Diffusion Generated Motion of Curves on Surfaces

Barry Merriman* and Steven J. Ruuth†

June 1, 2006

Abstract

We present a new method for computing the curvature-driven motion of a curve constrained to move on a given surface. It is based on the Diffusion Generated Motion algorithm, and retains both the novel simplicity of that method, as well as the natural extension to curves with junctions, to general geometric motion laws, and to higher dimensions. The result is an extremely simple algorithm for curvature-dependent motion on surfaces, wherein the only evolution operation is linear diffusion in three dimensional Euclidean space.

Keywords: Diffusion generated motion, curvature motion, implicit surfaces, diffusion, closest point representation

1 Introduction

The original Diffusion Generated Motion algorithm is a simple procedure introduced in [11, 12] to move a curve in the plane by the curvature motion law. Many generalizations and applications of the method have been explored [10, 13, 8, 15, 16, 17, 9, 4] and the rigorous theory is developed in [5, 1, 7, 8]. Our purpose here is to extend the method to moving curves on surfaces while retaining the extreme practical simplicity of the original diffusion generated motion procedure.

As a computational problem, moving curves on surfaces has previously been addressed using front tracking and level-set methods [2, 3]. While these approaches have the advantage that they allow for arbitrary motion laws, substantial technical difficulties can arise from geometric complications such as curves with triple-point junctions, surfaces with boundaries or without orientation, or extensions to higher dimensions. Here, we present a fundamentally different approach based on

*Department of Mathematics, University of California, Los Angeles CA (barrym@ucla.edu).

†Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, V5A 1S6 Canada (sruuth@sfu.ca). The work of this author was partially supported by a grant from NSERC Canada.

the Diffusion Generated Motion algorithm, which gives a more limited class of geometric motion laws, but is otherwise notable for the resulting simplicity and robustness of the method.

The paper is organized as follows. In Section 2 the model for moving a curve on a surface is described and in Section 3 a review of Diffusion Generated Motion is provided. Sections 4 and 5 generalize Diffusion Generated Motion to the case of curves on surfaces. Section 6 gives a justification of the method and Section 7 provides an extension to junctions. In Section 8 a variety of numerical experiments are given to illustrate the numerical properties of the method. Finally Section 9 concludes with a summary of the main findings of the paper and a discussion of future work.

2 Model Problem for Motion of a Curve on a Surface

Consider a curve $\mathcal{C}$ that is constrained to move within a surface $\mathcal{S}$. As is illustrated in Figure 1, our goal is to evolve $\mathcal{C}$ according to a local velocity given by the curvature of $\mathcal{C}$ *within* $\mathcal{S}$,

$$\frac{d\mathcal{C}}{dt} = \kappa_{\mathcal{S}}[\mathcal{C}].$$

Assuming $\mathcal{S}$ is isometrically embedded in $\mathbb{R}^3$, the curvature $\kappa_{\mathcal{S}}$ is simply the orthogonal projection of the three-dimensional curvature vector of $\mathcal{C}$ into the tangent planes of $\mathcal{S}$. Note that this motion is the curve shortening motion which arises from applying gradient descent to minimize the length of $\mathcal{C}$ within $\mathcal{S}$.

Curvature motion on surfaces either drives the curve to collapse to a point or to a condition of zero curvature within $\mathcal{S}$. The latter condition corresponds to a geodesic curve of the surface $\mathcal{S}$. When $\mathcal{S}$ is a plane, curvature motion always has the trivial limit where $\mathcal{C}$ collapses to a point. For general surfaces, however, the curvature of $\mathcal{S}$ leads to interesting stationary limiting curves in $\mathbb{R}^3$, such as great circles on the sphere, or spirals on a cylinder. Aside from the natural geometric significance of curvature motion, this and related motions on surfaces can arise in a wide variety of situations. For example, removing fine scales from a coastline on a globe leads to curve shortening motion on a sphere, or regularizing images on pottery leads to curve shortening motions on the pottery surface. Related motions also arise in models of biological pattern formation, such as the development of animal coat patterns. For a higher dimensional example, a color image is abstractly a two-dimensional surface $\mathcal{C}$ within the three-dimensional color space $\mathcal{S}$. This is most properly modeled as a curved space, i.e. not a simple Euclidean space, and mean curvature motion of $\mathcal{C}$ in $\mathcal{S}$ is a form of intrinsic color image blurring, smoothing or data compression.

For clarity, our presentation focuses on the curvature motion of (one-dimensional) curves on (two dimensional) surfaces. Nonetheless, the methods developed immediately generalize to moving a curve $\mathcal{C}$ of dimension $N-1$ by its mean curvature $\kappa_{\mathcal{S}}$ within a curved surface $\mathcal{S}$ of dimension N.

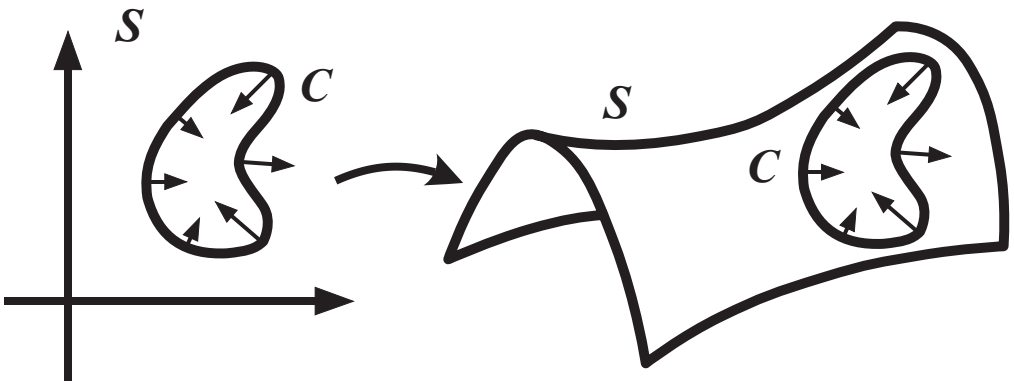

Figure 1: Motion of a curve on a surface $\mathcal{S}$, with velocity proportional to curvature. The trivial case where $\mathcal{S}$ is a plane is shown at left, while the case of interest here, where $\mathcal{S}$ is allowed to be an arbitrary surface, is indicated on the right.

Methods for treating more general curvature-dependent motion laws are also possible, as will be discussed throughout the paper.

3 Diffusion Generated Motion of Curves

In order to extend Diffusion Generated Motion to the motion of curves on surfaces, we begin with a brief review of the basic algorithm.

Refer to Figure 2. The curve to be moved, $\mathcal{C}$, is represented as the boundary of a region, $\mathcal{R}$, which in turn is represented by its characteristic function, $\chi(\mathbf{x})$,

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{R} \\ 0 & \text{otherwise.} \end{cases}$$

To move the curve in the normal direction with a speed equal to curvature, the Diffusion Generated Motion procedure alternates two steps. First, diffusion is applied to χ for a short time, Δt,

$$\chi(\mathbf{x}, \Delta t) = \mathrm{D}[\chi; \Delta t]$$

where $\mathrm{D}[\cdot; t]$ denotes the diffusive evolution operator for a time t. This is followed by "thresholding" at $\chi = \frac{1}{2}$ to obtain a new characteristic function,

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } \chi(\mathbf{x}, \Delta t) \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding region $\mathcal{R}(\Delta t) = \{\mathbf{x} \mid \chi(\mathbf{x}) = 1\}$ defines the updated boundary curve, $\mathcal{C}(\Delta t)$. Iterating this procedure (working entirely in the χ representation) defines a curve motion that is discrete in time but continuous in space. In the limit as the time step goes to zero, this converges to motion by mean curvature of the region boundary, $\mathcal{C}(t)$ [5, 1]. (If the thresholding is done at a value other than 1/2, it will produce velocity laws of the form $v = a + b\kappa$, with both a constant normal component and a curvature component.)

Note that the diffusive evolution operator, $\mathrm{D}[\chi; \Delta t]$ can be computed either by solving the linear diffusion equation

$$\frac{\partial \chi}{\partial t} = \nabla^2 \chi$$

for time Δt, or, equivalently, by convolving with the fundamental solution of the diffusion equation,

$$\chi(\mathbf{x}, \Delta t) = \chi(\mathbf{x}) * \mathrm{G}(\mathbf{x}, \Delta t)$$

where $\mathrm{G}(\mathbf{x}, t)$ is the three-dimensional Gaussian Kernel

$$\mathrm{G}(\mathbf{x}, t) = (4\pi t)^{-(3/2)} e^{-\mathbf{x}\cdot\mathbf{x}/(4t)}.$$

The latter form has the advantage that it immediately generalizes to convolution against any smoothing kernel, which can be used to generate a wide variety of geometric motion laws [8, 15].

Diffusion Generated Motion has a variety of interesting mathematical and numerical properties. Mathematically, it highlights a connection between the fundamental geometric property of curvature and the fundamental analytical process of diffusion, which may have a wider relevance beyond the algorithm. It also has connections with Phase-Field and Level Set models of interface motion (among continuum methods), and Cellular Automata and Huygens Principle models (among discrete methods) [15, 16]. From a modeling perspective, the method naturally generalizes to a broad range of curvature-dependent motion laws and to moving interfaces with triple-points or general junctions. It also has a number of desirable computational properties. For example, it handles topology changes and junctions with no special logic, it produces curvature motion without explicitly computing the curvature and it is unconditionally stable in time. The method also avoids introducing a small parameter ϵ which must be simultaneously small enough to adequately approximate the underlying limiting motion law yet large enough to yield an interface that can be spatially resolved. Finally, Diffusion-Generated Motion is simple to implement, even for complex situations such as mean curvature motion of interfaces in three-dimensions with topology changes and multiple junctions.

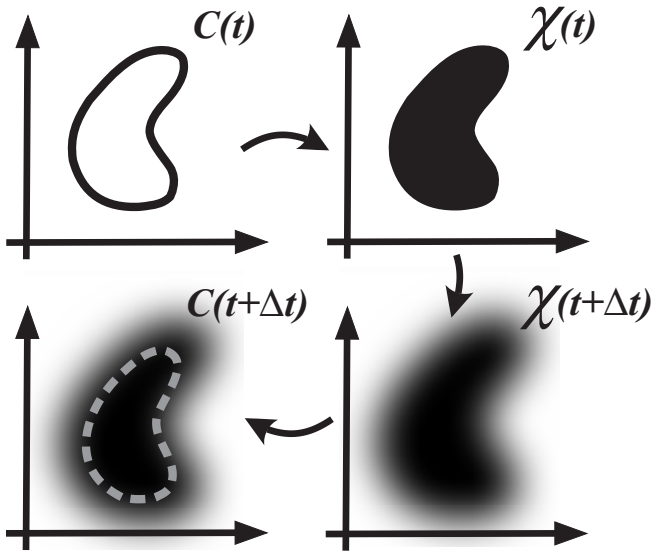

Figure 2: The major steps of the Diffusion Generated Motion Algorithm for advancing the curve by a timestep Δt: representation by a characteristic function, diffusion, and thresholding.

4 Difficulty with Intrinsic Extension to Curves on Surfaces

One approach for generalizing the standard Diffusion Generated Motion algorithm to the case of curves moving on surfaces is to replace the algorithm's diffusion step (defined in the plane) with the diffusion process that is intrinsic to the underlying surface. Such a generalization can be achieved either by using the intrinsic surface Laplacian (Laplace-Beltrami Operator) in the diffusion step, or equivalently by performing an intrinsic convolution with the fundamental solution kernel of this evolution. From a practical perspective, however, either approach can be quite complicated. Solving PDEs on surfaces generally requires writing the equations in special local coordinate systems (for example, spherical or toroidal coordinates). Several such coordinate patches will be required to cover the surface, so the simplicity of the standard diffusion equation is lost. On the other hand, a convolutional perspective does not simplify matters since the fundamental solution kernel for this process is generally not available in closed analytical form. Similar to the PDE approach, treating the convolution integral would lead to coordinate patches with appropriate Jacobian area-weighting factors.

Thus, while this direct extension would produce the desired curve motion within a surface, and extend all the interesting mathematical properties to that setting, it does not retain the attractive simple implementation of the original method. For this reason, we seek an alternative extension

with the goal of practical simplicity. In the next section, we introduce such an extension, and in a subsequent section we provide computational examples for diffusion generated curvature motions on a variety of surfaces, including surfaces with boundaries or edges. It is worth noting that while our concern here is finding a simple way to compute diffusion on surfaces, the same approach gives a simple way to evolve PDEs and carry out general evolution processes on surfaces. These intriguing possibilities will be investigated as part of future research.

5 A Simple Extension to Curves on Surfaces

To retain the simplicity of the original Diffusion Generated Motion (DGM) algorithm, our generalization to surfaces should ideally use only the standard diffusion process in the evolution step, rather than using diffusion intrinsic to the surface $\mathcal{S}$. Thus, we first embed the surface $\mathcal{S}$ in the standard three dimensional Cartesian space, $\mathbb{R}^3$. We must also "embed" the two-dimensional intrinsic diffusion process on $\mathcal{S}$ in $\mathbb{R}^3$. This can be accomplished as follows (see Figure 3): given a function χ defined on $\mathcal{S}$ that we would like to diffuse on $\mathcal{S}$ for a short time, Δt, we first extend χ to all of $\mathbb{R}^3$ by making it constant in the normal direction to $\mathcal{S}$. Then, we apply standard three-dimensional diffusion to the extended function for a short time Δt. Restricting the diffused function back to $\mathcal{S}$ gives an approximation of the intrinsically diffused form of χ, for use within the DGM algorithm. Intuitively, the fully three-dimensional diffusion is driven only by within-surface gradients, since the extended function does not vary normal to the surface. This implies that the method will in effect carry out the desired within-surface diffusion, at least in the short timescale of interest. Also, because we care only about the diffusion near $\mathcal{S}$ and for short times, details of the extension far from $\mathcal{S}$ and of the "far field" diffusion boundary conditions are not important.

Thus, we arrive at the following modified (partial) algorithm for the Diffusion Generated Motion of curves on surfaces: Let the surface $\mathcal{S}$ be embedded in $\mathbb{R}^3$ (without self-intersection) and let $\mathrm{E}[\cdot]$ denote an extension operator that extends any function defined on $\mathcal{S}$ to one defined on all of $\mathbb{R}^3$. The extension must have the property that it is is constant in the direction normal to $\mathcal{S}$ near $\mathcal{S}$ itself. To move a curve $\mathcal{C}$ on $\mathcal{S}$, let $\mathcal{C}$ be the boundary of a region $\mathcal{R}$ on $\mathcal{S}$, and let $\chi : \mathcal{S} \longrightarrow \{0, 1\}$ be the characteristic function of this region. Now, extend χ to all of $\mathbb{R}^3$,

$$\chi(\mathbf{x}) = \mathrm{E}[\chi].$$

Next, evolve this function on $\mathbb{R}^3$ by standard three-dimensional diffusion for short time Δt,

$$\chi(\mathbf{x}, \Delta t) = \mathrm{D}[\chi; \Delta t]$$

where D is the three-dimensional diffusion evolution operator, defined either by solving the diffusion equation as above, or by convolving with the corresponding three-dimensional Gaussian kernel.

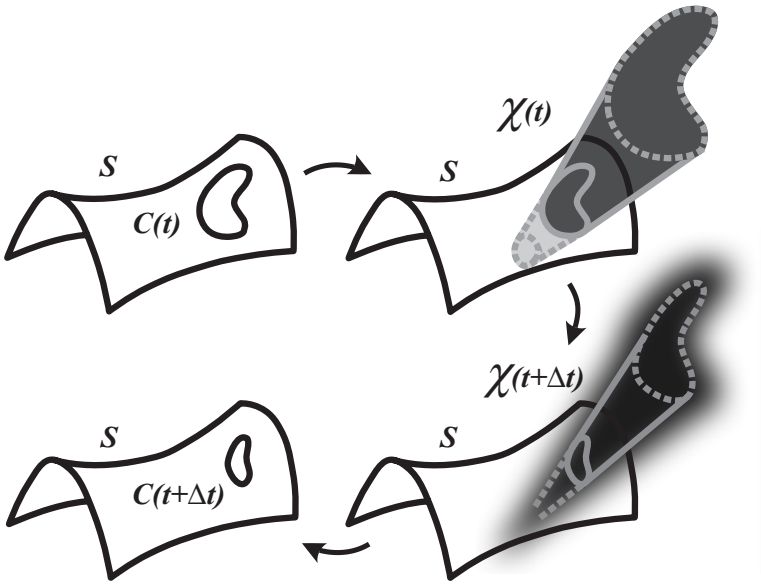

Figure 3: The major steps of Diffusion Generated Motion on a surface: representation as a three-dimensional characteristic function via closest point extension, diffusion in three dimensions, and thresholding in three dimensions.

Following this diffusion step, the updated characteristic function χ is obtained by thresholding at the 1/2 level:

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } \chi(\mathbf{x}, \Delta t) \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

From the characteristic function χ we naturally obtain the updated region $\mathcal{R}(\Delta t) = \{\mathbf{x} \in \mathcal{S} \mid \chi(\mathbf{x}) = 1\}$ and the desired boundary curve $\mathcal{C}(\Delta t)$. Iterating this procedure gives the time-discrete motion of the curve $\mathcal{C}(t)$ on $\mathcal{S}$, again in terms of the function χ defined on $\mathbb{R}^3$.

It is perhaps not immediately obvious that the threshold of 1/2 is consistent with curvature motion since the local curvature of the underlying surface $\mathcal{S}$ might come into play. Fortunately, (and as we shall show in Section 6) the threshold value of 1/2 is still valid for a *closest point extension*.

5.1 The Closest Point Extension

Ultimately, the simplicity of the above procedure depends on the simplicity of the extension process E[·] which will, in turn, depend on the representation of the surface $\mathcal{S}$. An especially simple

extension operation is obtained if the "closest point" representation is chosen.

The closest point function, denoted $\mathrm{CP}(\mathbf{x})$, is a vector valued function that returns the closest point on $\mathcal{S}$ to $\mathbf{x}$ for each point $\mathbf{x} \in \mathbb{R}^3$; see Figure 4. A clear connection exists between the closest point function and the signed distance function to $\mathcal{S}$, $d[\mathbf{x}; \mathcal{S}]$. Specifically, the signed distance function gives the distance from $\mathbf{x}$ to its closest point in $\mathcal{S}$, with the direction given by the gradient of $d[\mathbf{x}; \mathcal{S}]$, ie,

$$\mathrm{CP}(\mathbf{x}) = \mathbf{x} - d\nabla d.$$

Despite this connection, the closest point representation of a stationary surface is in several respects more general than a signed distance function representation. For example, closest point representations naturally represent open surfaces, surfaces with boundaries as well as surfaces without orientation (such as a Mobius strip). Closest point representations also trivially handle objects of arbitrary codimension and collections of objects of varying codimension.

For many interesting shapes, such as the sphere and torus, the closest point function can be determined analytically. More generally, it can be tabulated on a grid and interpolated as needed. In particular, if the surface $\mathcal{S}$ itself is defined explicitly by a triangulation, a tabulation of $\mathrm{CP}(\mathbf{x})$ on a grid in $\mathbb{R}^3$ can easily be generated from the triangulation.

We now return to the issue of how to choose the extension step. Suppose we are given the closest point function for $\mathcal{S}$. Then the extension of any function f defined on $\mathcal{S}$ to a point $\mathbf{x} \in \mathbb{R}^3$ is simply f evaluated at the closest point to $\mathbf{x}$,

$$\mathrm{E}[f](\mathbf{x}) = f(\mathrm{CP}(\mathbf{x})).$$

Applying this simple and computationally efficient extension to our generalization of Diffusion Generated Motion gives the final form of the algorithm: Given the characteristic function on $\mathcal{S}$ that corresponds to the initial curve, initialize the three-dimensional characteristic function as $\chi(\mathbf{x}) = \chi(\mathrm{CP}(\mathbf{x}))$. Thereafter, alternate diffusion and thresholding steps according to

$$\chi(\mathbf{x}, \Delta t) = \mathrm{D}[\chi; \Delta t];$$

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } \chi(\mathrm{CP}(\mathbf{x}), \Delta t) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathrm{D}[\cdot; \Delta t]$ is evolution by solving the diffusion equation or convolving with the Gaussian kernel in three dimensions.

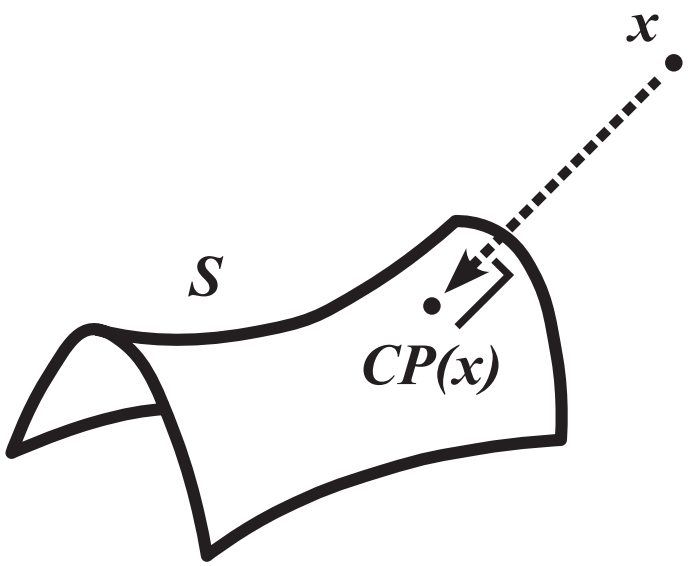

Figure 4: The closest point function for a surface $\mathcal{S}$.

5.2 A Note on Generality

This final method can formally be applied to any surface $\mathcal{S}$ that is represented by its closest point function. This includes surfaces with boundaries–i.e. surfaces that are not closed, like one hemisphere of a sphere–as well as surfaces with edges, such as a cube. Even surfaces that are entirely nonsmooth, such as fractals, can in principle be treated, since the closest point function always exists. Note, however, that the intuitive motivation underlying the method breaks down at surface boundaries, edges or any non-smooth points where the normal direction is not well defined. For example, at edges and non-smooth points the extended function exhibits some form of variation in the direction of the normal at the surface itself. These gradients in the extended function will interact with the surface diffusion evolution. The net effect is that some form of "boundary condition" or auxiliary forcing is implicitly applied to the curve where it contacts such surface features. How this influences the curve evolution is not immediately clear, in contrast to the curve evolution on the "smooth interior" of $\mathcal{S}$ which is always motion by curvature.

Empirically, at surfaces with boundaries or edges, the algorithm appears to impose the condition that the curve crosses any singular curves on $\mathcal{S}$ orthogonally. See Section 8.4 for an example. We conjecture that it may be possible to control this boundary/edge condition by modifying the normal extension in regions of $\mathbb{R}^3$ where the normal direction at the closest point is undefined. The relation between such altered extensions and the effective boundary conditions at singular points on $\mathcal{S}$ remains unexplored, and is not intuitively clear. One particularly desirable modification would be

one that imposes some form of geodesic-compatible crossing condition, so that moving curves can relax by curvature motion to global geodesics even on surfaces with edges, such as polyhedra.

6 Method Justification

It seems likely that both the asymptotics and the rigorous convergence theory that support the original Diffusion Generated Motion algorithm could be extended to our generalization for curvature motion on surfaces. A study of this type might set a target of showing that the method does generate an initial curve velocity equal to its curvature within $\mathcal{S}$, and that the timestepping procedure converges to full motion curvature motion. However, a more elegant justification is available by appealing to the standard DGM method in higher dimensions, as indicated in Figure 5.

Note that if the original DGM algorithm is applied to a surface in three dimensions–instead of a curve in two-dimensions–the resulting diffusion generated motion of the surface is precisely motion by *mean curvature*, $v = H = \kappa_1 + \kappa_2$, where κ_i are the principal curvatures of the surface [5, 1]. In the present context, we apply the method to the surface which is defined by the extended characteristic function $\mathrm{E}[\chi]$ and let the corresponding 0/1 interface surface be ψ. In the new method, we apply diffusion to this characteristic function in $\mathbb{R}^3$, and by the properties of the original method this displaces the surface ψ in $\mathbb{R}^3$ by its local mean curvature. Clearly, the intersection of this surface ψ with the substrate surface $\mathcal{S}$ is exactly the curve of interest, $\mathcal{C}$. It is also clear that the curvature of ψ normal to $\mathcal{S}$ is 0 by construction, which indicates that the maximum principal curvature of ψ is in the direction tangent to $\mathcal{C}$ in $\mathcal{S}$, and is exactly the curvature of $\mathcal{C}$ within $\mathcal{S}$. Taken together, we find that the mean curvature of ψ at $\mathcal{C}$ in $\mathcal{S}$ is exactly the curvature of $\mathcal{C}$ within $\mathcal{S}$. Thus, the standard DGM motion of ψ induces exactly the desired motion of $\mathcal{C}$ within $\mathcal{S}$ in the new method.

The above argument applies as long as we are in a neighborhood where $\mathcal{C}$ and $\mathcal{S}$ are locally smooth. At points where $\mathcal{C}$ or $\mathcal{S}$ are not smooth, there is no immediate justification. Nonsmooth points in $\mathcal{C}$ will generally be instantaneously smoothed away by the motion, and will thus subsequently move by curvature. In contrast, nonsmooth points of $\mathcal{S}$, such as boundaries, edges or corners, remain uncharacterized and the motion of the curve $\mathcal{C}$ at such points will generally not be purely by mean curvature.

7 Moving Curves with Junctions

As in the original Diffusion Generated Motion, there is an elegant extension to moving curves with triple-point junctions or general self-intersections. For completeness, we will briefly describe this extension for motion on surfaces, which is illustrated in Figure 6.

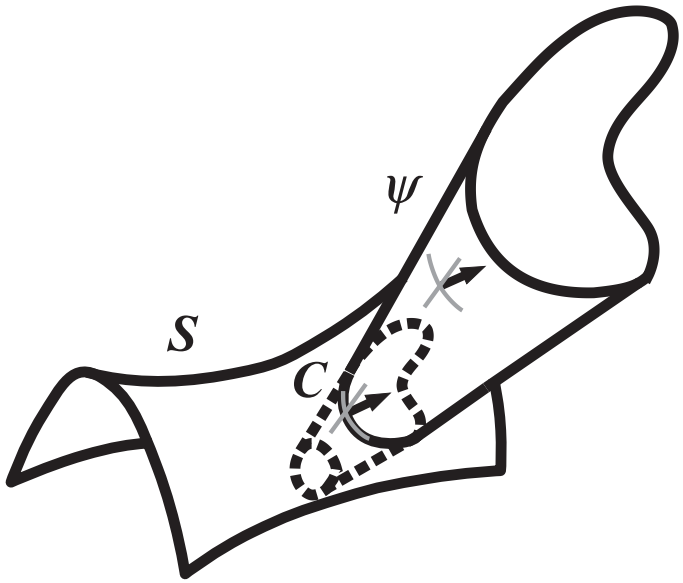

Figure 5: The general properties of Diffusion Generated Motion in higher dimensions induce the cylindrical surface ψ to move by mean curvature, which reduces to curvature motion of $\mathcal{C}$ within $\mathcal{S}$ due to the geometry of the construction.

Whereas a single curve, $\mathcal{C}$, divides the surface, $\mathcal{S}$, into two components, the region $\mathcal{R}$ and its complement, a curve with multiple junctions partitions the surface into multiple regions, $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$. We extend the method by representing each such region $\mathcal{R}_i$ by a characteristic function χ_i on $\mathcal{S}$. Then, each χ_i undergoes *independent* closest point extension to $\mathbb{R}^3$,

$$\chi_i(\mathbf{x}) = \chi_i(\mathrm{CP}(\mathbf{x})),$$

and then *independent* diffusion evolution,

$$\chi_i(\mathbf{x}, \Delta t) = \mathrm{D}[\chi_i; \Delta t].$$

The thresholding step which repartitions these "fuzzy" sets into a new partition of $\mathcal{S}$ is done by simply classifying each point $\mathbf{x}$ according to the largest of the evolved χ_i, or, in terms of the characteristic functions themselves,

$$\chi_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \chi_i(\mathbf{x}, \Delta t) \geq \chi_j(\mathbf{x}, \Delta t) \text{ for all } j = 1, 2, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

This defines the single timestep, and the motion proceeds by iteration.

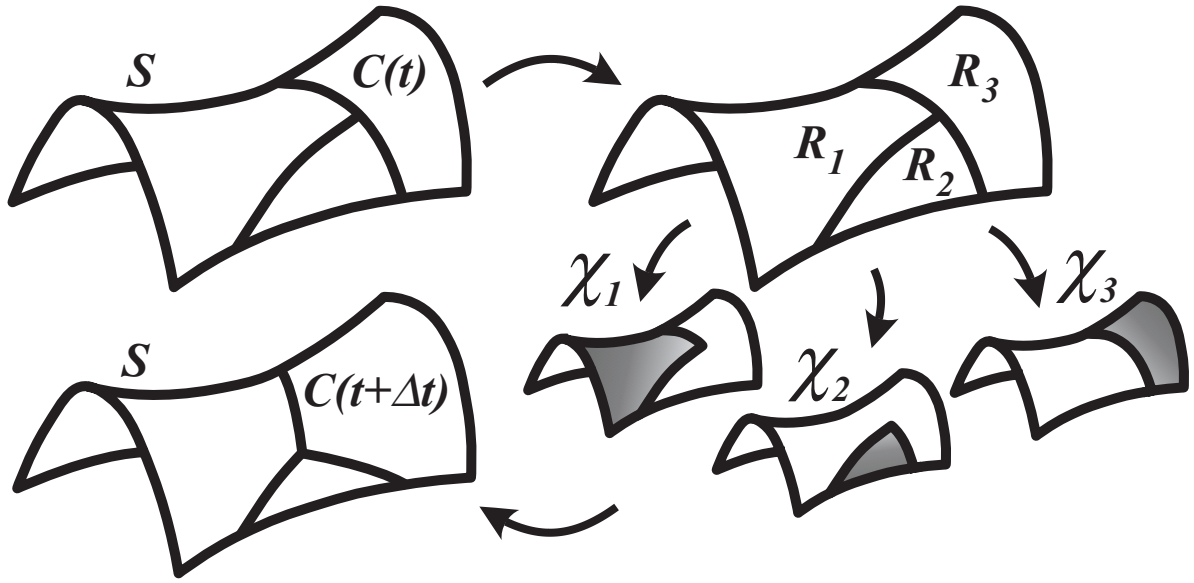

Figure 6: Extension of the procedure to a curve $\mathcal{C}$ with junction points. Each region defined by $\mathcal{C}$ is independently diffused and then the results are coupled by a joint thresholding procedure.

This extension has the property that at the curve $\mathcal{C}$ away from junctions, it reduces to the basic motion by curvature method, and at junctions, breaks up any higher order junctions into triple points, where it imposes the symmetrical equal angle condition. Note that the complex junction motion is in effect decomposed into n totally independent linear problems, with a single thresholding step supplying the complex nonlinear interaction.

Similar to the planar case, any desired angle conditions can be imposed at triple point junctions by replacing the symmetrical thresholding step with one that is nonsymmetrical. The thresholding step for nonsymmetrical junctions does not have a known analytical expression, but can be constructed numerically in a straightforward manner. See [13] for details on these thresholding methods and their construction. Also, note that the χ_i form a partition of unity on $\mathcal{S}$ initially, $\sum_{j=1}^{n} \chi_j = 1$, and this condition persists during the evolution, since it is preserved by the linear diffusion or convolution with a normalized kernel. This is a statement that no "vacuum regions" (or "curve fattening") can open up at the junctions, whereas such pathologies can occur in certain level-set formulations of junction motion.

8 Numerical Experiments

We now apply our methods to some illustrative examples of curvature motion on surfaces, including some elementary convergence tests and the evolution of junctions and pattern formation models. Following [14], FFTs on unequally spaced grids are used to carry out the spatial discretization. These methods lead to results which are essentially free of spatial discretization error, allowing us to better focus on the behavior of the underlying semi-discrete method.

8.1 Curvature Motion on the Sphere

Consider first the motion of a circular interface on a sphere evolving according to curvature motion. By symmetry, the interface remains a circle as it collapses. Moreover, it is straightforward to determine the state of the system at any time t since the radius of the collapsing circle is governed by an ODE system which can be solved to high precision with standard ODE methods.

We take the sphere radius to be 0.3 and the initial radius of the circle to be 0.2, as shown in Figure 7. Convergence is studied by comparing the numerical radius at time $t = 0.02$ against the exact result (0.10963040395421) for various Δt. The absolute errors in the final radius from a number of experiments are reported in Table I, below.

Δt	*Error*	*Conv. Rate*
0.04	0.03871	
0.02	0.01798	1.11
0.01	0.00615	1.55
0.005	0.00238	1.37
0.0025	0.00107	1.15

Table I. Errors for an evolving circle on a sphere.

These results are suggestive of an approximately first order error in the position of the front.

8.2 Curvature Motion on an Ellipsoid

Our second test considers the evolution of a smooth curve on the ellipsoid

$$\frac{(x-\frac{1}{2})^2}{(0.4)^2}+\frac{(y-\frac{1}{2})^2}{(0.3)^2}+\frac{(z-\frac{1}{2})^2}{(0.25)^2}=1.$$

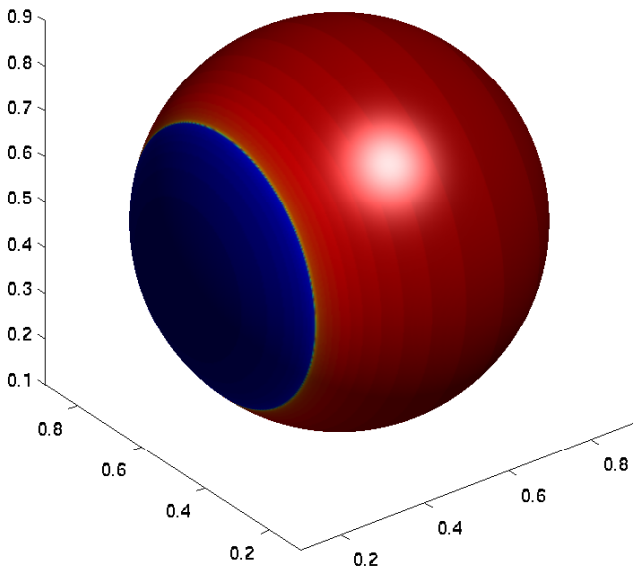

Figure 7: Initial conditions for the sphere. As shown in Table I, applying Diffusion Generated Motion to this problem moves the boundary between the regions according to curvature motion.

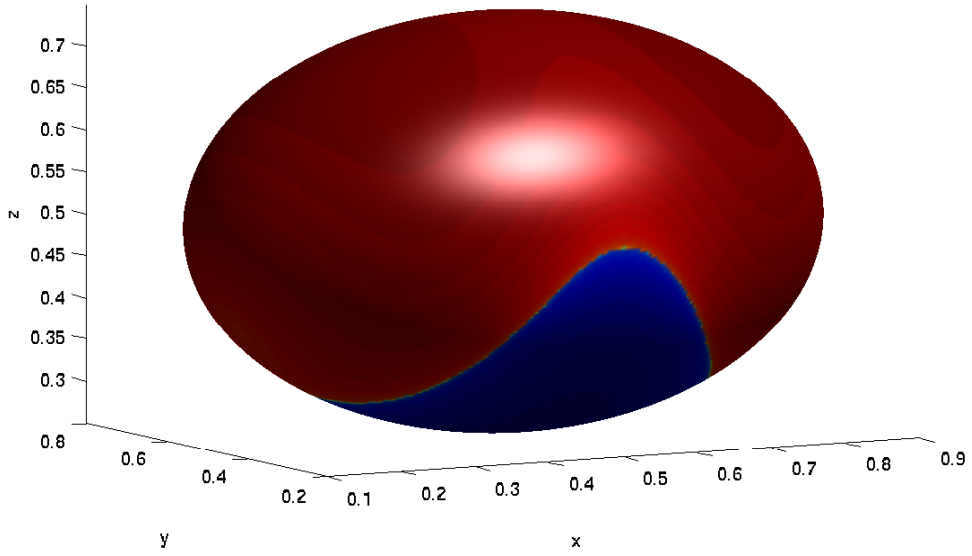

Figure 8: Initial conditions for the ellipsoid. As shown in Table II, applying Diffusion Generated Motion to this problem moves the boundary between the regions according to curvature motion.

The initial curve is taken to be the intersection of the ellipsoid with the surface

$$z = 0.1 \cos\left(4\pi\left(x - \frac{1}{2}\right)\right) + 0.4$$

as displayed in Figure 8. The closest point representation for this shape was obtained numerically on a $101 \times 101 \times 101$ grid using Matlab's Optimization Toolkit.

In this case the exact motion cannot be determined by solving a scalar ODE. Instead we study convergence by comparing the initial velocity produced by diffusion-generated motion against the true value at the point $p_0 = (0, 0.3, 0)$. In this case the true initial velocity (15.791368) is determined by multiplying a numerical approximation of the curvature at p_0 by the cosine of the angle between the principal normal to the curve and the tangent plane to the surface at p_0. The relative errors in the initial velocity from a number of experiments are reported in Table II, below.

Δt	*Error*	*Conv. Rate*
0.002	0.2584	
0.001	0.1560	0.73
0.0005	0.0924	0.76
0.00025	0.0504	0.88
0.000125	0.0256	0.98

Table II. Errors for an evolving curve on an ellipsoid.

These results are suggestive of an approximately first order error in the position of the front.

8.3 Diffusion Generated Motion for Multiple Regions

The evolution of symmetric junctions corresponding to n regions may be carried out using the algorithm for multiple junctions described in Section 7.

We illustrate the behavior of the algorithm by evolving five (randomly generated) initial regions on a torus. These initial regions form a variety of junctions as displayed in the upper left-hand corner of Figure 9. As shown in Figures 9 and 10 the evolution leads to several mergers, and an overall simplification of the system. In particular, note that stable, symmetric triple junctions are observed throughout the simulation and that no special treatment is required for the topological shape changes that arise as time advances.

A second example involving complex data on the sphere is given in Figure 11. Here the various regions represent countries or ocean, with the underlying country boundaries taken according to the WorldData.m package in Mathematica 5.2. This defines the boundaries of 174 countries as polygons in angular coordinates, using a total of 3074 nodes. In this example the process of alternating a Gaussian convolution with a symmetric thresholding step might be viewed as applying an *iterated median filter* to the initial country data. Iterated median filters and related iterated linear filters are processes that can be used to smooth images before extracting information or features. See, e.g., [6] for details.

8.4 Diffusion Generated Motion for a Surface with Edges

Part of the accuracy criteria for diffusion-generated motion on surfaces is that the effective support of the heat kernel must be small enough to resolve the basic curvature of the underlying surface, i.e.

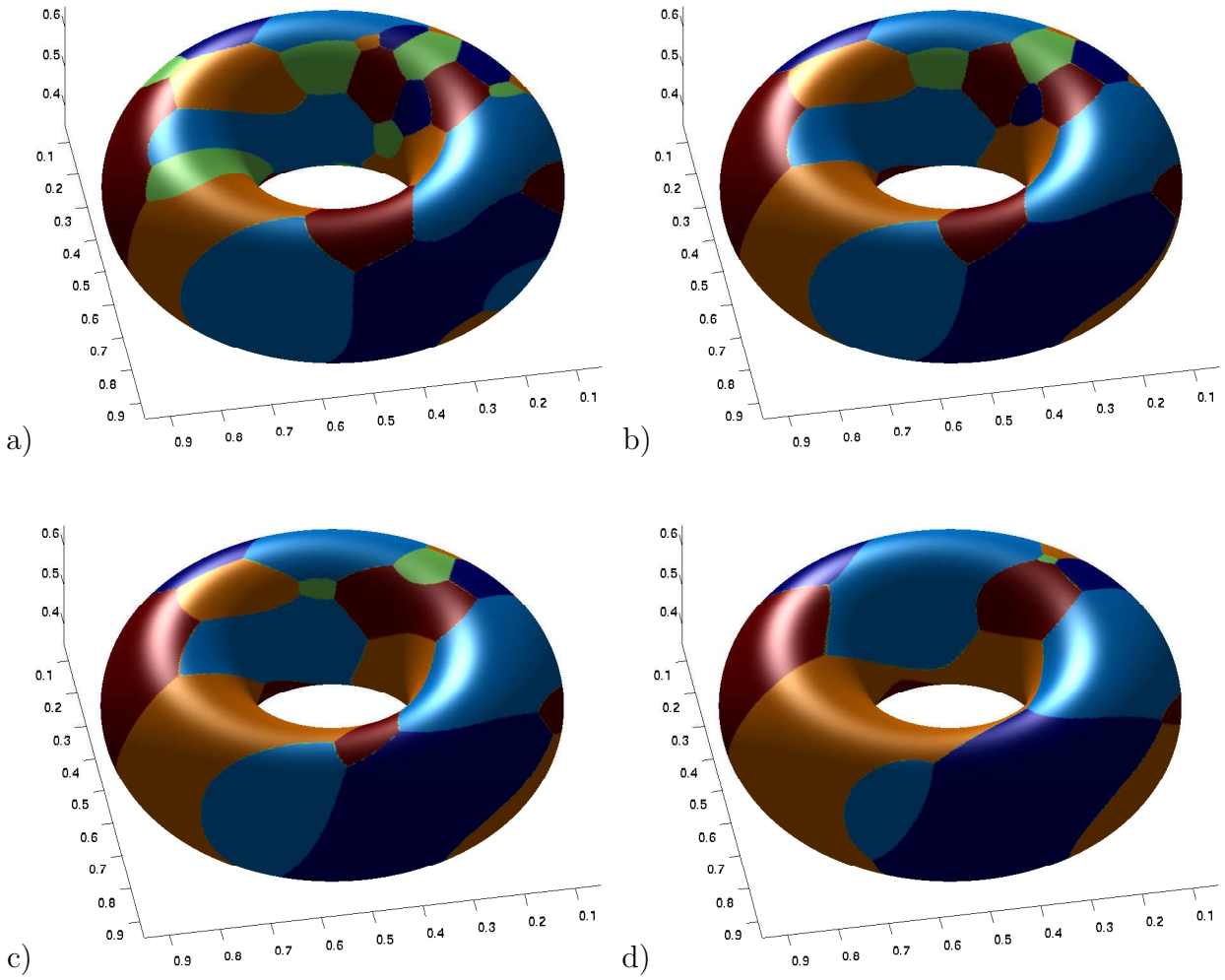

Figure 9: Junctions evolving on a torus, displayed at times a) 0.000, b) 0.001, c) 0.002, and d) 0.004. This simulation takes $\Delta t = 0.0001$ and uses an analytical representation of the closest point to the torus.

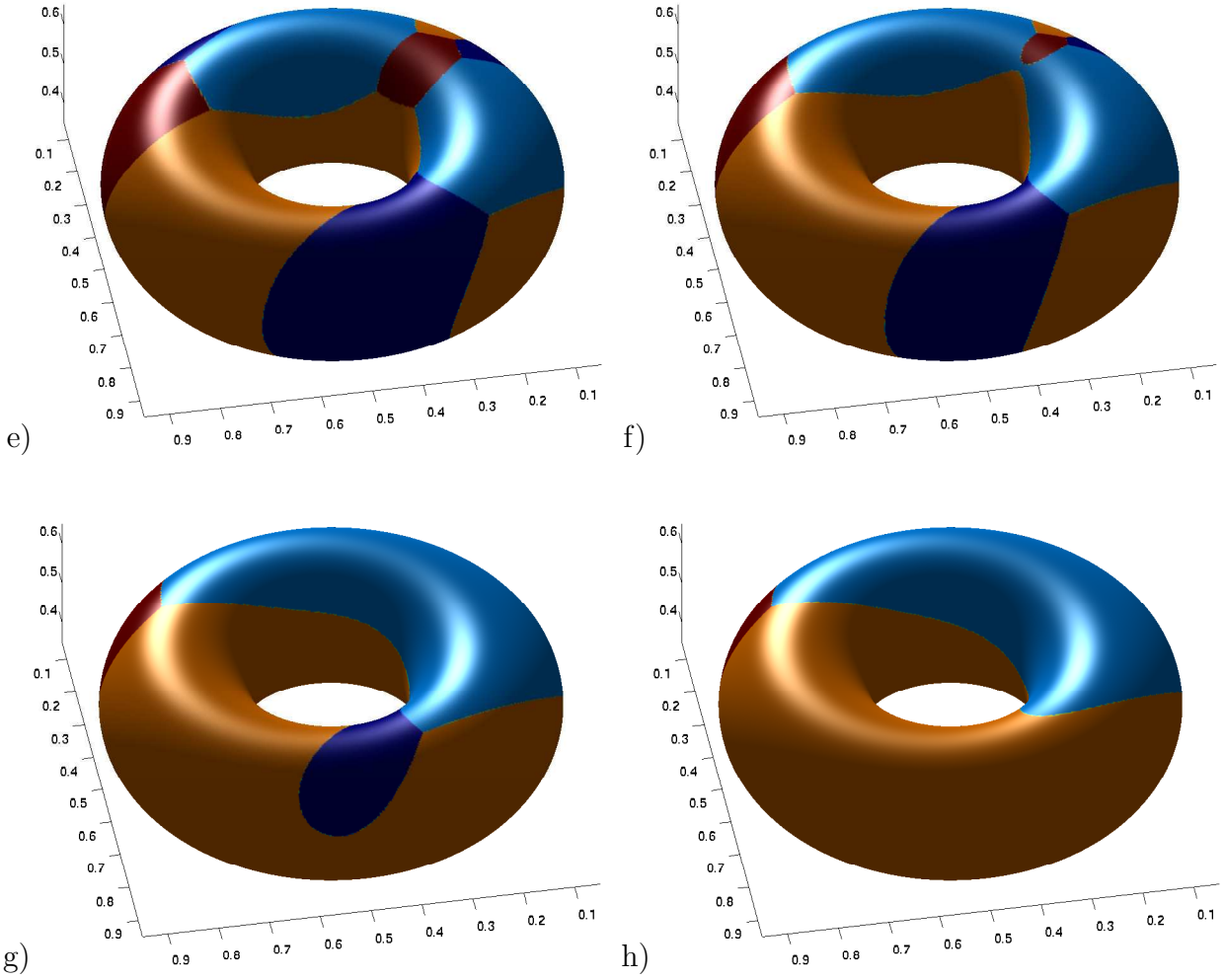

Figure 10: Junctions evolving on a torus (cont), displayed at times e) 0.008, f) 0.014, g) 0.019, h) 0.022.

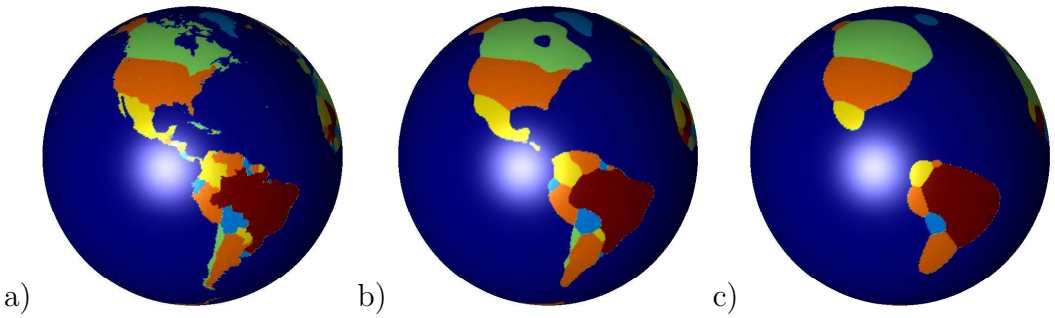

Figure 11: Curvature evolution of the Americas at times a) 0, b) 0.00004 and c) 0.0004 using a step size of 0.00004. Note that the Matlab visualization functions result in some spurious color mixing and minor visual artifacts at country boundaries.

the radius of curvature in each of the principal directions must be much greater than the effective kernel width, $\sqrt{\Delta t}$. This implies that at an edge, where one of the principal curvatures is zero, the motion is not expected to give the geodesic crossing condition.

To investigate the algorithm's behavior near edges, we consider the evolution of an initially straight interface on a box, as displayed in Figure 12. We find that the interface immediately evolves to intersect edges at right angles and that this angle condition at edges robustly persists throughout the calculation.

The source of this edge condition comes from the closest point extension, $\mathrm{E}[\cdot]$. This extension biases the convolutional average near an edge towards the value at the edge, leading to a right angle condition. We conjecture that an interesting variety of edge angle conditions can be set up by appropriately modifying the extension process for edges and corners and plan to investigate such extensions as part of future work.

8.5 A Pattern Formation Model

Young's model for vertebrate skin patterns [18] assumes that cells are in one of two states — differentiated (colored) and undifferentiated. Each differentiated cell produces two diffusive chemicals: a short range "activator" and a longer range "inhibitor". The activator stimulates the differentiation of nearby undifferentiated cells and the inhibitor stimulates nearby differentiated cells to become undifferentiated. The combined effect of these two chemicals is modeled as the weighted difference of concentrations.

Written in convolutional form (cf. [16]), Young's model in the plane sets $\chi_{\mathcal{R}} : \mathbb{R}^2 \to \{0, 1\}$ equal to the characteristic function for the differentiated region, $\mathcal{R}$. The differentiated region evolves

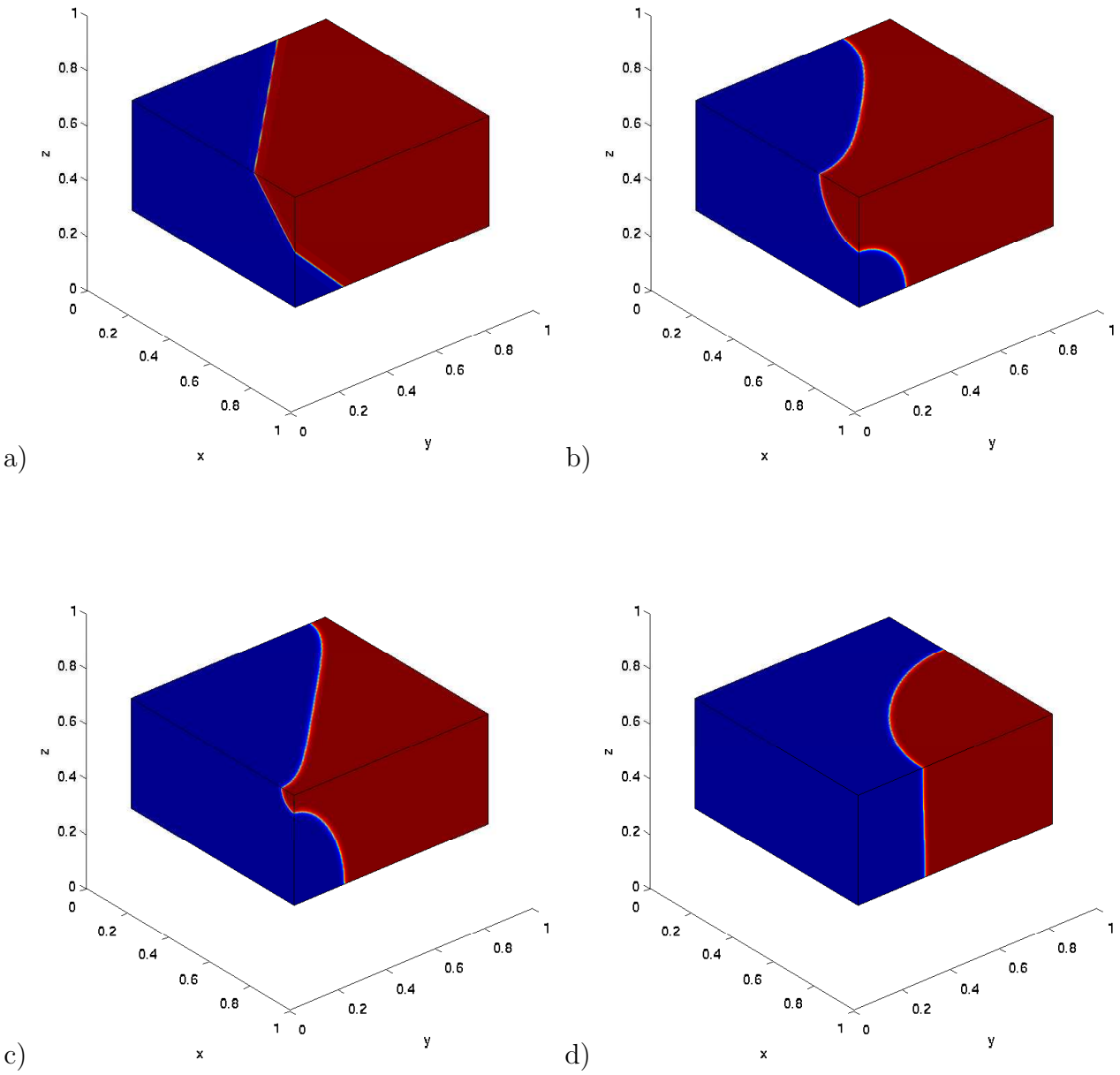

Figure 12: Curve evolving on a box, displayed at times a) 0.00, b) 0.02, c) 0.12, and d) 0.16. The box is defined by the corners $(0.5, 0.5, 0.5) + (0.4i, 0.4j, 0.2k), i, j, k = -1, +1$ and the initial interface is the intersection of the box with the plane $0 = x + y + z - 3/2$. This simulation takes $\Delta t = 0.0001$ and uses an analytical representation of the closest point to the box. Notice that the algorithm gives right angles at the edges.

according to the iteration

$$\mathcal{R}_{i+1} = \{\mathbf{x} : \chi_{\mathcal{R}_i} * K(\mathbf{x}) > 0\}$$

until a steady pattern is obtained. Young models the combined effects of activator and inhibitor using a convolution kernel K that contains both positive and negative parts.

The extension of Young's model from the plane (e.g., [18, 16]) to general surfaces follows in a similar fashion to the extension of diffusion-generated motion:

1. Values on the surface are extended throughout space using the closest point extension, $\mathrm{E}[\cdot]$.

2. The extended function is thresholded at the zero level to give a characteristic function defined over $\mathbb{R}^3$. Specifically, non-negative values of the extended function are set to 1 and negative values are set to 0.

3. This new characteristic function is convolved with the kernel K.

This completes one iteration, and the process is repeated until steady patterns are obtained. Similar to [16] we take $K(\mathbf{x})$ to be the difference of two heat kernels, although more general kernels (e.g., anisotropic kernels for anisotropic patterns) are also easily accommodated.

Examples of steady patterns on a torus and a Mobius strip appear in Figure 13. In the simulation on the torus, the closest point representation was provided analytically. For the Mobius strip, the closest point representation was obtained numerically on a $101 \times 101 \times 101$ grid. Similar to the case of the ellipsoid, this task was accomplished by finding the point which minimizes distance to the two-parameter representation of the Mobius strip

$$\begin{aligned}
x(u,v) &= \frac{1}{4} \cdot \left(1 + \frac{v}{2}\cos\left(\frac{u}{2}\right)\right)\cos(u) + \frac{1}{2}, \\
y(u,v) &= \frac{1}{4} \cdot \left(1 + \frac{v}{2}\cos\left(\frac{u}{2}\right)\right)\sin(u) + \frac{1}{2}, \\
z(u,v) &= \frac{1}{4} \cdot \frac{v}{2}\sin\left(\frac{u}{2}\right) + \frac{1}{2}
\end{aligned}$$

where $0 \leq u \leq 2\pi$ and $-1 \leq v \leq 1$. The second example is particularly noteworthy because it illustrates that a closest point representation of the underlying surface avoids any requirements for the surface to be closed or even be orientable, in contrast with standard level set representations of surfaces.

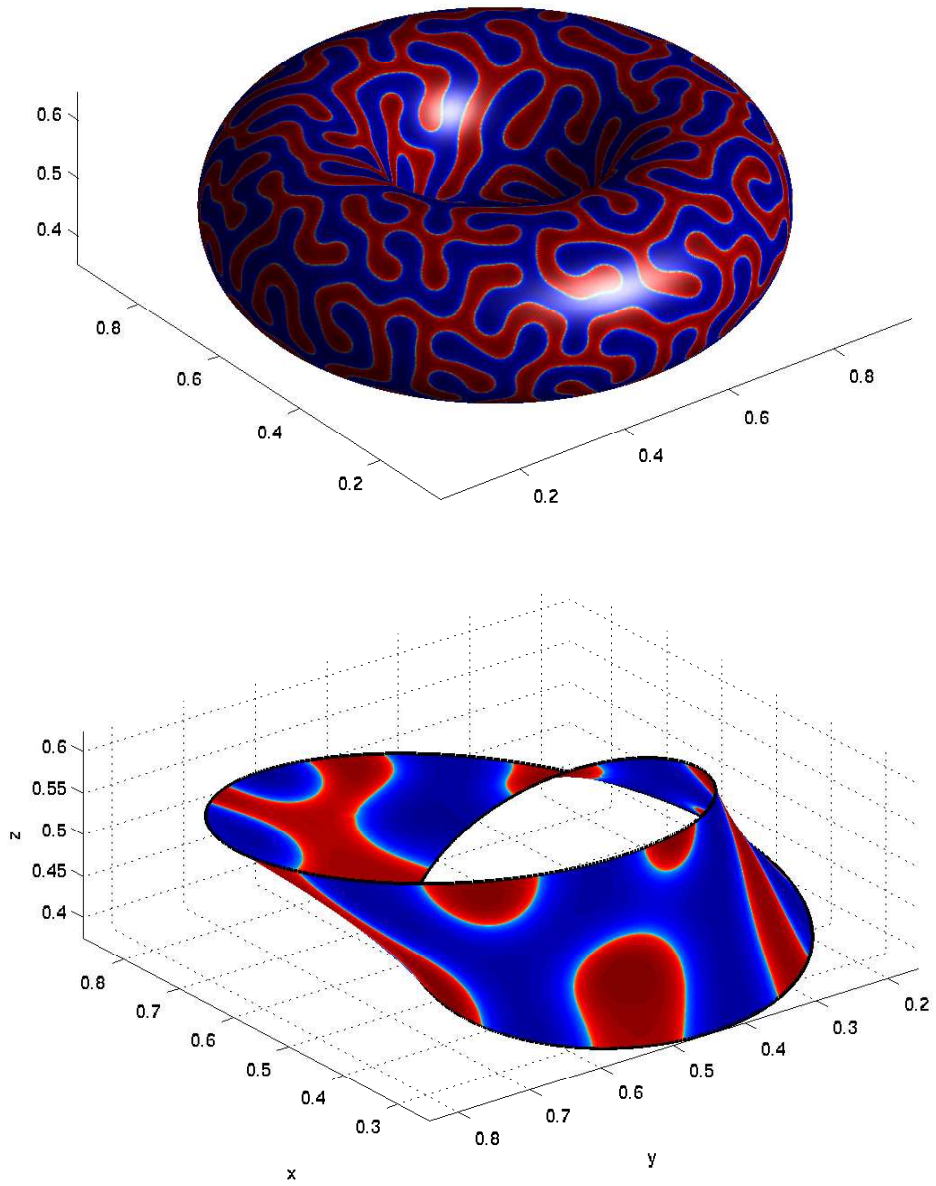

Figure 13: Steady patterns after 100 iterations on a torus (top) and a Mobius strip (bottom). Both cases choose a kernel of the form $K(\mathbf{x}) = \mathrm{G}(\mathbf{x}, \delta/6) - \mathrm{G}(\mathbf{x}, \delta)$ where $\mathrm{G}(\mathbf{x}, \delta)$ is the Gaussian heat kernel with a time-scale parameter δ. The torus simulation sets $\delta = 0.00025$ and the Mobius strip sets $\delta = 0.000025$.

9 Conclusions and Future Research

We have extended the Diffusion Generated Motion algorithm to the case of curves moving on surfaces, while retaining the simplicity of the original method. Key aspects to our approach are the use of standard diffusion in one dimension higher than the underlying surface and the use of the closest point representation of the surface. By choosing the closest point representation we achieve maximal algorithmic efficiency in the extension step as well as the flexibility to treat open surfaces, surfaces without orientations, objects of codimension two or higher or collections of objects with varying codimension. The resulting method inherits all the favorable properties of the original diffusion generated motion algorithm, and most notably the extreme simplicity. Our examples show that complex curve motions on surfaces, including merger, pinch-off, multiple junctions and the relaxation to complex geodesics, can all be treated by the algorithm. The examples also indicate the extension to other motion laws, such as those for modeling pattern formation, can be achieved through minor modifications of the convolution process.

The one practical limitation is that this formulation nominally requires computing diffusion in *two spatial dimensions* higher than the original problem itself—that is, a moving curve is inherently a one spatial dimension problem, whereas we perform the diffusion in three dimensions. However, in cases where computational efficiency becomes critical–such as higher dimensional surfaces–the diffusion calculations can be restricted to a neighborhood of the surface $\mathcal{S}$, or even just the vicinity of the curve $\mathcal{C}$. This ensures that the effective dimension of the computational problem approximates that of the intrinsic problem, although development of specific efficient algorithms remains to be carried out.

Certain issues are unique to this new setting, and require further exploration rather than mere generalization of the original properties of Diffusion Generated Motion. Most importantly, it is not clear how to impose boundary conditions on the moving curve $\mathcal{C}$ at boundaries or edges in the underlying surface, $\mathcal{S}$. Of particular interest are conditions that will allow all geodesics to be stationary solutions. It also remains to develop a suitable extension to surfaces that cannot be isometrically embedded in $\mathbb{R}^3$ without self-intersection, such as the classic Klein Bottle. However, in such cases it appears that precisely the same algorithm could be applied within a four-dimensional (or higher) Euclidean space in which surface is embedded without intersection. Another unresolved problem is to characterize the motions laws that arise when linear diffusion is replaced by convolution with a general smoothing kernel on $\mathbb{R}^3$.

Finally, note that our presentation has emphasized that the curvature motion of curves on surfaces can be derived via standard three-dimensional diffusion. Alternatively, we can interpret three-dimensional diffusion coupled with the closest point extension procedure as an *extremely simple* way to carry out the intrinsic surface diffusion process that was required for the naive generalization of DGM. From this standpoint, we see that this paper also provides a simple method for computing surface diffusion. We anticipate that many other evolution processes on surfaces

can be treated in a similar manner and will explore this novel direction for generalization in future research.

10 Acknowledgements

We thank Selim Esedoğlu and Richard Tsai for suggesting the topic of generalizing diffusion generated motion to surface flows.

References

[1] G. Barles and C. Georgelin. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal on Numerical Analysis*, 32(2):484–500, 1995.

[2] K.A. Brakke. The surface evolver. *Experimental Mathematics*, 1(2):141–165, 1992.

[3] L.-T. Cheng, P. Burchard, B. Merriman, and S. Osher. Motion of curves constrained on surfaces using a level-set approach. *J. Comput. Phys.*, 175(2):602–644, 2002.

[4] S. Esedoglu and Y.H.R. Tsai. Threshold dynamics for the piecewise constant Mumford-Shah functional. *J. Comput. Phys.*, 211(1):367–384, 2006.

[5] L.C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana University Mathematics Journal*, 42:553–557, 1993.

[6] F. Guichard, J.-M. Morel, and R. Ryan. *Contrast invariant image analysis and PDE's*. preprint. Web Address: www.cmla.ens-cachan.fr/Utilisateurs/morel/JMMBookOct04.pdf.

[7] H. Ishii. A generalization of the Bence, Merriman and Osher algorithm for motion by mean curvature. In A. Damlamian, J. Spruck, and A. Visintin, editors, *Curvature Flows and Related Topics*, pages 111–127. Gakkôtosho, Tokyo, 1995.

[8] H. Ishii, G.E. Pires, and P.E. Souganidis. Threshold dynamics type schemes for propagating fronts. *TMU Mathematics Preprint Series*, 4, 1996.

[9] B. Jawerth and P. Lin. Shape recovery by diffusion generated motion. *J. of Visual Communication and Image Representation*, 13(1-2):94–102, 2002.

[10] P. Mascarenhas. Diffusion generated motion by mean curvature. CAM Report 92-33, University of California, Dept. of Math, Los Angeles, 1992.

[11] B. Merriman, J. Bence, and S. Osher. Diffusion generated motion by mean curvature. In J.E. Taylor, editor, *Computational Crystal Growers Workshop*, pages 73–83. American Mathematical Society, Providence, Rhode Island, 1992. Also available as UCLA CAM Report 92-18, April 1992.

[12] B. Merriman, J. Bence, and S. Osher. Motion of multiple junctions: a level set approach. *J. Comput. Phys.*, 112(2):334–363, 1994.

[13] S.J. Ruuth. A diffusion-generated approach to multiphase motion. *J. Comput. Phys.*, 145:166–192, 1998.

[14] S.J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. *J. Comput. Phys.*, 144:603–625, 1998.

[15] S.J. Ruuth and B. Merriman. Convolution-generated motion and generalized Huygens' principles for interface motion. *SIAM Journal on Applied Mathematics*, 60(3):868–890, 2000.

[16] S.J. Ruuth, B. Merriman, and S. Osher. Convolution generated motion as a link between cellular automata and continuum pattern dynamics. *J. Comput. Phys.*, 151:836–861, 1999.

[17] S.J. Ruuth and B.T.R Wetton. A simple scheme for volume-preserving motion by mean curvature. *J. Scientific Computation*, 19(1):373–384, 2003.

[18] D.A. Young. A local activator-inhibitor model of vertebrate skin patterns. *Mathematical Biosciences*, 72:51–58, 1984.