IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2019, accepted July 15, 2019, date of publication July 17, 2019, date of current version August 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929531

Geo-Gap Tree: A Progressive Query and Visualization Method for Massive Spatial Data

WEI XIONG, RUIQING LI, JIN PENG, YE WU, NING GUO, AND NING JING
College of Electronic Science, National University of Defense Technology, Changsha 410073, China

Corresponding author: Wei Xiong (xiongwei@nudt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 41871284.

ABSTRACT Online visualization and query of massive geo-spatial data are facing increasing challenges with the explosive growth of location-based spatial datasets. In the practical scenario, online visualization is carried out in a progressive way, namely, a sketchy view map is first presented, and more detailed view maps are produced gradually as the viewport scale goes deeper. One approach is to use the multi-scale spatial index technique. However, it loses the original data attribute and cannot provide spatial statistics information. The paper is to provide an improved index structure, the Geo-Gap tree, which aims to enhance online interactive access to large spatial datasets, as well as enable one to compute statistical attributes like aggregation at the coarse level. Therefore, the first focus of Geo-Gap tree is improving the efficiency of tree building. For this purpose, an adaptive geohash coding is introduced to reduce the computing of neighboring objects. And, this phase can be improved in parallel once objects are partitioned. Compare to Gap tree, the cost of building the Geo-Gap tree can be greatly reduced. The second contribution is to choose data at different level based on sampling so that a sample for each level can be served as a progressive query result. The third contribution is an estimation of progressive query results, which ensure that progressive query accuracy can be controlled within the range of theoretical analysis. With the query continuing to execute, the query results become more and more accurate. The method is now integrated successfully into a high-performance geographic information system called HiGIS.

INDEX TERMS Progressive query, visualization, spatial index, sampling, estimation.

I. INTRODUCTION

The fast development of data acquisition tools led to the explosion in the data volume, thus making researchers and engineers heavily dependent on statistical analysis and visualization tools to investigate inner patterns of massive spatial information [1]. Taking the typical current online application in China as an example, maximum location requests to Baidu Map in one day have been more than 23 billion times. Didi Chuxing, a famous mobile transportation platform in China, offers more than 1000 car requirements per second, 9 billion routing requests in a day. In this context, thousands of retrieval per second, such as traffic situation and available vehicles, place higher demands on the concurrent load and response time of spatial query processing [2]–[5]. Firstly, the user experience of online query and analysis requires a faster response. Secondly, similar to the progressive transfer technology on the Internet, users may not need accurate query results, but quickly obtain a rough view. Moreover, in online spatial analysis, users may modify query conditions interactively to find more satisfactory answers. Traditional processing and visualization solutions use accurate calculating methods, which bring about a large result dataset and a long period of processing time. In many applications, users are inclined to observe the overview of the dataset rather than knowing many detailed data items [6].

Suppose that we are interested in finding a region with a particular feature (e.g., land use of cane). For this purpose, we probably would not need to load all of the spatial data at the level of meters. A scale of precision of 10 or 100 meters may be enough for the relevant analytical requirements, without the need to process the data at a higher level of detail [6]–[9].

To cope with the interactive mapping issues, the multi-scale spatial index was proposed as a fundamental and effective approach for dealing with real-time visualization.

The associate editor coordinating the review of this manuscript and approving it for publication was Waleed Alsabhan.

Its purpose is to enhance access to and visualization of massive vector data. Leilani Battle et al. proposed the ScalaR system [10] to deal with the issues of scaling the visualization system seamlessly from small datasets to enormous ones. ScalaR dynamically reduces the system resolution when the expected query result is too large to be effectively rendered on screen; bin-summarize-smooth [11] has to deal with both the challenges of what to display for very large datasets and how to display it fast enough for interactive exploration. Meanwhile, the mapbox [12] provides client-side rendering and has to make huge maps fast. Rather than composing a series of image files on a server, the program's ability to change styles on the fly expands the possibilities for how a map can be displayed.

However, applications in the big data era require not only the capability of immediate interactive visualization, but also the capability of online analysis of patterns contained in the dataset. For example, when users investigate a series of cane land parcels, they may not only want to browse the whole area at different scales, but also want to know statistical information such as the total area, the maximum and the average yield of parcels in viewports.

Spatial aggregation techniques can provide support for computing statistical information [13], especially useful in approximate queries in the big data scenario. There are two fundamental issues: one is the estimation of the confidence of the returned approximate result; the other is the method for data sampling. XDB [14] is an online engine integrating a non-spatial data sampling method called wander join, and this idea is extended to support spatial data query and analysis functions in SIMBA [15], but only for the point data type. Hu *et al.* [16] studied the independent range sampling problem and showed how to produce samples for various range queries. Namely, the samples returned by each query should be independent of the samples returned by the previous queries. Tong *et al.* [5] addressed the online minimum bipartite matching problem in real time spatial data, and provided some algorithms to give an accurate instead of sampling result for a special type point query. Christensen *et al.* [17] presented the STORM system to enable spatio-temporal online reasoning and management of large spatio-temporal data. Agarwal presented BlinkDB [18], which allowed users to make a trade-off between query accuracy and response time, thus enabling interactive queries over massive datasets by running queries on data samples and presenting results annotated with meaningful error bars. Zeng *et al*. [19] developed a probabilistic model for the statistical bootstrap process and showed how it can be used for automatically deriving error estimates for complex database queries. Qin *et al.* [20] introduced a general framework for parallel online aggregation, and a generic interface was defined to express any estimation model that completely abstracts the execution details. However, it has not provided clear evidence of confidence bounds. Wang *et al.* [21] investigated spatial online sampling and aggregation problems and presented two indexing structures, namely the LS-tree and RS-tree. These structures were used to produce spatial online samples and to execute various spatial online aggregation and analytics. Yet, it gave no error calculations. Yan *et al.* [22] applied the stratified sampling method to aggregate queries on big sparse data, but it relied on the knowledge of data distribution.

From the above, we may notice that past works on the multi-scale index focused only on techniques to improve the capability of quick visualization, while ignoring providing support for data analysis; and works on the spatial aggregation query mentioned the progressive mapping for massive spatial data less. Therefore, in certain applications in big spatial data, users may face a dilemma: either the query on big spatial data shows unnecessary details with long execution time, or the general mapping tiles can be obtained quickly, but no further information returned.

In order to provide both the multi-scale visualization ability for online mapping and real-time aggregation ability for online analysis, this paper proposes a novel index structure: the Geo-Gap tree, which is based on the gap tree and enhanced by the idea of geohash coding. One benefit is that the index building performance was improved over the traditional gap tree, the other is that the aggregation of the spatial dataset can be estimated immediately while mapping goes on. The progressive query idea is incorporated, and theoretical analysis of the error range for query results is given, which ensure the estimation will be better approaching the exact value with the queries going deeper.

The remainder of the paper is organized as follows. The concept of the gap tree and estimation is introduced in Section II. The structure of Geo-Gap tree and detailed methods are described in Section III. Extensive experimental results are shown in Section IV, and conclusions are given in Section V.

II. PRELIMINARIES

A. GAP TREE

The Gap tree was proposed by van Osterom to avoid the problem of empty islands in Reactive-tree [23], and is a multi-scale index structure for rapid visualization of large-scale spatial data. During the building process of gap tree, first of all, each spatial object n is assigned an id and calculated an importance value according to Equation 1.

In the next step, the least important object is removed, and it is merged with its neighboring object q, which has the maximum weight value determined according to Equation 2. Also, a new id is assigned to the new object, which is enlarged and becomes more important. The importance value of the enlarged object will be recomputed according to Equation 1. $L(n, q)$ is the length of intersecting boundary or the area of the intersection between n and its neighbors q, and $WeightFactor(n, q)$ is the weight factor of the intersection between n and q. We use $f(n)$ to denote the value of n and $f(n, q)$ to denote the maximum value of n and q. A list of symbols used in this paper is shown in Table 1. Therefore, the hierarchy can be constructed in the gap tree [9], [24]. To answer an ordinary range query Q with a gap tree, it started

TABLE 1. Notation used in the paper.

Notation	Meaning
P_i	The objects at the i level.
n	The objects in the raw dataset.
q	The neighbors of object n.
u	The objects in the query range.
i	The total level numbers.
Q	A range query.
$f(n)$	This is a function which calculate the value of object n.
$f(n,q)$	This is a function which calculate the value of n and q.
$f(s,q)$	This is a function which calculate the value of the stroke and its neighbors.
$L(n,q)$	The common boundary or common area between n and q.
$L(s,q)$	The total length of the stroke and its neighbors.
$A(s,q)$	The angle between the stroke and its neighbors.
$WeightFactor(n,q)$	The weight factor of the common part type between n and q.
$WeightFactor(s,q)$	The weight factor of the topological relation.
$limit(i)$	The setting of level i
$object\_Box()$	Classifier; different levels have different values.
$distance\_Box()$	Filter; different levels have different values.
$distance()$,$score()$,...	Methods of calculating values.

from the user-specified level i and returned the objects u if and only if $P_i \cap Q \neq \emptyset$.

$$f(n) = f(type, attributes, size) \tag{1}$$

$$f(n,q) = f(L(n,q), WeightFactor(n,q)) \tag{2}$$

The complexity of building the gap tree consists of two parts. First, suppose that only layer i is constructed, then the number of objects to be involved is n. The complexity of this part is $O(n)$, because it needs to compute the $f(n)$ for each object. This part of the overhead is 'mandatory'. Second, determine whether objects n belong to that level i by calculating its value $f(n)$. If an object n belongs to level i, then the object n should search all neighbors, as well as compute this value between object n and their neighbors q. Then, the objects n and q with maximum value $f(n,q)$ will be amalgamated. If object n does not belong to level i, it also should search all neighbors and compute the maximum value. Then, we need to judge whether $f(n,q)$ belongs to the level i; if it does, amalgamate; otherwise, skip this object. All objects must find their neighbors and compute the maximum value; either $f(n)$ belongs to or does not belong to level i. Therefore, the complexity of this part is $O(n^2)$. The focus of this paper, and the purpose of which, is precisely to avoid involving a plethora of neighbors, i.e., the $O(n^2)$ term. The $O(n)$ term, on the other hand, is the cost of the 'filter candidate' it takes to build in the index. If this part of the time is to be reduced, it can be segmented and built in parallel. However, the total overhead is probably unavoidable. Thus, the first goal is to reduce the number of neighbors and reduce its computational complexity.

B. LEVEL

The multi-scale tree consists of datasets with different levels of detail of the map. Thus, the dataset is consistent when the corresponding objects at the different levels are connected, thereby building a hierarchical structure. The term 'level' refers here to a numerical value, or scale, that can be mainly used to check whether objects belong to it. Lower levels in the tree correspond to finer resolution representations of the map [25], [26].

In the gap tree, it assigns an importance value to each object, and each object will be stored at a certain level according to its importance value $f(n)$ or $f(n,q)$.In previous works, there have been two ways, commonly, to set levels: a pixel method and an empirical method. The former iterates the area or perimeter to calculate the threshold of this level. The latter, however, does not need to be calculated with the objects [27]. Strictly speaking, the threshold set for each level relies on human experience, which is too subjective. Thus, the pixel method is usually considered and accepted.

C. SAMPLING

Sampling is a basic and effective approach for computing aggregates, such as the sum or average value computing. If P is a set of n objects, Q is a given range query, then the sampling is the returned objects from $Q \bigcap P$, which is not an exact result, but an approximated value with an error bound. To indicate how accurate the result is, a confidence and an error bound ϵ are provided with each result. An error bound is an interval in which the actual value falls within a high possibility. Suppose that the precise result is v and the approximate one is u; we have $P(|v-u| \leqslant \epsilon) \geqslant c$. Namely, the probability of the difference between the actual result and the estimated result is less than c [20]–[22].

D. ESTIMATION

When using sampling for progressive query, the results must be assured in an acceptable range. Here, estimation principles for aggregation queries will be introduced. Let $G = \{g_1, g_2, \cdots, g_n\}$ be the set of a non-empty sample, and let $X = \{x_g^1, x_g^2, \cdots, x_g^n\}$ be g's probability in the range, which has independent random variables bounded by the interval [0, 1]. The mean of these variables is defined:

$$\hat{X} = \frac{1}{n}(x_1 + x_2 + \cdots + x_n) \tag{3}$$

In Hoeffding's inequality, it provides an upper bound on the probability that the sum of independent random variables deviates from its expected value.

$$\mathbb{P}(|\hat{X} - E(\hat{X})| > \varepsilon) \leq e^{-2n\varepsilon^2} \tag{4}$$

Here, $E(\hat{X})$ is the expected value of $\hat{X}$. Meanwhile, $E(\hat{X}) = E(\overline{X})$, where $E(\overline{X})$ is the population mean. Therefore, given a confidence δ and the sample size n, Hoeffding's inequality derived from the statistical sampling defines the actual value.

III. MATERIALS AND METHODS

A. DETAILED DESIGN OF THE GEO-GAP TREE

The gap tree is a hierarchical structure that supports access to spatial datasets and realizes progressive visualization, but

TABLE 2. Functions for build Geo-Gap tree.

Function	Input	Output
$score()$	object n	weight of n
$distance()$	object n, object q	weight between n and q
$union()$	object n, object q	new object merged n and q

it suffers from some drawbacks. One of the most important reasons is, excessive neighbors require excessive computing time, and this leads to additional system overhead. In this section, the index structure, Geo-Gap tree, will be introduced, which addresses overcoming this drawback in two ways. Geo-Gap tree use an adaptive geohash coding for optimizing the neighboring search, and parallel strategy for improving building efficiency.

1) BUILDING ALGORITHM

Geohash is a latitude/longitude system that subdivides space into grids that are globally unique, hierarchical, recursive and one-dimensional. The design of the Geo-Gap tree is based on the adaptive geohash method we proposed in [28], which could represent both the location and the approximate size of the coded object directly by the meshing hierarchy and the corresponding coding length. Through this geohash coding, the greater the geographical proximity, the more they will have the same prefix. Following the idea above, the building index algorithm is quite straightforwardly, and it essentially reduces the amount of adjudged neighboring objects.

To build Geo-Gap tree, some functions are described as Table 2, which are used for computing the weight of spatial objects and the weight between them and their neighbors.

Firstly, for all of the objects that are at a level i, the $f(n)$ value of object n is computed according to a *score*() method. And all the $f(n)$ are putting in the classifier $object\_Box(i)$ corresponding to the level i. The classifier *object_Box*() is described in Equation 5. Secondly, a limited number of neighbors q of object n are found by calculating the geohash value of object n. The detailed process is described in Figure 1.

$$object\_Box(i) = \begin{cases} merged & f(n) \geqslant limit(i) \\ distance\_Box() & f(n) < limit(i) \end{cases} \quad (5)$$

$$distance\_Box(i) = \begin{cases} merged & l(n,q) \geqslant limit(i) \\ skip & f(s,q) < limit(i) \end{cases} \quad (6)$$

Then, the maximum $f(n,q)$ is computed by *distance*(). After the objects have passed through an *object_Box*() classifier, some objects n and q are amalgamated, as well as

Algorithm 1 Building Geo-Gap Tree

Input: i:The total level; n: object; $limit(i)$: the setting of level i.

Output: tree T

```
n.level = i;
while i > 0 do
    while n.level ≥ i do
        n.level ← i;
        f(n) ← n.score();
        object_Box(i) ← {f(n)};
        {q} ←calculate object n's geohash value and find limited number of neighbors
        f(n, q) ← max(n.distance(q))
        q ←when n.distance(q) reaches the maximum
        if f(n) ≥ limit(i) then
            u ←amalgamate n and q;
            u.level ← i − 1
            n.level ← i
            q.level ← i
        else
            distance_Box() ← {f(n, q)};
            u ←amalgamate n and q;
            u.level ← i − 1
            n.level ← i
            q.level ← i
        end if
        i ← i − 1
    end while
end while
return tree T
```

updated; their level becomes i, and the newly-generated object's level is marked $i-1$. Other objects can pass through a *distance_Box*() filter defined as Equation 6. The objects n and q must be amalgamated and the updated level is i for each object through the distance_Box() filter. Additionally, the newly-generated object's level is $i-1$ as well. When the object of level i is exhausted, it will be removed, and the program will choose all of the objects that are level $> i$. This process is repeated until $i \leq 0$. The pseudocode is described in Algorithm 1. Each of n objects should be computed once to get a score, the complexity of this phase is $O(n)$. The complexity of finding limited q neighbors is $O(nlog(n)) + qO(n)$ if merge sort strategy is applied. This phase can be improved in parallel once objects are partitioned. Compare to

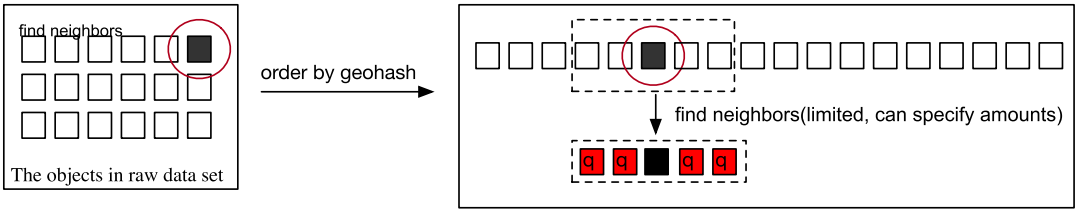

FIGURE 1. The process of choosing neighbors.

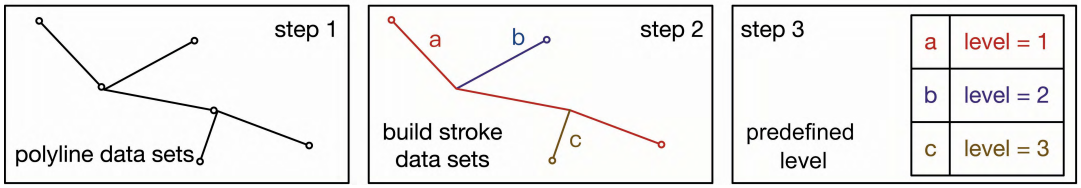

FIGURE 2. Data preprocessing for polyline datasets.

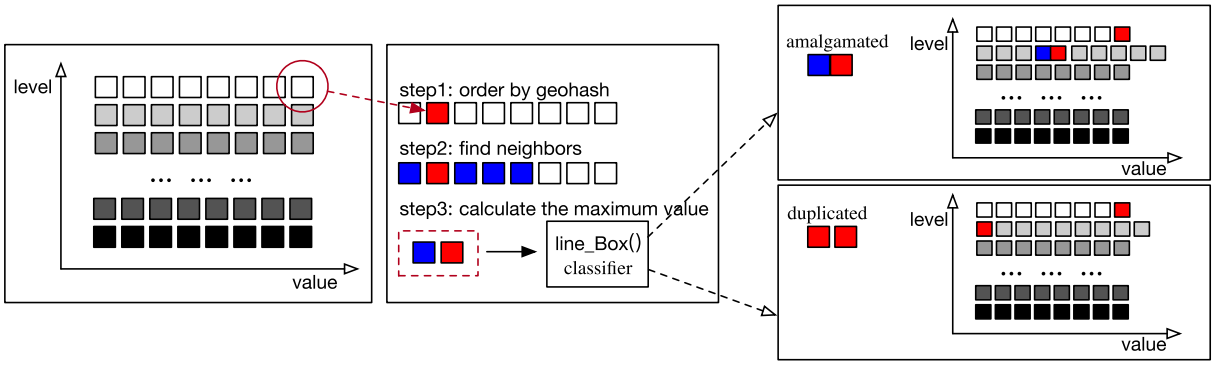

FIGURE 3. The process of building the Geo-Gap tree for the polyline datasets.

Gap tree, the cost of building the Geo-Gap tree can be greatly reduced.

2) EXTENDED GEO-GAP TREE STRUCTURE

The Geo-Gap tree described above suits only polygon datasets. In general, ideas behind Geo-Gap tree can be naturally extended to polyline datasets. In a spatial database, a long polyline chain may be represented by many segments (Step 1). Sometimes, it is highly desirable to rebuild individual polyline datasets. A feasible solution is to concatenate polyline segments into long polyline chains based on some criteria. The concatenated polyline chain is called the stroke [29]–[32] (Step 2). Firstly, the deflection angle, which is the deviation from 180 ° of the angle between two polyline chains, was used as a criterion for judging which two polyline segments should be concatenated. Then, by means of Equation 1, the value of each stroke is obtained. Additionally, each stroke can be divided by the value (Step 3). Until now, each object has its own level. The details of this process are described in Figure 2.

After that, a limited number of neighbors q are found by the computation of the geohash value of each stroke at the same level. Then, the maximum value is computed between each stroke and its neighbors by Equation 7; where $L(s, q)$ is the total length of the stroke and its neighbors, $A(s, q)$ is the angle between the stroke and its neighbors and $WeightFactor(s, q)$ is the weight factor of the topological relation.

$$f(s,q) = f(L(s,q), A(s,q), WeightFactor(s,q)) \tag{7}$$

$$line\_Box(i) = \begin{cases} merged & f(s,q) \geqslant limit(i) \\ duplicated & f(s,q) < limit(i) \end{cases} \tag{8}$$

Building the Geo-Gap tree for the polyline datasets started from the top level, and the maximum value passed a line_Box() classifier. The classifier *line_Box*() is described in Equation 8. Some strokes amalgamate with its neighbor q and update to the newly-generated objects' level. Other strokes should be duplicated and updated to their level. This process is repeated down to the bottom level. The details of this process are described in Figure 3.

B. ESTIMATION

A good estimation of progressive query results can ensure that the query accuracy can be controlled within the range of theoretical analysis. Our solution to estimate the hierarchy data is based on sampling, so that we may choose a sample for each level. Generally speaking, any aggregate of the whole population can be estimated from a sample, and the accuracy improves over more samples [21]. The detail of the sampling is outlined in the following algorithm.

1) STRUCTURE

To calculate the relationship between the original data and sample data for each level, as their relationship is already in the index, the table (the Geo-Gap table) is easily established. The detailed Geo-Gap table structure is in Figure 4, in which the $t()$ is a mapping relationship that represents the number of nodes that corresponds to an object in some layer. The generation method of the Geo-Gap table is described in Algorithm 2. This algorithm traverses the Geo-Gap tree to generate the table, the complexity is $O(n)$.

Initially, the notation $\{v\}$ is the objects of the Geo-Gap tree at the bottom level. The *Child_Box*() method looks up

object	t( )	level
a	b	0
c	b	0
…	…	…
b	b	i
d	d	i

FIGURE 4. The detailed Geo-Gap table structure.

Algorithm 2 Building the Geo-Gap-Table

Input: i: The total level numbers; T: Geo-Gap tree;
Output: Geo-Gap table

$n \leftarrow i$;
while $n \geq 0$ **do**
  $\{v\} \leftarrow$*object of T in level n*;
  for each $v \in \{v\}$ **do**
    Table_Box(v)
    Child_Box(v);
    if *Table_Box*$(v) = \emptyset$ **then**
      if *Child_Box*$(v) = \emptyset$ **then**
        $t(v) \leftarrow w$
      else
        $\{w\} \leftarrow$ *Child_Box*(v)
        if *Table_Box*$(w) = \emptyset$ **then**
          $t(v) \leftarrow w$
        else
          $t(v) \leftarrow$ *Table_Box*(w)
        end if
      end if
    end if
  end for
  $n \leftarrow n - 1$
end while
return Geo-Gap-table

TABLE 3. Functions for build Geo-Gap table.

Function	Input	Output
$Child\_Box()$	object n	children nodes of n in the tree
$Table\_Box()$	object n	corresponding nodes of n in the table

the subtree rooted at each v, and the *Table_Box*() method looks up the v recorded on the Geo-Gap table. The definitions are shown in Table 3. If *Child_Box*(v) and *Table_Box*(v) are empty, the mapping relationship $t(v) = v$ will be established. If *Child_Box*(v) is not null, traversing find its children $\{w\}$ by using the *Child_Box*() method and record the mapping relationship $t(v) =$ *Table_Box*(w).

2) SAMPLING

Sampling is a significant factor affecting the effect of progressive query. The layered sampling based method corresponding to different spatial objects is proposed. For example, Area is the main impact factor to the range query. The larger the area, the greater impact it contributed to the range query. In this paper, the sampling, based on the area-proportional method, defines a threshold F. The sampling process is the process of choosing children of the object according to the area proportion. If the cumulative value of the area ratio is less than the threshold, it is marked as a sample. When the cumulative value reaches the threshold, the next object is processed. For polyline objects, we choose the children with the same probability according to the stroke based method, and there is no need to compute the area proportion. The sample selection is composed of two main phases, namely sort and filter.

(1) Phase 1, sort: This phase calculates the original data area-proportion in the level i, per the equation $n = area(i)/area(N)$, where $area(N)$ is the total area where the object i lies, and $area(i)$ is the object i's area. Overall, this equation adjusts the original data sequence.
(2) Phase 2, filter: The purpose of this filter function is to choose samples. The threshold F defines the sample rate of the filter function. For example, $F = 50\%$ means that children whose area cumulative value reached 50% of their parent object will be selected.

The detailed process is described in Figure 5.

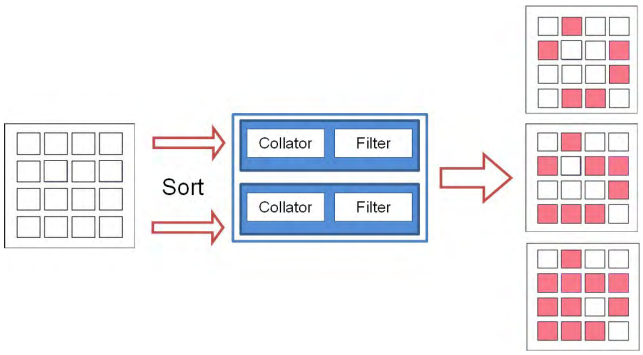

FIGURE 5. The process of choosing the sample.

3) QUERY ALGORITHM

The design of the query algorithm is based on the following three ideas:

(1) Remarks: The sample comes directly from the original data, which means that samples are presented in the very beginning, even at the root of the Geo-Gap tree. However, since the root is derived from the entire Geo-Gap tree, many of them may not be inside the query Q. Thus, samples are returned that are actually inside Q, while the rest will be rejected. This idea, known as rejection sampling, is a common technique to draw values from some arbitrary distribution.
(2) Estimation: Error-bounded uniform sampling based on Equation 1 is straightforward. Given a user-specified confidence δ and sample size n, the estimation's error

bound ε in Equation 9:

$$\varepsilon = \sqrt{-\frac{1}{2n}\ln\left[\frac{(1-\delta)}{2}\right]} \tag{9}$$

By this equation, the range of estimation $[a_g, b_g]$ may be provided in Equation 10:

$$[a_g, b_g] = \left[N(i)(\frac{e(u)}{N(s)} - \varepsilon), N(i)(\frac{e(u)}{N(s)} + \varepsilon)\right] \tag{10}$$

where $e(u)$ is the sample in range Q, $N(s)$ are samples and $N(i)$ is a population. Generally speaking, an aggregate of the whole population can be estimated from a sample, and the accuracy improves as more samples are obtained and the sample size increases.

(3) Sampling: The algorithm essentially mimics a breadth-first search. Consider the query as simply a binomial distribution. Upon request, pick a sample randomly from P and test if it is within Q. Return the sample if so; otherwise, dispose of it and repeat. As described in the above ideas, the query algorithm follows quite straightforwardly. More precisely, a list *Flag* of nodes is maintained from which we can take samples from a range Q. Initially, only the root of the Geo-Gap tree is in *Flag*. If the node u is not inside the range Q, its children will be rejected. If *Sample*(u) is not inside Q, it will be marked 'disable'. When *Sample*(u) is exhausted, u is removed from *Flag*, and the next level is added to *Flag*. The process repeats until user termination or the sampling rate is reached. The detailed process is described in Algorithm 3. In the worst case, a query need to search C times in the Geo-Gap Tree, C is equal to the height of the tree. Therefore, the complexity of this algorithm is $O(1)$.

Algorithm 3 Querying a Geo-Gap Tree

Input: Geo-Gap table; T: Geo-Gap tree; Q: range query; SR: sampling rate;
Output: samples from $P \cap Q$

```
Flag ←node of T in range Q at level u;
while sample < SR do
    if Flag = ∅ then
        return P ∩ Q = ∅
    end if
    if R(u) ∩ Q = ∅ then
        Child(u) ← R(u)'s sample in Geo-Gap table ;
        Remark Child(u) as disable
    else
        e ←a sample from R'(u)
        where R'(u) contains only non disabled elements in R(u)
        if e ∈ Q then
            Report e as a sample
        else if e ∉ Q then
            Remark e as 'disable'
        else
            Remove u from Flage
            Add u's children to Flag
        end if
    end if
end while
```

IV. RESULTS

In this section, several experiments are described. Table 4 lists the specifications of machines and the environments that are used to carry out the experiments.

To compare the index construction time, the performance of visualization and estimation, we used datasets with varied sizes and different types. All of these datasets are real datasets obtained from the land survey applications based on HiGIS [33]. The polygon datasets are land use data from Shaoyang city of Hunan, China, which includes the parcels information about urban planning. Polyline datasets are road network from Beijing city, China, which includes the roads within the 5th ring. After each dataset was processed and filtered, they were stored in a 'csv' file. The size of the raw data files is presented in Table 5.

TABLE 4. Experiment environment.

Items	Configuration
CPU	Intel(R) Core(TM) i5-4430 @3.00 GHz
Memory	8 GB
OS	Ubuntu 14.04
Database	PostgreSQL 9.6, PostGIS 2.2.3
Language	Python 2.7

TABLE 5. Experimental datasets to compare the construction time, visualization and estimation.

Dataset	Size	Type
DATASET1	27M	polygon
DATASET2	65M	polygon
DATASET3	359M	polygon
GEO4	50,000 records	polygon
GEO5	40,927 records	polyline

A. INDEX CONSTRUCTION COST: VARY THE DATA SIZE

The different methods were compared, using both the Geo-Gap tree and gap tree. First, the effects of the data volume on the construction index were examined in a way that compared their total construction times in three different data sizes of polygon (DATASET1 to DATASET3) and in different numbers of polygon and polyline features(using different sub set of GEO4 and GEO5), and set the construction level = 10. As shown in Figure 6 and Figure 7, as the input volume increased, both approaches were also increasing. For DATASET3, gap tree cannot be built in acceptable time where an 'X' is used to represent a long wait time. The reason for this was that, there was a relatively bigger initialization cost in the early stages of the build index. Meanwhile, the experimental results showed that the method based on geohash used in Geo-Gap tree was efficient for building an index tree, especially since the larger the data were, the more the improvement became obvious. However, the gap tree does not

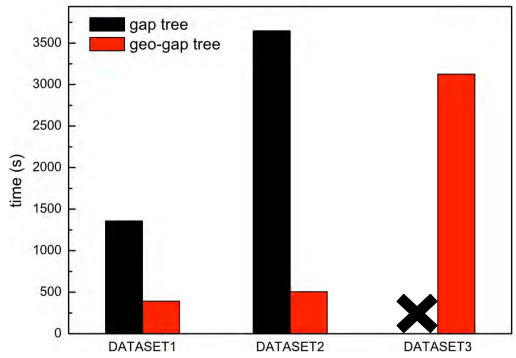

FIGURE 6. Comparison of construction time for polygon data varying data sizes.

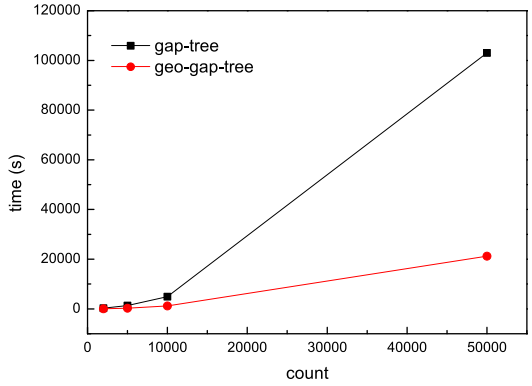

FIGURE 7. Comparison of construction time for polyline data varying feature numbers.

utilize the index, so it needs to traverse all objects, resulting in poor tree building performance.

B. QUALITY OF VISUALIZATION

The visual quality at different levels on the GEO4 datasets was also compared. The comparison of visualization results are shown in Figure 8, respectively. From the point of view of visualization quality, there is no significant difference between two methods. Meanwhile, the records of the level are shown in Table 6. Although the original gap tree had fewer objects on the coarse level, it has not been well used in a range query yet. When the gap tree was at the level $= 0$, it only had one object, so it always returned the same result for each range query. Therefore, Gap tree is good for visualization but not progressive query. The visual quality is also maintained well in Geo-Gap tree because of the appropriate sampling method. Especially for Polyline data, the main roads in each layer can be preserved.

Then, the visual quality of different methods on the GEO5 datasets was also compared. The visualization results are shown in Figure 9. As for the visualization, it was pretty obvious that the Geo-Gap tree was completely applicable for use in polyline datasets.

C. QUALITY OF ESTIMATION

The size q of a query result may be estimated as a result of sampling. Meanwhile, the accuracy of the estimate query is

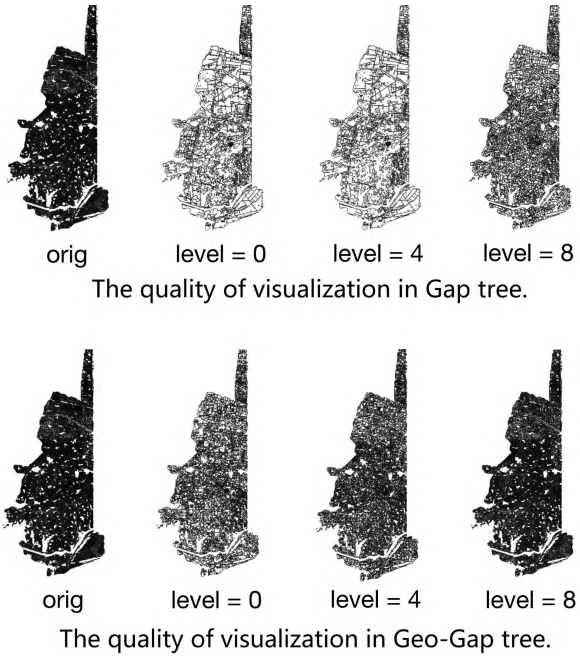

FIGURE 8. The quality of visualization.

TABLE 6. Records of the level.

Size	Gap tree	Geo-Gap tree
Original	50,000	50,000
Level 0	3,814	24,143
Level 4	48	2,045
Level 8	1	338

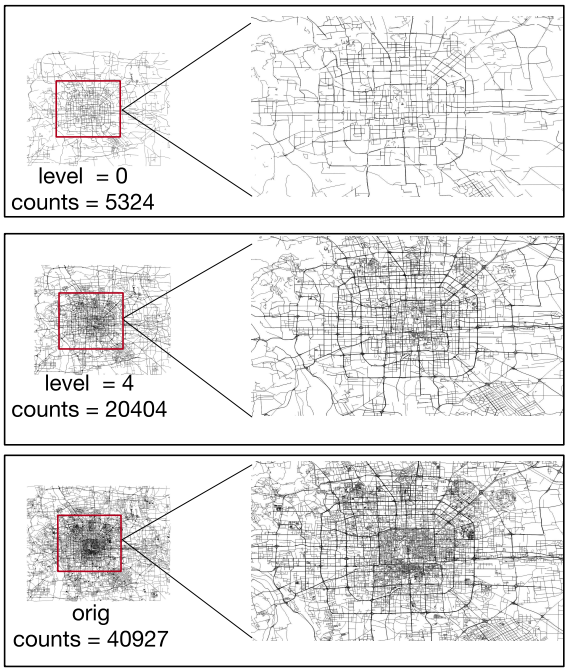

FIGURE 9. The quality of visualization in Geo-Gap tree: Polyline datasets.

of foremost importance. However, there are plenty of factors that impact estimation quality. In this section, this is considered and analyzed with respect to possible influences, such as

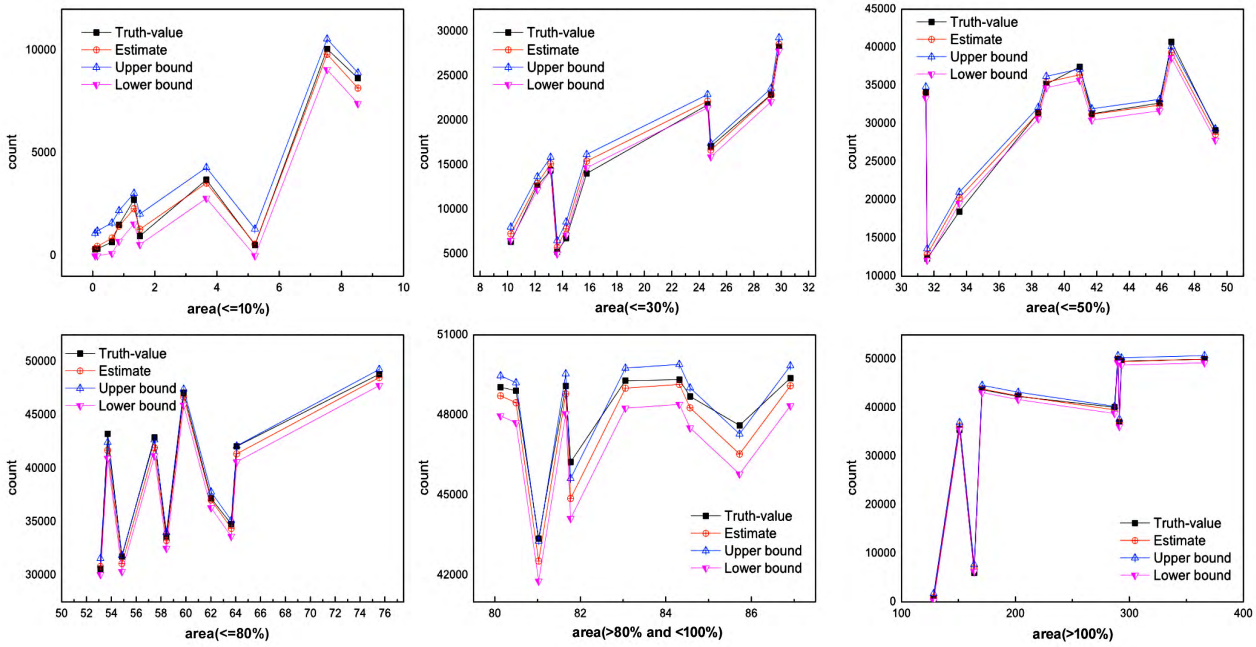

FIGURE 10. The estimation quality of varying query area: GEO4.

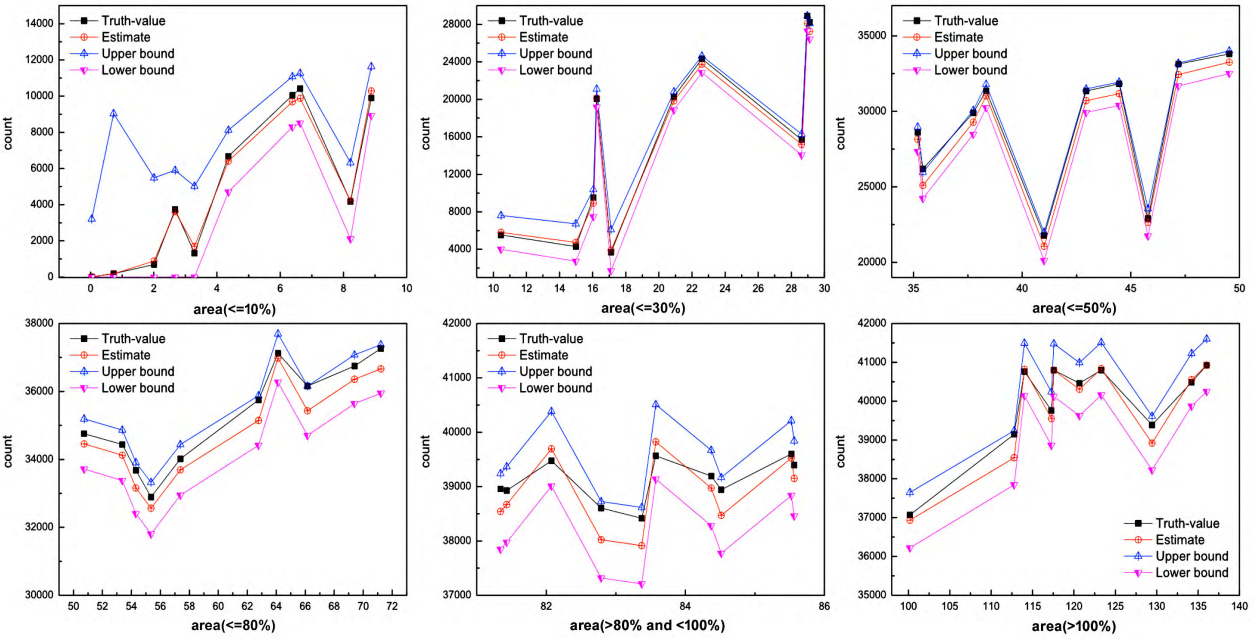

FIGURE 11. The estimation quality of varying query area: GEO5.

query area, the original counts, etc. The lower/upper bounds are minimum/maximum values of query results, which are estimated according to the Equation 10.

1) VARYING THE QUERY AREA

The estimation quality of the different query areas was tested in terms of which the datasets had 50,000 records (GEO4) and 40,927 records (GEO5); fixed as $F = 0.5$ in GEO4, recalling that F equals the sum of the area ratio of children. This effectively meant that F was the sampling rate. Figure 10 and Figure 11 show the accuracy of the estimate for different areas on the GEO4 dataset and GEO5 dataset respectively, where the confidence was 98%. The x-axis is the query area ratio, and it had categories that ranged from 0 to 10%, from 10% to 30%, from 50% to 70%, from 70% to 80% and for those with more than 80%. In reality, however, the query box

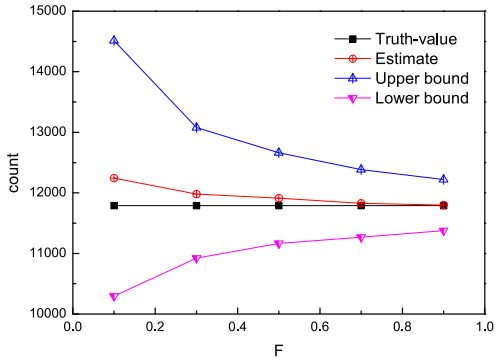

FIGURE 12. The result accuracy: Vary *F*, GEO4.

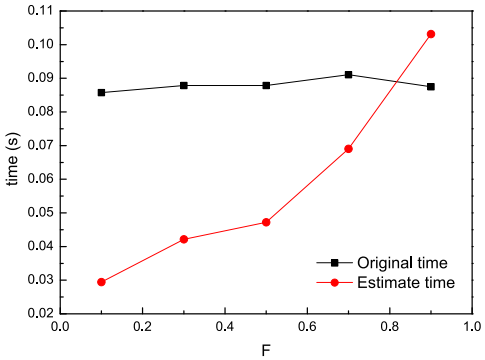

FIGURE 13. The query time: Vary *F*, GEO4.

may hover over the map area, but it does not cover everything. Therefore, we also had to design an experiment whereby the query area was more than 100% and imperfect coverage for the original data. Looking closely at Figure 10 and Figure 11, the estimate values retained their high quality estimation in different areas, which meant the difference between the query areas did not have a significant impact on the estimation quality.

2) VARY *F*

Figure 12 shows the accuracy of our methods when varying F and fixing the query area. The input dataset is GEO4. The final result of each range query was an average value that was calculated by taking ten times the random range queries under the same conditions. Meanwhile, the true value, the estimated value, the upper bound, the lower bound and time were recorded.

It is obvious that the higher the F, the more accurate the result was. However, in Figure 13, it is also found that the larger the F number of queries, the more time was needed for the queries. That's because more samples mean more computing costs. Therefore, there was a balance between query response performance and the sampling rate.

3) VARYING LEVEL

Actually, this index can support a progressive query, which means that with the increase of time, the estimated results will tend to be closer to the exact value. In the end, it should reach the exact value. The experiment was designed according to this characteristic that the query will be more accurate with the increase of time. First, the Geo-Gap tree for two different datasets was built (GEO4 and GEO5). Second, for each dataset, ten random range queries were performed, and the average value was used instead of the result of the range query for each dataset. The estimation accuracy is recorded in Figure 14. The x-axis in Figure 14 represents the different levels of Geo-Gap tree. It shows how the estimation accuracy increased gradually with the increase over time and that the query algorithm worked well for different datasets.

This is a clear indication that the Geo-Gap tree is not only applicable to progressive visualization, but also able to produce estimation whose approximation quality improves over time. The error caused by sampling is theoretically analyzed, the experiments show that the error is within the acceptable range.

D. EFFICIENCY

The query execution time in this experiment was also measured. Gap tree is only good at visualization, it cannot support aggregation query for spatial data. Therefore, we compare with PostGIS instead of Gap tree. Fixed confidence was 98%; F was 0.5; and the query level was zero. Figure 15 shows the performance of the Geo-Gap tree and PostGIS. The result obtained was the average time of ten range queries.

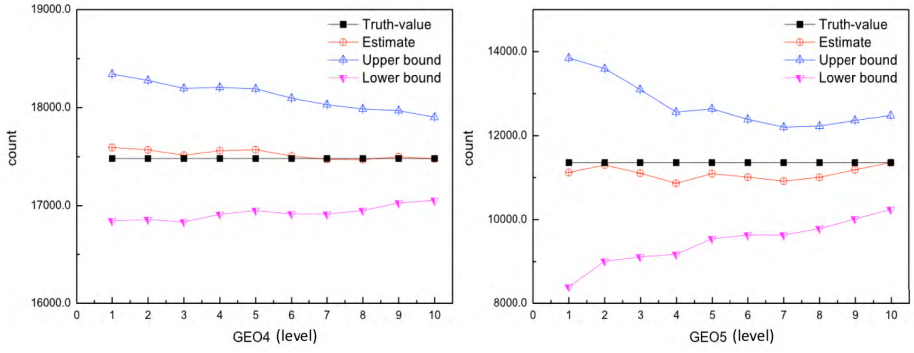

FIGURE 14. The estimated results in varying the level.

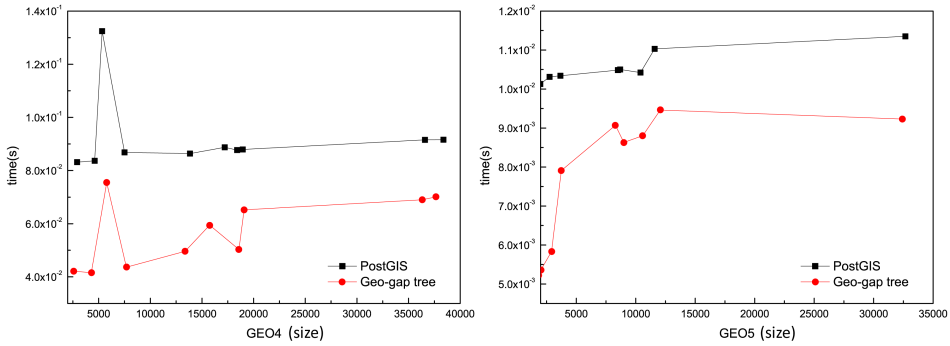

FIGURE 15. Different query time between PostGIS and Geo-Gap tree.

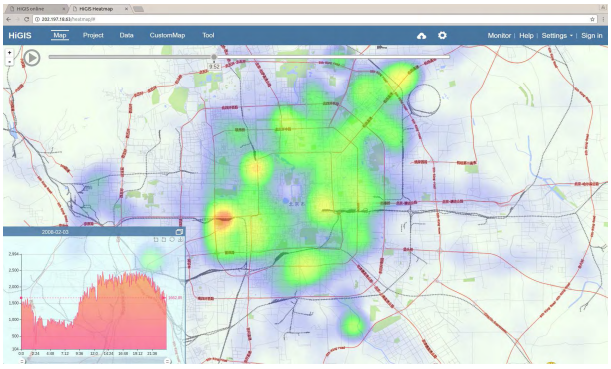

FIGURE 16. Heat map of transportation in Beijing.

Additionally, as expected, the Geo-Gap tree outperformed PostGIS in range query. The reason is obvious, Geo-Gap is a progressive query method that returns results without querying all the data.

V. CONCLUSION

In this paper, the multi-scale index (Geo-Gap index) was investigated, which not only keeps the necessary data for the progressive visualization of large datasets, but enables more accurate query estimation. First, the index based on the geo-hash method has been shown experimentally to be faster than the gap tree. Second, by adding a table structure (Geo-Gap table), it is able to estimate the original data in the process of progressive visualization, and the estimated value will be approximate to the actual value and eventually reach it. Finally, by comparing with the traditional range queries, it turns out that our method is more efficient. The method is now successfully integrated into a high-performance geographic information system called HiGIS. It works effectively in computing heat map of transportation in Beijing and taxi aggregation query in Chengdu as illustrated in Figure 16 and Figure 17. Therefore, this method shows much potential.

However, there are two directions that can be exploited in the future. First, although the Geo-Gap tree is able to widely support polyline datasets, no sampling operation has been performed. second, it will be worthwhile to also consider how to sample dynamically.

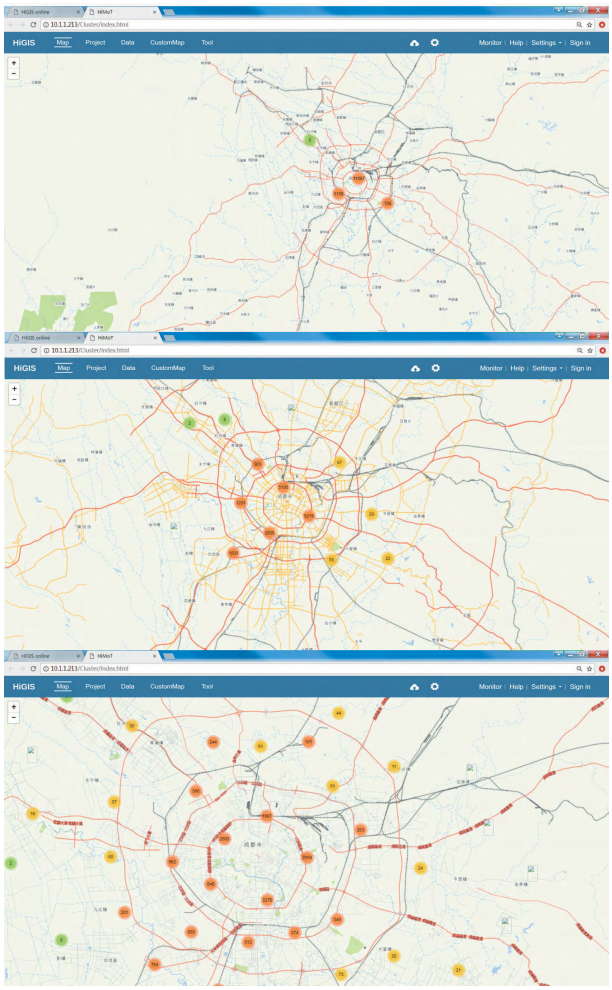

FIGURE 17. Taxi aggregation query in Chengdu at different level.

REFERENCES

[1] D. Fisher, "Big data exploration requires collaboration between visualization and data infrastructures," in *Proc. Workshop Hum. Loop Data Anal. (HILDA)*, New York, NY, USA, 2016, pp. 16:1–16:5.

[2] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big Data and cloud computing: Innovation opportunities and challenges," *Int. J. Digit. Earth*, vol. 10, no. 1, pp. 13–53, 2017.

[3] B. Zheng, H. Su, W. Hua, K. Zheng, X. Zhou, and G. Li, "Efficient clue-based route search on road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1846–1859, Sep. 2017.

[4] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *Proc. VLDB Endowment*, vol. 11, no. 11, pp. 1633–1646, 2018.

[5] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *Proc. VLDB Endowment*, vol. 9, no. 12, pp. 1053–1064, 2016.

[6] R. Šuba, M. Meijers, and P. V. Oosterom, "Continuous road network generalization throughout all scales," *ISPRS Int. J. Geo-Inf.*, vol. 5, no. 8, p. 145, 2016.

[7] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 755–766.

[8] F. Niu and C. Cheng, "Novel approaches to the multiscalar display of vector data," *Geo-Spatial Inf. Sci.*, vol. 15, no. 4, pp. 251–261, 2013.

[9] C. Cheng, F. Niu, J. Cai, and Y. Zhu, "Extensions of GAP-tree and its implementation based on a non-topological data model," *Int. J. Geograph. Inf. Sci.*, vol. 22, no. 6, pp. 657–673, 2008.

[10] L. Battle, M. Stonebraker, and R. Chang, "Dynamic reduction of query result sets for interactive visualizaton," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2013, pp. 1–8.

[11] D. Fisher, I. Popov, and S. Drucker, "Trust me, I'm partially right: Incremental visualization lets analysts explore large datasets faster," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2012, pp. 1673–1682.

[12] (2019). *Mapbox*. [Online]. Available: https://www.mapbox.com/

[13] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1997, pp. 171–182.

[14] F. Li, B. Wu, K. Yi, and Z. Zhao, "Wander join and XDB: Online aggregation via random walks," *ACM SIGMOD Rec.*, vol. 46, no. 1, pp. 33–40, 2017.

[15] D. Xie, F. Li, B. Yao, G. Li, Z. Chen, L. Zhou, and M. Guo, "Simba: Spatial in-memory big data analysis," in *Proc. 24th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, p. 86.

[16] X. Hu, M. Qiao, and Y. Tao, "Independent range sampling," in *Proc. 33rd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2014, pp. 246–255.

[17] R. Christensen, L. Wang, F. Li, K. Yi, J. Tang, and N. Villa, "STORM: Spatio-temporal online reasoning and management of large spatio-temporal data," in *Proc. SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1111–1116.

[18] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "BlinkDB: Queries with bounded errors and bounded response times on very large data," in *Proc. 8th Eur. Conf. Comput. Syst.*, 2013, pp. 29–42.

[19] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo, "The analytical bootstrap: A new method for fast error estimation in approximate query processing," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2014, pp. 277–288.

[20] C. Qin and F. Rusu, "Parallel online aggregation in action," in *Proc. Int. Conf. Sci. Stat. Database Manage.*, 2013, p. 46.

[21] L. Wang, R. Christensen, F. Li, and K. Yi, "Spatial online sampling and aggregation," *Proc. VLDB Endowment*, vol. 9, no. 3, pp. 84–95, 2015.

[22] Y. Yan, L. J. Chen, and Z. Zhang, "Error-bounded sampling for analytics on big sparse data." *Proc. VLDB Endowment*, vol. 7, no. 13, pp. 1508–1519, 2014.

[23] P. van Oosterom, "The reactive-tree: A storage structure for a seamless, scaleless geographic database," in *Proc. Auto-Carto*, 1991, pp. 393–407.

[24] P. Van Oosterom and V. Schenkelaars, "The development of an interactive multi-scale GIS," *Int. J. Geograph. Inf. Syst.*, vol. 9, no. 5, pp. 489–507, 1995.

[25] D. H. Ballard, "Strip trees: A hierarchical representation for curves," *Commun. ACM*, vol. 24, no. 5, pp. 310–321, 1981.

[26] A. Cecconi, *Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping*. Zürich, Switzerland: Univ. Zurich, 2003. doi: 10.3929/ethz-a-004553772.

[27] C. Cheng, F. Lu, and J. Cai, "A quantitative scale-setting approach for building multi-scale spatial databases," *Comput. Geosci.*, vol. 35, no. 11, pp. 2204–2209, 2009.

[28] N. Guo, W. Xiong, Y. Wu, L. Chen, and N. Jing, "A geographic meshing and coding method based on adaptive Hilbert-Geohash," *IEEE Access*, vol. 7, pp. 39815–39825, 2019. doi: 10.1109/ACCESS.2019.2906871.

[29] Q. Zhou and Z. Li, "A comparative study of various strategies to concatenate road segments into strokes for map generalization," *Int. J. Geograph. Inf. Sci.*, vol. 26, no. 4, pp. 691–715, 2012.

[30] B. Yang, X. Luan, and Q. Li, "Generating hierarchical strokes from urban street networks based on spatial pattern recognition," *Int. J. Geograph. Inf. Sci.*, vol. 25, no. 12, pp. 2025–2050, 2011.

[31] R. Weiss and R. Weibel, "Road network selection for small-scale maps using an improved centrality-based algorithm," *J. Spatial Inf. Sci.*, vol. 2014, no. 9, pp. 71–99. 2014.

[32] X. Wu, H. Zhang, Y. Xu, and J. Yang, *A Comparative Study of Various Properties to Measure the Road Hierarchy in Road Networks*. Singapore: Springer, 2017, pp. 157–166.

[33] W. Xiong and L. Chen, "HiGIS: An open framework for high performance geographic information system," *Adv. Elect. Comput. Eng.*, vol. 15, no. 3, pp. 123–132, 2015.

WEI XIONG was born in Hunan, China. He received the M.S. degree in computer engineering from the Naval University of Engineering, Wuhan, China, in 2001, and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2005, respectively. He is currently an Associated Professor with NUDT. His research interests include geographic information systems and spatial database.

RUIQING LI was born in Hunan, China, in 1993. He received the B.S. degree in information engineering and the M.S degree from the National University of Defense Technology, Changsha, China, in 2014 and 2017, respectively. His research interest includes spatial database.

JIN PENG was born in Hunan, China, in 1995. She received the B.S. degree in information engineering from the National University of Defense Technology, Changsha, China, in 2017, and the M.S. degree with the Spatial Information System Laboratory, National University of Defense Technology. Her research interests include spatial database and spatio-temporal data processing.

YE WU was born in Hunan, China. He received the M.S. degree in information engineering and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2008 and 2015, respectively. He is currently a Lecturer at NUDT. His research interests include geographic information systems and spatial database.

NING GUO was born in Jiangsu, China, in 1992. He received the B.S. degree in information engineering and the M.S. degree in information and communication engineering from the National University of Defense Technology, Changsha, China, in 2014 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Spatial Information System Laboratory. His research interests include GIS, spatial database, and spatio-temporal data modeling and processing.

NING JING was born in Chongqing, China, in 1963. He received the B.Sc. degree in communication and information systems, the M.Sc. degree in signal and information processing, and the Ph.D. degree in computer science from the National University of Defense Technology, in 1983, 1986, and 1990, respectively. He is currently a Professor and the Director of the Department of Information Engineering, National University of Defense Technology. His research interests include geographical information systems, database systems, planning and decision support in spatial resources, and spatial data analysis and visualization. He has served as an expert in earth observation and navigation area of the National High-Tech Research and Development Program of China. He is a Senior Fellow of the China Computer Federation (CCF), a Fellow of the Technical Committee of Database System of CCF, the Vice Director of the Technical Committee of Public Security, and a Fellow of the Technical Committee of Principles and Methods of the China GIS Association.