Multi-scale Display of Point Data Sets at the Client Side

Tianbao Wang[1, 2*], Ershun Zhong[1], Hao Lu[1,2], Huaihai Zheng[1,2]
[1]Institute of Geographic Sciences and Natural Resources, CAS, Beijing, China
[2] Graduate University of Chinese Academy of Sciences, Beijing, China
*Corresponding author: wangtb@lreis.ac.cn

***Abstract*—Multi-scale representation of geographic features is one of the research focuses of Geographic Information System (GIS). Most of the work has been done at the server side, especially when dealing with massive lines and polygons data by using spatial database technology. This paper discusses the issue about multi-scale display of point data sets, and presents a solution which is implemented at the client side. This solution overcomes the problem of losing some points of interest and their attribute. It also promotes the efficiency of displaying large point dataset with limited pixels. First, we process the level of details at the client side to avoid communicating with the server side at every scale but only at the first process. This strategy greatly decreases the time consumption in querying the server and network transmission. Second, when doing multi-scale manipulations at the server side (including spatial database), some points at certain scales will lose; some points more or less at certain scales will be disposed. However, by processing at the client side, because it contains all of the point data sets, this problem is solved. At last, this paper designs an adaptive algorithm to resolve the contradiction between small screen area and large point data sets with the tedious overlap phenomenon in displaying. A case study verifies the optimized display effect and improved efficiency of the proposed approach.**

Keywords- multi-scale; visualization; client side; point data sets

I. INTRODUCTION

Multi-scale representation plays an important role in progressive transmission of vector data over the web, self-adaptable visualization of spatial information, navigation in spatial cognition, scale match during the inter-operation and other applications. It the past, much more attention has been paid to the sever side, especially to spatial database technology. A multi-scale database (MSDB) is a representation-oriented level of detail (LOD) approach to solve a very complex process of deriving a generalized map from a detailed database [1-2]. This can be described as a combination of different datasets representing objects at several scales, which is known as a similar item Multi-representation database (MRDB). There are some kinds of technologies on building MSDBs proposed, such as Multi-scale snapshots, multi-scale spatial index, and multi-scale vector data structure. Other solutions usually involve ideas from more than one approach [3-6].

However, no matter which technique above is adopted, it is just a solution at the server side. Transmission occurs at every scale when the client retrieves the vector data, which costs plenty of time, especially in the WebGIS applications. In addition, the solutions at the server side, including spatial database, only focus on line and polygon vector data [7-11]. If point data sets are manipulated at the database at some certain levels, it usually loses some points of interest and their attributes, causing wrong results.

In order to save transmission time over the web and reduce communication times between the client side and the server side, the authors do some research on the multi-scale display of point data sets at the WebGIS client side. It overcomes the problem of missing information as well. The remainder of this paper is organized as follows. Section 2 introduces the process procedure and the core algorithm of the solution. Section 3 takes experimental point data sets to illustrate the result, and then discusses its benefits. The final section draws conclusions and discusses future work.

II. APPROACHES

The basic approach of the above solution is as follows: First, we query and require the whole point data sets from the server side, and then the points will be transferred to the client side. At the client side, with applications' view changing (zooming in or zooming out), the points will be calculated and located on the appropriate position for better display with diverse style.

A. Querying and Transmission Time

Querying and transmission time if often determined by computer configuration (e.g. I/O, CPU) and network speed. Another important factor is the data volume; more data will cost more querying and transmission time, approximately linear positive correlation between them.

A tacit advantage of processing the level of details at the client side is that data retrieve only occurs at the first time, so it doesn't communicate with the server side at every scale. In order to estimate the transmission time of these two ways, a common expedient is to set the scale internal just two times between the two neighboring levels. It is assumed that the point volume reduces to half with the scale increasing; therefore querying and transmission time is also reduces to half.

If set the querying and transmission time cost is t at the first level which consists of all the points, the time cost of the second level is nearly t/2, and the third level is t/4.The rest can be calculated at the same manner. Hence, the cost time sum of all the levels is shown in Eq. (1):

$$T = t + t/2 + t/4 + t/8 + \ldots + t/2^{n-1} \approx 2*t \quad (1)$$

Obviously, if we just got the whole points at the first time and process those at the client side, the cost time sum would be just t, almost half of T.

B. Core Algorithm of Optimizing Display Effect

Now that we acquire all the points at one time, we need to display all of them at any scale level. But the user's screen size is common limited; therefore some points will get overlaid on top of others, which are nearly in the same pixel [8]. In addition, it will consume a lot of time for loading and displaying (also known as rendering) so many points at the same time.

This paper is an attempt to present an algorithm to resolve the conflict of displaying lots of points at limited screen size with different scales or levels. The core concept is that at certain zoom level, according to the size of display screen and the input points, we use just one point with different size and symbol to substitute the overlying ones. Then with the map view zooming in or out, after recalculating, disperse the points which are not overlapped with each other and merger those new overlapping ones.

It's necessary to explain the input variables in order to explain the algorithm called "Marker Clustering".

- Scale: the scale at the current display level of map.
- Marker List: the points/ geographic features which are acquired from the server side and will be displayed on the screen size. Every marker, styled with a default symbol, has its own geographic coordinate represented by x and y.
- Radius: a value in the screen pixel coordinate system. It defines the threshold for being merged when the distance is smaller than it between two markers. Its value depends on the size of the marker's default symbol.
- Diameter: the corresponding threshold value in the geographic coordinate system. According to Scale and Radius, it can be computed.

The detailed algorithm is described as follows:

Step 1): Divide View to Grids and Initial Merging

Based on the value of input variables, at a certain scale, calculate the bounding box of all the markers using their coordinates, then according to Diameter, divide the display area to m*n equal grids and every marker is in one given grid.

Next, traverse the Marker List; give every marker a Key, indicating its grid column and row. If the Key is a new one, create a new MarkerCluster. Otherwise, it means that the current marker is located at the same grid with other markers, so merge it into the existing MarkerCluster.

- MarkerCluster: A MarkerCluster is a particular marker with a larger size and symbol different from the ordinary marker. The size of its symbol depends on the count of its children, which are the ordinary markers located in the same grid by Key. And the coordinate of a MarkerCluster is average of its children's positions, shown in Eq.(2):

$$X = \sum_{i=1}^{n} x_i / n \qquad Y = \sum_{i=1}^{n} y_i / n \tag{2}$$

At the end of this step, an initial MarkerCluster List has been got. This step is the most important part of the whole algorithm, and its flow is illustrated in Figure 1.

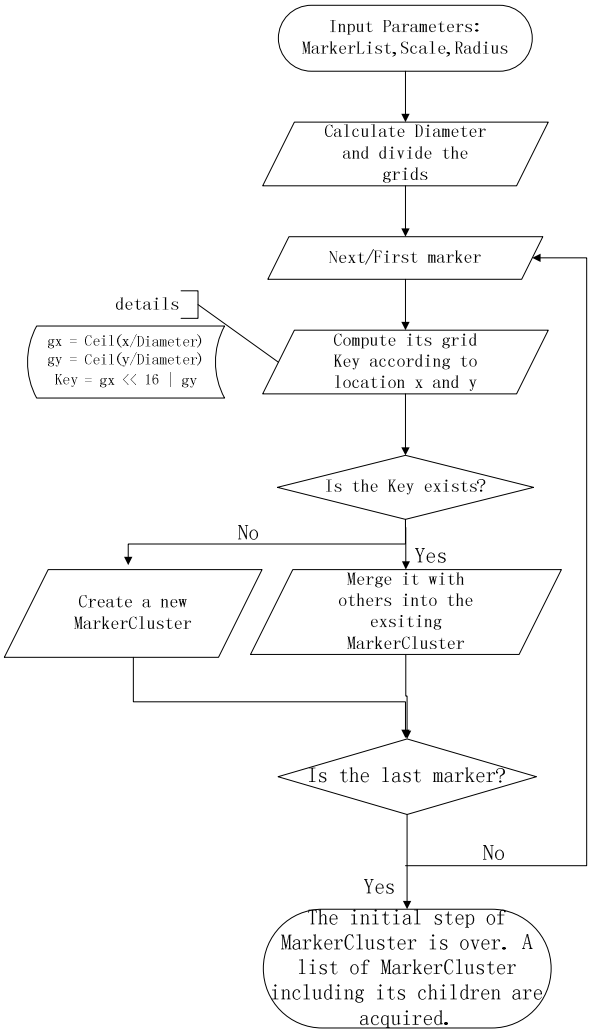

Fig.1. Flowchart of the core part of MarkerCluster Algorithm

Step 2): Search and Merge the Initial MarkerCluster List

Aiming to the result of step 1, for every MarkerCluster, search the immediate area around it, which are the surrounding eight MarkerClusters in different directions expressed in Figure 2. If the distance is less than threshold, merge the neighboring MarkerCluster and recalculate the new children number and the new location arcading the rules in the step above.

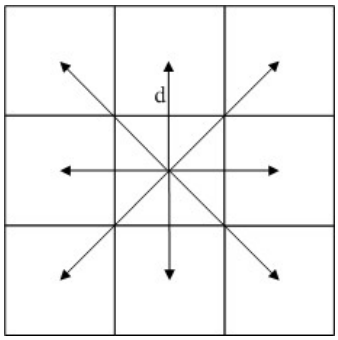

Fig.2. Search the surrounding eight directions to merge

After that we get the final MarkerCluster List with its new position and number of children.

Step 3): Style the Final MarkerCluster List and Marker List

Ultimately, the final MarkerCluster List is obtained. However, maybe some Markers do not belong to any MarkerCluster; they are alone but remained, so its number of children is one.

Then according to the number of the MarkerCluster's children, style it with symbol using matching size. Even style them with different color in order to distinct them easily. Those alone Markers are styled with the default symbol.

C. Render Time

The last phrase of the client approach, after querying, transmission and calculating, is that draw the Marker or MarkerCluster one by one on the display screen. Because a MarkerCluster contains varying amount Markers, decreasing the total number of Markers, render time is reduced naturally.

III. CASE STUDY

A. Experimental Environment

In order to evaluate the method proposed in this paper, we conducted an experimental WebGIS application. The server side and the client side are both implemented in a same Pentium Dual-Core PC with 1.67G CPU and 1.5G RAM. The map view or screen size is adjustable.

The "Marker Clustering" algorithm is designed to apply to point data sets; hence we just randomly generate thousands of points to justify our approach.

B. Experimental Result

The client side acquires all the 4,000 points from the server side, and then draws them on the screen using the "Marker Clustering" algorithm or not.

Figure 3(a) shows the screenshot of drawing all points without using special algorithm. Every marker is styled with the default symbol, same size and same color. There are some markers overlapping with each other so as to not distinguish one from others easily. When zoom in/out the map, phenomenon is still not changed.

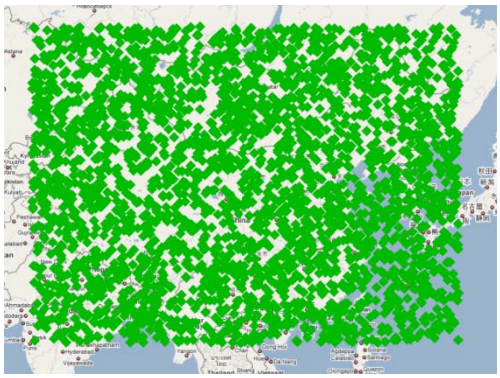

Fig.3.(a) Screenshot at level 5 not using Marker Clustering

Figure 3(b) is the result screenshot of drawing the same points and at the same scale, however, using "Marker Clustering" algorithm. It cleans up the tedious overlap phenomenon, even though the screen area is restricted. And with different quantity of MarkerCluster's children count, it's styled with appropriate size and color.

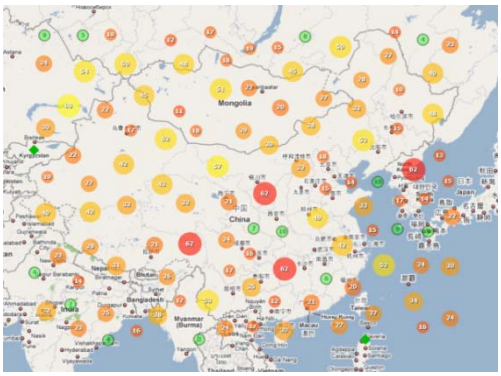

Fig.3.(b) Screenshot at level 5 using Marker Clustering

Figure 3(c) shows the screenshot at the level 4 using "Marker Clustering" algorithm. That is the map view above is zoomed out. When the map view goes change, the "Marker Clustering" algorithm will be executed one time and carry out the new MarkerCluster with the new diverse symbol.

Fig.3.(c) Screenshot at level 4 using Marker Clustering

Figure 3(d) shows the screenshot at the level 8 using "Marker Clustering" algorithm. That is the map view above is

zoomed in. At this level, MarkerCluster and ordinary Marker exist meanwhile. And ordinary Marker is styled with default symbol, MarkerCluster with different symbol.

Fig.3.(d) Screenshot at level 8 using Marker Clustering

C. *Response Time*

The total response time of throughout procedure consists of querying and transmission time, calculating time and rendering time. Querying and transmission time has been discussed.

Calculating time, namely the average-case time complexity of "Marker Clustering" algorithm, is estimated by O(N).

Rendering time can be reduced to one eight of common render manner without using "Marker Clustering". We test the number of points from 2,000 to 20,000, and make the comparison between the two approaches. But because the absolute time consumption is related with the points' location, the current scale and other factors, we don't list the test results in detail. The comparison between the relative time costs makes more sense.

IV. CONCLUDING REMARKS

The study presents an approach at the client side to multi-scale display point data sets. According to the approach, most of work, which usually has been done in the server side, is transferred to the client side. Besides making use of the capability of the client side, it reduces communications between them largely.

Not only that, "Marker Clustering" algorithm is introduced to resolve the contradiction between small screen area and large point data sets with the tedious overlap phenomenon in displaying. The algorithm is not complicated but effective, and it's suitable for almost all the discrete point features data set. In addition, it provides a good representation with diverse symbol.

This approach not only focuses on the display effect but also it promotes efficiency of rendering geographic features. Because the algorithm reduces the count of points which are need to be drawing on the screen, but not losing information.

Experimental applications are implemented and prove that the approach at the client side for point data sets is effective in the every phrase in multi-scale display, and works well for display effect.

Future research on improving the approach for multi-scale displaying of point data sets at the client side could include the following topics:

- Uncertainty analysis. The location of MarkerCluster is just average value of its children now. The attributes of them is not considered. And when map is panning, not zooming, some MarkerCluster will move to new location, which is uncertain.
- Performance optimization. Especially the step of "search and merge" could be designed to a more efficient algorithm.

REFERENCES

[1] T. Kilpelainen, "Multiple Representation and Generalization of Geo-database for Topographic Maps," Ph. D., Finish Geodetic Institute, Helsinki University of Technology, 1997.

[2] Y. H. Wang, et al., "Multi-Scale Conceptual Model for GIS Geographical Features," Jorunal of China University of Mining & Technology, vol. 32, pp. 376-381, 2003.

[3] C. X. Cheng, et al., "Extensions of GAP-tree and its implementation based on a non-topological data model," International Journal of Geographical Information Science, vol. 22, pp. 657-673, 2008.

[4] C. X. Cheng, et al., "A quantitative scale-setting approach for building multi-scale spatial databases," Computers & Geosciences, vol. 35, pp. 2204-2209, 2009.

[5] E. P. F. Chan and K. K. W. Chow, "On multi-scale display of geometric obects," Data & Knowledge Engineering, vol. 40, pp. 91-119, 2002.

[6] P. Van Oosterom and V. Schenkelaars, "The development of an interactive multi-scale GIS," International Journal of Geographical Information Systems, vol. 9, pp. 489-507, 1995.

[7] H. Wang, et al., "Adaptive Strategy on the Visualization of Electronic Map," Geomatics and Inforamtion Science of Wuhan University, vol. 29, pp. 525-528, 2004.

[8] T. H. AI, "Key technologies and strategeis in the development of multi-scale spatial database," Science & Technology Review, vol. 12, pp. 4-8, 2004.

[9] N. F.Q., et al., "Key Techniques in Multi-scale Display of Vector Data," Geomatics and Inforamtion Science of Wuhan University, vol. 34, pp. 869-872, 2009.

[10] Z. Li and S. Openshaw, "Algorithms for automated line generalization based on a antural principle of objective generalization," International Journal of Geographical Information Systems, vol. 6, pp. 373-389, 1992.

[11] B. Yang, "A Multi-resolution model of vector map data for rapid transmission over the internet," Computer & Geosicences, vol. 31, pp. 569-578, 2005.

[12] T. B. Wang, et al., "WebGIS client based Silverlight," Journal of Geo-information Science, vol. 12, pp. 69-75, 2010.