Convolution Generated Motion and Generalized Huygens' Principles for Interface Motion

Steven J. Ruuth* and Barry Merriman†

February 5, 1998

Abstract

A physical interface can often be modeled as a surface that moves with a velocity determined by the local geometry. Accordingly, there is great interest in algorithms that generate such geometric interface motion. In this paper we unify and generalize two simple algorithms for constant and mean curvature based interface motion: the classical Huygens' principle, and diffusion-generated motion. We show that resulting generalization can be viewed both geometrically as a type of Huygens' principle, and algebraically as a convolution generated motion. Using the geometric-algebraic duality from the unification, we construct specific convolution generated motion algorithms for a common class of anisotropic, curvature-dependent motion laws. We validate these algorithms with numerical experiments, and show that they can be implemented accurately and efficiently with adaptive resolution and fast Fourier transform techniques.

*Department of Mathematics, University of California at Los Angeles. (ruuth@math.ucla.edu).
The work of this author was partially supported by an NSERC Postdoctoral Scholarship, NSF DMS94-04942 and ONR NOOO14-97-1-0027.

†Department of Mathematics, University of California at Los Angeles. (barry@math.ucla.edu).
The work of this author was partially supported by NSF DMS94-04942, ONR NOOO14-92-J-1890.

1 Introduction

There are many natural phenomena in which fairly sharp interfaces form and propagate. Notable examples include the growth of crystalline materials, the evolution of detonation fronts in explosive materials, and the waves of excitation that occur in heart and neural tissue.

Asymptotic models for these processes often yield equations of motion for a surface moving with a normal velocity that is a function of the local surface geometry, i.e. a function of the local normal direction, curvature, and higher space and time derivatives of these quantities. For example, a variety of such models are discussed in [28, 18], and a particularly accurate family of geometric models for detonation-shock front dynamics is described in [29].

Given such models, it becomes important to develop algorithms which can realize geometric surface motions in simple, efficient and accurate ways, and which are amenable to mathematical analysis. Designing suitable algorithms is complicated by the fact that in many problems the interfaces can merge or break up, form triple point junctions or more complicated interface networks. It is particularly challenging to find algorithms that retain their simplicity, yet are robust enough to capture these topological features.

In this present work, we outline a simple and robust method that can achieve a wide variety of geometric surface motions. We start from two existing algorithms for the special cases of constant and mean curvature motion. We then unify and generalize these in order to achieve a larger class of velocity laws while retaining the strengths of the original methods.

The parent methods are the classical geometric Huygens' construction for moving a surface normal to itself with a constant velocity, and the diffusion-generated motion algorithm for moving a surface by mean curvature, described in detail in Section 3. The strength of Huygens' principle is its geometric formulation, which is easy to visualize and analyze. The strength of diffusion-generated motion is its exceptional computational simplicity. The unification of these disparate methods is itself mathematically interesting, since it highlights aspects of each that are otherwise not obvious. Using the insights thus gained, and the flexibility of the general form, we construct specific methods for a common class of anisotropic, curvature-dependent motion laws. We also show how to implement these algorithms in a computationally efficient and accurate manner. Numerical experiments are used to validate the performance of these new methods.

The outline of the paper follows. In Section 2, we review the relevant previous and related work. In Section 3, we review the standard geometric Huygens' principle, and reformulate it in algebraic form. We also review diffusion-generated motion by mean curvature, and reformulate it in geometric form. Section 4 unifies and generalizes the forms encountered in these two examples. In Section 5, we construct specific convolution generated motion algorithms for anisotropic curvature motion, and validate the resulting algorithm with numerical experiments. In Section 6, we construct algorithms for curvature-independent motions, and validate these numerically. Section 7 combines these methods to obtain more general anisotropic, curvature-dependent motions, and validates the proposed algorithm with numerical experiments. Section 8 summarizes our results and outlines current research. Finally, Appendix A concludes with a description of the fast discretization methods that were used throughout the paper.

2 Background

The developments in this present work were motivated by diffusion-generated motion by mean curvature, so we will review the history of this method, its relationship to other methods for surface motion, and its relation to other classes of mathematical models.

Motion by mean curvature is one of the fundamental models for surface motion, in which the surface normal velocity is simply proportional to the local mean curvature, $v_n = b\kappa$. The original development came about while trying to formulate a level set based method for motion by mean curvature in the presence of triple point junctions.

The level set method of Osher and Sethian [17] was introduced to compute (and define) arbitrary curvature-dependent surface motions, including topological changes. This provides a PDE based method for motion by mean curvature, including the pinch-offs which can occur in three dimensions. Standard numerical PDE methods apply to accurately discretize the equations of motion. However, the original level set method does not apply to the motion of triple point junctions. The level set method was ultimately extended to handle the motion of multiple junctions [15, 30], but the modifications were non-trivial.

In the course of investigating the multiple junction problem, Merriman, Bence and Osher [14, 15] developed the surprisingly simple "diffusion-generated motion by mean curvature" algorithm, which gave mean curvature motion without computing curvature. It also automatically captured topological change and had a direct extension to motion of triple point junctions and arbitrary networks of surfaces. This algorithm is described in detail in Section 3.2, but it essentially consists of moving a set boundary by alternately "diffusing" the set—i.e. applying the linear diffusion evolution equation to the set's characteristic function for a short time—and then recovering a new set via a "sharpening" step in which values of the diffused characteristic function are reset to 0 or 1, whichever is closer. It is essential to apply the diffusion for only a short time, since only the initial boundary motion generated by the diffusion is proportional to the mean curvature.

This procedure of alternately diffusing and sharpening is reminiscent of an operator splitting approximation of the reaction-diffusion, phase-field or Ginzburg-Landau equation type models [4, 3, 8, 6, 21, 20] for motion by mean curvature. In these PDE models, a reaction front develops, separating large regions of constant state. In the asymptotic limit of a strong reaction and weak diffusion, the reaction front moves by mean curvature. While this observation was part of the conceptual motivation for diffusion-generated motion, the ultimate algorithms are quite different. The phase-field models introduce an artificial small length scale—the width of the reaction zone—which, for numerical work, must be resolved by a computational grid or all accuracy is lost for the computed interface motion. This was proven rigorously in [15]. In contrast, diffusion-generated motion has no such artificial small scales. For numerical work, the computational grid need only resolve the natural length scales in the problem, i.e. the curvatures of the evolving surface. Thus diffusion-generated motion has in effect passed to the asymptotic limit of the phase-field class of models, obtaining a simplified and more accurate evolution scheme in the process.

Shortly after the introduction of diffusion-generated motion, there appeared a variety of rigorous convergence proofs for the basic version of the algorithm [7, 1, 13]. The more recent convergence proof of Ishii, Pires and Souganidis [11] covers their generalizations of the original algorithm, which includes the specific algorithms for anisotropic curvature motions we construct in Section 5.

The original numerical implementation of the method was based on naive

discretizations of the diffusion equation on a uniform grid, which gave crude but illustrative results. Ruuth [25, 22, 23] introduced discretizations based on fast Fourier transforms on adaptively refined grids. This approach provides enough accuracy and efficiency to make diffusion-generated motion competitive with more traditional surface evolution discretizations. Moreover, the discretization extends unchanged to the motion of multiple junctions, while other approaches tend to require more complicated implementations to accommodate such features.

In the original presentation of diffusion-generated motion [14], it was pointed out that the diffusive evolution is equivalent to convolution with a Gaussian kernel, and that convolution with any other similarly scaled spherically symmetric kernel would also generate motion by mean curvature. It was also pointed out that this provided a means of generalization, and that the use of non-spherically symmetric convolution kernels could be used to generate anisotropic motions. Further, it was observed that the standard geometric Huygens' principle described in Section 3 also can be written in convolutional form, and thus is subject to the same generalization. Conversely the diffusion-generated motion algorithm can be viewed as a certain geometric Huygens' principle construction. It is these original observations that are reiterated as our motivation in Section 3. The purpose of this present work is to fully develop these early observations, and also unite them with the advanced computational methods of Ruuth in order to produce efficient computational schemes for a more general class of anisotropic, curvature-dependent motions.

Independently of the work on diffusion-generated motion, Gravner and Griffeath [9] developed and studied a class of set evolution algorithms of a somewhat similar form, but with an entirely different perspective and goals. They were looking at ways to determine the limiting shapes produced by cellular automata models for excitable media. Their "threshold growth dynamics" was intended to be a model that captured the smooth, or long wavelength, features of cellular automata evolution rules. Motivated by extensive experiments with cellular automata, they considered an idealized problem of evolving a set by positioning some other shape, scaled to be small, so that the fraction of its volume remaining inside the main set exceeds some threshold, and then taking the set of all such shape centers as the updated set. In our language, we consider this to be the basic form (constant threshold λ) of the generalized Huygens' principle for anisotropic constant motion—i.e.

motion of the form $v_n = a(n)$—as discussed in Section 4. Their goal was to prove that such discrete evolution rules lead to a certain asymptotic limiting shape for the evolving set as time (the number of iterations) goes to infinity. They accomplish this in full rigor and generality, both on a continuum and a lattice.

In continuum terms, and from the viewpoint of convolution generated motion as developed in Section 4, we would say they were analyzing a discrete approximation to a certain anisotropic velocity law $v_n = a(n)$. This velocity law, in turn, could be determined from the general formulas for convolution generated motion velocities recently obtained in [11]. As an aside, note that at this continuum level it is a classical observation about crystal growth (dating back to Gross in 1908) that such anisotropic velocity laws result in well-described limiting shapes as t goes to infinity. However, rigorous proofs of this did not appear until recently. The work of Gravner and Griffeath contains one such proof, though they do not draw this connection explicitly. A simple direct proof for the standard continuum formulation, as well as more detailed discussion and references, are contained in a recent work of Osher and Merriman [16].

Gravner and Griffeath were primarily concerned with the long time limit of their set evolution process. In diffusion-generated motion and its generalizations studied here, the focus is on the point that the entire motion converges in a well behaved fashion to a continuous evolution law for a surface. Also, the evolutions Gravner and Griffeath consider are essentially approximations to anisotropic constant motion, as noted, not to mean curvature or other curvature-dependent motions. To generate curvature-dependent motions requires the use of a different type of scaling for the convolution kernel/shape. These are the respects in which their algorithms differ from diffusion-generated motion and its suggested generalizations. On the other hand, their method is identical to the generalizations of diffusion-generated motion independently suggested for the case of anisotropic constant motion—particularly since they also suggest using arbitrary convolution kernels to replace the geometric shapes. However, these generalizations actually differ from the one we employ for such motions in Section 6, because we make use of normal-dependent thresholds, $\lambda = \lambda(n)$, rather than the simpler constant ones. Constant thresholds are greatly restricted in the class of anisotropic velocities they can achieve, as demonstrated by the example we give in Section 6 as well as the general velocity formulas given in [11].

The "spatially continuous automata" of MacLennan [12] is another independent development that is similar to diffusion generated motion. It also arises from cellular automata, again as a method intended to capture the smoother aspects of automata rules. MacLennan achieves this simply by taking continuous versions of the spatially discrete aspects of cellular automata evolution. The resulting method consists of taking a continuous initial data function, evolving for a discrete timestep by convolving it with a continuous convolution kernel, and then applying a continuous pointwise sharpening step that tends to undo some of the blurring of the convolution step. This procedure is quite similar to diffusion-generated motion (and the general convolution generated motion we present in Section 4), except for one minor but crucial distinction. The simple asymptotics that yield motion by mean curvature in diffusion-generated motion come about precisely because the initial data is the discontinuous characteristic function, and because the sharpening step is discontinuous, replacing the blurred out characteristic function by a new discontinuous characteristic function. Replacing these by continuous analogues destroys simple sharp interface motions in the first few timesteps. Thus, while the spatially continuous automata do produce an interesting and varied class of evolutions, they do not tend to yield well behaved limiting interface motions amenable to asymptotic and rigorous analysis. In this regard, even though they capture the smooth features of cellular automata, they still retain too much of the original complexity.

The work of MacLennan, and of Gravner and Griffeath, illustrates an extremely interesting connection between the type of general convolution generated motions outlined in this present work, and cellular automata. It seems that convolution generated motion provides a natural intermediate mathematical model between automata and PDEs. On the one hand, it captures only the long wavelength limit of the automata, in the form of interface evolutions. On the other hand, the flexibility in choice of kernels and thresholds can be used to create motions more simply or more generally than PDE based surface evolution models such as level set method or phase-field methods. Moreover, this connection with cellular automata is still largely unexplored, since the curvature-dependent motions provided by diffusion-generated motion were not considered in the work of Gravner and Griffeath, much less other possible generalizations. In ongoing research, we are developing convolution generated motion algorithms for capturing a variety of effects observed in cellular automata models. It appears the convolution generated

formalism will lend itself to more efficient computation, as well as simplify the analysis needed to derive an algorithm that generates a desired behavior.

Based on the original diffusion-generated motion algorithm [14] and on the threshold growth dynamics method of Gravner and Griffeath [9], Ishii [10] and Ishii, Pires and Souganidis [11] have developed and rigorously analyzed a class of general methods similar to the ones considered in Section 4. Their most recent work is parallel to, but complementary to, the content of this present paper. For their general class of convolution generated motions, they determine explicit formulas for the limiting surface evolution velocity in terms of moments of the convolution kernels. They also give rigorous proofs that the convolution generated motions converge to these continuous surface evolutions as the timestep goes to zero. In contrast, our goal here is to link the generalized convolution generated motions to generalized versions of Huygens' principle, and use the resulting duality to develop specific convolution kernels for a desirable class of curvature-dependent motion laws. In addition, we want to develop and validate the associated discretization methods needed to effectively compute surface evolutions with these schemes. In this context, the work of Ishii, Pires and Souganidis provides a rigorous proof of convergence for the specific convolution generated motions constructed in Section 5, but is not general enough to apply to the constructions in Section 6. Their analysis also implies the class they consider cannot generate certain anisotropic curvature motions, which justifies our use of more general forms.

As can be seen from this brief overview and history, convolution generated motion has attracted considerable theoretical and computational interest, and has interesting relations and contrast with other methods for surface evolution. It has arisen independently in varied fields of research, and it provides an interesting bridge between cellular automata models and PDE models for systems with dynamic interfaces. We anticipate a great amount of future development as these connections and applications are explored more thoroughly.

3 Motivating Examples

Our generalizations are motivated by the desire to unify the standard geometric Huygens' principle for constant normal motion with the algebraic

method of diffusion-generated motion by mean curvature. Thus we begin by reviewing these procedures and their interesting properties. In preparation for generalization, we also show how each can be expressed in a form similar to that of the other.

3.1 The Standard Huygens' Principle

The standard Huygens' principle is a geometric technique for moving a curve with a constant normal velocity, c. The principle states that the evolved curve at a time Δt can be obtained from the initial curve by the following geometric construction (see Figure 1):

> Draw circles of radius $r = c\Delta t$ which are centered on the initial curve. The forward envelope of these circles is the curve at time $t = \Delta t$.

For our purposes, it is more convenient to reformulate this as (see Figure 2):

> Draw circles of radius $r = c\Delta t$, centered so they are entirely on one side of the curve and tangent to it. The locus of the circle centers forms the new curve position after a time $t = \Delta t$.

As the first step towards generalization, this geometric construction can be translated into an algebraic form. Represent curves as the boundaries of regions, and in turn represent regions by their characteristic functions, i.e. functions that are 1 on the region, 0 off the region. Suppose the original region has characteristic function χ. Select a cylindrically symmetric kernel, K, supported on a disc of radius $c\Delta t$ centered at the origin (e.g., K could be the characteristic function of this disc). Convolve χ with K. Then the updated region is defined as

$$\{\vec{x} : \chi * K(\vec{x}) > 0\},$$

and the updated curve is the boundary of this region.

Thus, the standard geometric Huygens' principle for motion with constant velocity is equivalent to the algebraic procedure of convolving the characteristic function for the original region with an appropriate kernel, and obtaining a new characteristic function from this via thresholding.

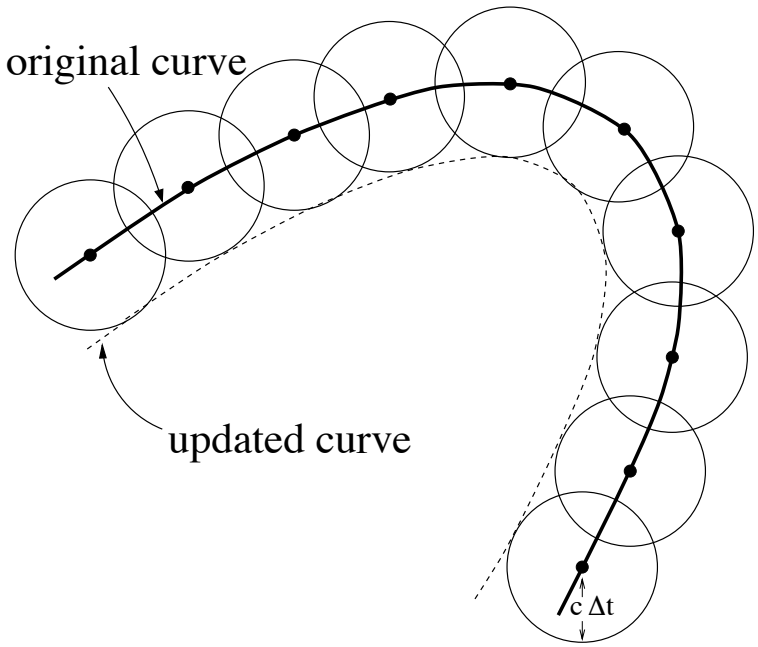


Figure 1: The standard Huygens' principle.

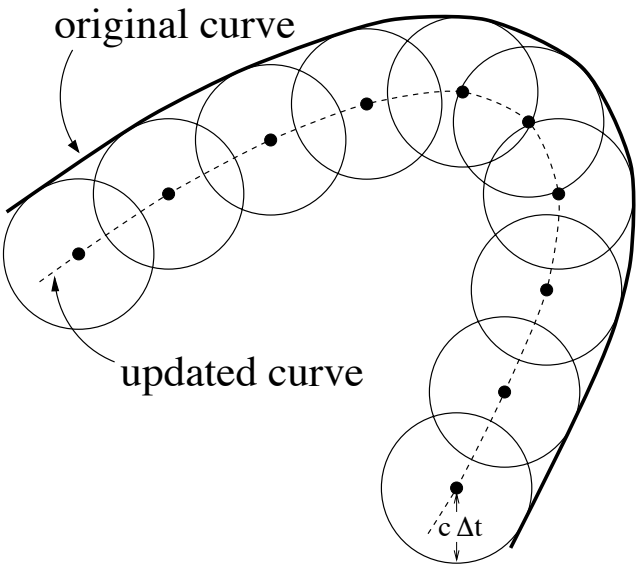


Figure 2: Huygens' principle (reposed).

3.2 Diffusion-Generated Motion by Mean Curvature

Diffusion-generated motion [14, 15] is a particularly simple convolution based algorithm for moving an interface by mean curvature. If the initial surface bounds a region with characteristic function χ, the updated surface at a time Δt is the boundary of the updated region

$$\left\{ \vec{x} : \chi * K(\vec{x}) > \frac{1}{2} \right\} \tag{1}$$

where K is a Gaussian of width $\sqrt{\Delta t}$,

$$K(\vec{x}) = K_G^{\Delta t}(\vec{x}) \equiv \frac{1}{4\pi\Delta t} \exp\left(-\frac{1}{4\Delta t} |\vec{x}|^2 \right)$$

"Diffusion-generated" refers to the fact that convolution with the Gaussian kernel can be realized by solving the diffusion equation for a time Δt, with χ as initial data. Thus this procedure can be described informally as diffusing the set for a short time, and then thresholding at the $\frac{1}{2}$ level to obtain a new set. It is intuitively clear that such a diffusion will cause a curvature-dependent blurring of the set boundary, and a formal analysis of the diffusion equation [14, 15] shows this should result in precisely motion by mean curvature.

An interesting variety of rigorous proofs have been given to show this simple algorithm converges to motion by mean curvature as the time step goes to zero [7, 1, 13, 11].

This algorithm has several remarkable properties: Motion by mean curvature is obtained in any number of dimensions without ever directly computing the mean curvature. Topological mergers such as pinch off, which occur in higher dimensions, are captured with no special algorithmic procedures. Note also that motion by mean curvature is a nonlinear evolution, yet the diffusive evolution is entirely linear, with the only nonlinear part of the algorithm being the final, trivial, thresholding step.

Perhaps most remarkable, this procedure has a direct extension to the motion of multiple junctions. Let the intersecting surfaces partition the domain up into regions with characteristic functions $\chi_1, \chi_2, \ldots, \chi_N$. Note $\sum \chi_i = 1$ everywhere, reflecting the partition. We independently diffuse each region—i.e. convolve χ_i with the Gaussian—to obtain smoothed out characteristic

functions $\chi_i(\Delta t)$. Note that these still sum to one, by the linearity of the convolution: $\sum \chi_i(\Delta t) = \sum K * \chi_i = K * \sum \chi_i = K * 1 = 1$. Thus the smoothed out characteristic functions still partition the domain into "fuzzy" sets. In order to obtain a partition into geometric sets, we simply define set i to be the set on which $\chi_i(\Delta t)$ is greater than all the other smoothed out characteristic functions. In the case of two regions, this reduces to the standard algorithm. This symmetrical χ comparison produces symmetrical triple point junctions, but arbitrary desired junction angles can be obtained by using a nonsymetrical comparison, as described in [15, 13, 24]. There are no rigorous results concerning this algorithm for multiple junctions, but the numerical experiments in [23, 25, 24] demonstrate its convergence.

Volume preserving motion by mean curvature [5, 19], i.e. $v_n = \kappa - \bar{\kappa}$ where $\bar{\kappa}$ is the surface average of the mean curvature, is another interesting flow that is easily approximated by diffusion-based methods. Such motions are realized by selecting the level that encloses the same volume as the original set, instead of the $\frac{1}{2}$ level [26]. Convergence of this procedure has been demonstrated numerically, but not proven analytically.

Any positive, radially symmetric kernel may be used in place of the Gaussian to obtain a convolution generated mean curvature motion, as was pointed out in [14] and proven rigorously in [10]. For example, in two dimensions we can take K to be the (normalized) characteristic function for a disc of radius r, centered at the origin,

$$K(\vec{x}) = \begin{cases} \frac{1}{\pi r^2} & \text{if } |\vec{x}| < r \\ 0 & \text{otherwise.} \end{cases}, \tag{2}$$

where $r \sim \sqrt{\Delta t}$.

Diffusion-generated motion by mean curvature was conceived of in its convolution form, in connection with solving reaction-diffusion equations. But just as the standard geometric Huygens' principle had an algebraic formulation similar to that of diffusion-generated motion, so does diffusion-generated motion have a geometric formulation similar to that of the standard Huygens' principle [14].

Geometrically, the version of the algorithm based on the disc kernel (2) can be described as positioning the discs so that exactly half their *area* lies inside the curve to be evolved, and then taking the locus of all disc centers as the new curve, as illustrated in Figure 3. We see this is just a slight

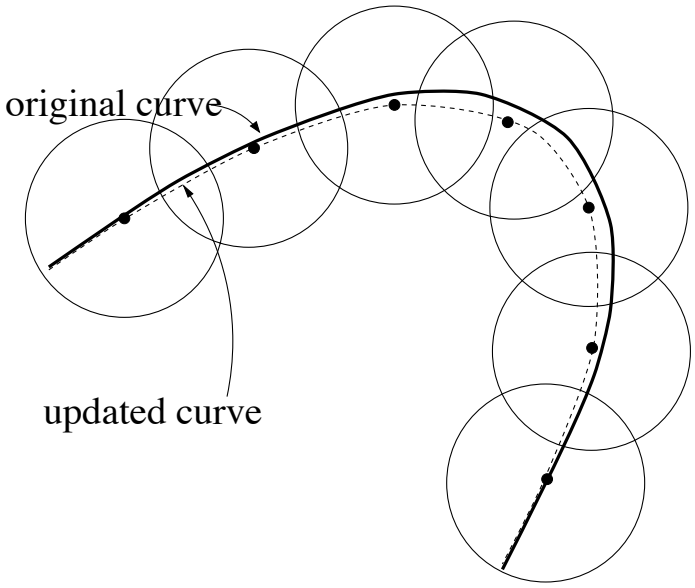


Figure 3: Huygens' principle for a curvature-dependent motion.

modification of the standard Huygens' principle shown in Figure 2, yet it yields a qualitatively different type of motion.

The geometric Huygens' principle for mean curvature motion is not as obvious as that for constant motion, so let us briefly note why it works. It is clear from Figure 3 that the most curved portions of the interface are displaced the most by this process, so that it induces some form of curvature-dependent motion. A simple geometric analysis (see Figure 4) shows that if the local radius of curvature of the curve is R, and we position a disc of radius $r \ll R$ so that it is cut exactly in half (by area) by the curve, then the disc center is displaced normal to the curve by a distance $d \sim \frac{r^2}{R}$. We would like this displacement to represent one timestep of motion by mean curvature, so we want $d = v_n \Delta t$, with $v_n \sim \kappa = \frac{1}{R}$. This will indeed be the case as long as $r \sim \sqrt{\Delta t}$. Note this also explains why the Huygens' principle for mean curvature uses discs of radius $r \sim \sqrt{\Delta t}$, while that for constant motion uses discs of radius $r \sim \Delta t$. (This distinction has practical importance for numerical implementations, since the kernels for constant motion have smaller supports and so require more spatial resolution than those for curvature motion.)

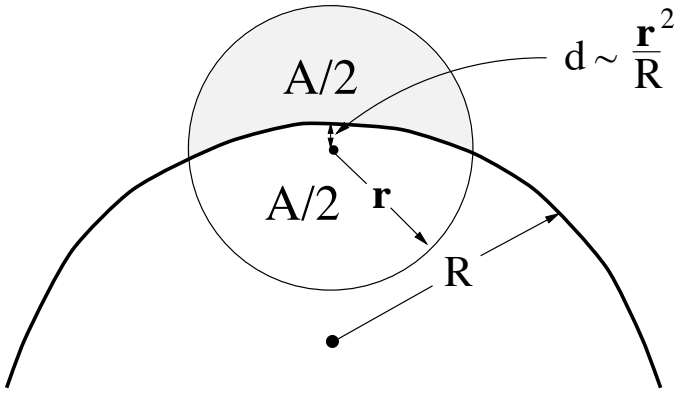


Figure 4: The geometry of the Huygens' principle for motion by mean curvature.

4 Generalized Huygens' Principles

We have seen that two basic surface motion laws—motion by a constant and motion by mean curvature—can be formulated in terms of both geometric and algebraic Huygens' principles, with the algebraic form based on convolutions. Using these examples as inspiration and motivation, we want to generalize Huygens' principle to a form that can represent a large class of surface motion laws.

4.1 Algebraic Generalization

We will first generalize Huygens' principle in its algebraic form. The algebraic form is preferred because it leads to a larger class of generalizations, and because its expression in terms of convolutions is better suited to mathematical analysis and numerical computation.

Two natural generalizations are:

1. Allow a general threshold, λ, in $\{\vec{x} : \chi * K(\vec{x}) > \lambda\}$.

This provides a continuum of Huygens' principles parameterized by λ, with $\lambda = 0$ corresponding to the standard Huygens' principle, and $\lambda = \frac{1}{2}$ corresponding to motion by mean curvature. In general, λ can also be allowed to depend on other quantities. For example, a variety of $v_n = a + b\kappa$ diffusion-generated motions can be obtained with $\lambda = \frac{1}{2} + c\sqrt{\Delta t}$ [11, 13, 24], so $\lambda = \lambda(\Delta t)$ is a useful form. More generally, in Section 6 we show it is useful and practical to select λ locally as a function of the normal direction defined by the level sets of $K * \chi$, $\lambda = \lambda(n)$.

2. Allow different convolution kernel functions[1], K.
The method formally allows arbitrary kernel functions, and asymmetrical kernels can be used to produce anisotropic motion laws, as originally suggested in [14]. For numerical work, we construct kernels which are simple, easy to compute with and can produce general motion laws.

Note that in the examples of motion by a constant and motion by mean curvature, the kernels also had a radius that scaled like Δt and $\sqrt{\Delta t}$, respectively. We do not consider generalizing this scaling relationship, since this is fixed by requiring that each application of the convolution generate one "timestep" of some motion law.

More specifically, the generalized method consists of selecting a fixed kernel $K(\vec{x})$, and then scaling it down in a mass preserving way to have a radius $r \ll 1$, i.e. $K(\vec{x}/r)/r^d$ in d dimensions. We convolve this scaled kernel with χ to generate one step of the motion, and after thresholding by λ the set boundary will be displaced by an amount that is some function of r, $s(r)$. If we demand that in the limit of small r this displacement be one timestep of a limiting motion law, $s(r) = v_n \Delta t$, the relation between r and Δt is fixed. Thus the r–Δt relationship is just an artifact of our definition of time evolution, and not suitable for independent generalization.

4.2 Geometric Generalization

These generalized algebraic Huygens' principles have associated geometric formulations. To translate back to geometric language, we need to give a

[1]Without loss of generality, we shall assume that the kernel has been normalized to satisfy $\int K(\vec{x}) d\vec{x} = 1$.

geometric description of the boundary of the set $\{\chi * K > \lambda\}$, i.e. the points $\vec{x}$ where $\chi * K(\vec{x}) = \lambda$. Note that we can visualize computing the value of $\chi * K$ at a point $\vec{x}$ by rigidly translating the graph of K so that the origin point is located at $\vec{x}$, and then computing the portion of the mass of this shifted K that is contained in the set represented by χ. Thus these boundary points are the location of the origin points when the graph of K is translated so that exactly the fraction λ of its mass is enclosed in the set. The geometrization of the principle is complete if we restrict the kernel K to be the (normalized) characteristic function of some geometric shape, since then we are translating this shape so that a fraction of its area is contained inside the set. In this case, the convolution based procedure can be stated entirely in geometric terms as follows:

> Select an arbitrary shape (generalizing the disc of the standard principle) and an "origin point" for the shape (generalizing the disc center), which can be any point inside or outside the shape. Allow the shape and its associated origin to be moved in the plane only by rigid translation (not rotations). Given an initial curve, everywhere possible position the shape so that a fraction λ of its total area is enclosed in by the curve. Then the updated curve is the locus of all the corresponding origin points.

Thus we see that in geometric terms the generalization of Huygens' principles corresponds to using shapes other than discs, taking the locus of designated points other than the disc centers, and positioning the discs fractionally outside the curve, rather than entirely outside or half in and half out. The use of general convolution kernels gives us a further extension that can be thought of as using "fuzzy" shapes instead of geometric shapes. This shows why the algebraic formulation lends itself to generalizations that would not be geometrically obvious.

It is also interesting to note that there are actually several different types of geometric Huygens' principles, depending on the dimension of the "shape", i.e., the dimension of the support of the kernel K. If K is a unit mass uniformly distributed on a two dimensional set, we have the Huygens' principles based on positioning this shape with its area partially enclosed by the curve, as just described. But, if instead K is a unit mass distributed uniformly (with respect to arc length) on a one dimensional curve—e.g. a circle—then we get a principle based on positioning this curve so that a fraction λ of its

total arc length is enclosed by the curve being updated. Finally, if K is a unit mass distributed uniformly on a finite set of N points, we get principles based on positioning the points so that some rational fraction $\lambda = \frac{m}{N}$ of them lie inside the curve being evolved. In all cases, once the geometric object is positioned with a fraction λ of its measure enclosed by the curve, the locus of corresponding origin points define the updated curve.

These Huygens' principles based on positioning lower dimensional objects are too singular for convenient numerical computation (since K is a singular distribution), but they can be easy to treat analytically, and thus provide a good source of exactly solvable examples.

4.3 Obtainable Motion Laws

It is natural to ask what class of motion laws can be obtained by these generalized Huygens' principles. This question has been posed and answered (independently of this present work) in the recent and comprehensive work of Ishii, Pires and Souganidis [11] for convolution generated motion with λ a constant or $\lambda = \lambda(\Delta t)$. They give explicit formulas for the limiting surface normal velocity v_n in terms of various moments of the kernel function, in any number of dimensions. Moreover, they also give rigorous proof that the convolution generated motions converge to their stated v_n motion laws in the limit as $\Delta t \to 0$.

One notable implication of their results is that it impossible to obtain anisotropic motion by mean curvature in three dimensions with this class of generalizations. I.e., the motion law

$$v_n = b(n)\kappa, \tag{3}$$

where κ is the mean curvature, can only be obtained in > 2 dimensions if b is constant—in which case the original diffusion-generated motion algorithm applies. The source of this limitation in higher dimensions can be understood by the same geometric analysis of curvature motion employed in two dimensions in Figures 4 and 5. Viewed in local geometry, it is clear that the principle curvatures of the surface have independent influences on the positioning of any non-spherical Huygens shape (kernel), so it is not simply their average, κ, that determines the motion.

However, at a more fundamental level these unobtainable motions simply reflect the limitations of the chosen form of the generalization. In contrast, the basic concept of convolution generated motion has a great deal of unexplored flexibility. For example, it turns out that by using two separate, spherically symmetric, convolution kernels, K_1 and K_2, and thresholding a convex combination of $K_1 * \chi$ and $K_2 * \chi$, it is easy to obtain the anisotropic mean curvature motion (3) that is impossible to obtain with a single kernel. For another example, by using two kernels and thresholding based on the differences between $K_1 * \chi$ and $K_2 * \chi$, it is possible to obtain a great variety of steady state and dynamic interface motions associated with pattern formation. Or, by allowing λ to have a nonlocal dependence on $K * \chi$ in diffusion-generated motion—specifically, define λ to be the level enclosing the same total volume as the original set—we obtain a simple algorithm for *volume preserving* motion by mean curvature. All these convolution generated motion techniques are beyond the scope of generalizations considered here, and are the subject of current research and forthcoming reports. The essential point is that at this early stage of development, convolution generated motion offers a wealth of unexplored generalizations. The limitations of any particular form simply provide motivation to look for others.

4.4 Huygens' Principles for Specific Motion Laws

In the previous section, we discussed the problem of determining the motion law, given a kernel. Even with a comprehensive answer to this, there still remains the inverse problem of constructing a specific kernel that achieves a given surface motion law, v_n. For numerical work, we have the further constraint that the kernel must be computationally convenient.

We will solve this problem explicitly by constructing Huygens' principles—i.e. the kernel K and fraction λ—for a fairly broad class of geometric curve motions, and validating their performance with numerical experiments.

Specifically, consider motions in two dimensions where the curve normal velocity, v_n, is a function of the local unit normal vector n and the local curvature, κ, $v_n = v_n(n, \kappa)$. We will construct specific Huygens' principles for all laws of the form

$$v_n = a(n) + b(n)\kappa$$

where $b(n) \geq 0$ and $b(n) = b(-n)$. The restriction $b \geq 0$ is necessary for the motion to be mathematically well posed. The additional constraint on b is a limitation of our construction, but it simply means that the motion does not depend on which of the two possible orientations, $\pm n$, we choose for the curve.

We carry out the kernel constructions and computations in two dimensions, to simplify the presentation. The construction for the constant part of the motion $a(n)$ extends trivially to higher dimensions. However, as noted in the previous section, it is impossible to achieve anisotropic motion by mean curvature in higher dimensions, so our construction for that part of the motion has no extension.

In what follows, we will represent the unit normal vector, n, by the angle, θ, it makes with some reference line, which we take to be a horizontal line in our illustrations. Thus the motion laws have the form

$$v_n(\theta, \kappa) = a(\theta) + b(\theta)\kappa$$

where $b(\theta) \geq 0$ and $b(\theta) = b(\theta + \pi)$.

Our approach is to obtain separate Huygens' principles for the anisotropic constant motion $v_n = a(\theta)$ and anisotropic mean curvature motion $v_n = b(\theta)\kappa$, and then combine these to obtain the Huygens' principle for the composite motion.

5 Anisotropic Curvature Motion

Here we address the problem of finding a Huygens' principle for the anisotropic, pure curvature motion

$$v_n = b(\theta)\kappa.$$

We will construct a suitable kernel and validate our construction with numerical experiments. Note that we will generate this motion with no explicit calculation of the normal angle θ or curvature of the curve—these details are all implicitly captured by our choice of kernel.

5.1 Construction of the Kernel

By adapting the geometric analysis developed to study diffusion-generated motion, we can construct non-symmetric shapes for generating anisotropic curvature motions of the form

$$v_n = b(\theta)\kappa. \tag{4}$$

Our goal is to determine a shape such that when we position translates of it exactly halfway (by area) inside the curve, the locus of all shape center points is the curve advanced by $v_n \Delta t$. To begin, we restrict attention to a small neighborhood of a point $\vec{p}$ on the curve being updated, as shown in Figure 5. Locally, we can approximate the curve by the circle of curvature, so we consider it to be the arc of circle of constant curvature κ. Select a coordinate system so that the y-axis is normal to and intersects the curve at $\vec{p}$, and the origin is located at the center of the unknown shape. If we assume the shape is going to be symmetrical about its "center", the x-axis must cut the shape in half by area, because it runs through the center. But we also assume the shape is to be positioned so that the curve cuts it in half. Thus, since both the curve and x-axis cut the shape in half, the total (signed) area between these cuts must add up to zero. This condition relates the displacement of $\vec{p}$ from the origin to the width of the shape—extending from $x = -d$ to $x = d$—and allows us to determine d in terms of b. More precisely, since the update is going to move $\vec{p}$ to the origin, we require this distance to be the desired propagation distance $v_n(\vec{p})\Delta t = b(\theta)\kappa\Delta t$. The local equation of the curve is $y = \frac{\kappa}{2}x^2 - b\kappa\Delta t$, so the zero area condition is

$$\begin{aligned} \int_{-d}^{d} \frac{\kappa}{2}x^2 - b(\theta)\kappa\Delta t \; dx &= 0 \\ \Rightarrow d &= \sqrt{6b(\theta)\Delta t} \end{aligned}$$

to leading order. Applying this result for all normal directions θ gives a simple equation for the boundary of the shape in polar coordinates,

$$r(\theta) = \sqrt{6b(\theta + \pi/2)\Delta t}.$$

In the derivation, we assumed the shape was symmetrical about its center, i.e. $r(\theta) = r(\theta+\pi)$, which implies the condition $b(\theta) = b(\theta+\pi)$. The asymptotics underlying this analysis require r, and hence b, to be continuous.

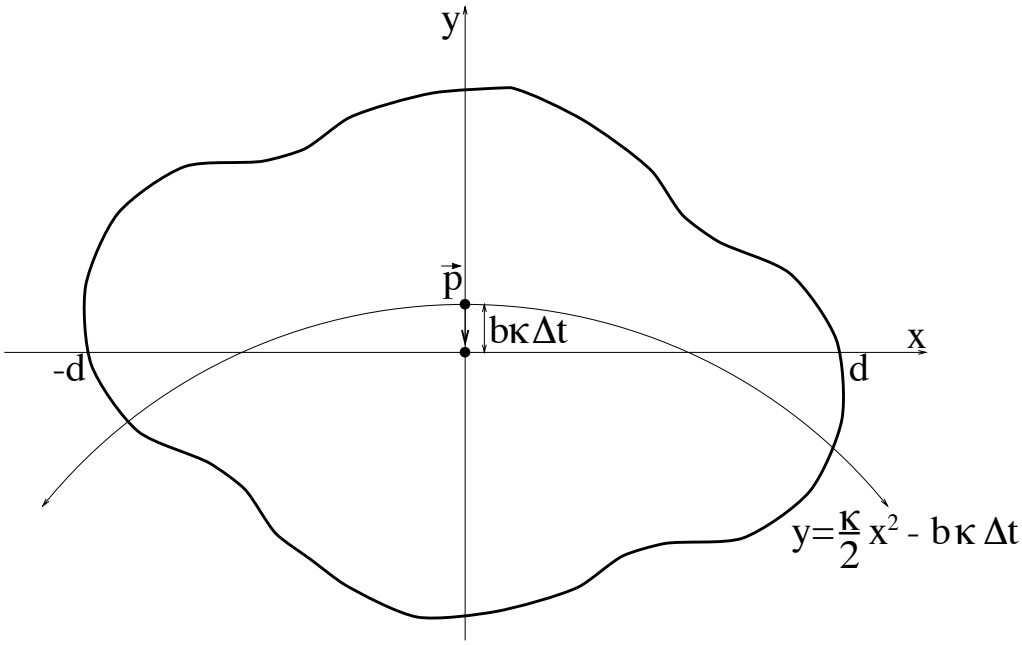


Figure 5: Derivation of a anisotropic shape for Huygens' principle.

Combining the results of this geometric analysis with our convolution formalism, we get that the desired motion law (4) is obtained in the limit as $\Delta t \to 0$ by advancing the front according to the symmetric update (1) using the (normalized) characteristic shape function,

$$K(r,\theta) = \frac{1}{\text{Area}(S)} \begin{cases} 1 & \text{if } (r,\theta) \in S \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

for the region

$$S = \left\{ (r,\theta) : r \leq \sqrt{6b(\theta + \pi/2)\Delta t} \right\}.$$

The analysis in [11] applied to this shape provides a rigorous proof of the convergence of this method for $v_n = b(\theta)\kappa$.

This generalized Huygens' principle for anisotropic curvature motion will also prove useful for deriving more general motions. See Section 7.

5.2 Numerical Experiments

We now apply our generalized Huygens' principle to the problem of evolving curves according to the anisotropic curvature model (4).

Consider, for example, the motion of the simple closed curve given in Figure 6. Using our new Huygens' principle for anisotropic curvature motion, the maximum error[2] in the position of the front at time $t = 0.05$ was computed for several Δt. The results for a number of experiments are reported in Table I, below.

Δt	*Error*
0.005000	0.02986
0.002500	0.01208
0.001250	0.00565
0.000625	0.00264

Table I. Errors for an anisotropic curvature motion.

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

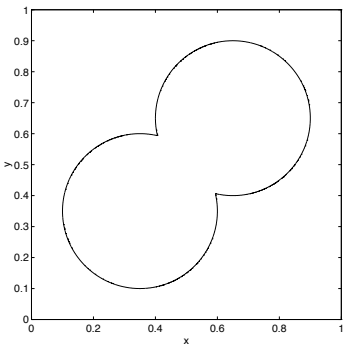
6 Anisotropic Curvature-Independent Motions

Here we construct a generalization of Huygens' principle to produce anisotropic curvature-independent motions,

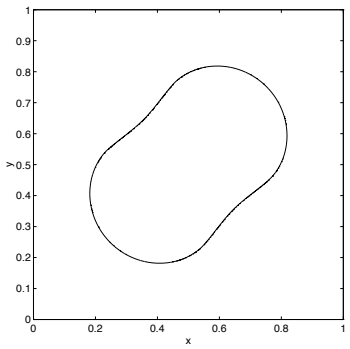
$$v_n = a(\theta) \tag{6}$$

and validate the algorithm with numerical experiments.

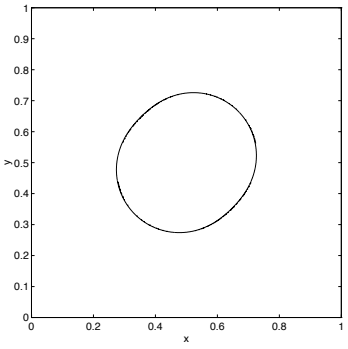
[2]Throughout this article, accurate approximations of the exact front location are computed using the Hamilton-Jacobi level set formulation of Osher and Sethian. See [17]. The introduction (Section 1) explains why the level set method does not supersede convolution generated motion in general.



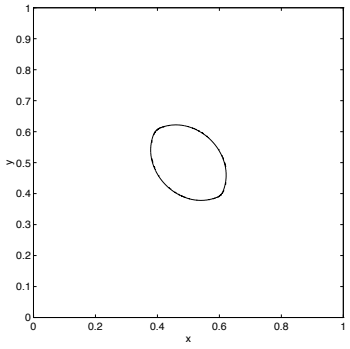
(a) $t = 0.0000$



(b) $t = 0.0167$



(c) $t = 0.0333$



(d) $t = 0.0500$

Figure 6: A test problem at various times, t. The normal velocity is given by $v_n = b(\theta)\kappa$ where $b(\theta) = \frac{1}{4} + \frac{5}{4}\left|\sin(\theta + \frac{\pi}{4})\right|$.

6.1 Limitations of Updates with Constant Threshold

A variety of anisotropic motions can be obtained by selecting appropriate kernels and by choosing the proportion, λ, of the kernel that lies inside the initial region. Unfortunately, even certain simple motions are impossible to simulate using this approach if λ is chosen to be a constant independent of the normal direction. We now demonstrate this fact for the geometric version of our generalized Huygens' principle for the simple motion law

$$v_n = |\sin(\theta)|. \tag{7}$$

To begin, assume that there is a shape, S, which generates the desired motion (7) and let S_0 be the origin point used to trace out the updated curve. The shape, S, is placed so that a proportion, λ, is inside the initial region. Because vertical line segments remain stationary under the motion law (7), S_0 must lie on the interface for both of the shapes displayed in Figure 7. But S_0 is fixed relative to S, so λ must equal $1/2$. The position of S_0 may now be determined using the fact that the updated curve lies a distance Δt outside the initial rectangle for horizontal segments. Carrying out this construction gives different positions for S_0—and hence different velocities of motion—when $\theta = \pi/2$ and $\theta = -\pi/2$ (see Figure 8), in contradiction to our assumed velocity law (7).

Because general anisotropic motions such as (7) cannot be reproduced using our generalized Huygens' principle with a constant λ, we focus our attention on methods which select λ as a function of the normal direction, i.e.,

$$\lambda \equiv \lambda(\theta).$$

6.2 Construction of a Normal Dependant Threshold

Based on the geometric form of Huygens' principle, the normal velocity $v_n = a(\theta)$ can be easily obtained by varying the proportion, λ, of a shape lying inside the initial region according to the normal direction.

To illustrate this idea, consider the motion of a straight line moving with a normal velocity $v_n = a(\theta)$. Geometric analysis (see Figure 9) shows that if

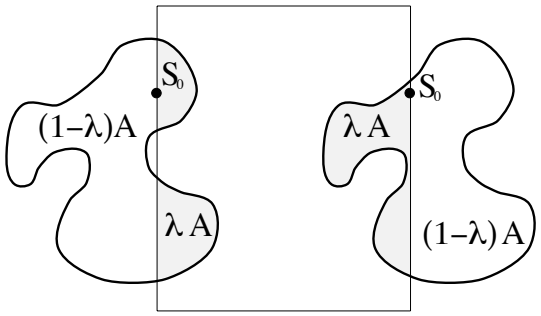


Figure 7: Choosing λ to achieve vertical interface velocity $v_n = 0$.

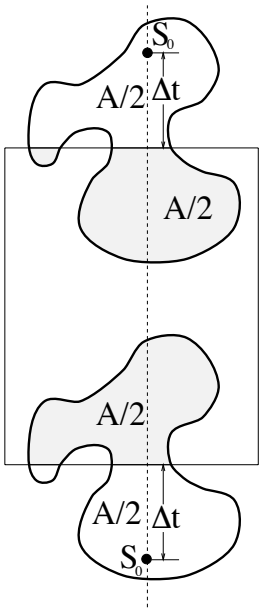


Figure 8: Contradictory position of the origin point S_0 for horizontal interface velocity $v_n = 1$.

discs of radius r are placed so that a proportion,

$$\lambda(\theta) = \frac{1}{2} - \frac{1}{\pi}\arcsin\left(\frac{a(\theta)\Delta t}{r}\right) - \frac{a(\theta)}{\pi\, r^2}\Delta t \cos\left(\arcsin\left(\frac{a(\theta)\Delta t}{r}\right)\right) \quad (8)$$

of each disc is inside the initial region, then the updated front location is given exactly by the locus of disc centers.

The motion of general smooth curves may also be approximated using this technique. For such problems, small[3] discs of radius $r = \mathcal{O}(\Delta t)$ are placed so that a proportion (8) lies inside the initial region. The locus of disc centers then gives an approximation of the front location after a time Δt (see, e.g., Figure 10). Equivalently, this motion may be generated by updating the region according to

$$\{\vec{x} : \chi * K(\vec{x}) > \lambda(\theta)\} \quad (9)$$

where χ is the characteristic function for the initial region, K is the (normalized) characteristic function for a small disc and $\lambda(\theta)$ is given by Eq. (8). Using the considerations of Section 5.1, it is easy to show that this approach gives a locally first order approximation

$$u_n = a(\theta) + \mathcal{O}(\Delta t \kappa)$$

of the desired motion law (6) for smooth curves.

Of course, other kernel functions may used to generate a first order approximation of the desired motion. We have found that a Gaussian kernel,

$$K(\vec{x}) = K_G^{\sigma^2}(\vec{x}) \equiv \frac{1}{4\pi\sigma^2}\exp\left(-\frac{1}{4\sigma^2}|\vec{x}|^2\right) \quad (10)$$

where $\sigma \sim \max_\theta |a(\theta)|\,\Delta t$ is an especially attractive choice because:

1. The smooth kernel, $K_G^{\sigma^2}(\cdot)$, can be represented using fewer Fourier basis functions than the characteristic function for a disc of radius Δt. Furthermore, the convolution product $\varphi(\vec{x}) = \chi * K_G^{\sigma^2}(\vec{x})$ is continuously differentiable. Thus, a first order approximation of the normal direction can be obtained by approximating $\nabla\varphi/|\nabla\varphi|$. See Appendix A.

[3]For example, the choice $r = \max_\theta |a(\theta)|\,\Delta t$ will suffice.

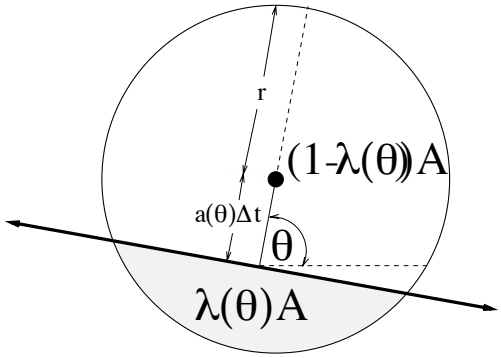


Figure 9: Determination of the proportion, $\lambda(\theta)$, of the disc which is inside the region.

2. The function $\lambda(\cdot)$ has a simple analytic expression,

$$\lambda(\theta) = \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(\frac{a(\theta)}{2\sigma/\Delta t}\right). \tag{11}$$

3. The Fourier transform of the Gaussian kernel is also given by a simple analytic expression,

$$\begin{aligned} \hat{K}_G^{\sigma^2}(j,k) &= \int_0^1 \int_0^1 K_G^{\sigma^2}(x,y)\exp(-2\pi ijx)\exp(-2\pi iky)\; dx\; dy \\ &= \exp(-4\pi^2(j^2+k^2)\sigma^2). \end{aligned}$$

(We use the Fourier transform of the kernel to compute the convolution product, $\chi * K$. See Appendix A.)

The accuracy of the method will depend on the effective radius of support, σ. If σ is too large, then the error will be dominated by artificial diffusion-generated curvature motion. Conversely, if σ is too small then very little damping of high frequency error modes occurs and oscillations may dominate

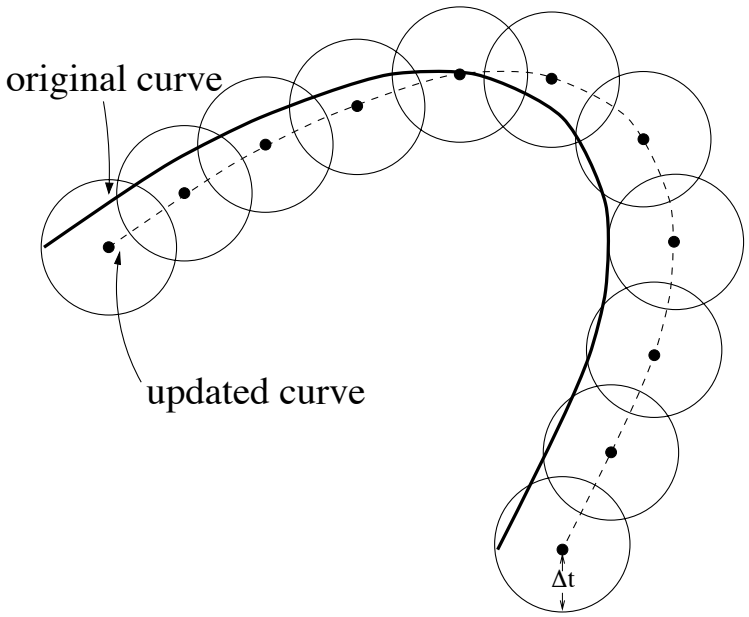


Figure 10: If a proportion $\frac{1}{2} - \frac{1}{\pi}\arcsin(\cos(\theta)) - \frac{\cos(\theta)}{\pi\Delta t}\cos(\arcsin(\cos(\theta)))$ of each disc is inside the initial region, then an approximation to the normal velocity $v_n = \cos(\theta)$ is generated.

the error. Small kernels are also more computationally expensive to use because they require more Fourier modes to be adequately represented (see Appendix A). Although the optimal choice of σ will be problem-dependent, it is clear that $\sigma \sim \max_\theta |a(\theta)| \Delta t$ since the kernel should have a radius of support that is comparable to the maximum displacement over one timestep. Indeed, our numerical studies simply select

$$\sigma = \max_\theta |a(\theta)| \Delta t \tag{12}$$

since this choice gives very good results for an interesting variety of problems. Note that this added dissipation is similar to the use of artificial dissipation to stabilize classical difference schemes for advection equations.

6.3 Numerical Experiments

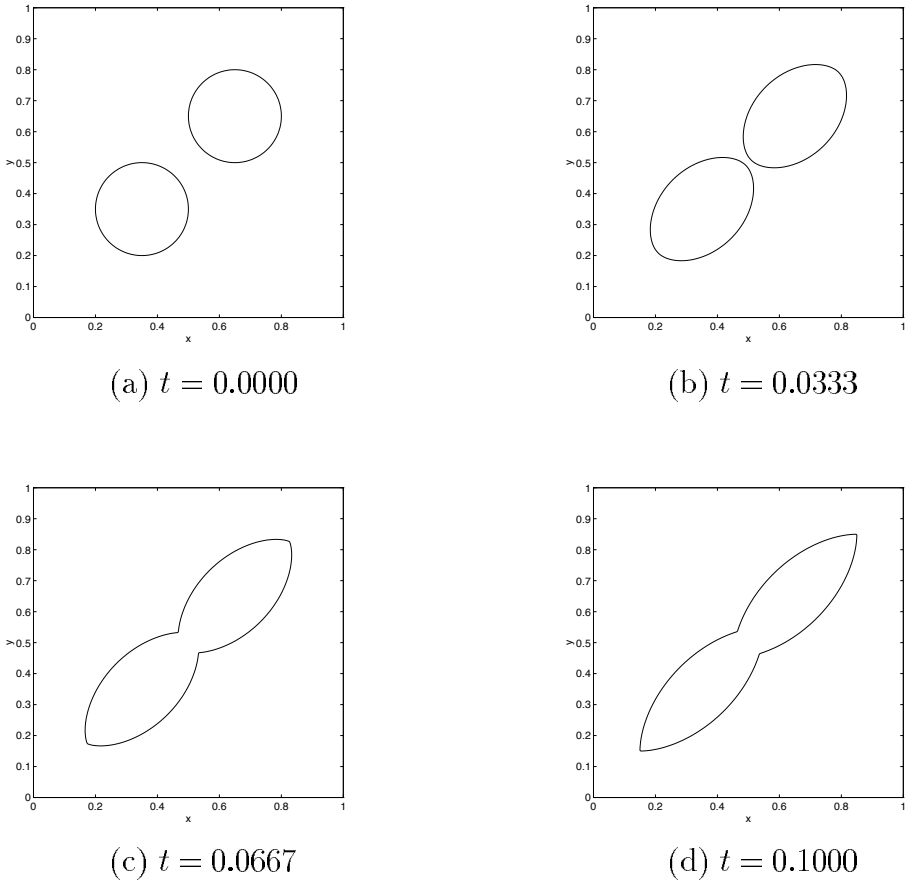
Several numerical experiments were carried out to study our methods. We now describe some of these experiments for the model problem given in Figure 11.

To begin, suppose that regions are updated using a kernel which is a characteristic shape function and that normal directions are approximated using $\nabla\chi * K(\vec{x})/|\nabla\chi * K(\vec{x})|$. For such methods, it was found that oscillations in the front could develop, leading to an $\mathcal{O}(1)$ error in the solution. Fortunately, such errors are greatly reduced by selecting a Gaussian kernel for updates and normal calculations. See, e.g., Figure 12.

Using the Gaussian kernel (10), and the proposed choice of σ (see Eq. (12)), solutions to the model problem were computed. Based on these results, the maximum error in the position of the front was calculated at time $t = 0.1$ for several Δt. The results for a number of experiments are reported in Table II, below.

Δt	*Error*
0.0200	0.03106
0.0100	0.01387
0.0050	0.00519
0.0025	0.00194

Table II. Errors for an anisotropic motion.



(a) $t = 0.0000$ (b) $t = 0.0333$

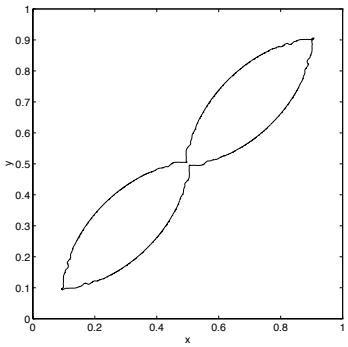
(c) $t = 0.0667$ (d) $t = 0.1000$

Figure 11: A test problem at various times, t. The normal velocity is given by $v_n = a(\theta)$ where $a(\theta) = \frac{1}{2} + \sin(2\theta)$.

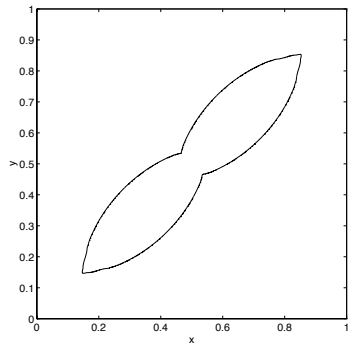
As expected from the discussion of the previous section, these results are suggestive of an approximately first order error in the position of the front.

7 General Anisotropic Motions

In the previous sections, we developed methods for generating a variety of anisotropic motions. We now combine these methods to produce more gen-



(a) A nonsmooth kernel. K is given by Eq. (2) with $r = 2\Delta t$.



(b) A Gaussian kernel. In this example, $K = K_G^{\Delta t^2}$.

Figure 12: Results using different kernel functions.

eral anisotropic motions,

$$v_n = a(\theta) + b(\theta)\kappa, \tag{13}$$

and validate algorithms with numerical experiments.

7.1 A Simple Algorithm

As we have seen in Section 5.1, a normal velocity

$$v_n = b(\theta)\kappa \tag{14}$$

is generated by advancing the front according to the symmetric update (1) using a (normalized) characteristic shape function (5). Other anisotropic motions,

$$v_n = a(\theta)$$

are obtained using non-symmetric updates of the form,

$$\{\vec{x} : \chi * K(\vec{x}) > \lambda(\theta)\}. \tag{15}$$

By combining the shape for anisotropic curvature motion with non-symmetric updates, methods for general motions (13) may be derived.

To see this, consider the evolution of a straight line moving with the general normal velocity (13). (A method for the isotropic case is derived in [13].) Now, position a shape

$$S = \left\{(r,\theta) : r \leq \sqrt{6b(\theta + \pi/2)\Delta t}\right\}$$

so that the center point is a distance $a(\theta)\Delta t$ away from the front. It is easy to see, geometrically, that a proportion

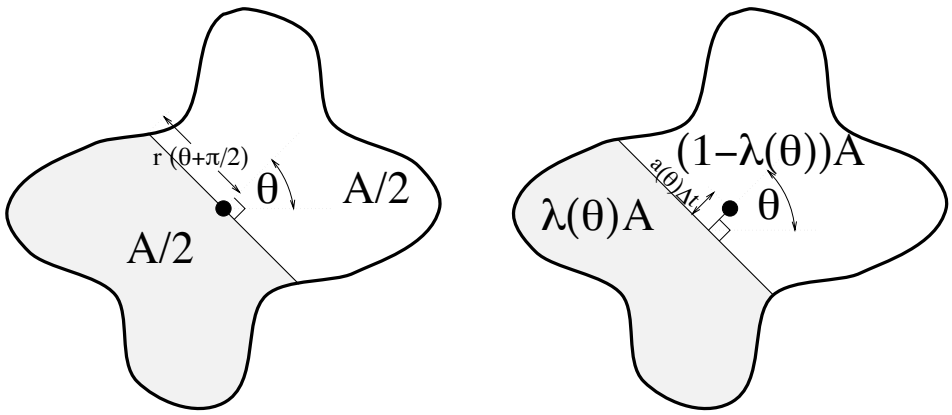
$$\lambda(\theta) = \frac{1}{2} - \frac{2a(\theta)\sqrt{6b(\theta + \pi/2)\Delta t}\Delta t}{\text{Area}(S)} \tag{16}$$

of S lies inside the initial region (see Figure 13). Thus, for straight lines the desired motion (13) is obtained using the non-symmetric update (15) with Eq. (16). Of course, even curved segments may be treated in this manner when $b(\cdot)$ is strictly positive since the shape S contributes a curvature component (14) to the motion. Based on this fact, we obtain the following method for general anisotropic motion (13):

ALGORITHM **Anisotropic**
GIVEN: A motion law, $v_n = a(\theta) + b(\theta)\kappa$, and an initial region, R.

BEGIN
(1) Construct the kernel, K, from Eq. (5).
Set χ equal to the characteristic function for R.
(2) REPEAT for all steps:
BEGIN
Set χ equal to the characteristic function for the updated region (15) where $\lambda(\cdot)$ is given by Eq. (16).
END
END

Remark: If $b(\theta) = 0$ for some θ, we may split the motion law (13) and alternately treat the curvature-dependent and curvature-independent components using the methods of Sections (5.1) and (6.2), respectively. When $b(\cdot)$ is strictly positive, however, the algorithm **Anisotropic** is usually preferred because fewer Fourier modes are (typically) required to adequately represent the relatively large kernel function and because only one update is required per time step.



(a) For anisotropic curvature motion $r(\theta) = \sqrt{6b(\theta + \pi/2)\Delta t}$ and $\lambda = 1/2$.

(b) For general anisotropic motions $\lambda(\cdot)$ depends on θ.

Figure 13: To achieve a normal velocity $v_n = b(\theta)\kappa$, we select a shape according to (a) and set $\lambda = 1/2$. This same shape produces a normal velocity $v_n = a(\theta)$ for a straight line if $\lambda = 1/2 - 2a(\theta)r(\theta + \pi/2)\Delta t/A$ (see (b)). For curved fronts, both contributions occur and an approximation to $v_n = a(\theta) + b(\theta)\kappa$ is produced.

7.2 Numerical Experiments

We now apply our generalized Huygens' principles to the problem of evolving curves according to the general anisotropic model (4).

Consider, for example, the motion of the closed curves given in Figure 14. Using our algorithm **Anisotropic**, the maximum error in the position of the front at time $t = 0.03$ was computed for several Δt. The results for a number of experiments are reported in Table III, below.

Δt	*Error*
0.00600	0.05499
0.00300	0.02933
0.00150	0.01579
0.00075	0.00809

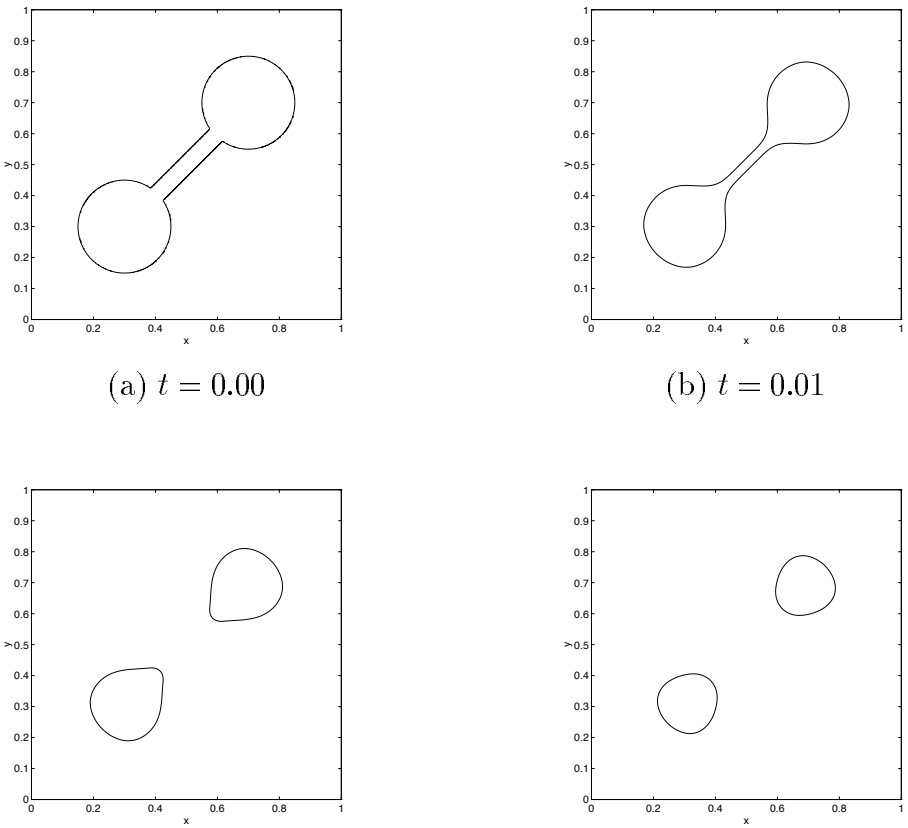
Table III. Errors for a general anisotropic motion.

As in previous examples, these results are suggestive of an approximately first order error in the position of the front.

8 Conclusion

We have developed the generalizations of the diffusion-generated motion first suggested in the original exposition [14], and in particular the algebraic-geometric duality for convolution generated motion and generalized Huygens' principles. We also used this framework to construct specific convolution kernels for generating the anisotropic curvature motions of the form $v_n = a(n) + b(n)\kappa$ in two dimensions, and by direct extension, for anisotropic constant motion plus isotropic mean curvature motion $v_n = a(n) + b\kappa$ (b constant) in higher dimensions. We implemented these numerically using adaptively refined fast Fourier transform techniques for evaluating the convolution, and validated the performance of the methods on problems that include topological change.

In current research, we are extending these new convolution generated motion techniques to perform anisotropic curvature-dependent motion of triple points and general networks of curves. Convolution generated motions for the *isotropic* case $v_n = a + b\kappa$ extend naturally to multiple junctions, just



(a) $t = 0.00$ (b) $t = 0.01$

(c) $t = 0.02$ (d) $t = 0.03$

Figure 14: A test problem at various times, t. The normal velocity is given by $v_n = a(\theta) + b(\theta)\kappa$ where $a(\theta) = -|\sin(\theta)| - |\cos(\theta)|$ and $b(\theta) = \frac{1}{10} + \frac{1}{20}\sin(2\theta)$.

as for the original diffusion-generated motion algorithm. But for anisotropic motions, there are more complicated conditions on the triple point equilibrium angles, and it remains to determine a suitable thresholding step that properly enforces these equilibrium angles as boundary conditions. This is a subject of ongoing research.

We are also developing multiple-kernel convolution generated motions, since these appear to provide a convenient way to generate interface velocities that are unobtainable with single kernels (e.g. anisotropic mean curvature motion in more than two dimensions). In this approach, the characteristic function is convolved with multiple kernels, $\chi * K_1, \chi * K_2, \ldots, \chi * K_N$, and these are combined in some convex combination or differencing combination prior to the thresholding stage. The focus of this research is on obtaining specific kernels for desired motions, and for determining how many kernels are necessary to generate various general classes of velocity laws in simple fashions.

Finally, we are working on developing the connection between convolution generated motion and cellular automata, both at the conceptual level and the computational level. Computationally, we are investigating the use of multiple-kernel motions to obtain spiral waves and steady state patterns seen in automata, but with much less computational effort and much greater accuracy in approximating the limiting interfaces.

9 Acknowledgments

Many thanks to Stan Osher for helpful discussions. We also thank Gregory Beylkin for helpful comments on fast convolution methods for piecewise constant functions.

A Numerical Considerations

The spatial discretization of the algorithm **Anisotropic** may be carried out in a variety of ways. A naive discretization using a pseudo-spectral method or using finite differences on a uniform grid typically is adequate for crude

but illustrative results. For fast, accurate results, the sharpening methods given in [25] may be used. We now outline the application of these methods to the algorithm **Anisotropic** for a square domain $D = [0,1] \times [0,1]$. For further details, see [25].

To begin, the characteristic function for the initial region, $\chi^0(\cdot)$, and the kernel, $K(\cdot)$, are approximated by Fourier tensor products,

$$\begin{aligned} \chi_n^0(\vec{x}) &= \sum_{-n \le j,k \le n-1} \hat{\chi}_{jk}^0 \exp(2\pi ijx) \exp(2\pi iky), \\ K_n(\vec{x}) &= \sum_{-n \le j,k \le n-1} \hat{K}_{jk} \exp(2\pi ijx) \exp(2\pi iky) \end{aligned}$$

using the algorithm **Sharpen**, defined below. Multiplying in Fourier space gives a simple estimate for the convolution product, $\varphi(\vec{x}) = \chi * K(\vec{x})$:

$$\varphi_n(\vec{x}) = \sum_{-n \le j,k \le n-1} \hat{\chi}_{jk}^0 \hat{K}_{jk} \exp(2\pi ijx) \exp(2\pi iky).$$

The Fourier representation of the characteristic function for the updated region,

$$\chi_n(\vec{x}) = \sum_{-n \le j,k \le n-1} \hat{\chi}_{jk} \exp(2\pi ijx) \exp(2\pi iky) \tag{17}$$

may now be determined using an adaptive quadrature method rather than a pseudo-spectral method (see Remark 1). Begin by defining

$$R = \{\vec{x} : \varphi(\vec{x}) > \lambda(\theta)\}$$

to be the approximation of the region we are following. By multiplying Eq. (17) by $\exp(-2\pi ijx)\exp(-2\pi iky)$, integrating over the domain and simplifying via the usual orthogonality conditions we find

$$\hat{\chi}_{jk} = \int_0^1 \int_0^1 \chi_n(\vec{x}) \exp(-2\pi ijx) \exp(-2\pi iky)\, dx\, dy. \tag{18}$$

But $\chi_n(\cdot)$ is an approximation of the characteristic function, $\chi(\cdot)$,

$$\chi(\vec{x}) = \begin{cases} 1 \text{ if } \vec{x} \in R \\ 0 \text{ otherwise} \end{cases}$$

so Eq. (18) may be re-written as

$$\hat{\chi}_{jk} \;=\; \int\!\!\int_R \exp(2\pi ijx)\exp(2\pi iky)\; dA.$$

Thus, $\chi_n(\cdot)$ may be obtained by integrating simple functions over a complicated, non-rectangular region, R. This task is accomplished by recursively subdividing the domain into small squares according to the following algorithm:

ALGORITHM **Sharpen**
GIVEN: A domain $D = [x_0, x_f] \times [y_0, y_f]$ and a function $\varphi_n(\vec{x}) = \chi_n^0 * K_n(\vec{x})$.

BEGIN
(1) Approximate the normal direction to the curve over D
using some approximation to $\nabla\varphi_n/|\nabla\varphi_n|$.
(2) Calculate λ using either Eq. (11) or Eq. (16).
(3) IF $\max(x_f - x_0,\, y_f - y_0) \leq$ MINIMUM_CELL_WIDTH
Approximate $D \cap R$ by 1 or 2 triangles using linear interpolation.
For each triangle, T, add a contribution $\int\!\int_T \exp(-2\pi ijx)\exp(-2\pi iky)\; dA$
to each Fourier coefficient, $\hat{\chi}_{jk}$.
RETURN
END IF
(4) IF $\varphi_n([x_0, y_0]) \geq \lambda$, $\varphi_n([x_0, y_f]) \geq \lambda$, $\varphi_n([x_f, y_0]) \geq \lambda$ and $\varphi_n([x_f, y_f]) \geq \lambda$
Assume $D \in R$ and add a contribution
$\int\!\int_D \exp(-2\pi ijx)\exp(-2\pi iky)\; dA$ to each $\hat{\chi}_{jk}$.
ELSE IF $\varphi_n([x_0, y_0]) \leq \lambda$, $\varphi_n([x_0, y_f]) \leq \lambda$, $\varphi_n([x_f, y_0]) \leq \lambda$ and $\varphi_n([x_f, y_f]) \leq \lambda$
Assume $D \cap R = \emptyset$. No contribution to the Fourier coefficients is made.
ELSE
Divide D into quadrants and **Sharpen** each quadrant.
END IF
END

Remarks (See [25] for further details.):

1. If a pseudo-spectral method is used to discretize the algorithm **Anisotropic**, then an $\mathcal{O}(1/n)$ error in the position of the front is generated at each

time step. Reducing these errors to an acceptable level is often impractical because such a large number of basis functions are required. Far fewer basis functions are needed using **Sharpen** because this approach gives a more accurate representation of the front. For example, the errors shown in Table III were obtained using $n = 64$. Essentially the same results were obtained using larger values of n.

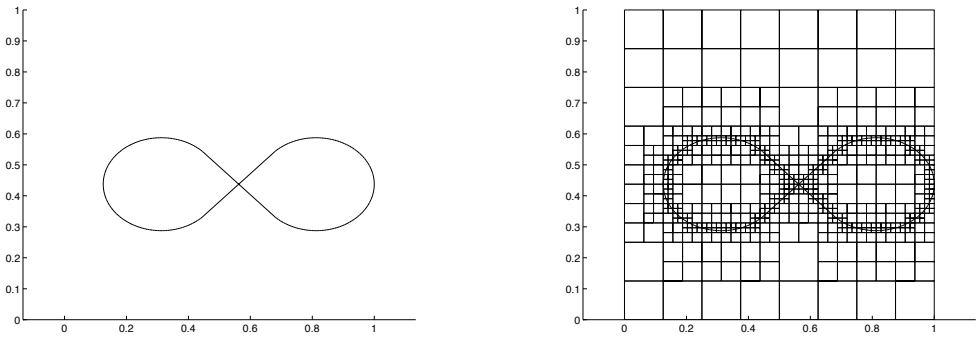
2. In step (1), the normal direction may be approximated using either centered differences, one-sided differences (e.g., [27]) or by differentiating the Fourier series $\varphi_n(\cdot)$. In our simulations, second order centered differences are preferred because they are simple, computationally inexpensive and produce errors which are comparable to other, more involved methods.

3. To capture the large-scale features of the shape, **Sharpen** should be applied to a number of equally-spaced subregions of the domain rather than to the domain itself. Typically, $2n \times 2n$ subregions are selected because the corresponding φ-values can be rapidly evaluated using a fast Fourier transform.

4. Small pieces of certain shapes are occasionally neglected by **Sharpen**. To capture the entire interface at the level of the finest grid, a gradual refinement can be used (e.g, Figure 15). This method proceeds according to the original algorithm, with the following additional consideration:

   > Whenever any cell is refined, check the subdivision level of the neighboring cells. Subdivide neighbors which are two or more levels of refinement coarser.

5. During mesh refinement, a large number of function evaluations are required. Because these occur on an unequally spaced grid (see, e.g., Figure 15), a fast Fourier transform cannot be used. Direct evaluation is often prohibitively expensive, however, since $\mathcal{O}(n^2 N_p)$ operations are required to evaluate $\varphi(\cdot)$ at N_p points. For these reasons, our implementations use a recent unequally spaced fast Fourier transform method [2].



(a) Region of interest. (b) A refined mesh.

Figure 15: A gradual refinement starting from a coarsest mesh.

This approach carries out N_p evaluations of $\varphi(\cdot)$ in

$$\mathcal{O}\left(N_p \log^2\left(\frac{1}{\epsilon}\right) + n^2 \log(n)\right)$$

operations where ϵ is the precision of the computation. Unequally spaced transform methods are also used for the rapid evaluation of the Fourier sums that arise in steps (3) and (4) of **Sharpen**.

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] G. Beylkin. On the fast Fourier transform of functions with singularities. *Applied and Computational Harmonic Analysis*, 2:363–381, 1995.

[3] L. Bronsard and Kohn. Motion by mean curvature as the singular limit of Ginzburg-Landau dynamics. *Journal of Differential Equations*, 90(2):211–237, 1991.

[4] L. Bronsard and F. Reitich. On three-phase boundary motion and the singular limit of a vector-valued Ginzburg-Landau equation. *Arch. Rat. Mech.*, 124:355–379, 1993.

[5] L. Bronsard and B. Stoth. Volume preserving mean curvature flow as a limit of a nonlocal Ginzburg-Landau equation. Technical Report 94-NA-008, Centre for Nonlinear Analysis, Carnegie Mellon University, 1994.

[6] G. Caginalp. The dynamics of a conserved phase field system: Stefan-like, Hele-Shaw, and Cahn-Hilliard models as asymptotic limits. *IMA J. Applied Math.*, 44(1):77–94, 1990.

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

[8] L.C. Evans, H.M. Soner, and P.E. Souganidis. Phase transitions and generalized motion by mean curvature. *Communications on Pure and Applied Mathematics*, 45(9):1097–1123, 1992.

[9] J. Gravner and D. Griffeath. Threshold growth dynamics. *Transactions of the American Mathematical Society*, 340(2):837–870, 1993.

[10] 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.

[11] H. Ishii, G.E. Pires, and P.E. Souganidis. Threshold dynamics type schemes for propagating fronts. *Preprint*, 1997.

[12] B.J. MacLennan. Continuous spatial automata. Technical Report CS-90-121, University of Tennessee, Dept. of Computer Science, Knoxville, 1990. Web Address: http://www.cs.utk.edu/~mclennan/fieldcomp-biblio.html.

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

[14] 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.

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

[16] S. Osher and B. Merriman. The Wulff shape as the asymptotic limit of a growing crystalline interface. *The Asian Journal of Mathematics*, 1(3), September 1997.

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

[18] P. Pelce, editor. *Dynamics of Curved Fronts*. Academic Press, Boston, 1988.

[19] J. Rubinstein and P. Sternberg. Nonlocal reaction-diffusion equations and nucleation. *IMA J. Appl. Math.*, 48:248–264, 1992.

[20] J. Rubinstein, P. Sternberg, and J.B. Keller. Fast reaction, slow diffusion and curve shortening. *SIAM J. Appl. Math.*, 49(1):116–133, 1989.

[21] J. Rubinstein, P. Sternberg, and J.B. Keller. Reaction-diffusion processes and evolution to harmonic maps. *SIAM J. Appl. Math.*, 49(6):1722–1733, 1989.

[22] S.J. Ruuth. An algorithm for generating motion by mean curvature. In *Proc. 12th International Conference on Analysis and Optimization of Systems Images, Wavelets and PDE's*, pages 82–91, Paris, France, 1996.

[23] S.J. Ruuth. *Efficient Algorithms for Diffusion-Generated Motion by Mean Curvature*. PhD thesis, University of British Columbia, Vancouver, Canada, 1996.

[24] S.J. Ruuth. A diffusion-generated approach to multiphase motion. CAM Report 97-28, University of California, Los Angeles, 1997.

[25] S.J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. CAM Report 97-33, University of California, Los Angeles, 1997.

[26] S.J. Ruuth. A diffusion-generated approach to a nonlocal, volume-preserving motion. *In Preparation*, 1998.

[27] J.A. Sethian and J.D. Strain. Crystal growth and dendritic solidification. *J. Comput. Phys.*, 98:231–253, 1992.

[28] J. Soulettie, J. Vannimenus, and R. Stora, editors. *Chance and Matter*. Elsevier Publishing Co., North-Holland, 1987. See the article of Langer, in particular.

[29] J. Yao and D.S. Stewart. On the dynamics of multi-dimensional detonation. *Journal of Fluid Mechanics*, 309:225–275, 1996.

[30] H. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127:179–195, 1996.