SED	DR	Squish	Traclus-MDL	CascadeSync
Geolife	6100.42	36621.08	14525.53	14712.10	6357.91	**15.38**
Tdrive	2288.57	35422.91	5101.41	5289.05	2344.51	**7.51**
Hurricane	16.46	49.80	30.98	33.54	13.78	**0.42**
Migration	1.79	12.09	4.40	4.44	2.28	**0.10**

contains exactly the 100 base trajectories. The mean value and area bounded by one standard deviation of error curve on the layer with the interaction range $\epsilon = 2.5$ m are drawn in Fig. 14. From the result, we can see that the *Split* operation can keep the representative error on a low level, and *Merge* operation also will not introduce the large error as in Fig. 13. This is because the trajectories are gradually added into the existing ROI network, it can maintain the correct representation. This suggests our model can be extended to compress and represent trajectory data on the fly.

6.3. Evaluation of compression

In this section, we compare the compression ratio with baseline algorithms to show the representative capability of CascadeSync. To compare them fairly, all comparing algorithms are bounded with the same error bound ζ. We set $\zeta = \epsilon$ in CascadeSync, since we have illustrated that the results are convincing.

As aforementioned, CascadeSync is employed twice on Geolife and T-Drive data, where the difference is whether the predefined intersections (i.e., semantic information) is used or not. By varying the error bound ζ in a reasonable range, we calculate the compression ratios of four data sets and plot the results in Fig. 16. Here, only some results are visualized in Fig. 15.

We find that the scales of compression ratio are different on these four data sets. For example, Geolife and T-drive both are trajectories in Beijing. However, the sampling rate of Geolife is hundreds of times larger than that of T-drive. Therefore, there is much more redundancy contained in Geolife, which results in a very high compression ratio. The ROI networks in Fig. 15(a)–(d) also show the differences. The ROI network of T-drive is more complicated than Geolife, which indicates there are many gaps on trajectories.

Nonetheless, the compression ratios of the four models are comparable in each data set. Douglas–Peucker is superior to other models, and our CascadeSync in free space inclines to exceed other models with the increase of the error bound. It is important to note that Traclu-MDL presents a flat line. Since the model does not take the error bound as a parameter, it cannot control the compression ratio. Meanwhile, CascadeSync with semantic information is not good at compression ratio, which is due to the semantic ROIs that are not allowed to move or reduce on the map. This phenomenon can also be revealed by ROI networks in Fig. 15(a–d): by increasing the error bound ζ, there exist fewer normal ROIs, and thus the road intersections gradually become the dominant ROIs on the map.

The experiment has shown the excellent compression ratio and representative capability of CascadeSync. Now we show the most salient feature of CascadeSync by comparing the runtime with other algorithms. The runtime of six algorithms on four real-world data is reported. Each algorithm runs ten times on each data set, and the average runtime is reported in Table 3. Note that the runtime in Table 3 and the compression ratio in Fig. 16 are measured simultaneously in the same experimental setting. CascadeSync is thousands of times faster than the other five methods. This is because the traditional models compress trajectory one by one, thus the runtime is proportional to the points in data set. Our model works globally, so the number of points would reduce exponentially so that the compression process would be accelerated.

6.4. Spatiotemporal trajectory query

The derived ROI network is a compact representation of trajectory data. It well integrates both global information and special

Table 4
Spatiotemporal queries on Geolife data.

Spatiotemporal queries			Number of trajectories		
No.	Location	Time range	$\epsilon = 500$ m	$\epsilon = 1000$ m	$\epsilon = 3000$ m
1	Peking University, Jinrong Street.	2008-11-28, 2009-01-07	3	8	51
2	Wangfujing, Beijing West Railway Station.	2009-02-14, 2009-03-20	4	18	44
3	Tiananmen Square, Beijing Railway Station, National Stadium.	2009-01-27, 2009-02-25	0	2	9
4	Zhongguancun, Guangximen, Jianguomen.	2008-06-18, 2008-06-22	1	2	12
5	Tiantan Park, Forbidden City, Beihai Park.	2008-04-02, 2009-11-23	15	139	578

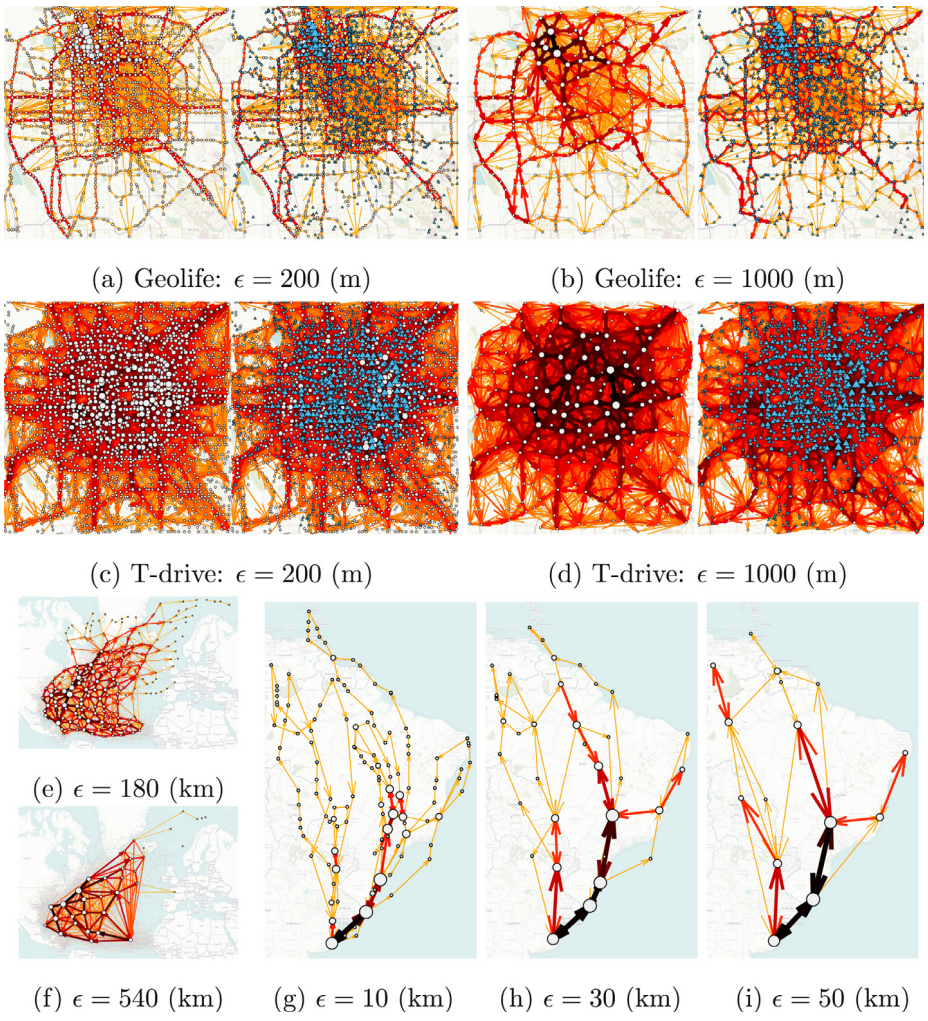


Fig. 15. The ROI networks on different data sets with different error bounds. In each subfigure, the left-hand side ROI network is generated without semantic ROIs, and the right-hand side ROI network is generated with 1000 random selected intersections. The size of ROI is proportional to the node weight (i.e., the number of raw GPS coordinates each ROI represented). The edge width and color shade are proportional to the transportation flow of two ending ROIs.

semantic information, which facilitates the downstream applications. In this section, we take the spatiotemporal retrieval task as an example. Given a set of landmarks or regions in a city as well as the time periods, we retrieve the trajectories passing all those regions during the corresponding time periods. Since our multi-resolution ROI network has recorded the passing trajectories with visiting time, trajectories can be easily retrieved by applying Algorithm 2 on the existing ROI networks.

For simplicity, we only illustrate the results of queries on Geolife data (see Table 4). Since most trajectories are located around Microsoft Research Asia (MSRA) and campuses of some universities, the data is sparse with respect to the whole Beijing city. For illustration, here we only consider three ranges (500 m, 1000 m and 3000 m, respectively), and give the first retrieval result from 2008-11-28 to 2009-01-07 in Fig. 17. There are only 3 trajectories passing the campus of Peking University and Jinrong Street when the ϵ-range is set as 500 m. The number of resulting trajectories grows, intuitively, by increasing the ϵ-range.

7. Conclusion

In this paper, we propose a synchronization-based semantic trajectory compression algorithm CascadeSync, by representing a given trajectory data set as a multi-resolution ROI network. Building upon the concept of synchronization, CascadeSync allows yielding multi-resolution trajectory abstractions (i.e., hierarchical ROI network) with and without available semantic information. More importantly, the abstracted trajectory representation well preserves the global trajectory information, and the semantic can be well integrated. For trajectory streams, we develop a simple yet efficient window-based method to represent trajectories on multiple ROI networks according to data recency. The experiments on both synthetic data and four real-world data sets have demonstrated its effectiveness and efficiency, and show its superiority to other five baseline methods.

Based on the appropriate representations, trajectory data can be further exploited to mine the underlying human mobility patterns [13,14]. The possible directions are: (1) to construct comprehensive user profiles and understand human behavior by modeling his/her traveling history. Meanwhile, if the social network data and the behavior data (such as phone contacting records, trading records or rating data) are available, the evolution of friendships can be investigated based on their daily co-traveling records. (2) Since the POIs of the city have been modeled in our trajectory representation as a network, the relationship between nearby POIs can be explored. On the one hand, the similarity can be redefined based on user's visiting

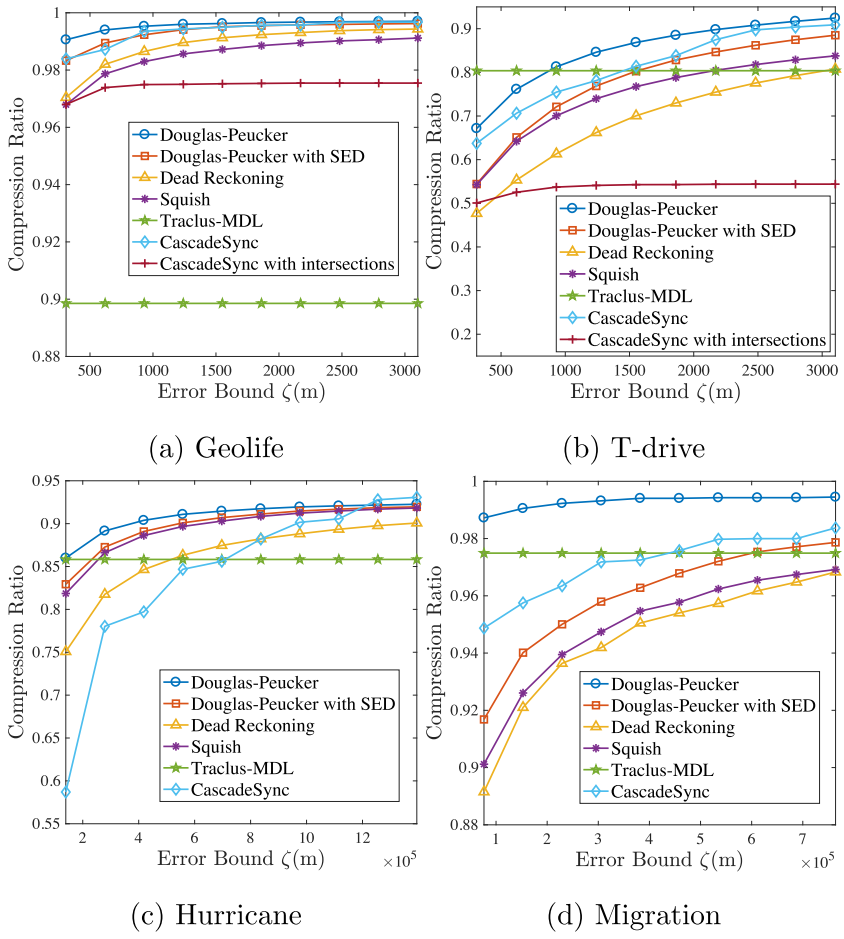
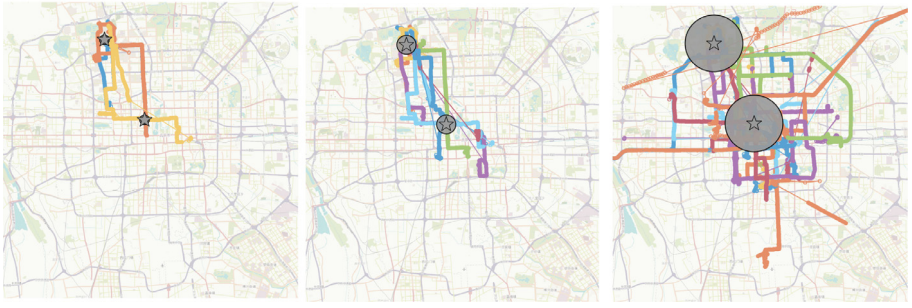


Fig. 16. Compression ratio with varying representative error bound.



(a) $\epsilon = 500m$, #Traj.=3 (b) $\epsilon = 1000m$, #Traj.=8 (c) $\epsilon = 3000m$, #Traj.=51

Fig. 17. Retrieval result of spatiotemporal trajectory query with different searching ranges. Two locations are selected, which are Peking University and Jinrong Street. The time range is from 2008-11-28 to 2009-01-07. The radius of circle indicates the range ϵ.

records instead of geographic distance. On the other hand, the relationship among nearby POIs can be researched. For example, is it a symbiosis or competition between a tea shop and the coffee shop next to it? (3) It helps understand the mechanism of POI recommender systems. Actually, recommendations in location-based social network are prevailing in recent research works [64]. However, most works only focus on pursuing the prediction performance, ignoring the interpretation of models [65]. By exploring the user behavior and social relationships in our hierarchical ROI network, CascadeSync allows enhancing the explainability of recommendation results.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61403062, 61433014, 41601025), Science-Technology Foundation for Young Scientist of SiChuan Province, China (2016JQ0007), Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of

China (161062), National key research and development program, China (2016YFB0502300) and the Sichuan Provincial Soft Science Research Program, China (2017ZR0208).

References

[1] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica 10 (2) (1973) 112–122.

[2] G. Trajcevski, H. Cao, P. Scheuermanny, O. Wolfsonz, D. Vaccaro, On-line data reduction and the quality of history in moving objects databases, in: Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Sccess, ACM, 2006, pp. 19–26.

[3] J. Muckell, P.W. Olsen, J.-H. Hwang, C.T. Lawson, S. Ravi, Compression of trajectory data: a comprehensive evaluation and new approach, GeoInformatica 18 (3) (2014) 435–460.

[4] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, R. Jurdak, Bounded quadrant system: Error-bounded trajectory compression on the go, in: Data Engineering, ICDE, 2015 IEEE 31st International Conference on, IEEE, 2015, pp. 987–998.

[5] X. Lin, S. Ma, H. Zhang, T. Wo, J. Huai, One-pass error bounded trajectory simplification, Proc. VLDB Endow. 10 (7) (2017) 841–852.

[6] F. Schmid, K.-F. Richter, P. Laube, Semantic trajectory compression, Adv. Spat. Temporal Databases (2009) 411–416.

[7] K.-F. Richter, F. Schmid, P. Laube, Semantic trajectory compression: Representing urban movement in a nutshell, J. Spat. Inf. Sci. 2012 (4) (2012) 3–30.

[8] K. Liu, Y. Li, J. Dai, S. Shang, K. Zheng, Compressing large scale urban trajectory data, in: Proceedings of the Fourth International Workshop on Cloud Data and Platforms, ACM, 2014, p. 3.

[9] Y. Chen, K. Jiang, Y. Zheng, C. Li, N. Yu, Trajectory simplification method for location-based social networking services, in: Proceedings of the 2009 International Workshop on Location Based Social Networks, ACM, 2009, pp. 33–40.

[10] R. Song, W. Sun, B. Zheng, Y. Zheng, PRESS: A novel framework of trajectory compression in road networks, Proc. VLDB Endow. 7 (9) (2014) 661–672.

[11] R. Gotsman, Y. Kanza, Compact representation of GPS trajectories over vectorial road networks, in: International Symposium on Spatial and Temporal Databases, Springer, 2013, pp. 241–258.

[12] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, W.-Y. Ma, Mining user similarity based on location history, in: Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, 2008, p. 34.

[13] D.C. Renso, D.S. Spaccapietra, D.E. Zimnyi, Mobility Data: Modeling, Management, and Understanding, Cambridge University Press, New York, NY, USA, 2013.

[14] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M.L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, et al., Semantic trajectories modeling and analysis, ACM Comput. Surv. 45 (4) (2013) 42.

[15] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 330–339.

[16] P. Sun, S. Xia, G. Yuan, D. Li, An overview of moving object trajectory compression algorithms, Math. Probl. Eng. 2016 (2016).

[17] N. Meratnia, A. Rolf, Spatiotemporal compression techniques for moving point objects, in: International Conference on Extending Database Technology, Springer, 2004, pp. 765–782.

[18] M. Potamias, K. Patroumpas, T. Sellis, Sampling trajectory streams with spatiotemporal criteria, in: 18th International Conference on Scientific and Statistical Database Management, IEEE, 2006, pp. 275–284.

[19] J.-G. Lee, J. Han, K.-Y. Whang, Trajectory clustering: a partition-and-group framework, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007, pp. 593–604.

[20] A. Soares Júnior, B.N. Moreno, V.C. Times, S. Matwin, L.d.A.F. Cabral, GRASP-UTS: an algorithm for unsupervised trajectory segmentation, Int. J. Geogr. Inf. Sci. 29 (1) (2015) 46–68.

[21] J. Muckell, J.-H. Hwang, V. Patil, C.T. Lawson, F. Ping, S. Ravi, SQUISH: an online approach for GPS trajectory compression, in: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, ACM, 2011, p. 13.

[22] Y. Zheng, Y. Chen, Q. Li, X. Xie, W.-Y. Ma, Understanding transportation modes based on GPS data for web applications, ACM Trans. Web (TWEB) 4 (1) (2010) 1.

[23] C. Long, R.C.-W. Wong, H. Jagadish, Direction-preserving trajectory simplification, Proc. VLDB Endow. 6 (10) (2013) 949–960.

[24] C. Long, R.C.-W. Wong, H. Jagadish, Trajectory simplification: on minimizing the direction-based error, Proc. VLDB Endow. 8 (1) (2014) 49–60.

[25] A. Civilis, C.S. Jensen, S. Pakalnis, Techniques for efficient road-network-based tracking of moving objects, IEEE Trans. Knowl. Data Eng. 17 (5) (2005) 698–712.

[26] R. Gotsman, Y. Kanza, A dilution-matching-encoding compaction of trajectories over road networks, GeoInformatica 19 (2) (2015) 331–364.

[27] I.S. Popa, K. Zeitouni, V. Oria, A. Kharrat, Spatio-temporal compression of trajectories in road networks, GeoInformatica 19 (1) (2015) 117–145.

[28] Y. Dong, D. Pi, Novel privacy-preserving algorithm based on frequent path for trajectory data publishing, Knowl.-Based Syst. (2018).

[29] Y. Zheng, Trajectory data mining: an overview, ACM Trans. Intell. Syst. Technol. 6 (3) (2015) 29.

[30] J.D. Mazimpaka, S. Timpf, Trajectory data mining: A review of methods and applications, J. Spat. Inf. Sci. 2016 (13) (2016) 61–99.

[31] J. Shao, X. He, C. Böhm, Q. Yang, C. Plant, Synchronization-inspired partitioning and hierarchical clustering, IEEE Trans. Knowl. Data Eng. 25 (4) (2013) 893–905.

[32] J.A. Acebrón, L.L. Bonilla, C.J.P. Vicente, F. Ritort, R. Spigler, The kuramoto model: A simple paradigm for synchronization phenomena, Rev. Modern Phys. 77 (1) (2005) 137.

[33] P. Seliger, S.C. Young, L.S. Tsimring, Plasticity and learning in a network of coupled phase oscillators, Phys. Rev. E 65 (4) (2002) 041906.

[34] A. Arenas, A. Diaz-Guilera, C.J. Pérez-Vicente, Synchronization reveals topological scales in complex networks, Phys. Rev. Lett. 96 (11) (2006) 114102.

[35] J. Shao, Synchronization on Data Mining: A Universal Concept for Knowledge Discovery, LAP LAMBERT Academic Publishing, Saarbrücken, 2012.

[36] J. Shao, Z. Han, Q. Yang, T. Zhou, Community detection based on distance dynamics, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1075–1084.

[37] W. Ying, F.-L. Chung, S. Wang, Scaling up synchronization-inspired partitioning clustering, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 2045–2057.

[38] J. Shao, Q. Yang, H.-V. Dang, B. Schmidt, S. Kramer, Scalable clustering by iterative partitioning and point attractor representation, ACM Trans. Knowl. Discov. Data 11 (1) (2016) 5.

[39] J. Shao, F. Huang, Q. Yang, G. Luo, Robust prototype-based learning on data streams, IEEE Trans. Knowl. Data Eng. (2017).

[40] J. Shao, C. Gao, W. Zeng, J. Song, Q. Yang, Synchronization-inspired co-clustering and its application to gene expression data, in: 2017 IEEE 11th International Conference on Data Mining, ICDM, IEEE, 2017.

[41] J. Shao, X. Wang, Q. Yang, C. Plant, C. Böhm, Synchronization-based scalable subspace clustering of high-dimensional data, Knowl. Inf. Syst. 52 (1) (2017) 83–111.

[42] Y. Kuramoto, Self-entrainment of a population of coupled non-linear oscillators, in: International Symposium on Mathematical Problems in Theoretical Physics, 1975, pp. 420–422.

[43] Y. Kuramoto, Chemical Oscillations, Waves, and Turbulence, vol. 19, Springer Science & Business Media, 2012.

[44] C.S. Kim, C.S. Bae, H.J. Tcha, A phase synchronization clustering algorithm for identifying interesting groups of genes from cell cycle expression data, BMC Bioinformatics 9 (1) (2008) 56.

[45] J. Shao, C. Böhm, Q. Yang, C. Plant, in: J.L. Balcázar, F. Bonchi, A. Gionis, M. Sebag (Eds.), Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20–24, 2010, Proceedings, Part III, Springer, Berlin, Heidelberg, 2010 (Chapter Synchronization Based Outlier Detection).

[46] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, vol. 96, no. 34, 1996, pp. 226–231.

[47] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, in: ACM Sigmod Record, vol. 28, no. 2, ACM, 1999, pp. 49–60.

[48] L. Liao, D. Fox, H. Kautz, Extracting places and activities from gps traces using hierarchical conditional random fields, Int. J. Robot. Res. 26 (1) (2007) 119–134.

[49] L. Wang, K. Hu, T. Ku, X. Yan, Mining frequent trajectory pattern based on vague space partition, Knowl.-Based Syst. 50 (2013) 100–111.

[50] C.-Y. Tsai, B.-H. Lai, A location-item-time sequential pattern mining algorithm for route recommendation, Knowl.-Based Syst. 73 (2015) 97–110.

[51] Y. Zheng, L. Zhang, Z. Ma, X. Xie, W.-Y. Ma, Recommending friends and locations based on individual location history, ACM Trans. Web 5 (1) (2011) 5.

[52] Y. Si, F. Zhang, W. Liu, CTF-ARA: An adaptive method for POI recommendation based on check-in and temporal features, Knowl.-Based Syst. 128 (2017) 59–70.

[53] J.-G. Lee, J. Han, X. Li, H. Gonzalez, TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering, Proc. VLDB Endow. 1 (1) (2008) 1081–1094.

[54] M. Lv, L. Chen, Y. Shen, G. Chen, Measuring cell-id trajectory similarity for mobile phone route classification, Knowl.-Based Syst. 89 (2015) 181–191.
[55] C. Böhm, C. Plant, J. Shao, Q. Yang, Clustering by synchronization, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 583–592.
[56] K. Deng, K. Xie, K. Zheng, X. Zhou, Trajectory indexing and retrieval, in: Computing with Spatial Trajectories, Springer, 2011, pp. 35–60.
[57] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, in: IEEE Trans. Robot. Autom., 1996.
[58] Y. Zheng, Q. Li, Y. Chen, X. Xie, W.-Y. Ma, Understanding mobility based on GPS data, in: Proceedings of the 10th International Conference on Ubiquitous Computing, 2008, pp. 312–321.
[59] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma, Mining interesting locations and travel sequences from GPS trajectories, in: Proceedings of the 18th International Conference on World Wide Web, 2009, pp. 791–800.
[60] Y. Zheng, X. Xie, W.-Y. Ma, Geolife: A collaborative social networking service among user, location and trajectory., IEEE Data Eng. Bull. 33 (2) (2010) 32–39.
[61] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, Y. Huang, T-drive: driving directions based on taxi trajectories, in: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2010, pp. 99–108.
[62] J. Yuan, Y. Zheng, X. Xie, G. Sun, Driving with knowledge from the physical world, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 316–324.
[63] P.D. Grünwald, I.J. Myung, M.A. Pitt, Advances in Minimum Description Length: Theory and Applications, MIT press, 2005.
[64] J. Bao, Y. Zheng, D. Wilkie, M. Mokbel, Recommendations in location-based social networks: a survey, GeoInformatica 19 (3) (2015) 525–565.
[65] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, 2018, arXiv preprint arXiv:1804.11192.