Computers & Geosciences 83 (2015) 146–152

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Robust line simplification on the surface of the sphere

J.L.G. Pallero

ETSI en Topografía, Geodesia y Cartografía, Universidad Politécnica de Madrid, Avenida del Mediterráneo, km 7.0, 28031 Madrid, Spain

ARTICLE INFO

Article history:
Received 20 May 2014
Received in revised form
28 April 2015
Accepted 20 July 2015
Available online 21 July 2015

Keywords:
Douglas–Peucker algorithm
Polyline simplification
Map generalization
Cartography
OpenMP
Computing

ABSTRACT

Polyline simplification is an important task in map generalization. Several solutions have been developed in order to perform this task automatically, but the vast majority of them consider that the objective line to simplify is lying on the plane. One of the most widely used methods is the so-called Douglas–Peucker algorithm, which is fast and in most cases produces good results. However, the Douglas–Peucker method was defined in its original form in order to be applied to polylines contained in the Euclidean 2D space, and it can lead to inconsistent results such as self-intersections. In this work, a robust (results without self-intersections) variation of the Douglas–Peucker for polylines on the surface of the sphere is presented. It produces correct results regardless of the morphology of the original line and the tolerance parameter size.

The algorithm is coded in standard C99 and it can be compiled for serial or parallel execution via OpenMP. Both, the algorithm itself and a program implementing it are distributed as free software. The solution validity was tested using the GSHHG geography database, which can be obtained free through the Web. Results about output accuracy, execution speed, and parallel implementation scalability are presented.

1. Introduction

Polyline simplification is an important task in map generalization. By using the adequate method, an accurate representation of the original elements should be obtained, while all redundant information in the original lines should be eliminated in keeping with the new scale. The main features in a map subject to be generalized by these techniques are shorelines, lake perimeters, rivers, roads, contour lines, etc. Line simplification is used not only in classical paper map series, but also in web services related to geographic information systems (Cecconi et al., 2002; Harrower and Bloch, 2006).

One of the most widely used methods in practice is the so-called Douglas–Peucker algorithm (Douglas and Peucker, 1973), mainly because of its ease of implementation, its high execution speed, and the results quality in most cases. By retaining a set of points from the original polyline, this method discards those out of tolerance from the generalized line. The main limitation of the Douglas–Peucker technique is the possibility for the resulting polyline to contain self-intersections (Saalfeld, 1999); hence, that technique would have to be considered non-robust. In addition, the original algorithm was defined only for polylines in the Euclidean 2D space, so it cannot be applied (in a consistent geometric sense) to data on the surface of the sphere, considering it as the first approximation to the Earth shape.

The original Douglas–Peucker algorithm could be applied directly to the polylines on the sphere using geographic coordinates, but the results would only be *correct* (where *correct* must be understood as *below a level of admissible error*) for very small areas where meridian arcs have approximately the same dimensions as parallel arcs. Another solution to the problem involves the original data projection onto the plane using a cartographic projection, then apply the Douglas–Peucker method to the resulting polyline, and finally reproject the resulting data onto the sphere. This approximation has two problems: (i) there is a computational time spent in the projection steps, which could be critic in some applications or when low-power hardware were used, and (ii) there is no cartographic projection in which the Euclidean geometry can be used in a general case (distance computations in arbitrary directions, intersection checking between segments, etc.; see for example Bugayevskiy and Snyder, 1995). The problem (ii) is the main one; the cartographic projection approximation can be more suitable than working directly in geographical coordinates for small areas and relative low tolerance values, but it can still suffer inaccuracies working with polylines occupying large areas, where the used projection deformation can vary in high values between different zones.

In Burt (1989), a spherical version of the Douglas–Peucker

E-mail address: jlg.pallero@upm.es

http://dx.doi.org/10.1016/j.cageo.2015.07.011
0098-3004/

method is described. By using rotations in the Euclidean 3D space, the original Douglas–Peucker algorithm is adapted to work with polylines on the surface of a sphere, so the distances are considered in the correct way as great circle arcs. However, Burt's approximation has the same weak point as the original algorithm, i.e., it can lead to non-robust results (self-intersections) for some original polyline configuration or/and due to high tolerance parameter values.

In Pallero (2013), a new robust Douglas–Peucker-based algorithm for line simplification on the plane is developed. Based on a sequence of segment intersection tests, robust results are achieved whatever the shape of polyline and the working tolerance are.

In this paper, the method presented in Pallero (2013) is adapted to work on the surface of the sphere using some ideas from Burt (1989). The new algorithm is explained in detail and its implementation in C is put under a free software license. In addition, the feasibility of its parallel execution is analysed using both standard x86 (64 bits) PC microprocessors and new ARM based low power boards, performing a set of tests employing data from the GSHHG geography database (Wessel and Smith, 1996). Finally, the main conclusions about performance and algorithm quality are drawn.

It should be noted that the algorithm, as the original Douglas–Peucker one, works only with individual polylines, so when a set of lines (e.g. contour lines) is generalized, intersections could appear between them.

2. Generic robust algorithm on the plane

In Pallero (2013), a Douglas–Peucker-based robust method for line simplification on the plane is described. Given a polyline composed of a set of points, and after a length tolerance for point rejection is established, the algorithm comprises the following steps:

1. Taking the last point (index number L in the original line) added to the output line (in the first step of the algorithm it will be the starting point), together with the vertex $L+2$, forms the *base segment*. Then, the distance between the vertex $L+1$ and the base segment is computed, and is compared with the established tolerance.
2. If the computed distance does not exceed the tolerance, a new base segment is created between the vertices L and $L+3$, and the distances between it and all the intermediate points from L to $L+3$ are computed again. While the greatest of these distances does not exceed the predefined tolerance, new base segments will be created between points L and $L+4\cdots L+n$ from the original line. In the limit, we could reach the last point from the original line.
3. When a base segment $L/L+k$ is found for which the distance to the farthest intermediate point is greater than the tolerance, the vertex $L+k-1$ will be a candidate to add to the output line. All vertices between L and $L+k-1$ are guaranteed in tolerance because the base segment $L/L+k-1$ was checked in the previous step.
4. In order to prevent future self-intersections, the possible intersections between the base segment $L/L+k-1$ and the remaining segments of the original polyline from $L+k-1$ to the last point are checked. If an intersection is found, the base segment is defined as $L/L+k-2$ and the segment intersections are tested again until no crosses are found. Then, the base segment can be defined generically as $L/L+k-1-i$ after this step, where i is 0 if no intersections were found. In the praxis, it may be expected that the possible intersections be located in close proximity to the base segment, so, in order to save processing time, the intersection checking could be not extended to the original line end, but only for a determined number of segments from point $L+k-1$ onwards. If this check limitation is used, it must be taken into account that the algorithm loses its robust character, so a previous study about the number of segments to check is needed.
5. In this step, the possible intersections between the base segment $L/L+k-1-i$ and the previous simplified polyline computed segments are checked. If an intersection is detected, the base segment is defined as $L/L+k-1-i-1$ and the intersection test is repeated until no intersections are found. As in the previous case, the segment number for checking can be fixed in order to save computational time. Finally, the base segment is defined as $L/L+k-1-i-j$, where j is 0 if no intersections were found. In the worst case, the base segment after steps 4 and 5 could be defined as the segment $L/L+1$ in the original polyline.
6. We go back to the first step. The vertex $L+k-1-i-j$ is added to the simplified polyline and is considered now the initial vertex for the next base segment.
7. The algorithm ends when the last base segment has as final vertex the last point from the original polyline, all the intermediate vertices are in tolerance, and no intersection exists between this last base segment and any of the previous segments forming the simplified polyline.

Suppose the N vertices of the original polyline stored in an array of indices from 0 to $N-1$. Fig. 1 shows the flowchart describing the explained algorithm, where the simplified polyline vertices are saved after the robust process.

3. Robust algorithm on the surface of the sphere

3.1. Non-robust part of the algorithm

In order to modify the Douglas–Peucker-based algorithm to work with data on the surface of the sphere, the key task consist in computing the distance between the base arc[1] and the points between its extremes using the spherical geometry.

The defined task could be carried out via spherical trigonometry as we have in each case the vertex coordinates defining a spherical triangle (the extremes of the base arc and the points to check), so it is easy to define a formulation in order to compute the required distance from the test points to the base arc. However, the high number of trigonometric functions involved in this computation will penalize the execution speed of the whole algorithm, so some work around in order to mitigate that this effect is mandatory. In Burt (1989), a solution based on rotations in the Euclidean 3D space is proposed.

Suppose a base arc $\widehat{AB}$ (henceforth, the working sphere will be the unit sphere) and a test point P, all of them with its coordinates (φ_A, λ_A), (φ_B, λ_B), and (φ_P, λ_P), where φ stands for latitude and λ for longitude. In order to compute the spherical distance between the point P and the arc $\widehat{AB}$ without any spherical triangle resolution (except for special cases, which will be explained later), and inspired in Burt (1989), the next algorithm is proposed:

1. Rotate the original cartesian geocentric coordinate system in a such way that the base arc $\widehat{AB}$ becomes $\widehat{A'B'}$, contained on the new system Equator, with the vertex A' lying on the origin $(\varphi_{A'}=0, \lambda_{A'}=0)$.

[1] Henceforth, the term *base arc* will be used instead of *base segment*, as the algorithm described is the one on the surface of the sphere.