Pergamon

Computers Math. Applic. Vol. 29, No. 3, pp. 49–62, 1995

0898-1221/95 $9.50 + 0.00

0898-1221(94)00228-2

Shortening Three-Dimensional Curves via Two-Dimensional Flows

R. KIMMEL
Department of Electrical Engineering
Technion-Israel Institute of Technology
Haifa, Israel 32000

G. SAPIRO
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A.

(Received September 1993; accepted October 1993)

Abstract—In this paper, a curve evolution approach for the computation of geodesic curves on 3D surfaces is presented. The algorithm is based on deforming, via the *curve shortening flow*, an arbitrary initial curve ending at two given surface points. The 3D curve shortening flow is first transformed into an equivalent 2D one. This 2D flow is implemented, using an efficient numerical algorithm for curve evolution with fixed end points.

Keywords—Geodesic curve, Geodesic curvature, Curve shortening flow, Numerical implementation.

1. INTRODUCTION

The theory of curve evolution has recently been studied in a number of different research areas such as differential geometry [1–4], parabolic equations theory [5], numerical analysis [6], viscosity solutions [7–9], computer vision [4,10–14], and image processing [15]. In this work, an algorithm for computing geodesic curves, which is based on the theory of curve evolution, is presented.

One of the most studied curve flows is the *Euclidean geometric heat equation*. In this case, the curve evolves by its curvature vector. In the case of planar closed curves, this means that the curve deforms in the direction of the Euclidean normal, with velocity equal to the Euclidean curvature. Gage and Hamilton [1] proved that a simple convex curve converges to a round point when deforming according to this flow. Grayson [2] proved that a nonconvex curve converges to a convex one. Then, any simple curve converges to a round point when deforming according to the Euclidean geometric heat flow.

Grayson [16] extended the planar work to smooth curves immersed in a Riemannian surface. He proved that the curve remains smooth, and it either converges to a point or its curvature converges to zero in the C^∞ norm.

One of the great 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 for the length functional [16]. For this reason, the flow is also called *curve shortening*

We wish to thank J.A. Sethian for being always so open and generous in providing us his works. Especially, we thank him for the reports [17–19], which are closely connected to the work here described.

flow. Therefore, as pointed out by Grayson himself [16], deforming a curve by its curvature vector is a very efficient way of finding geodesic curves.

The geodesic computation approach presented here is based on the curve shortening flow, performed on 3D surfaces. Given the surface and two points on it, based on this flow, the algorithm deforms an arbitrary initial embedded curve which ends at these points. From Grayson's results, the curve remains smooth and embedded, and if the end points are fixed, it converges to a surface geodesic ending at the given points.[1]

For the computer implementation of the algorithm, the three-dimensional curve flow is first transformed into an equivalent two-dimensional one, this step being crucial. This two-dimensional curve flow is implemented based on a numerical algorithm derived from [6], together with a simple algorithm, motivated by [17], for keeping the end points fixed.

The remainder of this paper is as follows. Section 2 presents basic concepts on surface differential geometry. The corresponding three- and two-dimensional evolution equations are given in Section 3. Section 4 deals with the numerical implementation. Examples are given in Section 5, and concluding remarks in Section 6.

2. SURFACE DIFFERENTIAL GEOMETRY

In this section, basic concepts on surface differential geometry are presented. For details, see [20].

DEFINITION 1. *Let* $\mathbf{S}$ *be a regular three-dimensional surface,* $\mathbf{T}(p,\mathbf{S})$ *the tangent plane to* $\mathbf{S}$ *at the point* $p \in \mathbf{S}$, *and* $\vec{\mathbf{N}}(p)$ *the normal to the surface at a point* $p \in \mathbf{S}$. *A regular connected curve* $\mathcal{C}$ *in* $\mathbf{S}$ *is said to be a geodesic if for every* $p \in \mathcal{C}$, *the parameterization* $\alpha(s)$ *of a neighborhood of* p *in* $\mathcal{C}$ *by the arc-length* s *satisfies:*

$$\frac{\partial^2 \alpha}{\partial s^2} \parallel \vec{\mathbf{N}},$$

i.e.,

$$\frac{\partial^2 \alpha}{\partial s^2} \perp \mathbf{T}(p,\mathbf{S}).$$

Since

$$\frac{\partial^2 \alpha}{\partial s^2} = \kappa \vec{n},$$

where κ is the Euclidean curvature, and $\vec{n}$ the unit normal to the curve, we obtain that a curve is a geodesic if the vector $\kappa\vec{n}$ is perpendicular to the surface tangent plane (or parallel to the surface normal vector) at each of its points.

DEFINITION 2. *Let* $\mathcal{C}$ *be an oriented regular curve contained on an oriented surface* $\mathbf{S}$. *The geodesic curvature vector of the curve at a given point* $p \in \mathcal{C}$, *is defined as the component of the vector* $\kappa\vec{n}(p)$ *on the tangent plane* $\mathbf{T}(p,\mathbf{S})$.

The geodesic curvature vector is given by $\kappa_g \vec{\mathcal{N}}$, where $\vec{\mathcal{N}}$ is a unit vector in the direction of the intersection between the plane normal to the curve and the surface tangential plane, and

$$\kappa_g := \left\langle \kappa\vec{n}, \vec{\mathcal{N}} \right\rangle.$$

According to its definition, $\vec{\mathcal{N}}$ may be written as

$$\vec{\mathcal{N}} = \vec{\mathbf{t}} \times \vec{\mathbf{N}},$$

where $\vec{\mathbf{t}} \in \mathbf{T}(p,\mathbf{S})$ is the unit tangent vector to the curve $\mathcal{C} \in \mathbf{S}$. (See Figure 1).

[1] After this work was completed, we noticed that related work was independently done by Chopp and Sethian [18,19].

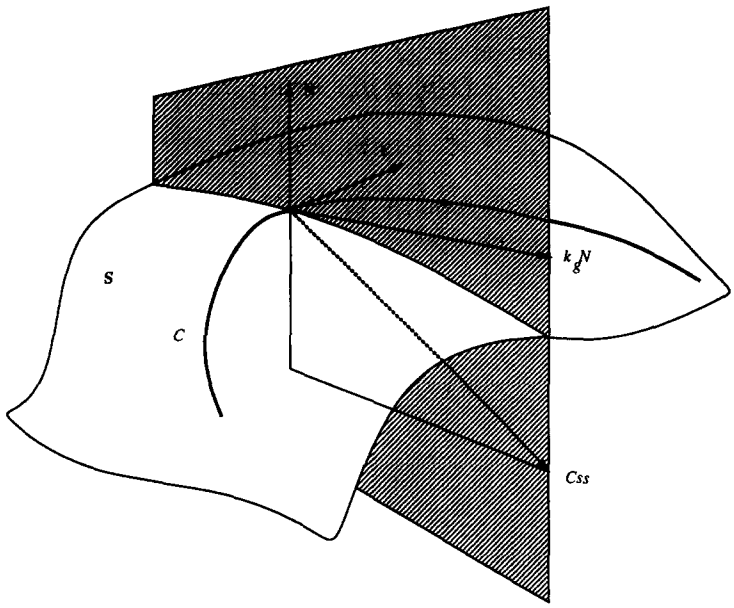


Figure 1. Geometry of the geodesic curvature vector.

DEFINITION 3. *κ_g is denoted as the geodesic curvature.*

From the definitions above, it follows that C is a geodesic curve if and only if $\kappa_g \equiv 0$. The following theorems show the importance of geodesic curves.

THEOREM 1. *There exists a geodesic curve from any point $s_1 \in \mathbf{S}$ to any point $s_2 \in \mathbf{S}$.*

THEOREM 2. *Let $\alpha : I \to \mathbf{S}$ be a regular parameterized curve with parameter proportional to arc-length. Suppose that the arc-length of α between any two points $s_1, s_2 \in I$, is smaller than or equal to the arc-length of any other regular parameterized curve joining $\alpha(s_1)$ to $\alpha(s_2)$. Then, α is a geodesic.*

The theorems above say that the minimal path between two points on the surface is always a geodesic. On the other hand, not every geodesic is a minimal path (it is only locally minimal). Therefore, finding geodesic curves on a surface, will give us paths which are at least locally minimal, with the possibility of also being globally minimal (depending on the surface structure). In the forthcoming sections, we present an algorithm which obtains, from an arbitrary curve, a geodesic one (both curves ending at the same given points).

3. THE SHORTENING FLOW

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 [16], 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. From Section 2, we obtain that the flow is given by

$$\frac{\partial \mathcal{C}(p,t)}{\partial t} = \kappa_g \vec{N}, \tag{1}$$

where p parameterizes the curve, and t stands for time. From (1), it follows immediately that the only possible stable curves are geodesic curves. Hence, applying (1) to the initial curve $\mathcal{C}_0$ will give us the required geodesic curve.

As we will see in Section 4, there exists a very efficient numerical algorithm for the computer implementation of planar curve flows. For this reason, we transform the 3D flow (1) into a 2D

one. Assume that the surface $\mathbf{S}$ is given as a graph, i.e., as a function $z(x,y)$, where x and y stand for the Cartesian coordinates on the (x,y)-plane. Then, the curve $\mathcal{C}(p,t)$ is given by the triplet $[x(p),y(p),z(p)]$, where $z(p)=z(x(p),y(p))$. Define the curve

$$\hat{\mathcal{C}} := [x(p),y(p)]$$

as the projection of the 3D curve $\mathcal{C}(p,t)$ on the (x,y)-plane. We show now how $\hat{\mathcal{C}}$ evolves when $\mathcal{C}$ deforms according to (1), and in this way obtain the desired 2D curve flow. For this, we have to find the corresponding velocity vector $\vec{\mathcal{V}}$ such that $\hat{\mathcal{C}}$ satisfies

$$\begin{cases} \frac{\partial \hat{\mathcal{C}}(p,t)}{\partial t} = \vec{\mathcal{V}} \\ \hat{\mathcal{C}}(p,0) = \hat{\mathcal{C}}_0, \end{cases}$$

when $\mathcal{C}(p,t)$ satisfies

$$\begin{cases} \frac{\partial \mathcal{C}(p,t)}{\partial t} = \kappa_g \vec{\mathcal{N}} \\ \mathcal{C}(p,0) = \mathcal{C}_0. \end{cases}$$

Of course, $\vec{\mathcal{V}}$ will be a function of $\hat{\mathcal{C}}$ and the graph-surface $\mathbf{S}$ (which together give enough information to restore the 3D curve $\mathcal{C}(p,t)$ from the 2D one $\hat{\mathcal{C}}(p,t)$).

We first compute the evolution velocity $\vec{v}$ of $\hat{\mathcal{C}}$, which corresponds to $\mathcal{C}$ evolving according to

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

A well-known result from the theory of planar (geometric) curve evolution is that the tangential component of the velocity vector of an evolving curve does not affect its trace (i.e., its geometry). The tangential component affects only the parameterization of the curve [21] (see also [14] for a proof). Therefore, in order to describe the geometric behavior of $\hat{\mathcal{C}}$, it is enough to observe the following 2D evolution rule

$$\frac{\partial \hat{\mathcal{C}}(p,t)}{\partial t} = v_n \hat{n},$$

where $\hat{n} = [n_1,n_2] = 1/\sqrt{x_p^2+y_p^2}\,[-y_p,x_p]$ is the unit normal[2] of the planar curve $\hat{\mathcal{C}}$, and v_n is the planar normal component of $\vec{v}$. The vector $\vec{v}$ itself is obtained by projecting $\vec{\mathcal{N}}$ onto the (x,y)-plane. Define π as the projection operator:

$$\pi \circ [x,y,z] := [x,y].$$

Then, v_n is obtained by first projecting $\vec{\mathcal{N}}$ onto the (x,y)–plane, i.e., $\pi \circ \vec{\mathcal{N}}$, and then projecting this result on the planar normal $\hat{n}$. See Figure 2. By recalling the definition of $\vec{\mathcal{N}}$, we write

$$\begin{aligned} v_n &= \left\langle \pi \circ \vec{\mathcal{N}}, \hat{n} \right\rangle \\ &= \left\langle \pi \circ (\vec{\mathbf{t}} \times \vec{\mathbf{N}}), [n_1,n_2] \right\rangle \\ &= \left\langle \pi \circ \left(\frac{\mathcal{C}_p}{|\mathcal{C}_p|} \times \frac{1}{\sqrt{1+z_x^2+z_y^2}} [-z_x,-z_y,1] \right), [n_1,n_2] \right\rangle \\ &= \sqrt{\frac{1}{1+z_x^2+z_y^2}} [(1+z_y^2)n_1^2 + (1+z_x^2)n_2^2 - 2n_1 n_2 z_x z_y]. \end{aligned}$$

Since $\vec{v}$ is obtained by projecting $\vec{\mathcal{N}}$ onto the plane, and v_n by the projection of $\vec{v}$ onto the normal direction, the normal component of the velocity $\vec{\mathcal{V}}$ is given by $\kappa_g v_n \hat{n}$. We, therefore,

[2]The x,y,t,p,s sub-indexes stand for partial derivatives.

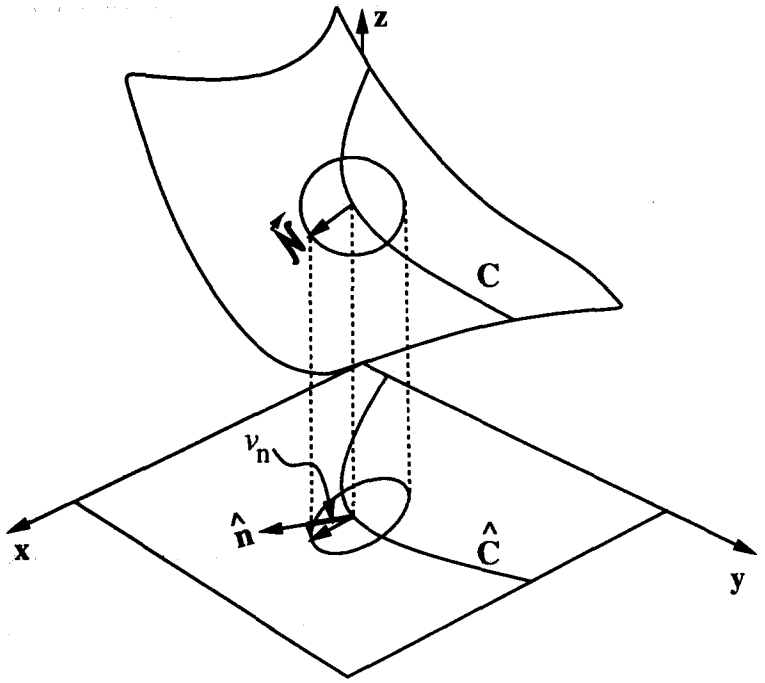


Figure 2. Geometry of the constant velocity projection.

proceed and compute κ_g just as a function of the planar curve $\hat{C}$ and the graph-surface $\mathbf{S}$, i.e., $\kappa_g = F(z_x, z_y, z_{xx}, z_{xy}, z_{yy}, \hat{n}, \hat{\kappa})$, where $\hat{\kappa}$ is the curvature of $\hat{C}$ and $z_x, z_y, \ldots$ are the surface partial derivatives. Following the geodesic curvature definition, we obtain

$$\begin{aligned}\kappa_g &= \left\langle \kappa\vec{n}, \vec{\mathcal{N}} \right\rangle \\ &= \left\langle \mathcal{C}_{ss}, \vec{\mathbf{t}} \times \vec{\mathbf{N}} \right\rangle,\end{aligned}$$

where $\mathcal{C}_{ss} = \kappa\vec{n}$ is the second derivative of $\mathcal{C}$ with respect to its arc-length s. Assume that $\hat{\mathcal{C}}$ is parameterized by its arc-length $\hat{s}$. Define the metric of $\mathcal{C}$ as

$$g := |\mathcal{C}_{\hat{s}}| = \sqrt{x_{\hat{s}}^2 + y_{\hat{s}}^2 + z_{\hat{s}}^2}.$$

Then, $\mathcal{C}_s = \mathcal{C}_{\hat{s}}/g$, and

$$\begin{aligned}\mathcal{C}_{ss} &= \frac{\partial}{\partial s}\left(\frac{\mathcal{C}_{\hat{s}}}{g}\right) \\ &= \frac{\mathcal{C}_{\hat{s}\hat{s}}}{g^2} - \frac{g_{\hat{s}}}{g^3}\mathcal{C}_{\hat{s}}.\end{aligned}$$

Observe that the second term of $\mathcal{C}_{ss}$ is parallel to $\vec{\mathbf{t}}$. Therefore,

$$\begin{aligned}\kappa_g &= \left\langle \frac{\mathcal{C}_{\hat{s}\hat{s}}}{g^2} - \frac{g_{\hat{s}}}{g^3}\mathcal{C}_{\hat{s}}, \vec{\mathbf{t}} \times \vec{\mathbf{N}} \right\rangle \\ &= \left\langle \frac{\mathcal{C}_{\hat{s}\hat{s}}}{g^2}, \vec{\mathbf{t}} \times \vec{\mathbf{N}} \right\rangle.\end{aligned}$$

By applying the chain rule to $z_{\hat{s}}$ and $z_{\hat{s}\hat{s}}$, we obtain

$$\kappa_g = \frac{1}{g^3\sqrt{1 + z_x^2 + z_y^2}}\left[\left(1 + z_x^2 + z_y^2\right)\left(x_{\hat{s}}y_{\hat{s}\hat{s}} - y_{\hat{s}}x_{\hat{s}\hat{s}}\right) + \left(z_y x_{\hat{s}} - z_x y_{\hat{s}}\right)\left(z_{xx}x_{\hat{s}}^2 + 2z_{xy}x_{\hat{s}}y_{\hat{s}} + z_{yy}y_{\hat{s}}^2\right)\right].$$

Since $\hat{s}$ is the arc-length parameterization of the curve $\hat{\mathcal{C}}$, we have

$$\hat{\kappa} = x_{\hat{s}} y_{\hat{s}\hat{s}} - x_{\hat{s}\hat{s}} y_{\hat{s}},$$
$$\hat{n} = [n_1, n_2] = [-y_{\hat{s}}, x_{\hat{s}}],$$

and

$$\kappa_g = \frac{1}{g^3\sqrt{1+z_x^2+z_y^2}} \left[\left(1+z_x^2+z_y^2\right) \hat{\kappa} + \langle \nabla z, \hat{n} \rangle \left(z_{xx} x_{\hat{s}}^2 + 2 z_{xy} x_{\hat{s}} y_{\hat{s}} + z_{yy} y_{\hat{s}}^2 \right) \right].$$

Therefore, while $\mathcal{C}$ evolves according to (1), its projection $\hat{\mathcal{C}}$ onto the (x, y) plane evolves according to

$$\hat{\mathcal{C}}_t = \left[\frac{1}{\eta} \left(\hat{\kappa} + \psi \frac{\langle \nabla z, \hat{n} \rangle}{1+z_x^2+z_y^2} \right) \right] \hat{n}, \tag{2}$$

where

$$\eta(\hat{n}, \nabla z) := n_1^2 (1+z_y^2) + n_2^2 (1+z_x^2) - 2 z_x z_y n_1 n_2, \tag{3}$$

and

$$\psi(\hat{n}, z_{xx}, z_{xy}, z_{yy}) := z_{xx} n_2^2 - 2 z_{xy} n_1 n_2 + z_{yy} n_1^2. \tag{4}$$

Solving (2) is equivalent to solving (1). In the next section, we present a numerical algorithm for the computer implementation of (2), while keeping the end points fixed.

4. NUMERICAL IMPLEMENTATION

We now present the basic concepts of the numerical algorithm that we have used for the implementation of equation (2). The basic algorithm is due to Osher and Sethian. For detailed analysis of this algorithm, see [6,22,23].

Let $\hat{\mathcal{C}}(p,t) : S^1 \times [0,\tau) \to \mathbb{R}^2$ be a family of planar curves satisfying the following evolution equation:

$$\frac{\partial \hat{\mathcal{C}}}{\partial t} = \beta \hat{n}. \tag{5}$$

(Recall that the tangential component of the velocity can be ignored.)

A number of problems must be solved when implementing curve evolution equations such as (5) on digital computer. Accuracy and stability are general requirements for any numerical algorithm. The numerical algorithm must approximate the evolution equation, and it must be robust. Sethian [22] has shown that a simple, Lagrangian, difference approximation requires an impractically small time step in order to achieve stability. The basic problem with Lagrangian formulations is that the marker particles on the evolving curve may come very close during the evolution. This can be solved by a redistribution of the marker particles, altering the equations of motion in a nonobvious way.

Osher and Sethian [6,22] proposed an algorithm for curve (and surface) evolution for the reliable numerical solution of this and even more complicated problems. They proposed to observe an implicit representation of the evolving curve, where the propagating curve is embedded as a level set of a higher dimensional function.

The embedding is as follows: the curve $\hat{\mathcal{C}}(p,t)$ is represented by the zero level set of a smooth and Lipschitz continuous function $\Phi : \mathbb{R}^2 \times [0,\tau) \to \mathbb{R}$. In the following, we assume that Φ is

negative in the interior and positive in the exterior of the zero level set. Consider the zero level set, defined by

$$\left\{\vec{\mathcal{X}}(t) \in \mathbb{R}^2 \,:\, \Phi(\vec{\mathcal{X}}, t) = 0\right\}. \tag{6}$$

We have to find an evolution equation of Φ, such that the evolving curve $\hat{\mathcal{C}}(t)$ is given by the evolving zero level $\vec{\mathcal{X}}(t)$, i.e.,

$$\hat{\mathcal{C}}(t) \equiv \vec{\mathcal{X}}(t). \tag{7}$$

By differentiation (6) with respect to t, we obtain:

$$\nabla\Phi\left(\vec{\mathcal{X}}, t\right) \cdot \vec{\mathcal{X}}_t + \Phi_t\left(\vec{\mathcal{X}}, t\right) = 0. \tag{8}$$

Note that for the zero level, the following relation holds:

$$\frac{\nabla\Phi}{\|\nabla\Phi\|} = -\hat{n}. \tag{9}$$

In this equation, the left side uses terms of the function Φ, while the right side is related to the curve $\hat{\mathcal{C}}$. The combination of equations (5) to (9) gives

$$\Phi_t = \beta\|\nabla\Phi\|, \tag{10}$$

and the curve $\hat{\mathcal{C}}$, evolving according to (5), is obtained by the zero level set of the function Φ, which evolves according to (10). Sethian [22] called this scheme an *Eulerian formulation* for front propagation, because it is written in terms of a fixed coordinate system. Even if the curve does not develop singularities, and no topological changes occur, the embedding process makes the algorithm more accurate and stable.

The next step of the algorithm consists of the discretization of the equation (10). If singularities cannot develop during the evolution, as in the Euclidean shortening flows, a straightforward discretization can be performed [6]. If singularities can develop, as in the case of $\beta = 1$, a special discretization must be implemented. In this case, the implementation of the evolution of Φ is based on a *monotone* and *conservative* numerical algorithm, derived from the theory of hyperbolic conservation laws [6,24]. For a large class of β-functions of this type, this numerical scheme automatically obeys the entropy condition, i.e., the physically correct solution derived from Hüygens-type principles [22,24]. In general, the numerical algorithm must give the corresponding viscosity solution [8,25]. For velocity functions such as $\beta = 1 + \epsilon\kappa$, a combination of both methods is used [6].

It is important to note that the discretization of the evolution equations is performed on a fixed *rectangular grid* [22] (see also the Appendix). This rectangular grid can be associated with the *pixel* grid of digital images, making this discretization method natural for image processing.

Since the evolving curve is given by the level set of the function Φ, we have to find this level set (Φ is discrete now). This is done by using a very simple (linear) contour finding algorithm as the one described in [23].

The Osher-Sethian described algorithm 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 [17]). This change is done 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. The function Φ is changed in such a way that the weighted average on the neighborhood of the end points is equal to zero. In our examples, we have actually added two lines connecting the curve end points to the picture boundary. This way, the curve divides the image in two, and the Φ function is defined positive on one part and negative on the other.

A stopping condition must be added to the algorithm. This condition can be related either to a threshold value on the geodesic curvature or to small changes in the evolving curve.

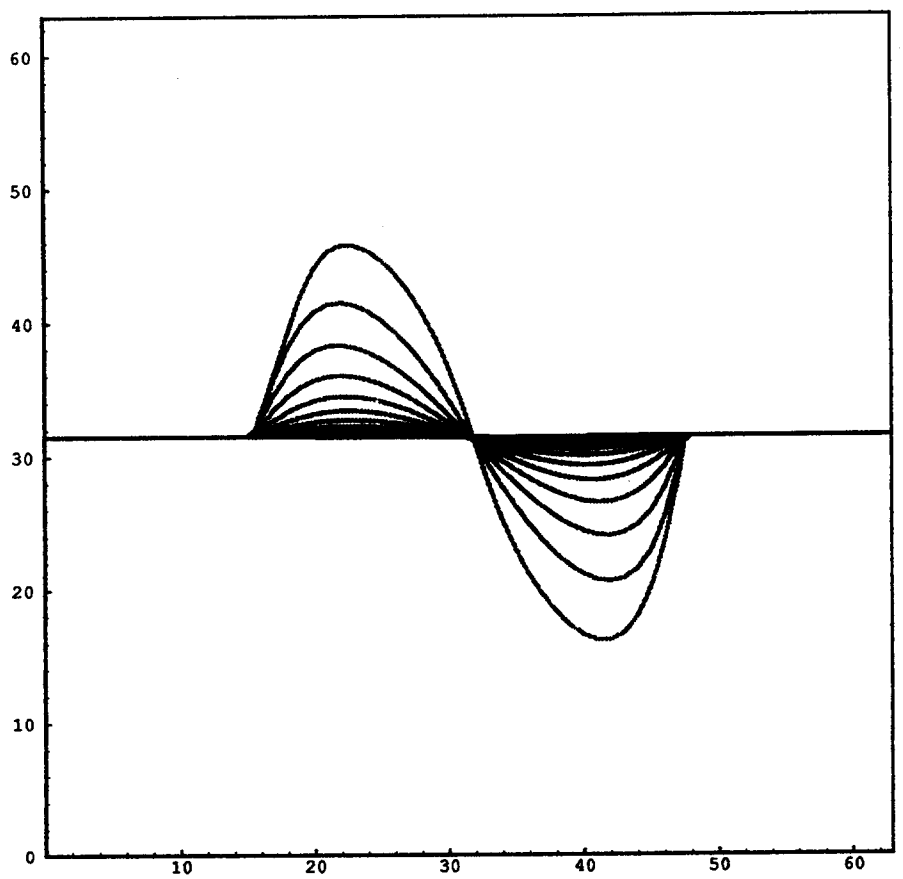
We conclude this section by remarking the proposed algorithm for curve shortening (geodesic computation):

1. Initialize Φ_0, using $\pi \circ \mathcal{C}_0$.
2. Evolve Φ according to (2) and (10), by using the corresponding numerical approximation described in the Appendix.
3. Adapt Φ, to keep the end points fixed.
4. Find the zero level set, $\mathcal{C}$ is obtained as $z(\hat{\mathcal{C}})$.
5. Check the stopping condition and according to it, go to 2 or stop.

5. EXAMPLES

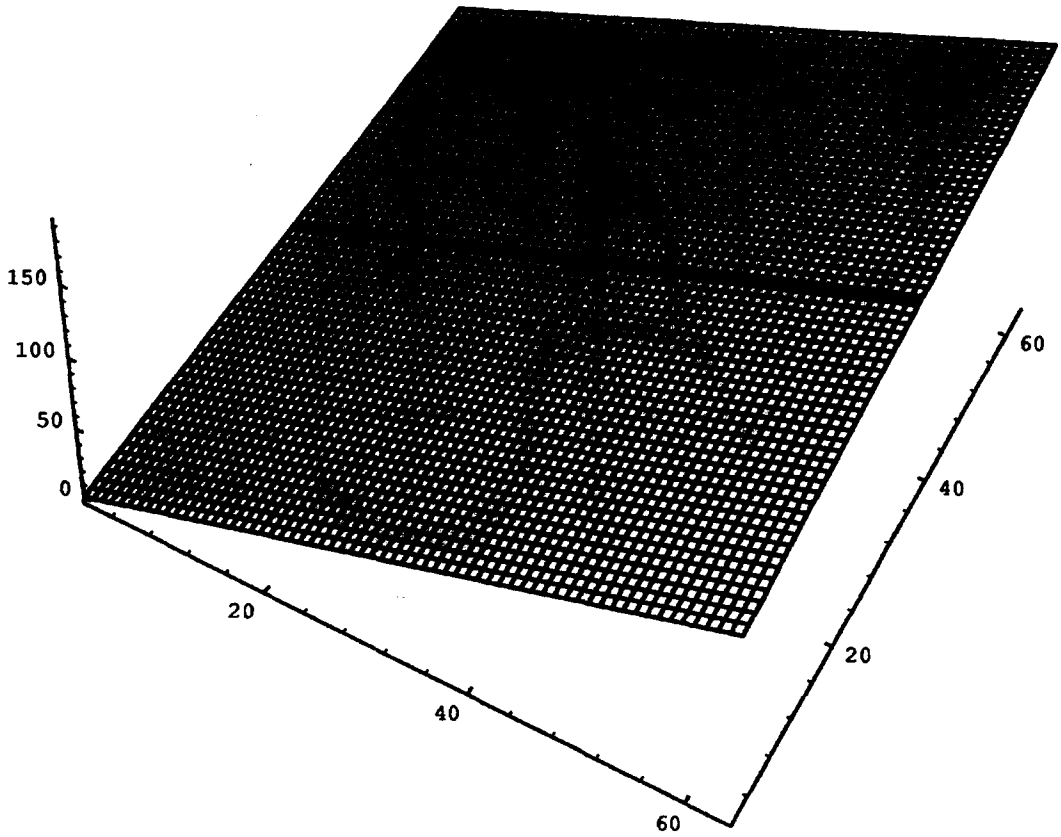
The above mentioned algorithm was implemented for different graph surfaces. The following pictures show a number of examples, where the black curve is the final geodesic.

The first simple example presents an initial *sin* curve on a tilted plane. The initial curve converges to the straight line connecting the end points. Straight lines are the only possible geodesic curves connecting two points on a plane. See Figure 3. In the second example, a *sin*-type initial curve on an "egg-box" surface is presented. The curve converges to a geodesic curve as shown in Figure 4. The third example shows a Gaussian mountain, where a geodesic curve is reached from a *sin*-type initial curve (Figure 5). Figure 6 shows the dependence of the result on the initial curve.



(a) Top view (the $\hat{\mathcal{C}}$ curve) of the process.

Figure 3. A simple planar example. The final geodesic curve is a straight line as expected.



(b) The final geodesic curve is presented on the 3D surface.

Figure 3. (cont.)

6. CONCLUDING REMARKS

In this paper, a curve evolution approach for the computation of geodesic curves on surfaces was presented. Given a 3D graph-surface and two points on it, we want to find a surface geodesic curve ending at these points. The algorithm starts from an arbitrary initial curve on the surface, which ends at the given points, and the geodesic curve is obtained by shortening it via the classical *curve shortening flow*. The 3D curve flow is represented by an equivalent 2D flow, which was implemented by an efficient numerical algorithm for curve evolution, together with a procedure for keeping the end points fixed.

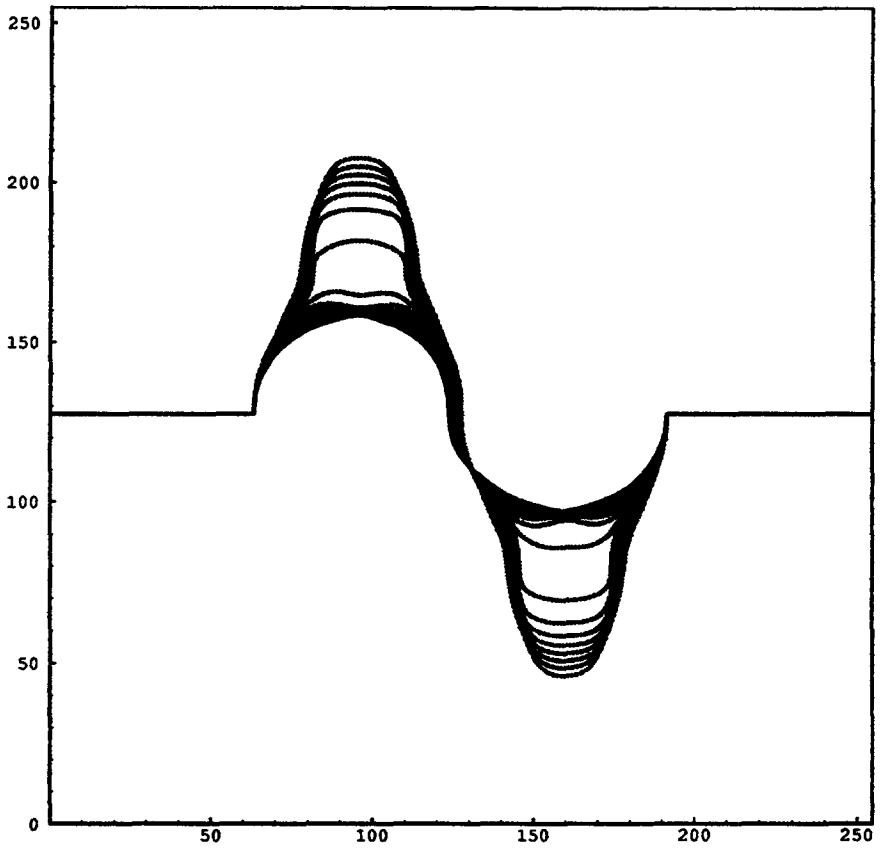
Geodesics can also be computed solving the Euler-Lagrange equations of the variational problem associated with the length functional (see, for example, [26]). The approach presented here can be interpreted as an efficient way of functional minimization (with constraints).

The main advantages of the proposed approach are the following:

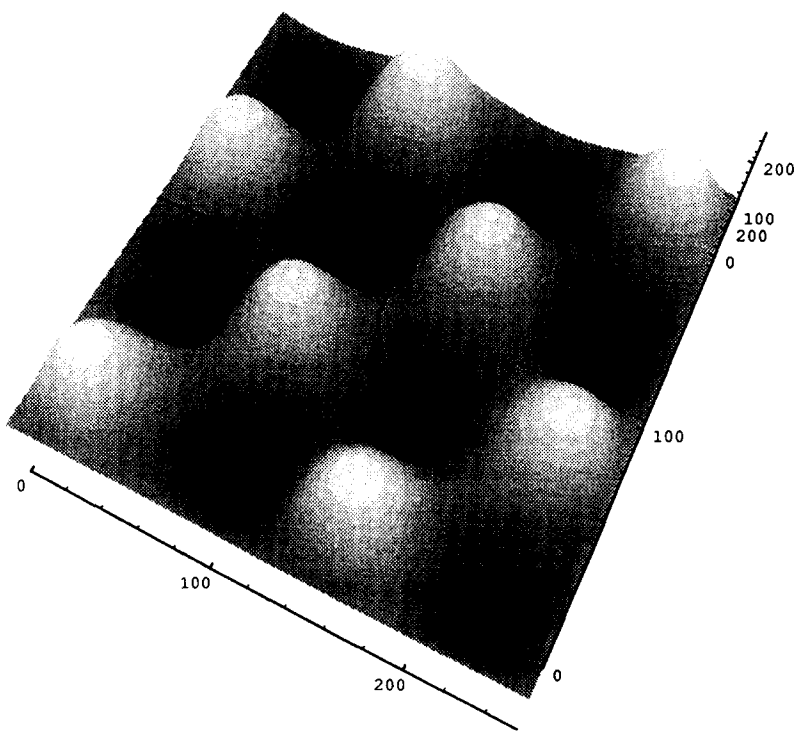
1. The curve shortening flow shrinks the curve as fast as possible.
2. The algorithm is based on the well-developed theory of curve evolution.
3. The computer implementation is based on a consistent and efficient numerical algorithm.

The computation of geodesic curves using graph search algorithms suffer from 'digital bias' problems, see [27]. The proposed algorithm can be used for correcting paths already computed by these methods. Note also that the algorithm can run several initial curves at the same time, and sub-pixel accuracy can be achieved.

In general, we have presented an approach for performing curve evolution, under the constraint that the curve remains on a given surface. This approach can be used for other curve flows as well. It also opens interesting theoretical questions, for example, the relation between weak (viscosity) solutions of the evolving 3D curve and those of the 2D one.

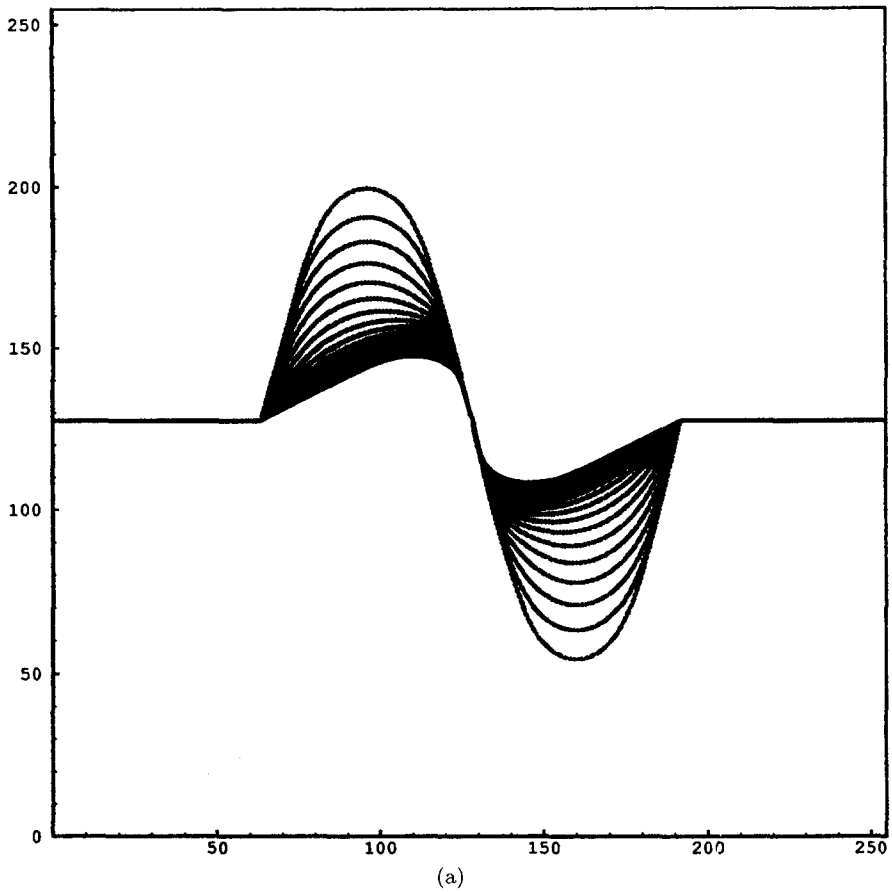


(a)

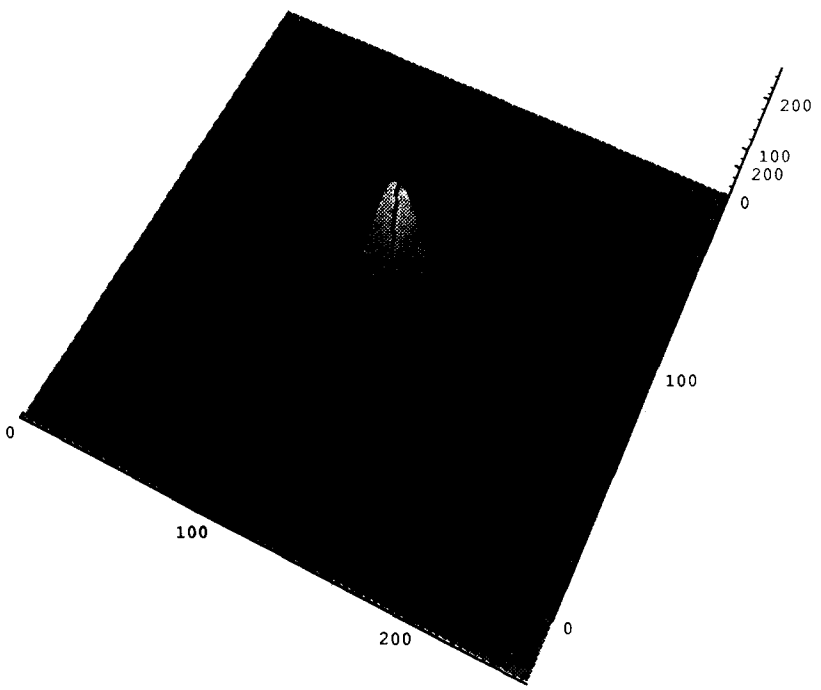


(b)

Figure 4. Example with an "egg-box" surface. The initial curve ($\hat{C}_0$) is given by a sin function on the (x, y)-plane.

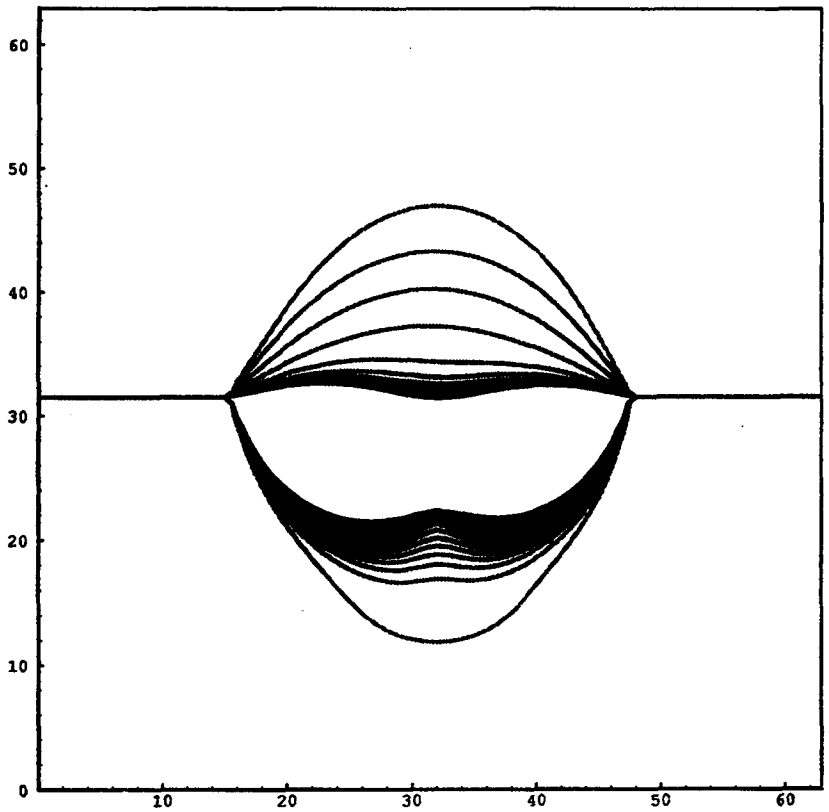


(a)

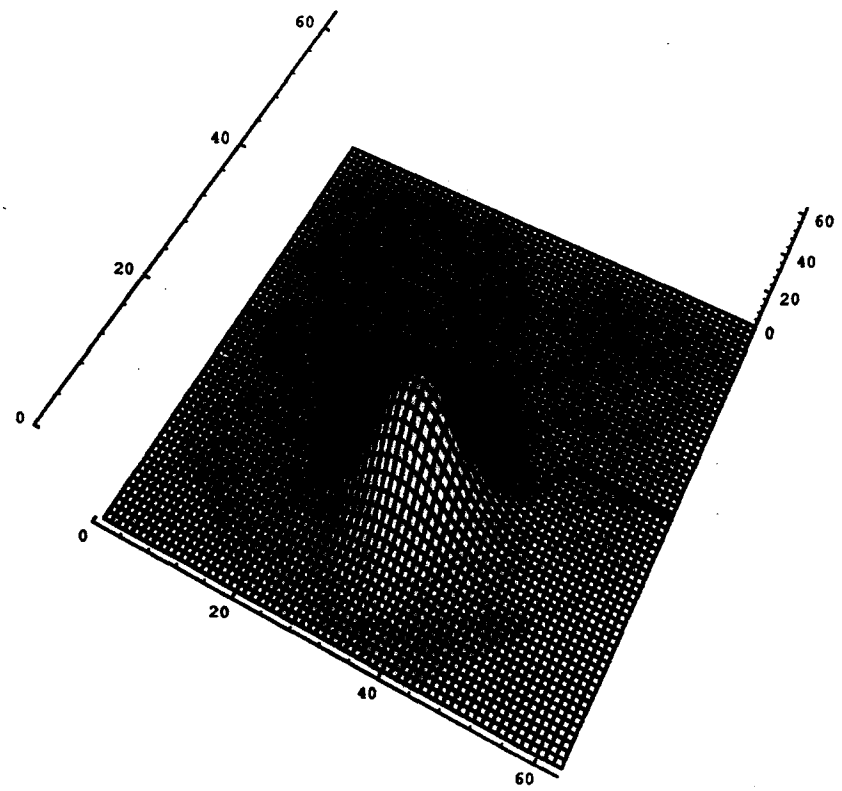


(b)

Figure 5. Example with a Gaussian surface. The initial curve is given by a sin function on the (x, y)-plane.

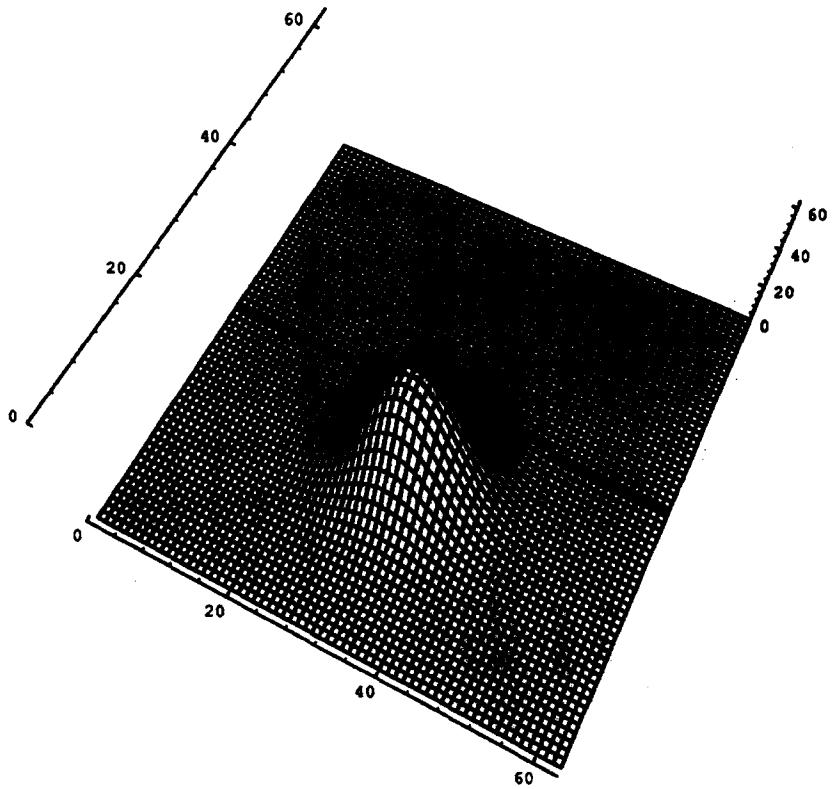


(a)



(b)

Figure 6. Example of two different initial curves evolving according to the proposed shortening flow.



(c)

Figure 6. (cont.)

APPENDIX

The exact numerical equation corresponding to (2) and (10) is now described. If the curve $\hat{C}$ is a level set of a function Φ, then its curvature $\hat{\kappa}$ can be computed as [22]:

$$\hat{\kappa} = \frac{\Phi_{yy}\Phi_x^2 - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_{xx}\Phi_y^2}{(\Phi_x^2 + \Phi_y^2)^{3/2}},$$

and the surface evolution equation corresponding to (10) is given by

$$\begin{aligned}\Phi_t &= \kappa_g v_n |\nabla\Phi| \\ &= \frac{\Phi_x^2\Phi_{yy} - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_y^2\Phi_{xx} + (z_x\Phi_x + z_y\Phi_y)(z_{xx}\Phi_y^2 - 2\Phi_x\Phi_y z_{xy} + z_{yy}\Phi_x^2)\frac{1}{1+z_x^2+z_y^2}}{\Phi_x^2(1+z_y^2) + \Phi_y^2(1+z_x^2) - 2z_x z_y\Phi_x\Phi_y}.\end{aligned}$$

Define

$$\Phi_{ij}^n := \Phi(i\Delta x, j\Delta y, n\Delta t).$$

The time derivative is implemented using forward difference approximation

$$\frac{\partial\Phi}{\partial t} \approx \frac{\Phi_{ij}^{n+1} - \Phi_{ij}^n}{\Delta t},$$

and the space derivatives are implemented using central approximation

$$\begin{aligned}\frac{\partial\Phi}{\partial x} &\approx \frac{\Phi_{i+1,j}^n - \Phi_{i-1,j}^n}{2h}, \\ \frac{\partial^2\Phi}{\partial x^2} &\approx \frac{\Phi_{i+1,j}^n - 2\Phi_{i,j}^n + \Phi_{i-1,j}^n}{h^2}, \\ \frac{\partial^2\Phi}{\partial x\partial y} &\approx \frac{\Phi_{i+1,j+1}^n + \Phi_{i-1,j-1}^n - \Phi_{i+1,j-1}^n - \Phi_{i-1,j+1}^n}{4h^2},\end{aligned}$$

where $h = \Delta x = \Delta y$. The same spatial derivative approximation is applied to Φ_y, Φ_{yy}, and to the surface derivatives $(z_x, z_y, z_{xx}, z_{yy}, z_{xy})$. Then, the 2D curve flow (2) (see also (10)) is implemented by taking a forward derivative approximation in time, which yields the desired numerical scheme.

REFERENCES

1. M. Gage and R.S. Hamilton, The heat equation shrinking convex plane curves, *J. Differential Geometry* **23**, 69–96 (1986).
2. M. Grayson, The heat equation shrinks embedded plane curves to round points, *J. Differential Geometry* **26**, 285–314 (1987).
3. G. Sapiro and A. Tannenbaum, On affine plane curve evolution, *Journal of Functional Analysis* (to appear).
4. G. Sapiro and A. Tannenbaum, On invariant curve evolution and image analysis, *Indiana Journal of Mathematics* (in press).
5. S. Angenent, Parabolic equations for curves on surfaces, Part II. Intersections, blow-up, and generalized solutions, *Annals of Mathematics* **133**, 171–215 (1991).
6. S.J. Osher and J.A. Sethian, Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, *Journal of Computational Physics* **79**, 12–49 (1988).
7. Y.G. Chen, Y. Giga and S. Goto, Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations, *J. Differential Geometry* **33**, 749–786 (1991).
8. M.G. Crandall, H. Ishii and P.L. Lions, User's guide to viscosity solutions of second order partial linear differential equations, *Bulletin of the American Mathematical Society* **27**, 1–67 (1992).
9. L.C. Evans and J. Spruck, Motion of level sets by mean curvature, I, *J. Differential Geometry* **33**, 635–681 (1991).
10. B.B. Kimia, A. Tannenbaum and S.W. Zucker, Toward a computational theory of shape: An overview, In *Lecture Notes in Computer Science*, pp. 402–407, Springer-Verlag, New York, (1990).
11. B.B. Kimia, A. Tannenbaum and S.W. Zucker, Entropy scale-space, In *Proc. of Visual Form Workshop Capri*, Plenum Press, (May 1991).
12. R. Kimmel and A.M. Bruckstein, Shape from shading via level sets, CIS Publication 9209, Department of Computer Science Technion, I.I.T., Haifa, Israel, (June 1992) (submitted).
13. R. Malladi, J.A. Sethian, and B.C. Vemuri, A topology independent shape modeling scheme, In *Proceedings of the SPIE—Geometric Methods in Computer Vision II*, pp. 246–255, (July 1993).
14. G. Sapiro and A. Tannenbaum, Affine invariant scale-space, *International Journal of Computer Vision* (in press).
15. L. Alvarez, P.L. Lions and J.M. Morel, Image selective smoothing and edge detection by nonlinear diffusion, *SIAM J. Numer. Anal.* **29**, 845–866 (1992).
16. M. Grayson, Shortening embedded curves, *Annals of Mathematics* **129**, 71–111 (1989).
17. D.L. Chopp, Computing minimal surfaces via level set curvature flow, Ph.D. Dissertation, Mathematics Department, Lawrence Berkeley Laboratory, (1991).
18. D.L. Chopp, Flow under geodesic curvature, Report 92–23, UCLA, (May 1992).
19. D.L. Chopp and J.A. Sethian, Flow under curvature: Singularity formation, minimal surfaces, and geodesics, (preprint), (October 1992).
20. M.P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, New Jersey, (1976).
21. C.L. Epstein and M. Gage, The curve shortening flow, In *Wave Motion: Theory, Modeling, and Computation* (Edited by A. Chorin and A. Majda), Springer-Verlag, New York, (1987).
22. J.A. Sethian, A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed, *J. Differential Geometry* **31**, 131–161 (1989).
23. J.A. Sethian and J. Strain, Crystal growth and dendritic solidification, *Journal of Computational Physics* **98** (1992).
24. G.A. Sod, *Numerical Methods in Fluid Dynamics*, Cambridge University Press, Cambridge, (1985).
25. S. Osher and C.W. Shu, High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations, *SIAM J. Numer. Anal.* **28**, 907–922 (1991).
26. J.M. Beck, R.T. Farouki and J.K. Hinds, Surface analysis methods, *IEEE CG&A*, 18–37 (1990).
27. J.S.B. Mitchell, D. Payton and D. Keirsey, Planning and reasoning for autonomous vehicle control, *International Journal of Intelligent Systems* **2**, 129–198 (1987).