Finding Shortest Paths on Surfaces by Fast Global Approximation and Precise Local Refinement

Ron Kimmel* **Nahum Kiryati****

Abstract

Finding the shortest path between points on a surface is a challenging global optimization problem. It is difficult to devise an algorithm that is computationally efficient, locally accurate and guarantees to converge to the globally shortest path. In this paper a two stage coarse to fine approach for finding shortest paths is suggested. In the first stage the algorithm of [10] that combines a 3-D length estimator with graph search is used to rapidly obtain an approximation to the globally shortest path. In the second stage the approximation is refined to become a shorter geodesic curve, i.e. a locally optimal path. This is achieved by using an algorithm that deforms an arbitrary initial curve ending at two given surface points via geodesic curvature shortening flow. The $3D$ curve shortening flow is transformed into an equivalent $2D$ one that is implemented using an efficient numerical algorithm for curve evolution with fixed end points, introduced in [9].

1 Introduction

Searching for the shortest path, also known as the *minimal geodesic*, between two points on a three dimensional surface is very important in robotic navigation and in three dimensional shape analysis. For example, it is the key to computational surface flattening [16], a potentially useful technique for brain research.

Standard computational procedures to obtain shortest paths on continuous surfaces involve numerical solutions of differential equations by numerical integration [2], and are computationally intensive. These methods are accurate but yield paths that are only locally optimal. When handling complex surfaces a good initial guess is needed to increase the likelihood of convergence to the globally optimal path.

In the *discrete geodesic problem*, the shortest paths between points on a *polyhedral* surface are to be determined. This is of importance in certain CAD-oriented and land-surveying applications where a polyhedral representation of solids and surfaces is natural. An algorithm to solve this problem was described in [13], in which the shortest path from a source point to a destination point is found in $O(n^2 \log n)$ time, where n denotes the number of edges in

*Lawrence Berkeley Laboratory, Mailstop 50A-2129, University of California, Berkeley, CA 94720, USA.
**Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel

the polyhedral surface. The length of the shortest path to any other destination can then be determined in $O(\log n)$ time, and the shortest path can be listed in $O(k)$ time, where k is the number of edges crossed by the path. Wolfson and Schwartz [20] argued that the algorithm of [13] is difficult to implement, and suggested an algorithm that is simpler to implement, but is of exponential computational complexity.

Kiryati and Székely [10] considered the problem of finding the shortest path between points on a continuous three dimensional surface that is given in a digitized form. Its basis is the observation that actual interpolation of the digitized surface may not be needed in order to estimate properties of the underlying continuous structure to fairly high accuracy. Their approach combined recently developed length estimators for digitized three dimensional curves with well known algorithms for computing shortest paths in sparse graphs, to estimate the minimal distances and the corresponding shortest paths in a systematic, computationally efficient way. Note that shortest path algorithms that are based on surface discretization and graph search are fast, but are of inherently limited accuracy [14].

A framework that combines the advantages of two different approaches in order to obtain the minimal geodesic on an essentially continuous $3D$ surface quickly and accurately, is introduced in this paper. A two stage coarse–to–fine approach is taken. The fast graph–search based algorithm [10] that operates on a voxel representation of the surface is used as a first stage to obtain a good initial approximation efficiently. This initial approximation is used as the input to the second stage, a shortening flow differential geometry technique [9]. The curve is evolved by a geodesic curvature flow to the geodesic curve closest to the initial approximation. In comparison to pure differential techniques, the speed of convergence is greatly improved and the risk of convergence to an insignificant local minimum is alleviated.

Curve shortening flow has attracted great interest in recent years in the field of Differential Geometry. In [7], Grayson extended the theory of planar curve evolution and noted several global properties of curvature evolution of smooth curves immersed in a Riemannian surface. He proved that the curve remains smooth, and it either converges to a point, or becomes a geodesic curve when its geodesic curvature converges to zero in the C^{∞} norm.

One of the most important properties of the flow–by–curvature curve evolution process is that it shrinks the curve as fast as possible, in the sense that flow lines in the space of closed curves are tangent to the gradient of the length functional [7]. For this reason, the flow is also called *curve shortening flow*. Therefore, deforming a curve by its geodesic curvature vector is a very efficient way of finding geodesic curves.

In the suggested combined approach, the curve shortening flow operates on a $3D$ surface given as an elevation array. Given the surface and two points on it, the curve shortening flow algorithm evolves an arbitrary initial embedded curve between these points. Grayson's results show that the curve remains smooth and embedded, and if the end points are fixed it converges to a geodesic curve on the surface between the given points. See [4, 5] for a similar approach.

For computerized implementation of the second (shortening flow) stage, the three dimensional curve flow is first transformed into an equivalent two dimensional curve flow. It is then implemented based on a numerical algorithm derived from [15], together with a simple algorithm, motivated by [3], for keeping the end points fixed.

2 Rapid Approximation of the Shortest Path

The first stage of the suggested approach is to rapidly obtain an approximation to the shortest path between points on a 3D surface. It is based on the algorithm described in [10].

Suppose that the surface of an object is given in a digitized form as a set of voxels in a three dimensional array. Assume that a voxel belongs to the digitized surface if it is traversed by the underlying continuous surface and if at least one of its direct (*i.e.*, 6-directional) neighbors is a "background" voxel. A path on the surface that connects two surface points can be represented in digital form by the set of surface voxels that the path traverses. This "digital path" can be represented by a 26-directional chain code.

The $3D$ length estimation problem is to estimate the length of an underlying continuous $3D$ curve given its chain code. The estimator recently developed in [11] is based on link classification in the 26-directional chain code representation of the curve. It has the general form:

$$\hat{L} = \Psi_1 N_1 + \Psi_2 N_2 + \Psi_3 N_3, \tag{1}$$

where N_1 is the number of direct links in the chain code, *i.e.*, links that are parallel to one of the three main axes, and N_2 and N_3 are respectively the number of minor and major diagonal links. Ψ_1, Ψ_2 and Ψ_3 are weights, and $\hat{L}$ is the estimated length.

Note that other forms of length estimators can be suggested. In particular, one might use neighborhoods larger than $3 \times 3 \times 3$ and represent digital paths using generalized chain codes that link voxels farther apart. This would lead to finer link classification and to potentially higher length estimation accuracy on planar surfaces. However, if the surface would not be sufficiently flat within the larger neighborhood, some of the longer links might correspond to hops rather than to feasible paths between points on the surface, resulting in degraded length estimation accuracy.

Assuming that the digitization is sufficiently fine so that the curve is reasonably straight within one or two voxels, the naive selection of weights $\Psi_1 = 1$, $\Psi_2 = \sqrt{2}$ and $\Psi_3 = \sqrt{3}$ would lead to consistent overestimation of the length. Assuming a uniform distribution of orientations, an *unbiased* estimator that achieves the least RMS error possible for unbiased estimators, was found [11] to be:

$$\hat{L} = 0.9016 N_1 + 1.289 N_2 + 1.615 N_3. \tag{2}$$

The fact that the estimator is unbiased implies that if the direction of the tangent varies along the curve, local estimation errors cancel out and lower total estimation errors are obtained.

Our approximation to the shortest path between two points is based on the digital path whose length estimate is the shortest. To efficiently find it, view the digital surface as a three dimensional graph, in which each vertex corresponds to a surface voxel. Given any specific definition of surface connectivity, pairs of vertices that correspond to pairs of neighboring surface voxels are connected by arcs in the graph. Here, every surface voxel is connected by an arc to every surface voxel in its 26-neighborhood. Since the number of arcs emanating from any vertex is upper bounded (by 26), the graph is sparse. (In practice, the number of arcs emanating from most vertices is about 8.) If each arc is assigned a cost according to the weight of its link type (direct, minor diagonal or major diagonal) in the $3D$ length estimator, then estimating minimal distances and shortest paths on a continuous surface given in digitized form, reduces to finding the shortest path in a (sparse) graph.

Algorithms for finding minimal distances and shortest paths in graphs are well known; see reference [6] for an overview. Here all arcs have positive weights, hence, Moore and Dijkstras' algorithm can be applied. Let N denote the number of vertices in the graph, *i.e.*, the number of surface voxels. The minimal distance from a vertex to all others in a sparse graph can be estimated in $O(N \log M)$ time, where M is the total number of arcs in the graph, and is proportional to N. Given the source voxel, the minimal distances to all other surface voxels are thus estimated in $O(N \log N)$ time. When a single destination is specified in advance, actual computing time can be significantly reduced by simultaneous propagation from the source and destination voxels until the first meeting of the distance wavefronts.

3 Curve Shortening Path Refinement

Using the results of the first stage as an initial approximation, the geodesic curvature shortening flow algorithm [9] will shorten the curve to a geodesic. In most practical situations, this geodesic will be a globally minimal path, i.e. the *minimal geodesic*. If there are several significantly different possible paths of nearly the same length as the shortest one, the approximation obtained in the first stage may not correspond to the absolutely shortest path. In that case, the geodesic obtained in the second stage might be just slightly longer than the minimal geodesic. Let us summarize the results of [9] as applied to our problem.

Given a regular surface $\mathbf{S}$ and two points $\mathbf{a}$ and $\mathbf{b}$ on it, we want to compute a geodesic curve ending at these points. Let $\mathcal{C}_0 : [p_1, p_2] \to \mathbf{S}$, be a given initial embedded smooth curve such that $\mathcal{C}_0(p_1) = \mathbf{a}$ and $\mathcal{C}_0(p_2) = \mathbf{b}$. Based on Grayson's results [7], if $\mathcal{C}_0$ is deformed via the curve shortening flow on $\mathbf{S}$, and if the two end points are kept fixed, the curve converges to a geodesic curve as fast as possible. The geodesic curvature flow is given by

$$\frac{\partial \mathcal{C}(p,t)}{\partial t} = \kappa_g \vec{\mathcal{N}}, \tag{3}$$

where p parameterizes the curve, t stands for time, and $\kappa_g \vec{\mathcal{N}}$ is the geodesic curvature vector of the curve $\mathcal{C}$. The geodesic curvature vector may also be written as:

$$\kappa_g \vec{\mathcal{N}} = \mathcal{C}_{ss} - \langle \mathcal{C}_{ss}, \vec{N} \rangle \vec{N},$$

where $\mathcal{C}_{ss}$ (the curvature vector) is the second derivative of the curve according to s, its arc-length parameterization, and $\vec{N}$ is the normal to the surface. From (3) and the results in [7] it follows immediately that the only possible stable curves are geodesic curves. Hence, applying (3) to the initial curve $\mathcal{C}_0$ will give us the required geodesic curve.

In order to use an efficient numerical algorithm, the $3D$ flow (3) is first transformed into a $2D$ one. Let the surface $\mathbf{S}$ be given by a function $z(x, y)$. Define the curve $\hat{\mathcal{C}} := [x(p), y(p)] = \pi \circ [x(p), y(p), z(p)]$ as the projection of $\mathcal{C}(p, t)$ on the (x, y)–plane. While $\mathcal{C}$ deforms according to (3) on the surface, $\hat{\mathcal{C}}$ evolves according to the following $2D$ curve flow

$$\frac{\partial \hat{\mathcal{C}}}{\partial t} = \langle \pi \circ \kappa_g \vec{\mathcal{N}}, \hat{n} \rangle \hat{n},$$

where $\hat{n}$ is the unit normal vector of the planar curve. Rewriting this evolution as a function of $\hat{\mathcal{C}}$, the planar curve, and the surface's first derivatives, we obtain

$$\frac{\partial \hat{\mathcal{C}}}{\partial t} = \frac{1}{1+\langle \nabla z, \hat{n}\rangle^2} \left(\hat{\kappa} + (z_{xx} n_2^2 - 2 z_{xy} n_1 n_2 + z_{yy} n_1^2) \frac{\langle \nabla z, \hat{n}\rangle}{1+|\nabla z|^2} \right) \hat{n} \ , \tag{4}$$

where $\hat{n} = [n_1, n_2]$ and $\hat{\kappa}$ are the unit normal and the curvature of the planar curve, respectively. Solving (4) is equivalent to solving (3).

4 Implementation

For implementing (4) while keeping the end points fixed, the numerical algorithm due to Osher and Sethian [15, 17, 18] is used.

Osher and Sethian proposed to observe an implicit representation of the evolving of the planar curve, in which the propagating curve is embedded as a level set in a higher dimensional function, thereby achieving better stability and accuracy of the numerical implementation. Let $\hat{\mathcal{C}}(t)$ be the zero level set of a smooth and Lipschitz continuous function $\Phi : \mathbb{R}^2 \times [0, \tau) \to \mathbb{R}$. Assume that Φ is negative in the interior and positive in the exterior of the zero level set. Then the evolution equation of Φ, such that the evolving curve $\hat{\mathcal{C}}(t)$ is given by the evolving zero level set of $\Phi(t)$, *i.e.*, $\hat{\mathcal{C}}(t) \equiv \Phi^{-1}(0)$, is given by[1]

$$\Phi_t = \tag{5}$$

$$\frac{\Phi_x^2 \Phi_{yy} - 2\Phi_x \Phi_y \Phi_{xy} + \Phi_y^2 \Phi_{xx} + (z_x \Phi_x + z_y \Phi_y)(z_{xx} \Phi_y^2 - 2\Phi_x \Phi_y z_{xy} + z_{yy} \Phi_x^2)\frac{1}{1+z_x^2+z_y^2}}{\Phi_x^2(1+z_y^2) + \Phi_y^2(1+z_x^2) - 2 z_x z_y \Phi_x \Phi_y} .$$

The curve $\hat{\mathcal{C}}$ that evolves according to (4) is obtained as the zero level set of the function Φ that evolves according to (5).

The discretization of the evolution equations is performed on a fixed *rectangular grid*. The spatial derivatives are implemented using central approximations, and the time derivative by a forward approximation.

The algorithm as described by Osher and Sethian works on closed curves, or curve flows with boundary conditions. In our case, we have to keep the two end points fixed. This is done by adding a step to the algorithm, which alters the Φ function after each iteration. This part of the algorithm was motivated by Chopp's work on minimal surface computation with fixed boundary conditions [3]. This change is made in order to ensure that the two end points will remain in their position, as two fixed points in the zero level set of the evolving function.

A stopping condition can be obtained by defining a threshold value on the geodesic curvature, in which case the zero set should be tracked and the geodesic curvature along it should be computed every iteration. An alternative stopping condition relates to small changes in the implicit representation of the evolving curve: $\int |\Phi(n\Delta t) - \Phi((n-1)\Delta t)| <$Threshold.

The implementation of the geodesic curvature shortening flow stage can be summarized as follows:

[1] See [9] for detailed derivation of expressions (4) and (5).

1. Initialize Φ_0, using $\pi \circ \mathcal{C}_0$. In our implementation we used a truncated distance map: Distance in a neighborhood of the zero set, and constant values elsewhere. The evolving Φ was re-initialized to become a distance map near the zero set every few iterations, simulating the narrow band algorithm introduced in [3], and developed for efficient implementations in [1].

2. Evolve Φ according to Equation (5) that corresponds to Equation (4), by using central difference approximation for the spatial derivatives and a forward difference approximation for the time derivative:

$$
\begin{aligned}
\Phi^n_{i,j} &\equiv \Phi(i\Delta x, j\Delta y, n\Delta t) \\
\Phi_t &\approx (\Phi^{n+1}_{i,j} - \Phi^n_{i,j})/\Delta t \\
\Phi_x &\approx (\Phi^n_{i+1,j} - \Phi^n_{i-1,j})/(2\Delta x) \\
\Phi_{xx} &\approx (\Phi^n_{i+1,j} - 2\Phi^n_{i,j} + \Phi^n_{i-1,j})/(\Delta x)^2 \\
\Phi_{xy} &\approx (\Phi^n_{i+1,j+1} + \Phi^n_{i-1,j-1} - \Phi^n_{i-1,j+1} - \Phi^n_{i+1,j-1})/(2\Delta x)^2
\end{aligned}
$$

3. Adapt Φ to keep the end points fixed. In order to ensure that the two end points will remain in their position along the evolution, the propagating Φ function, which is an implicit representation of the propagating planar curve, is changed every iteration such that the weighted average in the neighborhood of the end points is equal to zero. In our examples we have actually added two lines connecting the curve's end points to the boundary of the domain, and the values along these connection lines were kept fixed during the iterations. We have chosen to fix the values of the Φ function along the two lines connecting the fixed points to the boundary of the domain, i.e. the Φ values along those lines remain unchanged along the evolution. This way, the curve divides the domain into two parts, and the Φ function is defined to be positive on one part and negative on the other.

   Keeping the connection lines fixed is equivalent to introducing new initial conditions at each iteration. By considering every time step as a new problem that is defined by the same evolution rule, yet with new initial conditions, the re-initialization is a legitimate step that will not change the convergence property of the algorithm.

   For closed curves evolving according to geodesic curvature flow, the embedding is preserved and there is no need to control the propagation of Φ. In other cases, as in this case of fixing the end points, there is a danger that certain level sets might be attracted to local geodesics other than the minimal geodesic and converge to some equilibrium that is far from the desired result. In order to alleviate this problem it is necessary to supervise the behavior of the level sets to ensure that the zero set evolution is the dominant one, while the rest of the level sets are only swept by its influence. This means that the Φ surface should serve as the implicit representation of just the zero set, and should be restricted to being influenced only by the zero set and not by other level sets that might be attracted to unwanted local geodesics.

   Several numerical methods were developed to achieve this goal, e.g., the narrow band introduced in [3, 1], re-initialization of the Φ function every iteration [19], and the expansion of the velocity along the zero set to the whole image domain [12]. We have chosen to 'correct' the function every iteration so that locally $\Phi(t)$ is a distance map of the zero set in a narrow band of 6 pixel width (see also [1]). Outside this band, the function gets its maximum/minimum values. This step guarantees that the evolution of the zero level set is the correct one, and prevents other level sets from diverging and tearing the implicit evolution apart.

4. If the stopping condition, $\int |\Phi(n\Delta t) - \Phi((n-1)\Delta t)| <$Threshold, is not satisfied, go to 2.

5. Find the zero level set $\hat{\mathcal{C}}$. The final Φ function is an implicit representation of $\hat{\mathcal{C}}$. The zero level set $\hat{\mathcal{C}}$ is computed by any interpolation method of the Φ function between the grid points. The accuracy of the final result is however bounded by the numerical scheme. In our case, using central approximation for the spatial partial derivatives, leads to a bound $\mathcal{O}(\Delta x^2)$ on the accuracy of the results. Having $\hat{\mathcal{C}}$ (the projection of $\mathcal{C}$ on the coordinate plane) and $z(x, y)$, the final result $\mathcal{C}$ is obtained as backprojection from the (x, y) coordinate plane to the surface: $\mathcal{C} \equiv z(\hat{\mathcal{C}})$.

6. Stop.

5 Examples and Discussion

The suggested two stage algorithm was applied to finding shortest paths between given points on several test surfaces. The test surfaces were contained in a $64 \times 64 \times 64$ voxel array for the first stage. A $64 \times 64 \times \mathbb{R}$ elevation array representation was used in the second stage.

In the first example the shortest path between two points on a tilted plane is computed. This simple example is interesting since it is known[10] to demonstrate worst-case performance of the first stage of the algorithm. Indeed, the first graph search stage produced the broken white line in Fig. 1. The second shortening flow stage corrected the path to the black straight line. In all the examples, the gray level of the surface corresponds to elevation, the white path is the approximation obtained by the first stage, and the black path is the result of the second refining stage.

Fig. 2 demonstrates the operation of the algorithm on an "egg–box" type of surface. The length of the shortening paths along the iterations of the second stage is shown in Fig. 3. The determination of shortest path between two points on a terrain-like surface is shown in Fig. 4. Observe how the initial approximation produced by the first stage is refined to the geodesic in the second stage.

This paper combines two approaches to locate the minimal geodesic between two points on surfaces in an efficient way. The first graph–search based stage produces an initial approximation of the location of the minimal geodesic. This approximation is the initial condition for the second shortening flow based stage that refines the curve by shortening it to a geodesic.

A clear advantage of the first stage is its theoretical as well as practical computational efficiency. Another important feature is the direct operation on voxel data, without need for prior interpolation or polygonal approximation of the surfaces. Its main drawback is that

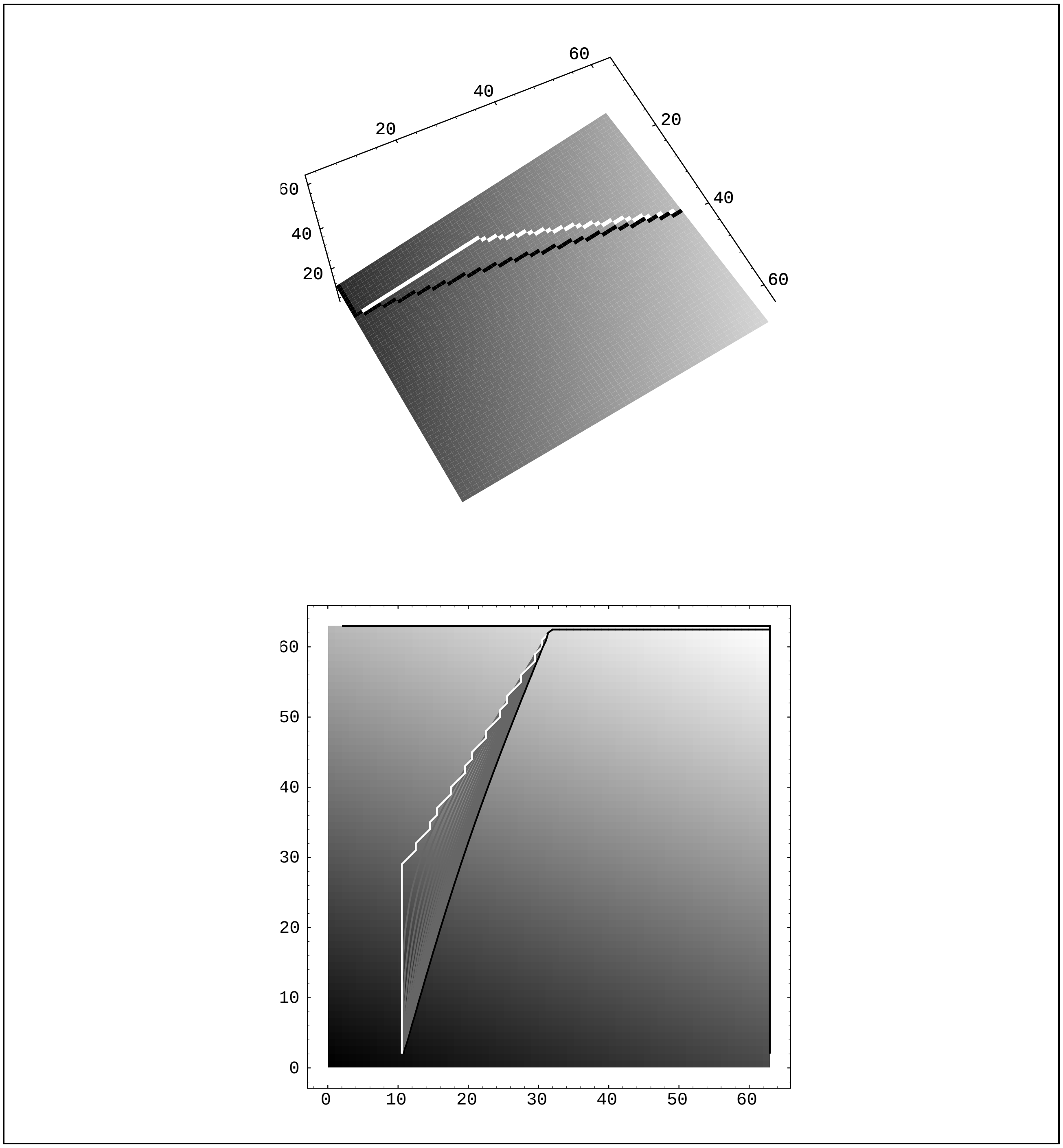

Figure 1: In the upper frame, the path obtained by the first stage is shown white on the tilted plane. It is refined in the second stage of the algorithm into the black path. The lower frame shows an orthographic projection of the surface from above. The white path is again the result of the first stage, the gray intermediate paths converge to the black refined path.

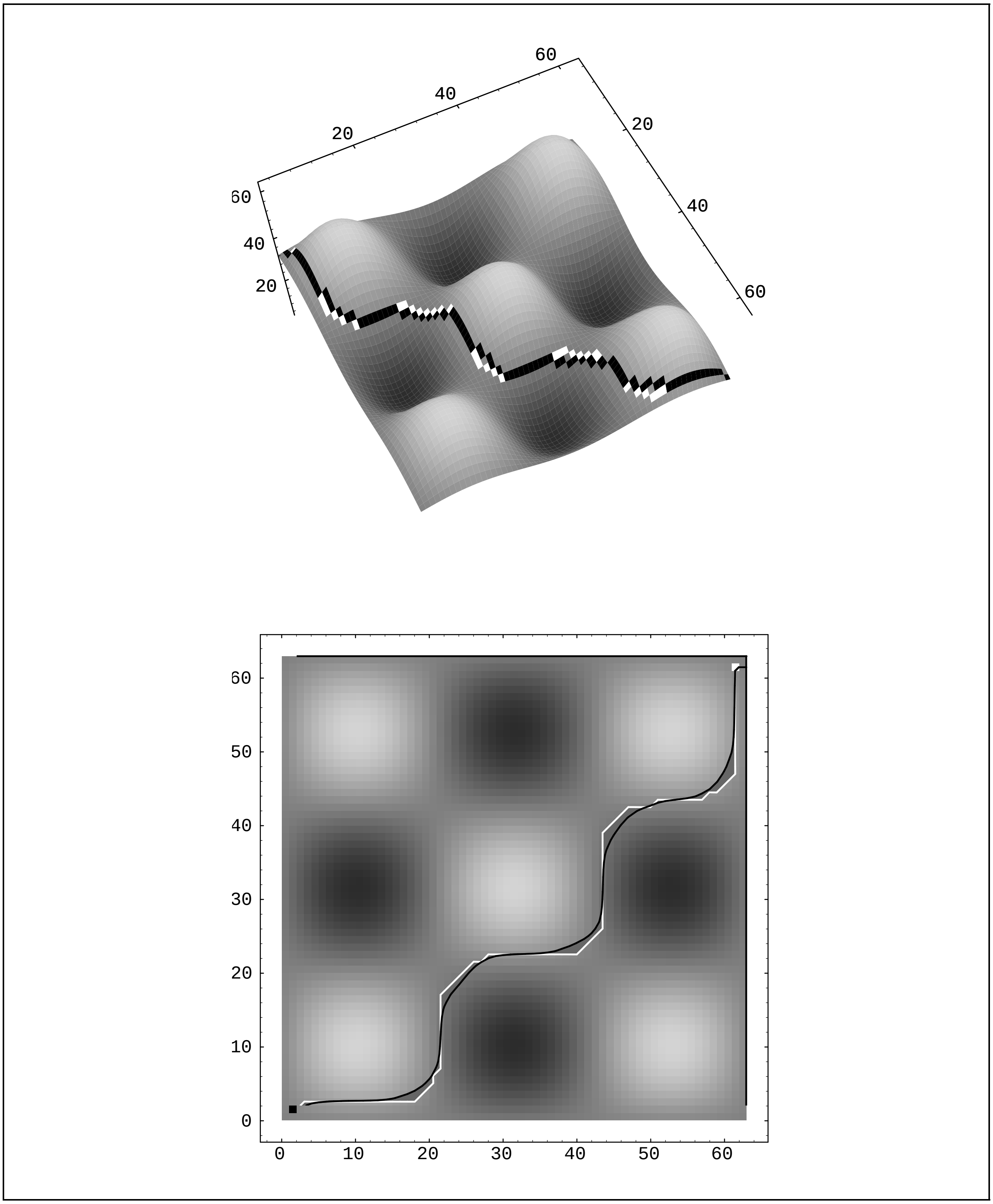

Figure 2: The white pixels on the surface belong to the estimated path and indicate where the paths differ. The estimated (white) path and the refined (black) one, nearly track an equal height contour, and avoid climbing on the hills.

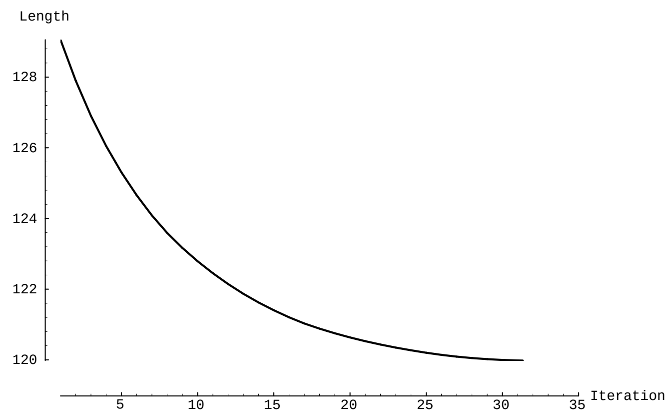

Figure 3: The rate of convergence of the second stage of the algorithm for the egg box surface. The length of the path in each iteration is linearly interpolated.

due to the limited accuracy of the underlying $3D$ length estimator, it only produces an approximation to the shortest paths. As seen in the examples, the quality of the approximation may vary. On highly regular and smooth surfaces, such as planes or spheres, local length estimation errors might add up along the digital paths and lead to significantly distorted approximations (see Fig. 1). On rough or irregular surfaces, local length estimation errors are likely to cancel out thanks to the unbiasedness of the $3D$ length estimator, and rather accurate approximations can be expected (see Figs. 2 and 4).

When different paths of lengths nearly equal to the minimal distance between the source and destination points exist, slight errors in minimum distance estimation could correspond to a large difference between the traces of the true shortest path and the approximation produced by the first stage of the algorithm. This means that the shortening flow stage might in the worst case converge to a local minimum that is distant from the global minimum. However, in that case, the local minimum would still be almost as deep as the global minimum, and the algorithm would not be trapped into insignificant local minima.

A curve evolution approach for the computation of geodesic curves on surfaces was used in the second stage. It is initialized by the approximation generated by the first stage, and a shorter geodesic curve is obtained by a geodesic curvature shortening flow. Since the second stage will always converge to the geodesic, the actual gain in terms of path length in the second stage essentially corresponds to the inaccuracy of the initial approximation. Fig. 3 exemplifies the rate of convergence of the iterative refinement procedure. Balancing the trade-off that arises in time-critical applications between the expected improvement in the path and the number of iterations allowed should eventually be application-dependent.

The $3D$ curve flow is represented by an equivalent $2D$ one, which is implemented by an efficient numerical algorithm for curve evolution, together with a procedure for keeping the end points fixed. The curve, propagating on the $3D$ surface, is projected onto the coordinates plane. This projection of the $3D$ flow into a $2D$ equivalent one permits an efficient computer implementation. The numerical algorithm is an iterative scheme based on central spatial derivatives. Thus, the accuracy of the final geodesic is of $\mathcal{O}(\Delta x^2)$ where Δx is the distance

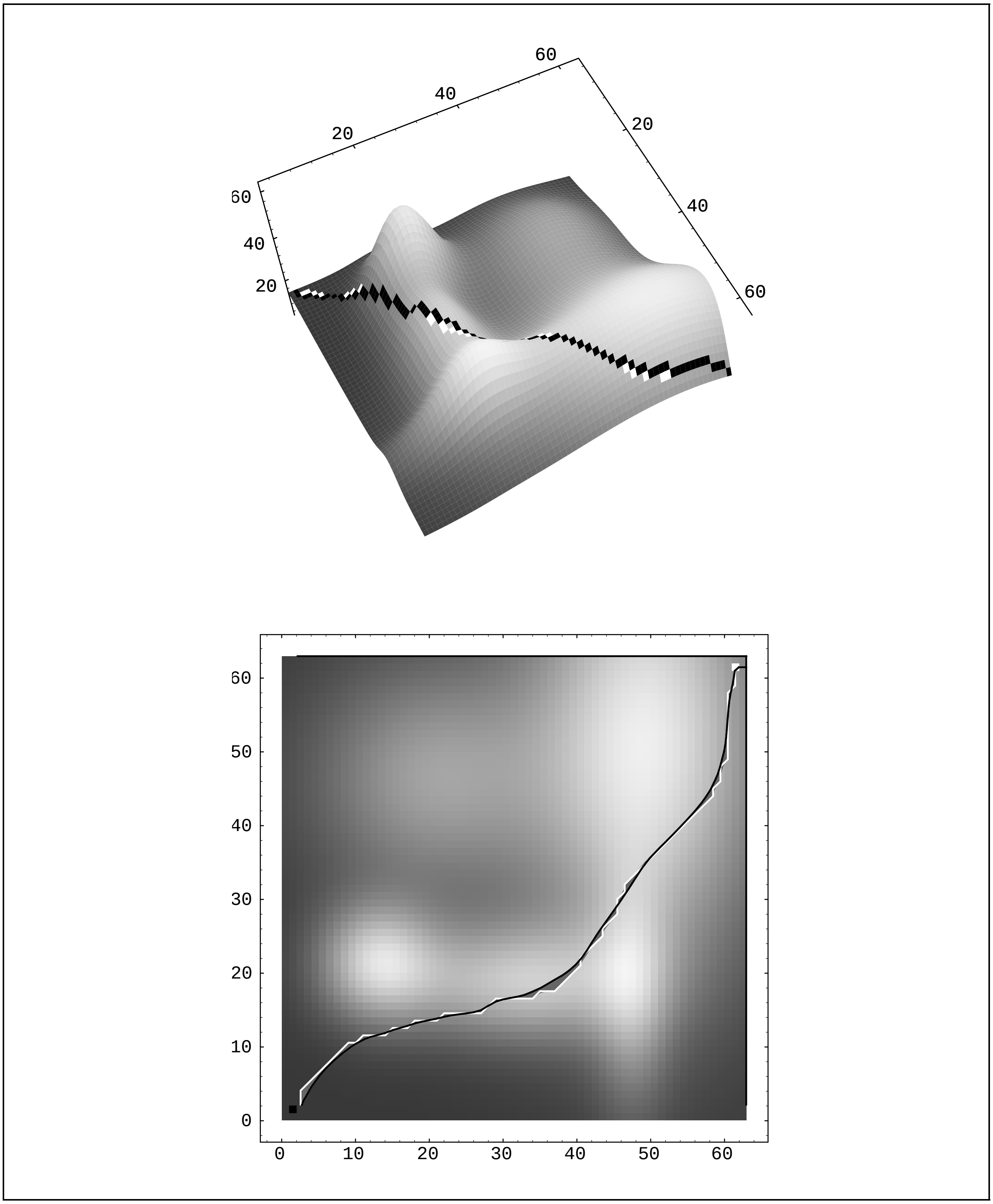

Figure 4: The white approximation is refined to the black geodesic path on a terrain-like surface.

between two neighboring grid points. It is obtained by selecting $\Delta t = \mathcal{O}(\Delta x^2)$ in the final iterations of the proposed numerical scheme. This bound can be verified by Taylor expansion of $\Phi(x, y, t)$. For detailed analysis of the accuracy of such numerical schemes see [8].

The geodesic curvature shortening flow is not limited to function surfaces. It is possible to handle complex surfaces, as was done by Chopp [4], by patching supporting planes which update each other. Each plane supports a piece of the surface. These planes share information at each iteration, so that a curve is defined and propagated on several coordinate planes simultaneously. In [4] the following strategy is used to "connect multiple patches": Use the distance function in the manifold space for initialization. The motion on each supporting plane is computed in the interior grid points. The values at the boundaries are taken from neighboring patches (by back projection via the manifold). The construction of the supporting coordinate planes is such that the boundary of any supporting plane is always the interior of another. For further details see [4].

The suggested combination of the two approaches exploits the computational efficiency of the graph search algorithm while compensating for estimation errors. Sub–voxel accuracy is achieved by using simple interpolation to find the location of the zero level set, at the last step of the shortening flow algorithm. Note that the two stage approach is rather general and allows various algorithms to be used as the first and second stages in the suggested framework.

Acknowledgments

We thank the referees for helping us improve the manuscript. We wish to thank A. Katz and N. Ram-on for testing the suggested algorithms. This research was supported in part by the Ollendorff Center of the Department of Electrical Engineering, by the Fund for the Promotion of Research at the Technion, and by the Applied Mathematical Science subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract Number DE-AC03-76SF00098.

References

[1] D. Adalsteinsson and J.A. Sethian, *"A Fast Level Set Method for Propagating Interfaces"*, *J. of Comp. Phys.*, Vol. 118, pp. 269-277, 1995.

[2] J.M. Beck, R.T. Farouki and J.K. Hinds, "Surface Analysis Methods," *IEEE CG&A*, pp. 18-37, 1986.

[3] D. L. Chopp, *Computing Minimal Surfaces Via Level Set Curvature Flow*, *J. of Comp. Phys.*, Vol. 106, pp. 77-91, 1993.

[4] D. L. Chopp, "Flow under geodesic curvature," Report 92-23, UCLA, May 1992.

[5] D. L. Chopp and J. A. Sethian, "Flow under curvature: Singularity formation, minimal surfaces, and geodesics," *J. Exper. Math.*, Vol 2(4), pp. 1-15, 1993.

[6] M. Gondran and M. Minoux, *Graphs and Algorithms*, Wiley, Chichester, 1984 (Ch. 2).

[7] M. Grayson, "Shortening embedded curves," *Annals of Mathematics*, Vol. 129, pp. 71-111, 1989.

[8] R. Kimmel, N. Kiryati and A.M. Bruckstein, "Analyzing and Synthesizing Images by Evolving Curves with the Osher-Sethian Method," Technical Report LBL-37950, UC Berkeley, November 1995.

[9] R. Kimmel and G. Sapiro, "Shortening Three Dimensional Curves via Two Dimensional Flows," *Computers & Mathematics with Applications*, Vol. 29, pp. 49-62, 1995.

[10] N. Kiryati and G. Székely, "Estimating Shortest Paths and Minimal Distance on Digitized Three Dimensional Surfaces," *Pattern Recognition*, Vol. 26, pp. 1623-1637, 1993.

[11] N. Kiryati and O. Kübler, "On Chain Code Probabilities and Length Estimators for Digitized Three Dimensional Curves," *Pattern Recognition*, Vol. 28, pp. 361-372, 1995.

[12] R. Malladi, J.A. Sethian, and B.C. Vemuri, "Shape Modeling with Front Propagation: A Level Set Approach", *IEEE Trans. PAMI*, Vol. 17, pp. 158-175, 1995.

[13] J.S.B. Mitchell, D.M. Mount and C.H. Papadimitriou, "The Discrete Geodesic Problem," *SIAM J. Comput.*, Vol. 16, pp. 647-668, 1987.

[14] J.S.B. Mitchell, D. Payton, and D. Keirsey, "Planning and reasoning for autonomous vehicle control," *International J. of Intelligent Systems*, Vol. 2, pp. 129-198, 1987.

[15] S.J. Osher and J.A. Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. of Comp. Phys.*, Vol. 79, pp. 12-49, 1988.

[16] E.L. Schwartz, A. Shaw and E. Wolfson, "A Numerical Solution to the Generalized Mapmaker's Problem: Flattening Nonconvex Polyhedral Surfaces," *IEEE Trans. PAMI*, Vol. 11, pp. 1005-1008, 1989.

[17] J.A. Sethian, "A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed," *J. Differential Geometry*, Vol. 31, pp. 131-161, 1990.

[18] J.A. Sethian and J. Strain, "Crystal growth and dendritic solidification," *J. of Comp. Phys.*, Vol. 98, 1992.

[19] M. Sussman, P. Smerake and S.J. Osher, "A Level Set Approach for Computing Solutions to Incompressible Two-Phased Flow", *J. of Comp. Phys.*, Vol. 114, pp. 146-159, 1994.

[20] E. Wolfson and E.L. Schwartz, "Computing Minimal Distances on Polyhedral Surfaces," *IEEE Trans. PAMI*, Vol. 11, pp. 1001-1005, 1989.