APPROXIMATE SPATIAL QUERY AND UPDATING ON VERTEX HIERARCHICAL STRUCTURE FOR INTERACTIVE VISUALIZATION

Agen Qiu¹*, Jiping Liu¹, Zhiran Zhang¹², Fuhao Zhang¹, Shenghua Xu¹

¹Chinese Academy of Surveying and Mapping; No. 28 Lianhuachi West Road, Haidian District, Beijing, 100830, China; ²Wuhan University; No.129 Luoyu Road, Wuhan, 430079, China; *Corresponding author; e-mail: qiuag@casm.ac.cn;

Abstract
Text of Abstract: The development of geospatial big data presents an urgent challenge for the interactive visualization of geographic data. By detecting hierarchical structure in geospatial data and approximate spatial query approaches, we developed an approximate spatial query method based on vertex sequence division and time constraint. This algorithm supports efficient querying through weighted breadth-first search (BFS) of trees, distributed in-memory computing and vertex binary tree on large spatial database. We then propose four locally updating algorithms that can achieve real-time generation and renewal of geographical element objects, including insert, delete, modify and tree rebalance operation of binary tree. Finally, we conduct extensive experiments on OpenStreetMap data to evaluate the proposed algorithms and data structures. Experiments show that our approximate spatial query and updating method can improve the query efficiency of data visualization, and the visual effect don’t have much difference between approximate spatial query and exact query.

Keywords: approximate spatial query, data updating, vertex hierarchical structure, interactive visualization

1 INTRODUCTION

As the data scale expanding and complicate, data volume is more and more big in visualization applications, the user demand for interactive visualization is higher and higher, and the range of application is becoming more and more widely. At the same time, visual application and service are increasingly demanding for computing and storage capacity of computers. The heterogeneity and diversity of data are major problems for visualization (Kumar 2015). It is an important challenge for large data visualization to support the selection of different display information and different types of display modes.

Traditional visualization tools and methods can hardly meeting the need of interactive visualization of big data (Hoskins 2014). The emergence of computers and the development of computer graphics make interactive visualization possible, cloud computing and advanced graphical interfaces contribute to the extensibility of big data (Sucharitha, Subash et al. 2014). Parallelization algorithms (Childs, Geveci et al. 2013, Kim, Ji et al. 2014) and data simplification algorithms (Elmqvist and Fekete 2009, Sarma, Lee et al. 2012) also significantly promote the development of visualization. However, direct visualization of large data sources may reduce the visual effect of data and hinder users' perception and cognitive ability. Many researchers use feature extraction and geometric modeling to reduce data size before the presentation of actual data. Several methods of interactive visualization through data reduction are discussed in Liu, Jiang et al. 2013., including filter (Ahlberg and Shneiderman 1994, Lee, Podlaseck et al., 2001), sampling (Mihalisin, Timlin et al., 1991, Sarma, Lee et al., 2012), polymerization (Carr, Littlefield et al., 1987, Nguyen and Huang, 2005) and modeling.

According to the difference of technical implementation, approximate query methods are mainly divided into two categories: online aggregation and sampling. Based on the statistical theory, query result of online aggregation is calculated in real-time, and then are used to compute confidence intervals (Acharya,Gibbons et al. 2000, Wu, Ooi et al. 2010). J.Hellerstein used online aggregation to set up the user interface, and change the model of long time retrieval in traditional database system. Based on the central limit theory and interval formula, Haas achieved single and multiple table query, such as AVG and COUNT query (Haas 1997). However, there are some limitations in this method, which is that the effect of spatial database is not stable. The second method – sampling presets the sample set for a particular query (Joshi and Jermaine 2009). According to the traditional sampling technique, Joshi proposed a stratified sampling method to combine the layered and bayesian framework. X. Meng described the simple random sampling method, and
extende it to stratified sampling, which can effectively reduce the size of intermediate output in MapReduce and greatly improve load balance (Meng 2013). Gibbons compared simple sampling with counting sampling (Gibbons and Matias 1999), and put forward a fast incremental maintenance technology. However, the design can not be used for spatial database because of the absence of substructure between elements. Surajif Chaudhuri analysed sampling theory, and proposed the idea that pre-calculated Samples could bridging the extremes of online and offline (Chaudhuri, Das et al. 2007).

Spatial data compression and simplification are also common application requirements. In recent years, vehicle trajectory simplification is a very representative example (Long, Wong et al. 2013, Long, Wong et al. 2014, Muckell, Olsen et al. 2014, Lin, Ma et al. 2017). Most studies do not discuss two important issues in spatial database applications: the large amount of data and data updates. Wang (Wang, Christensen et al. 2015) sampled data in the spatial database toward aggregate query. The main idea is that introducing sampling from general database into the spatial database, and then manage and organize these sampling data using relevant data structure. The main method is to take 1/2 sample and then organize the sampling data with an extended R tree. Due to the continuous growth of the spatial network data scale, the idea of using approximate calculation has produced a lot of results for network related queries in recent years. Based on the specific loss function, road network data is simplified and the approximate processing of related query is made based on the simplified network (Thorup and Zwick 2005, Tao, Sheng et al. 2011, Tao, Hu et al. 2013). This method has a good effect in real-time performance and give quite enlightening in the research of approximate spatial query and interactive visualization.

Spatial range query is the most basic spatial query, which is also called windowing query in two-dimensional case. Spatial range query can be considered an extension of range queries in high-dimensional space. There are a lot of possible scenarios in range query when query conditions have slightly changes, such as 3—sided range query. However, these studies do not deal well with the size of query results. In this paper, we let window query as the research type. Through vertex sequence division schema, hierarchical structure of geospatial data are built and organized. We then propose a locally updating algorithm that can achieve real-time generation and renewal of geographical element objects by approximate spatial querying.

2 METHODOLOGY

2.1 Vertex sequence division

Point, polyline and polygon elements are widely used to represent various geographical elements. In vector data set, the line is composed of two endpoints and a series of vertices which can mark line’s shape, and the polygon consists of a series of segments, these segments are connective, closed, disjointed. Therefore, in feature model, vertex is the smallest unit, and line object is defined on basis of the vertex object, the polygon object is defined on the basis of the line object. As a basic unit, vertex is an extremely important feature in feature model. In this paper, the approximate method is used to subdivide the vertex sequence. The approximate of the line object is realized by vertex sampling, and the approximate of the polygon object is realized by vertex sampling and line sampling.

The approximate process of a simple vertex sequence is summarized as follows. Given a polyline \( L_j = \{P_0, P_1, L, P_n\} \), The line \( L_j \) is not self - handed. If \( L_j \) is a closed line, that is \( P_i \neq P_j, \ i \neq j \) and \( S_{P_i, P_j} \cap S_{P_j, P_i} \in \{P_i\}, k = 0,1,...,n \). All vertices are only be passed once, and all adjacent vertices are composed of segments that have no internal intersections. For the sake of simplicity, this paper takes the parameters \( k = 2 \) in the recursive subdivision. Through a certain vertex selection strategy, \( L_j \) is divided into two subsequences by vertex \( P_j \), \( L_{xj} = \{P_i, P_{i+1}, ..., P_j\} \) and \( L_{yj} = \{P_j, P_{j+1}, ..., P_n\} \). Then repeat the above steps for each subline, until all the sublines only have two vertices.

In the above process, the inclusion relation of all vertex can be expressed directly by binary tree. The sequence of vertices illustrate that former vertices have higher significance. So when the whole process is over, a binary tree and sequential importance are built simultaneously. Each tree node corresponds to an vertex \( L_{xj} \), where \( i \) represents the sequence number of the tree node in the same layer, \( h \) represents the distance from the root tree node, the height of the node. In this binary tree, the leaf node represents a sequence of two adjacent vertices, and the intermediate node represents the vertex sequence with length greater than 2. The tree node is associated with the weight of the corresponding vertex sequence.
Each tree represents the vertex sequence of a space object, which can form a line object or a component ring of the surface object. For polygons with multiple islands or holes, all the trees formed by the inner and outer rings constitute binary trees, and of all trees constitute the forest of the data set.

2.2 Approximate spatial query based on binary tree

After the establishment of the tree hierarchy of geometric vertices, the operation of geographical elements is transformed into the operation of tree hierarchy. The approximation of the elements also translates into the sampling of the vertices contained in the elements. In the progress of approximate spatial query, if the weight value associated with the tree node is used as the sampling basis, then the approximate query is transformed into a top k query problem. Window approximate query can be reduced to a top k query problem in a certain range. The top k query based on tree structure can be processed by using the weighted breadth-first search algorithm. The traversal of tree node is realized by the weight descending order. Namely, tree nodes are selected according to the descending order of nodes’ weights. Because the minimum value is selected in the progress of traversal, vertices are stored in priority queue in order to improve performance.

There are two types of approximate query conditions: time constraint query and error constraint query. Because of the complex of query is $O(\log n + k)$, where $n$ represents the number of vertices for the elements, $k$ represents the number of vertices of the query result. Therefore, the time of query processing is linearly dependent on the data volume of the query result, and the time constraint can be converted into data volume constraint. In order to satisfy the time constraint query condition, we select vertices gradually and finish when the time is over.
This paper illustrates the process of approximate windowing query. We execute weighted breadth-first search algorithm in binary trees, and then select vertices in a certain range. The approximate query process is summarized as follows.

Input: query range \( W = \{x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}\} \), time threshold \( t \)

Output: polyline \( L'_i \)

1. Built a priority queue \( PQ \) and sampling set \( S_p \). Perform spatial operations on the root of the binary tree. If the space scope of the vertex in the subtree intersects with the window \( W \), then add this node to the priority queue \( PQ \);
2. If \( PQ \) is not null, select the node with the largest weight \( j_{P_i} \) from \( PQ \) and then add to the sampling set \( S_p \);
3. If \( j_{P_i} \) exist at \( W \), add its child nodes to \( PQ \);
4. If the algorithm executes in less than \( t \), repeat step 2 to 3; else, go to step 5;
5. The vertices of all the samples are arranged in sequence numbers and new line objects \( L'_i \) are dynamically generated. Finally, return the newly generated objects.

2.3 Update policy

The update of the hierarchical structure is the same as the query, which use the characteristics of data access in the relational database to update and maintain the data. The main operations include the reading of subsequence, the update of vertex weight, the deletion of old vertices and the addition of new vertices. Three types of update operations on sequences have different update domains. There are different update costs between sequence insertion, sequence deletion and sequence modification. Among them, sequence insertion is relatively simple, sequence deletion is more complex. There are two steps involved in modification, including sequence deletion and sequence insertion.

2.3.1 Insert operation

Set the original sequence as \( (P_0, P_1, \cdots, P_n) \), new sequence as \( (V_0, V_1, \cdots, V_m) \). If we insert a new sequence between the vertices \( P_i \) and \( P_{i+1} \) in the original sequence, the update domain of the insert operation is \( (P_i, P_{i+1}) \). Let’s say that the point \( P_{i+1} \) is a little bit further away from the root, we need to construct the new sequence \( (P_i, V_0, V_1, \cdots, V_m, P_{i+1}) \) as a subtree first, then take this subtree as the left subtree of \( P_{i+1} \). Before inserting the new subtree into the new location, the weight for all parent nodes from the root of the original sequence need to be recomputed. If these weights are greater than the original value, the weight associated with each tree node should be updated to the new value. Throughout the update process, the computational complexity of transforming \( (P_i, V_0, V_1, \cdots, V_m, P_{i+1}) \) into tree structure is \( O(m \log m) \).

The time complexity of the original tree nodes’ weight is \( O(m \log m) \). The input complexity is \( O(m + \log n) \). After the insert operation, the balance of the tree is changed, and the unbalance of the tree directly affects the reading performance of data. We set tree reconstruction threshold. In order to maintain the balance of the tree, after the insertion operation, if the new tree’s height exceeds this threshold, the tree structure will be reconstructed.

Figure 2 illustrates the operation of insertion. Where the gray tree nodes represent the vertices that need to be recalculated in the insertion process. The dark grey vertices represent the direct precursor and the successor nodes of the start and end nodes in new sequence, respectively. The direct precursor node of \( V_a \) is \( P_i \), the direct successor node of \( V_a \) is \( P_{i+1} \). The wavy line indicates that the intermediate node is omitted between the two nodes. Triangles represent subtrees, triangle with dotted line represents a subtree formed by a new sequence \( (V_0, V_1, \cdots, V_m) \).
2.3.2 Delete operation

Set the original sequence as \((P_0, P_1, \cdots, P_n)\). If \((P_{x_1}, P_{x_2}, \cdots, P_{x_m})\) is deleted from the original sequence, the update domain of the delete operation is the subtree with \(P_j\) as the root. Where \(P_j\) is the lowest common ancestor node of the vertex of \(P_i\) and \(P_{x_m}\). The vertex sequence of the subtree is \(P_{s_1}, P_{s_2}, \cdots, P_{s_{k-1}}, P_{s_k}, \cdots, P_{s_{k+m}}, \cdots, P_i\), with \(P_i\) as root node. \(l\) represents the height of \(P_j\) in binary tree, the length of the vertex sequence corresponding to the subtree of root node \(P_j\) is \(O(2^{h+l})\), where \(h = O(\log n)\) is the height of original tree. In the progress of sequence deletion operation, we remove node \(P_j\). Then construct the sequence \(P_{s_1}, P_{s_2}, \cdots, P_{x_1}, P_{x_2}, \cdots, P_{x_m}, \cdots, P_i\) into a subtree. The computational complexity is \(O(2^{h+l}(h-l))\). Figure 3 shows the progress of the sequence deletion operation, where the dark gray vertex is the first and end vertex of the deleted sequence.

![Figure 2. Binary tree updating when insert operation is performed](image)

![Figure 3. Binary tree updating when delete operation is performed](image)
2.3.3 Modify operation

Set the original sequence as \((P_1, P_2, \ldots, P_n)\). If the update operation is to replace the new sequence \((V_1, V_2, \ldots, V_m)\) with \((P_1, P_2, \ldots, P_n)\) in the original sequence. This operation includes two steps. Firstly, the subsequence \((P_{k+1}, P_{k+2}, \ldots, P_n)\) is deleted. Then insert new subsequence \((V_1, V_2, \ldots, V_m)\) between node \(P_k\) and \(P_{k+1}\). \(P_j\) is the lowest common ancestor node of the vertex of \(P_k\) and \(P_{k+1}\). Replace the corresponding vertices \(P_1, P_2, \ldots, P_{k+1}, P_{k+2}, \ldots, P_n\) of the subtree \(P_j\) to \((V_1, V_2, \ldots, V_m)\), and a new sequence \(P_1, P_2, \ldots, P_{k+1}, V_1, V_2, \ldots, V_m, P_{k+2}, \ldots, P_n\) is generated. A new subtree is generated and connected to the location of the node \(P_j\).

![Figure 4. Binary tree updating when modify operation is performed](image)

2.3.4 Tree rebalance operation

Local update operation on tree structure, include insert, delete and modify operation, greatly increase the update efficiency. These update operations are performed on local subtrees. However, the balance of tree will be destroyed after several operations. At this time, it is necessary to carry out the rebalancing operation. This operation rebuild subtree with serious imbalance and reduce tree height. Tree rebalance operation will guarantee the balance of sublines’ scale and binary tree if the polyline is stored in a hierarchy structure of binary tree. As a result, it can reduce the complexity of algorithms and enhance the efficiency of hierarchy structure.

![Figure 5. The rebalance of tree structure](image)

3 EXPERIMENTS AND DISCUSSION

3.1 Dataset

The dataset of the experiment derived from the whole library file Planet.osm of OpenStreetMap. We extracted total factor data of global coastline through OSMCoastline software. This dataset is the most accurate coastline data. The amount of data on this scale will have a serious impact on performance, both in query, transmission and mapping. Table 1 shows the item and characteristics of data.

Vertex division and approximate spatial query run in a Hadoop cluster computing environment with 8 cluster nodes. The experiment server is built on Redhat 6.5 with Intel Xeon E7-8870 of CPU, 128G of memory, 1000M of network card. The development environment for experiment is Eclispe3.7 and the Java version is jdk1.7.65.
### Table 1. Data description of global coastline

<table>
<thead>
<tr>
<th>Data item</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertices</td>
<td>43591835</td>
</tr>
<tr>
<td>polyline</td>
<td>878453</td>
</tr>
<tr>
<td>assembly objects</td>
<td>15175</td>
</tr>
<tr>
<td>the maximum vertex number of polyline</td>
<td>4370376</td>
</tr>
<tr>
<td>the maximum polyline number of assembly objects</td>
<td>52470</td>
</tr>
<tr>
<td>the number of enclosed polygon</td>
<td>572926</td>
</tr>
</tbody>
</table>

#### 3.2 Vertex division based on VW algorithm

Visvalingam-Whyatt (VW) algorithm is a simplified method based on curve graph analysis. The basic idea is summarized as follows. Given a threshold of distance $\varepsilon$ and a polyline $L = \{P_0, P_1, L, P_n\}$. For each vertex $P_k, 0 < k < n$, we calculate the area of a triangle which is formed by lines joining points $P_{k-1}, P_k, P_{k+1}$, i.e., $S \Delta P_{k-1} P_k P_{k+1}$. For vertex $P_k$ and any vertex $P_m$, $S \Delta P_{k-1} P_k P_m \leq S \Delta P_{k-1} P_k P_{m+1}, 0 < k, m < n$. If $S \Delta P_{k-1} P_k P_m \leq \varepsilon$, then delete $P_m$, draw a straight line joining points $P_{k-1}$ and $P_{k+1}$. Repeat the above steps for the new polyline, until all the areas of triangles satisfying the specified criterion function

$$f(S_m) = \max \{S \Delta P_{k-1} P_k P_{k+1} \leq \varepsilon \} \quad (1)$$

The algorithm is the process of deleting the vertex successively. If the order of the vertex deletion is arranged in reverse order, it can be regarded as the step selection of the line object to sample and form the vertex sequence process. The area of a triangle can be seen as the weight of vertices.

Based on the vertex sequence division method and distributed computing environment, we applied VW algorithm to built binary tree. The preprocessing consists of two items: (1) the calculation of vertices’ weight and the generation and storage of binary trees for polylines; (2) the generation and storage of vertex spatial index. The construction time of binary tree and quadtree indexing are showed as follows.

### Table 2. Distributed preprocessing time of binary tree

<table>
<thead>
<tr>
<th>Vertices number</th>
<th>10,000</th>
<th>20,000</th>
<th>30,000</th>
<th>40,000</th>
<th>50,000</th>
<th>60,000</th>
<th>70,000</th>
<th>80,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46.270</td>
<td>57.868</td>
<td>87.308</td>
<td>108.161</td>
<td>127.422</td>
<td>145.188</td>
<td>189.114</td>
<td>222.100</td>
</tr>
<tr>
<td>2</td>
<td>36.904</td>
<td>50.038</td>
<td>60.656</td>
<td>80.796</td>
<td>88.758</td>
<td>99.260</td>
<td>129.236</td>
<td>139.056</td>
</tr>
<tr>
<td>3</td>
<td>30.857</td>
<td>49.278</td>
<td>57.467</td>
<td>62.121</td>
<td>61.984</td>
<td>80.665</td>
<td>84.112</td>
<td>87.657</td>
</tr>
<tr>
<td>4</td>
<td>33.766</td>
<td>43.122</td>
<td>52.187</td>
<td>56.826</td>
<td>60.568</td>
<td>66.711</td>
<td>74.555</td>
<td>76.866</td>
</tr>
<tr>
<td>5</td>
<td>28.832</td>
<td>37.433</td>
<td>42.777</td>
<td>44.429</td>
<td>50.976</td>
<td>60.365</td>
<td>67.343</td>
<td>71.602</td>
</tr>
<tr>
<td>6</td>
<td>31.594</td>
<td>32.263</td>
<td>41.682</td>
<td>44.459</td>
<td>46.239</td>
<td>54.557</td>
<td>58.286</td>
<td>68.193</td>
</tr>
<tr>
<td>7</td>
<td>28.240</td>
<td>33.968</td>
<td>38.874</td>
<td>42.623</td>
<td>44.399</td>
<td>50.164</td>
<td>56.828</td>
<td>58.629</td>
</tr>
<tr>
<td>8</td>
<td>29.947</td>
<td>33.144</td>
<td>39.970</td>
<td>43.403</td>
<td>45.691</td>
<td>49.941</td>
<td>59.391</td>
<td>62.150</td>
</tr>
</tbody>
</table>
### Table 3. Construction time for quadtree spatial index

<table>
<thead>
<tr>
<th>Vertices number</th>
<th>5,000,000</th>
<th>10,000,000</th>
<th>15,000,000</th>
<th>20,000,000</th>
<th>25,000,000</th>
<th>30,000,000</th>
<th>35,000,000</th>
<th>40,000,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120.086</td>
<td>167.267</td>
<td>203.830</td>
<td>278.698</td>
<td>335.517</td>
<td>379.147</td>
<td>561.211</td>
<td>679.353</td>
</tr>
<tr>
<td>2</td>
<td>95.311</td>
<td>138.872</td>
<td>167.772</td>
<td>182.275</td>
<td>246.431</td>
<td>282.911</td>
<td>356.756</td>
<td>422.398</td>
</tr>
<tr>
<td>3</td>
<td>84.362</td>
<td>144.658</td>
<td>147.023</td>
<td>179.898</td>
<td>236.471</td>
<td>262.580</td>
<td>379.712</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>97.328</td>
<td>127.272</td>
<td>142.684</td>
<td>157.142</td>
<td>170.818</td>
<td>228.485</td>
<td>301.661</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>79.441</td>
<td>115.581</td>
<td>130.264</td>
<td>154.309</td>
<td>198.381</td>
<td>228.716</td>
<td>287.925</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>78.170</td>
<td>118.052</td>
<td>131.847</td>
<td>135.163</td>
<td>193.151</td>
<td>216.615</td>
<td>277.068</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>79.085</td>
<td>113.138</td>
<td>122.196</td>
<td>123.763</td>
<td>153.291</td>
<td>195.320</td>
<td>249.253</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>70.056</td>
<td>107.056</td>
<td>113.120</td>
<td>119.933</td>
<td>134.370</td>
<td>188.954</td>
<td>241.447</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows the distributed preprocessing time of vertices’ weights and binary tree for polylines. The different rows represent the computation time from 1 to 8 compute nodes. It can be seen from the table that the processing time decreases gradually as the number of compute nodes increases, which fully reflects the role of distributed in improving efficiency. In the large geospatial data, VW algorithm deal well with polylines’ simplification.

In order to reduce the storage consumption for algorithm and enhance the robustness and portability of the algorithm, we make data partition through building vertex hierarchy index structure combined with quadtree. Table 3 shows the construction time for spatial index. The different rows represent the computation time from 1 to 8 compute nodes. It can be seen from the table, the construction time decreases gradually as the number of compute nodes increases, which fully reflects the role of distributed in improving efficiency. This rate of ascension is slower than the increase of the compute nodes.

#### 3.3 Query efficiency analysis

The analysis of approximation spatial query is applied to the global coastline. We set query time as the threshold, and points number is chosen as the reference index to evaluate the result. In order to clearly illustrate the result of approximate window query, we analysis the visualization effects from different scales and different dimensions. We set query time to 0.5 second, when the time is over, the query stops.

As is shown in Figure 6, the red lines represent vector data of the World for approximate queries, and display on client with Google map. The number of returned points is 14,403. From Figure 7, it appeared that the boundary is basically consistent with the background map, white space represents the difference between query result based on VW algorithm and original data. If the initial data were displayed directly on the map, it would spend 50 seconds. The result shows that the response time going through the method is faster than when going on original data, and the query result is basically meet the requirements of visualization.
Figure 6. Query result for the World in 2D

Figure 7. Query result for the World in 3D

Figure 8. Query result for Australia in 2D

Figure 9. Query result for Australia in 3D
Figure 8 shows the query result of Australia, red lines represent vector data of Australia for approximate queries, and display on client with Google map. The number of returned points is 5,403. From Figure 9, it appeared that the boundary is basically consistent with the background map, white space represents the difference between query result based on VW algorithm and original data. If the initial data were displayed directly on the map, it would spend 10 seconds. The result shows that the response time going through the method is faster than when going on original data, and the query result is basically meet the requirements of visualization.

4 CONCLUSIONS

The development of geospatial big data presents an urgent need for the interactive visualization of geographic data. Although various algorithms exist for spatial querying, their heavy precomputation storage costs or query costs hinder their application to efficient data accessing and real-time visualization. Spatial approximation query is an effective way to balance spatial query performance and accuracy of calculation, it is based on binary tree which is built on vertex sampling, and could realize rapid query with certain time and high accuracy.

By detecting hierarchical structure in geospatial data and approximate spatial query approaches, we developed an approximate spatial query algorithm that combines vertex sequence division scheme with weight constraint. Querying according to the weight of vertices and time limit is a well-received technique, and distributed environments will bring more benefits to approximate queries as the volume of data increase. This paper analyzes the contradiction between the current spatial query technology and the requirement of data interactive visualization, and puts forward a vertex sequence division and binary tree construction scheme based on VW algorithm. With the continuous updating of geospatial data, it is necessary to realize the dynamization of vertex binary tree. In view of this, we defines cost functions and updating algorithm from four aspects, including insert, delete, modify and tree rebalance operation of vertex binary tree. Finally, we conduct extensive experiments on OpenStreetMap data to evaluate the proposed algorithms and data structures. The internet GIS proto-system is established based on distributed in-memory computing. The results show that our approximate spatial query and updating method can improve the query efficiency of data visualization, and the visual effect don’t have much difference between approximate spatial query and exact query.

In this paper, the proposed scheme is only given for approximate window query. Further research will focus on other applications such as clustering analysis, network analysis of approximate query processing.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China (No. 2016YFC0803108) and the Basic Research Fund of CASM (No. 7771701).

REFERENCES


