Computing Geodesic Paths on Manifolds

R. Kimmel and J.A. Sethian *

Dept. of Mathematics
and
Lawrence Berkeley National Laboratory
University of California, Berkeley 94720

Abstract

The Fast Marching Method [9] is a numerical algorithm for solving the Eikonal equation on a rectangular orthogonal mesh in $O(M \log M)$ steps, where M is the total number of grid points. In this paper we extend the Fast Marching Method to triangulated domains with the same computational complexity. As an application, we provide an optimal time algorithm for computing the geodesic distances and thereby extracting shortest paths on triangulated manifolds.

1 Introduction

Sethian's Fast Marching Method [9], is a numerical algorithm for solving the Eikonal equation on a rectangular orthogonal mesh in $O(M \log M)$ steps, where M is the total number of grid points in the domain. The technique hinges on producing a numerically consistent approximation to the operators in the Eikonal equation that select the correct viscosity solution; this is done through the use of upwind finite difference operators. The structure of this upwinding is then used to systematically construct the solution to the Eikonal equation through an optimal ordering of points during the update. The optimal ordering is executed using a heap operator to extract the next point in the update sweep. As such, the technique is a reminiscent of Dijkstra's method [4]; however, the resulting approximation is consistent in that it produces the correct shortest path on an orthogonal grid. For details about Fast Marching Methods, see [9, 10].

Barth and Sethian [2] have recently constructed operators for viscosity solutions to both the Eikonal equation and Hamilton-Jacobi equations on arbitrary

*This work was supported in part by the Applied Mathematical Science subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098, and the Office of Naval Research under under grant FDN00014-96-1-0381.

triangulated domains. These operators exploit the upwind nature to construct the correct entropy-satisfying weak solutions.

In this paper, we extend these ideas and build a Fast Marching Method for triangulated domains. As an application, we provide an optimal algorithm for computing the geodesic distances and thereby extracting shortest paths on triangulated manifolds.

The outline of this paper is as follows. First, we review the Fast Marching Method for orthogonal grids. Then, for motivational reasons, we analyze the structure of this method on a triangulated planar grid constructed directly from an orthogonal grid. We then follow with a general procedure for computing the solution of the Eikonal equation on arbitrary acute triangulated domains, followed by an extension to general (non-acute) triangulations. As an application, we compute geodesic distances and minimal geodesic paths on manifolds.

2 The Fast Marching Method on Orthogonal Grids

Here, we briefly review the Fast Marching Method for computing the solution to the Eikonal equation, for details, see [9]. The goal is to solve the equation

$$|\nabla T| = \mathcal{F}(x, y). \tag{1}$$

The key idea is to build an approximate to the gradient term which correctly deals with the development of corners and cusps in the solution. It is well–known that the above Eikonal equation becomes non–differentiable, and an appropriate weak solution must be built. The appropriate weak solution comes from satisfying the entropy condition; see, for example, [8]. The crucial point in this (or any such appropriate) numerical scheme is the correct direction of the upwinding and treatment of sonic points. For details and an extensive review, see [10]. The Fast Marching Method exploits this idea carefully considering the nature of upwind, entropy–satisfying approximations to the Eikonal equation.

In more detail, consider one particular upwind approximation to the gradient, given by (see [7])

$$\left[\begin{array}{c} \max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + \\ \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2 \end{array} \right]^{1/2} = \mathcal{F}_{ij}, \tag{2}$$

The central idea behind Fast Marching Methods is to systematically advance the front in an upwind fashion to produce the solution T. The key is the observation that the upwind difference structure of Eq. (2) means that information propagates "one way", that is, from smaller values of T to larger values. Hence, the algorithm rests on "solving" Eq. (2) by building the solution outward from the smallest T value. The algorithm is made fast by confining the "building zone" to a narrow band around the front, motivated by the narrow band introduced in Chopp [3], used in recovering shapes from images in Malladi, Sethian and Vemuri [6], and analyzed extensively by Adalsteinsson and Sethian in [1]. The idea is to sweep the front ahead in an upwind fashion by considering a set of points in a narrow band around the existing front, and to march this narrow band forward, freezing the values of existing points and bringing new ones into the narrow band structure. The key is in the selection of *which* grid point in the narrow band to update.

The Fast Marching Method algorithm is as follows: First, we tag points in the initial conditions as *Alive*. We then tag as *Close* all points one grid point away. Finally, we tag as *Far* all other grid points. Then the loop is

1. Begin Loop: Let *Trial* be the point in *Close* with the smallest T value.
2. Add the point *Trial* to *Alive*; remove it from *Close*
3. Tag as *Close* all neighbors of *Trial* that are not *Alive*: If the neighbor is in *Far* remove it from that list and add it to the set *Close*.
4. Recompute the values of T at all neighbors according to Eqn. (2) by solving the quadratic equation, using only values of points that are *Alive*.
5. Return to top of Loop;

This algorithm works because the process of recomputing the T values at upwind neighboring points cannot yield a value smaller than any of the accepted (*Alive*) points. This is known as a "monotone property", and means that we can march the solution outward, always selecting the narrow band grid point with minimum trial value for T, and readjusting neighbors, without having to "go back and correct an accepted value," see Figure 1. Another way to look at this

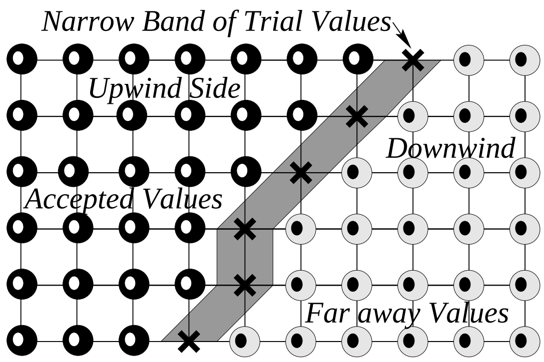


Figure 1: Upwind construction of accepted values.

is that each minimum trial value begins an application of Huyghen's principle, and the expanding wave front touches and updates all others. The speed of the algorithm comes from a heapsort technique to efficiently locate the smallest element in the set *Close*. If there are M total points in the computational domain, then, assuming no work for locating the smallest element in the set *Close*, we have a computational complexity of $O(M)$. Since the heap can be re-ordered in $O(\log M)$ steps, this yields a computational complexity of $O(M \log M)$. For more details, see [9, 10].

The above technique is strongly reminiscent of Dijkstra's method, which allows one to compute minimum costs of paths on networks. That technique, which is also $O(M \log M)$, is numerically inconsistent; given two points on the graph, it produces the network minimum (Manhattan) length, which may not be optimal. The Fast Marching Method is a consistent approximation to the continuous partial differential gradient operator, thus it provides "sub-grid" resolution, and hence finds the true shortest path.

Before proceeding, it is instructive to say a few words about the actual update procedure involved in "solving" Eqn. 2. Imagine a uniform square grid and suppose that the goal is to update the value of T at the center point i, j. We label the values of T at the surrounding grid points $T_A = T_{i-1,j}$, $T_B = T_{i+1,j}$, $T_C = T_{i,j-1}$, and $T_D = T_{i,j+1}$ (see Figure 2). Some of the values may be infinite, corresponding to "*Far*" values.

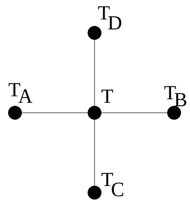


Figure 2: Construction of upwind solution on orthogonal mesh.

Standing at the center point i, j, we attempt to solve the quadratic given by each possibility. For example, we refer to possible contributors A and C and assume, without loss of generality, that $T_A \le T_C$. We attempt to solve

$$(T - T_A)^2 + (T - T_C)^2 = h^2 \mathcal{F}_{i,j}^2,$$

where h is the uniform grid spacing. Two possibilities are:

- There is a real solution T, with $T > T_A$ and $T > T_C$, to the quadratic
$$(T - T_A)^2 + (T - T_C)^2 = h^2 \mathcal{F}_{i,j}^2.$$
- There is a real solution T, with $T > T_A$ and $T \le T_C$, to the one-dimensional update (degenerate quadratic)
$$(T - T_A)^2 = h^2 \mathcal{F}_{i,j}^2.$$

For each possible up-down/left-right pair, we construct all possible real solutions; we then accept as the updated point the one that produces the smallest value of T. Judicious programming and careful attention to "*Far*" values makes the above search extremely fast. This is the Fast Marching Method described in [9, 10].

3 Fast Marching on a Particular Triangulated Planar Domain

Our goal now is to extend this method to triangular domains. In order to do so, we shall build a monotone update procedure on the triangulated mesh. As motivation, in this section we consider the obvious triangulation of a square grid in the plane. Imagine the triangulation given in Figure 3.

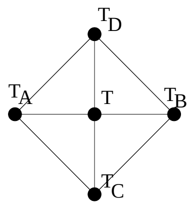


Figure 3: Simple triangulation for building a monotone update operator.

If we consider the possible contributors T_A and T_C, and look at the triangle formed by the points i, j, $i-1, j$ and $i, j-1$, we can easily write down the equation of the plane determined by the known values T_A and T_C, as well as the unknown value T, namely

$$\left[\frac{T-T_A}{h}\right] x + \left[\frac{T-T_C}{h}\right] y + T = z.$$

Computing the gradient, we then want to select a value of T such that

$$\left[\frac{T-T_A}{h}\right]^2 + \left[\frac{T-T_C}{h}\right]^2 = \mathcal{F}_{ij}^2.$$

In other words, we are tilting the plane by a value T at the center point i, j in order to have a gradient magnitude equal to $\mathcal{F}$. We note that this is the exact same construction as the one produced by the "orthogonal" construction. Note also that in this case the gradient vector, with origin at the center point, always points into the triangle from which it is updated. This is not necessarily the case for an arbitrary (obtuse) triangle. In order to establish monotonicity we will need to verify this condition.

The inability to solve the above quadratic corresponds to an inability to tilt at an appropriate angle, and the requirement that the solution T be greater than the contributors means that the solution is always constructed in an upwind manner. Using the same update rules as above, and the heap structure to maintain a list of grid points of "*Close*" points, this provides a method for executing the Fast Marching Method on such a simple triangulation.

4 Fast Marching Methods on Triangulated Domains

Our goal now is to extend this idea to an arbitrary triangulation. Here, we are essentially following the construction of upwind approximations to the gradient on triangulated meshes developed by Barth and Sethian in [2].

4.1 A Construction for Acute Triangulations

We start with an acute triangulation, and consider the triangulation around the grid point given in Fig. 4.

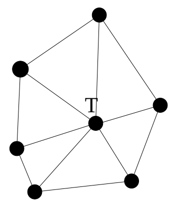


Figure 4: Acute triangulation around center grid point.

A large number of triangles may share the center vertex. Our procedure, motivated by the simple triangulation in the previous section, is to compute a possible value for T from each triangle that includes the center point as a vertex. Since several triangles can produce admissible values for T, we must select an appropriate value. There are several possibilities. In our examples we have chosen the one that produces the smallest new value for T; this will correspond to an algorithm similar to the one taken above. More elaborate upwind constructions on triangulated meshes are given in [2].

We now construct a simple update procedure for one non obtuse triangle ABC in which the point to update is C. We should verify that the update is from within the triangle, i.e. the altitude h should be inside the non obtuse triangle ABC. See Fig. 5.

This means that we search for $t = EC$ such that

$$\frac{t-u}{h} = \mathcal{F}.$$

Denote $a = BC$ and $b = AC$; we have by similarity that $t/b = DF/AD = u/AD$, thus $CD = b - AD = b - bu/t = b(t-u)/u$. Next, by the Law of Cosines: $BD^2 = a^2 + CD^2 - 2a\,CD\cos\theta$,and by the Law of Sines: $\sin\phi = \frac{CD}{BD}\sin\theta$.Now, using the right angle triangle CBG we have

$$h = a\sin\phi = a\frac{CD}{BD}\sin\theta = \frac{a\,CD\,\sin\theta}{\sqrt{a^2 + CD^2 - 2a\,CD\cos\theta}}. \tag{3}$$

We end up with the quadratic equation for t:

$$(a^2 + b^2 - 2ab\cos\theta)t^2 + 2bu(a\cos\theta - b)t + b^2(u^2 - \mathcal{F}^2 a^2 \sin^2\theta) = 0. \tag{4}$$

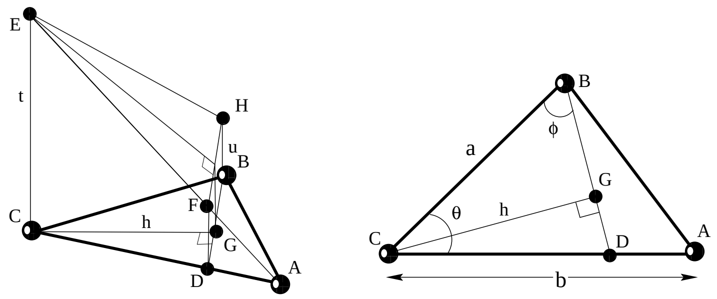


Figure 5: Given the triangle ABC, such that $u = T(B) - T(A)$, find $T(C) = T(A)+t$ such that $(t-u)/h = \mathcal{F}$. Left: A perspective view of the triangle stencil supporting the $T()$ values that form a tilted plane with a gradient magnitude equal to $\mathcal{F}$. Right: The trigonometry on the plane defined by the triangle stencil.

The solution t must satisfy $u < t$, and should be updated from within the triangle, namely:

$$a \cos\theta < \frac{b(t-u)}{t} < \frac{a}{\cos\theta}. \tag{5}$$

Thus, the update procedure is given as follows:

If $u < t$ **and** $a\cos\theta < \frac{b(t-u)}{t} < \frac{a}{\cos\theta}$ **then** $T(C) = \min\{T(C), t + T(A)\}$;

else $T(C) = \min\{T(C), b\mathcal{F} + T(A), a\mathcal{F} + T(B)\}$.

It is easy to verify that this equation is a finite difference approximation to the Eikonal equation. It is monotone by construction, consistent, converges to the viscosity solution, and can be used to extend the Fast Marching Method to acute triangulated domains.

4.2 Extension to General Triangulations

Finally, we note that we have required an acute triangulation. This is so that any front entering the side of a triangle has two points to provide values before the third is computed. In other words, for monotonicity, we restrict the update to come from within the triangle, i.e., the gradient of the solution at a grid point should point into the triangle from which it is updated. The most straightforward way to enforce this requirement is to build a non obtuse triangulation, thus making sure that the grid captures all incoming fronts.

In the case of an obtuse triangulation, the support may only include a limited (acute) section of incoming wavefronts. One approach to handle non-acute triangulations, which we now describe, is to locally build numerical support at obtuse angles by splitting these angles in a special way. An obtuse angle at vertex A can be updated by its neighboring points in a consistent way only at a limited section of upcoming fronts. Connecting the vertex to any point in this section splits the obtuse angle into two acute ones. The idea is to extend this section by recursively unfolding the adjacent triangle(s), until a new vertex B is included in the extended section. Then, the vertices are connected by a virtual directional edge from B to A (i.e. A may be updated by B). The length of the edge AB is equal to the distance between A and B on the unfolded triangles plane.

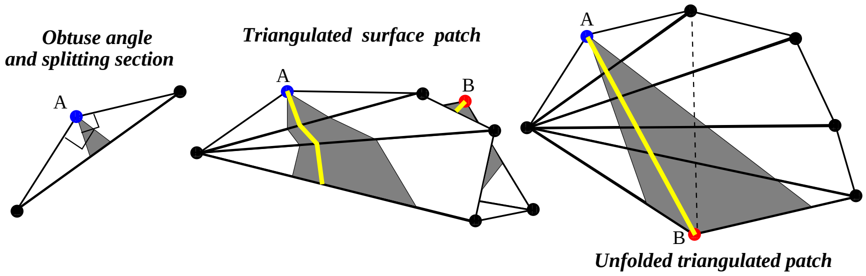


Figure 6: Left: The initialization of the construction for the splitting section. Middle: A triangulated surface patch. Right: The unfolded patch and the splitting section expansion up to the first vertex B, and the virtual edge connecting the two vertices AB.

We can perform a complexity analysis to check on the cost of this virtual node capturing. Let h_{max}, h_{min} be the maximal and minimal altitudes, respectively. I.e. triangles altitudes with maximal and minimal length, let θ_{max} be the maximal obtuse angle, and denote $\alpha = \pi - \theta_{max}$ the angle of the extended section, let θ_{min} be the minimal (acute) angle for all triangles. Let e_{max} be the length of the longest edge, and let l be the length of the virtual directional edge (AB in the above example). Furthermore, assume that α, θ_{min} are small enough angles so that $\sin\alpha \approx \tan\alpha \approx \alpha$.

Then, the angular width of the narrower section is $\alpha = \pi - \theta_{max}$, so that we have $\sin\frac{\alpha}{2} \leq \frac{e_{max}}{2l}$. This relation, for small α angles, yields $l \leq \frac{e_{max}}{\alpha}$. Denote by $l_{max} = e_{max}/\alpha$.

The maximal area of the extended sections is bounded from above by $a_{max} = \frac{e_{max}^2}{2\alpha}$, and the minimal area of an unfolded triangle is bounded from below by $a_{min} = \frac{(h_{min}\alpha)^2\theta_{min}}{2}$, see Figure 7. Therefore, the number of triangles that are needed to be unfolded before a vertex is found in the extended section is bounded by the ratio of these areas

$$m \quad = \quad \frac{a_{max}}{a_{min}} = \frac{e_{max}^2}{\theta_{min} h_{min}^2 \alpha^3}. \tag{6}$$

The accuracy of the first order scheme for acute triangles is of $O(h_{max}) \approx O(e_{max})$. The accuracy for the obtuse case with the above construction becomes

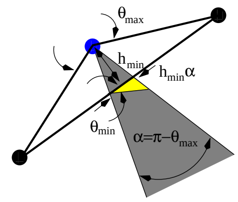


Figure 7: The smallest possible area of one triangle covered by the longest possible extended section bounds the number of unfolded triangles. The triangle with a_{min} is the bright shaded one.

$O(l_{max}) = O(e_{max}/(\pi - \theta_{max}))$, as expected. In the worst case scenario, the scheme accuracy depends on the largest edge and the widest angle.

The complexity of this virtual construction does not change substantially the operation count of the underlying algorithm. The construction of the virtual directional edges includes unfolding triangles for each obtuse angle until a vertex is detected in the extended splitting section. Since the number of unfolded triangles is bounded by a constant, the construction of the virtual directional edges takes $O(M)$, and running time complexity is still optimal $O(M \log M)$.

5 Construction of Minimal Geodesics

We can use this algorithm to compute distances on triangulated manifolds, and hence construct minimal geodesics. First, we solve the Eikonal equation with speed $\mathcal{F} = 1$ on the triangulated surface to compute the distance from a source point; we then backtrack along the gradient of the arrival time field by solving the ordinary differential equation

$$\frac{dX(s)}{ds} = -\nabla T,$$

where $X(s)$ traces out the geodesic path. We use second order Huen's integration method on the triangulated surface with a switch to a first order scheme at sonic points in the gradient. When integrating within a given triangle, its three neighboring triangles support the computation and are used to interpolate T as a second order polynomial whose six coefficients are computed from those given T values at the vertices.

Figure 8 presents a perspective view of the triangulation and shortest paths for two surfaces. Figure 8a shows shortest paths on regular triangulation of the surface given by the function $z(x, y) = 0.45 \sin(2\pi x) \sin(2\pi y)$ on $[0, 1] \times [0, 1]$, for grid size of 50×50. The minimal geodesics are painted on the triangulated surface and projected to the xy plane on the left. Figure 8b gives a polyhedron example, in which a different speed function $\mathcal{F}$ was assigned to each side, causing a Snell Law effect along the edges. The sped $\mathcal{F}$ is 2 at the close side with the start point, 1 at the second top side, and 4 at the side of the destination point.

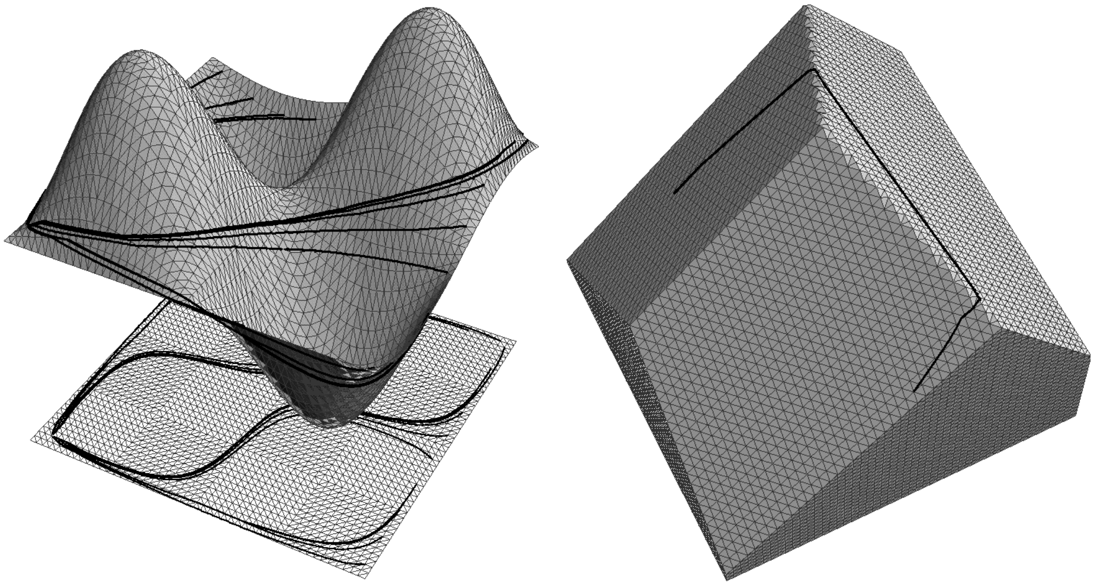


Figure 8: Computing minimal weighted geodesics on triangulated surfaces.

Finally, we show two additional computations to illustrate the algorithm performance on manifolds with an underlying non-acute (obtuse) triangulation. Figure 9 shows the computation of minimal geodesics on a torus genus one object, in which some shortest paths in fact cut through the middle.

The equi-distance curves are shown, as well as the shortest paths. In Figure 10, we compute geodesic distances on a synthetic head.

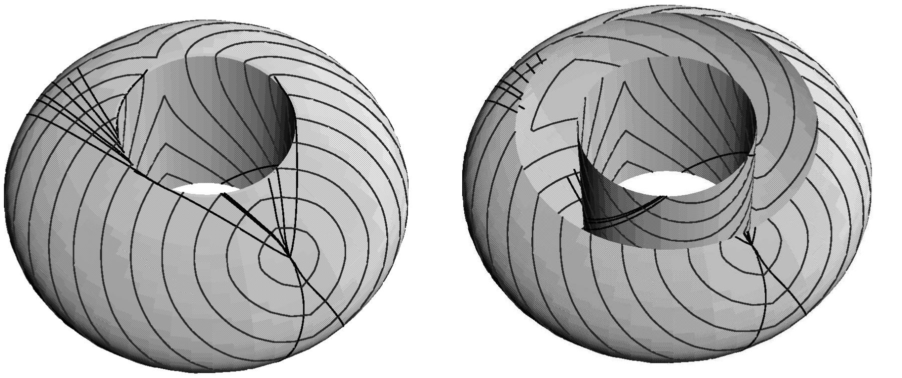
Acknowledgments: We thank T. Barth, L.C. Evans, and A. Vladimirsky for helpful discussions. All calculations were performed at the University of California at Berkeley and the Lawrence Berkeley Laboratory.

References

[1] Adalsteinsson, D., and Sethian, J.A., *A fast level set method for propagating interfaces*, Jour. Comp. Phys., Vol. 118, pp. 269–277, 1995.

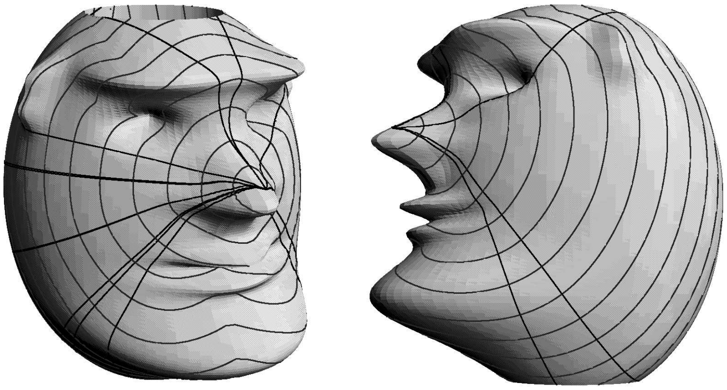
[2] Barth T., and Sethian, J.A., *Numerical Schemes for the Hamilton-Jacobi and Level Set Equations on Triangulated Domains*, submitted for publication, J. Comp. Phys., 1997.

[3] Chopp, D. L., *Computing minimal surfaces via level set curvature flow*, Journal of Computational Physics, Vol. 106, pp. 77–91, 1993.



Full View of Shortest Paths Cutaway View

Figure 9: Shortest Paths on a bead (genus one).



Side View Alternate Side View

Figure 10: Shortest Paths on Synthetic Head

[4] Dijkstra, E.W., *A Note on Two Problems in Connection with Graphs*, Numerische Mathematic, 1:269–271, 1959.

[5] R Kimmel, A Amir, and A M Bruckstein. Finding shortest paths on surfaces using level sets propagation. IEEE Trans. on PAMI, 17(1):635-640, 1995

[6] Malladi, R., Sethian, J. A., and Vemuri, B. C., *Shape modeling with front propagation: A level set approach*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 2., 1995.

[7] Rouy, E., and Tourin, A., *A Viscosity Solutions Approach to Shape–from–shading*, SIAM. J. Numer. Anal., 29(3):867–884, 1992.

[8] Sethian, J.A., *Curvature and the evolution of fronts*, Commun. in Math. Physics, Vol. 101, pp. 487–499, 1985.

[9] Sethian, J.A., *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93, 4, 1996.

[10] Sethian, J.A., *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Science*, Cambridge University Press, 1996.