Distance Preserving Terrain Simplification — An Experimental Study

Boaz Ben-Moshe[1] Matthew J. Katz[2] Igor Zaslavsky[2]

[1]Department of Computer Science
College of Judea and Samaria, Ariel 44837, Israel
benmo@yosh.ac.il
[2]Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
{matya,igorz}@cs.bgu.ac.il

Abstract

The terrain surface simplification problem has been studied extensively, as it has important applications in geographic information systems and computer graphics. The goal is to obtain a new surface that is combinatorially as simple as possible, while maintaining a prescribed degree of similarity with the original input surface. Generally, the approximation error is measured with respect to distance (e.g., Hausdorff) from the original or with respect to visual similarity. In this paper, we propose new algorithms for simplifying terrain surfaces, designed specifically for a new measure of quality based on preserving inter-point (geodesic) distances. We are motivated by various geographic information system and mapping applications.

We have implemented the suggested algorithms and give experimental evidence of their effectiveness in simplifying terrains according to the suggested measure of quality. We experimentally compare their performance with that of another leading simplification method.

1 Introduction

There are numerous papers dealing with terrain and surface simplification. A terrain can be modeled as a triangulation (e.g., of a rectangular region R), with a height (z-coordinate) assigned to each triangle vertex. Terrain models are commonly used to represent the surface of the earth.

Since terrain models can be huge, in particular when the resolution is high, it is often necessary to simplify them prior to using them for analysis or visualization. Methods for terrain simplification have been devised that transform a detailed terrain into a less detailed terrain, having fewer triangles, in such a way that the simplified terrain is "similar" to the original terrain in some sense. There are many possible ways to measure the degree of similarity between the original and simplified terrains; some are exact (e.g., specifying an exact numerical error tolerance ϵ such that the simplified terrain must lie within vertical distance ϵ of the original, at every point $(x, y) \in R$), while other methods rely on qualitative notions of similarity (e.g., based on human perception of similarity).

In this paper we propose a new way to measure quality of simplification that is especially appropriate for applications that compute and use geodesic distances between terrain points. Informally, a simplification (of the desired size) is considered "good" by this measure if for any random set $\mathcal{X}$ of pairs of points from the underlying rectangular region R, most of the distance information is preserved with respect to $\mathcal{X}$. That is, for most pairs $\{p, q\} \in \mathcal{X}$, the distance between p and q on the simplified terrain is not significantly different from the corresponding distance on the original terrain.

This criterion is quite different from the commonly used criteria, since, for example, we do not care if a very high and detailed mountain is replaced by a much lower and less detailed mountain, as long as this change is not expected to have a significant effect on the distances computed for a random set $\mathcal{X}$ of pairs of points. Notice that according to other criteria, e.g., the maximum vertical distance between the terrains, the simplified terrain in the above example can deviate significantly from the original one.

The distance-based quality of simplification measure suggested here is motivated by GIS and mapping applications, where often a requirement for dramatic simplification and a requirement for realistic distances come together. GPS devices are a typical example. We were also motivated by the paper by Gudmundsson et al. [8], who consider the problem that is studied in this paper for polygonal paths, rather than for terrains. Of course the nature of the solutions is different. While there exist (polynomial-time) exact and approximation algorithms for the former problem, we only seek good heuristics for the latter problem.

The rest of this paper is organized as follows. We first mention and discuss some related work. Then, in Section 2 we formally define the (geodesic) distance-based quality of simplification measure used in this paper to determine the similarity between the simplified and original terrains. In this section we also describe how in practice we approximate distances on a terrain, by transforming the corresponding triangulation into a denser graph.

In Section 3, we describe two general methods for terrain simplification. The first is the well-known elimination method, and the second is a meta method that applies the former method in a sophisticated manner. These methods are not designed for a specific similarity measure. The meta method first divides the input terrain into rectangular pieces, each with more or less the same number of vertices. It then assigns to each piece a simplification ratio based on the nature of the piece and the similarity measure in use. Next, it applies the elimination method to each of the pieces separately, and combines the simplified pieces into a single simplified terrain of the desired

size.

In Section 4, we devise two algorithms based on the elimination method — PLD and VP. The former attempts to preserve local distances, while the latter attempts to minimize the volume of the set of all points that lie between the original and simplified terrains. Next, we employ the meta method to obtain variants of these two algorithms, called PLD' and VP', respectively. For this, we must specify how the approximation ratio is determined given a rectangular piece. We have developed a software package, named DPTS, that includes implementations of the above four algorithms. Our package uses the Computational Geometry Algorithms Library (CGAL [5]). Figure 2 depicts several terrains that were produced by the package.

In Section 5 we report on some of our experiments with algorithms PLD, VP, PLD', VP', as well as comparisons with QSlim, a well-known quadric-based surface simplification package by Garland (see [7]). Our main conclusions are that (i) VP is significantly better than PLD, (ii) a dramatic improvement is achieved by replacing PLD by PLD', and (iii) VP' seems to be the most appropriate among the five algorithms, with respect to the distance-based quality of simplification measure.

Related work. Extensive work has been done on many aspects of terrain approximation; see Heckbert and Garland [9] for a survey. Most papers dealing with terrain simplification consider error norms such as maximum vertical distance, Hausdorff distance, etc. Ben-Moshe et al. [2] suggested a quality measure based on preserving inter-point visibility. Gudmundsson et al. [8] considered the problem studied in this paper for polygonal paths. Given a polygonal path $P = (p_1, \ldots, p_n)$ and an integer $1 < k < n$, compute a path $Q = (p_1, p_{i_2}, \ldots, p_{i_{k-1}}, p_n)$ of minimum dilation. They present both exact and approximation algorithms for this problem, as well as for the version where the stretch factor is given and one needs to minimize the number of vertices. Bose et al. [4] study the area-preserving simplification problem for x-monotone polygonal paths in the plane. In general, much work has been done on distance-preserving simplification from a theoretical point of view; see the new book by Narasimhan and Smid [10] on geometric spanner networks, and, e.g., papers [1, 6] that consider general graphs and are somewhat related to the problem studied in this paper.

2 A Distance-Based Quality of Simplification Measure

Let T (resp., T') be a terrain model consisting of n (resp., m) triangles, with $n > m$. We assume that T and T' are defined over a common underlying rectangular region, R, in the (x, y)-plane. For a point $p \in R$, let p_T (resp., $p_{T'}$) denote the point in $\mathbb{R}^3$ that is obtained by lifting p onto the surface of T (resp., T'). Given two points $p, q \in R$, let $GD_T(p, q)$ be the geodesic distance between p_T and q_T (i.e., the length of a shortest path on the surface of T between p_T and q_T).

Let $\mathcal{X}$ be a finite set of pairs of points in R. (One can think of $\mathcal{X}$ as the set of edges of a graph defined on a discrete set of points of R.) We define the similarity, in terms of geodesic distances, between T and T' with respect to $\mathcal{X}$. For each pair $\{p, q\} \in \mathcal{X}$, compute the ratio $\frac{GD_T(p,q)}{GD_{T'}(p,q)}$. Let $\mathcal{V}$ be the set of all these ratios. Then the *similarity* $\tau_{\mathcal{X}}$ between T and T' with respect to $\mathcal{X}$ (or, alternatively, the quality of simplification T' of T with respect to $\mathcal{X}$) is $\tau_{\mathcal{X}} = \frac{1}{|\mathcal{V}|} \cdot \Sigma_{v \in \mathcal{V}} |1 - v|$.

Our goal is to develop a simplification method that produces simplifications of good quality for small values of m, e.g., on the order of $0.5n$ or less, with respect to any "reasonable" set $\mathcal{X}$ of pairs of points in R. Here, we consider $\mathcal{X}$ to be a reasonable set if it is a subset of some pre-specified large (not necessarily finite) set $\mathcal{Z}$ of pairs of points in R, which is either generated randomly or is a typical subset of point pairs arising in the underlying application. Some natural choices of $\mathcal{Z}$

include (i) the set of all pairs of points in R, or (ii) the set of all pairs of grid points, for some regular grid on R, or (iii) the set of all pairs of (projections onto the (x, y)-plane of) some subset of the vertices of T. In Section 5 we report on experiments performed for choice (i), with the sets $\mathcal{X}$ defined to be all pairs determined by a random subset of points of R.

In practice we prefer to approximate the geodesic distances $GD_T(p, q)$ and $GD_{T'}(p, q)$, as described in the next subsection.

2.1 Approximating geodesic distances

It is common to use a graph G in order to approximate the geodesic distance between two points a and b on T. We define G as follows. Let δ be a parameter that depends on the average length of an edge of T and on the desired degree of accuracy. For each edge e of T, place $\lfloor len(e)/\delta \rfloor$ vertices in the interior of e. Now, for each triangle t of T, draw an edge for each pair of vertices on t's boundary (including the original 3 vertices).

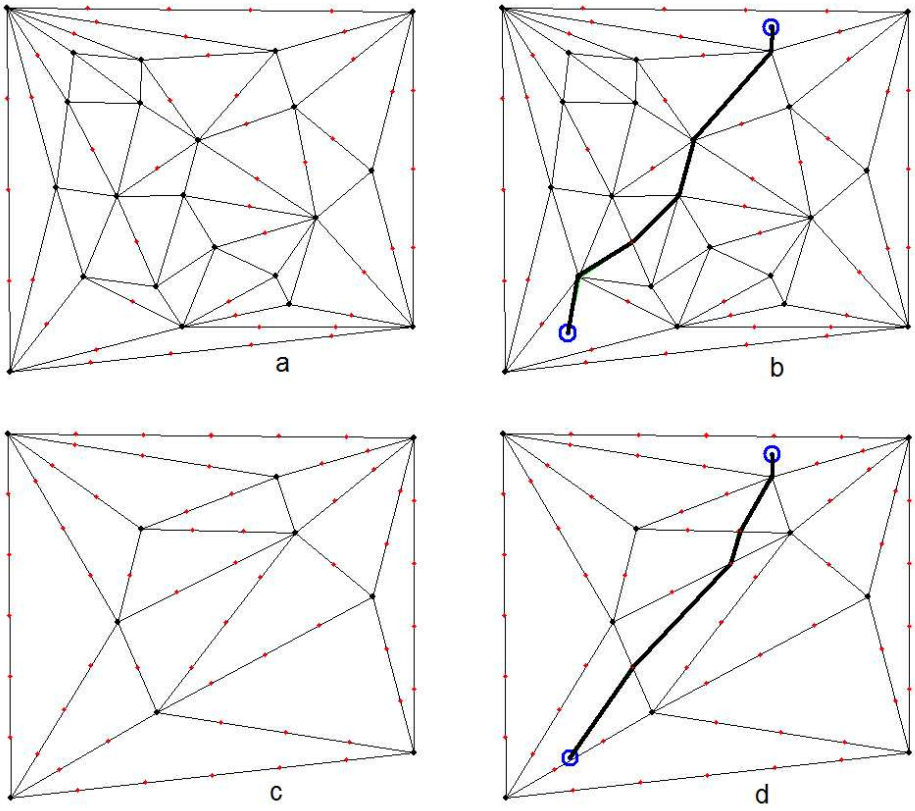

Figure 1: Approximating the geodesic distance between two points on a terrain. Top: a triangulation consisting of 22 vertices and 38 triangles plus 45 edge-interior vertices, i.e., 67 vertices in total. Bottom: a simplified triangulation consisting of 10 vertices and 14 triangles plus 56 edge-interior vertices, i.e., 66 vertices in total.

Let a, b be two points on T. We use G to approximate $GD_T(a, b)$ as follows; see Figure 1. If a and b happen to be vertices of G, then the distance between a and b is approximated by the

length of a shortest path in G between a and b; denote this length by $\Pi_G(a,b)$. Otherwise, let t_1 (resp. t_2) be the triangle to which a (resp. b) belongs. (If a (resp. b) is on an edge of T, then pick any one of the two possible triangles.) The distance between a and b is approximated by $\min_{u_1 \in V(t_1), u_2 \in V(t_2)}\{d(a,u_1) + \Pi_G(u1,u2) + d(u_2,b)\}$, where $V(t_i)$ is the set of vertices on t_i's boundary, $i = 1,2$, and $d(a,u1)$ (resp., $d(u2,b)$) is the Euclidean distance between a and u_1 (resp. u_2 and b).

3 General Methods for Terrain Simplification

In Section 4 we present two algorithms (PLD and VP) that are based on one of the standard methods for terrain simplification — the elimination method. This method is described in Subsection 3.1. Next, in Subsection 3.2, we describe a meta method that applies the elimination method in a sophisticated way. We use the meta method to obtain two variants PLD' and VP' of algorithms PLD and VP, respectively. Finally, we mention the well-known QSlim package 3.3 that we use in our experimentation.

3.1 The elimination method

Algorithms PLD and VP are based on the well-known elimination method.

1. Start from the original triangulation T.
2. For each vertex $v \in V(T)$, compute its **importance**.
3. While T is not yet **simplified enough**
   (i) Find a vertex $v \in V(T)$ with lowest importance.
   (ii) Remove v from T.
   (iii) **Update** T and the importance of the affected vertices.

In practice the vertices of T are stored in a priority queue H, where the priority of a vertex is its importance. In order to obtain an actual algorithm, one needs to define the **importance** of a vertex, the **simplified enough** condition, and the **update** operation after removing a single vertex from the current triangulation.

In the PLD algorithm, presented in the next section, the importance of a vertex $v \in V(T)$ is high if there exists a pair of vertices in v's immediate vicinity, such that the geodesic distance between them is large with respect to the Euclidean distance between them. In the VP algorithm, the importance of a vertex $v \in V(T)$ is proportional to the volume between T and the triangulation obtained by removing v from T. See Section 4.2 for the precise definitions.

3.2 The meta method

The meta method first divides the input triangulation into rectangular pieces, each with more or less the same number of vertices. It then applies the above simplification method (or any other simplification method) to each of the pieces separately. Finally, it combines the simplified pieces into a single simplified triangulation.

More precisely, one can think of the division stage as a preprocessing stage. In this stage, the original triangulation T is first divided into m rectangular pieces $T_1, \ldots, T_m$, each with roughly $|V(T)|/m$ vertices. Next, for each rectangular piece T_i a value is computed, taking into consideration the measure of quality of simplification that is being used. This value indicates how aggressive one can be when simplifying T_i. Now given a simplification algorithm such as PLD or VP, and parameter P2L (percent to leave) that tells us what percent of the vertices should remain in the output (simplified triangulation), the simplification algorithm is applied to each of the rectangular pieces separately. When applying the simplification algorithm to a rectangular piece T_i, P2L is adjusted according to the value that was computed for T_i. Finally, the simplified pieces are combined into a single triangulation with the desired number of vertices.

In our application, the value of a rectangular piece indicates how sensitive is this piece to the removal of vertices, with respect to the similarity measure described in Subsection 2. For example, a rectangular piece that is nearly planar, should have a value that indicates that it can be simplified very aggressively. See Section 4.3 for an exact description of how the piece values are computed.

3.3 QSlim

QSlim, developed by Garland and Heckbert [7], is a more general algorithm designed for simplifying all types of surfaces, not just terrains. QSlim uses simple edge contraction to perform simplification, while using a quadric error measure for visual fidelity and for efficiency. QSlim is well-known for its efficiency and high quality of approximation; we therefore picked it as a point of reference for the new algorithms presented in this paper.

4 Distance-Preserving Terrain Simplification Algorithms

Let T be a Delaunay triangulation representing a rectangular terrain (i.e., a height value is associated with each vertex of T), and let P2L (percent to leave) be a parameter that tells us what percent of the vertices of T should remain in the output (simplified triangulation). We begin this section with a detailed description of the *preserving local distances algorithm* (PLD) and the *volume preserving algorithm* (VP), that are based on the elimination method mentioned in Section 3. Next we describe our implementation of the meta method that yields algorithm PLD', if PLD is applied, and algorithm VP', if VP is applied.

In general, we deal with Delaunay triangulations. Thus, when a vertex v is removed from the current triangulation, it is done by calling the Delaunay delete operation, that updates the current triangulation. The set, A_v, of vertices that are *affected* by v's deletion, consists of all vertices whose set of neighbors has changed as a result of v's removal.

4.1 PLD

It remains to define the importance of a vertex, which is a value between 0 and 1. The importance of a vertex u is computed right at the beginning, and is updated whenever u belongs to the set of vertices that are affected by the deletion of a vertex (see above).

We use the following notation. $N[u]$ is the set of neighbors of vertex u in T, $ED(v, w)$ is the Euclidean distance (in 3-space) between v and w, and $GD_T(v, w)$ is the geodesic distance between v and w (i.e., the length of a shortest path on the surface of T between v and w).

Importance(T, u)

1. **if** u **lies on the boundary of** T **then return 1**
2. $r \leftarrow 1$
3. **for each** $v, w \in N[u]$ **do**
4. **if** $r > ED(v, w)/GD_T(v, w)$ **then**
5. $r \leftarrow ED(v, w)/GD_T(v, w)$
6. **return** $(1 - r)$

In words, the importance of u is high (i.e., close to 1), if u has a pair of neighbors, such that the geodesic distance between them is large with respect to the Euclidean distance between them. In this case, u will not be deleted, i.e., local distances are preserved.

4.2 VP

This algorithm attempts to preserve the volume in the sense defined below. It is therefore reasonable to expect that it would also perform well with respect to our quality of simplification measure.

The importance of a vertex u in this algorithm is proportional to the volume of the set of all points that lie between the current triangulation T and the triangulation that is obtained by (Delaunay) deleting u from T. That is, let T' be the triangulation obtained by deleting u from T. A point p (in 3-space) lies between T and T' if and only if it is either above T and below T' or above T' and below T. In order to determine the importance of u (in T), we approximate the volume of the set X of all such points. We now describe how this is done.

Let A_u be the set of vertices that are affected by the deletion of u (see above). Ignoring the third dimension, let R be the (axis-aligned) bounding rectangle of A_u. Let B be the 3-dimensional box bounding both T and T' over R. We approximate the volume of X as follows.

Volume(X)

1. **Let** P **be a set of** l **randomly generated points in** B
2. $Count \leftarrow 0$
3. **for each** $p \in P$ **do**
4. $a \leftarrow p_T$
5. $b \leftarrow p_{T'}$
6. **if** $(a.z < p.z < b.z)$ **or** $(b.z < p.z < a.z)$ **then**
7. $Count \leftarrow Count + 1$
8. **return** $(Count/l) * volume(B)$

Importance(T, u)

1. **return Volume(X)**

4.3 PLD' and VP'

We need to describe the division stage (see Section 3.2), and, in particular, we need to define the value of a rectangular piece.

The division itself is standard; it is similar to the division corresponding to a (2-dimensional) k-d tree [3], except that we limit the number of levels by a small constant c. That is, we divide the rectangle underlying T into two subrectangles by a horizontal or vertical line, such that the number of vertices of T in each of the resulting subrectangles is roughly the same. Next we divide each of these two subrectangles, etc. At the end of this process we obtain a division of the rectangle underlying T into $m = 2^c$ rectangles, where each rectangle underlies a rectangular piece T_i of T with roughly $|V(T)|/m$ vertices.

We now define the *value* of a rectangular piece T_i. Informally, this value is equal to the average ratio between the Euclidean distance between two points on T_i and the geodesic distance between these points. The value is computed as follows, where R is the rectangle underlying T_i.

CalcPieceValue(T_i)

1. **Let** P **be a set of** l **randomly generated points in** R
2. $val \leftarrow 0$
3. **for each** $p \in P$ **do**
4. **for each** $q \in P$, $q \neq p$ **do**
5. $a \leftarrow p_{T_i}$
6. $b \leftarrow q_{T_i}$
7. $val \leftarrow val + ED(a,b)/GD_{T_i}(a,b)$
8. $val \leftarrow val/\binom{l}{2}$

We now apply either PLD or VP to each of the m rectangular pieces. The value of a rectangular piece T_i (together with the overall percent-to-leave requirement) tells us how aggressive we can be when applying the simplification algorithm to T_i; that is, it determines the parameter P2L with which PLD/VP is applied to T_i. More precisely, P2L (for T_i) is calculated as follows.

$$P2L \leftarrow 100 - \frac{val(T_i)^2}{S} * m(100 - \text{overall percent-to-leave}) \ ,$$

where S is the sum, over all rectangular pieces T_j, of $val(T_j)^2$.

In practice, T_i is passed implicitly to the simplification algorithm (PLD or VP), by passing the original triangulation T and the rectangle underlying T_i. The output of the application of the simplification algorithm to T_i is a subset of the vertices of T_i that is added to the subset V' of the vertices of T that eventually determines the overall simplification T'. After applying the simplification algorithm to each of the pieces T_i, the overall simplification T' is obtained by computing the Delaunay triangulation of V'.

4.4 Running time

We analyze the algorithms above in terms of running time. Consider the elimination method described in Section 3.1. Assuming the Delaunay triangulation is being used as well as a heap storing the remaining vertices by importance, the expected running time of the elimination method is $O(|V|(\log|V|+I))$, where I is the expected running time of the computation of the **importance** of a vertex. We have also assumed that the **simplified enough** condition can be evaluated in $O(1)$ time.

Consider algorithm PLD. Since the average degree of a vertex of a Delaunay triangulation is $O(1)$, the expected running time of the computation of the importance of such a vertex is $O(1)$, and we conclude that the expected running time of PLD is $O(|V|\log|V|)$. Consider algorithm VP. The running time of the importance computation depends on l, the number of randomly generated points in box B. Assuming l is some constant, the expected running time of VP is also $O(|V|\log|V|)$. Finally, the division stage in algorithms PLD' and VP' can be done in $O(|V|\log|V|)$ time, assuming c and l are constants, see above. Thus the expected running time of PLD' and VP' is also $O(|V|\log|V|)$.

5 Experimental Results

In this section we report on some of our experiments with algorithms PLD, VP, PLD', VP', as well as comparisons with another software package — QSlim [7]. Tables 1-3 summarize our results.

5.1 Working environment

Our software package, DPTS, was developed in C++, under Windows XP, using the Computational Geometry Algorithms Library CGAL-3.2 [5]. The main data structures used are Delaunay_triangulation_2, with the Euclidean metric, for the projection of a terrain model onto the plane, and Triangulation_euclidean_traits_xy_3. Points are represented by Cartesian<double>. Although the points are in 3D, the predicates in the construction of the Delaunay triangulation are computed using only the x and y coordinates of the points. Some of the modules were computed from scratch, such as the module for approximating geodesic distances.

5.2 Terrain datasets

Three input terrains representing three different and varied geographic regions were used. Each input terrain covers a rectangular area of 40–6,000 square kilometers and consists of 5,000-15,000 vertices. We avoided using very flat terrains which are easy to simplify under our similarity measure. Instead, hilly terrains with interesting geographic elements, such as, craters, canyons, dunes, and lakes, were used. We had to pick relatively small terrains, since the quality of simplification computation (see below) is extremely time consuming. We note though that all our simplification algorithms are quite efficient and can handle terrains with hundreds of thousands vertices.

5.3 Experiments using the distance-based measure

For each input terrain T, 4 simplifications were computed of sizes 70%, 50%, 30%, and 10%, respectively, using each of the 5 simplification algorithms. (That is, for each input terrain T, 20

Simplification size w.r.t. input size	PLD'	PLD	VP'	VP	QSlim
10%	0.045268	0.091548	0.028649	0.026433	0.02929
30%	0.025535	0.032744	0.016866	0.018212	0.018491
50%	0.015798	0.016119	0.010795	0.011841	0.011123
70%	0.008815	0.008695	0.006209	0.005813	0.006272

Table 1: Southern Israel map.

Simplification size w.r.t. input size	PLD'	PLD	VP'	VP	QSlim
10%	0.048523	0.123608	0.028094	0.026186	0.02495
30%	0.023039	0.030387	0.018348	0.017794	0.017773
50%	0.015886	0.015514	0.012469	0.013613	0.013718
70%	0.00861	0.00802	0.007364	0.008152	0.008835

Table 2: Crater map.

different simplifications were computed). 3 sample sets, labeled A_1, A_2, A_3 and consisting of 100 points each, were generated by randomly selecting points in the rectangle R underlying T.

The quality of simplification T' of T with respect to sample set A is computed as follows (see also Section 2). For each of the $\binom{|A|}{2}$ pairs (p, q) of points in A, we compute the ratio $\frac{|GD_T(p,q)-GD_{T'}(p,q)|}{GD_T(p,q)}$, where $GD_T(p, q)$ is the geodesic distance on T between p_T and q_T. The error of T' with respect to A, denoted $Err_{T'}(A)$, is the average over all these $\binom{|A|}{2}$ ratios. The distance-preserving error of T' is $\frac{Err_{T'}(A_1)+Err_{T'}(A_2)+Err_{T'}(A_3)}{3}$.

Our results are presented in Tables 1–3. Consider, e.g., Table 1. This table summarizes our results for an input terrain representing a region in southern Israel and consisting of roughly 13,000 vertices. The first line of the table refers to the 5 simplifications, each consisting of roughly 1,300 vertices, that were computed using algorithms PLD', PLD, VP', VP, and QSlim, respectively. For each of these simplifications, the table shows its error (see above). For example, the distance-preserving error of the 10% simplification obtained by applying VP' is 0.028649. Figure 2 corresponds to the second line of Table 1.

Tables 1–3 lead us to the following conclusions (some of which may require additional experiments in order to fully validate them).

- As expected, the error decreases as the size of the simplification increases. That is, each of the columns is decreasing.
- VP is significantly better than PLD. (The latter is slightly better only in one case — Table 2,

Simplification size w.r.t. input size	PLD'	PLD	VP'	VP	QSlim
10%	0.088395	0.101355	0.065091	0.045292	0.042831
30%	0.036856	0.046502	0.027093	0.027616	0.027235
50%	0.024968	0.02958	0.017517	0.018293	0.016728
70%	0.014865	0.016158	0.010154	0.008359	0.011327

Table 3: Northern California map.

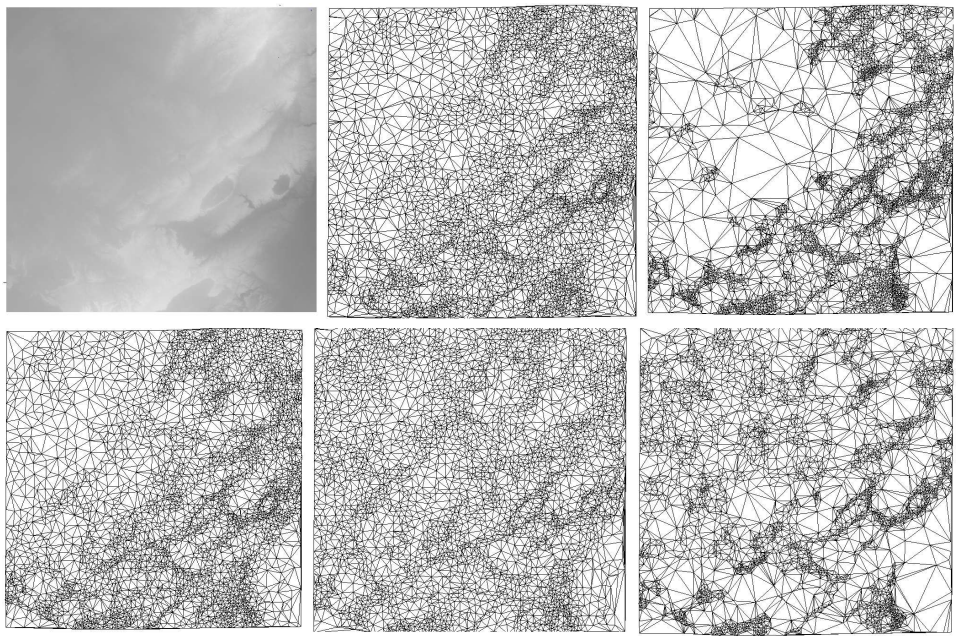

Figure 2: The input terrain of southern Israel (the brighter the higher), and the 5 simplifications, each of roughly 3900 vertices, that were computed; see Table 1, second line. Top left: input terrain. Top middle: VP. Top right: PLD. Bottom left: QSlim. Bottom middle: VP'. Bottom right: PLD'.

last line.)

- In general, a dramatic improvement is achieved by replacing PLD by PLD', especially when the simplification size decreases.
- The differences between the errors obtained for VP and VP' are small, where each wins 1/2 of the times.
- The error obtained for VP' is usually slightly smaller than that for QSlim; VP' wins 2/3 of the times. The advantage of VP' over QSlim increases when the simplification is not too small.

Figure 2 suggests an explanation for the inferiority of PLD. As can be seen, PLD tends to leave too many vertices in "abnormal" regions with sharp geographic features, and therefore too few vertices in "normal" regions. Since usually a large portion of the terrain consists of "normal" regions, and since the sample points are chosen randomly, geodesic distances are not well preserved for many of the pairs of sample points. By replacing PLD by PLD', we introduce a global consideration, which explains the significant improvement that is achieved.

Acknowledgment. The authors wish to thank Michael Elkin and Joseph Mitchell for helpful discussions.

References

[1] I. Abraham, Y. Bartal, H. T-H. Chan, K. Dhamdhere, A. Gupta, J. M. Kleinberg, O. Neiman, and A. Slivkins. Metric embeddings with relaxed guarantees. In *Proc. 46th IEEE Symp. Foundations Computer Science*, pages 83–100, 2005.

[2] B. Ben-Moshe, M. J. Katz, J. S. B. Mitchell, and Y. Nir. Visibility preserving terrain simplification — An experimental study. *Comput. Geom. Theory Appl.* 28(2-3) (June 2004), 175–190.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*, Second Edition. Springer-Verlag, 2000.

[4] P. Bose, S. Cabello, O. Cheong, J. Gudmundsson, M. van Kreveld, and B. Speckmann. Area-preserving approximations of polygonal paths. *Journal of Discrete Algorithms*, 4 (2006), 554–566.

[5] CGAL Editorial Board. CGAL-3.2 User and Reference Manual. 2006. `http://www.cgal.org/Manual/3.2/doc_html/cgal_manual/contents.html`

[6] D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *Proc. ACM-SIAM Symp. Discrete Algorithms*, pages 660–669, 2005.

[7] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH'97*, pages 209–216, 1997.

[8] J. Gudmundsson, G. Narasimhan, and M. Smid. Distance-preserving approximations of polygonal paths. *Comput. Geom. Theory Appl.* 36 (2007), 183–196.

[9] P. S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Manuscript.

[10] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.