Title
Fast Marching Methods for Computing Distance Maps and Shortest Paths

Author
Kimmel, Ron

Publication Date
1996-02-01

LBL-38451
UC-405
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics Division

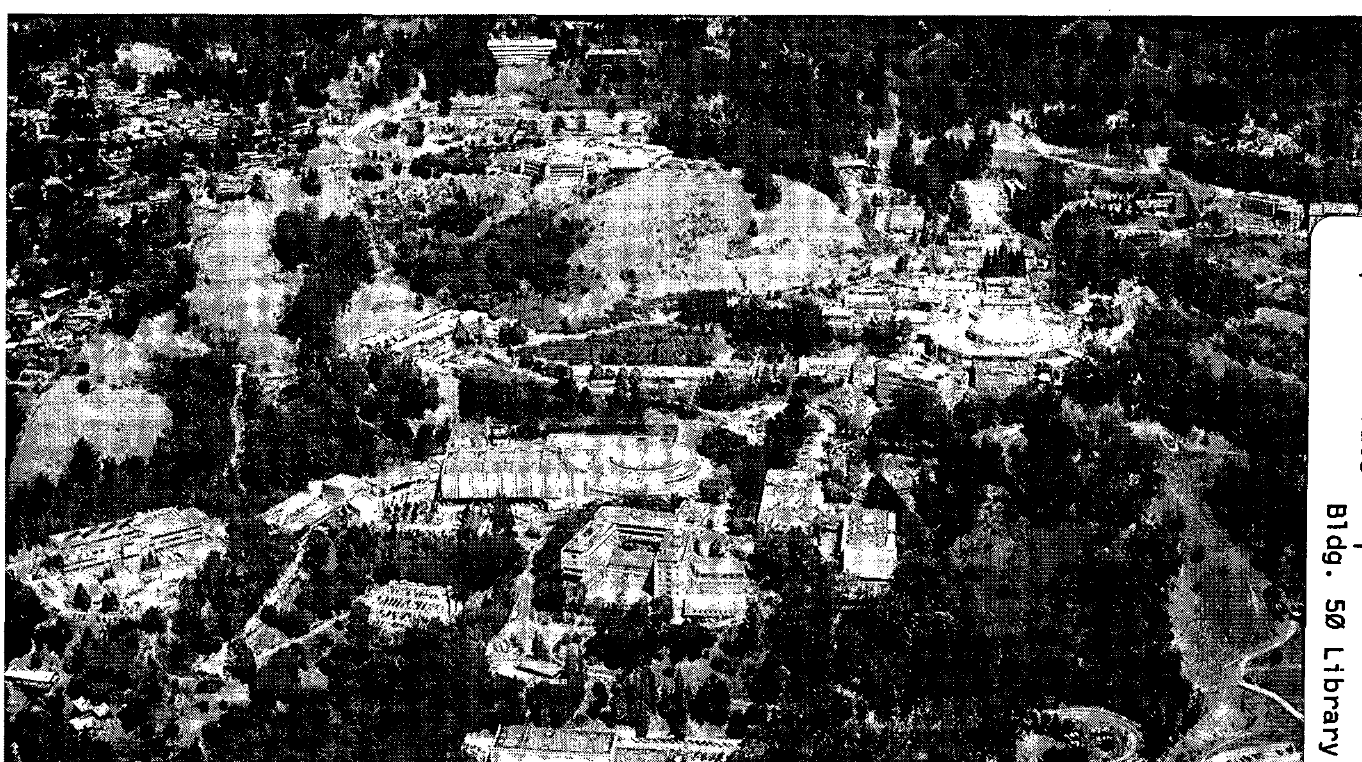
Mathematics Department

To be submitted for publication

Fast Marching Methods for Computing Distance Maps and Shortest Paths

R. Kimmel and J.A. Sethian

February 1996



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

LBL-38451

FAST MARCHING METHODS FOR COMPUTING DISTANCE MAPS AND SHORTEST PATHS*

Ron Kimmel
Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

J.A. Sethian
Department of Mathematics and Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720

February 1996

* This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098.

Fast Marching Methods for Computing Distance Maps and Shortest Paths *

Ron Kimmel[||] **J.A. Sethian**[**]

Abstract

In this paper, we present a fast technique for computing both paths of minimal cost, and minimal geodesics on surfaces. The technique exploits the fast marching method introduced in [15, 16] for solving the Eikonal equation and its extension to general static Hamiltonian-Jacobi equations given in [1]. The solution to the appropriate static Hamiltonian provides the arrival time of the shortest path, and is then coupled to high order ordinary differential equation solvers to construct the path itself. The resulting technique is an $\mathcal{O}(N \log N)$ procedure, where N is the total number of grid points. The technique works without change in any number of space dimensions. We provide upwind approximation schemes for the relevant Hamiltonian, and a series of examples of the construction of such geodesics, as well as additional comments about the use of such schemes in computing general distance maps and shape-offsetting.

*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.

||Mail-stop 50A-2152, Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720 Email: ron@csr.lbl.gov FAX: (510)486-5401 Tel: (510)486-5453

**Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720 Email: sethian@math.berkeley.edu

1 Introduction

In this paper, we present a numerical technique to calculate two minimal path problems. The technique is based on the fast marching technique of Sethian [15, 16] for constructing the solution to Eikonal equations and its extension to general static Hamiltonians given by Adalsteinsson, Kimmel, Malladi and Sethian[1].

The first problem may be stated as follows. Given a cost function $F(x_1, x_2, .., x_M)$, and a starting point $A \in \mathbb{R}^M$, find the path $\gamma(t) : [0, \infty) \rightarrow \mathbb{R}^M$ from A to any point $B \in \mathbb{R}^M$ which minimizes the integral

$$\int_A^B F(\gamma(t))dt, \tag{1}$$

where t is the arclength parameterization of γ; namely $|\gamma_t| = 1$. We shall refer to this as minimal cost path. The second problem is related: Given a surface $z(x_1, x_2, .., x_M)$, find the shortest path, known as a minimal geodesic, between two points $(A, z(A))$ and $(B, z(B))$ on that surface.

In this paper, we use the fast marching technique to construct the solution to both problems. Coupling this technique to a high order ordinary differential equation solver results in a method for constructing such paths with a computational complexity of order $\mathcal{O}(N \log N)$, where N is the total number of grid points used to discretize the domain $\mathbb{R}^M$. The algorithm works in any number of space dimensions; for ease of discussion, we limit ourselves to $\mathbb{R}^2$.

2 Underlying Equations

2.1 Equation for the Minimal Cost Path

Given the cost function $F(x, y)$, and a starting point A, let $u(x, y)$ be the minimal cost required to travel from A to the point (x, y), that is

$$u(x, y) = \min_{\gamma} \int_A^{(x,y)} F(\gamma(t))dt, \tag{2}$$

with $|\gamma_t| = 1$. The level set $u = C$ is the set of all points in $\mathbb{R}^2$ that can be reached with minimal cost C, and the minimal cost paths are orthogonal to the level curves, hence we have

$$|\nabla u| = F(x, y). \tag{3}$$

This is a particular simple static Hamiltonian known as the Eikonal equation, and our goal is to first construct $u(x, y)$ in all of $\mathbb{R}^2$. Then, given a point B in $\mathbb{R}^2$, explicit construction of the shortest path comes from back propagation from B to A via the solution of the ordinary differential equation

$$\mathcal{X}_t = -\nabla u \qquad \text{given} \qquad \mathcal{X}(0) = B, \tag{4}$$

until we reach the starting point A.

2.2 Equation for Geodesics on Surfaces

2.2.1 Construction of Stationary Hamiltonian

The construction of the appropriate static Hamiltonian for geodesics on surfaces is a little more complex. A geometric perspective based on the Osher-Sethian level set formulation [13] was developed in [8]. Here we follow that discussion.

Suppose we are given a scalar function $u(x,y)$. The level sets of u are defined by $\mathcal{C}(t) = \{(x,y) : u(x,y) = t\}$. Now, let us consider the level sets $\mathcal{C}(t)$ as a family of planar curves; thus, the t level set is given as $\mathcal{C}(t)$. The planar normal for each of these curves may be written as $\vec{n} = \nabla u/|\nabla u|$. The change of u in this normal direction is $du/d\vec{n} = \langle \nabla u, \nabla u/|\nabla u| \rangle = |\nabla u|$, and by the chain rule we have

$$\mathcal{C}_t = \frac{1}{|\nabla u|}\vec{n}. \tag{5}$$

Thus, we have cast the level sets of u as a time-dependent partial differential equation describing the evolution of the level sets in time t. Summarizing, there are two ways to look at the level curves; either as static curves corresponding to different constant values t of u, or as a parameterization by time t of the evolution of an initial curve $\mathcal{C}(t = 0)$. Furthermore, we may rewrite the above expression as an Eikonal equation in the form

$$|\nabla u| = \frac{1}{\mathcal{C}_t}\vec{n}. \tag{6}$$

Recall that our goal is to construct the shortest path on a surface $z(x,y)$ from an initial surface point $(A, z(A))$ to any surface point $(B, z(B))$. Following a geometrical interpretation, imagine the propagation of the initial surface curve $\mathcal{L}(t = 0)$ along the surface, where $\mathcal{L}(t = 0)$ corresponds to the single point $(A, z(A))$. Furthermore, imagine that this curve propagates with unit speed along the surface. Then the location of the curve at time t will correspond to the set of all points of distance t from the point $(A, z(A))$, and again, the shortest path will correspond to back propagation along the surface vector field orthogonal to the surface curves $\mathcal{L}$.

These surface curves may be constructed in a different way as follows. Since the surface $z(x,y)$ is a graph, there exists some speed function $\tilde{F}$ which provides the corresponding motion of the surface curves projected onto the xy plane. That is,

$$\mathcal{C}_t = \frac{1}{\tilde{F}(\nabla z, \vec{n})}\vec{n}, \tag{7}$$

where $\vec{n}$ is the unit normal vector of the planar curve $\mathcal{C}$, and $z(x,y)$ is the given surface. This is the result of projecting the evolution of the equal geodesic distance contours from the surface to the xy coordinate plane. Our goal is to determine this speed function $\tilde{F}$.

We first note that the evolution on the surface itself before the projection is given by

$$\mathcal{L}_t = \vec{\mathcal{N}} \times \vec{\mathcal{T}}, \tag{8}$$

where, using the notation $(p,q) = \nabla z$, the surface normal is $\vec{\mathcal{N}} = (-p, -q, 1)/\sqrt{1 + p^2 + q^2}$, and $\vec{\mathcal{T}}$ is the tangent to the current equal geodesic distance contour $\mathcal{L}$. This evolution can be used to compute the geodesic distance map from a given point or a set of points on the surface $z(x, y)$. Thus

$$\frac{1}{\tilde{F}(\nabla z, \vec{n})} = \langle \mathbf{\Pi} \circ \vec{\mathcal{N}} \times \vec{\mathcal{T}}, \vec{n} \rangle,$$

where $\mathbf{\Pi} \circ (x, y, z) = (x, y)$ is the projection operation onto the xy coordinate plane.

In the next section we compute the tangent $\vec{\mathcal{T}}$ to be $\vec{\mathcal{T}} = \frac{(-u_y, u_x, qu_x - pu_y)}{\sqrt{u_x^2 + u_y^2 + (qu_x - pu_y)^2}}$. We also use the relation $\vec{n} = \nabla u / |\nabla u|$. Again, solving for $|\nabla u|$, we have the static Hamilton Jacobi equation of the surface distance map u given by

$$|\nabla u|^2 = \tilde{F}(\nabla z, \vec{n})^2.$$

The result is given by $u = 0$ at A as boundary conditions to the solution of

$$\frac{u_x^2(1 + q^2) + u_y^2(1 + p^2) - 2pqu_xu_y}{1 + p^2 + q^2} = 1. \tag{9}$$

This results in the following Hamiltonian

$$H(u_x, u_y) = (1 + q^2)u_x^2 + (1 + p^2)u_y^2 - 2pqu_xu_y - (1 + p^2 + q^2). \tag{10}$$

Again, our goal is to find the surface u that solves the static Hamilton Jacobi equation given by $H(u_x, u_y) = 0$, with the boundary conditions $u = 0$ at A. In this case $H(v, w)$ may be checked to be always convex.

2.2.2 Back Propagation to Construct Geodesic Path

Finally, we explicitly show how to compute $\vec{\mathcal{T}}$, and how to construct the geodesic by solving the ordinary differential equation

$$\mathcal{X}_t = -\mathbf{\Pi} \circ (\vec{\mathcal{N}} \times \vec{\mathcal{T}}) \qquad \text{given} \qquad \mathcal{X}(0) = B, \tag{11}$$

$\vec{\mathcal{N}}$ is given as a function of ∇z, and $\vec{\mathcal{T}}$ is obtained by back projecting the level set of u onto z and then computing the tangent. Here, $\mathcal{X}$ is the projection of the geodesic on the xy coordinate plane.

Since the level sets of u are the projections of the equal geodesic distance contours, we can back project the tangents of those contours back to the surface and obtain $\vec{\mathcal{T}}$. The tangent of the level sets of u is given by $(-u_y, u_x)$, or

$$\mathbf{\Pi} \circ \vec{\mathcal{T}} = c(-u_y, u_x).$$

Writing the tangent as $\vec{\mathcal{T}} = (T_1, T_2, T_3)$, we have $(T_1, T_2) = c(-u_y, u_x)$. We also know that $\vec{\mathcal{T}}$ is in the tangent plane, and thus orthogonal to $\vec{\mathcal{N}}$. Therefore,

$$\langle \vec{\mathcal{T}}, \vec{\mathcal{N}} \rangle = 0.$$

Using the previous expression we may write

$$T_3 = c(qu_x - pu_y),$$

and compute the constant c by normalizing $\vec{\mathcal{T}}$ to be:

$$\vec{\mathcal{T}} = \frac{(-u_y, u_x, qu_x - pu_y)}{\sqrt{u_x^2 + u_y^2 + (qu_x - pu_y)^2}}.$$

We can now compute the vector product

$$\vec{\mathcal{N}} \times \vec{\mathcal{T}} = \frac{(u_x(1+q^2) - pqu_y, u_y(1+p^2) - pqu_x, pu_x + qu_y)}{\sqrt{(1+p^2+q^2)(u_x^2 + u_y^2 + (qu_x - pu_y)^2)}},$$

and conclude with the back projection evolution on the coordinate plane:

$$\mathcal{X}_t = -\frac{(u_x(1+q^2) - pqu_y, u_y(1+p^2) - pqu_x)}{\sqrt{(1+p^2+q^2)(u_x^2 + u_y^2 + (qu_x - pu_y)^2)}}. \tag{12}$$

Thus our goal is two-fold; first to solve the static Hamiltonian for the function u given by Eqn. (10), and then to solve the ordinary differential equation given by Eqn. (12).

3 Numerical Approximation

In this section, we discuss schemes to approximate the above equations.

3.1 The Fast Marching Method

In [15, 16], the fast marching method for the Eikonal equation was introduced. This method forms the core of our algorithm, and we now briefly review it.

The simplest example involves solving the Eikonal equation

$$|\nabla u| = F(x, y),$$

in which $F(x,y) > 0$. It may equivalently be written as

$$|\nabla u|^2 = F(x,y)^2, \tag{13}$$

for which the Hamiltonian is given by

$$H(v, w) = v^2 + w^2 - F^2.$$

With the simple boundary conditions $u(x_0, y_0) = 0$, a 'smooth' surface u that satisfies $H(u_x, u_y) = 0$ is considered to be the desired result.

It was shown in [16] that this equation may be efficiently solved on a rectangular grid by guaranteeing that the update of the grid points is done in a monotonic fashion, from the boundary conditions "outwards". This way, once the grid point with the smallest u value is selected, there is no way that it might be updated again. The goal then is to build a consistent approximation to the above gradient which is upwind/monotone in this sense. We now do so.

Define the backward and forward x partial derivative approximation of $u_{i,j} = u(i\Delta x, j\Delta y)$ to be $D_{ij}^{-x} \equiv (u_{i,j} - u_{i-1,j})/\Delta x$, and $D_{ij}^{+x} \equiv (u_{i+1,j} - u_{i,j})/\Delta x$, and similarly for y. Several consistent approximations to the above gradient are possible, for example, the scheme used in the level set approximation given in [13]. Let us use the roots that result from the following consistent numerical approximation given in [14] for Equation (13), namely

$$\max(D_{ij}^{-x}, -D_{ij}^{+x}, 0)^2 + \max(D_{ij}^{-y}, -D_{ij}^{+y}, 0)^2 - F^2 = 0. \tag{14}$$

Our goal is to construct values $u_{i,j}$ on the grid which satisfy this difference approximation and initial starting value $u = 0$ at one grid point (i_0, j_0).

The fast marching method uses a thin band of points, known as *NarrowBand* points, which lies in between *Alive* points where the value of $u_{i,j}$ has been correctly computed and *FarAway* points which have not yet been considered. The essential idea is to systematically sweep through the grid in an upwind fashion, converting points from *NarrowBand* to the *Alive*, and adding new points from *FarAway* into the *NarrowBand* points. Choosing the smallest element of the *NarrowBand* set for conversion to an *Alive* point insures that the grid of points is systematically considered in an upwind fashion; use of min-heap algorithm provides an efficient way of locating this minimum. For complete details, see [1, 16].

Algorithmically, the fast marching method can be written as:

1. Initialize
   (a) (Alive Points:) Initialize *Alive* to be an empty set.
   (b) (NarrowBand Points:) Let $NarrowBand = \{(i_0, j_0)\}$ and set $u_{i_0,j_0} = 0$.
   (c) (Far Away Points:) Let *FarAway* be the set of all the rest of the grid points $\{(i,j) : (i,j) \neq (i_0, j_0)\}$, set $u_{i,j} = \infty$ for all points in *FarAway*.
2. Marching Forwards
   (a) Begin Loop: Let $(i_{\min}, j_{\min})$ be the point in *NarrowBand* with the smallest value for u.
   (b) Add the point $(i_{\min}, j_{\min})$ to *Alive*; remove it from *Front*.
   (c) Tag as neighbors any points $(i_{\min}-1, j_{\min})$, $(i_{\min}+1, j_{\min})$, $(i_{\min}, j_{\min}-1)$, $(i_{\min}, j_{\min}+1)$ that are not *Alive*; if the neighbor is in *FarAway*, remove it from that set and add it to the *NarrowBand* set.
   (d) For each neighbor (i,j) compute $u_{i,j}$ as follows:
      - Let $a = \min(u_{i-1,j}, u_{i+1,j})$, $b = \min(u_{i,j-1}, u_{i,j+1})$, $F = \sqrt{1/I_{i,j}^2 - 1}$.
      - If $F > |a-b|$ then let $u_{i,j} = (a + b + \sqrt{2F^2 - (a-b)^2})/2$.
      - else let $u_{i,j} = F + \min(a,b)$.
   (e) Return to top of Loop.

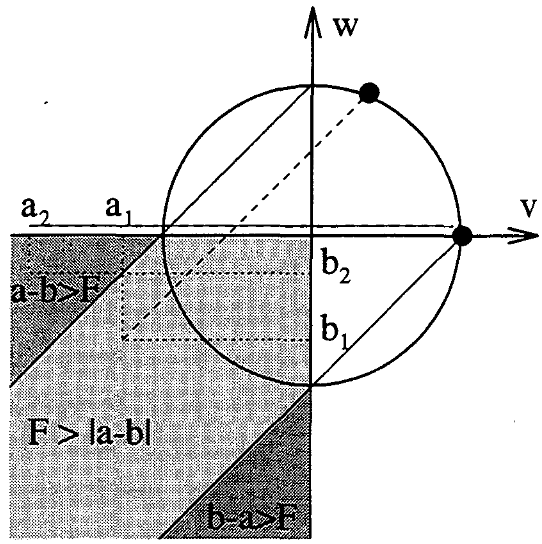


Figure 1: A geometric description of finding the solution to the numerical approximation of the Eikonal equation, see text.

The process of finding the roots for Equation (14) may be described geometrically. In Figure 1 we consider the circle to be the set of points at which $H(v,w) = 0$, in the vw plane. For every $a = \min(u_{i-1,j}, u_{i+1,j})$ and $b = \min(u_{i,j-1}, u_{i,j+1})$ we search for $u_{i,j}$ that solves Equation (14) along a line in the vw plane. If $F > |a-b|$, like $a = a_1$ and $b = b_1$ in Figure 1, then this line is defined by

$$(v(t), w(t)) = (t-a, t-b).$$

This is the diagonal dashed line starting at $(-a_1, -b_1)$. The solution corresponds to the value of t at the intersection with the circle after intersecting the v and w axis. These intersections correspond to the restriction on t to be larger than a and b.

In case $a - b > F$, like $a = a_2$, $b = b_2$, the search for the solution is the intersection of

$$(v(t), w(t)) = (t-a, 0),$$

with the circle, *i.e.* at $t = a + F$. For the last case $b - a > F$, the solution is the intersection of the vertical curve $(0, t-b)$ with the circle at $t = b + F$. By observing this geometric representation of the numerical scheme, it is easy to verify that in this simple case there always exists a solution.

An important observation is that the computational complexity of this approach is $\mathcal{O}(NlogN)$ where N is the number of grid points. It may be realized by using a min-heap data structure for the *NarrowBand* list of points (see [1]).

Note also that for the simplest selection of $F_{i,j} = 1$, u is the Euclidean distance function (possibly from a sub-grid resolution initial curve [9]) obtained in a very efficient way. The level sets of u may be obtained by applying a simple $\mathcal{O}(N)$ contour finder procedure. These are actually the offset curves of the initial contour, and the algorithm thus provides an extremely efficient way to construct shape offsets in computer-aided-design without need to consider splitting, merging and singularities in the offsets.

3.2 Extension of Fast Marching Method Solving Static Hamiltonians

In [1], the above technique was extended to a wide class of static Hamiltonians. Given the Hamiltonian for distance map on surfaces, namely

$$H(u_x, u_y) = (1 + z_y^2)u_x^2 + (1 + z_x^2)u_y^2 - 2z_x z_y u_x u_y - (1 + z_x^2 + z_y^2). \tag{15}$$

To approximate this Hamiltonian, we use the scheme proposed in [8], *i.e.* the minmod approximation for $u_x u_y$, and $\max(D_{ij}^{-x}, -D_{ij}^{+x}, 0)^2$ for u_x^2. Then, select the largest root, and verify that it is larger than those neighboring grid points that took active part in it computation. *i.e.* the u values at the neighboring points that result the selected quadratic equation are smaller than the selected root.

More precisely, we need to solve u that satisfies the following static HJ equation:

$$(1 + q^2)u_x^2 + (1 + p^2)u_y^2 - 2pq u_x u_y = 1 + p^2 + q^2,$$

or in a more compact form

$$a u_x^2 + b u_y^2 + c u_x u_y = d$$

(for $a, b, d > 0$). Applying the $\max(D^-, -D^+, 0)$ differential approximation for the u_x^2 and u_y^2, one may alternatively write

$$\begin{aligned} \max(D_{i,j}^{-x}, -D_{i,j}^{+x}, 0) &= \max(u_{i,j} - u_{i-1,j}, u_{i,j} - u_{i+1,j}, 0) \\ &= \max(u_{i,j} - \min(u_{i-1,j}, u_{i+1,j}), 0). \end{aligned}$$

Denote by $m^x \equiv \min(u_{i-1,j}, u_{i+1,j})$, and similarly for $m^y \equiv \min(u_{i,j-1}, u_{i,j+1})$. We are searching for $u_{i,j} = r$ that is the solution to a quadratic equation and larger than those points that generated it. We thus check the following 7 cases:

1. $a(r - m^x)^2 + b(r - m^y)^2 + c(r - u_{i-1,j})(r - u_{i,j-1}) = d$,
   for r that satisfies $m^x, m^y, u_{i-1,j}, u_{i,j-1} < r$.

2. $a(r - m^x)^2 + b(r - m^y)^2 + c(u_{i+1,j} - r)(u_{i,j+1} - r) = d$,
   for r that satisfies $m^x, m^y, u_{i+1,j}, u_{i,j+1} < r$.

3. $a(r - m^x)^2 + b(r - m^y)^2 + c(r - u_{i-1,j})(u_{i,j+1} - r) = d$,
   for r that satisfies $m^x, m^y, u_{i-1,j}, u_{i,j+1} < r$.

4. $a(r - m^x)^2 + b(r - m^y)^2 + c(u_{i+1,j} - r)(r - u_{i,j-1}) = d$,
   for r that satisfies $m^x, m^y, u_{i+1,j}, u_{i,j-1} < r$.

5. $a(r - m^x)^2 + b(r - m^y)^2 = d$,
   for r that satisfies $m^x, m^y < r$ and $\{u_{i-1,j}, u_{i+1,j} < r$ or $u_{i,j-1}, u_{i,j+1} < r\}$.

6. $r = m^x + \sqrt{d/a}$, where $r < m^y$.

7. $r = m^y + \sqrt{d/b}$, where $r < m^x$.

The largest r is selected as the desired solution.

Other schemes are possible; the goal is a consistent scheme that possesses the monotonic updating property that forces correctness of the algorithm results by restricting the computed value to be larger than those that generate it.

3.3 Computing the Back Trajectories

Once the solution u_{ij} is computed, our goal is now to construct the actual geodesic from this distance map. Let us write our trajectory equation (12) in the abstract form

$$\mathcal{X}_t = \mathcal{V}. \tag{16}$$

For numerical implementation we use both second order Runge-Kutta (the midpoint method) and Heun's method to integrate this ordinary differential equation. Bilinear interpolation is used to compute $\mathcal{V}$ at the current cell defined by the four closest grid points. This way ∇u is computed at the exact position form its central derivatives approximations at the grid points. Heun's method seems to yield better results near sharp corners.

4 Results

4.1 Minimal Geodesics

We present two examples of using the above back propagation technique to track minimal geodesic on two surfaces: In Figure 2, the surface is given by $z(x,y) = 0.4(1 - e^{-2(x^2+y^2)} - 0.75e^{-25((x-0.11)^2+(y-0.11)^2)})$, and in Figure 3, $z(x,y) = 0.25sin(2\pi 1.5x)sin(2\pi 1.5y)$ both are defined on the unit box $[-0.5,0.5] \times [-0.5,0.5]$. Here, we used the second order Runge-Kutta algorithm for the integration.

4.2 The Eikonal Case

Here we apply the fast marching scheme to problems involving minimal cost functions and paths of minimal cost in $\mathbb{R}^2$. As an example of our technique applied to the problem of finding minimal cost paths with sharp changes of the cost function, Figure 4 shows minimal paths in a two-valued domain. A cost faction is defined over the domain with a value of 1 for $i < 60$ and 0.2 for $i > 60$. By solving the Eikonal equation with a source point located at $(i,j) = (30,10)$, we first compute $u_{i,j}$. Then using Heun's method we back track the minimal paths from 28 different destination locations. One may observe the way these paths obey the Snell Law in optics as the 'light rays' change their course as they cross the interface line, following Fermat's Principle for light rays in an isotropic medium.

A smoother cost function is used in Figure 5. Here, $F(x,y) = 0.4(1 - e^{-2(x^2+y^2)} - 0.75e^{-25((x-0.11)^2+(y-0.11)^2)})$ on the unit box $[-0.5,0.5] \times [-0.5,0.5]$. Again, we can see the way the optimal paths are attracted by low cost area, and how they prefer to avoid the high cost area.

A possible application for medical image analysis taken from [4] is shown in Figure 6. It is shown how to track vanes in part of an intensity image $I : \mathbb{R}^2 \rightarrow [0,1]$ of the retina. The cost function in this case is $F(x,y) = 1 - I(x,y)$. The back tracking technique allows us to click at any destination point (in this case 5 points) and to isolate a branch connecting to the source point. Another medical application, based on the fast marching method to solve the Eikonal equation in $3D$ leading to shape modeling, was recently introduced in [12].

4.3 Timings

Based on the timing tests performed in [12], the time it takes to compute the minimal cost function on a $3D$ data structure of size $256 \times 256 \times 124$ is 76 seconds, running on SUN SPARC 1000. We note that just loading that data to the memory takes about 10 seconds. In our $2D$ examples, the execution time is less than a second. The back tracking procedure is even less time consuming, and took less than a second for the vanes image (including the display time on a SUN SPARC 10).

5 Additional Comments

Finally, we would like to comment on a nice interpretation that connects several recent results in the field of shape modeling, and an alternative method for computing minimal geodesics.

It was shown [8] that the minimal geodesic connecting two points is the minimal level set of the sum of their two corresponding distance maps. This is an alternative 'global' approach for finding minimal geodesics. However, since the sum of the distance maps is given as its samples on the grid, extracting its minimal set while keeping the connectivity between the two points is a non trivial operation. It actually requires thresholding the function with a value higher than the theoretical infimum. This ϵ-threshold yields a "fat" or "thick" set that needs to be refined into a thin contour. The fat set is given by $\{(x,y) : u_A(x,y) + u_B(x,y) < \inf_{\mathbb{R}^2}(u_A + u_B) + \epsilon\}$. Operating only on that ϵ-threshold set, it is possible to refine the result into an accurate curve that is the result of applying a contour finder. At this point, it is important to note that the intersection of two functions leads to accurate results when searching for contours on a grid. While the set $u(x,y) <$Threshold, needs to be refined.

The natural refinement procedures are based on the direct relation between Euclidean distance maps and curvature flow [5, 6], the Eikonal equation and geodesic active contours [2, 4, 7, 17], and between geodesic (surface) distance maps [8] and geodesic curvature flows [3, 10, 11]. The same relation may be used with other metrics by finding the relevant flow that minimizes the arclength defined by the given metric. The refinement is the result of evolving a curve according to its first variation induced by the specific metric, *i.e.* following the Euler Lagrange equations for minimizing the given metric arclength.

A natural numerical implementation for these refinement flows is obviously the level set approach [13]. In this case we need to perform calculations only within the 'ellipse' defined by the infimum+ϵ set. Thereby, we can efficiently refine the minimal geodesic connecting the two focal points, as a level set of a function, and achieve accurate results.

Although the above relations present a theoretical overview connecting some recent results, we recommend the back tracking technique which is an accurate and efficient operation.

6 Acknowledgment

We thank Ravi Malladi and David Adalsteinsson for many discussions on implementation considerations and timing tests.

References

[1] D Adalsteinsson, R Kimmel, R Malladi, and J A Sethian. Fatst marching method for computing solutions to static Hamilton-Jacobi equations. LBL report, LBNL UC Berkeley, CA 94720, Submitted 1996.

[2] V Caselles, R Kimmel, and G Sapiro. Geodesic active contours. In *Proceedings ICCV'95*, pages 694–699, Boston, Massachusetts, June 1995.

[3] D L Chopp and J A Sethian. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. *Jour. Exper. Math.*, 2(4):235–255, 1993.

[4] L D Cohen and R Kimmel. Edge integration using minimal geodesics. EE PUB No. 952, Technion–Israel Institute of Technology, Israel, January 1995. submitted.

[5] M Gage and R S Hamilton. The heat equation shrinking convex plane curves. *J. Diff. Geom.*, 23, 1986.

[6] M Grayson. The heat equation shrinks embedded plane curves to round points. *J. Diff. Geom.*, 26, 1987.

[7] S Kichenassamy, A Kumar, P Olver, A Tannenbaum, and A Yezzi. Gradient flows and geometric active contour models. In *Proceedings ICCV'95*, Boston, Massachusetts, June 1995.

[8] R Kimmel, A Amir, and A M Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. on PAMI*, 17(6):635–640, June 1995.

[9] R Kimmel and A M Bruckstein. Shape offsets via level sets. *CAD*, 25(5):154–162, March 1993.

[10] R Kimmel and N Kiryati. Finding shortest paths on surfaces by fast global approximation and precise local refinement. In *Proceedings of SPIE Vision Geometry III*, volume 2356, pages 198–209, Boston, Massachusetts, November 1994.

[11] R Kimmel and G Sapiro. Shortening three dimensional curves via two dimensional flows. *International Journal: Computers & Mathematics with Applications*, 29(3):49–62, March 1995.

[12] R Malladi and J A Sethian. An O(N log N) algorithm for shape modeling. *Proc. Natl. Acad. of Sci. USA*, submitted, Feb. 1996.

[13] S J Osher and J A Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. of Comp. Phys.*, 79:12–49, 1988.

[14] E Rouy and A Tourin. A viscosity solutions approach to shape-from-shading. *SIAM. J. Numer. Analy.*, 29(3):867–884, June 1992.

[15] J A Sethian. A review of the theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica*, in press, 1995.

[16] J A Sethian. A marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4), 1996.

[17] J Shah. Recovery of shapes by evolution of zero-crossings. Department of math., Northeastern Univ. Boston MA, MA, June 1993.

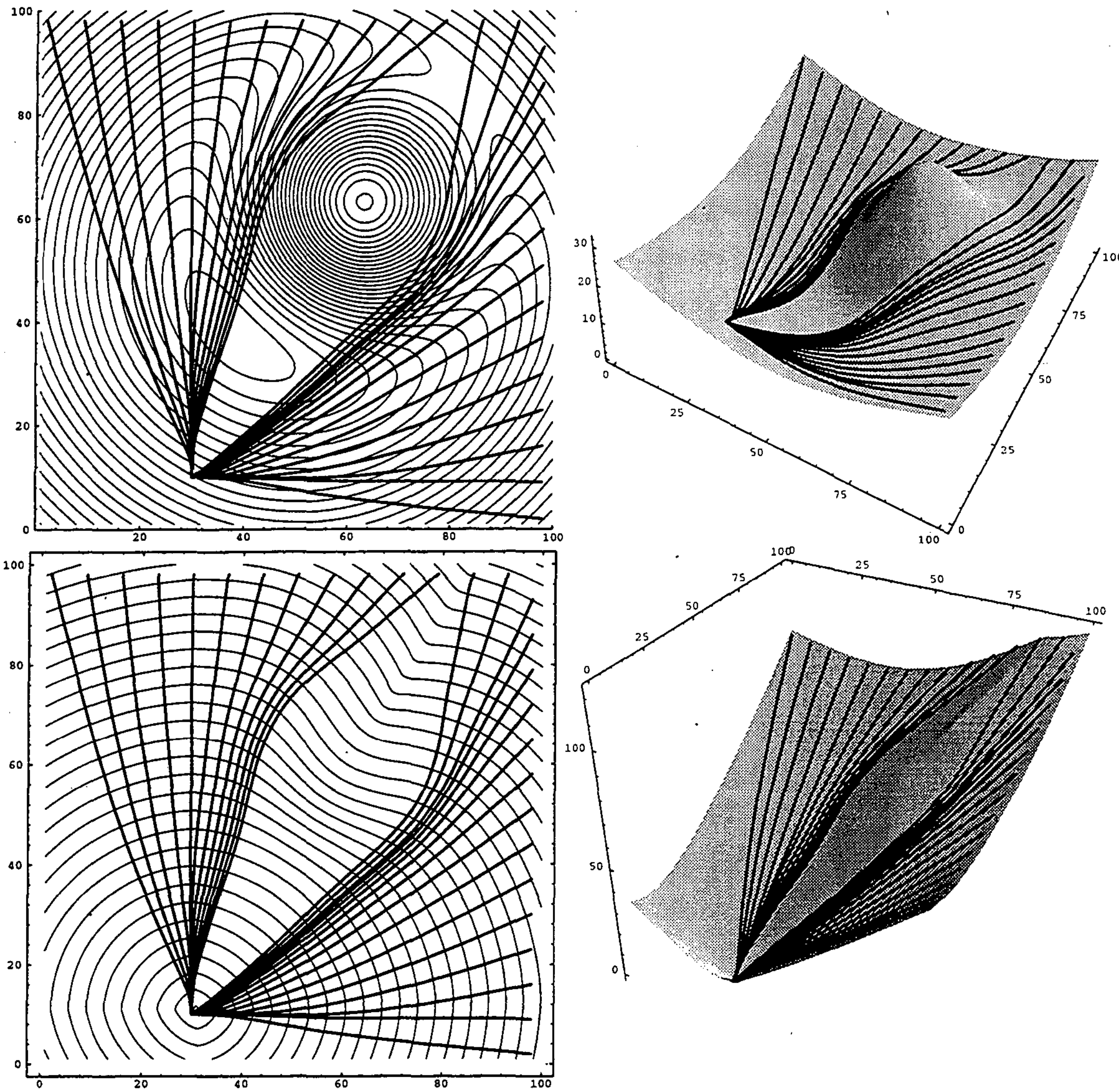


Figure 2: Finding the shortest paths on a surface: Upper left: The 28 geodesics connecting to the source point (30, 10) are the thick curves painted on the thin contours which are level sets of the surface z. Upper right: A perspective view over the surface z and the geodesics. Lower left: The geodesics are the thick curves on the level sets of the u geodesic distance map. Lower right: A perspective view over the geodesic distance map u and the minimal geodesics.

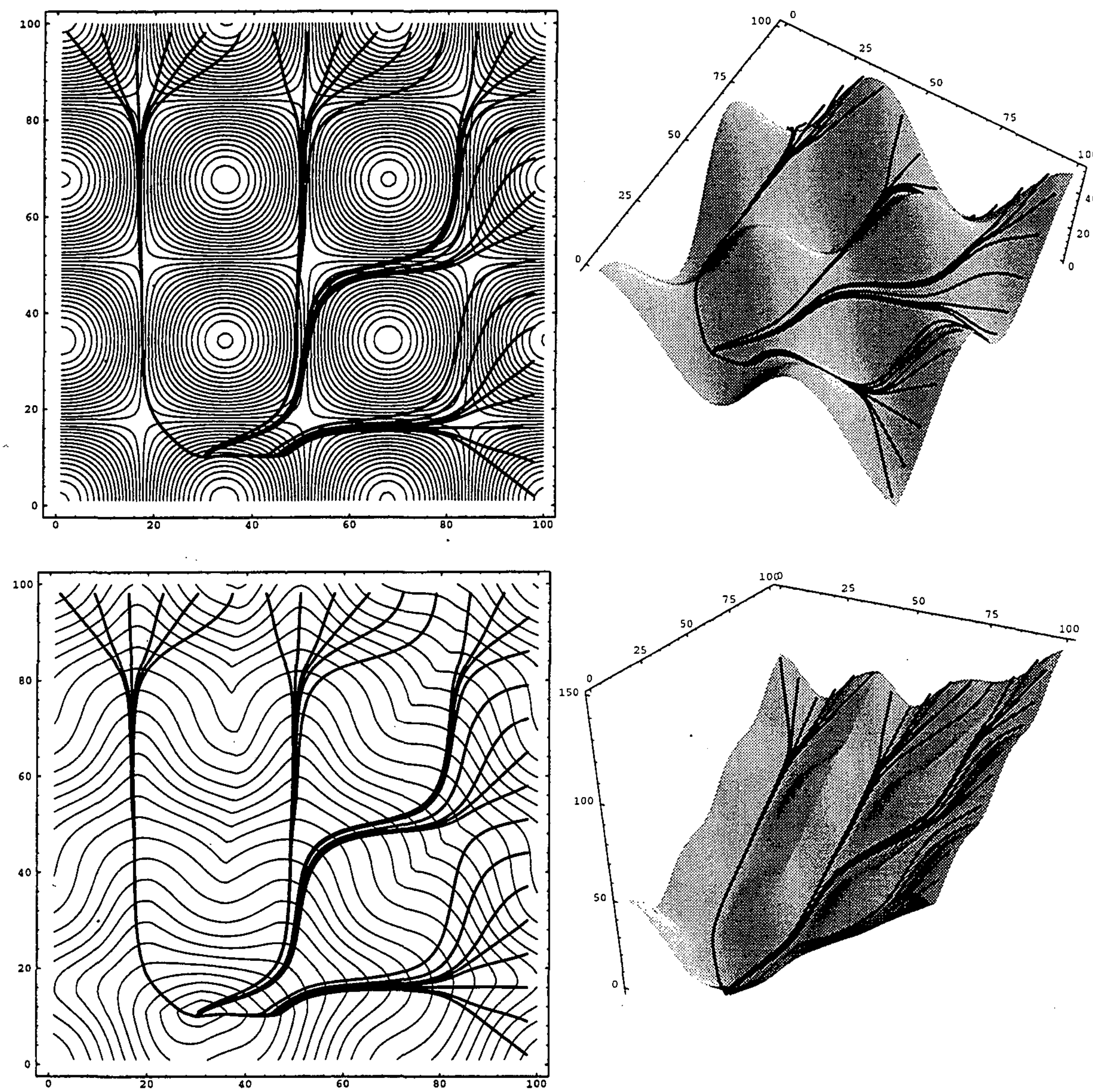


Figure 3: Finding the shortest paths on a surface: Upper left: The 28 geodesics connecting to the source point $(30, 10)$ are the thick curves painted on the thin contours which are level sets of the surface z. Upper right: A perspective view over the surface z and the geodesics. Lower left: The geodesics are the thick curves on the level sets of the u geodesic distance map. Lower right: A perspective view over the geodesic distance map u and the minimal geodesics.

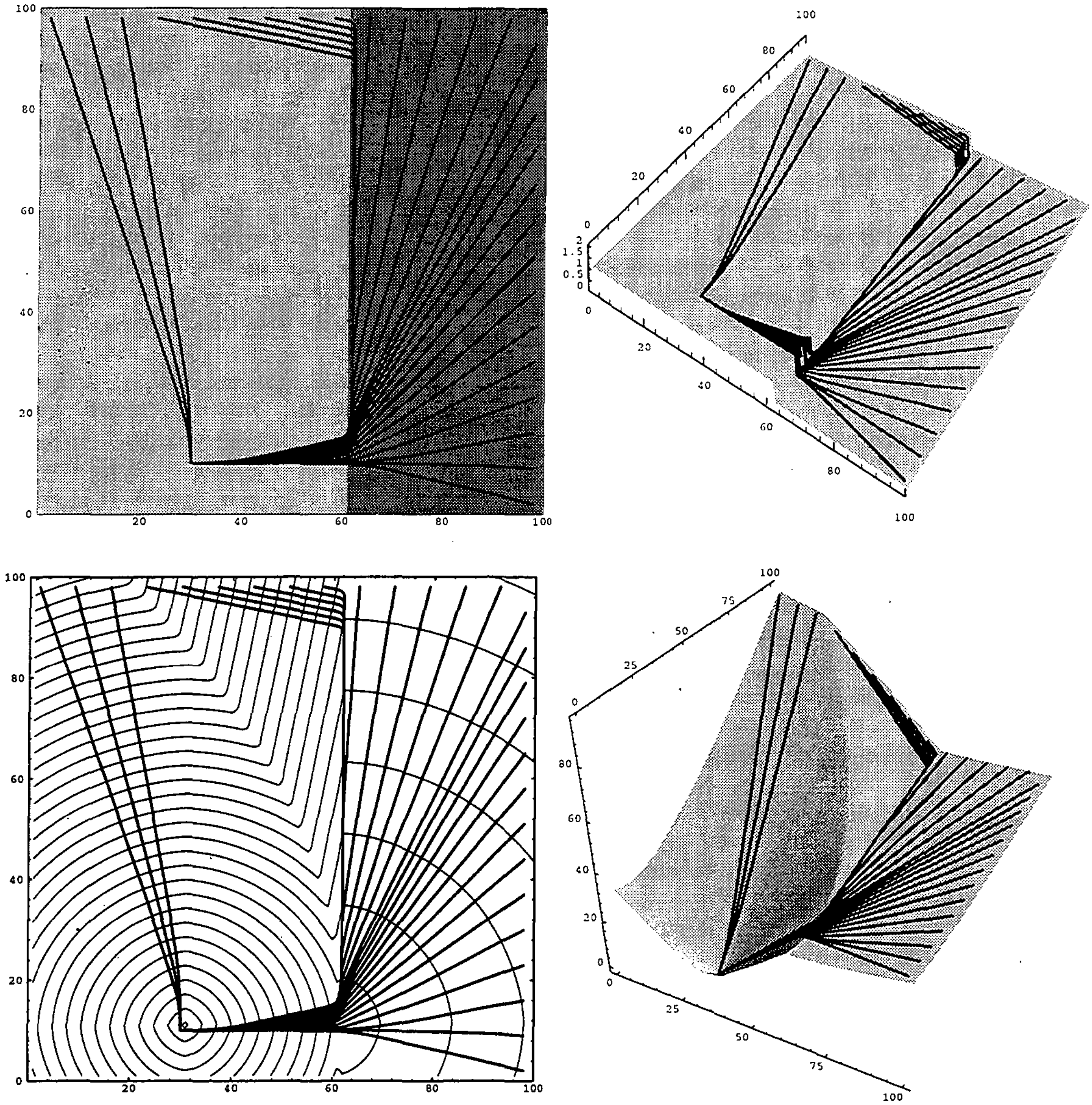


Figure 4: Finding minimal paths in a two valued domain by solving the Eikonal equation: Upper left: A 2D gray level image of the weighted domain with 28 minimal cost paths connecting to the source point at $(i, j) = (30, 10)$. Upper right: A perspective view over the cost function F and the minimal cost paths. Lower left: The minimal cost paths are the thick curves on the level sets of the u function. Lower right: A perspective view over the u function and the minimal cost paths.

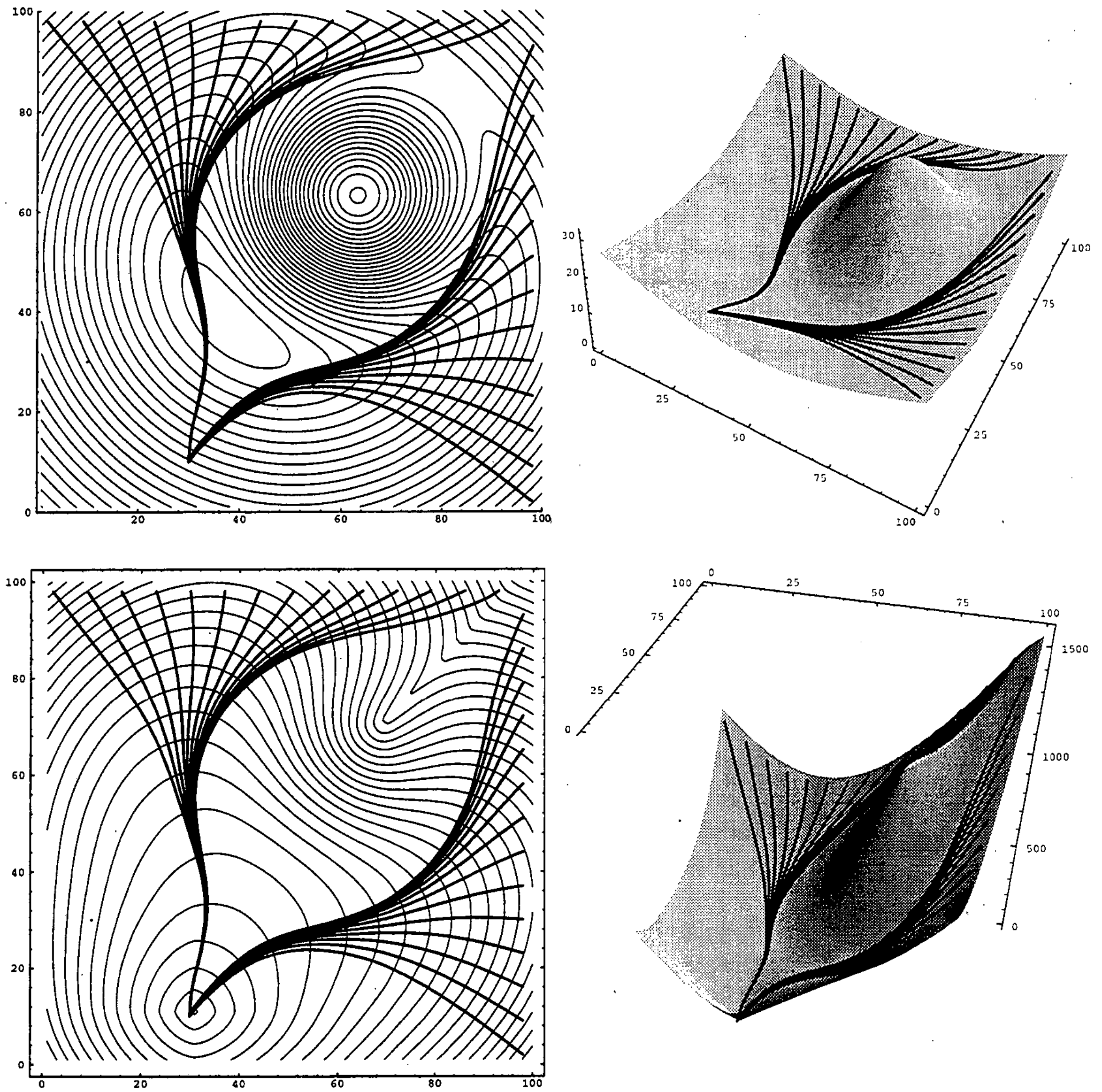


Figure 5: Finding minimal paths with a smooth cost function by solving the Eikonal equation: Upper left: The 28 minimal cost paths connecting to the source point $(30, 10)$ are the thick curves painted on the thin contours which are the level sets of the cost function. Upper right: A perspective view over the cost function F and the minimal cost paths. Lower left: The minimal cost paths are the thick curves on the level sets of the u function. Lower right: A perspective view over the u function and the minimal cost paths.

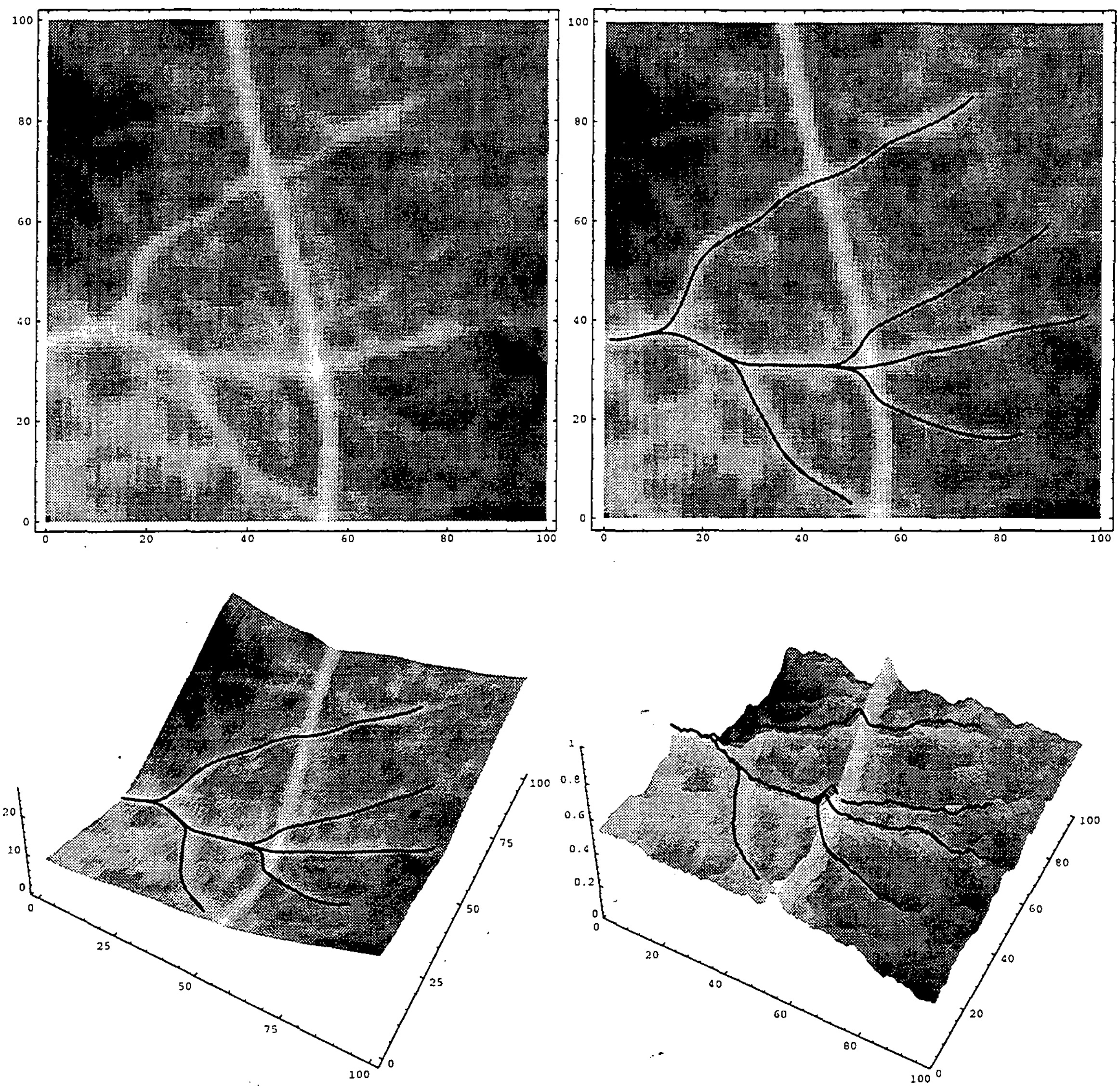


Figure 6: An example of medical image analysis based on solving the Eikonal equation: Upper left: The original vanes image $I_{i,j}$ (part of a retina image). Upper right: Isolating the 5 branches of the vane, painted on the original image. Lower left: A perspective view over the u function and the minimal cost paths. The u function is computed by considering $F_{i,j} = 1 - I_{i,j}$. Lower right: A perspective view over the image I as a function of the gray levels and the detected vane hovering above.

Ernest Orlando Lawrence Berkeley National Laboratory
Technical and Electronic Information Department
University of California | Berkeley, California 94720