Douglass-Peucker Algorithm

A simplification of polyline

Banyas Miron

July 7, 2016

Kyiv Algorithms Club

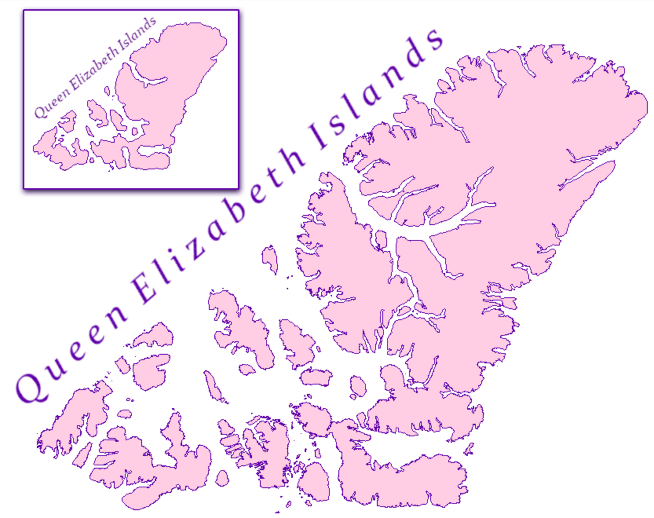
Table of contents

Introduction

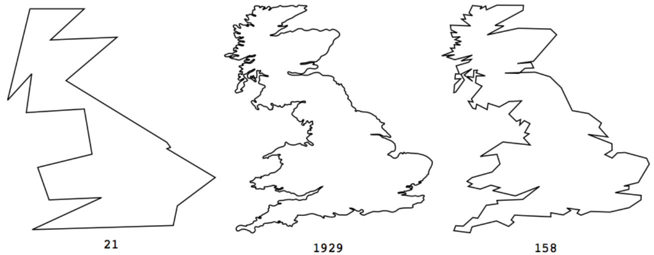
Application

- **Cartography:** map generalization.
- **Vector graphics.**
- **Robotics:** denoising of range data acquired by a rotating range scanner.

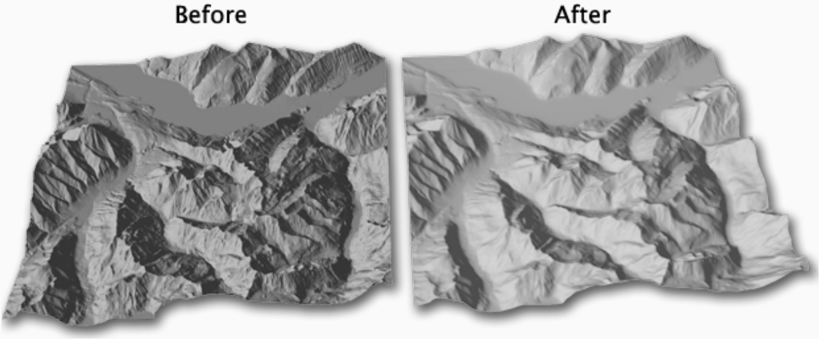
Map Generalization



Map Generalization



Map Generalization



Resources

- *C++ polyline simplification library and demo application* – `http://psimpl.sourceforge.net/`
- *JavaScript polyline simplification library* – `http://mourner.github.io/simplify-js/`
- *Java implementation of DPA* – `https://github.com/hgoebl/simplify-java`
- *Matlab implementation of DPA* – `https://www.mathworks.com/matlabcentral/fileexchange/21132-line-simplification/content/dpsimplify.m`

References

- Urs Ramer "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing*, 1(3), 244256 (1972)
- David Douglas & Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", *The Canadian Cartographer* 10(2), 112122 (1973)
- J.L.G. Pallero Robust line simplification on the plane *Computers & Geosciences* 61 (2013)

Basic Idea

Problem

Given a simple polygonal chain P (a polyline) defined by a sequence of points $p_1, p_2, ..., p_n$ in the plane.

Goal

Produce a simplified polyline $Q = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$ that for each point $p_l \in P$ the distance ρ between p_l and the segment $\overline{p_{i_j} p_{i_{j+1}}}$ $(i_j < l < i_{j+1})$ is less or equal a given tolerance ε.

Alternative Goal

Produce a simplified polyline $Q = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$ that for each point $p_l \in P$ the distance ρ between p_l and the segment $\overline{p_{i_j} p_{i_{j+1}}}$ $(i_j < l < i_{j+1})$ is minimal for a given number k.

Problem Solving

- The first and the last point (p_s and p_e) in the original polyline linked form a straight line $\overline{p_s p_e}$ that will be called a base segment.
- The point p_k $(s < k < e)$ with a maximal distance ρ_{max} to the base segment defined in the previous step is found.
- If $\rho_{max} \geq \varepsilon$ the resulting polyline is computed as a union of algorithm results on $\overline{p_s p_k}$ and $\overline{p_k p_e}$ base segment.
- Otherwise the algorithm returns base segment $\overline{p_s p_e}$.

Solving Visualization

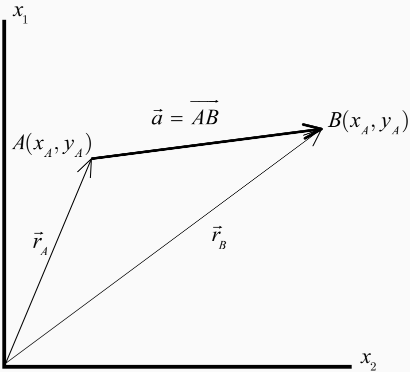
Solving of the Alternative Problem

The solving of the alternative problem is similar to the initial technique.

- At each iteration the algorithm finds the base segment with a maximal distance and divides it, as in the original algorithm. The resulting base segments are kept for the subsequent iterations.
- It is useful to use a priority queue for saving base segments.

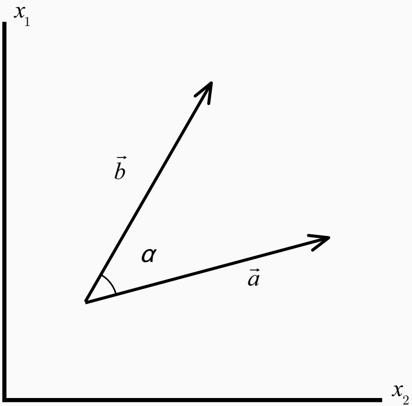
Distance to the segment

Vectors



$$
\begin{aligned}
\overrightarrow{a} &= \overrightarrow{AB} = \overrightarrow{r_B} - \overrightarrow{r_A} \\
&= (x_B - x_A, y_B - y_A) = (x_a, y_a)
\end{aligned}
$$

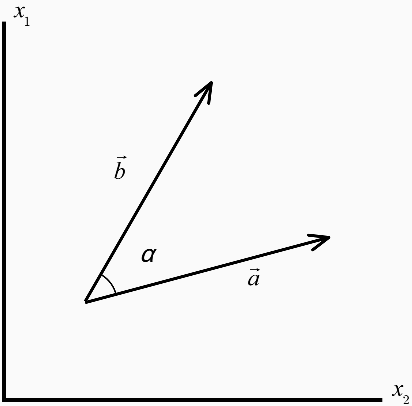
Scalar Product



$$\overrightarrow{a} \cdot \overrightarrow{b} = |\overrightarrow{a}||\overrightarrow{b}| \cdot \cos\alpha = x_a x_b + y_a y_b$$

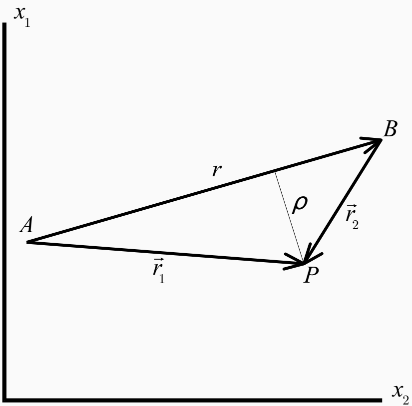
$$|\overrightarrow{a}| = \sqrt{\overrightarrow{a} \cdot \overrightarrow{a}} = \sqrt{x_a^2 + y_a^2}$$

Vector Product



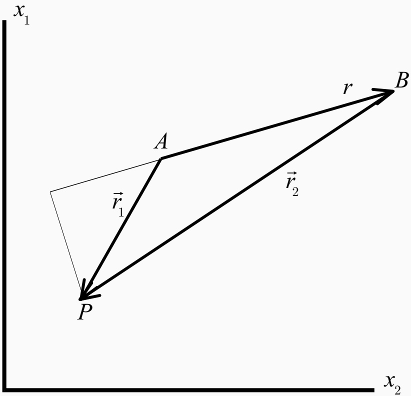
$$\overrightarrow{a} \times \overrightarrow{b} = |\overrightarrow{a}||\overrightarrow{b}| \cdot \sin \alpha$$
$$= x_a y_b - y_a x_b$$

Distance



$$\rho = |\overrightarrow{r_1}| \sin\alpha = |\overrightarrow{r_1}| \frac{|r \times r_1|}{|r|}$$

Improving



- if $r_2 >= \sqrt{r_1^2 + r^2}$ then $\rho = r_1$
- if $r_1 >= \sqrt{r_2^2 + r^2}$ then $\rho = r_2$
- else $\rho = |\overrightarrow{r_1}| \frac{|r \times r_1|}{|r|}$

Algorithm features

Complexity

The expected complexity of this algorithm can be described by the linear recurrence

$$T(n) = 2T(n/2) + O(n)$$

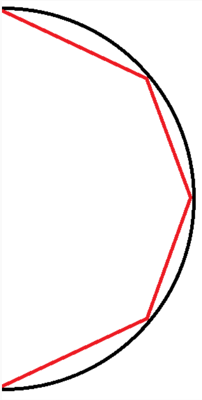
Using the **Master Theorem** we can find that the complexity is

$$T(n) \in \Theta(n \log(n))$$

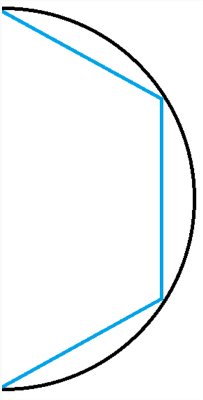
The **worst-case** complexity is $O(n^2)$

Optimality

The Douglas-Peucker algorithm is not optimal with respect to the number of points.



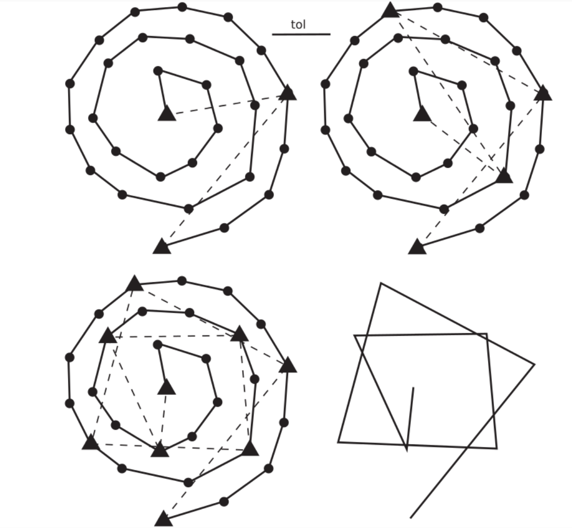
The answer of Douglas-Peucker Algorithm



The optimal answer with respect to the number of points

Robust

The Douglas-Peucker technique would have to be considered a non-robust algorithm since the resulting polyline may contain self-intersections.



Robust algorithm

Non-Recursive Non-Robust Variant

- The last point (N in the original line) added to the output line, is joined to vertex $N+2$ to generate the base segment. Then the distance between vertex $N+1$ and the base segment is computed and compared with the established tolerance.
- If the computed distance $\rho < \varepsilon$, a new base segment will be created between vertices N and $N+3$ and the distances between the base segment and all the intermediate points between N and $N+3$ will be computed again. As long as the greatest of these distances does not exceed the predefined tolerance, new base segments will be created between points N and $N+4 ... N+n$ from the original line. In the extreme case the last point from the original line could be reached.
- When a base segment $N/N+k$ is found for which the distance to the farthest intermediate point is greater than the tolerance, the vertex $N+k-1$ will be added to the output line. Hence the tolerance of all vertices between N and $N+k-1$ will be ensured since the base segment $N/N+k-1$ was checked at the previous step.

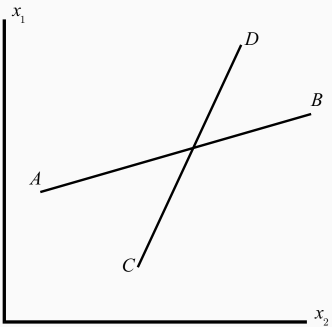
Visualization

Robust Variant

The robust modification of the non-recursive Douglas-Peucker technique involves the detection of the possible intersections between:

- The new segments of the output line and the non-processed segments of the original line.
- The new segments of the output line and the previous segments of the same output line.

Segment Intersection



$$\overrightarrow{AB} \times \overrightarrow{AC} \cdot \overrightarrow{AB} \times \overrightarrow{AD} \leq 0$$

$$\text{and } \overrightarrow{CD} \times \overrightarrow{CA} \cdot \overrightarrow{CD} \times \overrightarrow{CB} \leq 0$$

$$\text{and } \overline{AB} \subset \overline{CD} \text{ or } \overline{CD} \subset \overline{AB}$$

Visualization