Article

A Distributed Hybrid Indexing for Continuous KNN Query Processing over Moving Objects

Imene Bareche * and Ying Xia

Abstract: The magnitude of highly dynamic spatial data is expanding rapidly due to the instantaneous evolution of mobile technology, resulting in challenges for continuous queries. We propose a novel indexing approach model, namely, the Velocity SpatioTemporal indexing approach (VeST), for continuous queries, mainly Continuous K-nearest Neighbor (CKNN) and continuous range queries using Apache Spark. The proposed structure is based on a selective velocity partitioning method, i.e., since different objects have varying speeds, we divide the objects into two sets according to the actual mean speed we calculate before building the index and accessing data. Then the adopted indexing structure base unit comprises a nonoverlapping R-tree and a two dimension grid. The tree divides the space into nonoverlapping minimum bounding regions that point to the grids. Then, the uniform grid stores the object data of leaf nodes. This access method reduces the update cost and improves response time and query precision. In order to enhance performances for large-scale processing, we design a compact multilayer index structure on a distributed setting and propose a CKNN search algorithm for accurate results using a candidate cell identification process. We provide a comprehensive vision of our indexing model and the adopted query technique. The simulation results show that for query intervals of 100, the proposed approach is 13.59 times faster than the traditional approach, and the average time of the VeST approach is less than 0.005 for all query intervals. This proposed method improves response time and query precision. The precision of the VeST algorithm is almost equal to 100% regardless of the length of the query interval.

Keywords: continuous KNN query; moving object monitoring and management; distributed spatiotemporal indices; geospatial data systems; parallel processing

Citation: Bareche, I.; Xia, Y. A Distributed Hybrid Indexing for Continuous KNN Query Processing over Moving Objects. https://doi.org/10.3390/ijgi11040264

Publisher's Note

School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; xiaying@cqupt.edu.cn
* Correspondence: l201610003@stu.cqupt.edu.cn

1. Introduction

The widespread use of built-in global positioning system (GPS) devices has been directed to the expansion of location-based services (LBS) [1] and subsequently toward an exponential growth in the real-time LBS market [2,3]. Recent advancements concentrating on big spatiotemporal data and spatial computing such as smart cities [4], environmental monitoring and evaluation [5], location-based services [6–8], and the Internet of Things (IoT) rely on effective georeferenced data management. Geospatial data comprise information that are collected across both time and space; this information includes location, time, and eventually velocity information [9]. Different sensors transmit large scales of data at very high frequencies, resulting in large-scale dynamic data processing in real time, introducing the concept of continuous indexing and query. The continuous spatial query approaches should support a variety of complex searches using advanced indexing techniques. A series of time extensions with the dual representation of object-based, field-based, and geographic data was proposed in the early GIS modeling methods [10]. Hence, efficient continuous querying over moving objects has evolved as an essential process for numerous spatial computing applications due to the advent of mobile and ubiquitous computing. Spatiotemporal queries are helpful in different scenarios such as traffic control approaches, geographic information systems (GIS), and location-aware objects [11]. In this paper, we study the continuous K-nearest neighbor query (CKNN), an essential class of spatiotemporal queries that explores the K-nearest neighbors (KNNs) among a set of moving objects at each timestamp. An example of a CKNN query is to find two nearby pedestrian taxis based on the provided velocity and location of pedestrian traffic. In the past decade, most CKNN query approaches were integrated into a spatial indexing structure and discrete-time intervals, which returned the kNN set based on their location at the querying generation time. This instant search based on distance static computing might return an outdated resultset because the objects might be moving in the meantime. However, their locations were not updated along with the query evaluation, which reduced the accuracy of the query results. Therefore, a new continuous query processing algorithm is required to process CKNNs, taking into account the continuous updates [12].

Because location-aware devices and services are distributed across a time-aware spatial system, many CKNN queries are likely to be processed simultaneously. Reducing server performance is easy, and answering queries is long. Due to the real-time nature of locationaware applications, significant delays make answering queries useless. Therefore, new query processing algorithms that handle performance and scalability must include another set of CKNN queries. Many existing solutions to the KNN query object transfer problem were not developed to handle the large amount of data.Therefore, the indexing construction costs are expensive and impact the queries' response time subsequently, as it certainly follows a centralized setting where both query update maintenance and processing take place [13,14]. Continuous queries for data streams in spatial contexts analyzed by existing works were mainly based on Boolean bypass or approximate techniques, giving a random number or approximate result [15]. The CKNN query processing technology in database research is a concern that many researchers emphasize. It has many real-time applications, such as traffic distribution control, digital battlefield, personal navigation services, and other systems connecting mobile phone users. Therefore, the access to spatial information and spatial objects is limited to querying objects over a road network [16], which involves many challenges mainly related to the complexity of the spatial network features such as node identification and object pruning for query evaluation.

The primary purpose of this study is to enhance the efficiency and precision of a CKNN query over moving objects using distributed spatial indices. In addition, this work aims to effectively extract object information from distributed indices based on practical algorithms and efficient pruning technology to manage continuous query challenges. This paper will investigate and optimize CKNN query performances to avoid the limitations of the previous work mentioned above. For the experimental study, we considered different scenarios, the distance between two objects is defined as the shortest path between them in the network. The CKNNs set of a query point for a time stamp must be fully defined. In order to significantly reduce the iterative search and computation cost, we designed a method that uses the relative velocity information of the moving object and sets a number of iterations limited to two. The distance between two moving objects in every timestamp is presented only as a timeline operation and is easy to calculate using this approach. Our work expiates the significant deficiencies of the past related works and provides a more accurate and efficient method for the CKNN problem. The outline of this paper is as follows:

- We propose a novel indexing approach, namely the Velocity SpatioTemporal indexing approach (VeST), for continuous querying, mainly Continuous K-nearest Neighbor (CKNN) and continuous range queries.
- We design a compact multilayer index structure on a distributed setting and propose a CKNN search algorithm for accurate results using a candidate cell identification process.
- We provide a comprehensive vision of our indexing model and the adopted querying technique.
- We conducted a comprehensive set of experiments, compared our results with existing approaches, and employed different distribution techniques.

The rest of this paper is organized as follows. We review the related work for continuous KNN queries and moving objects in road networks in Section 2. Section 3 describes the preliminaries and the proposed approach. Section 4 presents the velocity spatiotemporal indexing approach, including the architecture, the index building phase, and the distributed query processing. Section 5 covers the dataset and simulation results on the performance of our method. In Section 6, we conclude the paper with directions for future work.

2. Literature Review

Indexing and querying paradigms of moving objects (MO) can be categorized into various classes according to the index structure base unit, such as grid-based indices, tree-like indices, and hybridized indices [17]. The intricacy of location-related approaches, in general, and continuous MO processing is affiliated with various factors such as data and variability. The indexed data are more volatile continuously, and there is a significant amount of data. Data autocorrelation transformations in the spatiotemporal element of moving objects transpire smoothly over time, excluding the constraints of the work related to CKNN, including the velocity uncertainty that is present in many of the works on moving objects' uncertain velocity, which concentrate solely on Euclidean spaces [18,19]. However, wheb assessing objects over a road network, the accuracy of location [20] concerns additional repetitive query re-evaluation. A fuzzy amount of CKNN search iterations occurs, and a significant amount of search algorithms are based on an unspecified number of iterations to find the area of the KNN, which results in an additional communication cost [18] in a distributed environment. The skewed distribution of data over nodes lessens the executions due to the nonuniform distribution over space.

A continuous K-nearest neighbor (k-NN) query supports multiple location-based services and continuously returns K-nearest object data to the query surroundings. Many of the current approaches to this issue have concentrated on centralized settings that indicate less scalability to address significant and scattered datasets. In [21], the authors proposed an efficient and distributed solution to kNN queries, so that objects could be moved to process more extensive data. The proposed solution included a new networkbased index called the Block Grid Index (BGI), and a BGI-based distributed KNN query method. BGI is a distributed layered grid-based in-memory index for KNN queries built within STORM. It starts from the assumption that moving objects belong to a region of interest that is partitioned in equal-sized cells without overlap, and each cell indexes its moving objects. There is no predefined pattern for the movement of objects. There are a minimum and maximum predefined number of objects per block; each block has a number N. There are a few advantages to their approach. BGI is easy to set up and maintain in a distributed environment. The querying algorithm can return the results after two search iterations, improving k-NN query performance. The effectiveness of their solution has been proven through extensive trials on millions of nodes.

The focus of many applications with dynamic objects is the processing of k-NN queries. Most current approaches are designed for centralized settings where queries are processed on the same server to address this issue. It is challenging, if not inconceivable, to scale the distributed setup to handle the large amount of data and synchronization queries that are increasingly found in such applications. To address this issue, Ziqiang et al. [22] proposed a set of explanations that could sustain scalable distributed processing of k-NN queries.
Foremost, they introduced a novel hybrid index structure called the Dynamic Stripe Index
(DSI) that better adjusted to various data distributions. The structure was divided into clusters, making it suitable for distributed processing. They also recommended the DSIbased K-NN Search Algorithm (DKNN). DKNN is more efficient and predictable than traditional methods, as it avoids the uncertain number of potentially expensive repetitions. DSI and DKNN were executed on top of the Apache S4, an open-source distributed streaming medium. In order to study the features of DSI and DKNN, they performed a comprehensive empirical study to examine the framework and compare it with three fundamental indexing and querying approaches. The experimental results showed that their proposal was highly scalable and outperformed competing methods.

Sensor networks produce a large scale of highly dynamic constantly updated data sent as packets in the data flow. The data stream's high frequency and continuous nature make it difficult to learn from basic observations. The article by Bolelang et al. [23] provided an upto-date overview of the visual analysis of geographic and global flow data and suggested a framework established by the gaps specified in the review. The framework consisted of a data prototype that featured sensor monitoring data, a user model that handled user queries and organized domain knowledge, a design prototype for discovered patterns and their corresponding visualizations, as well as a visual model to process rendering data. The conceptual model concluded that the flow of sensor feedback required tools that could handle multivariate multiscale time-series displays. Design models showed that the numerous valuable models combined proportions, deviations, and data.

The user model emphasized the necessity of handling missing data, high-frequency inconstancies, and review changes.

Many objects and many constant questions characterize the space-aware environment. Both things and constant questions can change position over time. In the article by Xiaoping et al. [24], they investigated the issue of reviewing queries from the Convolutional
Neural Network (CNN) in the space-time database. To maintain the performance of CNN query answers, a Shared Execution Algorithm (SEA-CNN) was introduced. SEA-CNN combines additional assessments with joint implementation to reduce the cost of updating answers to questions. In incremental assessment, only questions affected by the movement of objects are reconsidered. Each query is associated with the search area based on the answers to the previous query to reduce evaluation time. Compatibility queries are grouped into a standard search table in the shared implementation model. Therefore, the issue of reviewing numerous queries is resolved by the local connection between the lookup table and the object table. SEA-CNN is also a generally usable framework. Foremost, SEA-CNN does not make any inferences regarding the motion of an object (e.g., speed, orbit). They offered a theoretical analysis of SEA-CNN regarding implementation costs, memory requirements, and the consequences of tunable parameters. Extensive experience showed that SE-CNN was more scalable and efficient than R-tree-based CNN technology regarding I-O number and CPU cost.

Wireless sensor networks have been widely used in numerous applications, such as environmental monitoring, manufacturing management, business asset management, transport automation, and the medical industry. In the article by Hua et al. [25], they examine an interesting issue. Continuous monitoring of k-mean assembly of sensor readings in large sensor networks. Given a set of sensors whose measurements evolve, they want to keep the average k of measurement constant. The optimization aims to reduce network reporting costs. The goal is to inform the data center of current readings while maintaining as many sensors as possible. They recommend a reading report tree, a classification framework for data collection, and analysis to address this issue. They are also developing several economic reporting methods by continuously monitoring k sources by reading the reporting tree. First, a standard method of sampling using a report-reading tree can provide a good quality perspective. Second, they suggested a method for setting boundaries that can almost guarantee quality. Finally, they tested a slow approach that could significantly reduce intermediate computation. To investigate the features of the proposed method, they evaluated systematic simulation using artificial datasets.

The continuous K nearest neighbor search process (CKNN) captures the nearest k objects for the provided set of moving objects. It gives consistent results in real time with objects and query points for monitoring the transfer. Current CKNN query processes typically have index maintenance, real-time result updates, and query costs that may not completely resolve the issue. To address this issue, Yu et al. [26] recommended an additional search algorithm. It uses the Random Estimation (RE) method to process CKNN queries for large quantities of moving objects. Combined with an incremental search algorithm for processing CKNN indexing and querying over a massive volume of moving objects, the RE approach could rapidly determine a suitable search area for a continuous query established on the prior outcomes of the query. Compared to other approaches targeting continuous querying in a two-dimensional area of interest, this strategy achieved a higher estimations accuracy for identifying the number of items in a specific area, significantly improving the continuous kNN query processing efficiency.

The most critical part of an emergency response system is time. Therefore, it is essential to immediately store new accurate data in the database, retrieve data quickly, maintain the chronological order and strengthen the system, all processed in real time. One more critical element of environmental threat monitoring is enabling real-time queries founded on modern features. We also need the fastest access to the latest data. In addition, real-time data collected from sensors requires swift and efficient control to detour central memory saturation. Servigne et al. [27] proposed indexing methods supporting queries that targeted contemporary data. The proposed indexing solution was for real-time data of fixed sensors. The second was real-time and space data collected through active sensors. Finally, they offered a general solution for managing memory saturation in real time according to the importance of the data.

The portion of available spatiotemporal data is proliferating because of the advancements in mobile data acquisition technology and location-aware devices. Real-time processing of big spatial data has evolved into one of the examination frontiers in Geographic Information Systems (GIS). It aims to reduce the complexity of updating data by employing specific distributed spatial indexing structures that handle highly dynamic data. In order to process continuous querying over spatial data streams, Zhang et al. [17] proposed an extension to Apache Storm, which is an open-source distributed real-time computation system. They employed a strategy of spatial joins between moving datasets with a secondary static distributed spatial index to process continuous queries. However, the proposed grid-based partitioning techniques focused only on 2-dimensional point data. In addition, such indexing techniques were ineffective in spatial applications because of their dynamic data and query distributions. We have proposed a novel indexing approach mode for range queries to address these issues. We design a compact multilayer index structure on a distributed setting and propose a CKNN search algorithm for accurate and efficient results.

3. Preliminaries

We illustrate some definitions and symbols utilized in this paper. Table 1 lists the main notations that are used throughout this paper. We assumed a setting where N objects in a two-dimensional space move with different velocities where X_Vel and Y_Vel are the velocity according to the X-axis and Y-axis, respectively. Moving objects were modeled as point objects, and the space in which they move was modeled as a two-dimensional Euclidean space. The objects frequently sent updates about their location to the indexing primary node. Updates were sent as a triplet (obj_ID, x_loc, y_loc), where obj_ID is the object's unique identification number, x_loc represents the longitude position, and y_loc represents the latitude position of the object.

Definition 1. ***Moving Objects, MO*** *: a moving object is represented by a discrete sequence of tuples in the form {obj_id,(x_loc , y_loc), (x_vel, y_vel), t)}, where obj_id is the object identifier, (x_loc, y_loc) represents the current location, (x_vel, y_vel) is the object movement velocity, and t is the current time.*

Definition 2. ***Updates****: In a dynamic spatial data processing context, moving objects continuously send their new location updates to keep them reflected in the indexing structure. In order to ensure accuracy, the index must be able to process large volumes of updates rapidly to provide high-precision responses, also with varying query interval lengths.*

Definition 3. ***Spatial Queries****: Spatial queries are mostly generated and used in a dynamic number of objects applications. Such requests by users are expected to be responded to based on real-time or near real-time data analysis. There are different types of spatial queries; in this work, we focus on continuous spatial kNN queries.*

(a) Query Point: A query point is defined by a user (query issuer) ID. The user must be located inthe index to determine its region and the kNN set.

(b) K-nearest Neighbor Query, kNN query: Given a query point in the space, a kNN query mustretrieve the k-th closest objects to the query point.

(c) Continuous K-nearest neighbor queries, CKNN query: Given a query point in the space, aCKNN query must continuously track the k objects that are the closest ones to the query point. The objects of interest, also called candidate objects, and/or the query point might be moving during the query evaluation, which means the locations are updated continuously.

Table 1. Notation Definitions.

Notation	Definitions
N	Number (Count) of moving objects in the space
MO	Moving object
obj_id	A moving object identifier
x_loc	Longitude position of an object

y_loc	Latitude position of an object
x_vel	Velocity according to X axis
y_vel	Velocity according to Y axis
t	Time

4. Proposed Methodology

This paper proposes a velocity-dependent indexing method for moving object data based on similar velocity classification using the standard deviation equation on the velocities distribution, resulting in two separate subspatial perspectives to avoid data skewness and unuseful search iterations. We designed a region-based index for constantly moving objects in each scene in a distributed environment. We proposed an algorithm to select dynamic query candidates in CKNN [28].VeST was customized by combining time–space indexing with speed. When a new query was generated, it was first evaluated using various phases through the index [24] in order to find the best search volume for the queries to reduce the cost of updating the index. The Velocity SpatioTemporal (VeST) index obtained a query result similar to the mapped spatial nodes. It used the custom input method to enter the list of objects and the corresponding tree. This approach further improved the index structure building performances, including the index construction time as well as the size, and reduced index update costs. Figure 1 represents the Sequence diagram of query processing. Where a CKNN query was generated, the Velocity classifier analyzed the objects' actual velocities to compute the standards deviation and define the threshold to launch a velocity-dependent partitioning. Then the query manager started processing the query over the global index structure first. The query was evaluated and updated accordingly over the distributed nodes of the index to generate the query response.

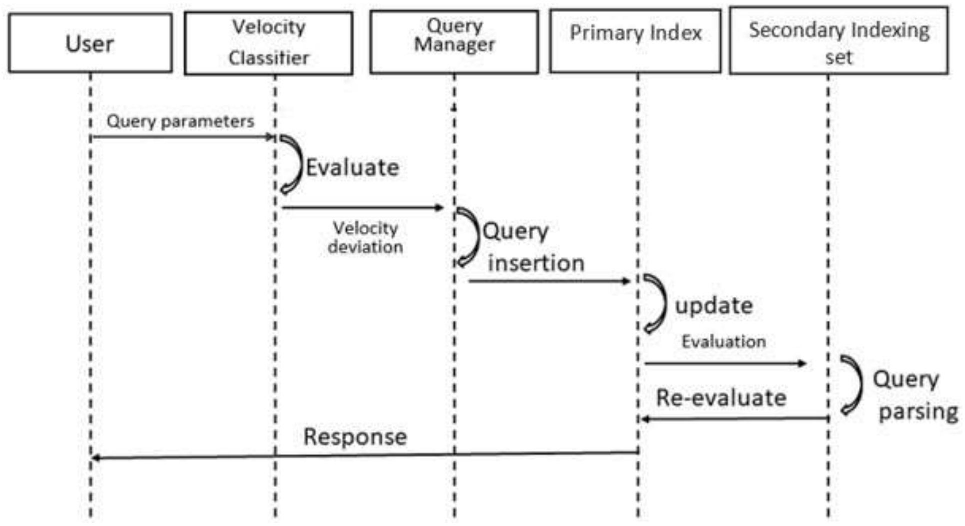
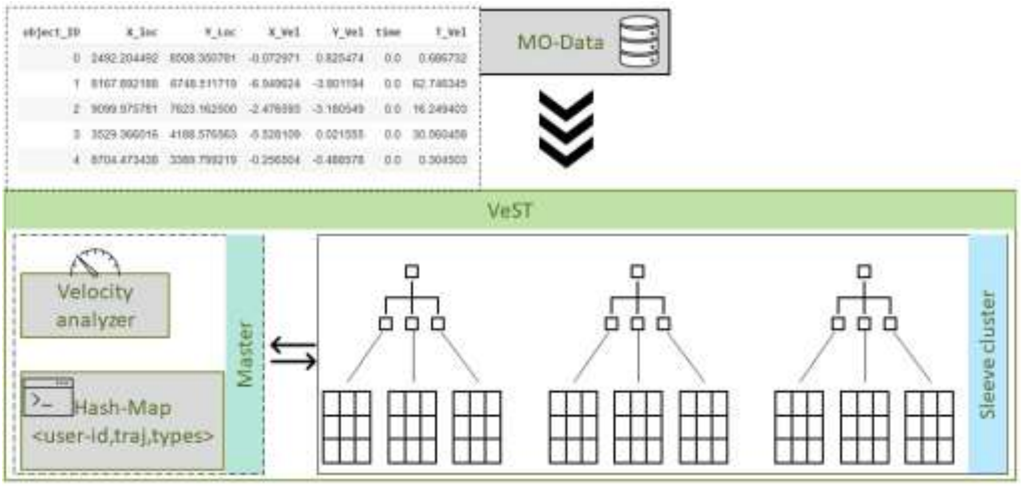


Figure 1. Sequence diagram of CKNN query processing.

5. Velocity Spatiotemporal Indexing Model

We have proposed a spatial index model for continuous querying. The proposed model is organized by combining spatiotemporal indexing with velocity. Figure 2 represents the architecture of the proposed velocity spatiotemporal index (VeST) for the moving object CKNN querying.



object_ID	X_loc	Y_loc	X_Vel	Y_Vel	time	T_Vel
0	2492.204492	8508.350781	-0.072971	0.825474	0.0	0.686732
1	8167.892188	6748.511719	-6.940624	-3.801194	0.0	62.748345
2	9099.975781	7623.162500	-2.478893	-3.160549	0.0	16.249400
3	3529.369016	4188.576563	-5.528109	0.021555	0.0	30.060458
4	8704.473438	3389.799219	-0.256504	-0.488578	0.0	0.304503

Figure 2. Architecture of velocity spatiotemporal index for moving objects.

5.1. Vest Architecture

Our velocity spatiotemporal index consisted of space–time and velocity data. It analyzed moving objects' velocity and then indexed their data across a hybrid distributed structure. VeST was designed to have two main parts, primary and secondary. A primary/secondary paradigm signifies that one server was configured to operate as the primary node. It was then directed to acquire all of the written queries. The primary node then performed and logged the queries, which were then dispatched to the secondary nodes to conduct and keep the identical data across all counterpart components. In the primary/secondary structure, the write functions were executed at the primary level and read functions at the secondary level [29]. Therefore, all search requests initially reached the primary node, a queue of submissions was preserved, and the read function was accomplished solely behind the fulfillment of the write operation. There is a common problem in a primary/secondary configuration, which is also witnessed when the primary node's queue becomes too large to be maintained. This architecture collapses, and the secondary nodes start behaving as the primary one.

5.2. Velocity-Based Partitioning Phase

The velocity-based portioning phase helps to reduce time costs and make the system more efficient as we presented and proved in our previous study [28]. The system evaluates the speed of objects for categorizing entities into different classes.

The velocity-based classification generated two classes, one was for the fast objects class, and the other was for a slow objects class. The selection was made according to a predetermined equation that estimated the velocity deviation of a real-time object established on the distribution of the objects. Total velocity (V_t) is calculated using Equation (1), where n is the total number of objects at a given time, and v_k is the velocity of the object. V_t is used to determine speed SP, which is specified in Equation (2). Finally, the SP is used to obtain σ using Equation (3), which is the threshold between fast and slow objects. This partition saves excessive access to entities in one place; in addition, it reduces the search period and splitting or merging costs.

$$V_t = \sum_{k=1}^{n} v_k \quad (1)$$

$$SP = \frac{\{V_t^2\}}{n} - \left\{\frac{V_t}{n}\right\}^2 \quad (2)$$

$$\sigma = \sqrt{SP} \quad (3)$$

Figure 3 illustrates the structure of the indexing model, its objects, their behavior, and operations. The *"DataAnalysis"* class contained the spatial attributes of the query and the velocity deviation, which were generated by the *"VelocityClassifer"* object in order to partition objects based on their velocities. The k parameter of a CKNN query was used to process iterative search operations over the index structure's partitions after the region's spatial split.

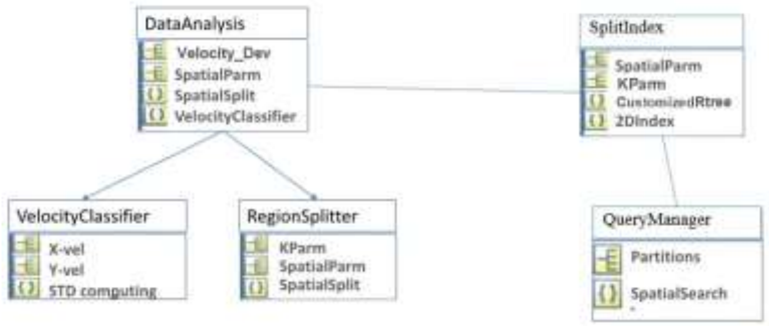


Figure 3. Conception of the indexing model.

5.3. Index Building Phases

The large scale of spatial data is continuously generated over time, and users can be of various types, such as taxis, people, or buses. Hence a hash map to store a key-value structure was integrated. Hash maps are appropriate for diminutive quantities of data and sustaining recurring transformations. It accumulates records of the territories that the user has visited and additional characteristics that establish corresponding data such as user type: (id, [trajectory], type). The hash map devotes an unrestrained search technique, a linear scan; hence, the time complexity of a query is O(m).

5.4. Distributed Query Processing

We proposed using a multilayer structure for distributed processing platform, as illustrated in Figure 4. Distributed processing modules consist of three layers, a first one for the primary indexing node, a second layer for the set of the secondary indexing set, and a spatial data partitioning layer. Individual nodes created their index for all locally reserved records in the secondary indexing layer. Therefore, each node had a shortlist of objects data. A query was initially broadcasted to every node, and then the derivatives were integrated. Therefore, each node had a more diminutive number of prolonged lists. Under the standard query evaluation procedure, a query was first routed to the node maintaining the list for a more concise list, which then dispatched its entire list to the node containing the main list [30].

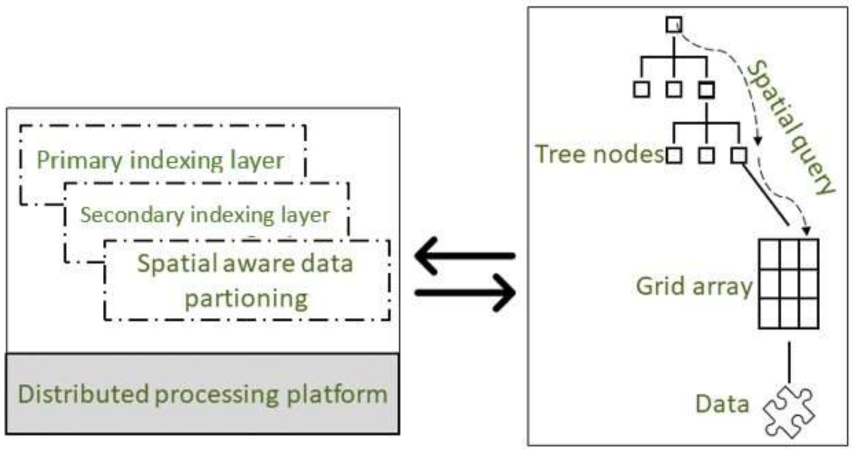


Figure 4. Multilayer structure of distributed processing platform.

6. Experimental Settings and Results

This section explains the proposed approach's simulation settings, exploratory data analysis, and evaluation results. We used an open-source taxi GPS tracking dataset [7]. The dataset contained more than 100,000 taxis' data within Shenzhen city with different records. The dataset contained an object ID, location on the x-axis, location on the y axis, velocity on the x-axis, velocity on the y-axis, and time when the update was issued.

6.1. Simulation Environment

The experiments were conducted on a cluster of nodes with a processing unit of Intel Core i7-8500y @ 3.00GHz and a Random Access Memory of 16 GB. The simulation setting employed during the exploratory period is outlined in Table 2.

Table 2. Configuration of simulation settings.

	Component	Description
1	Processing Unit	Intel Core i5-8500 @ 3.00GHz
2	RAM	16 GB
3	Operating System	Window 10
4	Integrated Development Environment	Jupyter notebook
5	Programming Language	Python
6	Python	v 3.6
7	Mathematical Functions Module	math V 3.9.1
8	Array Library	NumPy v 1.17.5
9	Data Analysis Library	pandas V 0.25.3
10	Plots Library	plotly V 4.4.1

6.2. Exploratory Data Analysis

Figure 5 shows the sampled locations of moving objects in the city. The location of taxis also sketched the outline of the city's road network. Figure 6 illustrates the objects after classifying them according to their velocities with regard to the actual velocity deviation. Figure 6a shows the visual representation of slow-moving objects with red dots, whereas Figure 6b shows the fast-moving objects with green dots.

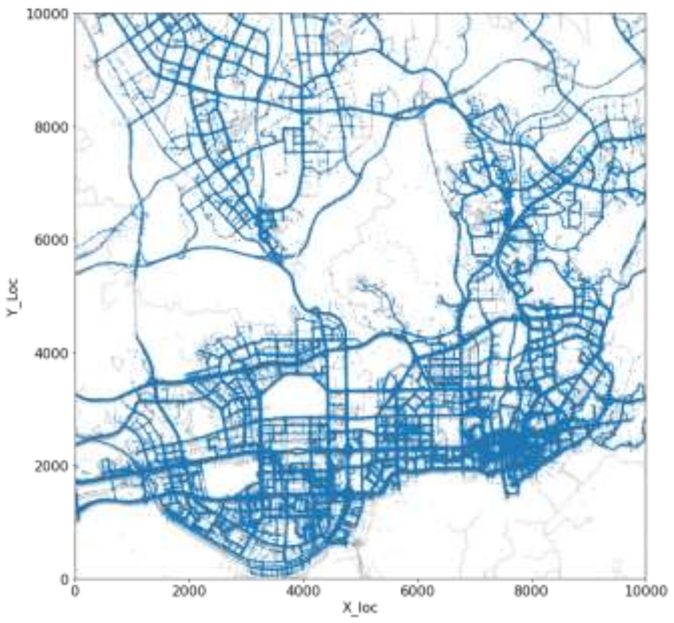
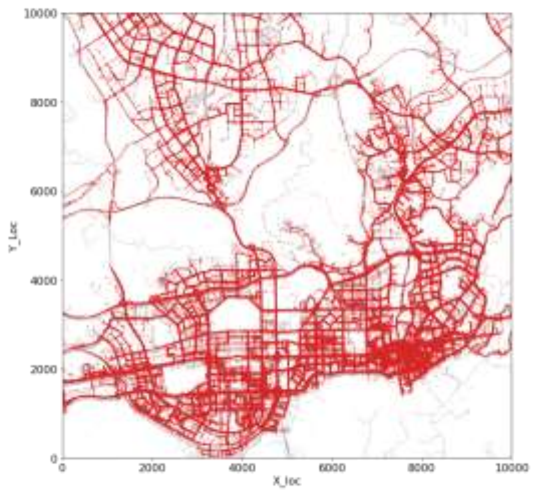
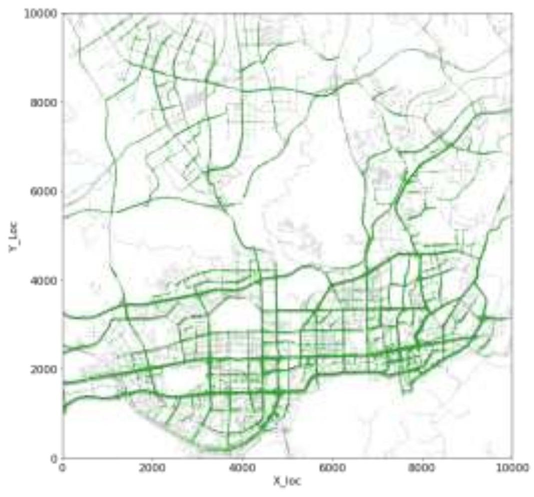


Figure 5. Sampled locations of moving objects in Shenzhen City road network.



(**a**) Slow-moving objects



(**b**) Fast-moving objects

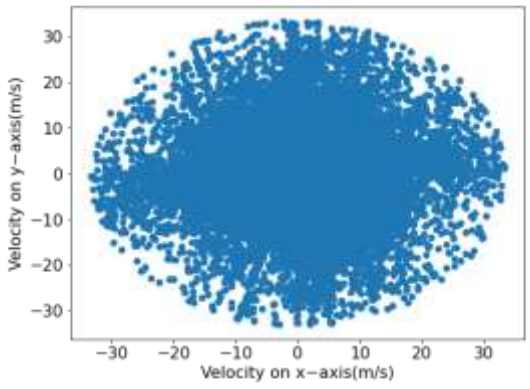
Figure 6. Sampled objects classified according to their velocity.

The total size of the data set used was 46.9 MB. The data were generated through OpenSourceMap with the space domain of 10,000 × 10,000 for Shenzhen City [31]. Table 3 shows the summary of the dataset. The dataset contained six columns and 1,025,486 rows, with only one row containing null values. The first column represented the ID of each object, where the minimum object ID was 0, and the maximum ID was 10928. The second and third columns were for the location X_loc and Y_Loc according to the X and Y axes, respectively, representing longitude and latitude; whereas, X_Vel and Y_Vel represented the velocity on the X and Y axes. The maximum velocity on the X-axis recorded was 33.25 and on the Y-axis was 33.19. The last column, "time", contained the tracking timestamps.

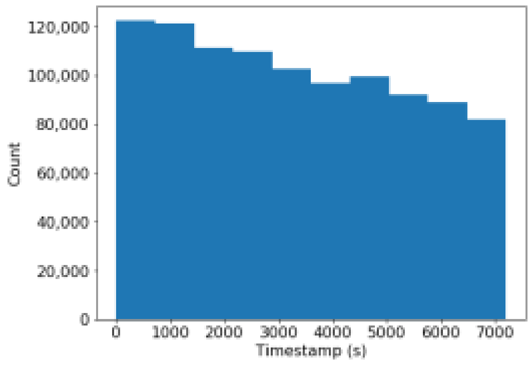
Table 3. Summary of Dataset.

	object_ID	X_loc	Y_loc	X_Vel	Y_Vel	Time
Mean	/	5631.91	3718.73	0.06	−0.04	/
STD	/	2326.78	2377.48	4.85	4.040	/
Min	0	0	2.23	−33.2	−32.97	0
Max	10,928	9991.1	9999.73	33.25	33.19	7199

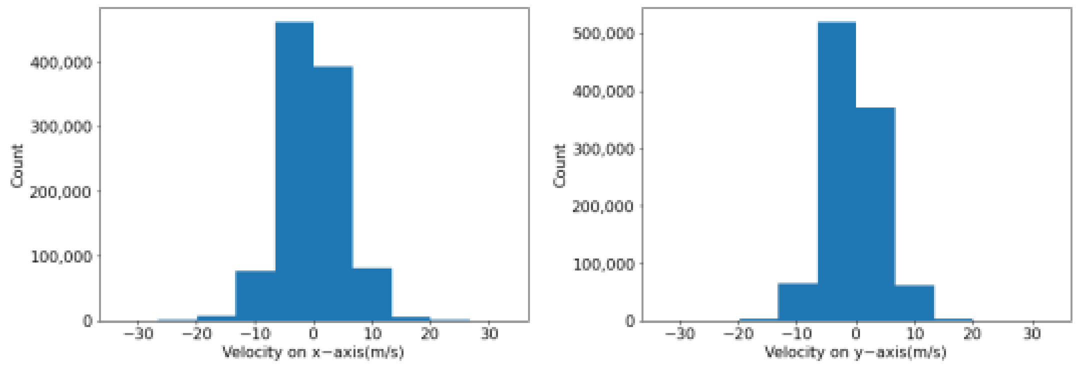
Figure 7 shows the distribution of different features of the dataset. Figure 7b shows the velocity distribution across the X and Y-axis. Most of the points lay between −30 and 30 m/s. Figure 7b shows the count of objects per time slot from 0 to 7199. It can be observed that the time between 0 to 1000 contained the highest number of objects and 6000 to 7000 contained the lowest number of objects. Figure 7c shows the count of objects according to the velocity on the X-axis. Figure 7d shows the count of objects according to the velocity on the Y-axis. In both the X and Y-axis velocity distribution, the maximum count was between −10 to 10.



(**a**) Velocity distribution



(**b**) Count of objects per time slot



(**c**) Count according to velocity on X-axis (**d**) Count according to velocity on Y-axis

Figure 7. Distribution of different features of the dataset.

6.3. Experimental Results

We conducted extensive experiments on the dataset presented in the previous section. We performed simulations and plotted various graphs to represent the proposed approach's performance from the perspective of the working hypotheses. Intending to investigate by how much the proposed approach was faster than the conventional approaches in terms of processing time, we compared the querying time for various query intervals for a CKNN search where k was set to 7. We compared the time when using the proposed approach and the Apache Spark to enhance the performances further and used the conventional approach. As can be seen in Figure 8 for query intervals of 100 (i.e., the number of queries we simulated in parallel was equal to 100), the proposed approach was 13.59 times faster than the competing approach. The effect of query interval length over the average time is depicted in Figure 8. The average time of the VeST approach was less than 0.005 s, as we can see, for all query intervals.

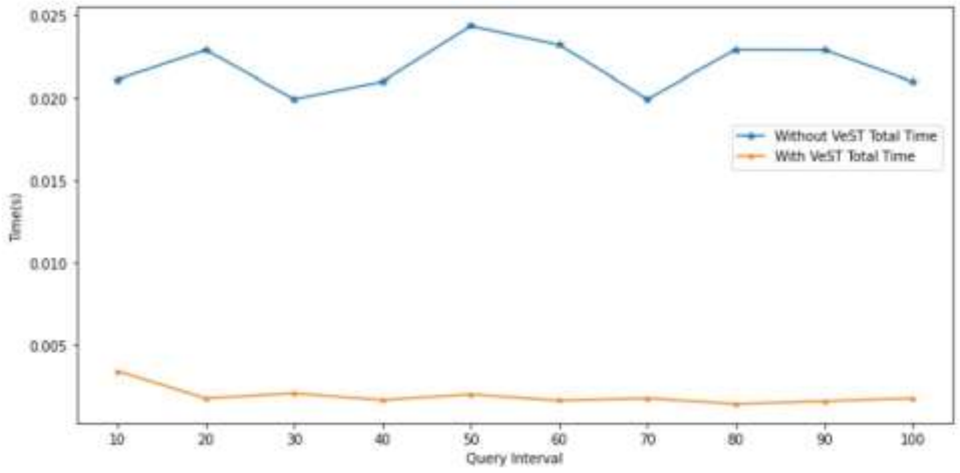


Figure 8. Effect of query interval length over query processing time.

We simulated three different data distribution patterns from the same dataset for our experimentation. In the foremost distribution pattern (Gaussian), 70% of the objects pursued the Gaussian distribution [32] across the network. We used Equation (4) to generate a Gaussian distribution for 70% of objects. The remaining objects were distributed uniformly. The second distribution pattern (Uniform) consisted of the objects that followed a uniform distribution across the network. In the third distribution pattern, objects followed the Zipf distribution. All the objects were normalized to a unit square.

$$\mathrm{gd}(x, \mu, \sigma) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}\left(-0.5(\frac{x-\mu}{\sigma})^2\right) \quad (4)$$

Figure 9 demonstrates the time of building the VeST as we altered the number of objects and their distribution pattern. If further parameters did not vary, the index construction time rose almost linearly with the increasing number of objects. As we made a better concentrated Gaussian pattern, there were additional split and merge processes in the R-tee layer for this one. As a result, the construction time was the highest among the three distribution patterns in most cases.

Figure 10 analyzes the impact of the query interval size on the CPU time of VeST and competing approaches. The proposal was compared with the IMA approach suggested by Mouratidis et al. [33] and with the CKNN approach proposed in [34]. The IMA method is based on iterative snapshot kNN search evaluations, unlike the VeST approach, where we avoid the unknown number of iterations. When objects' location updates occur, the IMA algorithm again evaluated the snapshot KNN query. In our experiments, the IMA update interval (UI) was set once to 5 time units and then to 10 time units to examine both cases and compare it with the VeST algorithm. The IMA algorithm with update intervals of 5 and 10 time units is termed IMA(UI = 5) and IMA(UI = 10), respectively. On the other hand, the CKNN algorithm divided the time interval into disjoint subintervals. These subintervals were evaluated successively in locating the KNNs of the query entity. The experiments showed that the CPU cost rises with the expanded query interval size for all the algorithms. This is because a significant query size derives additional objects to be considered for the continuous KNN search approach and assembles better discrete place updates of objects. Consequently, the IMA algorithm required more snapshot KNN queries (for both UI = 5 and UI = 10); therefore, the CPU time was higher in both cases UI = 5 and UI = 10. The KNN algorithm showed better performances compared to IMA because it divided the time into sequential timestamps; yet, it required more time for a complete KNN search and assumed that objects were moving at constant speeds, which was unrealistic. The simulation outcomes showed that the presented algorithm surpassed its competitors in all cases while continuously considering the dynamic speed feature.

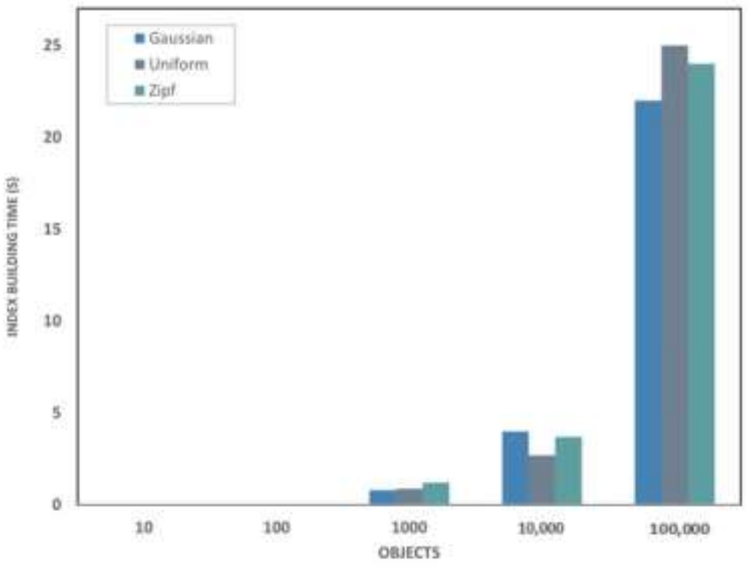


Figure 9. The construction time of VeST.

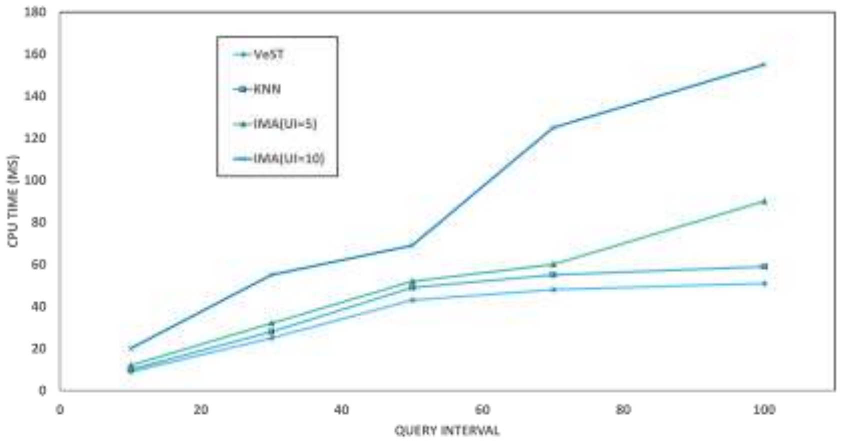


Figure 10. Effect of query interval length on CPU time.

Figure 11 illustrates the effect of the query interval size on the precision of the different approaches. The preciseness is the ratio of time units at which the retrieved continuous KNN query result is correct, as determined using Equation (5) where kNNreal is the number of the obtained resultset records number, and kNNreal is the actual number of the k nearest neighbors to the query point. In order to define the optimum k parameter value to avoid noise or underfitting problems, in this experimental scenario, we set k equal to 7 based on the standard deviation of objects to create an illustrative example.

$$Precision = \frac{\#(kNNget \cap kNNreal)}{\#kNNreal} * 100\% \qquad (5)$$

Due to the continuous location updates of objects, the precision was below 60% and 90% for IMA and kNN, respectively. It even reached 20% when the query interval was larger (UI = 10) because the object location updates in these approaches were supposed to be discrete. Therefore, two successive update time instances would obtain imprecise query outcomes. In contrast, the precision of the VeST algorithm was almost equal to 100% regardless of the length of the query interval.

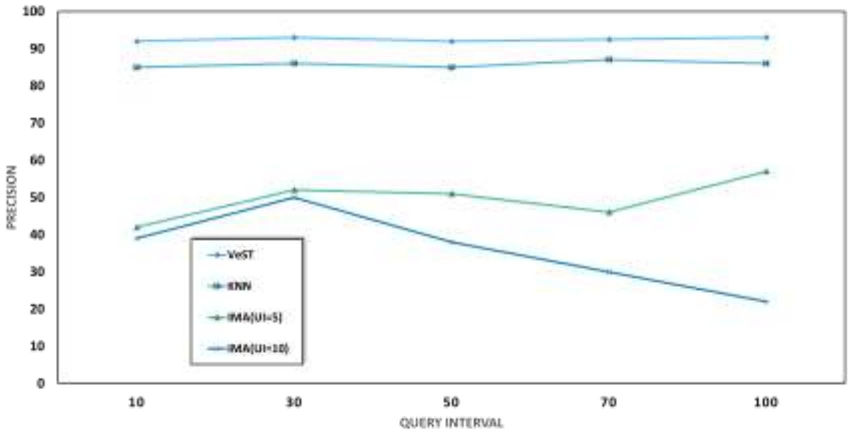


Figure 11. Effect of query interval length on precision.

7. Discussion

This paper explored challenges related to continuous queries over moving objects. The contributions of this paper are fourfold; first, presenting a novel approach for continuous querying; second, explaining the multilayer index structure; and third, the querying technique and the final step employed different distribution techniques to examine the effect of the proposed approach. This section discusses how the proposed framework addressed the identified gaps.

The first gap related to presenting a novel approach for continuous querying was addressed by proposing a novel indexing approach, namely the Velocity SpatioTemporal indexing approach (VeST), for continuous querying, mainly Continuous K-nearest Neighbor (CKNN) and continuous range queries. The second gap related to explaining the multilayer index structure was addressed by designing a compact multilayer index structure on a distributed setting and proposing a CKNN search algorithm for accurate results using a candidate cell identification process. The third gap related to the querying technique was addressed by providing a comprehensive vision of our indexing model and the adopted querying technique. The fourth and final gap related to employing different distribution techniques to examine the effect of the proposed approach. We conducted a comprehensive set of experiments, compared our results with existing approaches, and employed different distribution techniques. This proposed method reduced the update cost and improved the response time and query precision. The dataset was the same one in all figures but with different representations, which was a real-world dataset that was a taxi GPS tracking system recorded in Shenzhen City. We illustrated data features to provide a comprehensive data description. We have also represented the results of our proposal experiments on the dataset. We simulated three different data distribution models out of the same dataset to investigate the performances of the proposed approach for different data distributions with a varying number of objects.

Various factors can affect the performance of the proposed VeST approach. One factor is the number of nearest neighbors. The other major factor affecting the performance is the velocity of moving objects. The proposed approach is based on a velocity partitioning of the objects before the spatial indexing. Since the velocity feature is essential in our proposal and impacts the conduct of the indexing and the querying performance, we illustrated the dispersion of this information from different perspectives to show how it varies, which should be considered. In the future, analytical methods can be used to estimate the velocity distribution and hierarchical grids.

8. Conclusions

We proposed a novel indexing approach model, namely the Velocity SpatioTemporal indexing approach (VeST), for continuous querying, mainly Continuous K-nearest Neighbor (CKNN) and continuous range queries. We designed a compact multilayer index structure on a distributed setting and proposed a CKNN search algorithm for accurate results using a candidate cell identification process. We provided a comprehensive vision of our indexing model and the adopted querying technique. We conducted a comprehensive set of experiments, compared our results with existing approaches, and employed different distribution techniques to investigate the index structure building time in different scenarios. For query intervals equal to 100, the proposed approach was 13.59 times faster than the traditional approach. In addition, the average time of the VeST approach was less than 0.005 for all query intervals. This proposed method improved response time and query precision. The precision of the VeST algorithm was almost equal to 100% regardless of the length of the query interval. The simulation outcomes showed that the presented algorithm surpassed its competitors in all cases while continuously considering the dynamic speed feature. The effect of using analytical methods and hierarchical grids can be explored in the future to enhance the results.

Author Contributions: Conceptualization, Imene Bareche; methodology, Imene Bareche; software, Imene Bareche; validation, Ying Xia; formal analysis, Imene Bareche; investigation, Imene Bareche; resources, Ying Xia; data curation, Imene Bareche; writing—original draft, Imene Bareche; writing— review and editing, Imene Bareche and Ying Xia; visualization, Imene Bareche;validation, Imene Bareche and Ying Xia; supervision, Ying Xia; project administration, Ying Xia; funding acquisition, Ying Xia; All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by the Natural Science Foundation of China (41971365) and the key cooperation project of Chongqing municipal education commission of China (HZ2021008).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are openly available in [VPIndexer] at http://www.mathcs.emory.edu/~lxiong/aims/spindex/VPIndexer/data/sz/ ,referencenumber31 accessed on 20 November 2020.

Conflicts of Interest: The authors declare no conflict of interest regarding the design of this study, analyses, and writing of this manuscript.

Abbreviations

The following abbreviations are used in this manuscript:

LBS	Location-Based Services
GPS	Global Positioning System
MO	Moving Objects
KNN	K-nearest Neighbors
CKNN	Continuous KNN
IoT	Internet of Things (IoT)
GIS	Geographic Information System

BGI	Block Grid Index
DSI	Dynamic Stripe Index
DKNN	DSI-based K-NN
CNN	Convolutional Neural Network
SEA	Shared Execution Algorithm
RE	Random Estimation

References

1. Afanador, J.J.C.; Rivero, A.J.L.; Gallego, J.Á.R. Analysis of geolocation accuracy by GPS: dedicated support signal integration and collaborative network in location-based services. In Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, 24–27 June 2020; pp. 1–8.
2. Basiri, A.; Moore, T.; Hill, C.; Bhatia, P. Challenges of location-based services market analysis: current market description. In *Progress in Location-Based Services 2014*; Springer: Berlin, Heidelberg, Germany, 2015; pp. 273–282.
3. Khan, P.W.; Byun, Y.C. Smart contract centric inference engine for intelligent electric vehicle transportation system. *Sensors* **2020**, *20*, 4252. [CrossRef] [PubMed]
4. Arroyo Ohori, K.; Diakité, A.; Krijnen, T.; Ledoux, H.; Stoter, J. Processing BIM and GIS models in practice: experiences and recommendations from a GeoBIM project in the Netherlands. *Isprs Int. J. -Geo-Inf.* **2018**, *7*, 311. [CrossRef]
5. Kim, H.S.; Sun, C.G.; Cho, H.I. Geospatial big data-based geostatistical zonation of seismic site effects in Seoul metropolitan area. *Isprs Int. J.-Geo-Inf.* **2017**, *6*, 174. [CrossRef]
6. Hor, A.; Jadidi, A.; Sohn, G. BIM-GIS integrated geospatial information model using semantic web and RDF graphs. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci* **2016**, *3*, 73–79. [CrossRef]
7. Xu, X.; Xiong, L.; Sunderam, V.; Liu, J.; Luo, J. Speed partitioning for indexing moving objects. In Proceedings of the International Symposium on Spatial and Temporal Databases, Online, 23–25 August 2021; Springer: Berlin, Heidelberg, Germany, 2015, pp. 216–234.
8. Wu, C.; Zhu, Q.; Zhang, Y.; Du, Z.; Ye, X.; Qin, H.; Zhou, Y. A NOSQL–SQL hybrid organization and management approach for real-time geospatial data: A case study of public security video surveillance. *Isprs Int. J.-Geo-Inf.* **2017**, *6*, 21. [CrossRef]
9. de Oliveira, T.H.M.; Painho, M. Open Geospatial Data Contribution Towards Sentiment Analysis Within the Human Dimension of Smart Cities. In *Open Source Geospatial Science for Urban Studies*; Springer: Berlin, Heidelberg, Germany, 2021; pp. 75–95.
10. Dou, S.; Zhang, H.; Zhao, Y.; Wang, A.; Xiong, Y.; Zuo, J. Research on construction of spatio-temporal data visualization platform for gis and bim fusion. *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2020**, *42*, 555–563. [CrossRef]
11. Yuan, Z.; Liu, H.; Liu, Y.; Zhang, D.; Yi, F.; Zhu, N.; Xiong, H. Spatio-temporal dual graph attention network for query-poi matching. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020, pp. 629–638.
12. Zhu, H.; Yang, X.; Wang, B.; Lee, W.C.; Yin, J.; Xu, J. Processing Continuous k Nearest Neighbor Queries in Obstructed Space with Voronoi Diagrams. *Acm Trans. Spat. Algorithms Syst. (TSAS)* **2020**, *7*, 1–27. [CrossRef]
13. Cho, H.J. A Batch Processing Algorithm for Moving k-Nearest Neighbor Queries in Dynamic Spatial Networks. *J. Korea Soc. Comput. Inf.* **2021**, *26*, 63–74.
14. Lixiang, S.; Ke, F. Research on K Nearest Neighbor Skyline Query in Time Dependent Road Network. J. Phys. Conf. Ser. **2021**, *1848*, 012140. [CrossRef]
15. Yang, R.; Niu, B. Continuous k Nearest Neighbor Queries over Large-Scale Spatial–Textual Data Streams. *Isprs Int. J.-Geo-Inf.* **2020**, *9*, 694. [CrossRef]
16. Jiang, W.; Li, G.; An, J.; Sun, Y.; Chen, H.; Li, X. Research on Indexing and KNN Query of Moving Objects in Road Network Environment. In Proceedings of the International Conference in Communications, Signal Processing, and Systems, Changbaishan, China, 4–5 July 2020, pp. 1944–1950.
17. Zhang, F.; Zheng, Y.; Xu, D.; Du, Z.; Wang, Y.; Liu, R.; Ye, X. Real-time spatial queries for moving objects using storm topology. *Isprs Int. J.-Geo-Inf.* **2016**, *5*, 178. [CrossRef]
18. Fan, P.; Li, G.; Yuan, L.; Li, Y. Vague continuous K-nearest neighbor queries over moving objects with uncertain velocity in road networks. *Inf. Syst.* **2012**, *37*, 13–32. [CrossRef]
19. Mahmood, A.R.; Punni, S.; Aref, W.G. Spatio-temporal access methods: a survey (2010-2017). *GeoInformatica* **2019**, *23*, 1–36. [CrossRef]
20. Tao, Y.; Papadias, D.; Shen, Q. Continuous nearest neighbor search. In Proceedings of the VLDB'02: Proceedings of the 28th International Conference on very Large Databases, Hong Kong, China, 20–23 August 2002; pp. 287–298.
21. Yang, M.; Ma, K.; Yu, X. An efficient index structure for distributed k-nearest neighbours query processing. *Soft Comput.* **2020**, *24*, 5539–5550. [CrossRef]
22. Yu, Z.; Liu, Y.; Yu, X.; Pu, K.Q. Scalable distributed processing of K nearest neighbor queries over moving objects. *IEEE Trans. Knowl. Data Eng.* **2014**, *27*, 1383–1396. [CrossRef]
23. Sibolla, B.H.; Coetzee, S.; Van Zyl, T.L. A framework for visual analytics of spatio-temporal sensor observations from data streams. *Isprs Int. J.-Geo-Inf.* **2018**, *7*, 475. [CrossRef]
24. Xiong, X.; Mokbel, M.F.; Aref, W.G. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In Proceedings of the 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan, 5–8 April 2005; pp. 643–654.
25. Hua, M.; Lau, M.K.; Pei, J.; Wu, K. Continuous K-means monitoring with low reporting cost in sensor networks. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1679–1691.

26. Yu, Z.; Jiao, K. Incremental Processing of Continuous K Nearest Neighbor Queries Over Moving Objects. In Proceedings of the 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), Dalian, China, 25–27 December 2017; pp. 1–4.
27. Servigne, S.; Noël, G. Real time and spatiotemporal data indexing for sensor based databases. In *Geospatial Information Technology for Emergency Response*; CRC Press: Boca Raton, FL, USA, 2008; pp. 123–142.
28. Bareche, I.; Xia, Y. Selective Velocity Distributed Indexing for Continuously Moving Objects Model. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Melbourne, VIC, Australia, 9–11 December 2019, pp. 339–348.
29. Jayachandran, P.; Tunga, K.; Kamat, N.; Nandi, A. Combining user interaction, speculative query execution and sampling in the DICE system. *Proc. VLDB Endow.* **2014**, *7*, 1697–1700. [CrossRef]
30. Suel, T.; Mathur, C.; Wu, J.W.; Zhang, J.; Delis, A.; Kharrazi, M.; Long, X.; Shanmugasundaram, K. *ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval*; WebDB: International Workshop on the Web and Databases, Dallas, TX, USA 2003, Volume 3, pp. 67–72.
31. Xu, X.; Xiong, L.; Sunderam, V.; Liu, J.; Luo, J. VPIndexer Dataset, 2022. Available online: http://www.mathcs.emory.edu/~lxiong/aims/spindex/VPIndexer/data/sz/ (accessed on 20 November 2020).
32. Sinno, Z.; Bovik, A.C. Spatio-temporal measures of naturalness. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 1750–1754.
33. Mouratidis, K.; Yiu, M.L.; Papadias, D.; Mamoulis, N. Continuous nearest neighbor monitoring in road networks. In Proceedings of the VLDB 2006: Proceedings of the 32nd International Conference on very Large Data Bases, Seoul, Korea, 12–15 September 2006; pp. 43–54.
34. Huang, Y.K.; Chen, Z.W.; Lee, C. Continuous k-nearest neighbor query over moving objects in road networks. In *Advances in Data and Web Management*; Springer: Berlin, Heidelberg, Germany, 2009; pp. 27–38.