A Polygon Rendering Method with Precomputed Information

Seunghyun Park[#1], Byoung-Woo Oh[#2]
Department of Computer Engineering, Kumoh National Institute of Technology, Korea
[1] seunghyunpark12@gmail.com
[2] bwoh@kumoh.ac.kr (Corresponding Author)

Abstract—The spatial data contain a lot of points. The points consist of geographic coordinates. Representing the spatial data requires many calculating the geographic coordinates. Calculations of the coordinates take much time and reduce the performance. In order to enhance the performance of rendering the spatial data, performance enhancement for rendering the polygon which are in spatial data is needed. In this paper, we present that improving the performance of rendering for polygon by utilizing the scanline algorithm. The scanline algorithm has been widely used for rendering graphics systems. It fills inside of a given polygon. In order to filling the polygon, it requires some calculations and tables for storing the variables. Calculating and sorting the variables reduce the performance. We propose a method which precomputes the variables used by the scanline algorithm. By using proposed method, it can reduce the processed time. Proposed technique can be utilized to zoom in and out the map. We implement the proposed method and perform the experiment with Tiger/Line spatial data. We compare the processing time between scanline algorithm and proposed method.

Keyword-Scanline Algorithm, Rendering, Polygon, Precomputed Information

I. INTRODUCTION

Researchers are studying to process the spatial data efficiently. Researches for processing spatial data have been widely processed in many areas such as data structure, multimedia, parallel processing and etc. In data structure area, some researchers have searched for data structures which are useful to process spatial data efficiently [1], [2], [3]. Researches have been proceeded for image processing in the multimedia area [4], [5], [6]. In parallel processing area, some researchers have searched for processing spatial data by using GPU [7], [8], [9]. By advancing in mobile technology such as smart phones and tablet PC, spatial data are utilized in mobile environment. There are some difficulties to represent the spatial data. The Spatial data consist of many polygons. The spatial data consist of many polygons. Improving the performance of rendering the polygon can improve the performance. We propose a polygon rendering method by utilizing Scanline Algorithm. Scanline algorithm is widely used for rendering systems [10], [11]. Scanline Algorithm fills the inside of the polygon. The areas of filling are determined by intersection points where scanline and edges of the polygon intersect.

The spatial data contain many polygons. There exist a lot of points in the polygon. To represent the polygon on a map, it requires many calculations for the geographic coordinates. It takes much time to calculate the coordinates of the polygons. In order to reduce the rendering time, we propose that precomputing variables which are important to render the polygons. The variables are stored in Sorted Table. It can be reduce the sorting time when the sizes of the map are changed. The map can be zoomed in / out. The coordinates of the polygon are changed. In order to draw new sizes of the map, there is a need to calculate new coordinates. It takes much time. To solve this problem, we precompute the variables for rendering the polygon. When the map is changed at certain magnification, we multiply the magnification to the Sorted Table. There is no need to calculate new coordinates. The Sorted Table can be reused every time for changing the sizes of the map. It can reduce the processing time to render the polygon.

This paper is organized as follows. Chapter 2 introduces Scanline Algorithm. Chapter 3 describes the proposed method for representing the polygon using Scanline Algorithm and precomputing the variables for polygon rendering. Chapter 4 reports the result of the experiment. Finally, chapter 5 presents conclusion and future work.

II. RELATED WORKS

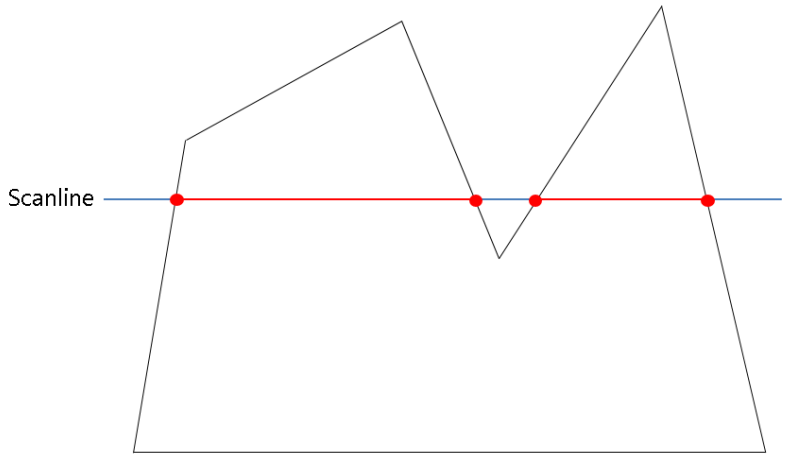

Fig. 1. Example of Scanline Algorithm

Scanline Algorithm is filling inside the polygon where scanline and edges of the polygon intersect. Scanline is a line which is horizontal to y-axis. Fig. 1 shows example of Scanline Algorithm. In Fig. 1, Scanline intersect with edges of polygon 4 points. Scanline algorithm fills the inside of the polygon which are intersected with scanline. Scanline moves from the lowest y coordinates of the polygon to the highest y coordinates. In order to filling inside the polygon, there are some calculating coordinates. To draw a line, line segment requires x and y coordinates and slope of the line. Scanline Algorithm requires 4 variables such as Y-Min, Y-Max, X-Val and Slope. Y-Min is a lower value of the y coordinates for vertices of the edge. Y-Max is a higher value of them. X-Val is the x coordinates of the vertex which has the Y-Min variable. Slope variable is the value of slope for edges. The variables are used for filling inside of the polygon. Scanline Algorithm uses tables for calculating and storing variables. In All_Edge Table, the variables are calculated between each edge. The variables are stored in Global Table. Before storing the variables, Scanline Algorithm treats one exception. Scanline Algorithm draws a line horizontally to y-axis. The slope which value is horizontal to y-axis is not necessary to draw a line. The indexes which have the above slope are not stored in Global Table. In the Global Table, the variables are sorted in order of the values of Y-Min. After sorting the variables in that order, the variables are sorted in order of the values of X-Val. Scanline moves along by the y-axis. When value of Scanline is same with the value of Y-Min, the indexes which have the value of Y-Min are stored in Active Table. In Active Table, a line is drawn and the values of X-Val are renewed. The value of Scanline is increased and draws a line sequentially. The value of Scanline is same with the value of Y-Max. The indexes which have the value of Y-Max is got rid of Active Table.

III. ALGORITHM FOR PRECOMPUTING THE INFORMATIONS FOR POLYGON RENDERING

We propose a method for polygon rendering with precomputed information. The main idea consists of two parts: storing variables which are precomputed in Sorted table and rendering the polygon utilizing those variables. The variables are stored in the Sorted table. The Sorted table are reused every time when the map is zoomed in / out. By reusing the Sorted table, it can reduce the calculating and sorting times.

A. Algorithm Utilizing in Proposed Method

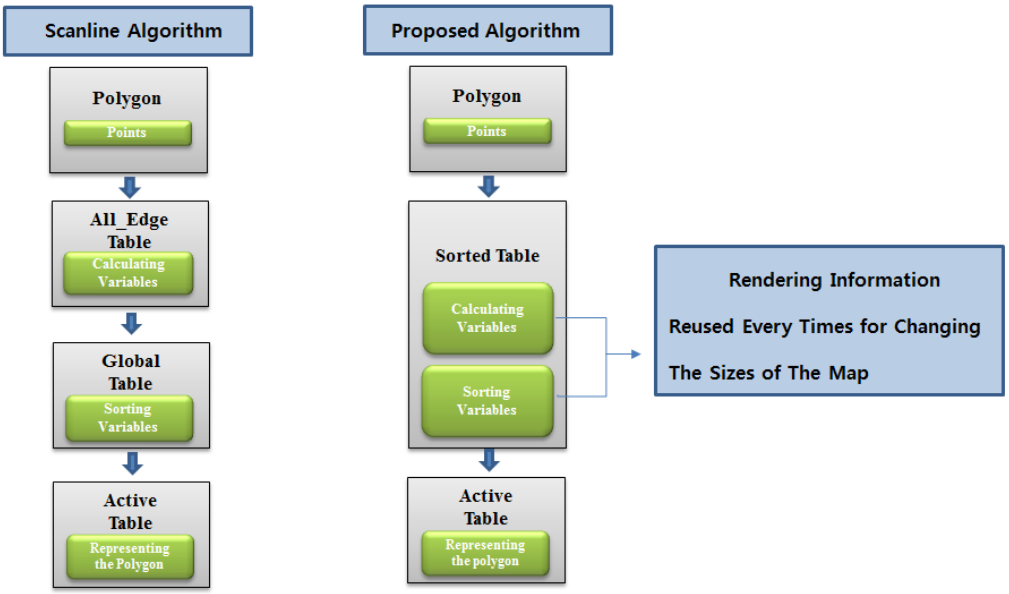

Fig. 2. Comparison with Scanline Algorithm and Proposed Algorithm

To represent the polygons on a screen, the spatial data which contain polygons are loaded to memory and processed calculations for coordinates. The spatial data contain many points which are in the polygon. In general, the spatial data are very large. By volumes of the spatial data are increased, the number of the calculations for coordinates are increased. It reduces the performance for representing the polygon. In this paper, proposed algorithm precomputes the variables for polygon rendering and stores the variables in the Sorted Table.

Fig. 2 shows difference between Scanline Algorithm and proposed algorithm. Scanline Algorithm uses 3 tables: All_Edge, Global and Active. In the All_Edge table, variables for Scanline Algorithm are calculated. The variables are stored in the Global table. In the Global table, the variables are sorted. To render the polygon, the variables in the active table are used. Calculating and sorting variables take much time.

In this paper, calculating and sorting variables are processed previously and storing the variables in the Sorted Table. As you can see Fig. 2, proposed algorithm does not use All_Edge and Global Table. Proposed algorithm calculates and sorts the variables at a time. Sorted variables are stored in the sorted table. Proposed algorithm can be utilized usefully when the sizes of the map are changed. When the map is zoomed in / out, the coordinates of the polygon are changed. The variables for rendering are recalculated. In order to solve this problem, proposed algorithm utilizes the sorted table. In Scanline Algorithm, the changed variables are re-calculated in the All_Edge table and re-sorted in the Global Table. Proposed algorithm can omit those steps. The variables for rendering are stored in the Sorted table. It sends the index of the sorted table to Active Table. By multiplying magnification to stored variables, it can reduce the calculating and sorting operations. The Sorted Table can be reused for every change of sizes of the map.

B. Precomputing the Variables

To apply proposed algorithm, some variables such as Y-Min, Y-Max, X-Val and slope are calculated. To calculate the variables, proposed algorithm is adjusted to each part of whole polygon. These variables are acquired from vertices of each edge. The variables are x and y coordinates of edges. Lower value of y coordinates is Y-Min. X-Val is x coordinate of point which has Y-min variable.

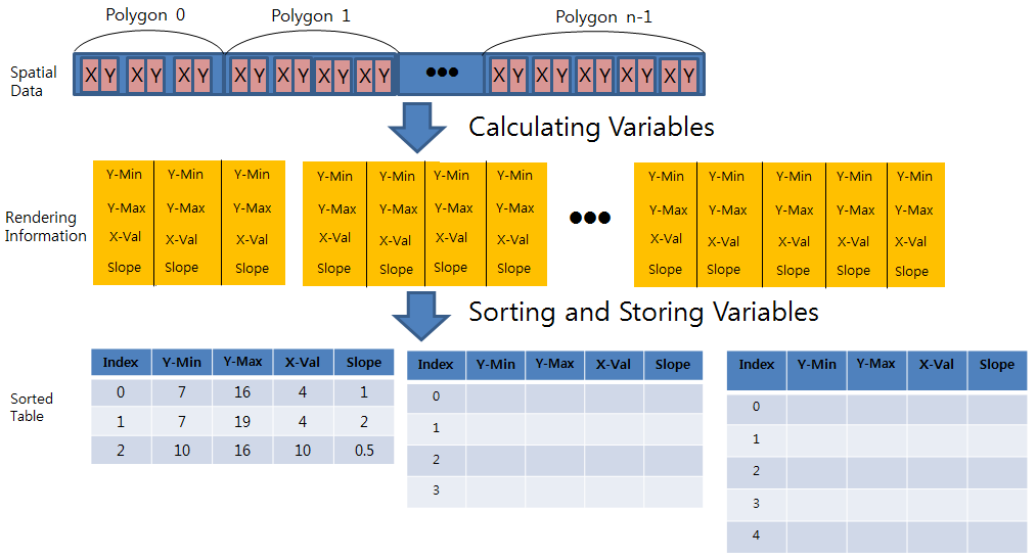

Fig. 3. Processing of Proposed Algorithm

Fig. 3 shows processes of calculating the coordinates. In the Fig. 3, the spatial data has n polygons. There are points which have x and y coordinates in the each polygon. The polygon 0 has 3 points. Each point is calculated. 4 variables such as Y-Min, Y-Max, X-Val and Slope are made. The variables should be calculated in the whole edges. After acquiring the variables of first edge, storing the next variables from linked edges which are connected with first edge is necessary.

These steps are continued to the last edges of the polygon. After calculating the whole variables of the edges in the table, the variables should be sorted. Before sorting the variables, values of slope are same with horizontal to y-axis. Because Scanline Algorithm fills the inside of the polygon along by y-axis, the slope which is horizontal to y-axis is unnecessary. After eliminating the index which has the above slope, sorting the table by the value of Y-min is processed. Because Scanline moves Y-Min to Y-Max, the table should be sorted by Y-Min value. If the table are not sorted, the polygon cannot be represented correctly. After sorting the table by the value of Y-min, the table are sorted by the values of X-Val which have same value of Y-Min. Because lines are drawn from the lowest value of X-Val to the largest value of X-Val, the value of X-Val should be sorted. In the fig. 3, the variable which has the smallest value of Y-Min stores in the index 0. The following variable is stored in index 1. Variable which has the biggest value of Y-Min stores in the index 2. Above processes are continued to other parts sequentially.

Sorted tables can be reused when map is zoomed in / out. The sizes of the map can be changed by random magnification. Coordinates of the polygon are also changed. The variables of the edges should be re-calculated. Re-calculating the variables takes much time. In order to reduce the re-calculating time, calculating and sorting the variables are processed in advance and storing the variables in Sorted Table. When the map is zoomed in / out by random magnification, Sorted Table is reused. By multiplying magnification to variables, it can reduce time to calculate and sort the variables.

C. Representing the Polygon

```
Render the Polygon

Read the Stored Table of First Part;

while(Scanline moves in the Polygon){
        if(Scanline meets the value of Y-MIN){
                Store the Indexes of the Stored Table to
                Active table;
                Render a Polygon to the Map;
                }
}

Read the Stored Table of Next Part;
```

Fig. 4. Algorithm for Rendering The Polygon Using Proposed Method

Fig. 4 shows algorithm for rendering the polygon using proposed algorithm. In order to render the polygons, reading Sorted Table is processed. When scanline meets Y-min, the Sorted Table sends the index of Y-min to Active table. The inside of the polygon is filled from the lowest X- Val to the highest of X-Val. After drawing

the line, value of the scanline is increased. X-Val is renewed by adding Slope to X-Val. The polygons are filled with changed X-Val. During those stages, Active table are changed.

There exists many each value of Y-Min. Whenever scanline meets Y-Min, the indexes are included in Active table. The range of filling polygon is Y-Min to Y-Max. When the value of scanline is same with Y-Max, Active table gets rid of the indexes which have Y-Max. After scanline moves the highest value of Y-Max which is in the Sorted Table, the polygon of a part is fully filled. Proposed algorithm renders other parts of the spatial data.

The Sorted Table can be reused for the map which sizes are changed. When the map is zoomed in / out at magnification, magnification is multiplied to the Sorted Table. It can reduce the calculating and sorting the variables for renewed coordinates. , The ranges of the scanline move are changed from new value of Y-Min to Y-Max. Finally, proposed algorithm renders the whole parts to the map.

IV. Experiment and Result

All experiments are performed on a machine Microsoft Windows 7 with an Intel® Core™i5 CPU running at 3.50GHz and 8 GB of memory.

A. Data Set and Implementing the Experiment

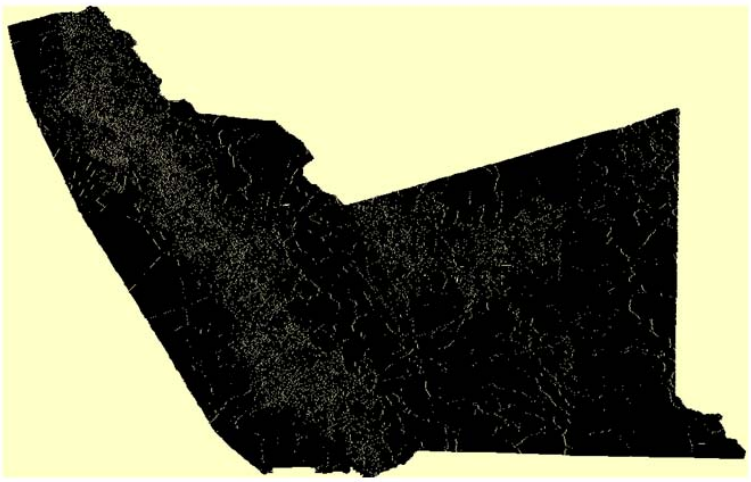

Fig. 5. Data Set for the Experiments

The data set for experiment is a faces of Alameda County of the California state. Type of the data set is polygon. The data set is provided by the Tiger/Line data. The TIGER/Line data is developed by the U.S. Census Bureau. The U.S. Census Bureau provides the data on its website [12]. The TIGER/Line data contain geographic features such as roads, rivers, and statistical geographic areas, etc. Figure 5 shows whole data set. The data set consists of 32,080 polygons.

In order to acquire reliable result, we executed experiment 1,000 times. We executed experiment representing the spatial data at true ratio and changing the magnification to 5.0 times.

B. Result

The experiment was executed by Scanline Algorithm and proposed algorithm. Execution times for representing the spatial data were gathered by representing the spatial data by true ratio. Other execution times were gathered by zooming in / out the map. These execution times were compared. Total Execution times were gathered by adding sorting times and drawing times. Total execution times which the experiment was executed at true ratio on Scanline Algorithm were took 21.48ms. Total execution times on proposed algorithm were 1.68ms. The gap of the two experiments is 19.80ms. When the map is zoomed in at 5.0 times, the gap of the tow experiment is 19.72ms. The gaps of the whole experiments are very similar. The execution times of the two experiments are increased linearly. In contrast, the proposed algorithm is much faster than Scanline Algorithm. Because proposed algorithm used Sorted Table which storing the variables for rendering, proposed algorithm reduce the times for sorting. The sorting times occupied most of the total execution times. The proposed algorithm got rid of sorting times and made performance better.

As can be seen in Figure 6, proposed algorithm is much faster than Scanline Algorithm when the sizes of the map are changed. The gap of execution times arose mainly in the Sorting times. Sorting the variables for rendering require many operations and takes much time. The proposed algorithm gets rid of the Sorting time by using Sorted Table. Sorted Table stores the variables which are calculated and sorted in advance. When drawing a map which sizes are changed, the proposed algorithm loads Sorted Table and multiplies magnification to

Sorted Table. In contrast, drawing the changed map on Scanline Algorithm recalculate and resort the variables for rendering. This could reduce the total execution times

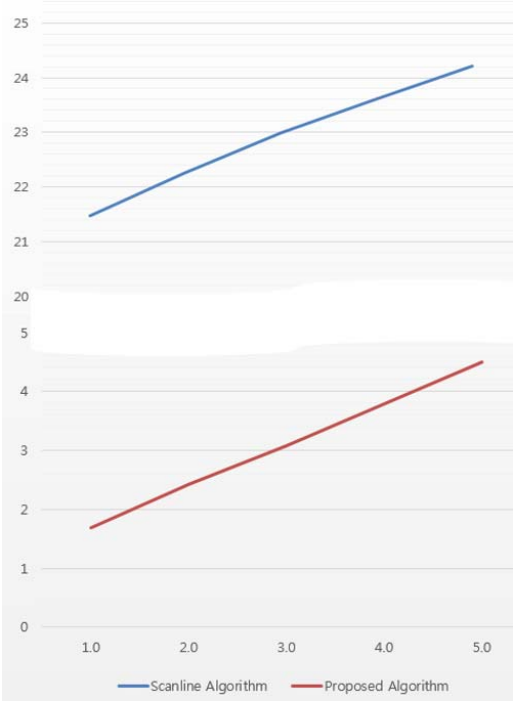

Fig. 6. Comparison of the Execution Times between Proposed Method and Scanline Algorithm

TABLE 1. Result of the Experiments

Magnification	1.0	2.0	3.0	4.0	5.0
Scanline Algorithm	21.48 ms	22.26 ms	22.98 ms	23.61 ms	24.21 ms
Proposed Algorithm	1.68ms	2.41 ms	3.08 ms	3.79 ms	4.49 ms

V. CONCLUSION

This paper proposed the method for rendering the polygon. The proposed method can reduce the calculations. The spatial data are usually large. The spatial data contain many polygons and points. In order to represent the spatial data to a map, the polygon which are in the spatial data should be rendered. In order to render the polygon, it requires the calculations for the geographic coordinates. The number of the calculations is increased as points which are in the polygon are increased. This reduces the performance for rendering the polygon. Furthermore, the sizes of the map can be changed. The coordinates of the polygon are changed. There is a need to calculate the new coordinates. Therefore, this paper proposed the method which using the Sorted Table. The Sorted Table stores the variables for rendering the polygon. The variables are precomputed. It can reduce the number of calculations and sorts. The Sorted Table are used for the changing the sizes of the map. The Sorted Table is existed in memory before the map is drawn. When the sizes of the map are changed, The Sorted Table is loaded for rendering the spatial data. It can enhance the performance. In order to prove that proposed method is efficient, the execution times are compared with drawing time between Scanline Algorithm and proposed method. The execution times on proposed method are much faster than executed on Scanline Algorithm. The gaps of execution times were 19.72ms. The both execution times at increasing magnification increased linearly.

Our future works could focus on applying graphic API such as DirectX, OPEN GL and etc. Applying the graphic API to the proposed method could make the proposed method faster by rendering the polygon by accessing memory directly.

ACKNOWLEDGMENT

This paper was supported by Research Fund, Kumoh National Institute of Technology.

REFERENCES

[1] I.S. Sitanggang, R. Yaakob, N. Mustapha, A.A.B. Nuruddin, "An extended ID3 decision tree algorithm for spatial data," in Proc. Of 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), 2011, pp. 48-53.
[2] B. Ghimire, S. Bhattacharjee, S.K. Ghosh, "Analysis of Spatial Autocorrelation for Traffic Accident Data Based on Spatial Decision Tree," in Proc. Of 2013 Fourth International Conference on Computing for Geospatial Research and Application, 2013, pp. 111-115.
[3] L. Guobin, J. Tang, "A New Method about spatial data query based on R*Q-Tree," in Proc. Of 2010 Sixth Conference on Natural Computation, 2010, pp. 1044-1047.
[4] C. T. Lu, L.R. Linag, "Wavelet Fuzzy Classfication for Detecting and Tracking Region Outliers in Meteorological Data," in Proc. Of 12th annual ACM International Workshop on Geographic Information Systems, 2004, pp. 285-265.
[5] T. Yonezawa, K. Ueda, "Real-time 3D data reduction and reproduction of spatial model using line detection in RGB image", in Proc. Of 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems, 2014, pp. 727-730.
[6] C. Stein, M. Limper, A. Kuijperm, "Spatial data structures for accelerated 3D visibility computation to enable large model visualization on the web," in Proc. Of 19th International ACM Conference on 3D Web Technologies, 2014, pp. 53-61.
[7] S. Puri, S.K. Prasad, "Efficient parallel and distributed algorithms for GIS polygonal overlay processing," in Proc. Of 2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2013, pp. 2238-2241.
[8] M. McKenney, S. Hill, G. D. Luna, L. Lowell, "Geospatial overlay computation on the GPU," in Proc. Of 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2011, pp. 473-476
[9] J. Zhang, S. You, L. Gruenwald, "Large-scale spatial data processing on GPUs and GPU-accelerated clusters," SIGSPATIAL Special, Vol. 6, Issue 3, pp. 27-34, Nov. 2014.
[10] D.Kim, K. Cha, S. Chae, "A high-performance OpenVG accelerator with dual-scanline filling rendering," IEEE Transactions on Consumer Electronics, Vol. 54, Issue 3, Aug. 2008.
[11] W.C. Lin, J.H. Ye, D.W, Yang, S.Y Huang, M.D. Shieh, J. Wang, "Efficient scissoring scheme for scanline-based rendering of 2D Vector Graphics," in Proc. Of 2012 IEEE International Symposium on Circuits and Systmes, 2012, pp. 766-769.
[12] (2015) U.S Census Bureau, TIGER products website. [Online]. Available: http://www.census.gov/geo/www/tiger

AUTHOR PROFILE

Seunghyun Park received his B.S degree in computer engineering from Kumoh National Institute of Technology, Korea in 2014. He is in course of M.S degree. His research interests include parallel processing and mobile software.

Byoung-Woo Oh received his B.S., M.S. and Ph.D. degree in computer engineering from the Kon-Kuk University, Korea in 1993, 1995 and 1999 respectively. From 1999 to 2004, he was a senior researcher in the Electronics and Telecommunications Research Institute (ETRI), Korea. He is a Professor in Department of Computer Engineering, Kumoh National Institute of Technology, Korea since 2004. His research interests include spatial database, automotive telematics, Geographic Information System (GIS) and Location Based Service