Visualization System of Massive 2D Seismic Data

Wenqin Li
College of Mathematics & Computer Science, Yangtze Normal University, Fuling, Chongqing, China
Email: lwq912@126.com

Daqing Wang
College of Mathematics & Computer Science, Yangtze Normal University, Fuling, Chongqing, China
Email: zhangxiong1262007@gmail.com

***Abstract*—With the fast development of Visualization Technology, massive data need to be converted into intuitive graphics in more and more areas, such as in oil and geological exploration fields. In order to achieve highly efficient visualization of 2D seismic data, the following important methods are employed to improve the speed of rendering. The first method is to process massive data in point coordinate (x, y) forms. The second method is to draw local area of graphics by figuring out what points need to be painted for that graphics of massive data that sometimes can't be shown completely on a displayed screen, which results in a high rate of drawing and less usage of memory. The third method is to pick out a sparsing algorithm by K-Neighboring in order to see the whole picture of massive 2D data or to zoom out the whole picture to guarantee the quality of pictures. And sometimes processors need to view detailed graphics of a region of 2D seimic data, then it proposed bilinear interpolation algorithm. Overall, visualization system of massive 2D seismic data presented in this paper is based on the methods proposed above and uses Qt as a development language. Finally, high efficiency is achieved by drawing local area data and bitmap-cache mechanism when massive 2D data need to be displayed on more screens; and also high speed is obtained to render by operating 2D graphics such as by zooming out through K-Neighboring sparsing algorithm.**

***Index Terms*—Massive 2D seismic data, local area data, sparsing algorithm, bilinear interpolation algorithm, Qt**

I. Introduction

Displaying graphics of images are handled by visualization technology of 2D graphics structure data; and massive data are obtained in seismic exploration. Generally, we need to process hundreds of MB or GB, with hundreds of thousands, even millions of 2D data. In the 1960s [1], Visualization Technology was proposed and applied deeply in various research areas subsequently. It converts data into graphics or images displayed on the screen and ultimately provides interactive procession, through interpreting data and employing computer graphics and image processing technology, and helps us to dig into the correlation and regulation between data. With the development of science and technology, the trend from two-dimensional visualization to three-dimensional visualization is inevitable, but the application of two-dimensional visualization still has some irreplaceable advantages. For example, in geological mining, oil exploration, terrain simulation and other areas, two-dimensional visualization technology could be helpful in processing quality control of seismic data [2]. Therefore, based on seismic data which use different storage formats, we designed and implemented two-dimensional visualization and related operations, and employed huge seismic attributes data [3].

This paper adopt the Red hat Enterprise Linux Server Release 5 as the development platform, and use Qt as the development language in eclipse-based integrated development environment. Through the actual software development, we verify that the two-dimensional seismic data visualization can give users an intuitive, readable explanation and analysis of two-dimensional seismic data. Its design and realization possesses practical use and promotion value, and it has an advantage of better portability and scalability.

II. Structure Design of 2D Visualization

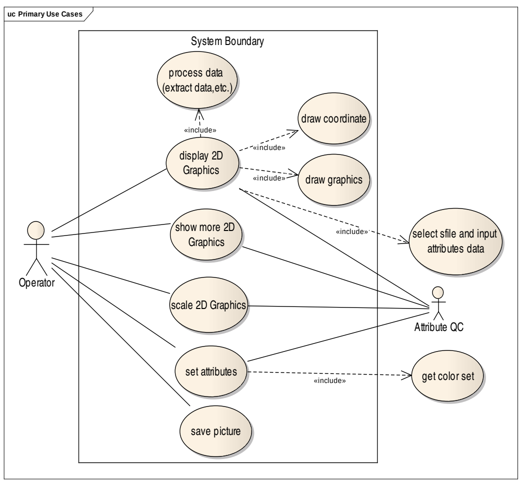


Figure 1. Use case diagram of 2D visualization system

Manuscript received April 28, 2011; revised June 18, 2011; accepted June 28, 2011.
Project number : KJ091306. Corresponding author: Wenqin Li

doi:10.4304/jsw.7.7.1625-1632

This paper partitions the visualization system of massive 2D seismic data into the following modules: extracting data, drawing 2D graphics, showing more 2D graphics in a screen, scaling 2D graphics, and scratching pictures [4]. Fig. 1 shows the use case diagram. The two-dimensional visualization system of seismic data is based on the use case diagram for robustness analysis diagram, timing diagram of the design, and the design class diagram.

And this thesis is elaborated from the following two perspectives of massive 2D visualization: (1) By drawing local 2D area to enhance the rate of drawing and scrolling graphics of massive 2D seismic data; (2) Using K-Neighboring sparsing algorithm to overview or zoom-out pictures of massive 2D seismic data.

III. HIGH-EFFICIENCY OF DRAWING MASSIVE 2D DATA

In the visualization process of massive 2D seismic data, sometimes, we need to collect accurate data to display graphics and to show as more details of 2D data as possible, which can be operated by zooming in graphics. Meanwhile, large volume of 2D data cannot be shown completely on just one screen but on more than one screen by moving the X or Y scrollbar. For the vast amounts of data, the whole 2D seismic data may not be transferred into memory, which can avoid a high rate of memory usage or memory shortage owing to an excessive amount of data into memory in one time. In the following, we try to introduce the method of drawing local area 2D data in current screen and use a bitmap caching mode to achieve fast rendering of 2D visualization.

A. *Drawing Local Area Graphic of Massive 2D Data*

Through drawing local area data in current screen as we say in this paper as split-screen displaying of graphics, the key point is to figure out local data of current screen that need to be drawn in data files and need changing coordinate of current screen [5]. By dragging the scrollbar we can see the continuality of the whole 2D visualization. Therefore, we need a deep understanding of Qt scrollbar.

Fig. 2 shows the schematic of drawing local area graphics of massive 2D data. As we can see that, firstly, we get paint or repaint event, and then use the following method with QScrollbar which will be introduced below to figure out the real 2D seismic data of current screen; secondly, we seek out the position of data and offset of data in massive 2D data file; thirdly, we can use memory-map-mechanism to load needed data into memory quickly; and lastly, we use Qt to convert local data into graphics.

Scrollbars in Qt are achieved through the QScrollBar, which provides horizontal and vertical scrollbar. Fig. 3 shows the QScrollBar horizontal scrollbar chart.

The steps of figuring out local area 2D data in file are as following:

(1) Match scrollbar step value with real 2D data range

As we can see in Fig. 2, document length represents the real length of the whole 2D data shown in pixel, meaning the scrollbar's position, which range from minimum() to maximum(). Page step states the length of scrollbar, which also means the current screen length matched with scrollbar value. Thus, relationship of them is shown in formula 1.

$$document\ \mathrm{length} = \mathrm{maximum}() - \mathrm{minimum}() + pageStep() \quad (1)$$

(2)Compute current screen's maximum and minimum 2D data

Suppose the real range of 2D seismic data is minData to maxData, and a pixel denote real 2D seismic data range is span, width and height shows current screen width and height respectively; then we can use the following formula 2 and formula 3 to figure out the minimum of current real 2D seimic data and maximum of current real 2D seismic data.

$$\mathrm{Current\_minimum\_real\_data} = (\mathrm{min} + \frac{\mathrm{value}()}{\mathrm{span}}) \quad (2)$$

$$\mathrm{Current\_maximum\_real\_data} = (\mathrm{min} + \frac{(\mathrm{value}() + \mathrm{pageStep}())}{\mathrm{span}}) \quad (3)$$

And values of boundary are figured out by formula (4) as below.

$$\begin{cases} x' = Rect.left() + (xData - \min XData) \times \dfrac{Rect.width() - 1}{\max XData - \min XData} \\ y' = Rect.bottom() - (yData - \min YData) \times \dfrac{Rect.height() - 1}{\max YData - \min YData} \end{cases} \quad (4)$$

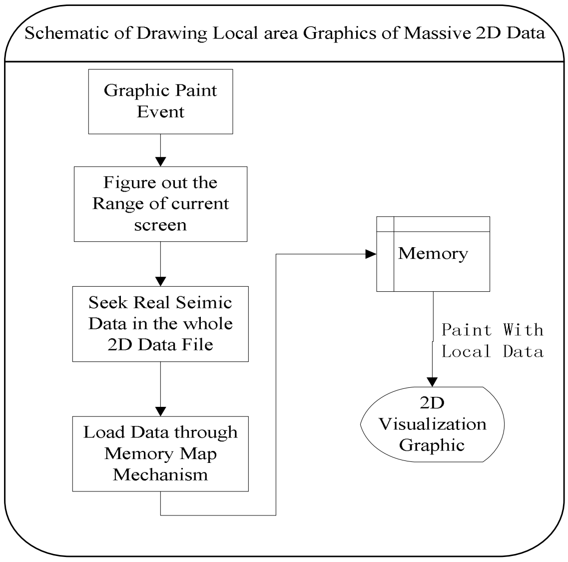


Figure 2. Schematic of local area graphics of massive 2D Data

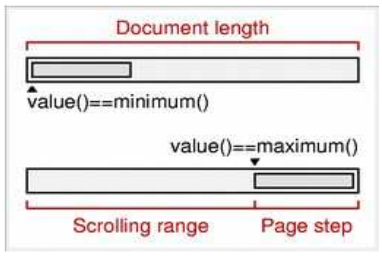


Figure 3. Structure of QScrollbar

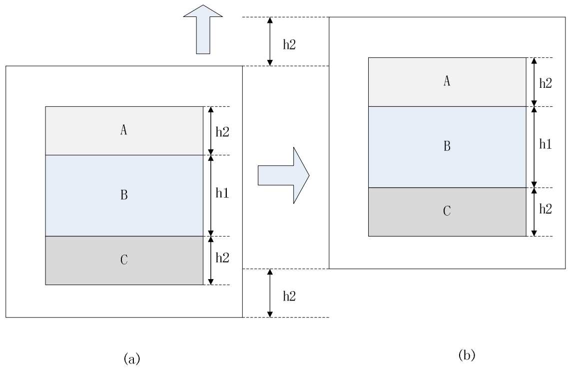


Figure 4. Figure out minimum and maximum real 2D data of current screen

By step (1) and step (2), we can figure out range of local data, and then read local area data from the whole 2D data file, which can highly improve the rate of drawing graphics, as shown in Fig. 4.

B. Use Bitmap Cache Mode to Improve Rate of Rendering

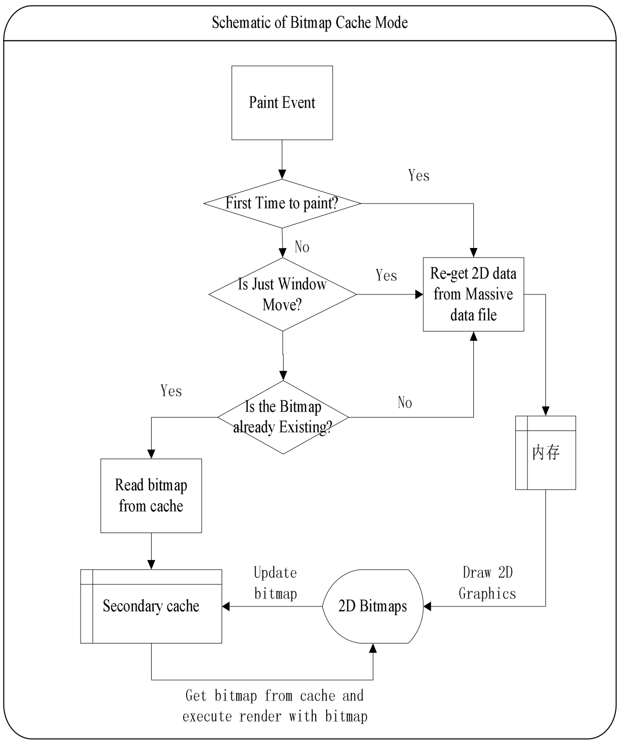


Figure 5. Schematic of bitmap cache mode

Often visualization of 2D data involves zoom-in and zoom-out operations. For the above operation, the most important is the operational efficiency. As we know, any operations which cause repainting of GUI in Qt will bring users slow-operation experience, which would be resource consumption in machinery. The main reason for the problem above is that every redrawing causes the visualization system to re-read 2D graphics data. In fact, users always change only some parts of the current screen or just change the position of the displaying window. In this case, there is no need to re-read the whole data and to re-draw the whole picture. As a result, we propose a mechanism by using bitmap cache mode.

As we can see from Fig. 5 above, the schematic of Bitmap Cache Mode is elaborated. Firstly, we receive paint event or repaint event from lower layer, by judging if the paint event is carried out for the first time. If yes, we should re-get 2D data from massive data file and redo the drawing procession of 2D graphics; otherwise, we should just fetch bitmap from secondary cache under the premise that the bitmap of the current screen needing to update has already existed in cache.

According to the description above, in order to improve the efficiency of this case, the operation of redrawing will be done through fetching out a bitmap which is stored in the secondary cache; there is no need to retrieve data from disk by IO and map data but directly access to the secondary cache. This greatly improves the operation efficiency.

IV. K-NEIGHBORING SPARSING ALGORITHM

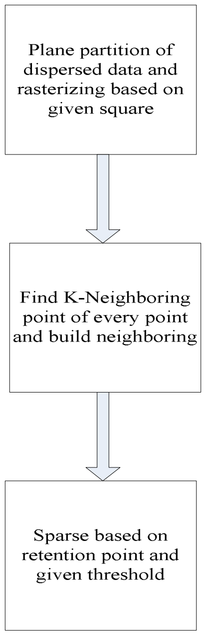


Figure 6. The program of sparsing for dispersed two-dimension data

The massive data are obtained in seismic exploration. Commonly, hundreds of MB or GB two-dimension earthquake data would be obtained, where hundreds thousand, even millions of data points had to be dealt with. For three-dimension earthquake data, they are huger. The efficiency for the direct treatment of massive data is very low when the visualization model is built. When a figure is plotted, coordinate mapping and even floating-point operation has to be dealt with, which would affect displaying speed of figure. So we hope to sparse massive data in some conditions. At present, the common sparsing algorithm is Douglas-Poiker algorithm [6] and vertical dimension constraints the algorithm [7]. The vertical dimension between observation point and the beginning and ending points is whether it is more than the given threshold in Douglas-Poiker algorithm viewed from the overall angle. Its excessive recursive number, however, enhances the difficulty of computer and algorithm deal with usually curved data. The vertical dimension between observation point and the neighboring line segment is whether it is more than the given threshold in vertical dimension constraints algorithm viewed from the local angle. The speed of computer and program are more conveniently controlled, but the computational complexity for two-dimension data is huge. To enhance the speed of sparing, a new Sparsing algorithm based on K-Neighboring has been given in this paper. The program has been shown in Fig. 6.

A. Use plane partition methods to divide massive 2D data

Massive 2D data are seen as a rectangle, and they could be compartmentalized u*v little squares and ordering these squares. The cell length of little square can be obtained from the following formula

$$\mathrm{cell\_len} = \beta \bullet \sqrt{\frac{k}{n}(\max X - \min X)(\max Y - \min Y)} \quad (5)$$

where maxX and minX are maximum and minimum of X rectangle, maxY and minY are maximum and minimum of Y rectangle separately, β is scale factor of adjusting length of rectangle, the optimal value is 0.8～1.2 [8], n is all two-dimension data point, k is the number of neighboring point. The number of gridding in the X and Y direction is

$$\begin{cases} u = \lceil (\max X - \min X) / cell\_len \rceil \\ v = \lceil (\max Y - \min Y) / cell\_len \rceil \end{cases} \quad (6)$$

where $\lceil \ \rceil$ is top integral function. The Hash function for (px, py)in gridding is

$$\begin{cases} i = (\mathrm{int})(px - \min X) / cell\_len \\ j = (\mathrm{int})(py - \min Y) / cell\_len \end{cases} \quad (7)$$

where i and j are the key brick number of X and Y direction for (px, py).

B. Adjacent relation of dispersed two-dimension data

To reflect the drawing plane information at the observation point by the coordinate of neighboring point, the adjacency relation of data points have to be analyzed. With the definition, P={p1,p2,…,pn} is a set of all two-dimension data points, the K data numbers, which distance is the minimum, is the K-neighboring of pi, which is denoted as Neib(pi) [9]. The amount of space of N data points is N*k structure of boundary node. To obtain a better result and promote the speed of computing, the value of K is taken 10～20 commonly. The step to get K-neighboring of pi is as follows:

(1)To obtain the K-neighboring of one data point, we have to find the nearest K data points in 8 adjacent square grids, which are marginal to the grid of pi, which is the key to Plane partition of dispersed data. Then, the key brick number of pi can be computed. This method can save time to find K-neighboring of data point.

(2)To describe the K-neighboring of every data point, an undirected graph, which is formed by the line of two adjacent points, is built. In Fig. 7, the structure of every node of vertex has been shown, where index is the serial number of grid of the point, that is, the key brick number, and pointer is the needle, pointing to edge linked list of this data point. In Fig. 8, the boundary node of the data point and its adjacent point, it is formed by the node of vertex and distance has been shown. The space occupied by the K-neighboring of N two-dimension data points is N*k structure of boundary node. Because k is less than N, describing adjacent structure with adjacency matrix causes the waste of space. Evidence indicates that linked list is the best chance; the process of building linked list has been shown in Fig. 9.

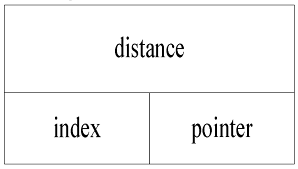


Figure 7. Node of vertex

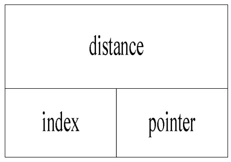


Figure 8. Structure of boundary nodes

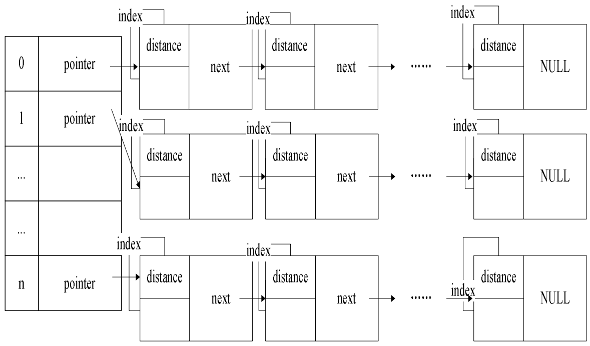


Figure 9. Data structure of K-Neighboring

C. Sparse 2D points through specified data points

The key of the algorithm is to obtain K-Neighboring of every point then find the organization of linked list for structure of K-Neighboring formed by k-1 boundary. To achieve expected point number of retention, the sequence of linked list is built by the distance of boundary and the set of two-dimension data is rarefied. The priority queue is built by the minimum distance of two data points. The distance of Hausdorff [10], the set of two data set, is described by d(S1, S2). Under the condition of $i = 1, 2, 3, \cdots, n$ and $j = 1, 2, 3, \cdots, n$, where $i \neq j$, the following relation could be obtained

$$d(p_i, S_1 - p_i) \leq d(p_j, S_1 - p_j) \quad (8)$$

where p_i should be rejected. Because Qt does not provide bool data types, the char data type is introduced, which length is n. If Deleted[i] is zero, the data point pi will not be deleted [11, 12, 13]. If Deleted[i] is one, the data point pi will be deleted. The idea of algorithm is expressed as follows:

(1)Initialization

minDist =a great number

existlNum = data points of the initializing set, that is n

wantedNum = data points of rarefied retention

Deleted[i] = 0 (i=1, 2, …, totalNum) all points is not deleted

(2)If existNum > wantedNum, go to step 3. Otherwise, the process of sparsing is completed, and the expected result has been obtained. Then, all points, which Deleted data array value is zero, are deleted from primitive one-dimension array.

(3)If the value of Deleted[i] is one, the mark is deleted for every point, go to pi+1 point. Otherwise, the first boundary node of adjacent linked list for pi, which is not deleted, that is, value of Deleted [index] is zero. If distance < minDist of boundary node, minDist = distance, and indexToBeDeleted = index, which is used to record variables, that is, the sequence of grid. If Deleted[indexToBeDeleted] = 1, the point marked indexToBeDeleted is deleted. If existNum = existNum-1, go to step 2. Studies on all points Cycle, two points of the minimum distance of is obtained [14, 15].

D. Bilinear Interpolation Algorithm

Processors always need the effect that deal with the less original data or gaps between two points by interpolate some data to make graphics smoother, which will bring in further help for the interpretation and analysis of seismic data. Especially in colored graphics, it often needs to process data by this method.

As we know, there are interpolation algorithms as following: nearest neighboring interpolation, linear interpolation, bilinera interpolation, spline interpolation .etc.

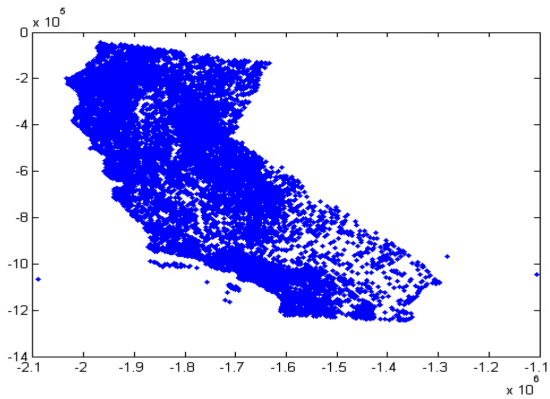
In this paper, bilinear interpolation is applied. It uses gray values of points around a point in coordinate in X and Y direction for linear interpolation. Assume that point (x, y) as the original point on the grahic, and point (x’, y’) is point of target graphic, then the bilinear interpolation is to find point (x’, y’)’s gray value in the original coordinate , and assume that u, v are float variables in the range of [0, 1], then point (x’, y’)’s gray value mapped to the original coordinate can be figured out by corresponding gray values of points (x, y), (x+1, y), (x, y+1), (x+1, y+1). Also it assumes that f(x, y) present gray value of point (x, y), then use bilinear interpolation of point (x’, y’) can be computed by following formula.

$$f(x',y') = (1-u)(1-v)f(x,y) + u(1-v)f(x+1,y) + (1-u)vf(x,y+1) + uvf(x+1,y+1)$$

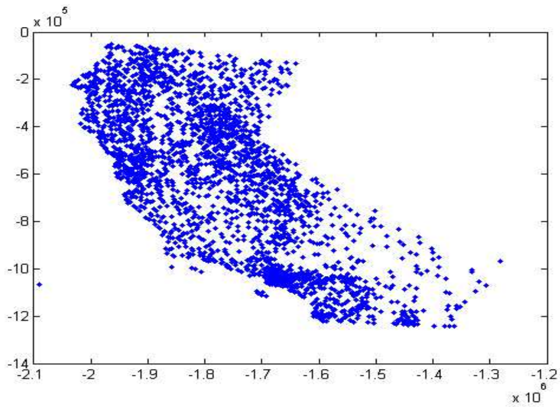
(9)

E. Verification

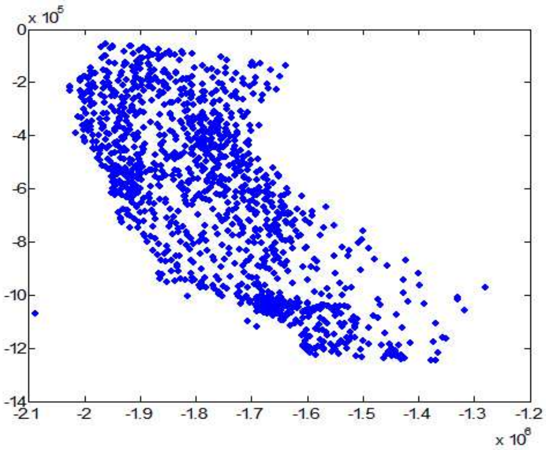
When the two-dimensional seismic data need to whole show, because large amounts of data mapped to the graphics device, a certain number of pixels are needed. The original mass of 2D seismic data exists point-one mapping relationship with the memory storage, because in the pixel-based graphics device, pixels are integers, not decimal form. And some 2D seismic data are chaotic, rules can not be found between data points which scattered arranged and distribution. So this type of data researchers need to see the overall distribution of seismic data and the trend when demands on precision is not very high , which helps them from the overall structure of the strata have a general understanding, and thus the post interpretation and reservoir locations may provide a basis. The two-dimensional data visualization is one of the key data to fast graphics rendering, so we can according to two-dimensional scattered data on the mass of the whole show, the first pumping dilute the data, reducing the amount of data need to coordinate map point which will be less, and thus need to draw to the graphics device also fewer data points, which played to the role of graphics drawing speed. After sparing of a series of two-dimension data of earthquake with this algorithm, Fig. 10 has been obtained. The primordial two-dimension data of earthquake, where the data points are 12512, has been shown. The figures, which are rarefied via K-neighboring and simplified based on the distance of specified data point and point number, have been shown in Fig. 10. The results of two-dimension earthquake data are before and after rarefying. (a) The holistic figure of primordial two-dimension data of earthquake where the data points are 12512; (b) The figure of specified 2503 data points, the used time of CPU is 25s; (c) The rarefied figures of 2271 data points based on specified threshold 0.5, the used time of CPU is 4s.



(a)



(b)



(c)

Figure 10. The results of two-dimension earthquake data before and after rarefying

From Fig. 10, we could see that the point numbers are about the same in Fig. 10(b) and Fig. 10(c), however, their difference in used time is large. Two algorithms promoted the speed of drawing by simplifying data points, but the optimization based on time and time travel used more time.

Fig. 10(a) shows the graphic of original seismic data, and Fig 10(b) in the neighborhood take a k-value k-5, only 2,503 data points, while Fig. 10(c) in the neighborhood take a k-value k-10, only 1,252 ~~a~~ data points, thus greatly improving the graphics rendering speed. Sometimes need to deal with two-dimensional seismic data up to hundreds of thousands or even millions of data points, data points, scattered through the use of vast amounts of data pumping thin plane algorithm (k-value close to the k value is usually 10-20), maintaining the original data points in the distribution of trends in the two-dimensional seismic data has greatly enhanced the visual rendering speed.

V. Experimental Results

In this paper, we use the schematic of Drawing Local Area Graphic of Massive 2D Data and Bitmap Cache Mode methods to improve paint or repaint rate of massive 2D data, and also we use K-Neighboring Sparsing Algorithm to show the whole graphics or zoom out the graphics of massive 2D data, which ensures the accuracy of the premise of graphics, quickly realizes the two-dimensional seismic data on the redraw. The test includes functional testing and performance testing of two parts. Functional testing includes: the mass of scattered 2D seismic data and local data show that the overall display, non-scattered seismic data (channel volume data) and the overall display local data, graphics scaling, according to the color on the color of the two-dimensional seismic data of the color graphics display, a small amount of data after interpolation of graphical display, a number of properties of two-dimensional seismic data sets display a graphical overlay. Performance test, reading data mainly from the massive amounts of data through the memory mapped file mechanism, graphical display of 2D seismic data speed through the zoom speed test after the graphical display. Through the different Segy, GM and other two-dimensional seismic data in different formats seismic attributes two-dimensional graphics data sets to test the two-dimensional visualization of seismic data visualization module functionality and performance. Fig. 11 and Fig. 12 show the actually massive 2D seismic data through the method of drawing local area data, Fig. 13 shows the result of Bitmap Cache Mode to improve Painting Speed, and Fig. 14 shows graphics of massive 2D seismic data nearly 2GB by applying K-Neighboring Sparsing Algorithm. Finally, Fig. 15 shows Graphic of a small amount of 2D seismic data and Fig. 16 shows using bilinear interpolation method of Fig.15's 2D data. Specific details are as follows.

Fig. 11 shows the first screen of 2D seismic data, due to large amount of data, using a split-screen graphics to draw the local data, the Y axis data range from 3,405,768 to 3,411,052 of TracelsNum attribute data, X axis data's range is 619,975~635,652 of FieldFileNum attribute data. Fig. 12 shows the group (FieldFileNum, TracelsNum) seismic attribute data of the second-screen graphics, X-axis data range with 6-3 (a), but the Y axis data range from 3,401,925 to 3,405,768 Graphics rendering by local data, and then use the scroll bar to update the current screen data and graphical display, which improves graphics rendering speed.

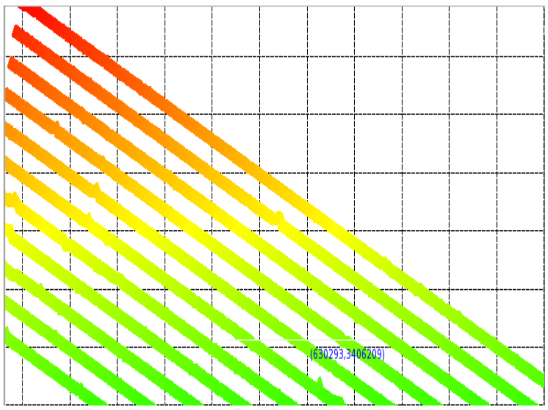


Figure 11. Drawing local area graphics of massive 2D seismic data

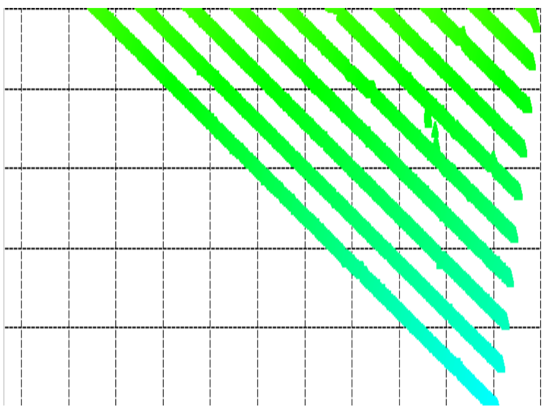


Figure 12. Drawing local area graphics of massive 2D seismic data

As long as the user change the graphics display area in the course of events caused redraw graphics, graphics have not changed in substance, in order to improve efficiency in this case, redrawing the software interface will be drawn in two-dimensional graphics to bitmap stored in the secondary cache, when made into a redraw event, the software no longer retrieves the drawing from memory, data mapping, but directly from the secondary cache in bitmap, re-map to achieve redraw. This greatly improves the operating efficiency. The software only needs to change the graphical display, re-obtain the data to redraw, and update the secondary cache in the bitmap, in other cases, direct access to re-paste a bitmap achieve redraw. All reasons validated in Fig. 13.

Fig. 14 shows the attributes of seismic data set (SourceX, SourceY) of the original graphics, because of a large amount of data of this graph, the tall and more than 120,000 points. Thus, appropriate scaling method achieves the graphics, zooming in and out.

Fig. 15 shows the twice enlarged after the Fig. 16 shows a graphical doubled reduce the graphics. Graphic shows the superposition of multiple sets of data is re-sampled and the redraw of a process. The traditional method is to overlay graphic shows the need for unity of all the data stored (such as unified on the same array or file), then the data is consistent graphics. Drawbacks of this approach is due to a number of image data stored in an array, which can not be distinguished, so the user’s requirements, to the different points of the color image data connectivity can not be achieved, but not all the data stored together conducive to data management. Therefore, this bold attempt of the different seismic data sets were drawn at different drawing equipment (such as QPixmap), the graphics data are separated, then multiple images of overlapping and transparency of data processing to image in multiple data a graphics display. This separation of elements by drawing the superposition principle can achieve unlimited stack, thus more easily study the properties of elements to reduce blindness. Multi Image Display in a specific process, the two-dimensional graphics display module to the first drag each new graph, draw the image to be displayed in a separate graph, and then get a graph of the mapping data for each range, based on each. The data rate range of graphic drawing data to determine an overall range of data for each individual graphic according to the new data range scaling, the final ratio will be adjusted after the graphics overlap multiple images in the same data in a graphical display. For each piece of graphic drawing data, two-dimensional graphics display module are stored in different arrays, then the first address for each array index pointer are stored, in order to achieve the various data image drawing data distinctly. This graphic for display in the same image in different data connection will be able to achieve the color-coded, so that treatment can clearly see images in the graphics data in different position and shape.

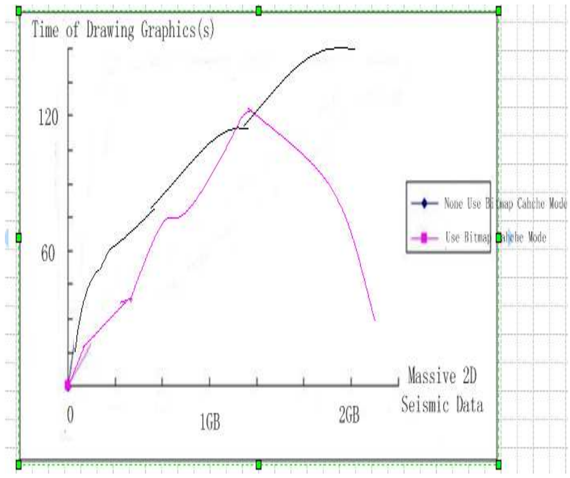


Figure 13. Comparison between use and none use Bitmap Cache Mode

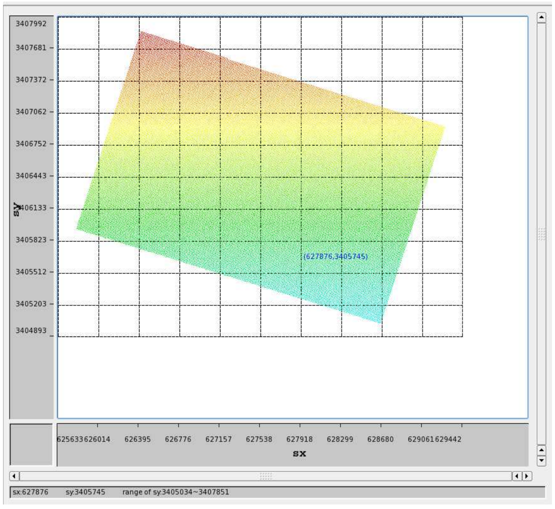


Figure 14. Use K-Neighboring sparsing algorithm to draw massive 2D seismic data

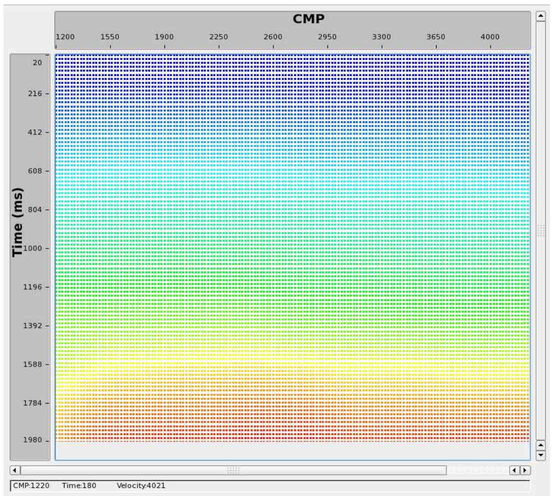


Figure 15. Graphic of a small amount of 2D seismic data

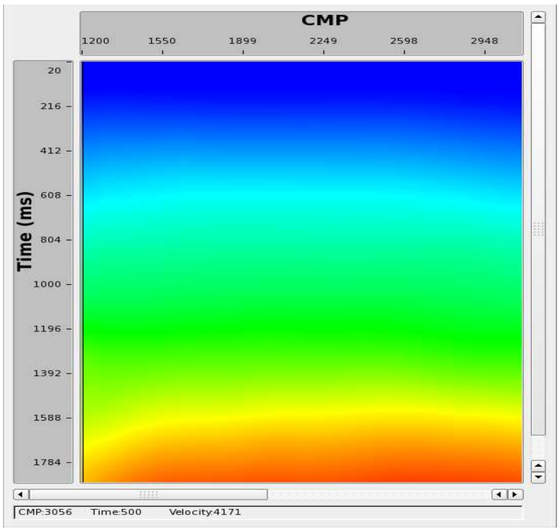


Figure 16. Use bilinear interpolation method of Fig.15's 2D data

VI. CONCLUSIONS

This paper presents a visualization system of massive 2D seismic data. Firstly, we should process massive data in point coordinate (x, y) forms. Secondly, as we all know, pictures can't be shown completely in a displayed screen, so we draw local area of graphics by figuring out what points need to be painted, which result in a high rate of drawing and less usage of memory. Thirdly, it picks out a Sparsing algorithm by K-Neighboring when we need to see the whole picture of massive 2D data or when we need to zoom-out the whole picture while to guarantee the quality of pictures. In this paper, we base on Qt to implement high rate rendering of massive 2D data.

REFERENCES

[1] Z. Wang, "Sparsing algorithm of massive dispersed two-dimensional data based on K-Neighboring," *In: Proceedings of IEEE International Conference on Apperceiving Computing and Intelligence Analysis,* pp. 376 – 379, 2010.

[2] F. Admasu, K Toennies, "Anisotropic 3D seismic features for robust horizons correlation across faults," *In: Proceedings of IEEE International Conference on Image Processing. Carolina, USA,* pp. 221-225, 2010

[3] M. Fokaefs; D. Serrano; B. Tansey, "2D and 3D visualizations in WikiDev2.0," *In: Proceedings of IEEE International Conference on Software Maintenance,* pp:1-5,2010

[4] T. Yamashita, H. Muneo, and K. Kajiwara, "Petascale Computation for Earthquake Engineering," *Computing in Science & Engineering,* vol. 1, pp. 2-4,2011

[5] G. Tang, J. Ma, "Application of Total-Variation-Based Curvelet Shrinkage for Three-Dimensional Seismic Data Denoising," *Geoscience and Remote Sensing Letters,* vol. 8, pp. 103-107,2011

[6] N. Wu, Y. Li, and B. Yang, "Noise Attenuation for 2-D Seismic Data by Radial-Trace Time-Frequency Peak Filtering," *Computing in Science & Engineering,* vol.8, pp. 874-878,2011

[7] B. Sassi, E. Djamel, "2D and 3D visualization of AspectJ programs," P*rogramming and Systems (ISPS)*, pp. 183-190, 2011

[8] J. Zhang, Y. Yang, and Q. Dai; "A novel 2D-to-3D scheme by visual attention and occlusion analysis," *The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1-4, 2011

[9] D. Pineo, C.Ware, "Data Visualization Optimization via. Computational Modeling of Perception," *Visualization and Computer Graphics*, pp. 1-3, 2011

[10] R. Jianu, C. Demiralp, and D. Laidlaw, "Exploring Brain Connectivity with Two-Dimensional Neural Maps," *Visualization and Computer Graphics,* vol. 99, pp.3-5, 2011

[11] A. Shalbaf, H. Behnam,and P. Gifani, "Automatic detection of end systole and end diastole within a sequence of 2-D echocardiographic images using modified Isomap algorithm," *Biomedical Engineering (MECBME)*, pp.217-220,2011

[12] D, Hong, T. Wallapak, and J. Wong, "Colon fold contour estimation for 3D visualization of colon structure from 2D colonoscopy," *images Biomedical Imaging: From Nano to Macro,* pp. 121-124,2011

[13] D. Patel, C. Giertsen, and J. Thurmond, "The Seismic Analyzer: Interpreting and Illustrating 2D Seismic Data," *Visualization and Computer Graphics,* vol. 14, pp. 1571-1578, 2008

[14] Y. Tian, Z. Weng, and X. Mao, "Study on key technologies of three-dimensional GIS in geosciences," *Environmental Science and Information Application Technology (ESIAT),* pp.166-169, 2010

[15] H. Wu, J. Li, and A. Ruan, "Seismic imaging of oceanic crust across the northwestern subbasin," *South China Sea Oceans*, pp.1-4,2010

Wenqin Li was born at Qu Xian County, Sichuan Province, China on October 1, 1976. She got her bachelor's degree of engineering in the field of education of computer science in College of Computer Science, Southwest Normal University, Chongqing, China in July 1999. She also got her master's degree of engineering in applied technology of computer science in College of Computer Science, Chongqing University, Chongqing, China in June 2008.

She has worked as a teacher in Yangtze Normal University since 1999 and later in 2004 she was promoted as a lecturer. What interest her are data mining and education informationization.

Daqing Wang was born at Wuxi County, Chongqing, China on July 14, 1976. He got his bachelor's degree of engineering in the field of computer science and technology in College of Computer Science, Southwest Normal University, and Chongqing, China in June 2005.

He has been a teacher in Yangtze Normal University since 2000. In 2007, he became a technician in Yangtze Normal University.

His research fields include intelligent computer and software engineering.