



# Query the trajectory based on the precise track: a Bloom filter-based approach

Zengjie Wang, et al. *[full author details at the end of the article]*

Received: 20 March 2020 / Revised: 6 January 2021 / Accepted: 18 February 2021 /  
Published online: 15 March 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

Fast and precise querying in a given set of trajectory points is an important issue of trajectory query. Typically, there are massive trajectory data in the database, yet the query sets only have a few points, which is a challenge for the superior performance of trajectory querying. The current trajectory query methods commonly use the tree-based index structure and the signature-based method to classify, simplify, and filter the trajectory to improve the performance. However, the unstructured essence and the spatiotemporal heterogeneity of the trajectory-sequence lead these methods to a high degree of spatial overlap, frequent I/O, and high memory occupation. Thus, they are not suitable for the time-critical tasks of trajectory big data. In this paper, a query method of trajectory is developed on the Bloom Filter. Based on the gridded space and geocoding, the spatial trajectory sequences (tracks) query is transformed into the query of the text string. The geospace was regularly divided by the geographic grid, and each cell was assigned an independent geocode, converting the high-dimensional irregular space trajectory query into a one-dimensional string query. The point in each cell is regarded as a signature, which forms a mapping to the bit-array of the Bloom Filter. This conversion effectively eliminates the high degree of overlap and instability of query performance. Meanwhile, the independent coding ensures the uniqueness of the whole tracks. In this method, there is no need for additional I/O on the raw trajectory data when the track is queried. Compared to the original data, the memory occupied by this method is negligible. Based on Beijing Taxi and Shenzhen bus trajectory data, an experiment using this method was constructed, and random queries under a variety of conditions boundaries were constructed. The results verified that the performance and stability of our method, compared to R\*tree index, have been improved by 2000 to 4000 times, based on one million to tens of millions of trajectory data. And the Bloom Filter-based query method is hardly affected by grid size, original data size, and length of tracks. With such a time advantage, our method is suitable for time-critical spatial computation tasks, such as anti-terrorism, public safety, epidemic prevention, and control, etc.

**Keywords** Fast trajectory query · Geography big data · The Bloom filter · Track query

## 1 Introduction

With the widespread use of sensors and GPS, a massive number of trajectory data are accumulated every day, such as taxi trajectory data, bus trajectory data, and other moving objects' trajectory data. In some important applications, such as anti-terrorism, infectious disease prevention, suspect tracking, etc., tracks queries often require high timeliness and accuracy. However, the given query point set is small (several points to thousands), but the trajectory database is massive (millions of trajectory points to billion-level), so it is hard to obtain precise tracks in a short time. Therefore, how to efficiently query the track from the trajectory big data is an important issue in the trajectory retrieval field.

At present, the main methods to solve the efficiency problem of trajectory query are divided into two major aspects: tree-based spatial index and signature-based methods. The tree-based spatial index method uses the spatial index mechanism to improve trajectory retrieval efficiency. This mechanism divides the geo-space regularly and then uses the tree structure to organize the trajectory data hierarchically. However, the unstructured essence and the spatiotemporal heterogeneity of the trajectory cause the tree-based structure to have a high overlap of search region, lowering query performance. The signature-based method uses the signature and storage structure of the trajectory to simplify and filter the trajectory. Because all the trajectory signatures are scanned and filtered before accessed, a large number of disk accesses are reduced. This type of method has certain advantages in trajectory organization and storage, and can better achieve the fuzzy matching and similarity measurement of trajectories, however, its problems such as difficulty in structure embedding and high complexity of signature calculation, would cause the inefficient trajectory search. These two methods both have improved the query performance to a certain extent, but they are far from the high efficiency, and their memory occupation and query time increase shapely when facing the trajectory big data. These two methods always have a bottleneck in efficiency while querying the trajectory based on the single point, and there is no research to search the trajectory based on the independent tracks (the subsequence of the trajectory).

The general trajectory retrieval takes the point as the minimum search object, instead of the point sequence (trajectory-subsequence, track). In fact, the essence of the track is a sequence of position records of moving objects in space and time [1], and the long time series trajectory consists of many tracks. A track is composed of a series of points, which could be regarded as a single independent object for the input condition of the query process. In this method, we attempt to deconstruct the trajectory into the independent tracks for the query.

The trajectory points are high-dimensional and disordered in time and space. However, when they are regarded as a track, they can be regarded as point stream data, which are very similar to text stream data in structure and semantics. Both stream data are composed of a series of points/characters, and the text string and the track as a whole both have special meanings. Therefore, we try to convert the irregular and disordered trajectory subsequence into a track string, and the gridded space and geocoding provide convenience for the mapping of tracks to strings. The geo-space is divided into regular cells by grid and each grid is given a unique independent geocode. The points located in the same cell would be regarded and simplified at the same point, thereby eliminating the random distribution of trajectory

points in each grid. The combination of grids makes irregular trajectory point sequence turn into regular encoded track strings and maintains the uniqueness after each track encoding process. After gridded and encoded, the track can be mapped into the bit-array of the Bloom Filter.

The Bloom Filter [2] has great advantages in string retrieval, and it represents a specific string through the combination of several bits, allowing the memory to be utilized efficiently. The query of the Bloom Filter is implemented with some independent hash functions, and the time complexity is closed to  $O(1)$  and this query is not affected by the length of the string. This advantage is particularly obvious when dealing with big data. The characteristics of the Bloom Filter provide the convenience of processing the massive encoded tracks. Therefore, it is possible to use the Bloom Filter to query trajectory big data, quickly and precisely.

In this paper, we try to use the Bloom Filter to construct a track query method, combining geographic grid, trajectory encoding, hash mapping, etc. to achieve a fast and precise query of massive trajectory data. This track query method mainly involves the construction of the geographical grid, the tracks encoding, the creation of the Bloom Filter, and the query of tracks, as shown in Fig. 1.

The contributions of this paper are shown as follows:

We have proposed a new fast retrieval method for trajectory data, called the BF-based track query method, which can quickly retrieve target track from massive trajectory big data in time complexity  $O(1)$ .

We use the track as the smallest unit of the query, rather than a single trajectory point, allowing spatial trajectory query to be converted into a string query.

We conduct extensive experiments study based on different vehicle trajectory datasets, to test the effectiveness, efficiency, stability, and robustness of our method. The time and efficiency of our method under different data volumes are compared. The experiment results demonstrate the effectiveness and efficiency of our method.

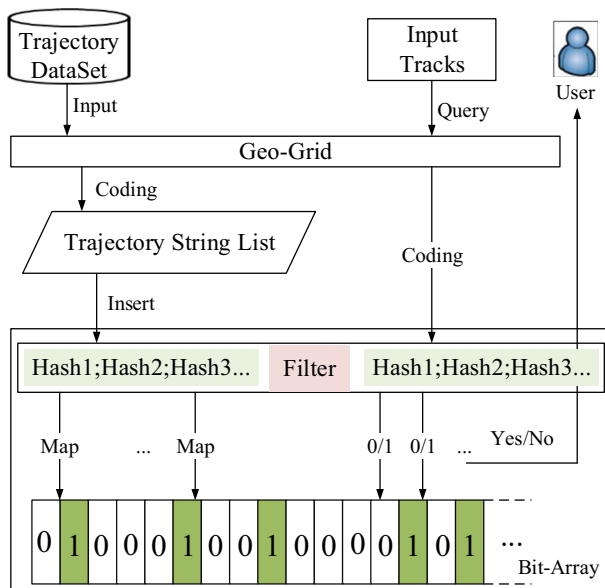


Fig. 1 The process of query method based on the Bloom Filter

## 2 Related work

To the best of our knowledge, previous studies of the tree-based index for improving trajectory query efficiency, mainly focus on the geo-space partition, minimizing coverage of retrieval region, and reducing search region. Most of these index structures are based on the development and evolution of the R-tree and its variants, and the core of their index is the tree-based structure.

**Geo-space partition** Geo-space partition uses the grid to divide the whole space region regularly, including SETI [3], SEB-Tree [4], etc. The SETI (Scalable and Efficient Trajectory Index) structure [3] proposed by V. Prasad Chakka et al. defines that geo-space is partitioned into non-overlapping regions, and each region builds a specific index: the R-Tree [5]. But in each sub-region, R-Tree will bring retrieval region overlap. SEB-Tree index structure tries to represent trajectory segments by their start and ends timestamp within a spatial grid. Although geo-space partition has reduced query time and has a good performance for regular spatial data, it is hard to process the irregular and random massive trajectory data.

**Minimizing coverage of retrieval space** The main access methods for minimizing coverage of retrieval space are R-Tree and its variants [5–7]. The R-tree index structure [5], proposed by Guttman, uses quadratics and linear algorithms to split spatial objects, but the tree-based index structures have a heavy MBR (minimum bounding rectangle) overlap. Greene's [8] split method, proposed by Greene, has much fewer pages overlap than Guttman's strategy. R\*Tree [6], proposed by Beckmann, uses the topological split algorithm to reduce page overlap, and the reinsertions further optimize the performance of the tree for the index of spatial points. Bulk loaded R\*Tree using Sort-Tile-Recursive (STR) [7], the leaf pages do not overlap at all, but it is hard to process the point sequence (the track) and the trajectory data required to be known beforehand. R-Tree and its variants do not use index trajectory directly, but most index structures make them a core and important part.

**Reducing search region** The trajectory is randomly distributed in the whole Euclidean space in principle, but most vehicles or moving objects usually move on geo-networks(road), instead of on the whole geo-space. Researchers have proposed many index structures based on fixed geo-networks. For objects constrained to move on fixed networks, Frenzos [9] proposed FNR-Tree (Fixed Network R-Tree), which is to construct a 1-dimensional R-trees on top of a 2-dimensional R-Tree to index trajectory segments. Li Guohui and Zhong Xiya [10] proposed an index structure called IMTFN (Indexing Moving objects Trajectories on Fixed Networks), which is based on the FNR-Tree and improves it. Victor Teixeira De Almeida and Ralf Hartmut Güting [11] proposed MON-Tree(Moving Objects in Networks Tree) index structure, which uses the top-level two-dimensional R-tree and a hash table structure to index the road sections in the traffic network. The index based on the fixed network can reduce the overlap of index space and redundancy to some extent. However, the trajectory index based on the fixed-network still has the problem of index coverage, because most index structures have to maintain an R-Tree as their core part.

**Signature-based methods** On the other hand, many researchers try to solve the bottleneck of track query performance from the aspects of trajectory signature and storage structure. Chang J et al. [12] proposed a signature-based trajectory indexing scheme for moving the object's

trajectory, which has better performance than the TB-tree and FNR-tree. Shim et al. [13] introduced a new signature-based indexing scheme that supports similar sub-trajectory retrieval. Scanning and filtering all signatures before accessing the trajectory data reduces a large number of disk accesses thus achieving better performance. PIST [14], introduced by Botea et al., tries to partition points set into a variable-sized grid and then uses the quad-tree to query the data. But it does not adapt to new data being added dynamically, and it focuses on indexing points rather than trajectories. The TrajStore [15], introduced by Philippe et al., proposes a new adaptive storage system that indexes the trajectory data based on the quad-tree index and clustering methods. Same as PIST, it partitions points rather than trajectories, and it hard to deal with the long-sequence trajectory and trajectory big data. The SharkDB [16] attempts to use in-memory storage to achieve the purpose of fast track retrieval and no I/O cost, but the memory would overload when the SharkDB faces the big data of trajectory. Although researchers have done a lot of research on the storage structure and storage modes of the trajectory, no method is found to meet the rapid and precise query of the fast-growing massive trajectory data.

**The Bloom filter** The typical Bloom Filter was put forward by Burton Howard Bloom [2] in 1970, but it does not support some operations, such as the delete, insert, and other operations. To solve these problems, researchers introduced a series of Bloom Filter variants based on Burton's work. For delete and insert operations, the Counting Bloom filters [17] proposed by Fan et al., provide an operation that can delete and insert elements without having to re-create a new filter. Almeida et al. proposed Scalable Bloom filters [18], which dynamically insert elements while ensuring a false positive rate (FPR). The Bloom Filter can also be used for the spatial query. The Spatial Bloom filters, proposed by Palmieri et al. [19] aim to store location information, especially when the location is confidential. For performance improvement, the Parallel Partitioned Bloom Filters [20] can perform parallel hash calculations for insertion and query; the Distributed Bloom filters [21] support parallel processing of elements. The data structure proposed by Kiss et al. [22] can exclude not only false negatives but also false positives, which is extremely important for those FPR intolerable scene applications. At present, the Bloom Filter has perfected the complete system of queries or indexes for the specific goals and scenarios. And the retrieval method based on the Bloom Filter, which is different from the index of the tree-based structure, is a Boolean decision, not a logical judgment. It will help to improve retrieval efficiency.

In short, the tree-based index structure design satisfies the spatial relationship among spatial geometry objects. It has certain advantages for the regular distribution of spatial objects and spatial relationship queries, but it is not suitable for Spatio-temporal trajectory data, especially the long trajectory. Also, the research on the storage and organization of trajectory data has improved the query efficiency of trajectories to a certain extent, but when massive trajectory big data are processed, there are still excessive memory usage and frequent I/O. Therefore, it is particularly important to develop a BF-based trajectory query method.

## 3 Method

### 3.1 Problem definition

Figure 2 gives an illustration of a GPS trajectory. The trajectory, which consists of plenty of points, is the path of a body as the path travels through space. The track is the record left by

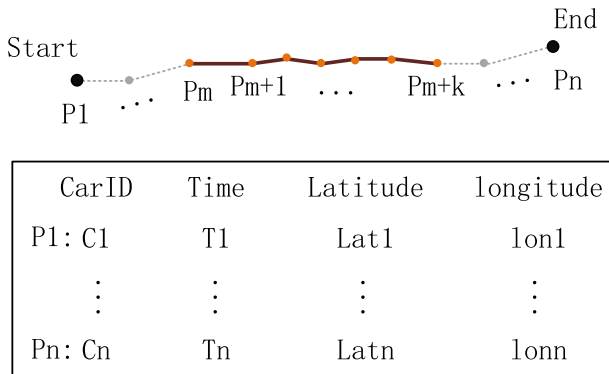


Fig. 2 GPS trajectory and corresponding subsequence

something that has passed along, such as the track of a ship or the track of a car. That is to say, a trajectory is made of a track or points sequence. For the original vehicle trajectory data, we can extract a series of independent tracks, each of which can be represented as a single track, as depicted in the top part of Fig. 2.

Let  $p_i$  denote the start of the trajectory, and  $p_n$  denote the end time of the trajectory. The track of a GPS trajectory is a sequence of points between the point  $p_m$  and the point  $p_{m+k}$ . Put formally,

$$l = \{p_m, \dots, p_{m+k}\} \tag{1}$$

The given GPS trajectory data are defined as  $T$ , where the track is represented as the sequence  $l = \{p_m, \dots, p_{m+k}\}$  of the trajectory point:  $p_i$  represents a single trajectory point, represented by the tuple  $p_i = \langle id, timestamp, latitude, longitude \rangle$ , where  $id$  represents car's ID,  $timestamp$  represents the time of the trajectory point  $p_i$  records,  $latitude$  represents the latitude coordinate of  $p_i$  coordinate, and  $longitude$  represents the longitude coordinate of  $p_i$ , as depicted in the bottom part of Fig. 2. The track  $Q$  to be queried is represented as  $Q = \{q_1, q_2, \dots, q_n\}$ . The geographical grid is defined as  $G_{geo}$  and the range is represented as  $T_{msr}$ , where  $minX, minY, maxX, maxY$  is the minimum spatial range of the coordinates of all trajectory points in the trajectory set  $T$ , and the cell size in the geographic grid is represented as  $G_{geo}$  meters. The grid is divided into longitude direction and latitude direction from the beginning of the upper right corner of the grid, which is represented as  $Code = \begin{cases} x_1, x_2, \dots, x_n \\ y_1, y_2, \dots, y_n \end{cases}$ . For example, based

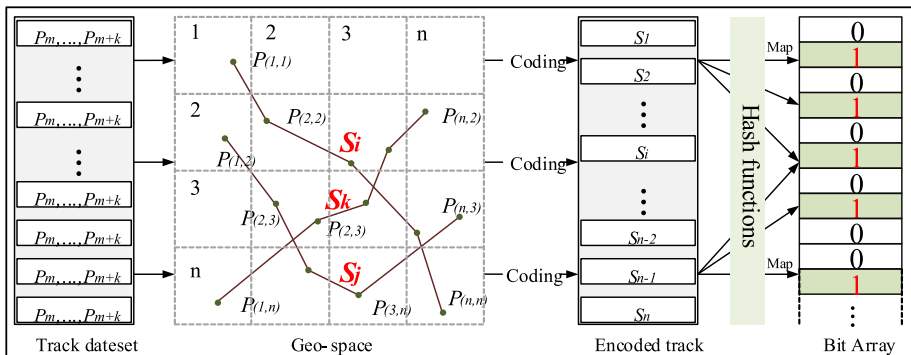


Fig. 3 Schematic diagram of geographical grid representation of the trajectory

on the trajectory path represented by the geographical grid, Fig. 3 shows how to map the trajectory sequence to the Bloom Filter.  $S_i$  represents a track, each of which contains several trajectory points. For example, the trajectory  $S_i$  would be encoded as  $Code_{(S_i)} = id-112232n3nn$  (Table 1).

**Definition 1 (track).** The track is a subsequence of trajectory, which is a subset of massive vehicle trajectory dataset, represented by  $l$ , continuous in time and space, and has a certain linguistic significance. In this paper, the track represents the spatial path of the vehicle in a period. Given a trajectory data set  $T$ , we can construct a series of tracks according to the original irregular trajectory points.

$$l = \{p_1, p_2, \dots, p_n\}, \quad l \subseteq T, \quad T = \sum_{i=0}^n l$$

**Definition 2 (geo-grid)** The geo-grid, represented by  $G_{geo}$ , is the geographic grid, which is mainly used for regularizing geographic space and track coding. Given a trajectory dataset  $T$ , the geo-grid will be constructed based on  $T$ 's minimum bounding extent  $T_{msr}$  and the single grid size  $g_{size}$ , and every single grid has an independent geocode, as shown in Fig. 3 left part.

$$G_{geo} = \{T_{msr}, g_{size}\}$$

**Definition 3 (geocoding).** Geocoding is the process of mapping the tracks to the encoded tracks. The intersection of the trajectory data and the geo-grid can get the encoded trajectory set  $\sum_{i=0}^n S_i$ , as shown in Fig. 3 middle part.

$$\sum_{i=0}^n S_i = G_{geo} \cup \sum_{i=0}^n l$$

**Table 1** Table of notations

Symbols	Description
$p_i$	the trajectory point
$l$	a sequence of points, track
$\sum_{i=0}^n l$	all tracks set
$T_{msr}$	the minimum spatial range of all trajectory points
$Q$	a track to be queried
$g_{size}$	the cell size of geo-grid
$S_i$	a track after encoded
$T$	the trajectory data
$FPR$	false-positive rate
$G_{geo}$	the geo-grid
$\sum_{i=0}^n S_i$	a set of the encoded track
$BF$	the Bloom Filter
$hf$	the hash function of the Bloom Filter
$B_{arr}$	the bit array of the Bloom Filter
$I_{TBF}$	the trajectory index based on Bloom Filter

**Definition 4 (the Bloom Filter retrieval)** The retrieval based on the Bloom Filter, represented by  $I_{TBF}$ , is an indexing structure constructed on hash functions and bit-array of the Bloom Filter. The Bloom Filter generates  $m$  hash values based on each encoded track and maps hash values one by one into the bit-array, as shown in Fig. 3 right part.

$$I_{TBF} = \left\{ \sum_{i=0}^n S_i, \sum_{j=0}^m hf, B_{arr} \right\}$$

### 3.2 Bloom filter

Since the Bloom Filter was put forward by Burton Howard Bloom [2] in 1970, it has been widely used in big data queries to improve query efficiency and reduce memory overload. The query time of Bloom Filter is under a constant range and the cost of storage space is small, so it has good practical value. It is mainly used in the data dictionary, data judgment, collection, and intersection, such as dictionary query [23], Spam Filtering system [24], and so on.

The essence of Bloom Filter is to use several bits to represent the elements in the bit-array. The essence of this paper is to map the string elements in the collection into a bit array through  $k$  independent hash functions. As depicted in the right part in Fig. 4,  $S_i$  represents the string set. The string obtains multiple addresses in the bit-array through multiple hash operations and sets the corresponding value on the address to 1.

### 3.3 Framework

In this section, we will present our framework of this method based on the Bloom Filter. Our framework consists of geo-grid construction, tracks encoding, and querying track. Algorithm 1 outlines the construction process. In the first phase, we obtain the minimum spatial range ( $T_{msr}$ ) based on the trajectory data, and then use  $T_{msr}$  and  $g_{size}$  to create a geographic grid and assign a specific code. Finally, we get an encoded geo-grid  $G_{geo}$ . The second phase aims to perform trajectory encoding. Through geo-grid, each independent track is encoded to  $S_i$ , and a set  $\sum_{i=0}^n S_i$  is formed. In the third phase, we construct the Bloom Filter  $I_{TBF}$ , whose elements are the encoded tracks. We create the Bloom Filter based on the size of  $\sum_{i=0}^n S_i$  and  $FPR$ . Then according to the results obtained in the first two phases, the  $I_{TBF}$  will be constructed.

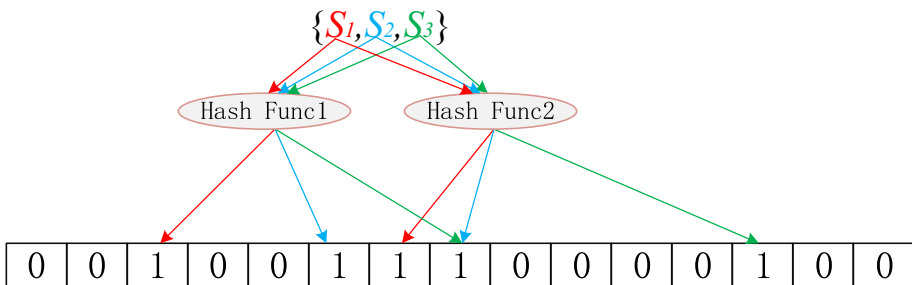


Fig. 4 An example of the Bloom filter



When we have a track  $Q$  to be queried, the  $Q$  needs to be encoded first. Then the  $Q$  will be input to  $I_{TBF}$ . After a series of processing  $I_{TBF}$  can judge whether the track  $Q$  exists in the trajectory dataset. The specific details will be discussed further in the following two sections.

### 3.4 The geographic grid and geocoding

The geographic grid (geo-grid,  $G_{geo}$ ) is a unified and simple geospatial partitioning and positioning reference system. According to unified rules, the area is continuously divided according to distance or latitude and longitude to form regular or irregular polygons. Each polygon is called a grid or cell (a unit grid/cell) and is given a unique code. The geo-grid used in this paper is regular and decided by distance. Other forms of the grid (irregular grid, Voronoi, etc.) can also be used, depending on specific geospatial applications. The grid size of the geo-grid and the study areas are decided by the real problem and user selection. When the geo-grid is created, each grid is given a unique code, and each track will get a unique code. Each of these tracks includes the vehicle ID and the points, ensuring that each track is unique. The trajectory geocoding process is shown in Fig. 4.

### 3.5 Track insert and query process

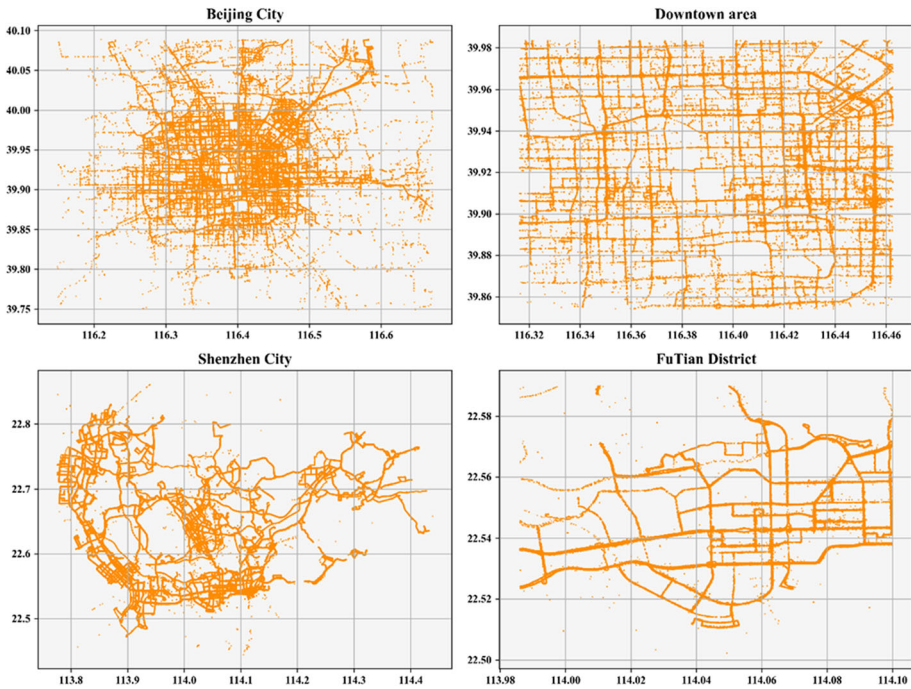
Unlike the traditional index structure, the index table of the Bloom Filter is represented by a bit array. The Bloom Filter initial bit array is set to 0. The number of a hash function is decided by the data size of Trajectory instead of the user set. An element (track) calculated by hash functions only occupies several bits of the bit-array. And a bit can also be shared by multiple elements, as shown in Fig. 4. The creation of the Bloom Filter is mainly to construct bit array and hash functions, and its elements from the string set of trajectory point sequence are formed by geo-coding. After the  $k$  hash operations of each track string, which will be mapped to the bit array randomly, the Bloom Filter of the track will be constructed.

The query process of the track—to judge whether the input track exists in the set of the current trajectory dataset—is based on Bloom Filter. Because the input is a sequence of coordinate points, it is necessary to use the geo-grid to encode these points, the same as the construction process. The results are mapped to the bit array one by one, and it is judged that if the result of the hash operation has one bit different from the bit array, it can be concluded that the input track does not exist in the trajectory set, and if it is all the same as the bit array, it can be judged that the path is most likely to exist in the trajectory set. The whole method construction and query process are shown in Fig. 1.

## 4 Experiments and results

### 4.1 Experiment data description

The data used in this study come from the T-drive taxi trajectory data set provided by Microsoft Research Institute [25, 26] and Shenzhen City bus trajectory data, as shown in Fig. 5. Because each taxi and bus trajectory is a long period of space-time point sequence, to facilitate experiments and performance testing, the data will be segmented to simulate the independent tracks of the vehicle. According to the topological relationship, time threshold, or other segmentation methods, the tracks are formed by



**Fig. 5** Trajectory data overview of the study area (Beijing City and its Downtown area; Shenzhen City and its FuTian District)

several points of one trajectory. The segmented points sequence will constitute the vehicle track set.

The experimental platform is Windows 10, the processor is Intel (R) Core (TM) i7-9750H CPU 2.60 GHz, running memory 16GB, and the compiler version is MS VC++ 12.0. The third-party library of the Bloom Filter is ArashPartow/bloom [27]. The R\*Tree index is Boost.Geometry.Index [28]. The experimental data are the 2008 Beijing taxi trajectory data [25, 26] and the 2014 Shenzhen city bus trajectory data, Guangdong Province, China.

## 4.2 Results

In this paper, we conducted experiments on taxi trajectory data and bus trajectory data, the conclusions drawn are the same. As shown in Fig. 6, the first line shows the result of the taxi trajectory data, and the second line is the buses'.

Compared with the R\*Tree index, the query efficiency of this method is verified. For the precise query, the query time based on Bloom Filter is much shorter than that based on the R\*Tree index. Figure 6a shows the comparison of the query time of R\*Tree and the Bloom Filter with different numbers of data. The horizontal axis represents the number of points and the vertical axis represents 2000 query times. The number of points on each track is more than 5. Although the data sources are different, we can see that the query time based on R\*Tree is much longer than that based on the Bloom Filter, more than 1000 times according to the experiments. For the R\*Tree index, the query time complexity is  $O(\log n)$  [5], and the query mode of Bloom Filter is based on the hash function, and its time complexity is  $O(1)$ , which also directly illustrates that the index based on Bloom Filter is negligible in terms of query

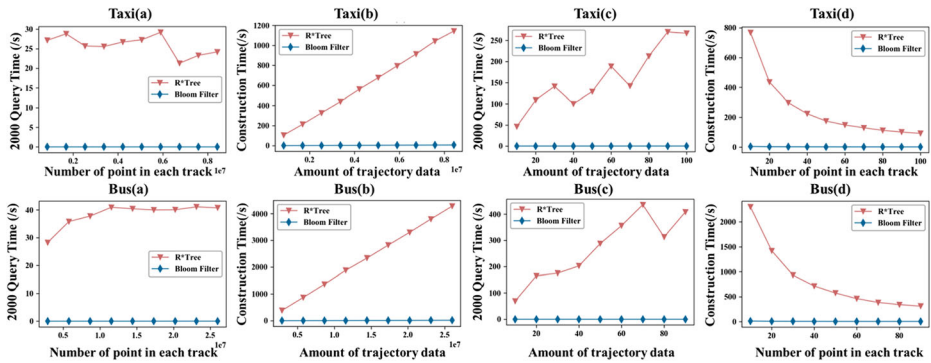


Fig. 6 Query and construction time of the Bloom Filter and R\*Tree index

time. Similarly, there is a large disparity in the time consumed by the construction of these two methods. The time consumed by the R\*tree-based method increases linearly with the increase of data. However, the time changes of the Bloom Filter is negligible. Also, the time consumption of the two methods is no longer on the same level. As shown in Fig. 6b, the horizontal axis represents the number of total points, and the vertical axis represents the cost time of constructing the method.

In the case of the same trajectory data, we tested the impact of a different number of points included in each track during the query and construction. The test results are as follows: The number of points included in each track does not affect the Bloom Filter. With the increase of the number of points in the track, the R\*Tree index has an overall upward trend, although it fluctuates, as shown in Fig. 6c. The horizontal axis represents the number of points in each track, and the vertical axis represents 2000 query times. Figure 6d shows the time consumed when the indexes were constructed, in which the two kinds of indexes have the same total number of track points, and the number of track points in each track is different. Also, although the two lines are slowly approaching, they still differ by several orders of magnitude in time consumption.

To further confirm the performance advantage of Bloom Filter for massive trajectories in other aspects, the following tests were performed in this paper:

The experimental result shows that the Bloom Filter has great advantages in memory consumption by using its bit-array, as shown in Fig. 7 and Table 2. Figure 7 shows the memory occupation of the Bloom Filter, including the memory occupied by the bit array, the encoded track set, and the original trajectory data. The transverse axis represents the number of points, and the longitudinal axis represents the amount of memory occupied, in KB. From the table and graph, we can conclude that compared with the original data, the memory consumption of the bit array is much smaller than the original data. With the increase of the number of the original data, the memory consumption of the bit-array increases linearly and slowly.

For the query efficiency of this method, the experimental results are as follows: in Fig. 8a, the transverse axis represents the number of points contained in each track, and the longitudinal axis represents the time consumed for 10,000 times of repeated query of the same tracks in seconds. The number of points in every single track has little effect on the query time. Moreover, the number of points in the different original trajectory set (4,073,886, 8,415,326, 12,871,223) has little effect on the retrieval time, which also satisfies the conclusion explained above, that is, the query time of Bloom Filter is determined by hash functions, and the time

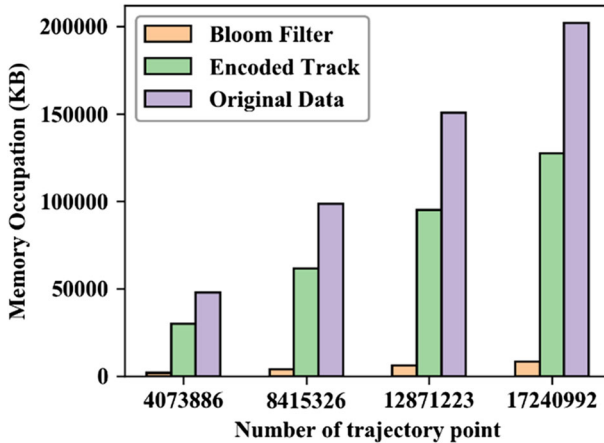


Fig. 7 The memory occupied Comparison of different amount of trajectory data

complexity is about (1). Also, the influence of grid size of geo-grid on query time is tested in this paper, such as shown in Fig. 8b: transverse axis represents the number of query points each time, longitudinal axis represents the time consumed for 10,000 query times in seconds, and the five curves represent the time consumed to retrieval track based on different grid sizes (100\*100, 500\*500, 1000\*1000, 3000\*3000, 8000\*8000, in meters). Through experiments, we can conclude that the grid size has little effect on the query efficiency of tracks with different lengths.

## 5 Discussion

### 5.1 FPR influence

For the Bloom Filter, the only weakness is that it has a certain false-positive rate, which means the Bloom Filter may misjudge elements that do not exist in the dataset, but those elements that exist in the set will not be misjudged as not belonging to the set, so we call it false positive instead of a mistake. Formula  $p = \left(1 - e^{-\frac{nk}{m}}\right)^k$  [2] represents the calculating method of the false positive rate.  $n$  denotes the number of input elements,  $k$  is the number of hash functions,  $m$  is the size of bit-array, and  $p$  is the false positive rate. In the experiment, the false positive rate is set to a certain value, which means that the overall false-positive rate needs to be less than this value. After many tests on the two data-sets, the false positive rate is far less than the set value. The tracks to be queried in the experiment are randomly generated, and the results are shown

Table 2 Comparison of memory occupied by different trajectory data size

Number of Trajectories	Bloom Filter(KB)	Encoded String (KB)	Original Trajectory (KB)
4,073,886	1909	29,830	47,741
8,415,326	3944	61,744	98,617
12,871,223	6032	94,975	150,835
17,240,992	8080	127,300	202,043

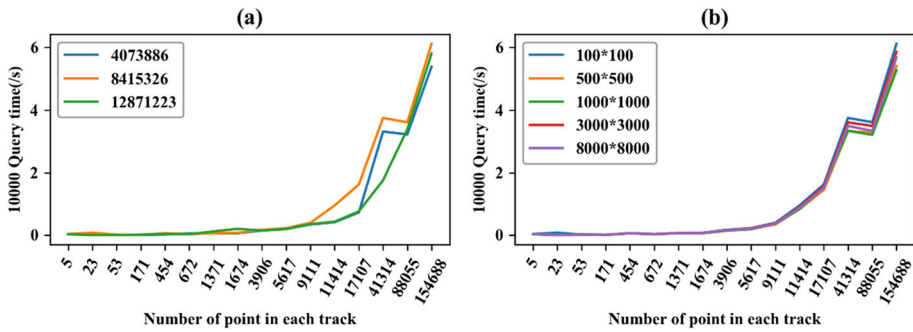


Fig. 8 Comparison of the influence of querying different track points on retrieval efficiency

Table 3 FPR of Beijing Taxi Trajectory (NTaxi means the number of taxi trajectory points)

NTaxi FPR	250 K	500 K	750 K	1000 K
0.00001	0	0	0	0
0.00005	1	1	1	1
0.0001	1	1	2	2
0.0005	7	14	16	20
0.001	4	12	21	61
0.005	337	385	479	569

in Tables 3 and 4, and Fig. 9. Figure 9 shows the comparison between the experimental value and the set value of the FPR, where *FPR* is the set value, *exp\_Taxi* is the experimental FPR of the taxi trajectory data, and *exp\_Bus* is the bus trajectory. The horizontal axis of Fig. 9 represents the number of different trajectory points in the two trajectory data sets, and the vertical axis represents FPR. Tables 3 and 4 give specific experimental values. In this paper, we tested FPR under different data volumes, but for the different scenarios, FPR could be different, which needs further study.

### 5.2 Drawbacks and future work

At present, there are many shortcomings in this method, and a lot of work is needed to complete this method. This work only distinguishes the tracks in the way of precise matching and does not further verify the fuzzy query. The tracks are simulated by segmented taxi and

Table 4 FPR of Shenzhen Taxi Trajectory (NBus means the number of bus trajectory points)

NBus FPR	1000 K	2000 K	3000 K	4000 K
0.00001	0	0	0	0
0.00005	1	3	3	3
0.0001	2	4	7	12
0.0005	13	27	53	90
0.001	35	159	219	294
0.005	1055	1679	2165	2807

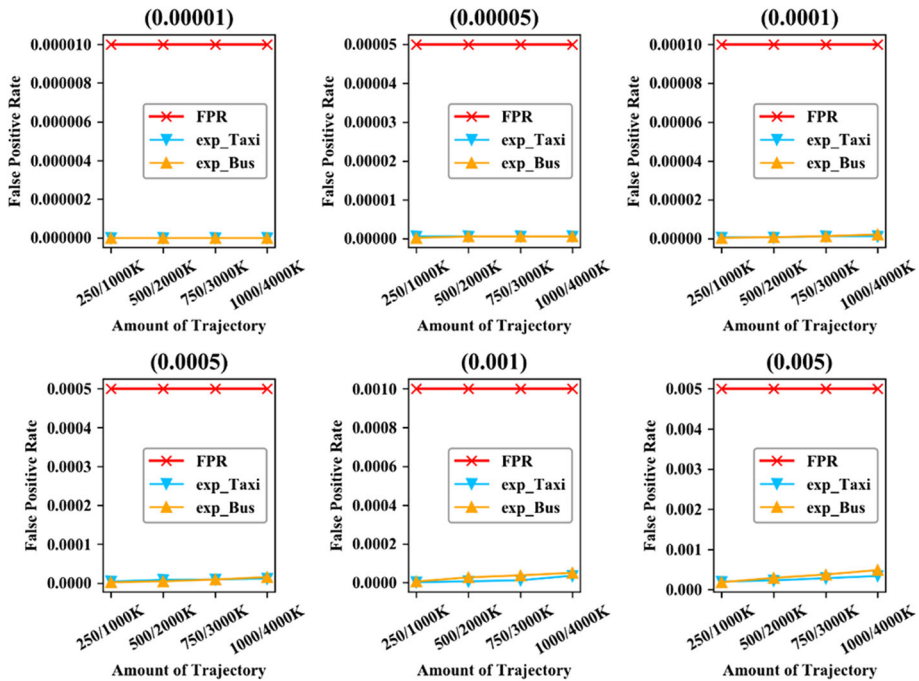


Fig. 9 False positive rate of set valves and experience values

bus trajectory, and there is no further processing of the merging, deletion, and simplification of the points in the real tracks. For more effective geocoding, the geographical grid also needs to be further studied according to the spatial distribution of the points.

Although this work is only the first step to verify the use of the Bloom Filter to query tracks, it is an important step. If we want to complete the construction of the Bloom Filter-based trajectory query method, an in-depth research is needed in the insertion, update, deletion operation, and other aspects. For later research, we plan to use the Counting Bloom Filters [17] to improve update and delete operations and use the structure proposed by Kiss et al. [22] to study the efficiency and performance of trajectory query without FPR.

## 6 Conclusion

This method verifies that the Bloom Filter has a great advantage of querying trajectory point sequence, in terms of both time efficiency and memory occupancy. Moreover, the point sequence, instead of querying every single point, is regarded as the minimum retrieval granularity in this method, which can convert the track query into a string query. Also, this method has robust stability, and the retrieval efficiency is hardly affected by the length of the input tracks, the size of the geo-grid, and data size. In the process of querying, to judge the existence of a certain track only needs to compare the hash results one by one, and the comparison is the IO operations on bit-array instead of the original data. The most important is that this method can effectively avoid a low recognition rate of traditional tree-oriented index structure and index overlapping.



Big data contain a great deal of information and are widely used. A suitable query method should be constructed according to the different applications of big data. In the application of trajectory big data, our method provides a new model for the fast retrieval of tracks and also provides a new idea for querying spatial trajectory data. Therefore, our method could be applied to criminal suspect trajectory retrieval, epidemic prevention and control, and so on. When the suspects escape, they will inevitably leave their trajectory, even if it is discrete track points. Based on the method in this paper, the points are combined into tracks and quickly retrieved in trajectory big data to obtain tracks that match the suspects' trajectory in the same period. Even if the trajectory is not of the suspects', it may be the witnesses', which can quickly provide clues for the investigation of the case. For those who come in contact with the 2019-nCoV or other infectious diseases patients, their trajectories can be collected as input tracks. Based on these tracks of infected persons, searching for the same or related tracks in the trajectory big data can find close contacts with the infected persons in a short time.

**Acknowledgments** This work was supported in part by the National Natural Science Foundation of China under Grant 41625004, 41971404 and the Key Projects of Research and Development under Grant 2016YFB0502301.

## References

1. Li X, Han J, Kim S, Gonzalez H (2007) ROAM: rule- and motif-based anomaly detection in massive moving object data sets. In: Proceedings of the 2007 SIAM International Conference on Data Mining, pp 273–284. <https://doi.org/10.1137/1.9781611972771.25>
2. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. *Commun ACM* 13:422–426. <https://doi.org/10.1145/362686.362692>
3. Prasad V, Adam C, Everspaugh C, Patel JM (2002) Indexing large trajectory data sets with SETI. Conference on Innovative Data Systems Research
4. Song Z, Roussopoulos N (2003) SEB-tree: an approach to index continuously moving objects. In: Chen M-S, Chrysanthos PK, Sloman M, Zaslavsky A (eds) *Mobile Data Management*. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 340–344
5. Guttman A (1984) R-trees: a dynamic index structure for sparial searching. *ACM Sigmod Record* 14(2):47–57. [https://doi.org/10.1007/978-3-319-23519-6\\_1151-2](https://doi.org/10.1007/978-3-319-23519-6_1151-2)
6. Beckmann N, Kriegel H-P, Schneider R, Seeger B (1990) The R\*-tree: An efficient and robust access method for points and rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data. ACM, New York, pp 322–331. <https://doi.org/10.1145/93605.98741>
7. Leutenegger ST, Lopez MA, Edgington J (1997) STR: a simple and efficient algorithm for R-tree packing. In: Proceedings 13th International Conference on Data Engineering. Soc. Press, IEEE Comput, pp 497–506. <https://doi.org/10.1109/ICDE.1997.582015>
8. Greene D (1989) An implementation and performance analysis of spatial data access methods. In: [1989] Proceedings. Fifth International Conference on Data Engineering, pp 606–615. <https://doi.org/10.1109/ICDE.1989.47268>
9. Frentzos E (2003) Indexing objects moving on fixed networks. In: Hadzilacos T, Manolopoulos Y, Roddick J, Theodoridis Y (eds) *Advances in spatial and temporal databases*. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 289–305
10. Li G (2006) Indexing Moving Objects Trajectories on Fixed Networks *J Comput Res Dev* 43:. <https://doi.org/10.1360/crad20060509>
11. Brunette W (2017) Extended abstract: building mobile application frameworks for disconnected data management. In: Proceedings of the 2017 workshop on MobiSys 2017 Ph.D. forum, co-located with MobiSys 2017. pp 15–16. <https://doi.org/10.1145/3086467.3086475>

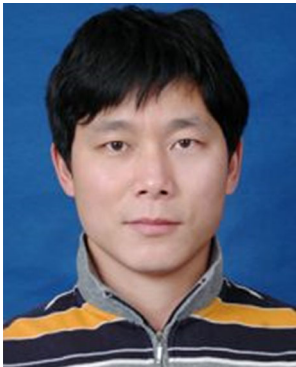
12. Chang J, Um J, Kim Y (2008) A new signature-based indexing scheme for trajectories of moving objects on spatial networks. In: Bubak M, van Albada GD, Dongarra J, Sloot PMA (eds) Computational science – ICCS 2008. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 731–740
13. Shim C-B, Chang J-W (2004) Signature-based indexing scheme for similar sub-trajectory retrieval of moving objects. *KIPS Trans* 11D:247–258. <https://doi.org/10.3745/KIPSTD.2004.11D.2.247>
14. Botea V, Mallett D, Nascimento M, Sander J (2008) PIST: an efficient and practical indexing technique for historical Spatio-temporal point data. *Geoinformatica* 12:143–168. <https://doi.org/10.1007/s10707-007-0030-3>
15. Cudre-Mauroux P, Wu E, Madden S (2010) TrajStore: an adaptive storage system for very large trajectory data sets. *Proc - Int Conf Data Eng* 109–120. <https://doi.org/10.1109/ICDE.2010.5447829>
16. Wang H, Zheng K, Xu J, et al (2014) SharkDB: an in-memory column-oriented trajectory storage. *CIKM 2014 - Proc 2014 ACM Int Conf Inf Knowl Manag* 1409–1418. <https://doi.org/10.1145/2661829.2661878>
17. Fan L, Cao P, Almeida J, Broder AZ (2000) Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans Netw* 8:281–293. <https://doi.org/10.1109/90.851975>
18. Deng F, Rafiei D (2006) Approximately detecting duplicates for streaming data using stable bloom filters. *Proc ACM SIGMOD Int Conf Manag Data* 25–36. <https://doi.org/10.1145/1142473.1142477>
19. Lin D, Yung M, Zhou J (2015) Information security and cryptology: 10th international conference, Inscrypt 2014 Beijing, China, December 13–15, 2014 revised selected papers. *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)* 8957:3–4. <https://doi.org/10.1007/978-3-319-16745-9>
20. Vu VH, Wang K (2014) Random weighted projections, random eigenvectors. *Random Struct algorithms* 1–30. <https://doi.org/10.1002/rsa>
21. Sanders P, Schlag S, Muller I (2013) Communication efficient algorithms for fundamental big data problems. *Proc - 2013 IEEE Int Conf big data. Big Data* 2013:15–23. <https://doi.org/10.1109/BigData.2013.6691549>
22. Kiss SZ, Hosszu E, Tapolcai J, et al (2018) Bloom Filter with a False Positive Free Zone. *Proc - IEEE INFOCOM 2018-April*:1412–1420. <https://doi.org/10.1109/INFOCOM.2018.8486415>
23. Bender MA, Farach-Colton M, Goswami M et al (2018) Bloom filters, adaptivity, and the dictionary problem. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pp 182–193. <https://doi.org/10.1109/FOCS.2018.00026>
24. Yan J, Cho PL (2006) Enhancing collaborative spam detection with Bloom filters. In: 2006 22nd Annual Computer Security Applications Conference (ACSAC'06), pp 414–428. <https://doi.org/10.1109/ACSAC.2006.26>
25. Zheng Y, Xie X, Ma W-Y (2009) Mining interesting locations and travel sequences from GPS trajectories. In: *Proceedings of International Conference on World Wide Web 2009, Proceeding*. <https://doi.org/10.1145/1526709.1526816>
26. Yuan J, Zheng Y, Zhang C et al (2010) T-drive: driving directions based on taxi trajectories. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, New York, NY, USA*, pp 99–108. <https://doi.org/10.1145/1869790.1869807>
27. Partow A (2018) bloom. <https://github.com/ArashPartow/bloom>
28. boost::geometry::index::rtree. [https://www.boost.org/doc/libs/1\\_65\\_1/libs/geometry/doc/html/geometry/reference/spatial\\_indexes/boost\\_geometry\\_index\\_rtree.html](https://www.boost.org/doc/libs/1_65_1/libs/geometry/doc/html/geometry/reference/spatial_indexes/boost_geometry_index_rtree.html)

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.





**Zengjie Wang** was born in Chuzhou, Anhui, China, in 1994. He received the B.S. degree in Geographic information science from Shandong University of Technology, in 2018. He is currently pursuing the M.S. degree in cartography and geographic information system with the School of Geography, Nanjing Normal University. His research interests include data analysis, spatial analysis and geometric algebra GIS.



**Wen Luo** received the B.S. degree in geography science from Hubei University, Wuhan, China, in 2008, the M.S. degree in geographical information system from Nanjing Normal University, Nanjing, China, where he has been working toward the Ph.D. degrees since 2011 and majored in the field of geographical information system. His research interests are geographical information system, geometric algebra and computer algorithms. He managed one national science funding projects as the principle investigator, also participated in more than four other national science funding projects.



**Linwang Yuan** received the B.S. degree in geography science, the M.S. degree and the Ph.D. degree in geographical information system from Nanjing Normal University, Nanjing, China, in 1995, 1998 and 2001, respectively. He is currently a Professor of the Key Laboratory of VGE, Ministry of Education. His research interests include geographic information science and geographical modeling. He managed five national science funding projects as the principle investigator, including the National Science Fund for Distinguished Young Scholars.



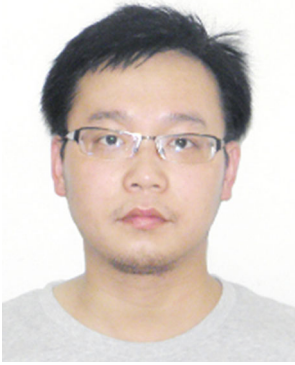
**Hong Gao** was born in Tianshui, Gansu, China, in 1993. He received the B.S. degree in School of Geography, Nanjing Normal University, in 2015. He received the M.S. degree in University of Chinese Academy of Sciences in 2018. He is currently pursuing the Ph.D. degree in Geographic Information System from School of Geography, Nanjing Normal University. His main research interest is spatio-temporal data analysis with spatial statistics, time series model, machine learning and other mathematical method, including spatial predictive mapping and traffic flow prediction.



**Fan Wu** was born in Huaihua, Hunan, China. He received the B.S. degree in geographical information system from Sun Yat-sen University, Guangzhou, China in 2015. Then, he has been working for his M.S. degree since 2018 and majored in the field of Cartography and geographical information system in Nanjing Normal University, Nanjing, China. His research interests involve geographical information science, semantic GIS and geometric algebra.



**Xu Hu** received the bachelor's degree in Geographic Information Science from Southwest Petroleum University, Chengdu, China. He is currently pursuing the master's degree with the school of geography and science, Nanjing Normal University, China. He is currently involved in expressway traffic flow simulation, especially on the simulation of expressway traffic flow by quantum random walk



**Zhaoyuan Yu** received the B.S. degree in geography science, and Ph.D. degrees in GIS and Cartography from Nanjing Normal University, Nanjing, China, in 2005 and 2011, respectively. He is currently an Associate Professor of the Geography Department of Nanjing Normal University, Nanjing, China. His research interests include geographic information system, geographical modeling and geometric algebra. He managed four national science funding projects as the principle investigator, also participated in more than ten other national science funding projects. He has got the First Prize of Natural Science of Ministry of Education of China, 2017, the First Chinese Science and Technology Progress Award in the field of Geographic Information Science, China Association in GIS, 2017, and the Excellent Young Scholars Award of Jiangsu Province in the field of remote sensing and geographical information science, 2016.

## Affiliations

**Zengjie Wang**<sup>1</sup> • **Wen Luo**<sup>1,2,3</sup> • **Linwang Yuan**<sup>1,2,3</sup> • **Hong Gao**<sup>1</sup> • **Fan Wu**<sup>1</sup> • **Xu Hu**<sup>1</sup> • **Zhaoyuan Yu**<sup>1,2,3</sup>

✉ Zhaoyuan Yu  
yuzhaoyuan@njnu.edu.cn

<sup>1</sup> Ministry of Education, Key Laboratory of Virtual Geographic Environment (Nanjing Normal University), Nanjing, China

<sup>2</sup> State Key Laboratory Cultivation Base of Geographical Environment Evolution (Jiangsu Province), Nanjing, China

<sup>3</sup> Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing, China