

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

Haosheng Huang, Yan Li

Research Group Cartography, Vienna University of Technology  
Spatial Information Research Center, South China Normal University

# Outline

- ◆ 1. Introduction
- ◆ 2. Do we need so much detail for visualization?
- ◆ 3. A visual lossless data reduction method for SVG Map
- ◆ 4. Conclusions & Future work

# 1. Introduction

- ◆ SVG is a promising standard for publishing spatial data over the internet.
- ◆ When publishing very large geospatial dataset, we have to wait for a long time for the SVG map to load.

# 1. Introduction (cont.)

- ◆ Delay of loading SVG maps:
  - Network transmission delay
    - ◆ Data amount, network transmission speed.
  - Rendering delay on the browser side
    - ◆ Performance of SVG viewers, complexity of SVG maps, data amount.
- ◆ Reducing the data amount of SVG maps will
  - shorten the network transmission time, and then improve the network transmission performance
  - Improve the rendering performance

# 1. Introduction (cont.)

## ◆ Data reduction

- Lossless data reduction
  - ◆ Gzip, LZR, Apache's mod\_deflate module
- Lossy data reduction
  - ◆ Line simplification, map generalization

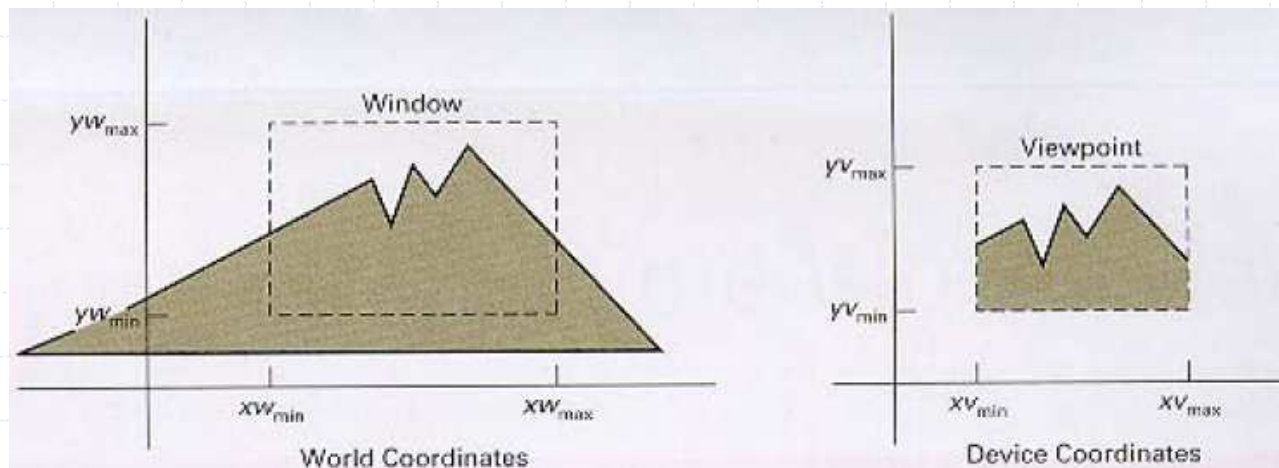
## 2. Do we need so much detail for visualization?

- ◆ Spatial datasets are always stored in DB or some initial data formats (Shape, E00, or GML).
- ◆ People always “move” the geographic coordinates directly from initial dataset to SVG document.
  - SVG’s “viewBox”: bounding of the spatial datasets (the real world area).
    - ◆ “94928 2172873 790615 595213”: map of Guangdong Prov.
- ◆ Do we need so much detail for visualization?

## 2. Do we need so much detail for visualization? (cont.)

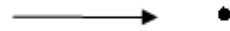
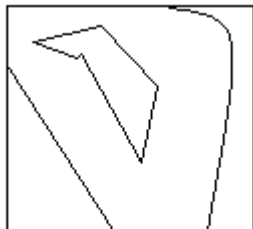
### ◆ Window-to-viewport transformation

- Window: a work-coordinate area selected for display (what is to be viewed)
- Viewport: an area on a display device to which a window is mapped (where it is to be displayed)



## 2. Do we need so much detail for visualization? (cont.)

- ◆ Pixel on the screen is the smallest unit people can distinguish.
- ◆ If: the display area (viewport) is 1280\*964 pixels
  - For map of Guangdong, all the points in area of  $(790615/1280)*(595213/964)=617*617$  will be mapped to one pixel.



11:10

617\*617 in Guangdong Map

one pixel on the screen

*We don't need so much detail for visualization.*

8 of 20

### 3. A visual lossless data reduction method for SVG Map

- ◆ Basic concepts: Based on window-to-viewport transformation, resize SVG Map's viewBox to fit the viewport while preserving the visual effect.
- ◆ Three steps:
  - Calculate SVG Map's new viewBox, and transform spatial object's coordinate pairs correspondingly.
  - Use relative coordinates in a "path"
  - Filter out unnecessary coordinates in a "path"

### 3. A visual lossless data reduction method for SVG Map(cont.)

- ◆ Step1: Calculate SVG Map's new viewBox
  - Viewport's size:  $vheight * vwidth$
  - If unzoomable, the viewBox can be set as the same size of viewport:
    - ◆  $svgheight = vheight$
    - ◆  $svgwidth = vwidth$
  - If zoomable, the viewBox can be set as
    - ◆  $svgheight = vheight * max\_scale$
    - ◆  $svgwidth = vwidth * max\_scale$
    - ◆  $max\_scale$  is current map's maximum visual scale

### 3. A visual lossless data reduction method for SVG Map(cont.)

- ◆ SVG Map's new viewBox will be set as
  - "0 0 vheight\*max\_scale vwidth\*max\_scale"
- ◆ Point P(x1, y1)'s new coordinates (x1', y1')
  - $x1' = \text{int}(x1 * \text{zipvalue} - x * \text{zipvalue})$
  - $y1' = \text{int}(y1 * \text{zipvalue} - y * \text{zipvalue})$
  - where
    - ◆ SVG Map's old viewBox: "x y width height"
    - ◆  $\text{zipvalue} = \text{vheight} * \text{max\_scale} / \text{height}$
    - ◆ function int() returns the integer portion of a number.

### 3. A visual lossless data reduction method for SVG Map(cont.)

#### ◆ Example for step1:

- SVG Map's old viewBox "94928 2172873 790615 595213"
- Point P(507090.78653 2743826.33313) in the SVG Map
- Maximum visual scale of the SVG Map: 1000%
- the display area (viewport) is 1280\*964 pixels

#### ◆ After step1:

- new viewBox "0 0 12800 9640"
- Point P's new coordinates (6672 9243)

### 3. A visual lossless data reduction method for SVG Map(cont.)

- ◆ Step2: Use relative coordinates in a "path"
  - For a spatial object, only the start point is stored as absolute coordinates, other points are stored as relative coordinates to the last previous point.
  - Use lowercase commands (lowercase "l", but not uppercase "L") to link these coordinates together.

### 3. A visual lossless data reduction method for SVG Map(cont.)

- ◆ Step3: Filter out unnecessary coordinates in a "path"
  - After step1 and step2, lots of strings like "l 0 0" can be found in a "path"
    - ◆ Reason: under the current visual context, several successive points in this spatial object are mapped to the same pixel in the screen.
  - Delete all the "l 0 0" from every "path"

### 3. A visual lossless data reduction method for SVG Map(cont.)

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

### 3. A visual lossless data reduction method for SVG Map(cont.)

#### ◆ Some preliminary tests:

- SVG maps are chosen from different data sources randomly
- the display area (viewport) is 930\*700 pixels
- Maximum visual scale of the SVG Maps: 800%

# 3. A visual lossless data reduction method for SVG Map(cont.)

## ◆ Result

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%

- The precision of original coordinates has high impact on zip rate.

# 4. Conclusion & Future work

## ◆ Conclusion:

- Based on window-to-viewport transformation, we reduce the data amount of SVG Map while preserving the visual effect.
- The method can also help to protect copyright of spatial data.

# 4. Conclusion & Future work (cont.)

## ◆ Future work

- More tests.
- Apply the method to other SVG applications.

Thank you !