Testing Shortcuts to Maintain Simplicity in Subdivision Simplification

Craig Falls Yuanxin Liu Jack Snoeyink
University of North Carolina—Chapel Hill
Chapel Hill, NC

Diane Souvaine
Tufts University
Medford, MA

Abstract

Cartographers collect more data than they need, and so must simplify coastlines, boundaries, and other linear features to display a map at a given scale. Many simplification methods, however, can introduce intersections that were not originally present, corrupting the features.

Kulik suggests a simple shortcut operation for polygonal lines: remove a point p_i and connect its former neighbors p_{i-1} and p_{i+1} directly, but only if the triangle $\triangle p_{i-1}p_ip_{i+1}$ is empty of other points. We show geodesic triangulations support shortcut operations and triangle tests in $\mathcal{O}(\log^2 n)$ time for connected subdivisions of size n. This can be integrated into simplification methods that support cartographic preferences so that they can also avoid self-intersection.

1 Introduction

There are many simplification algorithms in the literature of geographic information systems (GIS) [1], computer vision [11], and computational geometry. The practical literature typically presents heuristics that run in linear or near linear time, while the computational geometry literature tends toward optimal simplification under various criteria (e.g. minimum error for a given number of segments, or minimum number of segments for a given error) using algorithms that run in quadratic to cubic time [9]. These algorithms are compared by Heckbert and Garland [8].

In cartographic applications, self-intersections usually indicate errors in digitization or processing [13]. Nevertheless, most algorithms simplify collections of polygonal lines by considering each in isolation, and run the risk of introducing intersections and topological changes in the data. The cartographers' favorite algorithm [4] can even introduce self-intersections within one polygonal line.

A few simplification algorithms have been developed in computational geometry that preserve map topology. De Berg *et al.* [3] give a method that simplifies an n-point polygonal line without passing over itself or over m specified points in $\mathcal{O}(n(n+m)\log n)$ time. Estkowski and Mitchell [5] give a heuristic for simplifying parallel lines, such as elevation contours, in quadratic time. Others have used the Voronoi diagram to identify regions in which it is safe to perform simplification without creating intersections [12, 14].

Unfortunately, Estkowski and Mitchell [5] argue that requiring simplicity in subdivision simplification using the original points makes the problem MIN PB-complete and hard to approximate within a factor of $n^{1/5-\delta}$. Even for a single path that winds through a (non simply-connected) region, it is NP-hard to determine if there exists a polygonal line of the same homotopy class that has at most k line segments [7].

In Section 2.1, we describe the shortcut operation and triangle test suggested by Lars Kulik [10]. In Section 2.2 we integrate the shortcut operation with line simplification and prove that the homotopy class remains unchanged in Section 2.3. Finally, in Section 3 we show how a balanced geodesic triangulation can perform each triangle test and valid shortcutting operation in $\mathcal{O}(\log^2 n)$ time, after $\mathcal{O}(n \log n)$ preprocessing using linear space.

2 Definitions and Preliminaries

Although we are primarily concerned with line simplification, our solution will handle connected subdivisions with high degree vertices. Thus, we define the problem generally, using the terminology of *embedded graphs*.

Let $\mathcal{S}$ be the subdivision of the plane defined by a set of line segments. Each vertex p is the endpoint of some lines L_p. The endpoints of L_p other than p itself are called the neighbors of p and $d(p) = |L_p|$ is called the *degree of* p. We say that two points p and q in a can *see* each other if the open line segment $\overline{pq}$ does not intersect any other line segments of $\mathcal{S}$.

The common case for line simplification is a sequence of vertices $P = \{p_1, p_2, \ldots, p_k\}$ in which each pair of adjacent vertices are joined by an edge and all vertices except the first and last have degree 2. The vertices of P are called a *polygonal line*.

2.1 Shortcut Operation and Triangle Test

Our *shortcut operation* removes a point p_i from the subdivision. Whether p_i can be removed depends on its degree $d(p_i)$. If $d(p_i) = 0$ then p_i is deleted unconditionally. If $d(p_i) = 1$ then it is still deleted unconditionally, along with its incident line segment. If $d(p_i) = 2$ then

p_i is deleted and a line segment is added from p_{i-1} to p_{i+1}, as in Figure 1. If $d(p_i) > 2$, no shortcutting operation can be performed without first splitting the point or deleting incident edges, as shown in Figure 2.

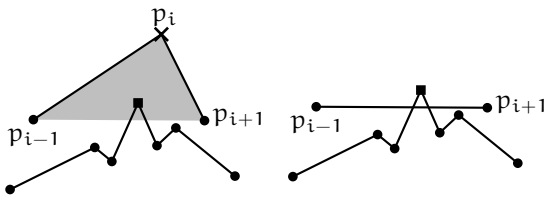

Figure 1: The triangle test and shortcut operation we support. In this case, the update would not be allowed because the triangle is not empty.

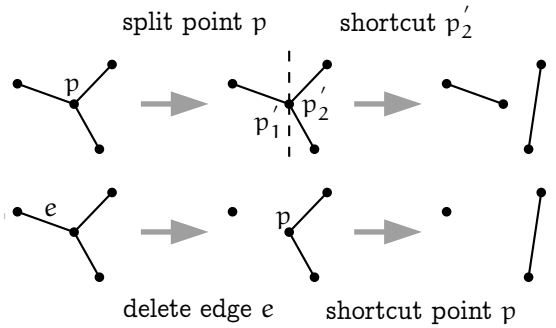

Figure 2: A shortcut operation on a point p is defined only when $d(p) \leq 2$. When $d(p) > 2$ (e.g., p_i is a *node* in an ARC/INFO planar map structure), its degree can be reduced by deleting an incident edge e or splitting p into two very close points p_1' and p_2' and combining a pair of faces about p into a single face.

When $d(p_i) = 2$, the shortcut operation can introduce intersections, as in Figure 1. In order to prevent such intersections, we shortcut p_i only when it passes a *triangle test*—i.e. the triangle $\triangle p_{i-1}p_ip_{i+1}$ contains no vertex.

2.2 Line Simplification

The shortcut operation can be incorporated into several approaches to line simplification. One approach is to assign a priority to vertices by looking at nearby angles and vertices, so that updating priorities takes constant time. Example measures of importance are the angle deviation from 180^o, triangle area, edge length, and Douglas-Peucker recursion depth. A priority queue is maintained in $\mathcal{O}(\log n)$ time so that the least important vertex is shortcut provided it passes the triangle test. If the triangle test fails then the vertex is simply removed from the queue. This operation is iterated until a target number of vertices remain or a target priority is reached. By Theorem 2, this process requires $\mathcal{O}(n \log^2 n)$ time.

2.3 Homotopy

A polygonal line $P = \{p_1, \ldots, p_k\}$ defines a class of paths that are deformable to P without intersecting other vertices or edges L of the planar subdivision $\mathcal{S} = \mathbb{R} \setminus L$. Specifically, a homotopy of paths in $\mathcal{S}$ is a family of paths $P_t : [0,1] \to \mathcal{S}$ such that $P_t(0) = p_1$ and $P_t(1) = p_k$ are fixed, and the map $F : [0,1] \times [0,1] \to \mathcal{S}$ given by $F(s,t) = P_t(s)$ is continuous. The paths P_0 and P_1 are said to be *homotopic*. Homotopy defines an equivalence relation on paths in a topological space. The equivalence class of a path P under this relation is called the *homotopy class* of P.

When we perform line simplification, it is desirable to choose a representation in the same homotopy class to preserve the topological relationships with nearby features. Choosing a representation from outside the homotopy class can move a house to the wrong side of a road, or a city to the wet side of a coastline.

Theorem 1 *In a planar subdivision with segments L that do not cross, applying shortcuts will never create a crossing. Applying shortcuts to the degree 2 vertices of any polygonal line preserves the homotopy in the space of $\mathbb{R} \setminus L$.*

Proof. For a degree 2 point p_i on the polygonal line $P_0 = \{p_1, \ldots, p_k\}$, we first test if the triangle $\triangle p_{i-1}p_ip_{i+1}$ contains any points. If $\triangle p_{i-1}p_ip_{i+1}$ contains a point, no shortcut is performed and homotopy class is preserved. If $\triangle p_{i-1}p_ip_{i+1}$ is empty of points, we replace edges $\overline{p_{i-1}p}$ and $\overline{pp_{i+1}}$ with a new edge $\overline{p_{i-1}p_{i+1}}$, producing a new polygonal line $P_1 = \{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_k\}$. The path P_0 can be continuously deformed to P_1 if $\triangle p_{i-1}p_ip_{i+1}$ is empty (of both points and line segments).

It remains to prove that in a planar subdivision containing edges $\overline{p_{i-1}p}$ and $\overline{pp_{i+1}}$, if the triangle $\triangle p_{i-1}pp_{i+1}$ contains no points, then it contains no (portions of) line segments.

Suppose (for contradiction) that triangle $\triangle p_{i-1}pp_{i+1}$ contains no points but its interior is intersected by some line segment l. We know by the triangle test that both endpoints of l are outside of $\triangle p_{i-1}pp_{i+1}$. Therefore, l must intersect two sides of $\triangle p_{i-1}pp_{i+1}$. By the pigeon hole principle, one of those sides must be $\overline{p_{i-1}p}$ or $\overline{pp_{i+1}}$, which is a contradiction, because no segments in the subdivision intersect. □

3 Shortcut Operations and Triangle Tests in Geodesic Triangulations

A linear-time triangle test is trivial—simply test whether any vertex of G lies inside the given triangle.

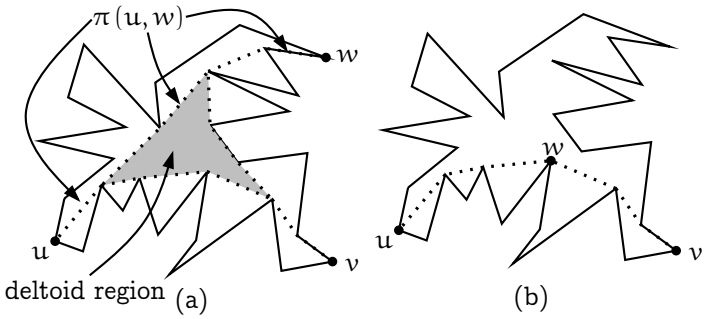

Figure 3: A geodesic triangle $\tilde{\triangle}uvw$ is either (a) a simple polygon made up of three concave chains and three polygonal chains emanating from the three vertices where the concave chains are joined, or (b) just a path with an empty deltoid region and one empty tail.

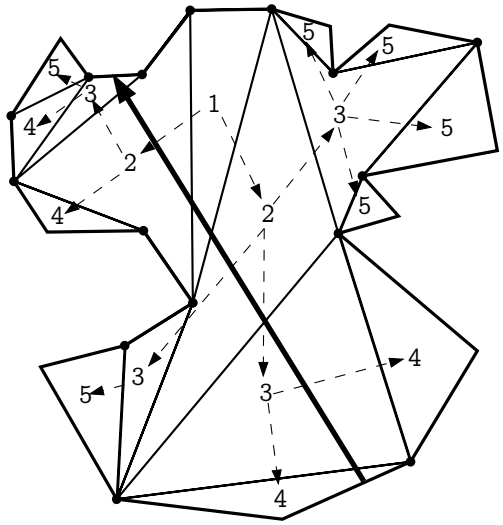

Figure 4: A geodesic triangulation of a polygon and its dual tree T (dashed arrows). The geodesic triangles intersected by a ray $\vec{r}$ (heavy arrow) form a path in T.

This can be accelerated in practice by sorting by x coordinate or using buckets to test only those features of G that overlap the triangle in x and y coordinates. In this section we describe how a balanced geodesic triangulation of a planar subdivision can be maintained under shortcut and used to answer triangle tests in $\mathcal{O}\big(\log^2 n\big)$ time per operation.

For an efficient dynamic data structure, no vertex or edge should participate in too many auxiliary segments or pointers, or else its removal will cause too much rebuilding. We conjecture that this cannot be guaranteed by a constrained triangulation or a trapezoidation of G. Instead, we use a geodesic triangulation.

For a simple polygon P, the *geodesic path* $\pi(p, q)$ is the shortest path joining points p and q that does not go outside of P. For three vertices u, v, and w of P, the *geodesic triangle* $\tilde{\triangle}uvw$ is the union of the paths $\pi(u, v)$, $\pi(v, w)$, and $\pi(w, u)$. See Figure 3.

A *geodesic triangulation of a simple polygon* P is a decomposition of P's interior into geodesic triangles whose boundaries may overlap but do not cross. See Figure 4.

Like a triangulation of a convex polygon, a geodesic triangulation induces a degree-3 dual tree T, where each node in T corresponds to a geodesic triangle and there is an edge between the node $\tilde{\triangle}uvw$ and the node $\tilde{\triangle}xyz$ if they share two of their vertices (e.g., if $x = v$ and $z = w$), as shown in Figure 4. The nodes of T corresponding to the geodesic triangles whose boundaries are intersected by some ray in P will always form a path in T. A geodesic triangulation is said to be *balanced* if the diameter of T is $\mathcal{O}\big(\log |T|\big)$.

Theorem 2 proves that the triangle test and shortcut operation suggested by Lars Kulik can be performed in $\mathcal{O}\big(\log^2 n\big)$ time using $\mathcal{O}(n)$ space. The theorem follows from the fact that geodesic triangulations support the following in the same time and space:

Shooting rays Chazelle *et al.* [2] describe a strategy for ray-shooting in balanced geodesic triangulations—first locate the geodesic triangle whose interior contains the starting point for the query ray $\vec{r}$ and iteratively traverse geodesic triangles along the direction $\vec{r}$ until reaching an edge. This strategy crosses at most $\mathcal{O}\big(\log n\big)$ geodesic triangles. Because the deltoid region of each triangle is represented by three convex hulls, the ray-shooting query for any one triangle takes $\mathcal{O}\big(\log n\big)$ time.

Inserting and deleting edges Geodesic triangulation can be maintained under the operations of inserting and deleting vertices and edges in $\mathcal{O}\big(\log^2 n\big)$ time per operation. In order to keep the geodesic triangulation balanced, it is necessary to perform edge flipping operations. Edge flipping in the triangulation corresponds to rebalancing operations in its dual tree T. Goodrich and Tamassia [6] show how to use red-black tree balancing on T to maintain balance in the geodesic triangulation.

Theorem 2 *Given a connected subdivision $\mathcal{S}$ of the plane, a balanced geodesic triangulation can dynamically support the triangle test and shortcut operation for a point p_i in $\mathcal{O}\big(\log^2 n\big)$ time using $\mathcal{O}(n)$ space.*

Proof. Each face of a connected subdivision $\mathcal{S}$ is a simple polygon. Let P be the polygon in which the triangle $\triangle p_{i-1}p_ip_{i+1}$ lies. Because $\mathcal{S}$ is connected, triangle $\triangle p_{i-1}p_ip_{i+1}$ is empty if and only p_{i-1} can see p_{i+1} in P. Triangle tests can be performed by shooting a ray from p_{i-1} to p_{i+1}.

Performing a shortcut operation in a geodesic tri-

angulation consists of removing the edges $\overline{p_{i-1}p}$ and $\overline{pp_{i+1}}$ and adding the edge $p_{i-1}p_{i+1}$. Geodesic triangulations support each of these operations in $\mathcal{O}(\log^2 n)$ time [6]. □

4 Conclusions and Open Problems

We have shown that balanced geodesic triangulations support the shortcut operation suggested by Lars Kulik and that such an operation can be used to simplify lines, for example contours in maps. The following open problems remain:

- Can we get the query and update times down to $\mathcal{O}(\log n)$? Dynamic point location algorithms have $\mathcal{O}(\log^2 n)$ time complexity, so it would be notable if this special case of dynamic point location is easier.
- Can the same $\mathcal{O}(\log^2 n)$ time be achieved for queries and updates using a simpler structure? Maintaining geodesic triangles requires a primary, secondary, and tertiary data structure that is unlikely to be implemented in GIS software. This question can be answered positively if there is a simple efficient method for maintaining a monotone decomposition—triangle queries can be implemented using an Overmars/van Leeuwen convex hull structure [15].
- Goodrich and Tamassia [6] show only how to maintain a geodesic triangulation under the assumption of a connected subdivision. GIS data (and in particular isolines) are often disconnected—can this problem be circumvented? If one can maintain a connected subdivision by adding fake edges while maintaining a bounded degree on vertices, it may be possible to generalize many results for connected subdivisions to any subdivision.
- Which simplification criteria (*i.e.* importance measures) work well in practice?

Acknowledgments

We thank Lars Kulik for posing this question. This research has been partially supported by NSF ITR grant 0086013, and NSF CCF grant 0431027. We thank Ryan Coleman for stimulating discussions.

References

[1] Barbara Buttenfield and Robert McMaster, editors. *Map Generalization: Making Rules for Knowledge Representation*. Wiley, 1991.

[2] Bernard Chazelle, H. Edelsbrunner, M. Grigni, Leonidas J. Guibas, J. Hershberger, Micha Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12:54–68, 1994.

[3] M. de Berg, M. van Kreveld, and S. Schirra. A new approach to subdivision simplification. In *Proc. 12th Internat. Sympos. Comput.-Assist. Cartog.*, pages 79–88, 1995.

[4] 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. *Canadian Cartographer*, 10(2):112–122, December 1973.

[5] Regina Estkowski and Joseph S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *Proc. 17th ACM SCG*, pages 40–49, 2001.

[6] M. T. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *J. Algorithms*, 23:51–73, 1997.

[7] Leonidas J. Guibas, John E. Hershberger, Joseph S. B. Mitchell, and Jack Scott Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *Int. J. Comp. Geom. Appl.*, 3(4):383–415, 1993.

[8] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, 97. URL `citeseer.ist.psu.edu/heckbert97survey.html`.

[9] Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Comput. Vision Graph. Image Process.*, 36:31–41, 1986.

[10] L. Kulik, M. Duckham, and M. Egenhofer. Ontology-driven map generalization. *Journal of Visual Languages and Computing*, 16:245–267, 2005.

[11] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning methodologies – a comprehensive survey. *IEEE Pat. Anal. Mach. Int.*, 14(9):869–885, 1992.

[12] Andrea Mantler and Jack Snoeyink. Safe sets for line simplification. Presented at CGC Workshop, 2000.

[13] Robert Brainerd McMaster. Automated line generalization. *Cartographica*, 24(2):74–111, 1987.

[14] Nabil H. Mustafa, Eleftherios Koutsofios, Shankar Krishnan, and Suresh Venkatasubramanian. Hardware-assisted view-dependent map simplification. In *Proc. 17th ACM SCG*, pages 50–59, 2001.

[15] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23: 166–204, 1981.