Using data reduction to improve the transmission and rendering performance of SVG maps over the Internet

Haosheng Huang[1] and Yan Li[2]

[1]Institute of Geoinformation and Cartography, Vienna University of Technology

[2]School of Computer, South China Normal University

Abstract: SVG is a promising standard for publishing geospatial data over the Internet. But when using SVG to publish very large geospatial dataset, we always have to wait a long time for the SVG map to load. The delay of loading SVG maps over Internet includes the network transmission delay, and the rendering delay in client side. This paper proposes a visual lossless data reduction method to reduce the data amount of SVG maps while preserving the visual effects (visual anti-distortion). In order to prove the proposed method is feasible and operable to improve the transmission and rendering performance of SVG maps over the Internet, we implement the method and use some sample data for method evaluation.

Keywords: data reduction, visual lossless data reduction, relative coordinates, SVG Map

1. Introduction

With the rapid development of the Internet, web-based spatial information system is an inevitable trend. W3C's SVG[1] is a promising standard for publishing geospatial data over the Internet. Many scientists or engineers have tried to use SVG to create a web spatial information system or to publish the maps on internet. They used the shape elements and graphic style of SVG for map display and developed some prototype systems.

But when using SVG to publish (represent) very large geospatial dataset, we always have to wait a long time for the SVG map to load. The delay of loading SVG maps over Internet includes two parts: the network transmission delay and the rendering delay in client side. The latter mainly relates to the performance of SVG Viewers (such as Adobe SVG Viewer), and the complexity (such as xlink, filling, animation, filter effects, gradients, patterns, clipping, masking and compositing) and the data amount of SVG documents (maps). The network transmission delay is mainly caused by the data amount of SVG maps and the network transmission speed. In general, the former is far longer than the latter. Thus, reducing the data amount of SVG maps will shorten

the network transmission time, and then improve the network transmission and rendering performance.

Data reduction is to filter out some less-important information and then reduce the data amount. If the data after reduction is acceptable (for example, visually acceptable), the data reduction method is feasible. There are two kinds of data reduction: lossless data reduction, and lossy data reduction. Both of them can be used in reducing the data amount of SVG maps.

Lossless compression (reduction) algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error [2]. Lossless compression schemes are reversible so that the original data can be reconstructed. There are a lot of lossless compression algorithms available, such as Lempel-Ziv[3], DEFLATE (Gzip)[4] , LZR. Some HTTP Servers also provide compression module, such as Apache web server's mod_deflate module [5], to allow data to be compressed before being sent to the client over the network, and then be auto-decompressed by web browsers such as Internet Explorer (IE), Firefox, Opera, Safari. We can use these algorithms (HTTP Servers) to reduce the data amount of SVG map before sending the SVG map to the client, and then decompress (reconstruct) the data to the original SVG map.

Lossy data compression (reduction) is possible if some loss of fidelity is acceptable. Generally, a lossy data compression will be guided by research on how people perceive the data in question [2]. Lossy compressions will lose some data and is unreversible. There are some lossy data compression algorithms which is special for GIS data (geometric data) which can be used prior to sending the data, such as line simplification (Douglas-Peucker algorithm [6]), map generalization. But these algorithms are too complicated (such as high time and space complexity), and still under developing. Based on computer graphics' viewing transformation principle (the mapping of a part of a world-coordinate scene to device coordinates), this paper proposes a visual lossless data reduction method to reduce the data amount of SVG maps while preserving the visual effects (visual anti-distortion). The SVG maps created by this method can adjust their data amount dynamically to the size of the output screen and the maximum visible scale of the dataset.

The paper is arranged as follows. In section 2, we introduce the viewing transformation principle, and discuss why we can reduce the data amount. Section 3 proposes a visual lossless data compression method for SVG map based on viewing transformation principle. In section 4, we carry out some tests to evaluate the suggested method. Finally, the conclusions and future work are presented in section 5.

2. Viewing transformation principle

In this section, we will introduce computer graphics' viewing transformation principle, and then explain why we can reduce the data amount of SVG map.

2.1 Viewing transformation principle (window-to-viewport transformation)

In computer graphics, a world-coordinate area (always rectangle) selected for display is called a

window, an area on a display device to which a window is mapped is called a viewport. The window defines what is to be viewed; the viewport defines where it is to be displayed. In general, the mapping of a part of the world-coordinate scene to device coordinates is referred to as a viewing transformation. Sometimes the 2D viewing transformation is simply referred to as the window-to-viewport transformation.

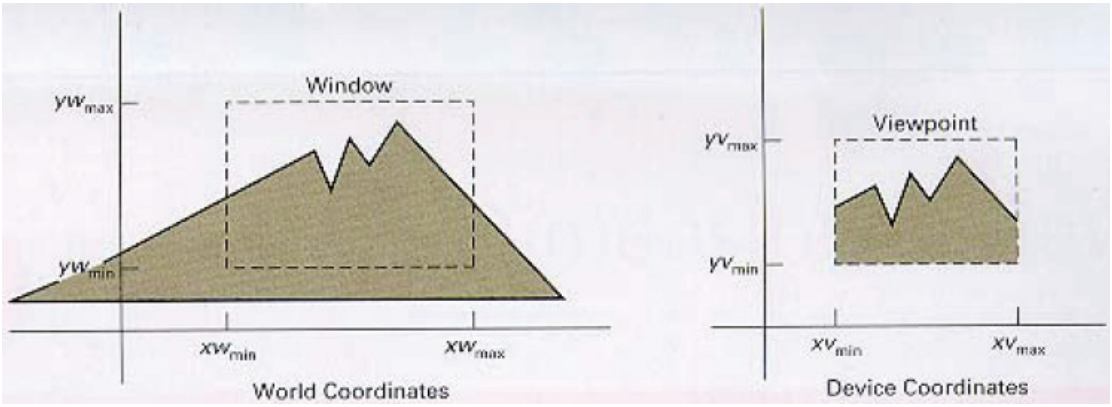

Figure 1. Viewing transformations

Transformations from world to device coordinates involve translation, rotation, scaling, and clipping.

2.2 Do we need so much detail?

Geospatial datasets are always stored/ represented in database (such as SQL Server 2000, Oracle) or some initial dataset (such as Shape, E00, or GML file). When using SVG to publish geospatial dataset (a real world area selected for display), people always "move" the geographic (world) coordinate pairs directly from initial dataset to SVG document. The value of attribute "viewBox" in root svg element is always set as the bounding of the geospatial dataset (the real world area). The bounding is always represented as a list of four world-coordinate numbers min-x, min-y, width and height, separated by whitespace, which specify a world-coordinate area (rectangle) selected for display. For example, when we try to use SVG to publish the administerial district map of Guangdong Province (China), the value of "viewBox" always set as "94928 2172873 790615 595213" which is the bounding of Guangdong Province.

In order to visualize the SVG map, a display area on the monitor should be selected to put the SVG map. As pixel is the smallest unit that can be displayed on a screen, according to the viewing transformation principle, suppose that the display area of client side's screen (monitor) to render the above SVG-based administerial district map is 1280*964 pixels, all the points in area of (790615/1280)*(595213/964) =617*617 in Guangdong district map will be mapped to one pixel (Figure 2). Thus, when using SVG to visualize geospatial dataset, it is unnecessary to use world coordinates in SVG map.

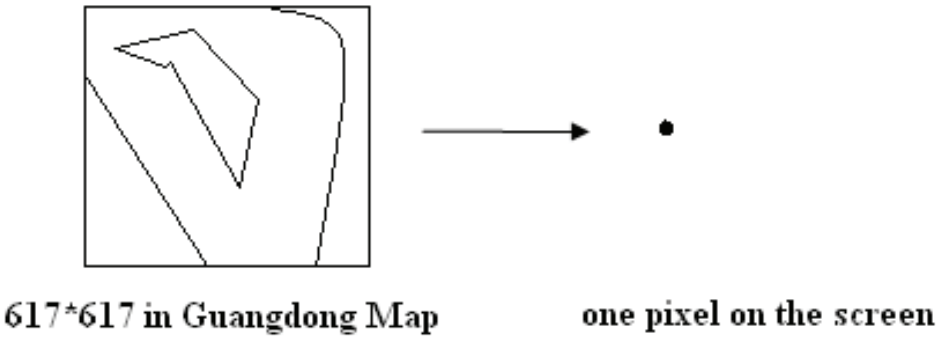

Figure 2. Do we need so much detail?

In the following, we will introduce a visual lossless data compression method for SVG map based on viewing transformation principle.

3. A visual lossless data compression method for SVG map

The basic concept of the suggested method is: based on window-to-viewport transformation, resize SVG Map's viewBox to fit the viewport while preserving the visual effect. It includes three steps: 1)Calcluate SVG Map's new viewBox, and transform spatial object's coordinate pairs correspondingly. 2) Use relative coordinates in a "path". 3) Filter out unnecessary coordinates in a "path".

3.1 Calcluate SVG Map's new viewBox

Suppose that the size of the viewport on the computer screen is vheight*vwidth. If the SVG Map is unzoomable, the size of the viewBox can be set as the same size of viewport: svgheight=vheight, svgwidth=vwidth. If the SVG Map is zoomable, the size of the viewBox can be set as: svgheight=vheight*max_scale, svgwidth=vwidth*max_scale, where max_scale is current map's maximum visual scale.

Thus SVG Map's new viewBox will be set as "0 0 vheight*max_scale vwidth*max_scale".

And then we calculate the transformation expressions for the objects in this map. If P(x1,y1) is a point in the original map, its new coordinates (x1', y1') will be: x1'=int(x1*zipvalue - x*zipvalue), y1'=int(y1*zipvalue - y*zipvalue), where "x y width height"is SVG Map's old viewBox (the bounding of the map), zipvalue= vheight*max_scale/height. Function int() returns the integer portion of a number.

Here is an example. If the bounding of the map is "94928 2172873 790615 595213", P(507090.78653 2743826.33313) is a point in this map, the maximum visual scale of this map is 1000%, and the display area (viewport) is 1280*964 pixels, we carry out the calculation with the above expressions, and we get the result: the new viewBox can be set as "0 0 12800 9640", Point

P's new coordinates is (6672 9243).

3.2 Use relative coordinates in a "path"

In the initial dataset, spatial objects (such as line) are stored as a set of absolute coordinates. In fact, we can only record the start point of the spatial object as absolute coordinates, and record other points in this spatial object as relative coordinates to the last previous point, and then use lowercase commands(for example, use lowercase "l", but not uppercase "L") of attribute "d" in "path" to link these coordinates together.

3.3 Filter out unnecessary coordinates in a "path"

After the above steps, lots of strings like "l 0 0" will appear in the value of "d" attribute in "path". The reason for this is: under the current visual context, several successive points in this spatial object map to the same pixel in the output screen. We delete all this kinds of substring from "d" attribute.

After these three steps, the data amount of SVG Map will be reduced, while preserving the visual effect.

4. Some preliminary tests

In this section, we will design some preliminary tests to evaluate the suggested method.

SVG maps are chosen from different data sources randomly, and the display area (viewport) is set as 930*700 pixels, maximum visual scale of the SVG Maps as 800%. Table 1 shows the test results.

Table 1 the preliminary test results

Size of original map（B）	Precision of original coordinates	Size of zipped map （B）	Rate
58,720	0.0001	12,610	21.47%
139,363	0.001	50,691	36.37%
2,501,718	0.00000001	353,952	14.15%
2,966,041	0.0001	669,993	22.59%
3,676,149	0.0001	809,267	22.01%
4,848,807	0.00000001	674,307	13.91%
8,669,872	0.001	2,959,166	34.13%
27,868,418	0.0001	6,841,799	24.55%

From the above table, we can find that the suggested method can be used to reduce the data amount of SVG Map. We can also find that precision of original coordinates has high impact on zip rate.

5. Conclusion and Future work

This paper introduce a visual lossless data reduction method (based on window-to-viewport transformation) to reduce the data amount of SVG maps while preserving the visual effects. We also implement the method and use some sample data for method evaluation. After analyzing the experiment results, the following conclusions can be drawn:

1) The proposed visual lossless compression method can be used to reduce the data amount of SVG map while maintaining the visual effects, and then improve the transmission and rendering performance of SVG maps.

2) The SVG maps created by this method can adjust their data amount dynamically to the size of the output screen and the maximum visible scale of the dataset.

3) The proposed method can help to protect copyright of geospatial dataset. The users can only copy the map with the transformed coordinates but not the original geographic coordinates. They can only use these maps for displaying under specific visual context, but they can not use them for other processing such as spatial analysis.

Our next step is to develop some actual applications to evaluate this method. Furthermore, we would like to apply the method to other SVG applications.

Acknowledgements

We would like to give acknowledgement to the support of the project, by Science and Technology Department, Guangdong Province, for it continued financial support in 3 stages (2002B32101, 2004B32501001, 2005B30801006) of this research project.

Bibliography

[1] SVG, http://www.w3.org/TR/SVG11/

[2] Data Compression, http://en.wikipedia.org/wiki/Data_compression

[3]Lempel-Ziv-Welch, http://en.wikipedia.org/wiki/Lempel-Ziv-Welch

[4]DEFLATE, http://en.wikipedia.org/wiki/DEFLATE

[5]Apache Module mod_deflate, http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

[6] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of

points required to represent a digitized line or its caricature. TheCanadian Cartographer, 10(2):112–122, Dec. 1973.

[7] Donald, H. : Computer graphics (2. ed.), NJ : Prentice-Hall, 1994.