VISUALIZATION TECHNIQUES OF SPATIAL DATA (SS16)
Balázs Dienes

Exercise IV Report

As last assignment to Visualization Techniques of Spatial Data, I developed a generalization method in order to automatically generalize two-dimensional building outlines. My cartographic generalization method was written in Java and includes two logical operations, namely **simplification** and **symbolization**.

I. Introduction of the input data

My input data is a text file (.txt) that includes coordinates in the following format: x_1 y_1, x_2 y_2, x_3 y_3, […], x_1 y_1. As I ran into difficulties when using coordinates exported from ArcGIS (exported coordinates contained far too many decimal digits), I decided to use simple coordinates instead.

I used "**spaces**" to split the "**x**" and "**y**" in each **coordinates**; while "**comma + space**" combinations were used to split **adjacent (!) vertices**. "**Enters**" were used to split **polygons**; consequently, coordinates in one line represent one polygon.

Each vertex has three attributes: "**x coordinate**", "**y coordinate**", and being "**deletable**". Deletableness of each vertex can be either TRUE or FALSE. In the beginning, the "deletable" attribute of all vertices are false.

II. General idea of the cartographic generalization method

The developed generalization method focuses on buildings that have irregular outlines. As deleting unnecessary coordinates was my final objective, I did not see a need to connect vertices. As a result, I did not work with real edges and real polygons, only with points.

1) Simplification

The first applied operation, simplification, focuses on distances between vertices. As buildings appear smaller when using a smaller scale, distances between vertices also decrease. To exclude vertices with unnecessarily short distances between each other, I applied thresholds. I defined thresholds as followings:

if scale is less than	1:50,000	buffer = 0 unit (no edges are removed)
if scale	1:50,000 – 1:100,000	buffer = 1 unit (short edges are removed)
if scale is greater than	1:100,000	buffer = 2 units (longer edges are removed)

Scale level can be entered when running the program: *"Please, define scale: 1:"*.(PLEASE NOTE: do not use decimal points or commas when defining scale level!)

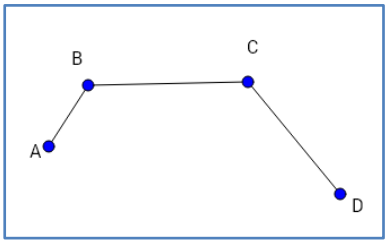

Figure 1

The operation removes vertices between which distances are shorter than the threshold defined by the scale (e.g. segment BC on figure 1). As the operation cannot delete "sides" of "polygons" but remove coordinates, it has to calculate distances between adjacent vertices as a first step. If the distance between adjacent vertices is smaller than or equals to the threshold defined; the "deletable" attribute of **one of the two points** is changed to "TRUE". As one "side" of a "polygon" is described by a start point (point B) and by an end point (point C), the operation must decide, **which one** of them should be marked as "deletable" (point B or point C). In order to solve this issue, length of the "neighboring sides" (segment AB and segment CD)

is decisive. Vertices with a **shorter neighboring side** are marked as "deletable" (point B, in case of figure 1).

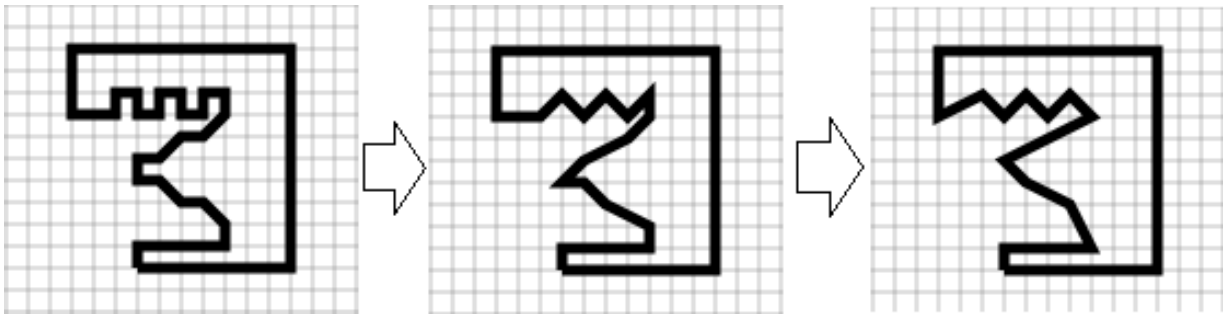

Figure 2

There are cases when too many adjacent points would be deleted, e.g. when all sides are shorter than the defined threshold, or when only two vertices would remain. I applied the "deletable" attribute instead of deleting points straight away in order to avoid deleting too many points. As the last step of simplification, the program creates a list of "non-deletable points". If less than 4 elements are in the list of "non-deletable points", no points are deleted and the second operation, symbolization, comes into force.

2) Symbolization

I applied symbolization in order to deal with buildings where my simplification would leave behind: nothing, a point, a segment, or a triangle. These results clearly do not remind on the shape of the original polygons. The applied symbolization operation replaces these "problematic" polygons (that have less than 4 "non-deletable points") to rectangles that cover them perfectly.

The program identifies the **highest and lowest "x" coordinates and "y" coordinates** of each polygon. If less than 4 "non-deletable points" would remain after simplification, the second operation, symbolization, defines four new coordinates, (x_{min}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{min}), (x_{max}, y_{max}), and clears the original vertices of the polygon (figure 3). This way, polygons are completely removed and replaced by the smallest rectangles that would cover them completely.

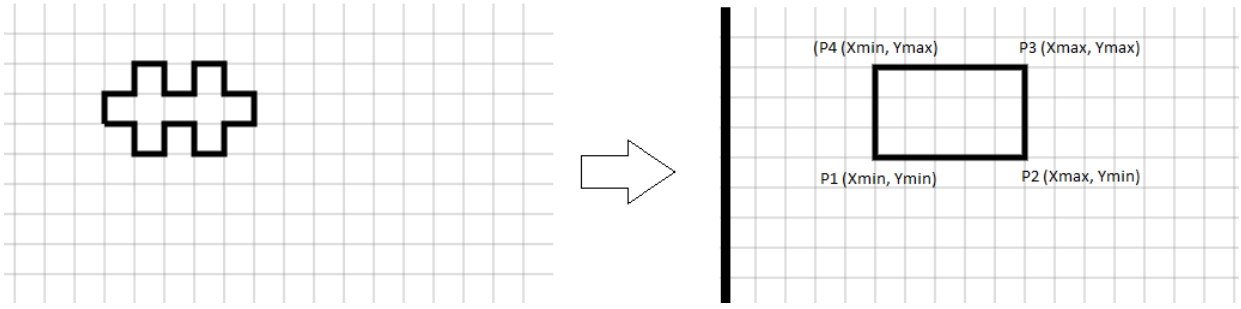

Figure 3

III. Discussion of results

I find it a major achievement that my generalization method involves the application of scale levels. When scale is small, more points are deleted; when scale is big, fewer points are deleted. The example output file was created with a scale of 1:100000 (where threshold is equal to 2 units).