International Journal of
*Geo-Information*

*Article*

# GeoSOT-Based Spatiotemporal Index of Massive Trajectory Data

**Chunyao Qian [1], Chao Yi [1], Chengqi Cheng [2], Guoliang Pu [2], Xiaofeng Wei [2,*] and Huangchuang Zhang [2]**

[1] School of Earth and Space Sciences, Peking University, Beijing 100871, China; cyqvanilla@pku.edu.cn (C.Q.); yichaopku@pku.edu.cn (C.Y.)
[2] College of Engineering, Peking University, Beijing 100871, China; ccq@pku.edu.cn (C.C.); pgl@pku.edu.cn (G.P.); zhanghuangchuang@pku.edu.cn (H.Z.)
* Correspondence: weixiaofeng@pku.edu.cn; Tel.: +86-010-6275-5390

Received: 6 May 2019; Accepted: 15 June 2019; Published: 18 June 2019

**Abstract:** With the rapid development of global positioning technologies and the pervasiveness of intelligent mobile terminals, trajectory data have shown a sharp growth trend both in terms of data volume and coverage. In recent years, increasing numbers of LBS (location based service) applications have provided us with trajectory data services such as traffic flow statistics and user behavior pattern analyses. However, the storage and query efficiency of massive trajectory data are increasingly creating a bottleneck for these applications, especially for large-scale spatiotemporal query scenarios. To solve this problem, we propose a new spatiotemporal indexing method to improve the query efficiency of massive trajectory data. First, the method extends the GeoSOT spatial partitioning scheme to the time dimension and forms a global space–time subdivision scheme. Second, a novel multilevel spatiotemporal grid index, called the GeoSOT ST-index, was constructed to organize trajectory data hierarchically. Finally, a spatiotemporal range query processing method is proposed based on the index. We implement and evaluate the index in MongoDB. By comparing the range query efficiency and scalability of our index with those of the other two space–time composite indexes, we found that our approach improves query efficiency levels by approximately 40% and has better scalability under different data volumes.

**Keywords:** trajectory data; spatiotemporal index; spatiotemporal range query; GeoSOT

## 1. Introduction

With the development of sensor technologies and with the pervasiveness of intelligent mobile terminals, movements of humans and objects are increasingly being recorded as trajectories. Such trajectories offer a considerable amount of useful data that can be explored. In recent years, a broad range of LBS (location-based service) applications have been developed and improved through trajectory data mining [1], such as traffic flow statistical analysis [2], vehicle scheduling strategies [3], movement pattern analysis [4] for individuals or groups of moving objects, making sense of trajectories, and other applications of urban services. These applications significantly benefit common individuals, traffic organizations, and government agencies.

Before mining trajectory data, one must access different samples of trajectories or different parts of a trajectory many times. This stage is referred to as trajectory data indexing and retrieving [5], and it constitutes a fundamental step of many trajectory data mining tasks. In the case of tourist activities, for instance, for an application seeking to globally analyze activities performed by tourists during their stay in Beijing, the whole track left by tourists in Beijing is extracted (spatial constraints “inside Beijing”). In addition, to analyze what tourists do in one day in Beijing or what they do on specific days (e.g., on weekends), each daily track of the tourists must be extracted (temporal constraints "from Saturday to Sunday") [6]. For user similarity measurements, most applications usually select several trajectory segments in particular spatiotemporal units to model the structure of human mobile behavior [7]. Obtaining trajectories within a specific period of time and space is essential for these applications to conduct further research. The problem is that the large amount of data leads to long query periods, which reduces the efficiency at which applications can mine data and provide corresponding services, especially for online applications requiring the instant mining of trajectory data (e.g., detecting traffic anomalies). Since query efficiency can be improved to some extent through the establishment and use of reasonable indexes, we can design an appropriate index to solve these problems and to allow more time for data mining and analysis than for the query phase.

In accounting for the dynamic characteristics of trajectory data and the strong demand for spatiotemporal range queries, we contend that an optimal spatiotemporal index must meet the following requirements: (1) The index must treat the spatial and time dimensions equally. For many platforms that provide trajectory data management and analysis services, spatial and temporal restrictions are placed on data queries. When faced with massive data, it is necessary to ensure that filtering by time and spatial restrictions is not performed in two operations. It is best to filter three dimensions (latitude, longitude, and time) at a time. (2) The index should not be sensitive to high-frequency data updates. Considering the dynamic nature of moving objects, the index should not be updated with the object's movements at all times, as otherwise maintenance costs would be too high. (3) The index should be easy to integrate into existing data management systems. While the method proposed in [8,9] can establish corresponding effective index structures for a query problem to facilitate query processing, it is extremely complicated and difficult to integrate into an existing DBMS (database management system). We believe that a good spatiotemporal index should be "generalized" and easy to integrate and should not overwhelm the existing data organization scheme.

In this paper, we propose a novel index to solve the efficiency problem of spatiotemporal range queries. The indexing method extends the GeoSOT spatial subdivision model to the time dimension by combining the principle of spatial partitioning with time partitioning. Based on the GeoSOT spatial subdivision model, we applied the principle of time division and constructed a new spatiotemporal indexing model called the GeoSOT spatiotemporal index (GeoSOT ST-index). An effective processing method for large-scale spatiotemporal range queries is also introduced. After processing, the range query for trajectory data in three-dimensional space (time and 2D space) is converted into a one-dimensional code query in the GeoSOT index table. The rapid retrieval of trajectory data is achieved.

## 2. Related Work

To improve the efficiency of spatiotemporal retrieval, a common solution is to design and construct an appropriate index structure to greatly reduce the search space. Existing spatiotemporal indexes can be classified into two categories: object-oriented indexes and spatial-partitioning-based indexes.

Object-oriented spatiotemporal indexing technology mainly focuses on the deformation and expansion of two-dimensional spatial index structures. These indexes use individual trajectories' minimum bounding boxes (MBBs) as an indicator. These indexes usually have an R-tree-like structure, such as 3D R-tree [10], SETI [11], SEB-tree [12], and TB-tree structures [13]. These types of indexes are trajectory oriented. Regardless of how long a trajectory lasts or how large a spatial region it covers, an MBB is created for it. When the amount of data quickly increases, considerable overlapping and coverage between MBBs results, and index performance and query efficiency decrease significantly [14]. In addition, trajectory-oriented indexes must be updated as trajectory data are updated, leading to high maintenance costs [15,16].

Spatial-partitioning-based indexes have been widely employed in recent years due to their simplicity and include the quadtree [17] and grid indexes [18]. Such indexes divide space in advance so that when moving objects keep moving within the same spatial unit, the index does not need to be updated. When faced with high-frequency concurrent operations, space-partitioning-based indexes are far superior to R-tree family indexes [19]. Zheng [20] proposed the GAT (grid index for activity trajectories) for organizing trajectory segments and activities hierarchically. The indexing method employed involves dividing the whole space into a d-grid ($2^d \times 2^d$ grids) and further constructing a (d –1)-Grid, ..., 1-Grid. An inverted index [21] is built from the d layer to the upper layer to record the grid through which the moving object passes. Wang [22] proposed a spatiotemporal index for a road network based on quadtree. Adopting the space-first strategy, the geospatial space is first constructed based on the quadtree. The geographic hash code, i.e., the GeoHash value, is designed for spatial regions of different levels. Then, each Geohash code is combined with a time index to form a spatiotemporal composite index. Yang [23] developed GCOTraj, which partitions a large spatiotemporal data space into multidimensional grid cells and orders these grid cells in two different ways. The index performs well when dealing with large-scale spatial range queries. However, when the time query range is as large as the spatial query range, it performs poorly, as the time index is timestamp-based and thus only stores the start and end time stamps for each page. As increasing numbers of mobile terminals record trajectory data, the amount of data increases sharply over time. Thus, they are not suited for large-scale querying in time and space. As a spatial-partitioning-based index model, the GeoSOT (geographical coordinate subdividing grid with one dimension integer coding on $2^n$-tree) has been widely used for spatial data indexing and retrieval [24–28]. The above works proposed means of organizing and retrieving spatial data based on GeoSOT, but they only encode spatial information for the object without encoding its time information. In [29], a multiscale time partition and integer coding (MTIC) method is proposed. When using this method to index the time information of an object, high efficiency queries of multiple scales can be achieved.

## 3. Materials and Methods

### 3.1. GeoSOT Global Subdivision Model

The GeoSOT global subdivision model is a spatial indexing model based on spatial partitioning. With the intersection of the prime meridian and equator taken as the central point, GeoSOT recursively divides the surface of the earth into four grid cells [30]. Through quadtree recursive subdivision, these grid cells form a multilevel grid of latitude and longitude. This approach is novel in that the original surface space is expanded from 180° × 360° to 512° × 512°, and 1° (1′) is expanded from 60′ (60″) to 64′ (64″). The result is a one-dimensional integral grid code on the $2^n$-tree [31]. Figure 1. shows the GeoSOT subdivision and encoding procedure.

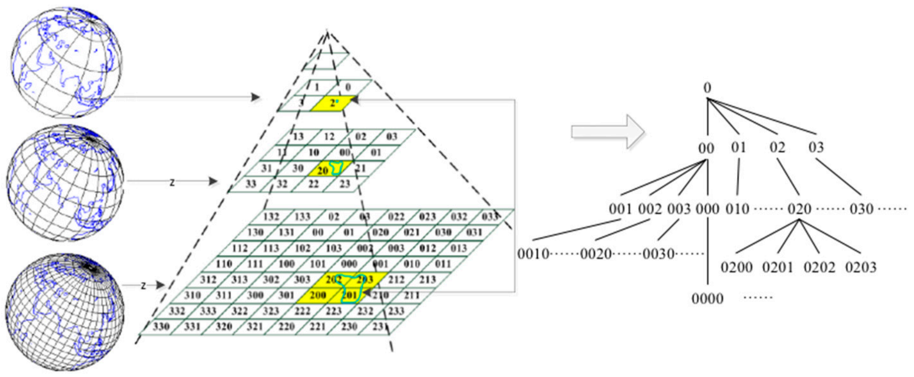

**Figure 1.** The GeoSOT subdivision and encoding procedure.

Each grid cell of each level is encoded by a geographical code according to the Z-order filling curve [32]. The code exhibits the following characteristics: (1) Uniqueness: Each grid cell has a globally unique code corresponding to a rectangle on the Earth's surface. (2) One-dimensional: GeoSOT uses a one-dimensional number or binary bit string to represent a region of two-dimensional space.

(3) Recursiveness: The GeoSOT lower-level grids are divided by the upper-level grid, essentially resulting in spatial Z-order filling curve padding. The four GeoSOT grid cells of a "Z" belong to the same parent grid, and their binary codes have the same prefix.

### *3.2. GeoSOT-Based Spatiotemporal Index Model*

Trajectory data are typical spatiotemporal data and present dynamic characteristics. Even over one second, a large number of mobile terminals update their locations. Thus, we consider extending the GeoSOT spatial partitioning model to a spatiotemporal partitioning model. With this approach, the index will not be updated as long as the moving object does not exceed the space–time unit.

First, we define the scope of the GeoSOT ST-index model:

Spatial extent: geographic longitude [−256°, 256°], geographic latitude [−256°, 256°]

Time extent: [1969, 2096]

As noted above, the spatial extent is consistent with GeoSOT. When selecting a time period, we mainly consider the following two points. The first point relates to the timeliness of the trajectory data. Since GPS trajectories have only been produced over the past few decades, we start with the year in which the Internet was first created, i.e., 1969. Second, the GeoSOT system has $2^n$ integer characteristics (that is, the spatial units of all granularities including degrees, minutes, seconds, and sub-seconds are integers), which requires that the time interval is also preferably of a power of 2. In theory, as long as the number of years is satisfied, a power of 2 can be used as the time range of division (e.g., 128 years, 256 years, and 512 years). However, the longer a time period is, the more subdivisions should be made to obtain the time granularity of a given day or hour. Therefore, a 128-year period (the 7th power of 2) is selected in this paper. As long as seven rounds of subdivision are applied, a one-hour resolution is obtained, fully meeting the needs of many trajectory data mining applications to retrieve data from hour to year intervals. In addition, to maintain integer characteristics of the grid, the time period should also be extended accordingly. The specific expansion scheme used is shown in Table 1. Each time grid includes the integer year, month, day, and hour code.

**Table 1.** Time division scheme.

| | Year | Month | Day | Hour |
|---|---|---|---|---|
| Original range | 1969–2096 | 1–12 | 1–31 | 1–24 |
| Extended range | 1969–2096 | 1–16 | 1–32 | 1–32 |

By recursively performing Octree [33] partitioning on the space–time cube defined by the GeoSOT ST-index model, a multilevel cube of latitude, longitude, and time with hierarchical characteristics is attained. The subdivision and coding method of the GeoSOT ST-index model is shown in Figure 2. The temporal and geographical scale of the cube at different levels is shown in Table 2. The encoding method is based on the Z-order three-dimensional space filling curve, and the trend of the filling curve is consistent with that of GeoSOT. We refer to this as the GeoSOT space–time code (ST-code).

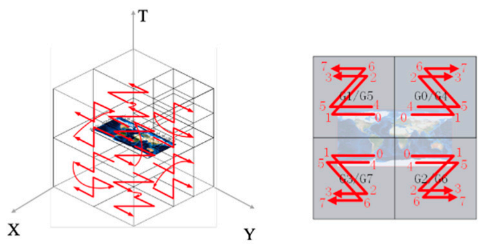

**Figure 2.** The subdivision and encoding method of the GeoSOT ST-index model.

**Table 2.** GeoSOT spatiotemporal cube scales of different levels.

| Level | Scale | Level | Scale | Level | Scale |
|---|---|---|---|---|---|
| 1 | 64 years 256° | 8 | 8 months 2° | 15 | 2 days 1′ |
| 2 | 32 year 128° | 9 | 4 months 1° | 16 | 1 day 32″ |
| 3 | 16 years 64° | 10 | 2 months 32′ | 17 | 16 h 16″ |
| 4 | 8 years 32° | 11 | 1 month 16′ | 18 | 8 h 8″ |
| 5 | 4 years 16° | 12 | 16 days 8′ | 19 | 4 h 4″ |
| 6 | 2 years 8° | 13 | 8 days 4′ | 20 | 2 h 2″ |
| 7 | 1 year 4° | 14 | 4 days 2′ | 21 | 1 h 1″ |

For a given level, any trajectory point falls into only one space–time unit whereas a space–time unit can contain multiple trajectory points. The GeoSOT ST-index structure of the trajectory data is shown in Figure 3. When a moving object frequently updates its position within a space–time grid, it will not cause the index to update. A reduction in the number of updates means there are no long locks, definitely improving query efficiency levels.

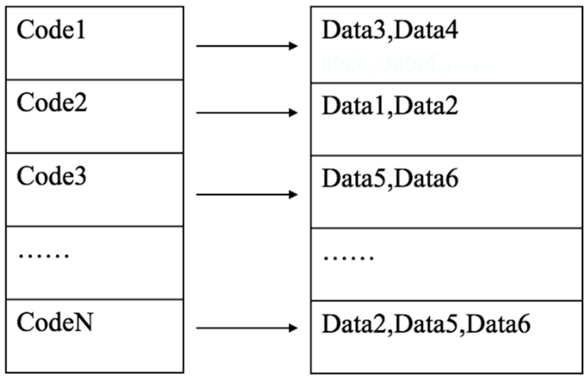

**Figure 3.** GeoSOT ST-index structure.

When inserting a new trajectory, we must first establish the mapping relationship between the trajectory point and GeoSOT ST-code and then insert the new trajectory into the corresponding code node point by point. When a trajectory is to be deleted, the first step is to convert the geographical location and time information of the trajectory into corresponding GeoSOT ST-codes. Then, the trajectory ID of each code node is removed. Query operation is the focus of this paper. Specific instructions are presented in Section 4.

### *3.3. GeoSOT Space–Time Code for One Trajectory Point*

A trajectory point is usually represented as a triplet $p = (x, y, t)$, where $x$ and $y$ represent the latitude and longitude coordinates of the moving object, respectively, and $t$ represents the timestamp at each position. A trajectory is a series of chronologically ordered points, i.e., $T = <p_1, p_2, \dots, p_n>$ along with a moving object id. The purpose of the GeoSOT spatiotemporal index for trajectory data is to establish the relationship between a grid and trajectory point. The specific steps involved are as follows:

(1) The optimum level: The subdivision level can be determined according to specific application requirements. When not specified, the default is the 21st level. One can obtain the temporal scale $rt_n$ and spatial scale $rs_n$ of the nth level space–time cube from the resolution lookup table.

(2) The decimal value of the longitude, latitude and time grid:

First, the latitude and longitude of the trajectory point are converted into a degree-minute-second format, i.e., D°M′S″. The timestamp is then converted into a year, month, day and hour format, i.e., YMDH. Then, the longitude, latitude, and time are respectively converted into a GeoSOT space–time decimal value with Equations (1)–(3):

$$\mathrm{Code_{Lon}} = \begin{cases} \mathrm{LonD}/\,\mathrm{rs_n} & 1 <= n <= 9 \\ (\mathrm{LonD} * 64 + \mathrm{LonM})/\,\mathrm{rs_n} & 9 <= n <= 15 \\ (\mathrm{LonD} * 64 * 64 + \mathrm{LonM} * 64 + \mathrm{LonS})/\,\mathrm{rs_n} & 15 < n <= 21 \end{cases} \quad (1)$$

$$\mathrm{Code_{Lat}} = \begin{cases} \mathrm{Lat_D}/\,\mathrm{rs_n} & 1 <= n <= 9 \\ (\mathrm{Lat_D} * 64 + \mathrm{Lat_M})/\,\mathrm{rs_n} & 9 <= n <= 15 \\ (\mathrm{Lat_D} * 64 * 64 + \mathrm{Lat_M} * 64 + \mathrm{Lat_S})/\,\mathrm{rs_n} & 15 < n <= 21 \end{cases} \quad (2)$$

$$\mathrm{Code_{Time}} = \begin{cases} \mathrm{Y}/\,\mathrm{rt_n} & 1 <= n <= 6 \\ (\mathrm{Y} * 16 + \mathrm{M})/\,\mathrm{rt_n} & 6 <= n <= 10 \\ (\mathrm{Y} * 16 * 32 + \mathrm{M} * 32 + \mathrm{D})/\,\mathrm{rt_n} & 10 < n <= 15 \\ (\mathrm{Y} * 16 * 32 * 32 + \mathrm{M} * 32 * 32 + \mathrm{D} * 32 + \mathrm{H})/\,\mathrm{rt_n} & 15 < n <= 21 \end{cases} \quad (3)$$

(3) The GeoSOT ST-code of one trajectory point

The Z curve is applied to transform three-dimensional spatial data into a one-dimensional array and to represent the encoding of cubes [34]. The code is simply calculated by interleaving the binary representations of its coordinate values, which call for the same length of binary representations in each dimension.

The above three values ($Code_{Lon}$, $Code_{Lat}$, and $Code_{Lon}$) are transformed into n-bit binary sequences *LatB*, *LonB*, and *TimeB*, respectively. Then, the values are converted into the corresponding Morton code [35] by interleaving bits of the three binary sequences (see Figure 4). Since the maximum number of subdivisions per dimension is 21, a binary sequence of no more than 63 bits is attained after interleaving. This sequence can be stored as a 64-bit unsigned integer.

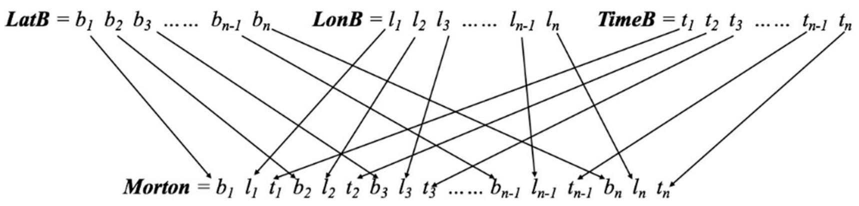
LatB = b1 b2 b3 …… bn-1 bn
LonB = l1 l2 l3 …… ln-1 ln
TimeB = t1 t2 t3 …… tn-1 tn
Morton = b1 l1 t1 b2 l2 t2 b3 l3 t3 …… bn-1 ln-1 tn-1 bn ln tn

**Figure 4.** Encoding procedure of the GeoSOT ST-code.

When the code is used as the primary key in the index table, point records falling in the same space–time cube are integrated as values under the key. Query operations can be executed by code matching. As another advantage of this code, when it is sorted, the two nodes adjacent to each other at the same level of local space and time can also be adjacent in storage.

## 4. Spatiotemporal Range Query Based on the GeoSOT ST-Index

### 4.1. Spatiotemporal Retrieve Model for Trajectory Data

The spatiotemporal range query takes a spatial query region and time interval as input. The goal is to find all data of the specified query range. When the spatiotemporal range query is performed based on the GeoSOT ST-index, the three-dimensional (latitude, longitude, and time) query can be converted into a one-dimensional query for the GeoSOT ST-code, which can greatly improve the query efficiency levels. To achieve this goal, it is necessary to establish the relationship between the spatial and temporal query range and GeoSOT ST-codes, i.e., by mapping the query range into GeoSOT space–time codes. The spatiotemporal retrieve model for trajectory data is shown in Figure 5.

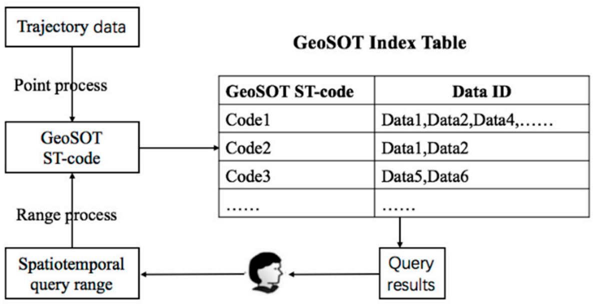

**Figure 5.** The retrieve model based on the GeoSOT ST-index.

## 4.2. *Query Processing*

Query processing is designed to convert the spatiotemporal query range into GeoSOT space–time codes. The key task is to select the appropriate level of GeoSOT ST-codes set corresponding to the query range. When the level is too high, the query range may map into many small grids (see Figure 6a), rendering retrieval conditions too trivial and reducing query efficiency. When the level is too low, the query range may reflect only a small proportion of the subdivision grids (see Figure 6b), leading ST-code retrieval to involve too much data that does not intersect with the query area [36]. Query accuracy cannot, in turn, be guaranteed. To solve the above problem, we propose mapping the query region to one to eight GeoSOT ST-codes to ensure that the actual query range is as continuous as possible and that the query result is as accurate as possible.

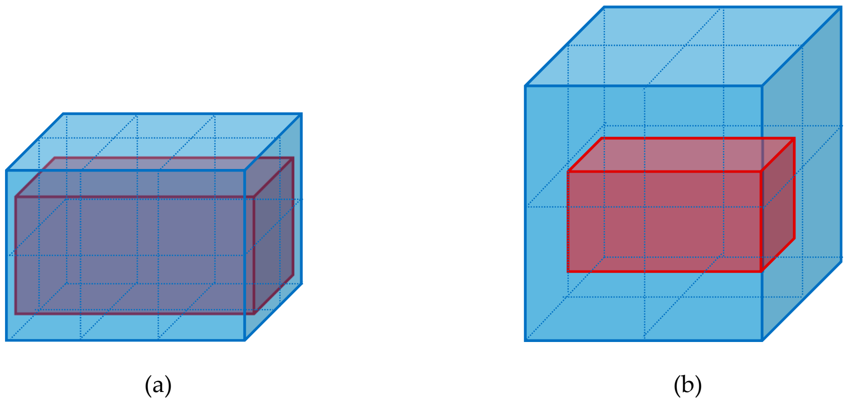

**Figure 6.** A comparison of covering the same space–time cube with subdivision grids of different sizes.

The main premise of query range processing is to find the minimum partition level *N* from the scale table, where the space–time cube is just greater than or equal to the query cube. The GeoSOT ST-code of the eight vertices in the query area is calculated at this level. Usually, a query cube may cover 1~8 space–time units of the Nth layer. The specific steps involved are as follows:

(1) The coverage of the query range along the longitude, latitude, and time dimensions

$$\text{Dis\_L} = \text{Lmax-Lmin} \tag{4}$$

$$\text{Dis\_B} = \text{Bmax-Bmin} \tag{5}$$

$$\text{Dis\_T} = \text{Tmax-Tmin} \tag{6}$$

(2) The optimum level

*Dis_L*, *Dis_B*, and *Dis_T* are compared, and the maximum value is selected as the granularity $\delta$ of the query cube, i.e., $\delta = max(Dis\_L, Dis\_B, Dis\_H)$. Assuming that *N* is the recommended level, the scale lookup table is *K* as in Table 2, *n* is a level of table *K*, and *scale(n)* is the spatiotemporal scale of the Nth layer. Then, recommended rules for the optimum level are given by Equations (7) and (8).

$$\text{if } \delta = \text{scale}(n),\ n \in K,\ \text{then } N = n. \tag{7}$$

$$\text{if } \text{scale}(n+1) < \delta < \text{scale}(n),\ n \in K,\ \text{then } N = n+1 \tag{8}$$

(3) Space–time codes covering the entire query cube

After determining level *N*, space–time codes of eight vertices of the query cube are calculated according to Equations (1)–(3). The GeoSOT space–time codes obtained after de-duplication are then treated as the final retrieval condition.

## 5. Performance Results and Discussion

To evaluate the effectiveness and scalability of the GeoSOT spatiotemporal index, the spatiotemporal query method based on the GeoSOT spatiotemporal index is applied to the nonrelational database MongoDB. In the era of big data, nonrelational databases are becoming more widely used. MongoDB, as a document-based nonrelational database, is chosen in this paper because it supports a flexible design pattern and has a rich data format suitable for trajectory data storage. In addition, the built-in spatial index of MongoDB can be used to draw comparisons to our proposed index.

The dataset used for our experiments is a GPS trajectory dataset collected by Microsoft [37,38]. This dataset includes approximately 10 million GPS trajectories of 10,357 taxis in Beijing. The MongoDB database was deployed in the single-node mode on a physical server. The server was equipped with one CPU (2 GHz Intel Core i5 CPU), 8 GB of RAM, and a 256 G hard disk. The server ran on the MacOS10.13.2 operating system.

We created a single trajectory data table in MongoDB and imported all of the experimental data. Each trajectory point was treated as an independent document and recorded the following fields: taxi ID, longitude, latitude, timestamp, GeoSOT ST-code, speed, and direction. The GeoSOT ST-index was implemented by establishing a B-tree for the GeoSOT ST-code field.

### *5.1. The Efficiency of GeoSOT ST-Code Generation*

This experiment mainly evaluated the space–time code generation time of the 10th~20th level for the same data volume. The experimental results show (Figure 7) that the encoding time tended to increase with the partition level. However, at any level, the conversion of ST-code and trajectory data was completed within 3 microseconds. The encoding time can be neglected in practical applications.

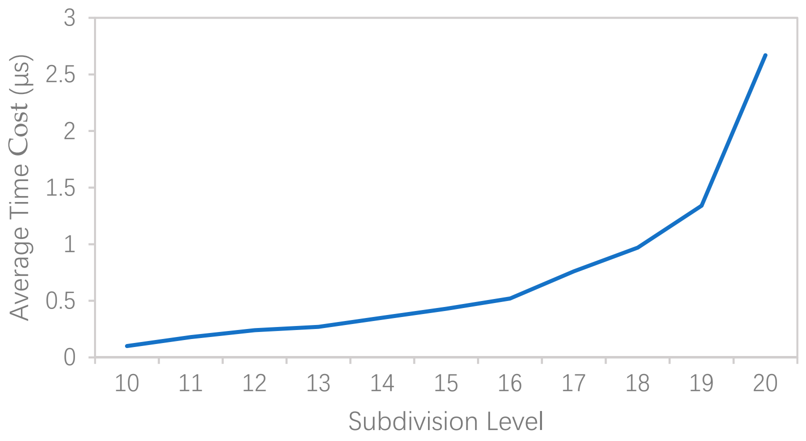

**Figure 7.** Average Encoding Efficiency.

## 5.2. Comparisons of Spatiotemporal Range Query Efficiency Levels

This experiment verified the effectiveness of the GeoSOT ST-index for range queries. We compared the GeoSOT ST-index with the other two composite indexes. One composite method was realized with three independent indexes: a B-tree index on latitude, a B-tree index on longitude, and a B-tree index on time. The other method was realized with two independent indexes: a 2D location index on the longitude and latitude fields (a built-in spatial index in MongoDB) and a B-tree index on time. The spatiotemporal range query performances of the three indexes were compared using the same amount of data. We randomly selected points in Beijing and use them as a north–south corner to generate rectangles of 0.01° × 0.01°, 0.02° × 0.02°, 0.04° × 0.04°, and 0.08° × 0.08°. Time intervals were set to 1 h, 2 h, 4 h, and 8 h. We performed 50 query tests on each spatiotemporal range and calculated the average time. The experimental results are shown in Figure 8.

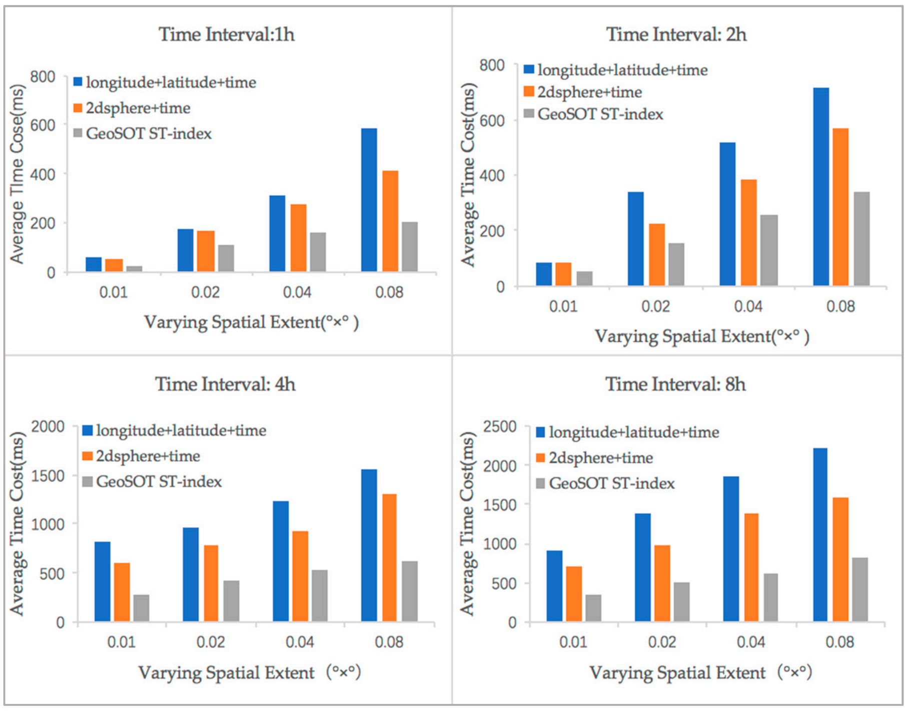

**Figure 8.** Comparisons of spatiotemporal query efficiency.

From the results of the effectiveness test, the GeoSOT ST-index performs better than the other two indexes. On average, the proposed index consumes approximately 40% less time than the MongoDB spatiotemporal composite index. In addition, an increase in the query range has less of an effect on the query efficiency of the GeoSOT ST-index and more of an impact on the other two composite indexes. This phenomenon indicates that composite indexes can only filter one-dimensional data (space or time) at a time while other dimensions can only be filtered afterward. As the query area expands, a large number of records must still be filtered after first filtering by the primary index. The GeoSOT ST-index reduces the number of three-dimensional queries made in one dimension, filtering latitude, longitude, and time simultaneously. The GeoSOT ST-index is especially good at dealing with range queries of large spatial ranges and long time intervals.

## 5.3. Comparisons of Scalability Performance

This experiment was carried out to evaluate the GeoSOT ST-index's scalability and to demonstrate the extension of query consumption time observed under different data volumes. We randomly selected the time period of interest "2011-03-07 09:29:08"~"2011-03-07 15:01:09" and a rectangular bounding box of interest with coordinates (116.612857 °E, 39.856973 °N) from the lower left corner and coordinates (116.692943 °E, 39.937059 °N) from the upper right corner. We fixed this time and space range and conducted query tests under different data volumes.

The results are shown in Figure 9. Under a fixed query range, as the amount of query data increased, the three indexes consumed more query time, but the GeoSOT ST-index showed a more gradual increase in the amount of time consumed. When the amount of data was increased 10-fold, the query time remained stable. This scalability can be attributed to the fact that the essence of spatiotemporal querying with the GeoSOT ST-index is to perform a one-dimensional search on a B-tree, which exhibits O(logN) time complexity. Additionally, since the GeoSOT ST-index is implemented by encoding each track point record and by building a B-tree for the space–time code in the database, space complexity is defined as O(N) where N is the number of track points.

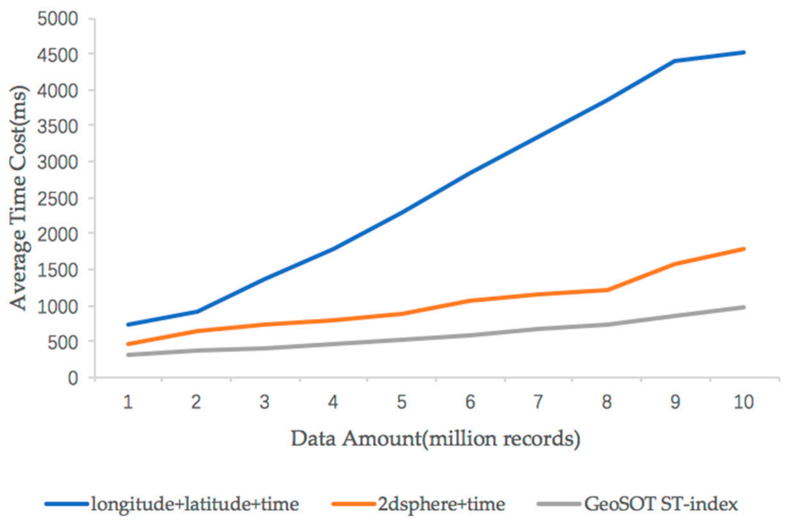

**Figure 9.** Comparisons of scalability performance.

It can be inferred that when the amount of data increases to one billion or even one trillion, the GeoSOT ST-index will be much more efficient than the other indexes. We thus think our proposed index meets the current requirements of massive trajectory data management applications.

## 6. Conclusions and Directions for Future Work

This paper proposes a time-space partitioning-based indexing method for massive trajectory data. This method establishes a global spatiotemporal indexing model by extending the GeoSOT subdivision model. In theory, this method can index trajectory data for small towns to the global scale with a time span of 128 years. We applied the index model to the NoSQL database MongoDB by combining the GeoSOT ST-code and B-tree index. The index has the advantages of being easy to integrate with existing database systems and offering efficient range query capacities through standard SQL. In addition, the cost of the index update is low. When data are updated, one must only add or remove data table records. Through experimental comparisons, we found that both the range query efficiency and scalability of the GeoSOT ST-index are superior to those of the other two spatiotemporal composite indexes, especially when the query scale is large in both space and time.

Future work will focus on more complex queries of the index model proposed in this paper. More comparisons will be drawn to existing solutions, such as ArcGIS and GeoMesa, to better assess the performance of the proposed solution.

**Author Contributions:** Chunyao Qian conceived, designed, and performed the experiments and wrote the manuscript; Chengqi Cheng and Chao Yi supervised the study; and Guoliang Pu offered helpful suggestions and reviewed the manuscript; Xiaofeng Wei revised the manuscript critically; Huangchuang Zhang polished the language. All authors have read and approved of the submitted manuscript, have agreed to be listed and have accepted this version for publication.

**Funding:** This research was supported by the National Key Research and Development Program of China (Grant No. 2018YFB0505300 and Grant No. 2017YFB0503703) and the National Natural Science Foundation of China (Grant No. 41801301).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zheng, Y. Trajectory Data Mining: An Overview. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 29. [CrossRef]
2. Ding, Z.; Huang, G. Real-time traffic flow statistical analysis based on network-constrained moving object trajectories. In Proceedings of the International Conference Database & Expert Systems Applications (DEXA), Linz, Austria, 31 August–4 September 2009.
3. Wang, X.; Liu, Z.; Jia, Y. A Rush-Hour Vehicles Scheduling Strategy in Online Car-Sharing System Based on Urban Trajectory Data Analysis. In *International Conference on Internet of Vehicles*; Springer: Cham, Switzerland, 2017.
4. Renso, C.; Baglioni, M.; de Macedo, J.A.F.; Trasarti, R.; Wachowicz, M. How you move reveals who you are: Understanding human behavior by analyzing trajectory data. *Knowl. Inf. Syst.* **2013**, *37*, 331–362. [CrossRef]
5. Deng, K.; Xie, K.; Zheng, K.; Zhou, X. Trajectory Indexing and Retrieval. In *Computing with Spatial Trajectories*; Springer: New York, NY, USA, 2011.
6. Parent, C.; Spaccapietra, S.; Renso, C.; Andrienko, G.; Andrienko, N.; Bogorny, V.; Theodoridis, Y. Semantic trajectories modeling and analysis. *ACM Comput. Surv.* **2013**, *45*, 1–32. [CrossRef]
7. Xing, X.; Li, M.; Hu, W.; Huang, W.; Song, G.; Xie, K. A Spatial-temporal Topic Segmentation Model for Human Mobile Behavior. International Conference on Web-Age Information Management. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2014.
8. Jin, P.; Zhang, X.; Yue, L. NBR-tre: A Novel Spatio-Temporal Index for Urban Traffic Networks. *Geomat. Inf. Sci. Wuhan Univ.* **2010**, *35*, 147–151.
9. Jun, G.; Shengnan, K.; Qing, Z. An Efficient Trajectory Data Index Integrating R-tree, Hash and B*-tree. *Acta Geod. Cartogr. Sin.* **2015**, *44*, 570–577.
10. Zhu, Q.; Gong, J.; Zhang, Y. An efficient 3D R-tree spatial index method for virtual geographic environments. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 217–224. [CrossRef]
11. Chakka, V.P.; Everspaugh, A.; Patel, J.M. Indexing large trajectory data sets with SETI. In Proceedings of the Conference on Innovative Data Systems Research, Asilomar, CA, USA, 5–8 January 2003.
12. Song, Z.; Roussopoulos, N. SEB-tree: An Approach to Index Continuously Moving Objects. In Proceedings of the Mobile Data Management, International Conference, Melbourne, Australia, 21–24 January 2003.
13. Pfoser, D.; Jensen, C.S.; Theodoridis, Y. Novel approaches in query processing for moving object trajectories. In Proceedings of the International Conference on Very Large Data Bases, Cairo, Egypt, 10–14 September 2000; pp. 395–406.
14. Li, G.; Tang, J. A New R-tree Spatial Index Based on Space Grid Coordinate Division. In Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011), Melbourne, Australia, 19–21 November 2011.
15. Kwon, D.; Lee, S.; Lee, S. Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree. In Proceedings of the Third International Conference on Mobile Data Management, Singapore, 8–11 January 2002.
16. Xiong, X.; Aref, W.G. R-trees with Update Memos. In Proceedings of the International Conference on Data Engineering, Atlanta, Georgia, 3–7 April 2006.
17. Ding, R.; Meng, X. A quadtree based dynamic attribute index structure and query process. In Proceedings of the International Conference on Computer Networks & Mobile Computing, Beijing, China, 6–19 October 2001.
18. Huang, M.; Peng, H.; Xia, L. A grid based trajectory indexing method for moving objects on fixed network. In Proceedings of the International Conference on Geoinformatics, Beijing, China, 18–20 June 2010.
19. Guan, X.; Bo, C.; Li, Z.; Yu, Y. ST-hash: An efficient spatiotemporal index for massive trajectory data in a NoSQL database. In Proceedings of the 2017 25th International Conference on Geoinformatics, Buffalo, NY, USA, 2–4 August 2017.
20. Zheng, K.; Shang, S.; Yuan, N.J.; Yang, Y. Towards efficient search for activity trajectories. In Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, Australia, 8–12 April 2013.

21. Ilic, M.; Spalevic, P.; Veinovic, M. Inverted index search in data mining. In Proceedings of the 2014 22nd Telecommunications Forum Telfor (TELFOR), Belgrade, Serbia, 25–27 November 2014.
22. Wang, K.; Chen, N.; Chen, Z. Spatio-Temporal Indexing Method of Big Trajectory Data Based on MongoDB. *Comput. Syst. Appl.* **2017**. [CrossRef]
23. Yang, S.; He, Z.; Chen, Y.P.P. GCOTraj: A storage approach for historical trajectory data sets using grid cells ordering. *Inf. Sci.* **2018**, *459*, 1–19. [CrossRef]
24. Qi, K.; Cheng, C.; Hu, Y.N.; Fang, H.; Ji, Y.; Chen, B. An Improved Identification Code for City Components Based on Discrete Global Grid System. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 381. [CrossRef]
25. Li, S.; Cheng, C.; Chen, B.; Meng, L. Integration and management of massive remote-sensing data based on GeoSOT subdivision model. *J. Appl. Remote Sens.* **2016**, *10*, 034003. [CrossRef]
26. Lv, X.; Cheng, C.; Xi, F. Study on Geographic Network Address of Geospatial Big Data Storage Management. *Geogr. Geo-Inf. Sci.* **2015**, *31*, 1–5.
27. Xi, F.; Cheng, C.; Chen, D.; Dong, F. An efficient hierarchical data placement algorithm for massive spatial data storage systems. In Proceedings of the Geoscience Remote Sensing Symposium, Melbourne, Australia, 21–26 July 2013.
28. Zhai, W.; Zhe, Y.; Lin, W.; Wu, F.; Cheng, C. The nonsql spatial data management model in big data time. In Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium, Milan, Italy, 26–31 July 2015; pp. 4506–4509.
29. Tong, X.; Cheng, C.; Wang, R.; Ding, L.; Zhang, Y.; Lai, G.; Chen, B. An Efficient Integer Coding and Computing Method for Multiscale Time Segment. *Data Knowl. Eng.* **2019**, *119*, 123–138. [CrossRef]
30. Cheng, C.; Tong, X.; Chen, B.; Zhai, W. A Subdivision Method to Unify the Existing Latitude and Longitude Grids. *Int. J. Geo-Inf.* **2016**, *5*, 161. [CrossRef]
31. Cheng, C. *An Introduction to Spatial Information Subdivision Organization*; Science Press: Beijing, China, 2012.
32. Kilimci, P.; Kalipsiz, O. Indexing of spatiotemporal Data: A comparison between sweep and z-order space filling curves. In Proceedings of the International Conference on Information Society, London, UK, 27–29 June 2011.
33. Su, Y.T.; Bethel, J.; Hu, S. Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. *ISPRS J. Photogramm. Remote Sens.* **2016**, *113*, 59–74. [CrossRef]
34. Jiang, H.; Kang, J.; Du, Z.; Zhang, F.; Huang, X.; Liu, R.; Zhang, X. Vector Spatial Big Data Storage and Optimized Query Based on the Multi-Level Hilbert Grid Index in HBase. *Information* **2018**, *9*, 116. [CrossRef]
35. Alis, C.; Boehm, J.; Liu, K. Parallel Processing of Big Point Clouds Using Z-Order Partitioning. In Proceedings of the International Archives of the Photogrammetry Remote Sensing, Prague, Czech Republic, 12–19 July 2016.
36. Jin, A.; Cheng, C.Q.; Song, S.H.; Chen, B. Regional Query of Area Data Based on Geohash. *Geogr. Geo-Inf. Sci.* **2013**, *29*, 31–35.
37. Yuan, J.; Zheng, Y.; Xie, X.; Sun, G. Driving with knowledge from the physical world. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011.
38. Yuan, J.; Zheng, Y.; Zhang, C.; Xie, W.; Xie, X.; Sun, G.; Huang, Y. T-drive: Driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; pp. 99–108.