Thaw: A Tool for Approximating Cut Loci on a Triangulation of a Surface

Jin-ichi Itoh and Robert Sinclair

CONTENTS

The cut locus from a point on the surface of a convex polyhedron is a tree containing a line segment beginning at every vertex. In the limit of infinitely small triangles, the cut locus from a point on a triangulation of a smooth surface therefore tends to become dense in the smooth surface, whereas the cut locus from the same point on the smooth surface is also a tree, but of finite length. We introduce a method for avoiding this problem. The method involves introducing a minimal angular resolution and discarding those points of the cut locus on the triangulation for which the angle measured between the shortest geodesic curves meeting at these points is smaller than the given angular resolution. We also describe software based upon this method that allows one to visualize the cut locus from a point on a surface of the form $(x/a)^n + (y/b)^n + (z/c)^n = 1$, where n is a positive even integer. We use the software to support a new conjecture that the cut locus of a general ellipsoid is a subarc of a curvature line of the ellipsoid.

1. INTRODUCTION

The cut locus is a fundamental object of study in global differential geometry [Berger 00], which has applications reaching all the way to parasitology [Dujardin and Duriez 95]. It is closely related to Voronoi diagrams [Aurenhammer 91], the notion of "ridge trees" [Agarwal et al. 97], and the medial axis transform [Wolter and Friese 00]. The cut locus also naturally appears in the recognition problem of computer vision [Pennec 98].

The cut locus from a point p on a surface is the closure of the set of all points that have at least two shortest paths connecting them to p [Wolter 79]. A cut point of p along a geodesic (shortest, or locally length-minimizing path) passing through p is the first point on the geodesic where it ceases to minimize its arc length to p, and the cut locus of p is the set of all cut points of p [Kobayashi 67]. For a more technical definition, see Section 4 of Chapter III of [Sakai 96].

2000 AMS Subject Classification: Primary 53-04; Secondary 53C20

Keywords: Cut locus, ellipsoid, computational global differential geometry

1058-6458/2004$0.50 per page

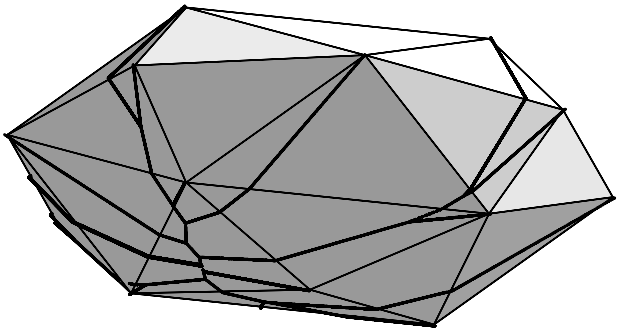

FIGURE 1. The cut locus from a point (not visible, but on the other side) on the surface of a polyhedron. Note that every vertex is the beginning of a branch of the cut locus.

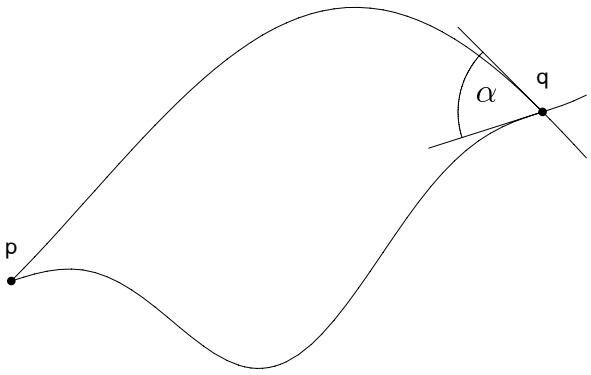

FIGURE 2. Let p be a point on a two-surface and q a point of its cut locus with two equal-length shortest segments between p and q. Let α be the angle at which these geodesics meet at q. We only accept this point q if $\alpha > \Delta\theta$, where $\Delta\theta$ is the minimal angular resolution.

It is known [Buchner 77, Buchner 78] (see also [Ozols 74]) that generic cut loci in low-dimensional manifolds are triangulable and structurally stable. In dimension two, a generic cut locus is simple to describe: each point q has a neighborhood in the cut locus that is either (i) a straight line through q, (ii) a straight line starting at q, or (iii) three straight lines meeting at q to form a "Y." See also [Bishop 77], where it is shown that ordinary cut points of a point m are dense in the cut locus of m.

The tool presented here is the second in a series of software tools written for the visualization of the cut locus from a point on a two-dimensional surface. Loki [Sinclair and Tanaka 02] is a rather large and complex C++ program, which can compute the cut locus on a torus-like surface from a closed-form parametrization or metric to great accuracy. However, Loki's complexity makes it difficult to adapt it to other surfaces. It is also quite slow in producing low-resolution data.

Thaw (the name of the software tool to be presented here—see Figure 3) is intended to be the prototype of a more "quick and dirty" but also more flexible approach. Its starting point was actually considered quite unlikely at the time Loki was written: to use a triangulation of a surface, rather than closed-form parametrizations or metrics, as the surface description. It is well known that the cut locus from a point on the surface of a polyhedron (see Figure 1) contains many more line segments than the cut locus of a corresponding smooth surface, to the point that it does not seem practicable to compute an approximation to the smooth surface's cut locus via this route. This becomes particularly clear when one recalls that the cut locus of a two-dimensional submanifold of $\mathbb{R}^3$ with nondegenerate distance-squared function is of measure zero [Hendricks 92], but also that the cut locus of a polyhedral surface of positive curvature has a branch beginning at every vertex [Agarwal et al. 97]. In the limit of infinitely many triangles, one can therefore expect that the cut locus of the polyhedral surface comes to fill the entire surface. One would appear to be in danger of being overwhelmed by branches that belong to the cut locus of the triangulations but not of the smooth surface.

However, some further thought led us to the conclusion that it might be possible nonetheless. Thaw is indeed able to approximate the cut locus of a surface that is approximated only by a triangulation. This has been made possible by a procedure that blurs or melts (indeed *thaws*) away those branches that belong to the cut locus of the triangulation but not of the smooth surface.

The cut loci from points on polyhedral surfaces of positive curvature are quite well understood [Volkov and Podgornova 71], and there has been much work on the question of counting and finding shortest paths on such surfaces [Sharir and Schorr 86, Mitchell et al. 87, Mount 90, Agarwal et al. 97, Kaneva and O'Rourke 00]. We are, however, interested in the cut loci of smooth surfaces *approximated by polyhedral surfaces*, rather than the cut loci of the polyhedral surfaces themselves. This is an important distinction [Polthier and Schmies 99]. If we were primarily interested in the cut loci of polyhedral surfaces, then we could make use of highly efficient algorithms for computing the length of the shortest path on these surfaces [Agarwal et al. 97, Kapoor 99, Kaneva and O'Rourke 00], but we will actually need a list of *all* locally minimal paths between two points (up to some given length).

At present, Thaw is only capable of handling triangulations in which the sum of the angles at every vertex is less than 2π (i.e., convex surfaces). We have based our algorithm upon the mathematical formulation of [Agarwal et al. 97], which in turn builds upon the work of [Sharir and Schorr 86] and [Mitchell et al. 87], where cut loci (called ridges) on polyhedral surfaces are defined for convex polyhedra. In the convex case, the cut locus from a point is the closure of the set of points to which there are two or more minimizers, where a shortest path cannot pass through a vertex of the surface (Lemma 4.1 of [Sharir and Schorr 86]). In the nonconvex case, we have the possibility of the shortest path passing through a vertex. As Huygen's principle tells us (see, for example, Section 86 of [Coulson 55] for a definition), each point on a wavefront can be considered as the source point of a new (initially circular) wavefront. [Kaneva and O'Rourke 00] discuss the nontrivial implementation extensions that were necessary to allow them to find shortest paths on nonconvex surfaces. When extending the definition of the cut locus to nonconvex surfaces, constructing candidate shortest paths is more difficult because we do not a priori know the angle through which a candidate path may pass through any given vertex. All we know is that this angle is greater than or equal to π (Lemma 3.4 of [Mitchell et al. 87]). However, once the set of candidate shortest paths has thus been extended, if the definition of the cut locus remains unchanged as the closure of the set of points to which there are two or more minimizers, then the cut locus is not necessarily a one-dimensional graph but can contain an open set. In other words, it is not obvious how to extend the definition of the cut locus to general polyhedral surfaces. We will not attempt to answer this question here, since we are in fact only interested in the cut loci of smooth convex surfaces. Our algorithm will correspond to the definition given above, which is appropriate for smooth surfaces and convex polyhedra.

Our restriction to convex surfaces may be removed in future versions of Thaw, but this first requires the introduction of new elements to emulate Huygen's principle at those vertices with angle sum greater than 2π. The algorithms of [Kaneva and O'Rourke 00], [Kapoor 99] and [Sava and Fomel 98] are of interest in this context. Second, it would require a good definition of the cut locus on negatively curved polyhedral surfaces.

Thaw is, however, already able to approximate the cut loci of surfaces Loki cannot handle, has already had some influence on recent analytical work, and has allowed us to experimentally confirm the conjecture that the cut loci of general ellipsoids are subarcs of curvature lines (see the Remark to Proposition 3.5.4 of [Klingenberg 82] for a definition).

Note that the problem of finding the cut locus of a solid ellipsoid from its surface has already been solved [Degen 97], but we are interested in the cut locus on the surface of an ellipsoid from a point on that surface.

The cut loci from points on the surface of general ellipsoids have been studied for some time (see Section 3.5 of [Klingenberg 82]). For example, it is known that the cut locus of an umbilic point of an ellipsoid coincides with the antipodal umbilic point, and the cut loci from certain points on ellipsoids of revolution are known to be arcs (Remarks to Theorem 2.1.14 in [Klingenberg 82]).

In [Polthier and Schmies 98], the concept of "straightest geodesics" was defined, these having the advantage of being unique solutions to the initial value problem for geodesics on polyhedral surfaces. They were used, in conjunction with an adaptive particle propagation scheme similar to [Lambare et al. 96], to compute the evolution of distance circles on polyhedral surfaces [Polthier and Schmies 99]. As we do, Polthier and Schmies also faced the problem of the local branching of the wavefront related to the discretization introduced by approximating a smooth geometry with a polyhedral surface. Their solution was to use a reasonably fine mesh to suppress the type of local branching related to the discretization, and then, when constructing the branched texture map used to visualize evolving waves, to introduce a threshold (in Section 5.1 of [Polthier and Schmies 99]) that effectively smoothes the Lagrangian manifold (see Section 4 of [Lambare et al. 96] for a definition). This threshold applies to both angle and time. Polthier and Schmies always apply their threshold formula, but, in the case of smooth surfaces, they also use a finer mesh in order to enable the threshold formula to work more accurately and to be able to really distinguish between polyhedral and geometric conjugate points.

Our work differs from that of Polthier and Schmies in that their aim was to visualize the geodesic flow and its branching behaviour, whereas ours is to compute approximations to cut loci on smooth surfaces. The quite specific demands of computing an approximation to the cut locus led us to take a different approach. In particular, we find that computed approximations to cut loci are extremely sensitive to any errors caused by the representation used to approximate wavefronts.

Given the known weaknesses of particle (or ray) methods for approximating wavefronts [Lambare et al. 96], particularly in the presence of conjugate points, we decided to represent the wavefront *exactly* (up to numeri-

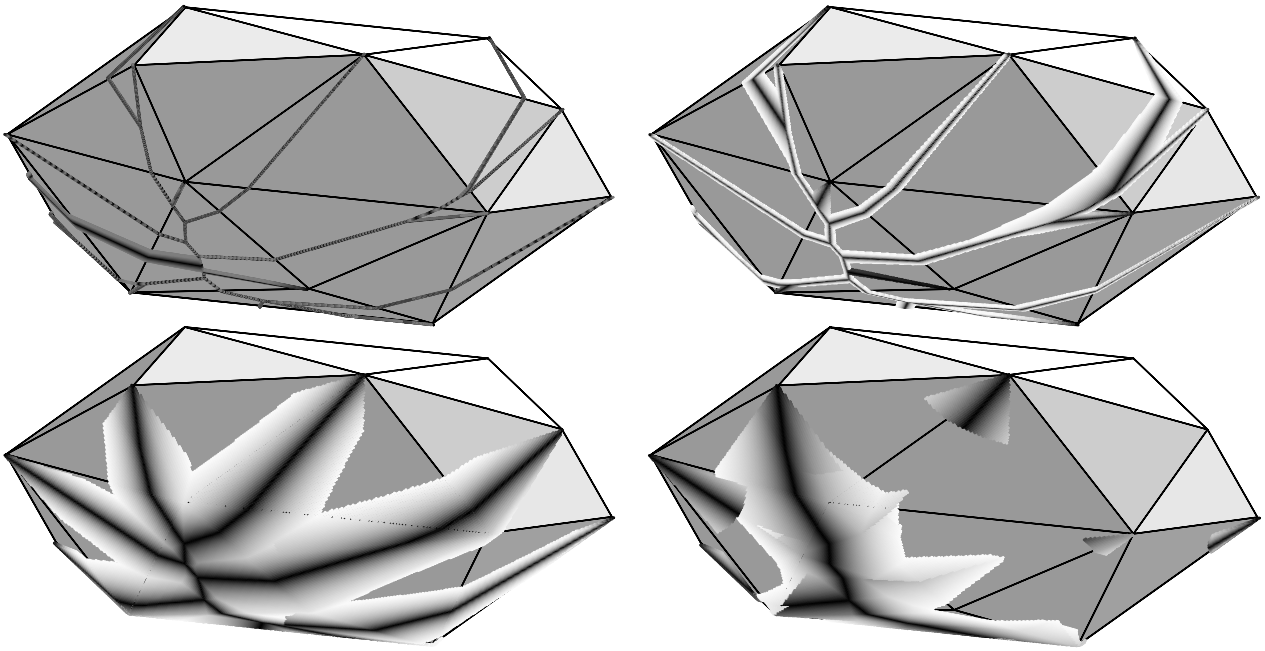

FIGURE 3. The effect of increasing $\Delta\theta$ from 0.005 (top left) to 0.05, 0.5, and 1 (bottom right). The process reminds one of the melting of an ice crystal—hence the name Thaw.

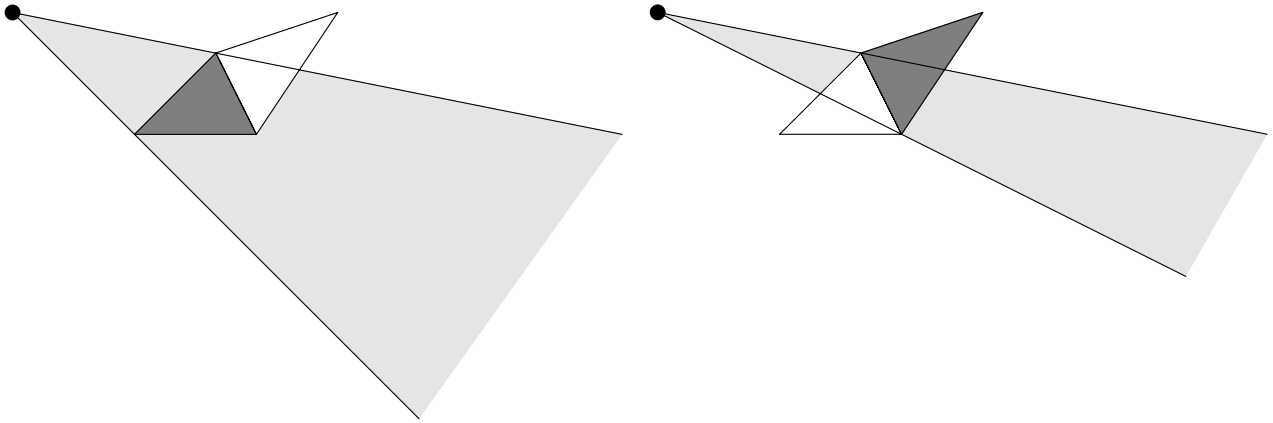

FIGURE 4. Two adjacent triangles, and the transfer of source data from the left to the right one. In the left diagram, we can see a source illuminating the left-hand triangle. Its light shines partly onto the right-hand triangle. The right-hand triangle is therefore also informed about this source, but only about that part of its light (the grey wedge, which should be imagined to extend infinitely far from the source, which is shown as a black point) that actually shines on it. In the right diagram, we can see the data the right-hand triangle is given concerning this source.

cal error) as a list of the centres of the circular arcs of which the actual wavefront is composed, as well as the angular limits of each arc. We call this a "light source technique," since the wavefront from one actual source on a polyhedral surface appears to an observer living on the surface as the "light" from many different sources, each shining within clearly defined angular limits. From a mathematical point of view, we use essentially the same definitions as Sections 2.1 and 3.1 of [Agarwal et al. 97], but with a different implementation. Our algorithm resembles that of [Kapoor 99], which also uses a waveform propagation method based upon sequences of arcs of circles, but theirs is more complicated because it also involves the removal of arcs that are irrelevant to their aim of computing shortest paths and shortest distances (but not the cut locus). Also, the implementation of Chen and Han's shortest path algorithm in [Kaneva and O'Rourke 00] makes use of the concept of a "cone," and [Sharir and Schorr 86] define "slices" in their Section 5, both which are essentially identical to the wedges of our implemen-

tation (see Figure 4). That we are able to model wavefronts exactly, rather than as piecewise geodesic curves [Polthier and Schmies 99], allows us to significantly reduce error in our computation of the cut locus. As will become clear, there are in fact many further sources of error, which still make computing an approximation to the cut locus extremely difficult.

Recall that the mathematical formulation [Agarwal et al. 97] upon which our method is based applies only to convex surfaces. Since, however, the cut locus on a two-dimensional ellipsoid (even on one of revolution) is still not known [Berger 00], we are still able to address important mathematical questions even with such a limitation, and we have concentrated instead on questions of efficiency.

Our technique involves little more than a straightforward application of elementary two-dimensional Euclidean geometry, so a thorough mathematical formulation will not be included in this paper. The interested reader is referred to [Agarwal et al. 97].

2. THE ALGORITHM

Thaw assumes that a surface is approximated by a triangulation embedded in three-space. This condition may be removed in future versions, since it is not necessary from a purely mathematical point of view, but it has been useful during debugging to be able to "see" the surface being treated.

The central idea is to introduce a minimal angular resolution $\Delta\theta$ and to accept only those points of the cut locus for which the two shortest (and closest) geodesics meet at greater than this angle. See Figure 2. This $\Delta\theta$ essentially cuts off "artificial" line segments of the cut locus of the given triangulation, such that (one hopes) the cut locus of the corresponding smooth surface becomes visible. As will be seen below, Thaw actually makes use of $\Delta\theta$ as a "focusing" (or "blurring") parameter, such that one has the feeling of focusing a camera on a cut locus when using the software. See Figure 3 for an idea of what is meant by this.

On an abstract level, as was the case with Loki, we first construct an approximation to the exponential map, and then compute an approximation to the cut locus using this information. In practice, however, we perform these two processes simultaneously (computing an approximation to the cut locus within any triangle as soon as possible during the construction of the exponential map, and then discarding that triangle). This reduces memory requirements significantly. Nonetheless, in the following we will discuss the algorithm as if these two processes were strictly consecutive.

2.1 Approximating the Exponential Map

For any given triangle, we construct a local two-dimensional orthogonal coordinate system with distances inherited from three-space. The origin of these coordinate systems is always one corner of the triangle in question.

A fundamental step of the algorithm is the unfolding of two adjacent triangles of a triangulation, such that they both lie in one plane. Everything else involves only two-dimensional Euclidean geometry (for example, Figure 4 is to be understood literally, with points, lines, and triangles all in the same plane). Take the triangles ABC and ABD of Figure 5. We would like to rotate ABD around the common edge AB until D' (the image of D under this rotation) lies in the same plane as ABC and in such a way that the two triangles do not overlap when unfolded. We make use of the fact that the distances between points connected by an edge are known—they can be immediately computed from the triangulation chosen—and that the triangulation is not singular. Let d_{AB} be the distance from A to B, d_A the distance from D (or D') to A, and d_B the distance from D (or D') to B. Then, there are two points on the plane with distances d_A and d_B from A and B respectively. These have coordinates

$$\left(A_x + \frac{\ell_A\,(B_x - A_x)}{d_{AB}} \pm \frac{\ell\,(B_y - A_y)}{d_{AB}},\right.$$
$$\left. A_y + \frac{\ell_A\,(B_y - A_y)}{d_{AB}} \pm \frac{\ell\,(A_x - B_x)}{d_{AB}}\right),$$

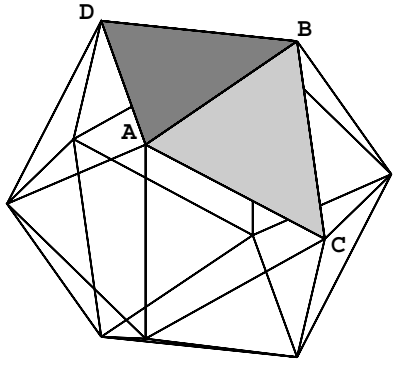

FIGURE 5. Two adjacent triangles of an icosahedron.

where A has coordinates (A_x, A_y), B has coordinates (B_x, B_y),

$$\ell_A = \frac{d_A^2 - d_B^2 + d_{AB}^2}{2\, d_{AB}},$$

and

$$\ell = \frac{1}{2}\sqrt{2\,(d_A^2 + d_B^2) - d_{AB}^2 - \frac{(d_A^2 - d_B^2)^2}{d_{AB}^2}}.$$

To ensure that there is no overlap, we choose the point which is further from C. Knowing the coordinates of D' in terms of the coordinates of A and B in the local coordinate system of triangle ABC allows us to uniquely determine a linear map from the one triangle's local coordinate system to the other's.

We represent these linear maps by a rotation matrix and an offset vector for each edge of every triangle. We precompute and store these matrices and vectors so that the many transformations which must be performed during a cut locus computation are as fast as possible. The associated memory usage does not seem to be a limiting factor when regarding Thaw's execution as a whole.

We regard the starting point (the cut locus is from this point) as a classical source of light, but only allow it to shine through edges (not vertices, and this is the reason why Thaw cannot handle triangulations with vertices of negative curvature yet—see [Agarwal et al. 97]) from triangle to triangle. Every triangle owns a list of images of the starting point. Each image is defined in terms of its coordinates with respect to the current triangle's coordinate system.

First, the triangle containing the starting point is assigned four light sources with the coordinates of the starting point with respect to this triangle. These four sources each light up one quadrant (one shines to the right and upwards, another to the left and upwards, etc.). Then, each of these sources (and all others which will appear as the algorithm proceeds) is treated as follows (see Figure 4):

The light from a source is considered to shine within a wedge (that is, only in certain directions, these directions forming a continuous set), the angle at the source always being less than or equal to $\pi/2$. Any edge of the current triangle through which light from the source shines outwards is intersected with this wedge to form a new wedge. The new wedge represents the light shining from the source into the next triangle sharing this same edge. The coordinates of the source and the wedge are converted into coordinates with respect to the other, adjacent triangle, and the new source (and wedge) data is added to the list of sources of the other triangle.

In this manner, we follow light from the starting point as it propagates through the surface. A stopping condition for this process is provided by only adding new source data to a triangle if the distance to the new source is less than the distance to the closest source so far by three times the length of the longest edge of the triangle.

2.2 Approximating the Cut Locus

In the following, we will make use of the example of the non-rotationally-symmetric ellipsoid

$$\frac{x^2}{1^2} + \frac{y^2}{0.9^2} + \frac{z^2}{0.4^2} = 1, \tag{2–1}$$

upon which we have placed a starting point at

$$(x, y, z) \approx (\,-0.150794,\, 0.296542,\, 0.372816\,). \tag{2–2}$$

Given a triangle with longest edge length r and a point x inside the triangle, we can compute a number that should give us an indication of the distance to the cut locus of the smooth surface. First, we compile a list of all sources that are visible to (or shine upon) x. See Figure 6. From those, we identify the source closest to x, calling it $\mathcal{S}_1$, at distance ρ_1. Then, we look at all the

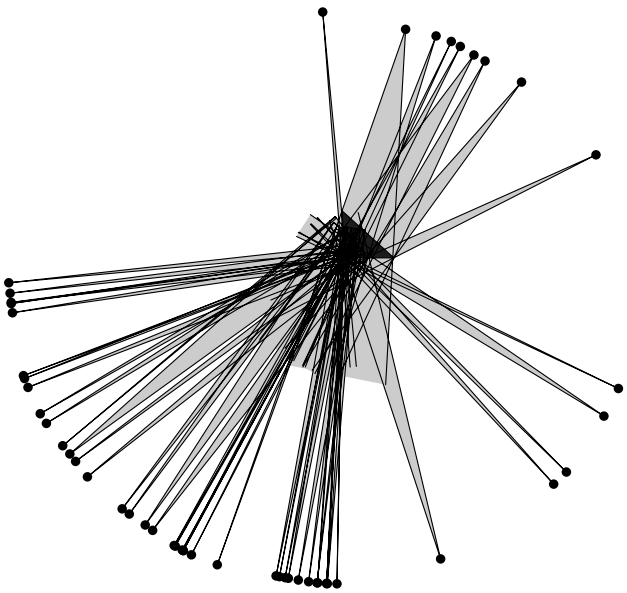

FIGURE 6. The fact that we are working with a triangulation of a surface means that if we imagine ourselves situated in one of the triangles, our view of the starting point is segmented, as if we had the eyes of an insect. Therefore, we will see the starting point in many more different directions than would have been the case if we were situated inside the smooth surface. In this illustration, we can see the images of the starting point as seen by a triangle close to (perhaps including) the cut locus.

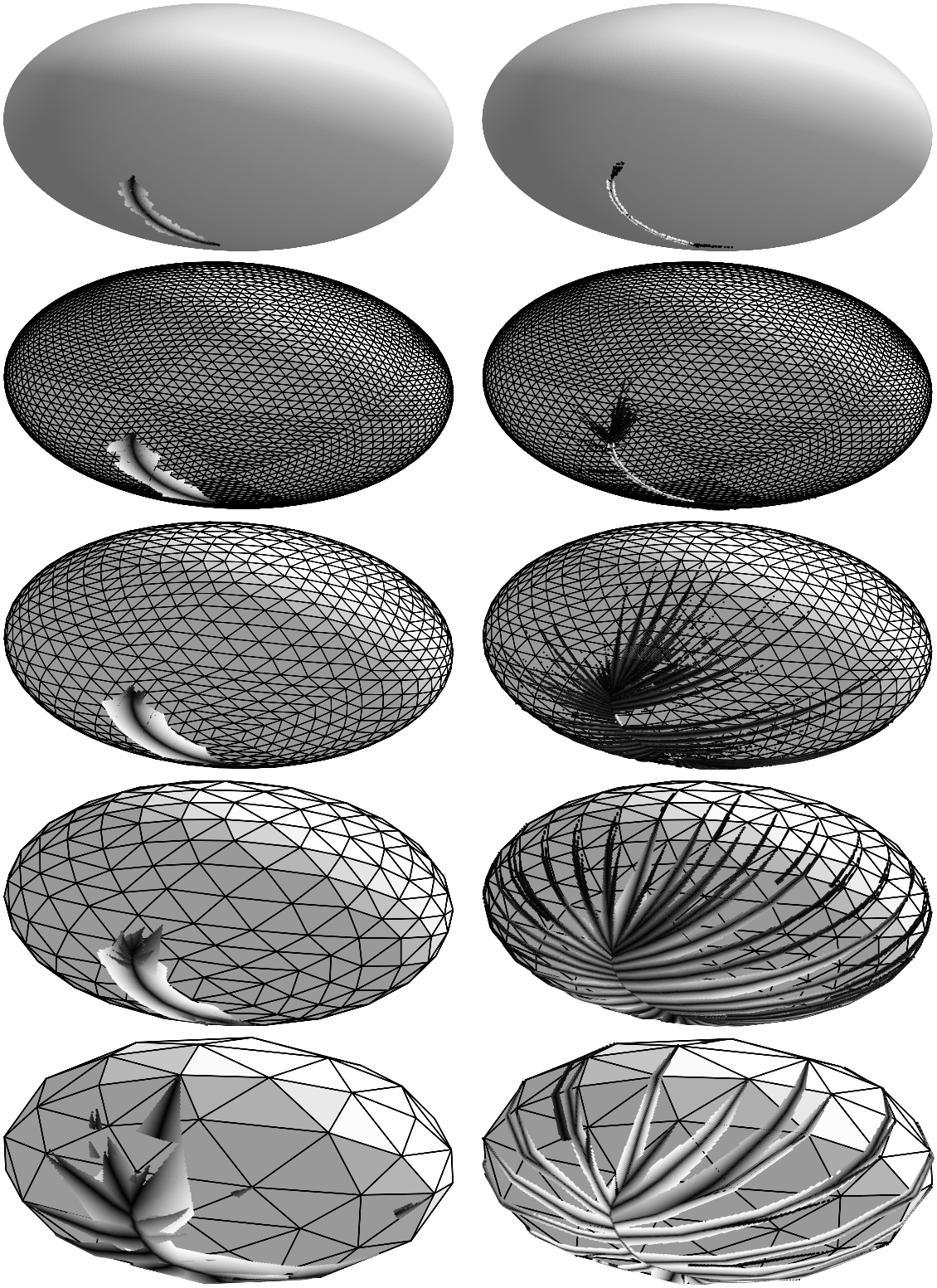

FIGURE 7. Approximations to the cut locus of an ellipsoid from a nonsymmetrically placed starting point. In the left and right columns, $\Delta\theta = 0.5$ and 0.05, respectively. The number of triangles increases as one moves up each column, from 128 to 32768.

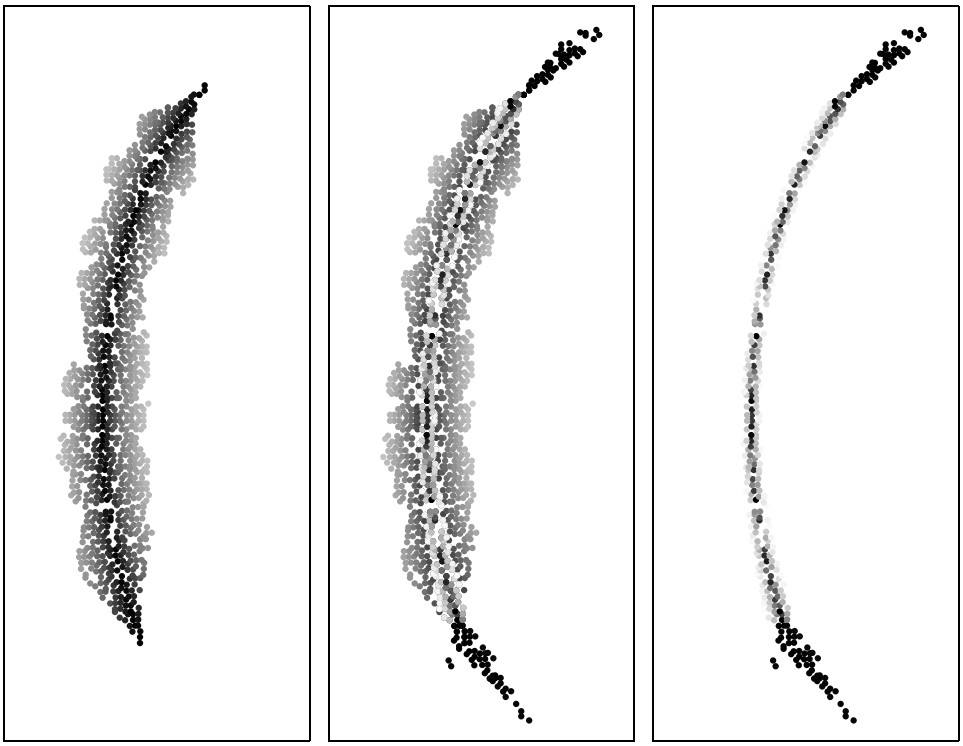

FIGURE 8. A comparison of two approximations to the same cut locus (see the upper two illustrations in Figure 7 and Figure 17). On the left $\Delta\theta$ has been set to 0.5, and on the right it has been set to 0.05. In the middle, both of these approximations have been plotted together. One can clearly see where the cut locus of the smooth surface probably lies.

other candidate sources that are at least $\Delta\theta$ away from $\mathcal{S}_1$. Of those, we choose the closest one, at distance ρ_2 from x. We then compute the number

$$\frac{r\Delta\theta}{|\rho_1 - \rho_2| + r\Delta\theta}. \tag{2–3}$$

This is used to assign a shading to the chosen point. The closer to unity this number is, the closer one should be to the cut locus. An obvious question is "which cut locus?" Thaw is based upon the claim that if one takes the limit of infinitely fine triangulation and $\Delta\theta \to 0$, then it should be the cut locus of the smooth surface. We do not prove this claim in this paper, but the results presented here (Figures 7 and 8, for example) suggest that a suitably formulated version of it should be true.

This process is repeated for many points on every triangle. All of the illustrations of cut loci presented here have been produced in this manner. Of course, the density of points chosen will have an influence on the quality of the illustration. Any gaps in what must otherwise be continuous lines (there is such a gap in the lower left-hand corner of Figure 1) are due to the choice of points of evaluation and are not actual gaps (which a cut locus cannot have).

2.3 Checking against Loki

As we have already mentioned, Loki [Sinclair and Tanaka 02] is unable to compute the cut locus on a surface diffeomorphic to a sphere, since it can only handle a single chart. We can, however, slightly alter Loki and instead compute a subset of the cut locus (up to some distance ρ from the starting point) by computing the exponential map only up to this distance ρ, chosen such that the chart used does not become too singular.

For example, we can parametrize the surface given by (2–1) by the mapping

$$(u, v) \mapsto \big(x(u, v), y(u, v), z(u, v)\big), \tag{2–4}$$

whereby

$$\begin{aligned} x(u,v) &\approx -0.150794 + 0.019671/t(u,v), \\ &\quad + 0.387865 \cdot u/t(u,v) - 0.329491 \cdot v/t(u,v) \end{aligned} \tag{2–5}$$

$$\begin{aligned} y(u,v) &\approx 0.296542 - 0.047757/t(u,v) \\ &\quad - 0.762751 \cdot u/t(u,v) - 0.135715 \cdot v/t(u,v), \end{aligned} \tag{2–6}$$

$$\begin{aligned} z(u,v) &\approx 0.372816 \\ &\quad - 0.303958/t(u,v) + 0.144943 \cdot u/t(u,v), \end{aligned} \tag{2–7}$$

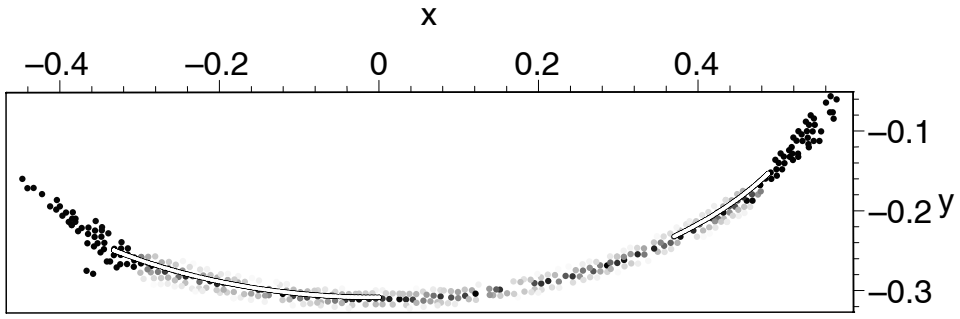

FIGURE 9. Here one can see the coordinates of points that we claim to be close to the cut locus of the smooth surface (2–1). The z-coordinates of these points are given by $z = -0.4 \cdot \sqrt{1 - x^2 - y^2/0.9^2}$. The white curve superimposed upon these points is a subset (including the conjugate points, but only up to a distance of 2.1 from the starting point) of the same cut locus computed using a slightly modified version of Loki [Sinclair and Tanaka 02] provided here as a check.

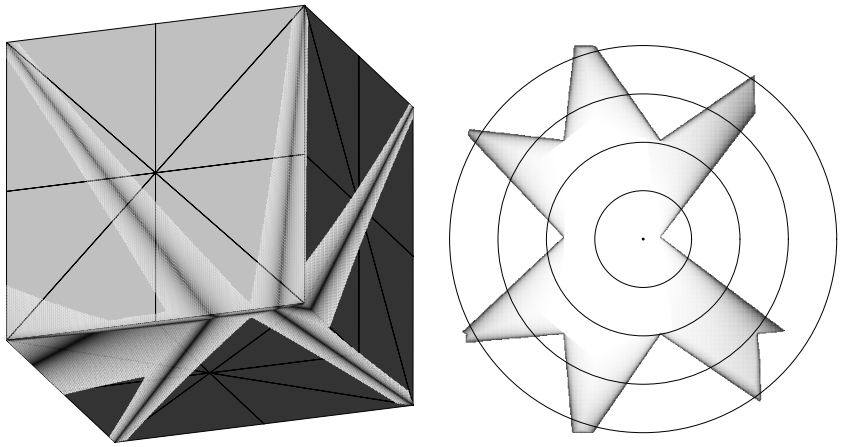

FIGURE 10. The grid used to approximate the cut locus of the surface $x^{1000} + y^{1000} + z^{1000} = 1$ from the point $(0.632455, 0.948683, 1.000000)$ with `N` $= 0$.

and

$$t(u,v) \approx 0.398410 + 0.002086\,v - 0.305684\,u + 0.686150\,u^2 + 0.090094\,v^2. \quad (2\text{–}8)$$

Note that $(0,0) \mapsto (-0.150794,\ 0.296542,\ 0.372816)$, which is the starting point given in (2–2).

Using this parametrization, one can compute the exponential map using Loki up to a distance of $\rho = 2.1$, and therefore compute those points of the cut locus which lie a distance not more than 2.1 from the starting point. See Figure 9 for the actual comparison. Thaw and Loki are seen to agree.

2.4 The Computational Grid

Thaw triangulates adaptively (see Figure 11), using a simple but effective strategy. We begin with a regular triangulation of the unit cube (Figure 10), and then project points onto the surface at hand. Each triangle is then subdivided into four (triangles) by introducing one point on each edge. This is done recursively `N` times. We now have a number of points on the surface, connected to form the triangulation, but we have the freedom to move these points as long as we do not make any of the triangles degenerate; we can do this with the aim of minimizing discretization error. Thaw does this in a simple manner. Remember that subdivision is performed by choosing one point on each edge. Thaw takes each such edge and tries seven equidistant points, linearly interpolating between the endpoints of the edge. Each point is projected onto the surface, and the distance from the surface is recorded. The one which maximizes this distance is then chosen (being considered to be the point at which subdivision will have the most effect).

Note that we do not necessarily guarantee that the sum of angles at any vertex is actually less than or equal

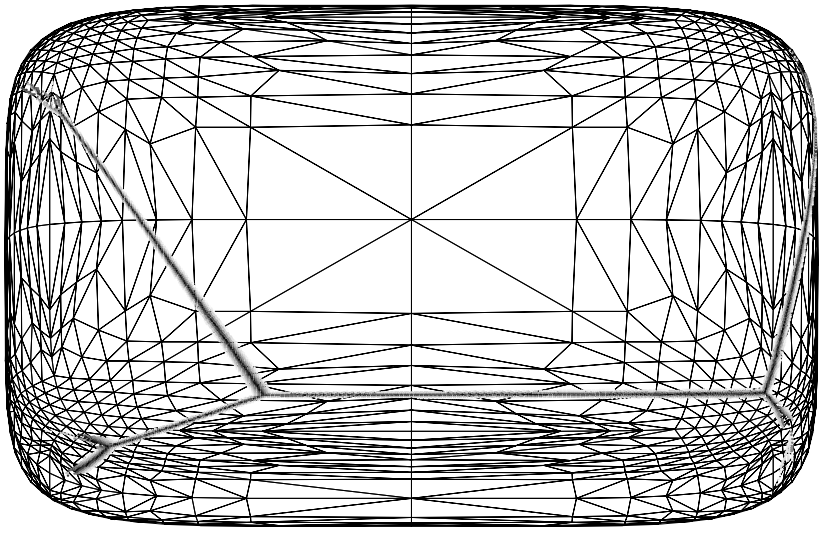

FIGURE 11. The grid used to compute the cut locus of the surface $x^6 + (y/0.7)^6 + (z/0.2)^6 = 1$ from the point $(-0.573639, 0.521062, 0.192467)$ with `N` $= 3$.

to 2π, and that may appear to be a fatal oversight, since it allows gaps to develop in the wavefront (although these will close quickly due to the convexity of the smooth surface being approximated). One must keep two things in mind. One is that the surface being triangulated is convex, so any such errors will disappear rapidly as the number of triangles increases. The other, perhaps more important, observation is that we find that the new error introduced by such large angles is actually (in all the cases we have studied) significantly less than the reduction in error adaptivity brings. In other words, there is a reduction in total error.

3. THE CUT LOCUS OF A ZOLL SURFACE OF REVOLUTION

Our next example will be of a surface whose cut locus is known explicitly. This example is taken from Chapter 4 of [Besse 78], where the cut locus is given in Figures 4.37 and 4.38. We will use the same notation here.

The surface is a Zoll surface of revolution, with

$$h(\cos r) = \cos r \, \frac{\sin^2 r}{2}$$

and an embedding in $\mathbb{R}^3$:

$$\begin{aligned} x(r,\theta) &= \sin r \, \cos\theta, \\ y(r,\theta) &= \sin r \, \sin\theta, \\ z(r,\theta) &= \frac{1}{2}\int_{\pi/2}^{r} (1+\cos u) \\ &\quad \cdot \sqrt{(1-\cos u)(2-\cos u)\,(2+\cos u+\cos^2 u)}\, du. \end{aligned}$$

We compute the cut locus from the point

$$(r_p, \theta_p) = \left(\frac{\pi}{2}, 0\right).$$

As shown in complete detail in Section D of Chapter 4 of [Besse 78], the cut locus is a "Y." This provides us with an excellent nontrivial example with which to test our algorithm. See Figures 12 and 13. Thaw is able to reproduce the known results. It is, however, not our intention to reinvent the wheel, and for that reason the final version of Thaw is not designed to study such surfaces about which essentially everything is already known.

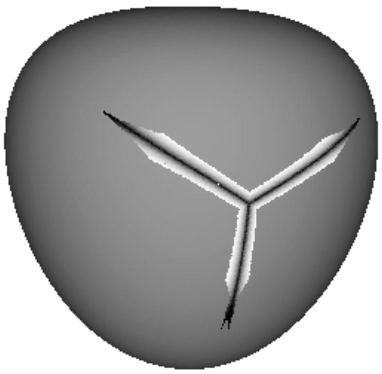

FIGURE 12. The cut locus from a point on a Zoll surface of revolution with $h(\cos r) = \cos r \cdot (\sin^2 r)/2$, corresponding to Figures 4.37 and 4.38 in [Besse 78].

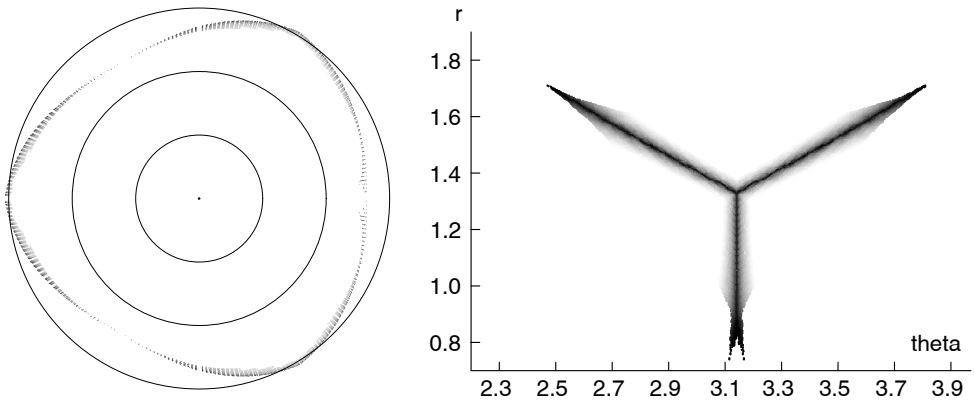

FIGURE 13. Left: A polar representation of the cut locus from a point on a Zoll surface of Revolution, corresponding to Figure 4.37 in [Besse 78]. Note that our figure has an arbitrary rotation. The circles mark the distances 1, 2, and 3 from the starting point. Right: The cut locus from a point on a Zoll surface of revolution, corresponding to Figure 4.38 in [Besse 78]. Note the "goat's beard" near $(\pi, 0.8)$. It is a typical Thaw artifact at a conjugate point.

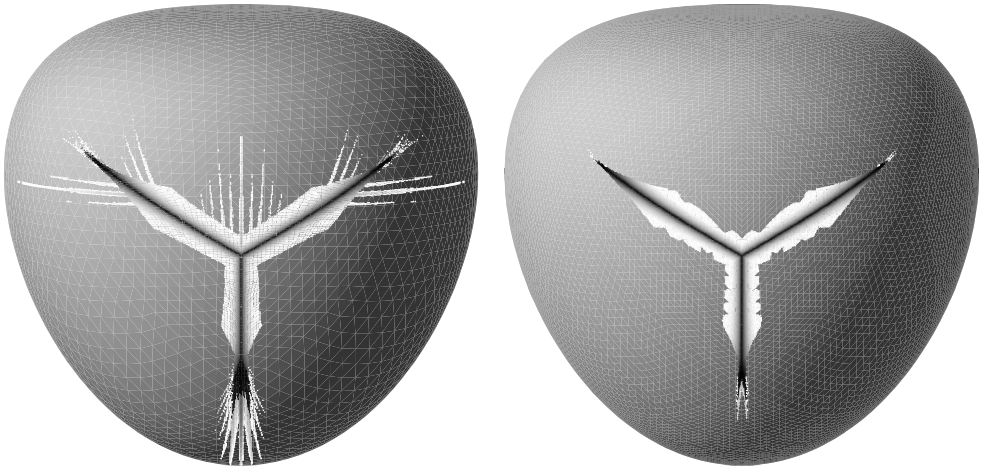

FIGURE 14. The cut locus from a point on a Zoll surface of revolution, computed using different numbers of triangles ($\mathtt{N} = 5$ and $\mathtt{N} = 6$ on the left and right respectively). These illustrations were produced with a different shading algorithm than that used to produce Figure 12. The fact that the "goat's beard" is an artifact is easily seen by studying both of the illustrations above.

3.1 Blurring Artifacts

The "goat's beard" artifact in Figure 13 is perhaps cause for concern, although the nature of the algorithms that we are using is such that artifacts cannot be completely avoided. What is unsatisfactory about this particular artifact is that it appears to be a *clear* indication of a feature which does not in fact exist. The reason for its clarity is the rule that we are using to determine when to leave a point on the surface uncoloured or to apply Expression (2–3). That rule states that there must be at least two source images, separated by at least $\Delta\theta$. This means that neighbouring points which differ only slightly (one sees two sources separated by slightly more, the other by slightly less of an angle) are rendered quite differently, since one of them may be coloured while the other may not be. In the case of Figure 13, points belonging to the "goat's beard" are coloured almost black, while their neighbours are not coloured at all, leaving the impression of a very clear boundary around the artifact.

It was therefore decided to change the colouring algorithm to make artifacts less clear (and therefore less misleading), while leaving the colouring of points that are on the cut locus essentially unchanged. What we do is the following:

1. find the closest source image (at distance ρ_1 and angle θ_1).

2. for every other source image (at distance ρ_j and angle θ_j), compute the colouring

$$\frac{\min\left\{\arccos(\cos(\theta_1-\theta_j))/\Delta\theta, 1\right\}^2 \cdot \rho_1 \cdot \Delta\theta}{10\left[\rho_j - \rho_1 + \rho_1(\Delta\theta/10)\right]} \quad (3\text{–}1)$$

and then take the maximum (darkest) over all j.

The result can be seen in Figure 14. In comparing the left and right illustrations in this figure, one can see that the artifact is surrounded by white "streamers," which quickly disappear as the number of triangles increases. At the same time, the black lines making up the artifact also are seen to shorten dramatically with increasing `N`. It is quite clear from these illustrations that the "goat's beard" is probably an artifact. To that extent, we have achieved what we wanted with (3–1).

4. THE CUT LOCUS OF AN UMBILIC POINT OF A GENERAL ELLIPSOID

It is a well-known fact that the cut locus from an umbilic point on a general ellipsoid consists of only one point: the conjugate point that is the antipodal umbilic point [Klingenberg 82]. Our experience with a Zoll surface of revolution suggests that Thaw has greatest difficulty in identifying conjugate points. It would, therefore, seem that approximating the cut locus from an umbilic point of a general ellipsoid is an excellent test case to examine. We believe that such difficult tests are vitally important in establishing the limits of and also confirming the robustness of our algorithm.

We take the surface

$$\left(\frac{x}{0.2}\right)^2 + \left(\frac{y}{0.6}\right)^2 + z^2 = 1 \quad (4\text{–}1)$$

and the umbilic point at $(-0.115470, 0, 0.816497)$ as our starting point.

Thaw's output is summarized in Figure 15, and its performance in Section 7.1. One notices that Thaw does indeed converge quite slowly, but apparently to the correct result.

5. THE CUT LOCI OF GENERAL ELLIPSOIDS

We begin with two conjectures, proving the first and providing experimental support for the other:

Proposition 5.1. *For any surface given by $(x/a)^{2m} + (y/b)^{2m} + (z/c)^{2m} = 1$, where m is a positive integer, the cut locus from any point (x_0, y_0, z_0) on the surface contains the antipodal point $(-x_0, -y_0, -z_0)$.*

Proof: Let $f : (x, y, z) \mapsto (-x, -y, -z)$ be the reversion. Take a shortest geodesic $\gamma(t)$ from (x_0, y_0, z_0) to $(-x_0, -y_0, -z_0)$. Note that $f(\gamma(t))$ is a shortest geodesic from $(-x_0, -y_0, -z_0)$ to (x_0, y_0, z_0). Then, $\gamma(t)$ and $f(\gamma(-t))$ are two shortest geodesics from (x_0, y_0, z_0) to $(-x_0, -y_0, -z_0)$. □

This first arose (for us) as a conjecture from observations of our experimental data. Once it was proven, we used it as a check in the development of Thaw.

Conjecture 5.2. *On a general ellipsoid $(x/a)^2 + (y/b)^2 + (z/c)^2 = 1, (a > b > c > 0)$, for any point $p; u_1 = a, u_2 = b$, which is not an umbilic point, contained in the upper half, the cut locus of p is an arc on $u_1 = a$ (a curvature line) in the lower half, where (u_1, u_2) are the elliptic coordinates.*

See Definition 3.5.3 in [Klingenberg 82] for a definition of elliptic coordinates. We will use the notation of Section 3.5 of [Klingenberg 82] in the remainder of this section. In that notation, the surface defined by (4–1) becomes

$$\frac{x_0^2}{0.04} + \frac{x_1^2}{0.36} + x_2^2 = \frac{x_0^2}{a_0} + \frac{x_1^2}{a_1} + \frac{x_2^2}{a_2} = 1 \quad (5\text{–}1)$$

(in other words, $(a_0, a_1, a_2) = (0.04, 0.36, 1.0)$).

We choose a starting point at random:

$$(x_0, x_1, x_2) = (-0.151128, -0.350718, 0.295520).$$

This has elliptic coordinates

$$(u_1, u_2) = (0.237164, 0.929661).$$

According to our conjecture, the cut locus should be an arc on $u_1 = 0.237164$. As Figure 16 quite clearly shows, the match is striking.

As our second example, we return to the surface given by (2–1),

$$\frac{x_0^2}{0.16} + \frac{x_1^2}{0.81} + x_2^2 = 1, \quad (5\text{–}2)$$

and also use the same starting point given by (2–2). In elliptic coordinates, this starting point is at

$$(u_1, u_2) = (0.7339745189, 0.9863580085).$$

According to our conjecture, the cut locus should be an arc on $u_1 = 0.7339745189$. Figure 17 also confirms this.

These, and many other entirely similar examples, have convinced us that the conjecture is true.

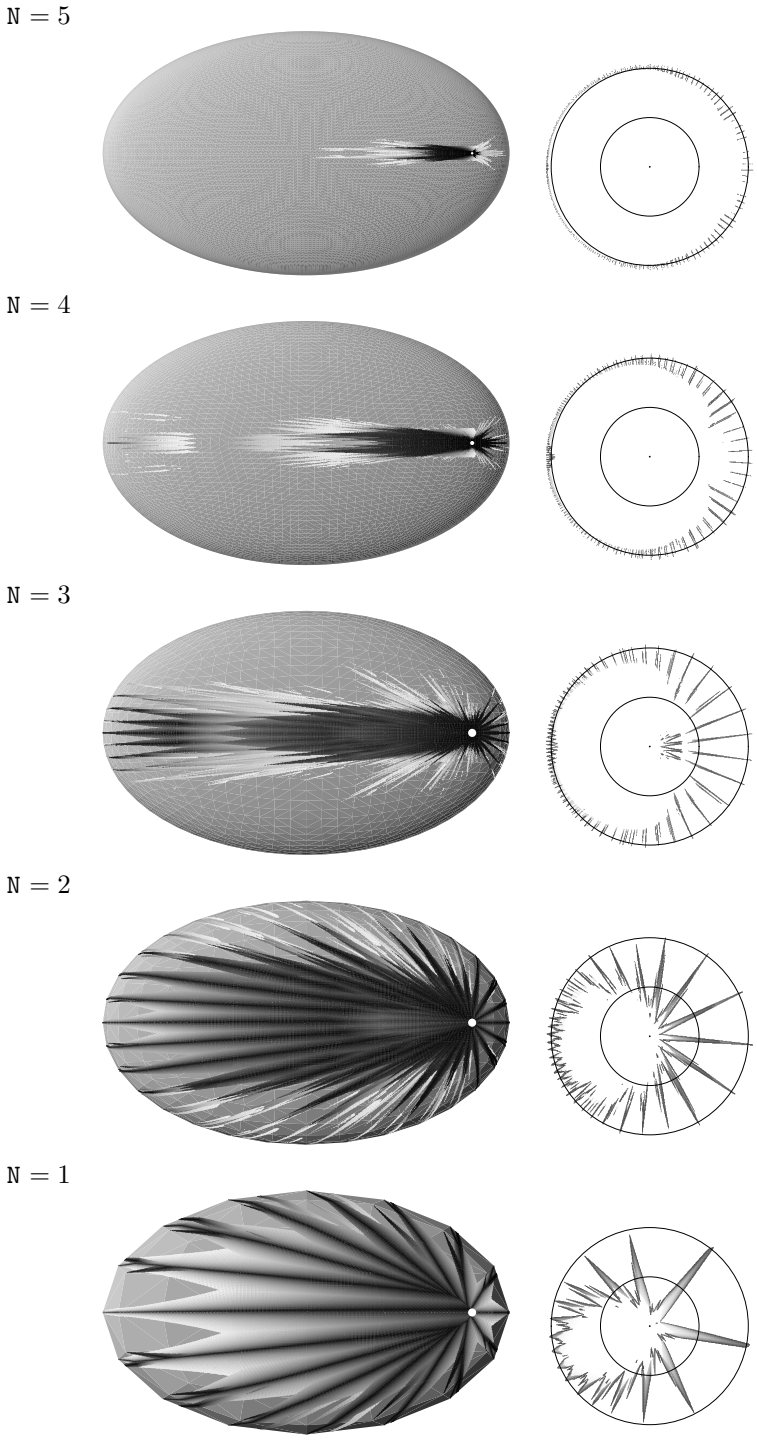

FIGURE 15. Approximations, involving ever greater numbers of triangles, of the cut locus of an umbilic point of the general ellipsoid given by $(x/0.2)^2 + (y/0.6)^2 + z^2 = 1$. It is a well-known fact that the cut locus is a single point, shown as a large white dot.

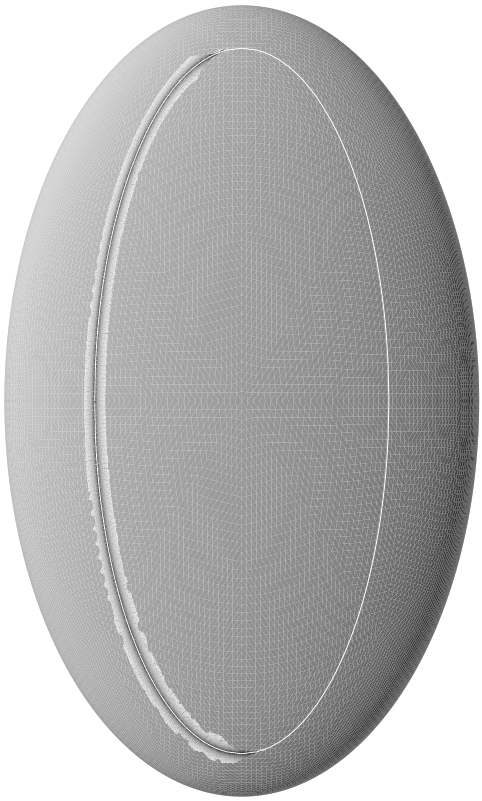

FIGURE 16. The cut locus from the point with elliptic coordinates $(u_1, u_2) = (0.237164, 0.929661)$ on the surface given by $x_0^2/0.04+x_1^2/0.36+x_2^2 = 1$ as computed by Thaw, beneath the white curvature line given by $u_1 = 0.237164$.

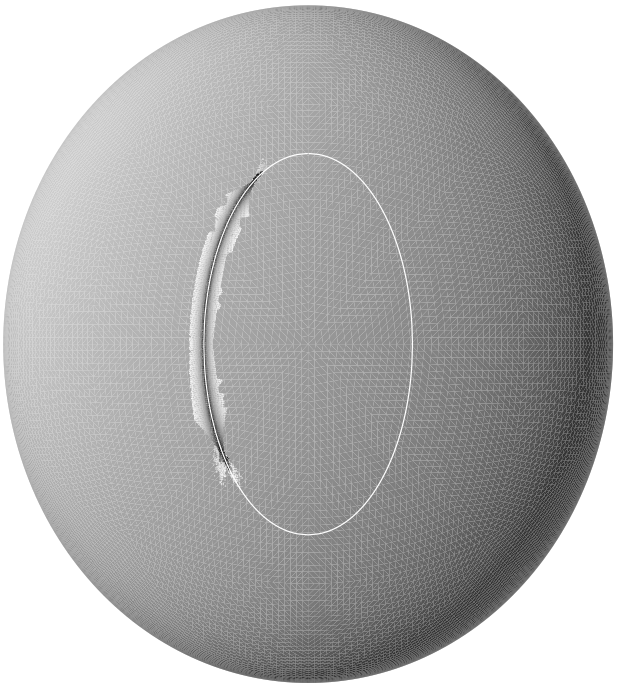

FIGURE 17. The cut locus from the point with elliptic coordinates $(u_1, u_2) = (0.7339745189, 0.9863580085)$ on the surface given by $x_0^2/0.16 + x_1^2/0.81 + x_2^2 = 1$ as computed by Thaw, beneath the white curvature line given by $u_1 = 0.7339745189$.

6. CUT LOCI ON QUARTIC OR EVEN HIGHER-ORDER SURFACES

We hope that Thaw will be of use in investigating the cut locus on surfaces with $n = 2m \geq 4$, about which relatively little is known. Figure 18 illustrates the cut locus from a point on the quartic surface

$$x^4 + y^4 + z^4 = 1.$$

The growth of memory and CPU time with the number of triangles (increasing `N`) of this example will be presented in Section 7.1.

7. THE SOFTWARE

Thaw approximates the cut loci of given surfaces by specifying the three positive coefficients a, b, and c, and the even positive integer power $n = 2m$ in

$$\left(\frac{x}{a}\right)^n + \left(\frac{y}{b}\right)^n + \left(\frac{z}{c}\right)^n = 1. \qquad (7\text{–}1)$$

`N` (the number of subdivisions performed in generating the computational grid—see Section 2.4) and `D` (an integer that controls the density of points at which (2–3) or (3–1) are evaluated) are parameters. The output file `thaw.wrl` is a Virtual Reality Modeling Language file [VRML 97] and is a full three-dimensional description of the surface and cut locus approximation.

Thaw is freely available and can be obtained by contacting R. Sinclair. There are three programs that work together to allow one to work efficiently. These are the following:

- `thaw.c` contains the source of the actual program Thaw. It outputs to two files:
  - `thaw.wrl`, which is an ASCII file containing a three-dimensional VRML description of the output which one can view using an internet browser with suitable plugin. This truly three-dimensional illustration can be rotated and examined at will. The starting point from which the cut locus has been computed is displayed as a small blue sphere on the surface. Its antipodal point (see Proposition 5.1) is displayed as a small green sphere.

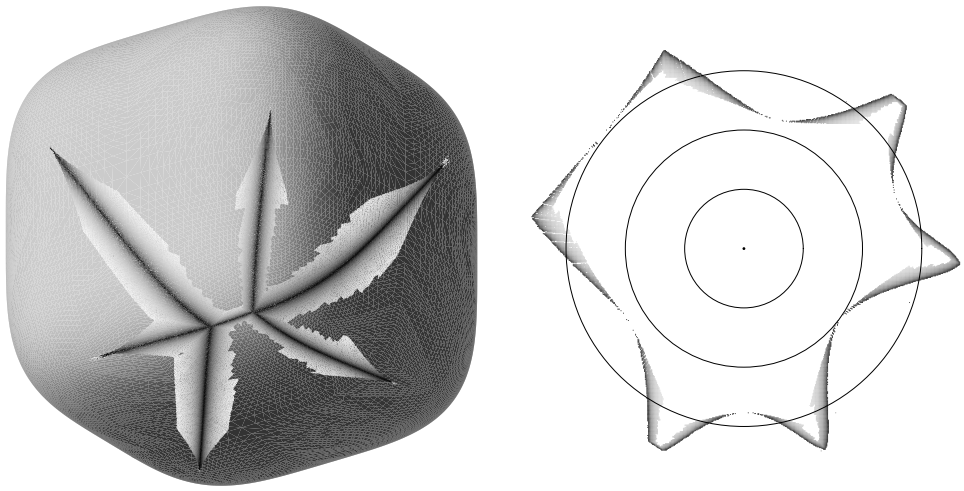

FIGURE 18. The cut locus from the starting point $(0.533843, 0.800764, 0.844080)$ on the surface given by $x^4 + y^4 + z^4 = 1$.

  - `thaw.eps`, which is an encapsulated PostScript [Adobe 85, Adobe 92] illustration of a polar representation of the cut locus.

- `post.c` makes a two-dimensional encapsulated PostScript rendering of Thaw's three-dimensional output (i.e., of `thaw.wrl`) with chosen orientation.

- `rast.c` reduces the size of the PostScript file `thaw.eps` such that it can reasonably be included in a LaTeX [Lamport 94] file.

7.1 Performance

Thaw allocates blocks of 16,384 sources at a time. On our system, such a block fills 1,048,576 bytes (1,024 Kbytes).

We will begin with memory and CPU time data from the most difficult tests performed so far. These were the computation of the cut locus of an umbilic point of a general ellipsoid for various triangulations. See Section 4 and Figure 15.

In examining memory requirements, we must begin with the size of the executable itself, and the memory required to store the triangulation of the ellipsoid. These are shown in Table 1. The data are fit well by $300 + 11.4 \cdot 4^{\mathtt{N}}$ Kbytes, meaning that we can approximate the memory usage of Thaw by

$$300 + (11.4 + C) \cdot 4^{\mathtt{N}} \quad \text{Kbytes}, \tag{7–2}$$

where C is a function of the geometry, as we shall see.

In this case of the cut locus of an umbilic point of a general ellipsoid, the memory usage associated with the allocation of blocks (Table 2) can be approximated by $C = 49.0 \cdot 4^{\mathtt{N}}$ Kbytes.

The CPU time appears to grow alarmingly as `N` increases. See Table 2. For `N` = 5, we have a CPU time of 62,348 seconds (approximately 17 hours). It appears that the algorithm's time complexity is about

$$62.0 + 0.02 \cdot 20.1^{\mathtt{N}}. \tag{7–3}$$

The fact that the growth is exponential is a result of our triangulation (subdivision) algorithm as described in Section 2.4, but one may still feel disappointed. On the one hand, we must remember that Thaw was never expected to be highly efficient. Thaw's strength lies in its ability to produce reasonable quality diagrams in a short

N	memory (Kbytes)
1	360
2	500
3	1,052
4	3,248
5	12,028

TABLE 1. Memory required by the executable and to store the triangulation of an ellipsoid.

N	triangles	CPU time (secs.)	memory (blocks)
1	192	62	1
2	768	83	1
3	3,072	176	4
4	12,288	2,466	13
5	49,152	62,348	49

TABLE 2. Computational costs for approximating the cut locus of an umbilic point of a general ellipsoid.

amount of time, not in its asymptotic behaviour for large N. On the other hand, see [Agarwal et al. 97, Mount 90] for an indication of the complexity issues that we must face. For example, the construction of a superset of the shortest-path edge sequences on the surface of a convex polytope would have a complexity of $O((4^{\mathtt{N}})^6)$ [Agarwal et al. 97]. In other words, Thaw appears to be in the ballpark of what one can expect.

It is, however, worth looking at a more typical example. The illustration of Figure 16 is such a one. There, $\mathtt{N} = 5$ and $\mathtt{D} = 2$. Twenty blocks were allocated, allowing us to calculate the total memory usage as 32,508 Kbytes. The computation required 28,578 seconds (approximately 8 hours) of CPU time. This is not unreasonable, if one is prepared for such a wait.

Finally, we imagine that Thaw will most typically be applied to surfaces such as

$$x^4 + y^4 + z^4 = 1.$$

See Figure 18. The CPU time and memory requirements for this surface are shown in Table 3.

The memory usage associated with the allocation of blocks can be roughly approximated by $41 \cdot 4^{\mathtt{N}}$ Kbytes. Using Equation (7–2), this implies the approximate memory usage

$$300 + 52.4 \cdot 4^{\mathtt{N}} \quad \text{Kbytes} \tag{7–4}$$

and CPU time of

$$61.0 + 0.007 \cdot 22.2^{\mathtt{N}} \tag{7–5}$$

for this surface.

In all of the more intensive computations ($\mathtt{N} \geq 5$), we find that the simultaneous computation of exponential map and cut locus approximation (discussed in Section 2), which allows those triangles which have already been processed to be deallocated during processing, usually results in a reduction in memory usage by a factor of 8 to 9.

N	triangles	CPU time (sec)	memory (blocks)
1	192	61.160	1
2	768	63.100	1
3	3072	139.440	3
4	12288	1,606.620	8
5	49152	35,506.120	26

TABLE 3. Computational costs for approximating the cut locus on the surface $x^4 + y^4 + z^4 = 1$.

8. CONCLUSION

We have shown that it is possible to compute an approximation to the cut locus from a point on a surface via a triangulation of the surface and provided a working software tool: Thaw. What is lacking is a theoretical justification, and this must be the subject of future work.

We have used Thaw to experimentally confirm a new conjecture: the cut loci from points on general ellipsoids are subarcs of curvature lines.

Further work should involve extending the software so that it can handle triangulations in which there are vertices with an angle sum of greater than 2π. This, and an appropriate definition of the cut locus on general polyhedral surfaces, should enable Thaw to illustrate the cut loci of quite complicated two-surfaces (such as ones with high genus).

ACKNOWLEDGMENTS

We would like to thank Professor M. Tanaka of Tokai University, Japan, for having provided us with some preliminary formulations of statements from his ongoing work on the cut loci of ellipsoids of revolution. We were able to compare these with the output of a prototype of Thaw. The lessons learned from this exchange were extremely useful in identifying weaknesses in the software at that stage.

We would like to thank Dr. K. Polthier of the Technical University of Berlin for critical reading of the manuscript, leading to many significant improvements, and also an anonymous reviewer for pointing out the difficulties of trying to extend the definition of the cut locus to negatively curved polyhedral surfaces.

REFERENCES

[Adobe 85] Adobe Systems Inc. *PostScript Language Tutorial and Cookbook*. Reading, MA: Addison Wesley, 1985.

[Adobe 92] Adobe Systems Inc. *Encapsulated PostScript File Format Specification Version 3.0*. Adobe Developer Support PN LPS5002, 1992.

[Agarwal et al. 97] P. K. Agarwal, B. Aronov, J. O'Rourke, and C. A. Schevon. "Star Unfolding of a Polytope with Applications." *SIAM Journal on Computing* 26:6 (1997), 1689–1713.

[Aurenhammer 91] F. Aurenhammer. "Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure." *Computing Surveys* 23:3 (1991), 345–405.

[Berger 00] M. Berger. *Riemannian Geometry During the Second Half of the Twentieth Century*, University Lecture Series, 17. Providence, RI: AMS, 2000.

[Besse 78] A. L. Besse. *Manifolds All of Whose Geodesics Are Closed*, Ergebnisse der Mathematik und ihrer Grenzgebiete, 93. Berlin: Springer-Verlag, 1978.

[Bishop 77] R. L. Bishop. "Decomposition of Cut Loci." *Proceedings of the American Mathematical Society* 65 (1977), 133–136.

[Buchner 77] M. A. Buchner. "Simplicial Structure of the Real Analytic Cut Locus." *Proceedings of the American Mathematical Society* 64:1 (1977), 118–121.

[Buchner 78] M. A. Buchner. "The Structure of the Cut Locus in Dimension Less than or Equal to Six." *Compositio Mathematica* 37 (1978), 103–119.

[Coulson 55] C. A. Coulson. *Waves*, Seventh edition. London: Oliver and Boyd, 1955.

[Degen 97] W. L. F. Degen. "The Cut Locus of an Ellipsoid." *Geometriæ Dedicata* 67 (1997), 197–198.

[Dujardin and Duriez 95] L. Dujardin and T. Duriez. "A Mathematical Model for the Shape of the Hooks of Cestodae." *Acta Biotheoretica* 43 (1995), 217–225.

[Hendricks 92] H. Hendricks. "Sur le cut-locus d'une sous-variété de l'espace euclidien. Négligeabilité." *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique* 315:12 (1992), 1275–1277.

[Kaneva and O'Rourke 00] B. Kaneva and J. O'Rourke. "An Implementation of Chen & Han's Shortest Paths Algorithm." In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pp. 139–146, 2000.

[Kapoor 99] S. Kapoor. "Efficient Computation of Geodesic Shortest Paths." In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, edited by J. S. Vitter, L. Larmore, and T. Leighton, pp. 770–779. New York: ACM Press, 1999.

[Klingenberg 82] W. Klingenberg. *Riemannian Geometry*. Berlin: Walter du Gruyter, 1982.

[Kobayashi 67] S. Kobayashi. "On Conjugate and Cut Loci." In *Studies in Global Geometry and Analysis*, edited by S. S. Chern, pp. 96–122, Studies in Mathematics 4. Washington, DC: MAA, 1967.

[Lambare et al. 96] G. Lambare, P. S. Lucio, and A. Hanyga. "Two-Dimensional Multivalued Traveltime and Amplitude Maps by Uniform Sampling of a Ray Field." *Geophysical Journal International* 125:2 (1996), 584–598.

[Lamport 94] L. Lamport. *LaTeX: A Document Preparation System, User's Guide and Reference Manual*, Second edition. Reading, MA: Addison-Wesley, 1994.

[Mitchell et al. 87] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitrious. "The Discrete Geodesic Problem." *SIAM Journal of Computing* 16:4 (1987), 647–668.

[Mount 90] D. M. Mount. "The Number of Shortest Paths on the Surface of a Polyhedron." *SIAM Journal of Computing* 19:4 (1990), 593–611.

[Ozols 74] V. Ozols. "Cut Loci in Riemannian Manifolds." *Tohoku Mathematical Journal, II. Ser.* 26 (1974), 219–227.

[Pennec 98] X. Pennec. "Toward a Generic Framework for Recognition Based on Uncertain Geometric Features." *Videre: Journal of Computer Vision Research* 1:2 (1998), 58–87.

[Polthier and Schmies 98] K. Polthier and M. Schmies. "Straightest Geodesics on Polyhedral Surfaces." In *Mathematical Visualization*, edited by H. C. Hege and K. Polthier, pp. 135–150. Berlin: Springer, 1998.

[Polthier and Schmies 99] K. Polthier and M. Schmies. "Geodesic Flow on Polyhedral Surfaces." In *Data Visualization '99, Proceedings of the Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization in Vienna*, edited by E. Gröller, H. Löffelmann, and W. Ribarsky, pp. 179–188. Vienna: Springer-Verlag, 1999.

[Sakai 96] T. Sakai. *Riemannian Geometry*, Translations of Mathematical Monographs, 149. Providence, RI: AMS, 1996.

[Sava and Fomel 98] P. Sava and S. Fomel. "Huygens Wavefront Tracing: A Robust Alternative to Ray Tracing." *Society of Exploration Geophysicists Extended Abstracts* ST-17 (1998), 1961–1964.

[Sharir and Schorr 86] M. Sharir and A. Schorr. "On Shortest Paths in Polyhedral Spaces." *SIAM Journal of Computing* 15:1 (1986), 193–215.

[Sinclair and Tanaka 02] R. Sinclair and M. Tanaka. "Loki: Software for Computing Cut Loci." *Exper. Math.* 11:1 (2002), 1–25.

[Volkov and Podgornova 71] Ju. A. Volkov and E. G. Podgornova. "The Cut Locus of a Polyhedral Surface of Positive Curvature." *Ukrain. Geometr. Sb.* 11 (1971), 15–25.

[VRML 97] VRML97: International Standard ISO/IEC 14772-1:1997.

[Wolter 79] F. -E. Wolter. "Distance Function and Cut Loci on a Complete Riemannian Manifold." *Archiv der Mathematik* 32 (1979), 92–96.

[Wolter and Friese 00] F. -E. Wolter and K. -I. Friese. "Local and Global Geometric Methods for Analysis Interrogation, Reconstruction, Modification and Design of Shape." In *Proceedings of Computer Graphics International 2000*, pp. 137–151. Los Alamitos, CA: IEEE Computer Society, 2000.

Jin-ichi Itoh, Department of Mathematics, Faculty of Education, Kumamoto University, Kumamoto 860–8555, Japan (j-itoh@kumamoto-u.ac.jp)

Robert Sinclair, Department of Mathematics & Statistics, The University of Melbourne, Parkville, Victoria 3010, Australia (R.Sinclair@ms.unimelb.edu.au)

Received June 11, 2002; accepted February 12, 2004.