Research Article

Agen Qiu, Zhiran Zhang*, Xinlin Qian, and Wangjun He

Error-bounded and Number-bounded Approximate Spatial Query for Interactive Visualization

https://doi.org/10.1515/geo-2018-0039
Received Mar 21, 2018; accepted Jul 24, 2018

Abstract: In the big data era, an enormous amount of spatial and spatiotemporal data are generated every day. However, spatial query result sets that satisfy a query condition are very large, sometimes over hundreds or thousands of terabytes. Interactive visualization of big geospatial data calls for continuous query requests, and large query results prevent visual efficiency. Furthermore, traditional methods based on random sampling or line simplification are not suitable for spatial data visualization with bounded errors and bound vertex numbers. In this paper, we propose a vertex sampling method—the Balanced Douglas Peucker (B-DP) algorithm—to build hierarchical structures, where the order and weights of vertices are preserved in binary trees. Then, we develop query processing algorithms with bounded errors and bounded numbers, where the vertices are retrieved by binary trees' breadth-first-searching (BFS) with a maximum-error-first (MEF) queue. Finally, we conduct an experimental study with OpenStreetMap (OSM) data to determine the effectiveness of our query method in interactive visualization. The results show that the proposed approach can markedly reduce the query results' size and maintain high accuracy, and its performance is robust against the data volume.

Keywords: approximate spatial query; interactive visualization; bounded errors; bounded numbers; B-DP algorithm

***Corresponding Author: Zhiran Zhang:** School of Resources and Environmental Sciences, Wuhan University, Wuhan 430079, China; Email: zzranature@163.com
Agen Qiu: Chinese Academy of Surveying and Mapping, Beijing 100830, China; Email: qiuag@casm.ac.cn
Xinlin Qian: Chinese Academy of Surveying and Mapping, Beijing 100830, China; Email: xinlinqian@vip.qq.com
Wangjun He: Chinese Academy of Surveying and Mapping, Beijing 100830, China; Email: hewj@casm.ac.cn

1 Introduction

With the rapid growth in data velocity, volume and variety, methods to efficient query and visualize massive amounts of geospatial data are attracting increasing attention. In an interactive exploration of spatial data, users want to zoom in or out to a particular area on a map without long wait times. These operations mean changes to the users' query condition—the query scope. A query request will be executed on a database when the query scope changes. On large spatial datasets, waiting for the exact analytical or query results may take a very long time. It is a dilemma that allows users to adjust the tradeoff between the query cost and the approximation quality.

Although various algorithms exist for spatial querying, the heavy calculation burden of spatial querying and large query result set will take a longer processing time in space querying and present a lower speed in transmission and rendering, so the implementation of interactive and real-time visualization is more difficult [1–3]. In particular, when the underlying data amount is large, reporting all points that satisfy a query condition and displaying them on screen could be expensive, and it may also reduce the visual effect of the data, hindering the users' perception and cognitive ability at the same time, since there could be too many points [4]. An approximate spatial query is a technique that samples a small portion of the data to process and returns an approximate result with an error or time bound. Providing approximate answers to spatial queries gives users the ability to focus their explorations quickly and effectively. It is even better if the query results are displayed on screen with a satisfactory visual effect.

Recently, new methods have arisen in the fields of databases, computers and visualization. Scholars in the database field propose online aggregation [4–6] and sampling [7–9] methods to solve the storage problem of large query results [5, 10–12]. Currently, to solve the problem of a heavy calculation burden in querying, cloud computing, distributed computation [13–19] and advanced graphical user interfaces contribute to the scalability of big data [20] in the computer field. These techniques provide good performance in terms of querying time. However, these tech-

niques exhaust the computation resources and block other time-sensitive jobs. In the fields of spatial databases and visualization, the combination of querying the data from databases and data simplification, such as data filtration and sampling [4, 21], model simplification [22], binning algorithm [23] and mixed methods [24–27], have become efficient methods to simplify the large query results. Alternatively, a majority of sampling methods are used to build data sketches offline and answer queries at runtime.

Data simplification, such as sampling or filtering, can reduce outliers and retain the basic structure of the data, thereby reducing the query's execution time. At the same time, a reasonable data simplification scheme can extract less feature data, and the difference between simplified data and original data is so small that it is practically undetectable by the naked eye. Traditional offline sampling methods cannot provide error bounds and only minimize data [10, 11]. Some methods, such as online aggregation [5] and bootstrapping [28], can provide error bounds; however, they aim to query and provide interactive visualization of nonspatial data. These methods cannot be directly applied to the interactive visualization of geospatial big data when considering the remaining geometric and topology characteristics in the process of data sampling.

The goal of the paper is to select a small set of vertices from a full-detailed spatial database through setting error or number thresholds, and make the visualization more efficient and extremely fast at the same time. The method described in this paper realizes approximate query processing of big spatial data. In vertex sampling method, the data visualization error was defined by Hausdorff Distance, and the spatial objects are sampled by the Balanced Douglas Peuker algorithm. In addition, based on the tree structure, error-bounded and number-bounded spatial query methods reduce the response time of the spatial query and make the real-time interaction possible. The algorithm is tested on OSM data and is found to achieve good performance.

2 Methods

2.1 Research Process

The point, polyline and polygon are widely used to represent various geographical features. In vector datasets, a polyline is composed of two endpoints and a series of vertices that can mark a line's shape; a polygon consists of a series of segments, and these segments are connective, closed and disjointed. Therefore, the vertex is the smallest unit in the feature model, the polyline is defined on the basis of the vertex, and the polygon is defined on the basis of the polyline. As a basic unit, the vertex is an extremely important feature in the 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 research is divided into four steps, and Figure 1 shows the basic process of this method.

1. Vertex sampling. Building vertex sequences for space objects by using the line simplification algorithm. The two-line simplification algorithm, *i.e.*, the traditional Douglas Peucker (DP) algorithm, and our B-DP algorithm will be illustrated in section 2.2.
2. Binary tree construction. A binary tree is built by the B-DP algorithm. It can directly represent the vertices of spatial objects in a hierarchical structure and a particular sequence. Section 2.3 describes the generation and connection of the binary tree.
3. Approximate spatial querying with bounded errors and bounded numbers. The vertices of features are retrieved by the binary trees' breadth-first-searching, and query execution can be terminated whenever the error or number reaches a satisfactory level.
4. Interactive visualization. The approximate query results are visualized in real time.

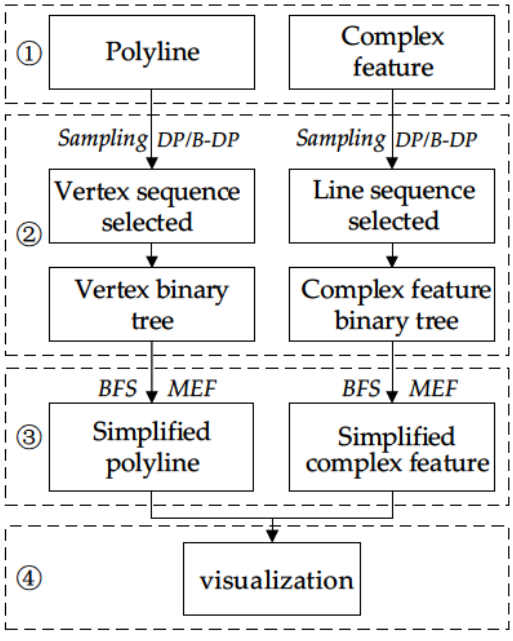


Figure 1: Research process

2.2 Vertex Sampling Method

A better sampling scheme will increase the query efficiency with high accuracy. In this section, two sampling methods are described, including the DP and B-DP algorithms. DP algorithm is a classic line simplification algorithm that can effectively simplify line objects. However, a sampling scheme that leads to a balanced hierarchical structure and satisfies the global error constraint is required. Therefore, we introduce a balanced factor in the DP to build a balanced binary tree. The advantages and construction method will be illustrated in section 2.2.2.

2.2.1 DP Algorithm

The purpose of a DP algorithm is to compress a large number of redundant vertices and find a similar curve with fewer vertices. The algorithm defines 'dissimilar' based on the maximum distance between the original curve and the simplified curve. The DP algorithm is summarized as follows [29,30]. Given a polyline $L_j = \{P_0, P_1, \cdots, P_n\}$ with a set C, and a simplified polyline L'_j with a subset $C' \subseteq C$. Initially, for vertex P_k, P_m,$0 < k, m < n$, if

$$dist\left(P_m\left\langle\overline{P_0, P_n}\right\rangle\right) = \max\left\{dist\left(P_k\left\langle\overline{P_0, P_n}\right\rangle\right)\right\} \geq \varepsilon \quad (1)$$

Which indicates that vertex P_m is to be kept, *i.e.*, $P_m \in C'$, otherwise it marks the straight line segment $\overline{P_0, P_n}$ as the simplified polyline. Polyline L_j is divided into two sublines by vertex P_m, then the above steps for the sublines are repeated until all the vertices satisfy the specified criterion function, as follows:

$$f(S_k) = \max\left\{dist\left(P_i\left\langle\overline{P'_{k-1}P'_k}\right\rangle\right)\right\} \leq \varepsilon \quad (2)$$

where $\varepsilon = const$, $P_i \in S_k$, and $\overline{P'_{k-1}P'_k}$ represents the straight line segment from P'_{k-1} to P'_k.

The DP algorithm is a global algorithm based on the whole curve, and samples the vertex by considering the entire character of the line object. The number of vertices $num\left(C'\right)$ in L'_j is decided by the constant ε. If $\varepsilon \leq 0$, then $num\left(C'\right) = num(C)$. An example of polyline simplification based on the DP algorithm is shown in Figure 2(b).

2.2.2 B-DP Algorithm

The DP algorithm is described in Figure 2(b), which uses the maximum-distance criterion and divides the polylines according to the vertices of maximum distance. Although the DP algorithm does not have to introduce new vertices and results in a low number of vertices, the sizes of the two sublines produced by each iteration may not be balanced. Therefore, we introduce a balance parameter α based on the DP algorithm to solve this problem. The B-DP algorithm is summarized as follows. $P_{n/2}$ is the center vertex of set C.

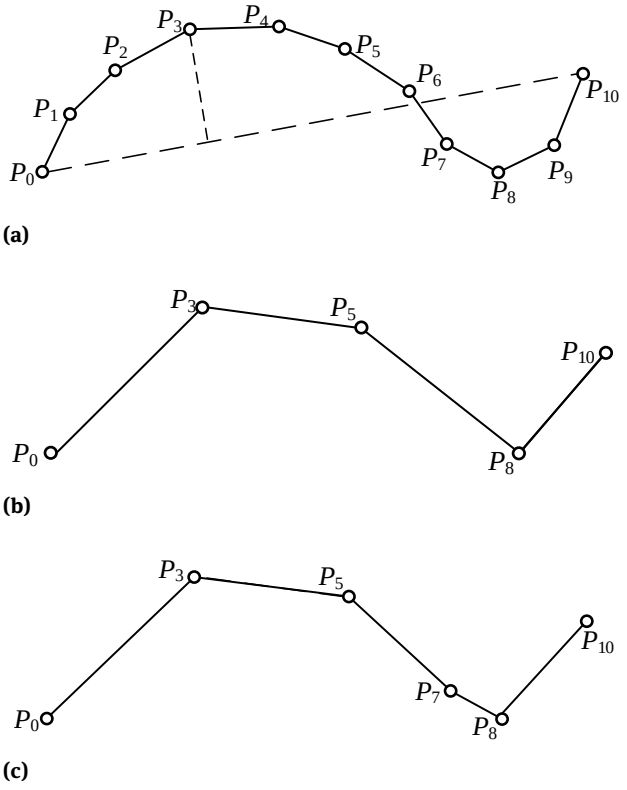


Figure 2: Example of polyline simplification: (a) polyline with a straight line; (b) the result of the DP algorithm; (c) the result of the B-DP algorithm ($\alpha = 0.3$)

1. Calculate the distance from vertex $P_i\,(0 < i < n)$ to straight line $\overline{P_0, P_n}$, if there exists vertex $P_m\,(0 < m < n)$ satisfying $dist\left(P_m\left\langle\overline{P_0, P_n}\right\rangle\right) = \max\left\{dist\left(P_i\left\langle\overline{P_0, P_n}\right\rangle\right)\right\} \geq \varepsilon$, do step (2); else, do step (3);
2. Select vertex P_t as the split point and mark P_t to be kept, *i.e.*, $P_t \in C'$, if the condition $dist\left(P_t\left\langle\overline{P_0, P_n}\right\rangle\right) = \max\left\{dist\left(P_k\left\langle\overline{P_0, P_n}\right\rangle\right)\right\}$ is met, $t, k \in \left\lfloor\frac{n}{2}\right\rfloor + \left[-\frac{(1-2\alpha)n}{2}, \frac{(1-2\alpha)n}{2}\right]$, $0 \leq \alpha < 0.5$;
3. Mark the straight line segment $\overline{P_0, P_n}$ as the simplified polyline;
4. Polyline L_j is divided into two sublines $L_{left} = \{P_0, P_1, \cdots, P_t\}$ and $L_{right} = \{P_t, P_{t+1}, \cdots, P_n\}$. Then, repeat steps 1 and 2 on the sublines.

Where $\varepsilon = const$, $\overline{P_0, P_n}$ is the connection line between P_0 and P_n; $dist\left(P_m\left\langle\overline{P_0, P_n}\right\rangle\right)$ represents the vertical distance from point P_m to line $\overline{P_0, P_n}$; $\max\left\{dist\left(P_k\left\langle\overline{P_0, P_n}\right\rangle\right)\right\}$

represents the maximum distance from point P_k to line $\overline{P_0, P_n}$, $k \in \lfloor \frac{n}{2} \rfloor + \left[-\frac{(1-2\alpha)n}{2}, \frac{(1-2\alpha)n}{2} \right]$. An example of polyline simplification based on the B-DP algorithm is shown in Figure 2(c).

Using the B-DP algorithm, the maximum-distance is also the criteria of division, and the approximate intermediate vertex is selected as the split point. This method guarantees the balance of the sublines' scale and binary tree if the polyline is stored in a binary tree, which will be illustrated in section 2.3. As a result, it can reduce the complexity of the algorithms and enhance the efficiency of the hierarchical structure.

2.2.3 Error Calculation

Error calculation is the core concept in an approximate spatial query. The difference between an original line and a simplified line is called the error, which is matched through measurements according to application scenarios. There are many measures of errors, such as the length ratio [31], sinuosity [32] and position error [33]. For visualization, the error between the original line and simplified line represents pixel-value differencing of these two lines. The characteristic is that there may be minimal pixel differences on the visual interface, while the original and simplified line has great difference. We define the Hausdorff distance based on the relative error of the pixel.

The generalized Hausdorff distance [34] is defined as follows:

$$d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\} \tag{3}$$

where X and Y are two non-empty subsets of a metric space, *sup* represents the supremum, *inf* the infimum and $d(x, y)$ the Euclidean distance between x and y.

L_j' is a simplification of L_j through the sampling method, such as the DP or B-DP algorithm, and the Hausdorff distance between L_0 and L_0' is as follows:

$$d_H\left(L_j, L'_j\right) = \max\left\{d\left(P_i, L'_j\right)\right\} \tag{4}$$

where $P_i \in L_j$, $d(P_i, L_j') = min\left\{d\left(P_i, \langle \overline{P'_k, P'_{k+1}} \rangle\right)\right\}$, $\forall \overline{P'_k, P'_{k+1}} \in L_j'$, $d\left(P_i \langle \overline{P'_k, P'_{k+1}} \rangle\right)$ is the distance from P_i to segment $\overline{P'_k, P'_{k+1}}$, which is also called the error of P_i.

An example is given in Figure 5. One square represents one pixel. Gray squares represent that they are passed by the line. The visual error between L_j and L_j' is the Hausdorff distance, which is defined based on pixels. Compared with the procedure of the B-DP algorithm, it is not difficult to find that the error is calculated during the execution of the B-DP algorithm. Therefore, the advantage is that the line is simplified, and the error is obtained without additional computation.

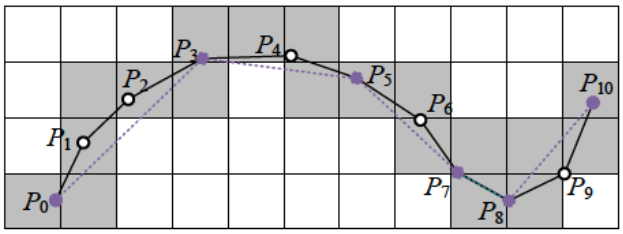


Figure 3: Visualization error of line

2.3 Binary Tree of Vertices

The order of the vertices is generated by the vertex sampling method. The previous vertices that were sampled have higher weights than the later sampled vertices. If we store the vertices in order in a balanced binary tree, it will not only reflect the hierarchical structure of the vertices but will also accelerate the query time in a large amount of the spatial data. This section describes the tree generation method based on the B-DP algorithm and the connection method of the binary tree.

2.3.1 Binary Tree of B-DP Algorithm

Various methods can be used to select a vertex where a polyline is divided into two sublines and to divide the vertices in a polyline into two subsets. If the threshold ε is sufficiently small in the DP or B-DP algorithm, all vertices in the polyline will be kept. In each iteration, the set C is divided into two subjects. This iteration continues until all the vertices are marked in set C'. The set C' has a unique sequence under a certain criterion, and it can be interpreted as the nodes of a binary tree, where the leaves of the tree are associated with the ordered set C'. A binary tree represents a polyline where the leaves are associated with the maximum-distance of vertices. The binary tree contains the vertices in sequence and establishes a hierarchical structure.

The maximum-distance is the criteria of the DP algorithm, and it cannot guarantee the balance of the tree. In some cases, the binary tree degenerates into an approximate chain or chain, and the time complexity of the query is linear $O(n)$; it can increase the query costs and reduce the query efficiency accordingly. A balanced binary tree is

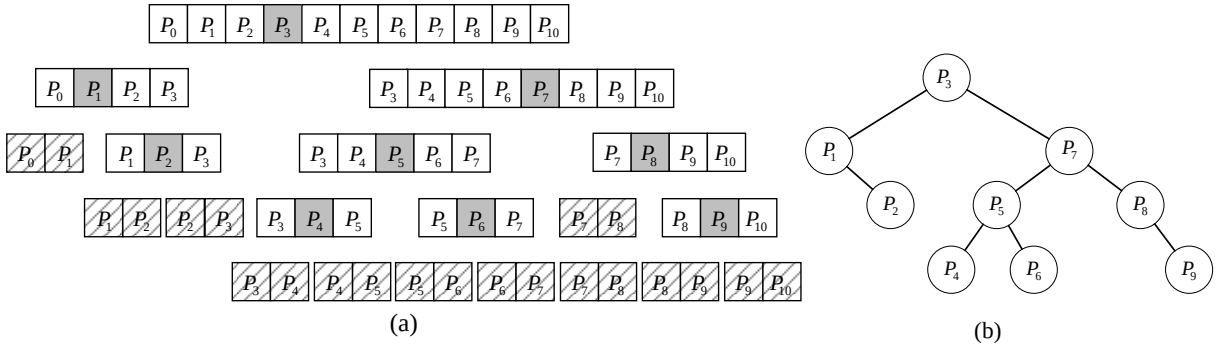


Figure 4: Example of the binary tree generation based on the B-DP algorithm: (a) vertex recursive sampling; (b) tree generated for Figure 2(a)

generated through the B-DP algorithm. Because the B-DP algorithm selects split points in certain central vertices, it can effectively balance the tree through a parameter. The time complexity of the query, insertion and deletion are generally well maintained in $O(logn)$. Therefore, the B-DP algorithm is a very useful tree generation method that greatly reduces the time complexity.

Figure 4(a) depicts a sampling progress of the polyline in Figure 2(a). The gray vertices are the vertices selected during sampling. A binary tree is generated according to the vertices sampling sequence. The original sequence of the vertices in the polyline will be reduced by the preorder traversal of the binary trees.

2.3.2 Connection of the Binary Tree

A complex feature is composed of multiple polylines linked end-to-end. The feature represents complex geographic entities in the real world, including closed planar entities (*e.g.*, larger areas of water, boundaries and settlement places) and linear features with wide geographical ranges (*e.g.*, roads, boundary lines and water systems). A complex feature not only is more complex than a single polyline but also contains important geographic dataset content. Thus, a complex feature is also the main query object in the progress of the approximate spatial query. In this section, we will discuss how to connect binary tree structures $T_0, T_1, \cdots, T_n$ corresponding to multiple polylines $L_0, L_1, \cdots, L_n$ to the tree structure $T_R = \varpi(R)$ corresponding to a complex feature $R = L_0 + L_1 + \cdots + L_n$.

The basic procedure of the binary tree's connection is described as follows. Perform the B-DP algorithm for all vertices in R, while split points can only be selected from the beginning points $P_{L_i,0}$ and end points $P_{L_i,n}$ of such lines L_i, $(0 \le i \le n)$. The binary tree is established until all beginning and end points are selected through recursive implementation of the B-DP algorithm.

1. Transform the sequence of the polyline $\{L_0, L_1, \cdots, L_n\}$ into vertex sequence $C = \{P_{L_0,0}, P_{L_0,1}, \cdots, P_{L_1,0}, P_{L_1,1}, \cdots, P_{L_n,0}, P_{L_n,1}, \cdots, P_{L_n,h}\}$, where the connection points only appear once.
2. Calculate the distance from vertices $P_i \in \{P_{L_0,1}, \cdots, P_{L_1,0}, P_{L_1,1}, \cdots, P_{L_n,0}, P_{L_n,1}, \cdots, P_{L_n,h-1}\}$ to straight line $\overline{P_{L_0,0}, P_{L_n,h}}$, the maximum-distance vertex P_m satisfying $dist\left(P_m \left\langle \overline{P_{L_0,0}, P_{L_n,h}} \right\rangle\right) = \max\left\{dist\left(P_i \left\langle \overline{P_{L_0,0}, P_{L_n,h}} \right\rangle\right)\right\}$, and record the maximum-distance E_{L_0,L_n};
3. Select the maximum-distance vertex $P_k \in \{P_{L_1,0}, P_{L_2,0}, \cdots, P_{L_n,0}, P_{L_n,h}\}$ as the split point. Then, set C is divided into two subsets, *i.e.*, C_{left} and C_{right}.
4. Establish a tree node T_{L_i,L_j}, which is associated with E_{L_0,L_n};
5. Repeat steps 2 to 4 on subsets C_{left} and C_{right}, and the generated tree nodes are the left and right child node of T_{L_i,L_j};
6. Return T_{L_i,L_j}.

where $P_{L_i,0}$ and $P_{L_i,n}$ are the beginning and end points of lines L_i, $(0 \le i \le n)$; $\overline{P_{L_0,0}, P_{L_n,h}}$is the connection line between $P_{L_0,0}$ and $P_{L_n,h}$; $dist\left(P_i \left\langle \overline{P_{L_0,0}, P_{L_n,h}} \right\rangle\right)$ represents the vertical distance from point P_i to line $\overline{P_{L_0,0}, P_{L_n,h}}$. E_{L_0,L_n} represents the maximum distance of $dist\left(P_i \left\langle \overline{P_{L_0,0}, P_{L_n,h}} \right\rangle\right)$; Subset C_{left} contains vertices from $P_{L_0,0}$ to P_k; T_{L_i,L_j} represents the root node of the binary tree.

2.4 Approximate Spatial Query for Interactive Visualization

A spatial range query is one of the most basic spatial query types. It is also called a windowing query in two dimensions. In terms of data visualization, the screen, such as a computer display or phone screen, gives the scope of the display and query, and all the data returned from databases that satisfy the error constraint are presented on the screen. Therefore, the visualization of geographic data is the result of a spatial range query. This section describes the approximate range query method based on binary trees, which are built by the aforementioned methods. The definition of a window query is illustrated in section 2.4.1. We then discuss the error matching method between original features and simplified features. Finally, we present the approximate spatial query algorithm bounded error and bounded vertex count in the face of geospatial big data.

2.4.1 Window Query Definition

Window queries refer to the space objects within the scope of a given range. The geospatial data of real-time visualization are the result of the window querying. The approximate window query refers to the approximate space object that will intersect with a given query scope, *i.e.*, a bounding box. The definition of an approximate window query is described as follows.

Given a bounding box $W = \{x_{min}, y_{min}, x_{\max}, y_{\max}\}$ and dataset $D = \{L_0, L_1, \cdots, L_n\}$, $x_{\min}, y_{\min}$ represents the minimum coordinates of x and y, respectively; $x_{\max}, y_{\max}$ represents the maximum coordinates of x and y, respectively. Vertices within this range may be selected and shown on the screen. The basic idea of approximate window query processing is as follows: (1) generate the binary tree of a polyline by the vertex sampling method; and (2) execute the breadth-first traversal and take out the vertices in a specified window according to the descending order of the error. Then, the result dataset $Q_W(W, D) = \{L_i | L_i \cap W \neq \emptyset\}$ is generated. If part of the vertices of polyline $L_i \in Q_W(W, D)$ are located outside of the window, we will only keep the starting point or ending point to replace the original polyline L_i.

2.4.2 Approximate Window Query with Bounded Errors

In this section, we propose a query method combined window query with error constraint. The error-bound approximate query method is summarized as follows. Given an error threshold ε and a bounding box $W = \{x_{min}, y_{min}, x_{\max}, y_{\max}\}$, the query result is $Q_A(W, \varepsilon, L_0)$, $L_0 = \{P_0, P_1, \cdots, P_n\}$.

1. Set up a priority queue PQ and a sampling set S_P;
2. Add node $P_i \in L_0$ to the priority queue if the subtree of T_{L_0} is located in W, where T_{L_0} is the binary tree of L_0, and P_i is the root node of T_{L_0};
3. If $PQ \neq \emptyset$, select the node P_k with the maximum error, add node P_k to S_P if the error of P_k is greater than ε and then perform step 2 on P_k, otherwise, perform step 5;
4. If $PQ = \emptyset$, perform step 5;
5. Add all father nodes of P_{s_p} in S_P;
6. Arrange all vertices in S_P according to their subscript number, generate and return a new polyline dynamically.

In step 5, we add all father nodes of P_{s_p} in S_P. This is because if we only select the top k nodes, the new polyline generated will not continue. S_P combines with all the father nodes in the binary tree to compose a complete subtree. All the nodes in this new subtree are also a result of the B-DP algorithm in a certain threshold. We can also conclude that the error of visualization is still less than ε. The reason is that (1) the errors of the vertices that are selected in step 3 are all greater than ε, (2) if the error of the father node is greater than ε, it will already be selected in S_P, and (3) if the error of father node is smaller than ε, it also satisfies the assumption.

2.4.3 Approximate Window Query with Bounded Vertex Numbers

In this section, we propose a query method combined window query with vertex count. This approximate query method is summarized as follows. Given a number threshold δ and a bounding box $W = \{x_{min}, y_{min}, x_{\max}, y_{\max}\}$, the query result is $Q_A(W, \delta, L_0)$, $L_0 = \{P_0, P_1, \cdots, P_n\}$.

1. Set up a priority queue PQ and a sampling setS_P;
2. Add node $P_i \in L_0$ to the priority queue if the subtree of T_{L_0} is located in W, where T_{L_0} is the binary tree of L_0, and P_i is the root node of T_{L_0};
3. If $PQ \neq \emptyset$, select node P_k with maximum error, add node P_k to S_P if the number of S_P is smaller than δ and then perform step 2 on P_k, otherwise, perform step 5;
4. If $PQ = \emptyset$, perform step 5;

5. Arrange all vertices in S_P according to their subscript number, and dynamically generate new polyline;
6. Return new polylines and the size of S_P.

3 Experimental Study

3.1 Datasets

The dataset of the experiment is derived from the entire library file from Planetosm of OpenStreetMap (OSM). OSM is a global geographical feature dataset, and its collection, editing, analysis and application functions have formed a complete system based on the Internet [35]. We extracted the total factor data of the global coastline through the OSMCoastline program. This dataset has the most accurate coastline data, and its scale is at least an order of magnitude higher than that of the Global Self-consistent, Hierarchical, High-resolution Geography Database (GSHHG) (http://www.soest.hawaii.edu/pwessel/gshhg/) and Natural Earth Dataset [36]. The largest feature contains more than 4 million vertices, and the number of features with more than 100,000 vertices is over 270. The table space size of the relational database is 27 GB. The amount of data at this scale will have a serious impact on the performance in terms of the query, transmission and mapping. Table 1 shows the volume and characteristics of the data.

Table 1: Data description of global coastline

Data item	Number
vertices	43,591,835
polyline	878,453
complex feature	15,175
the maximum polyline number of complex feature	52,470
the number of enclosed polygon	572,926

The experiment server is built on Redhat 6.5 with an Intel Xeon E7-8870 CPU, with 128 G of memory and a 1,000 M network card. The development environment for the experiment is Eclipse 3.7 and the Java version is jdk1.7.65.

3.2 Efficiency of Vertex Sampling

To better examine the advantages and disadvantages of the proposed method, we design a contrast test and use Visvalingam-Whyatt (VW) [37] as our reference. The VW algorithm is a classic simplified method based on curve graph analysis. The volume of the polyline is reduced greatly by taking advantage of the area threshold and deleting vertices with the smallest area circularly. There are not thresholds in computing, so the data volume of the two methods are equal to the original data. Our method can exactly guarantee the balance of the binary tree.

There are three elements in OSM data: Nodes, Ways and Relations [35]. A Node defines the location of a point. Ways define the open polylines, closed polylines and areas. A Relation defines the relationships between elements, which may consist of a series of nodes, ways, or other relations. A Relation may represent a complex feature. Therefore, we should perform tests from two items: (1) the calculation of vertex error and generation of the binary tree for a polyline, *i.e.* way; (2) the calculation of the vertex error and generation of the binary tree for a complex feature, *i.e.* relation. We set two group experiments for the polyline and complex feature. For each group, we extract ten sets of data with different numbers of objects from the global coastline.

Figure 5(a) and (b) shows the preprocessing times of vertex sampling for polylines and complex features, respectively. As we can see, as the number of input objects increases, both approaches take more time. This is because the running time grows with respect to the vertices number for the B-DP and VW algorithms. The time complexity of the B-DP algorithm is $O(n \log n)$, and B-DP method consumes less machine time than VW algorithm. The progress time of the VW has a higher growth rate. This result means that our vertex sampling method saves execution time. Therefore, we build binary trees for all objects of the global coastline and perform spatial query experiments for interactive visualization in sections 3.3 and 3.4.

3.3 Effects of Error Bounds

In this section, how the proposed method performs under different errors and window sizes is discussed. The machine time and number of vertices are chosen as the reference index to evaluate the result. The machine time is a measure that sums up the query time and the transmission time to the client. This time reflects how many computation resources a query consumes.

For a certain dataset, more vertices will be shown on the screen as the scale of the map decreases. Given a certain error, if the scale becomes small, more vertices should be selected. We build an online simplified coastline over three different scales, including the World (–180°~180°,

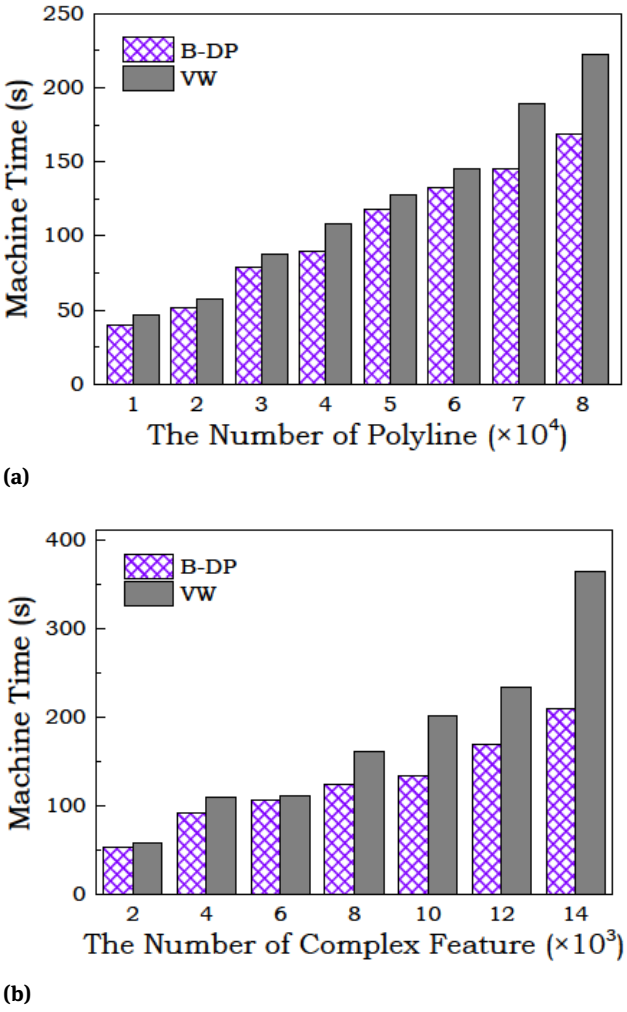


Figure 5: Efficiency of vertex sampling (a) machine time of polyline; (b) machine time of complex feature

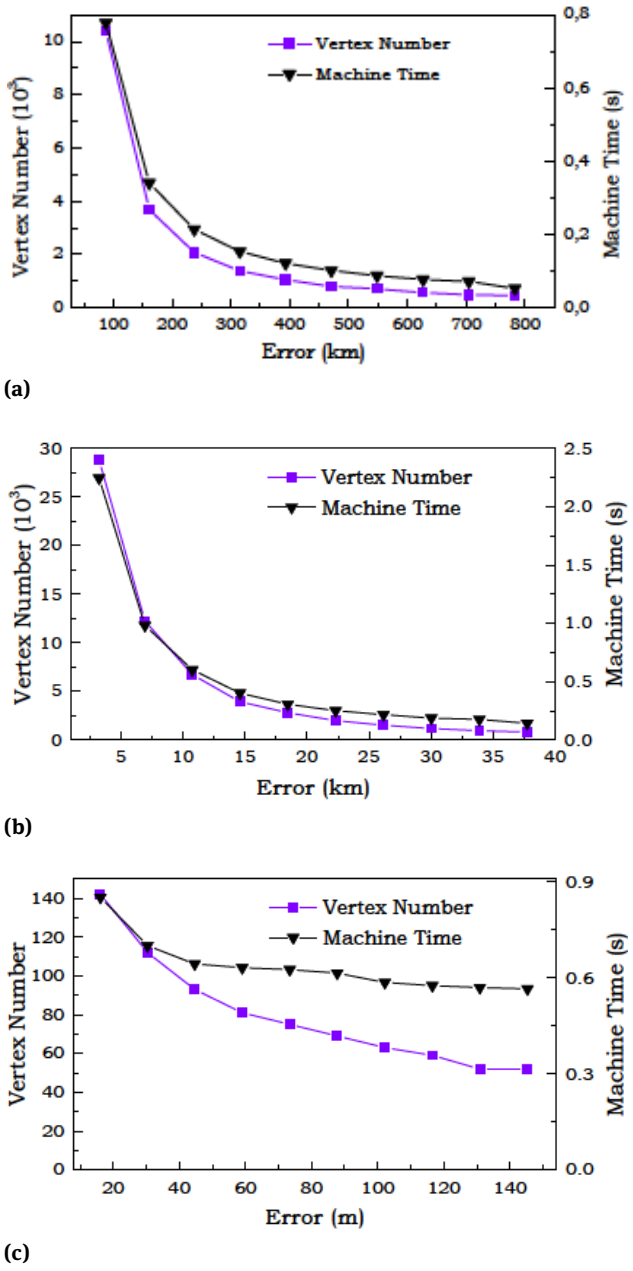
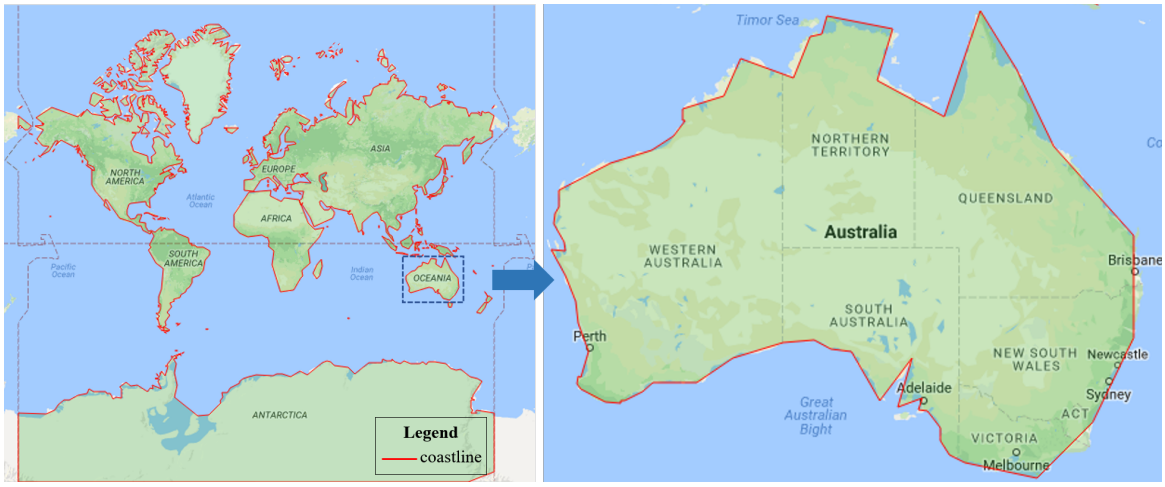


Figure 6: Effects of error bounds (a) machine time for the World; (b) machine time for North America; (c) machine time for a small island in the USA

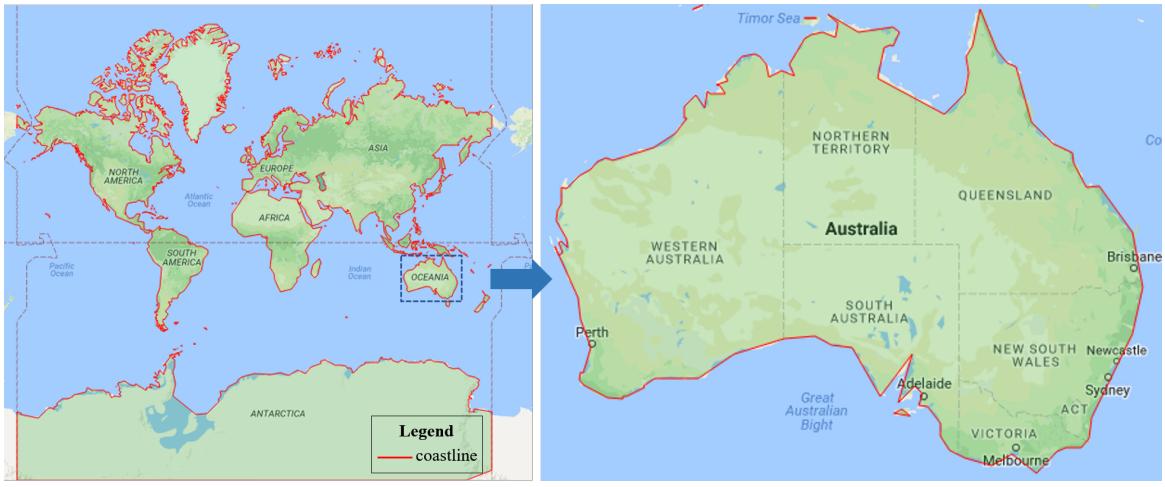
−90°~90°), North America (−120°~−58°, 20°~52°) and a small island in America (−72.01°~−71.84°, 41.03°~41.09°). The number of vertices in the World is 43,591,835; the number of vertices in North America is 4,191,417; the number of vertices on the island is 210.

In Figure 6, we use 10 different errors to report the machine time of query processing. The blue line represents the machine time, and the black line represents the vertex number of the query result. The performance results we show here are the average of a number of selected queries. For three different scopes, the machine times and the number of vertices returned drop with respect to the increase in the error bound, and the change in the machine time is the same as the vertex number. This result is due to the decrease in the total number in each scope. We can see that our methods perform well for different scopes of visualization. Therefore, our approximate spatial query method is able to produce a small number of samples in a short time with different error bound constraints.

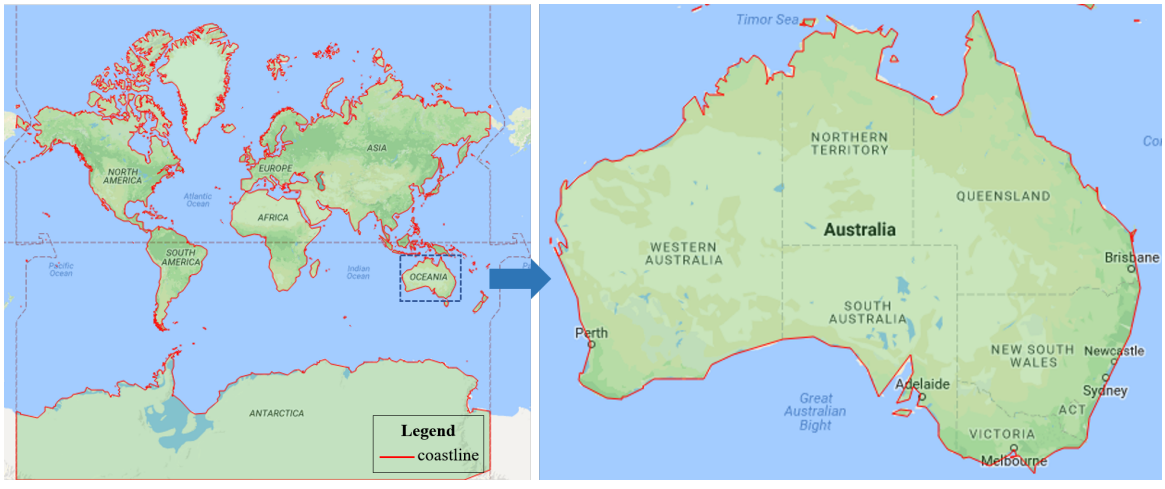
As we can see, the vertex number of the query results and the machine time tend to be stable as the error value increases. The explanation is that if the error is sufficiently large, less vertices will be selected. However, these selected vertices may not make a continuous line. Thus, all the father nodes remain in the trees of the selected ver-



(a)

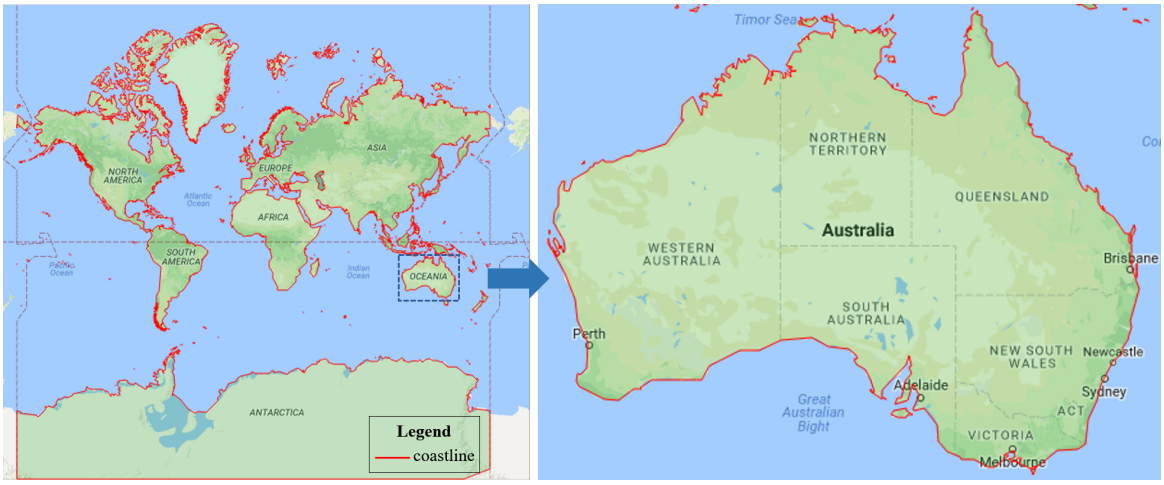


(b)



(c)

Figure 7: Query result for the World (a) 2000 vertices; (b) 5000 vertices; (c) 10000 vertices; (d) 50000 vertices



(d)

Figure 7: Query result for the World (a) 2000 vertices; (b) 5000 vertices; (c) 10000 vertices; (d) 50000 vertices

Table 2: Query efficiency for number bounds

ID	World		North America		Island	
	number	time (ms)	number	time (ms)	number	time (ms)
1	1,000	432	500	863	20	823
2	2,000	604	1,000	1,083	40	781
3	3,000	802	2,000	1,280	60	759
4	4,000	993	3,000	1,421	80	802
5	5,000	1,178	4,000	1,629	100	761
6	6,000	1,370	5,000	1,799	120	761
7	7,000	1,581	6,000	1,947	140	766
8	8,000	1,798	7,000	2,099	160	772
9	9,000	1,952	8,000	2,223	180	787
10	10,000	2,120	10,000	2,379	200	775

tices to maintain the continuity. That means that too large an error may play only a small part.

3.4 Effects of Number Bounds

The vertex number is also major factor of visualization efficiency. In this subsection, we evaluate the effectiveness of our method through limiting the vertex number, *i.e.*, the first k vertices are selected according to the order of error. The machine times are chosen as the reference index to evaluate the result. We build an online simplified coastline over three different scales, including the World, North America and a small island in the USA.

As shown in Table 2, as the input number increases, the machine times of the World and North America grow larger. For a certain number, such as 2,000 and 6,000, it will take more machine time in the World than that in North America. This difference is because these vertices constitute a small part of the total vertices for the World, although the same number of vertices are selected in two different scales. The machine times of the small island show that when the scale is very large, *i.e.*, the total number of whole vertices is very small, the machine time will be stable. That means that the benefit of the approximated spatial query with a bounded number is not so obvious at a large scale.

To clearly illustrate the results of the approximate window query with bounded number, we analyze the visualization effects from two different window sizes. As is shown in Figure 7 and Figure 8, the red lines represent the vector data of the coastline for the approximate queries and display on the client with Google map. The result shows that as the number of vertices increases, more vertices are selected, and the shape of the boundary will

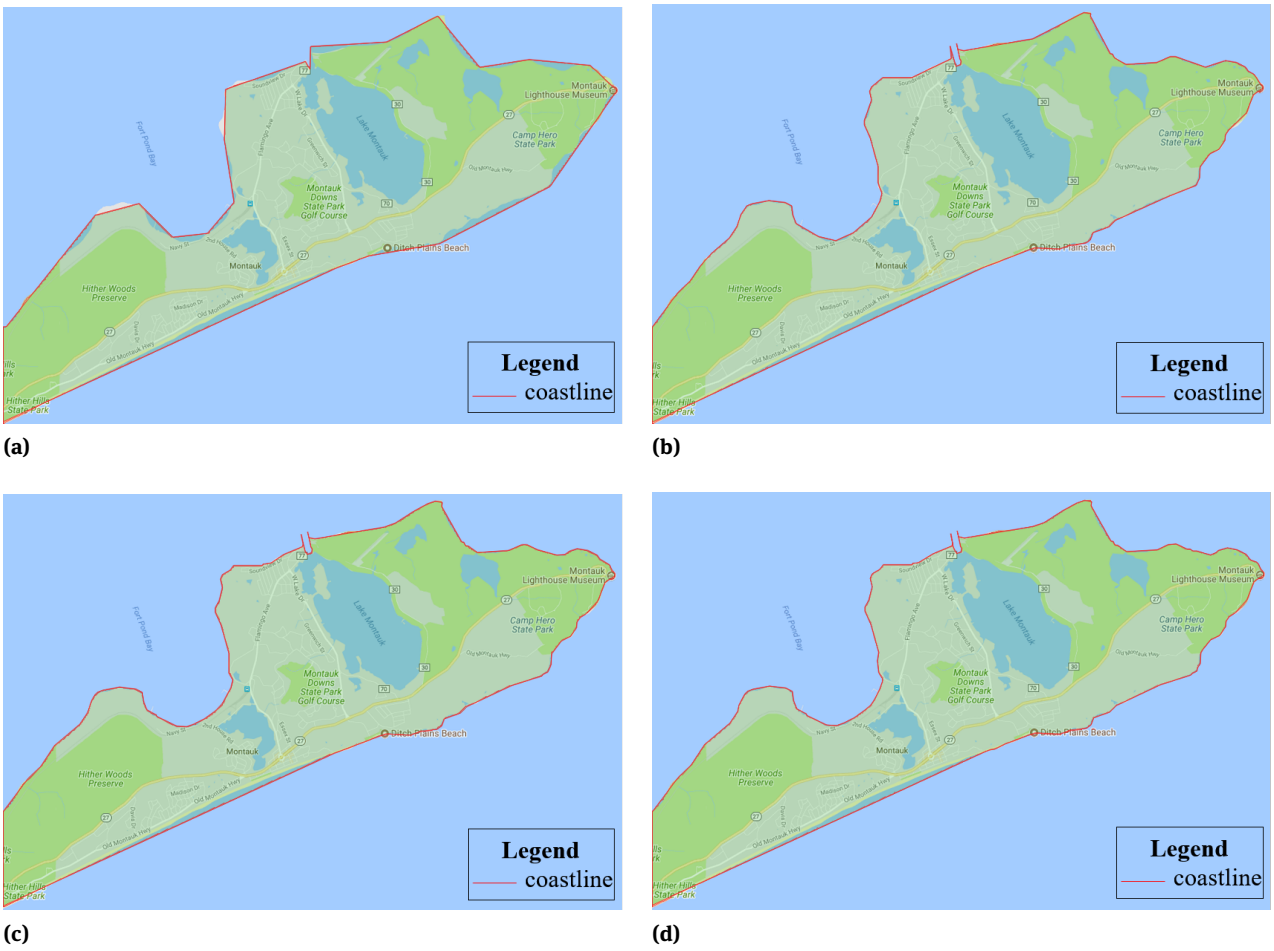


Figure 8: Query result for the small island in the USA (a) 20 vertices; (b) 80 vertices; (c) 120 vertices; (d) all vertices

be more consistent with the background map. As we can see, when the number of vertices reaches 5000, the shape of the boundary is basically consistent with that for the Google map. However, the result number is only 0.1 percent of the total number.

From the above analysis, the approximate spatial query with bounded number based on binary tree has very strong practicability and could significantly reduce the query time with a higher accuracy. With the contraction of the scale, our method can extract a small part of the original feature with high accuracy and creditability.

4 Discussion

The main focus of the current paper was to obtain a small portion of data and display it with an error or number bound. Data simplification has been studied from different perspectives, as follows: database [12], geographic information systems [30, 38], digital image analysis [39], and computational geometry [40]. According to the optimization goals, there are three constraints, as follows: (1) the spatial constraint, *i.e.*, selecting points within a limited space; (2) the error bound constraint, *i.e.*, selecting points while satisfying a pre-defined error bound; (3) the number bound constraint, *i.e.*, selecting points while satisfying a pre-defined number bound; and (4) the time bound constraint, *i.e.*, selecting points while satisfying a pre-defined time bound. The first to third items are the problems we discussed in this paper.

The spatial constraint represents the range of the visualization or screen, which is expressed by the latitude and longitude. This constraint is a basic condition in the visualization of geospatial data, which can be used together with error and the number constraint. Query window is defined as a spatial constraint in this paper. There are less vertices in the scope of the window with the scale increases, that means more points outside the window will

not be considered in query progressing. Therefore, spatial constraint narrows the scope of querying which is important for the improving of inquiry efficiency.

In our opinion, error is an important concept in approximate spatial query and visualization. Once the error is determined, only the vertices with larger errors than the threshold will be selected. It must be noted here that the visual effects are always different in different scales for a certain error value. For example, 100m is a large error for the visualization of the small island in the USA, however, it will take a very long query time for the World and the query result will be too large to display. Therefore, it is unrealistic for users to set different errors for different scales. Pixel is a suitable unit for error. An error less than one pixel produce the same visual effects for different scales and different latitudes. Three group experiments in section 3.3 use ten errors respectively. For each group, the pixel errors are 1 to 10 with the error increase. The results show that this method gives good error estimations and provides dynamic error bounds when the query window zooms in or out.

We also discussed the influence of the vertex number constraint. The purpose of the number constraint is to analyze the query efficiency and accuracy under different scales of the query window. As we all know, in the progress of interactive visualization, the errors of visualization may be different in different window sizes. The query result may work well with the original data at the large scale but may have a massive difference at the small scale. A number function or other dynamic scheme can be defined in future research.

In addition, there are other uncertain factors and deficiencies while using this method. The distribution characteristics of spatial data may have an impact on the data sampling and query [41]. In the case of highly concurrent requests, the geospatial database will encounter a much heavier burden than will the static data serving. A caching server can be used to relieve the stress of database server in those situations, and a study of the cache scheme and solutions is expected in the near future.

Overall, the error-bounded and number-bounded approximate spatial query method for interactive visualization in this paper not only solves the spatial-error-number bound constrained problem but is also effective for large spatial data. In addition, our method provides some inspiration for future studies about map rendering and spatial analysis.

5 Conclusion

In conclusion, this paper investigates a declarative approach to the spatial data sampling and query problem in visualization. By designing a B-DP algorithm, we produce the order and errors of the vertices, which can simplify the data in a short time and increase the efficiency and transmission of the query. One can extract spatial online vertices and use these ordered vertices to perform spatial visualization with constrains. To verify the effect of our method, we perform experiments on the OSM global coastline and took visibility and zooming consistency into consideration. The results illustrate that our methods are efficient and scalable.

Our work leads to a number of interesting and important future directions to explore. For large-scale road network, exact shortest path always requires much computation time. Approximate shortest path based on simplified polyline with some precisions can be accepted. In addition, the computational efficiency will be improved because vertex sampling method significantly reduces the size of the network. The study of approximate shortest path is expected in the near future. With the continuous updating of geospatial data, it is necessary to realize the dynamization of vertex binary tree. In view of this, updating algorithms on part of binary tree, including insert, delete and modify, are challenges for the future study.

Acknowledgement: We greatly appreciate the editors and two anonymous reviewers for their insightful and constructive comments on our work. This research was supported by National Key Research and Development Program of China (No. 2016YFC0803108), National Natural Science Foundation of China (No. 41701461), and the Basic Research Fund of CASM (No. 7771812).

References

[1] Agrawal R., Kadadi A., Dai X., Andres F., Challenges and opportunities with big data visualization. International Conference on Management of Computational and Collective Intelligence in Digital Ecosystems. ACM. 2015, 169-173.

[2] Li D., Towards geo-spatial information science in big data era. Acta Geodaetica et Cartographica Sinica, 2016, 45, 379-384.

[3] Liu J., Zhang F., Wang L., Dong C.; Wang Y.; Xu S., Research and prospect on spatial decision support service based on big data. Science of Surveying and Mapping, 2014, 39, 8-12+17.

[4] Wang L., Christensen R., Li F., Yi k., Spatial online sampling and aggregation. Proceeding of the VLDB Endowment, 2015, 9, 84–95.

[5] Hellerstein J.M., Haas P.J., Wang H.J., Online aggregation. ACM SIGMOD Record, 1997, 26, 171–182.
[6] Nguyen Q.V., Huang M.L., EncCon: An approach to constructing interactive visualization of Large Hierarchical Data. Information Visualization, 2005, 4, 1-21.
[7] Gibbons P.B., Matias Y., New sampling-based summary statistics for improving approximate query answers. ACM SIGMOD Record, 1998, 27, 331–342.
[8] Agarwal S., Mozafari B., Panda A., Milner H., Madden S., Stoica I., Blinkdb: queries with bounded errors and bounded response times on very large data. ACM European Conference on Computer Systems, 2013, 29–42.
[9] Chakrabarti K., Garofalakis M., Rastogi R., Shim K., Approximate query processing using wavelets. The VLDB Journal, 2001, 10, 199–223.
[10] Rosch P., Lehner W., Sample synopses for approximate answering of group-by queries. International Conference on Extending Database Technology. 2009, 403-414.
[11] Babcock B., Chaudhuri S., Das G., Dynamic sample selection for approximate query processing. ACM SIGMOD International Conference on Management of Data, 2003, 539-550.
[12] Chaudhuri S., Das G., Narasayy, R., Optimized stratified sampling for approximate query processing. ACM Transactions on Database Systems. 2007, 32, 9.
[13] Ghemawat S., Gobioff H., Leung S.T., The google file system. ACM Sigops Operating Systems Review, 2003, 37, 29–43.
[14] Aji A., Wang F., Vo H., Lee R., Liu Q., Zhang X. et al., Hadoop GIS: a high performance spatial data warehousing system over MapReduce. Proceedings VLDB Endowment, 2013, 6, 1009–1020.
[15] Eldawy A., Mokbel M.F., A demonstration of Spatial-Hadoop: an efficient MapReduce framework for spatial data. VLDB Endowment, 2013, 6, 1230–1233.
[16] Li Z., Hu F., Schnase J.L., et al., A spatiotemporal indexing approach for efficient processing of big array-based climate data with MapReduce. International Journal of Geographical Information Systems, 2017, 31, 17-35.
[17] Dean J., Ghemawat S., Mapreduce: simplified data processing on large clusters. Communications of the ACM, 2008, 51, 107–113.
[18] Alarabi L., Eldawy A., Alghamdi R., Mokbel M.F., TAREEG: a MapReduce-based Web Service for extracting spatial data from OpenStreetMap. ACM SIGMOD Record, 2014, 897-900.
[19] Malewicz G., Austern M.H., Bik A.J., Pregel: a system for large-scale graph processing. ACM SIGMOD International Conference on Management of Data, 2010, 135–146.
[20] Agrawal V., Subash S.R., Prakash P., Visualization of big data: its tools and challenges. International Journal of Applied Engineering Research, 2014, 9, 5277-5290.
[21] Sarma A.D., Lee H., Gonzalez H., Madhavan J., Halevy A., Efficient spatial sampling of large geographical tables. SIGMOD, 2012, 193-204.
[22] Cao H., Wolfson O., Trajcevski G., Spatio-temporal data reduction with deterministic error bounds. The VLDB Journal, 2006, 15, 211–228.
[23] Jugel U., Jerzak Z., Hackenbroich G., Markl V., VDDA: Automatic visualization-driven data aggregation in relational databases. The VLDB Journal, 2016, 25, 53-77.
[24] Liu Z., Jiang B., Heer J., imMens: Real-time visual querying of big data. Computer Graphics Forum, 2013, 32, 421–430.
[25] Wu E., Battle L., Madden S. R., The case for data visualization management systems: vision paper. VLDB, 2014, 7, 903-906.
[26] Kang X.. Graph-based synchronous collaborative mapping. Geocarto International, 2015, 30(1):28-47.
[27] Fisher D., Big data exploration requires collaboration between visualization and data infrastructures. In Proceedings of the Workshop on Human-In-the-Loop Data Analytics, 2016, 16.
[28] Laptev N., Zeng K., Zaniolo C., Early accurate results for advanced analytics on MapReduce. Proceedings of the VLDB Endowment, 2012, 5, 1028-1039.
[29] Ramer U., An iterative procedure for the polygonal approximation of plane curves. Computer Graphics & Image Processing, 1972, 1, 244-256.
[30] Douglas D.H., Peucker T.K., Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 1973, 10, 112–122.
[31] McMaster R.B., A statistical analysis of mathematical measures of linear simplification. The American Cartographer, 13(2):103-116.
[32] Jasinski M.J., The comparison of complexity measures for cartographic lines (90-1). Ncgia Technical Reports, 1990.
[33] Shahriari N., Tao V., Minimising positional errors in line simplification using adaptive tolerance values. 2002, 153-166.
[34] Huttenlocher D.P., Klanderman G., Rucklidge W.J., Comparing images using the Hausdorff Distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993, 15, 850-863.
[35] Haklay M., Weber P., OpenStreetMap: User-generated street maps. IEEE Pervasive Computing, 2008, 7, 12-18.
[36] Nathaniel V.K., Tom P., Introducing natural earth data – Naturalearthdata.com, Geographia Technica, Spatial Issue, 2010, 82-89.
[37] Visvalingam M, Whyatt J.D. Line Generalization by Repeated Elimination of Points. Computer Graphics Forum, 1993, 30(1), 46-51.
[38] Mcmaster R.B., Automated line generalization. Cartographica the International Journal for Geographic Information and Geovisualization, 1987, 24, 74-111.
[39] Hobby J.D., Polygonal approximations that minimize the number of inflections. Acm-Siam Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 1993, 93-102.
[40] Agarwal P.K., Varadarajan K.R., Efficient algorithms for approximating polygonal chains. Discrete and Computational Geometry, 2000, 23, 273-291.
[41] Yan Y., Chen L.J., Zhang Z., Error-bounded sampling for analytics on big sparse data. Proceedings of the VLDB Endowment, 2014, 7, 1508-1519.