Journal of Computational Physics **151,** 836–861 (1999)
Article ID jcph.1999.6223, available online at http://www.idealibrary.com on IDEAL®

Convolution-Generated Motion as a Link between Cellular Automata and Continuum Pattern Dynamics

Steven J. Ruuth,[1] Barry Merriman,[2] and Stanley Osher[3]

University of California, Department of Mathematics, 405 Hilgard Avenue, Los Angeles, California 90095-1555

Received October 2, 1998; revised February 3, 1999

Cellular automata have been used to model the formation and dynamics of patterns in a variety of chemical, biological, and ecological systems. However, for patterns in which sharp interfaces form and propagate, automata simulations can exhibit undesirable properties, including spurious anisotropy and poor representation of interface curvature effects. These simulations are also prohibitively slow when high accuracy is required, even in two dimensions. Also, the highly discrete nature of automata models makes theoretical analysis difficult. In this paper, we present a method for generating interface motions that is similar to the threshold dynamics type cellular automata, but based on continuous convolutions rather than discrete sums. These convolution-generated motions naturally achieve the fine-grid limit of the corresponding automata, and they are also well suited to numerical and theoretical analysis. Because of this, the desired pattern dynamics can be computed accurately and efficiently using adaptive resolution and fast Fourier transform techniques, and for a large class of convolutions the limiting interface motion laws can be derived analytically. Thus convolution-generated motion provides a numerically and analytically tractable link between cellular automata models and the smooth features of pattern dynamics. This is useful both as a means of describing the continuum limits of automata and as an independent foundation for expressing models for pattern dynamics. In this latter role, it also has a number of benefits over the traditional reaction–diffusion/Ginzburg–Landau continuum PDE models of pattern formation, which yield true moving interfaces only as singular limits. We illustrate the power of this approach with convolution-generated motion models for pattern dynamics in developmental biology and excitable media.

[1] This research was partially supported by an NSERC Postdoctoral Scholarship, NSF DMS-9706827, and ONR N00014-97-1-0027.

[2] This research was partially supported by NSF DMS-9706827 and NSF DMS-9615854.

[3] This research was partially supported by NSF DMS-9706827 and NSF DMS-9615854.

0021-9991/99 $30.00

Key Words: convolution; cellular automata; threshold growth; excitable media; interface; motion by mean curvature; spectral method.

1. INTRODUCTION

There are many natural and industrial problems in which sharp interfaces form and propagate. Notable examples include the growth of crystalline materials, the processing and enhancement of images, the evolution of ecological systems, and the waves of excitation that occur in heart and neural tissue.

To model these and other complex phenomena, a wide variety of cellular automata have been used [5, 12, 27, 36]. A particularly fundamental class is the threshold dynamics type discussed in Section 2. Briefly, in these models, the spatial domain is divided into a fixed lattice and each lattice point is assigned one of a finite number of states. The state of each cell is updated by forming a weighted sum over a neighborhood and thresholding in some fashion.

Unfortunately, several difficulties arise when these models are used as the basis for numerical simulations. As we shall see, these automata give a crude approximation of the front and a discontinuous approximation to the motion. This leads to a number of qualitative problems, including an inadequate treatment (or complete absence) of curvature effects and unwanted anisotropies. Automata-based discretizations also have the disadvantages that they may not give a practical means for evaluating quantities defined on the interface or for independently varying parameters defined in the model. Furthermore, whenever accurate approximations of the limiting motion are sought, these simple lattice-based techniques become prohibitively slow, even for simple problems in two dimensions.

In this present work, we consider threshold dynamics type schemes which are based on convolutions rather than discrete sums. These convolution-generated motions naturally give the limiting form of automata as the mesh spacing goes to zero and can be discretized efficiently and accurately using adaptive grid refinement with fast Fourier transforms. These methods are particularly well suited to modeling continua because they accurately approximate the continuum limit (in space) and thus eliminate the qualitative grid effects that are problematic in automata-based discretizations. Numerical experiments for convolution-based models arising in developmental biology and excitable media are used to validate the performance of our methods.

The outline of the paper follows. Section 2 reviews threshold dynamics models for interface motion. In Section 3, we review convolution-generated motion and give an overview of the class of obtainable motion laws. Section 4 discusses the numerical approximation of convolution-generated motion and shows that discretizations based on fast Fourier transforms on adaptively refined grids are faster and more accurate than direct or pseudospectral approximations. In Section 5, we apply our proposed discretization to a model arising in developmental biology and compare our results to those from the usual automaton. Section 6 applies our proposed discretization to a convolution-based model for excitable media and compares the result to automata-based methods. This also indicates the computational advantages of convolution-generated motion over reaction–diffusion/Ginzburg–Landau PDE models for pattern dynamics. In Section 7, we summarize our results and outline current research. Finally, the Appendix concludes with a specification of the convolution-based model for excitable media that was used in Section 6.

2. CELLULAR AUTOMATA MODELS

Cellular automata are discrete dynamical systems. They consist of a lattice of sites, each of which may take on a finite number of "states," or values. The site values evolve in synchronous, discrete time steps according to an evolution rule that specifies the updated value in terms of the current values at neighboring sites [35].

We will focus on a particularly important class of automata and discuss some of their limitations as a modeling and simulation tool.

2.1. *Threshold Dynamics*

An important class of automata can be obtained by imagining each neighbor's contribution to be a simple "vote" for or against a certain outcome; any number of affirmative votes above a certain threshold will yield that outcome. For example, consider the voting automaton displayed in Fig. 1. Here, there are two states, 1 and 0. A sum of the cell's own vote and that of its eight nearest neighbors is formed (see Fig. 1b). Where this sum is greater than or equal to the threshold value (which in this case is 3) the cell is assigned state 1, and state 0 elsewhere. By denoting the state of cell (j, k) at time step n by C_{jk}^n, we obtain a simple

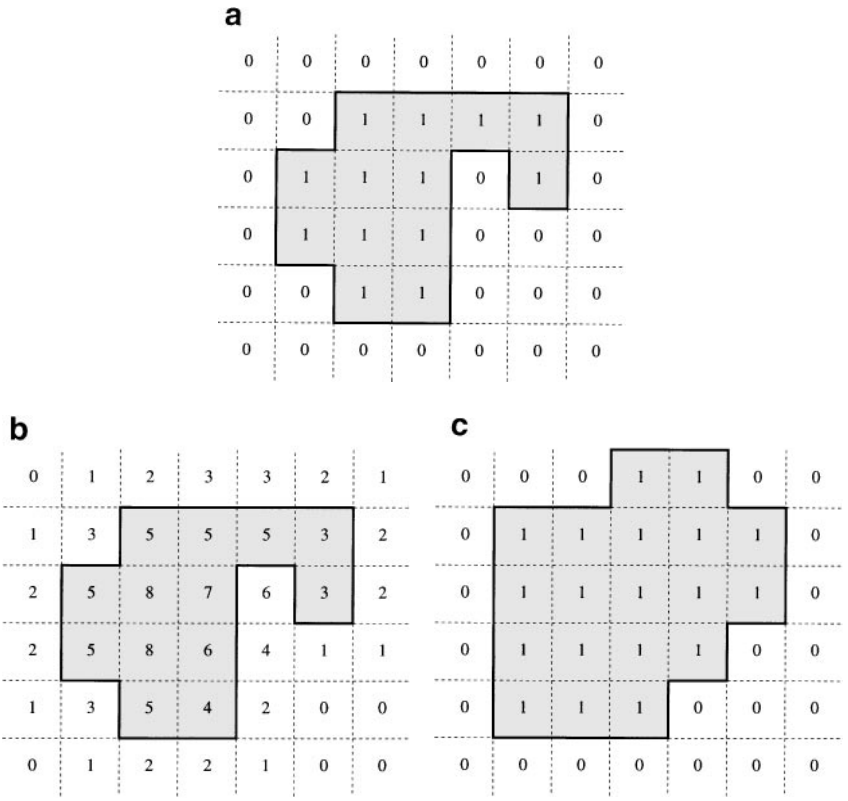

FIG. 1. One step of a simple automaton consists of summing over the nearest neighbors and thresholding. In this example, we sum over a nine-point neighborhood and take the updated set to be all points which have a sum of 3 or more. (a) Original region. (b) Sum of neighbors. (c) Final region.

analytic representation for the automata model,

$$C_{jk}^{n+1} = \begin{cases} 1 & \text{if } \sum_{-1\le j',k'\le 1} C_{j-j',k-k'}^{n} \ge \lambda \\ 0 & \text{otherwise,} \end{cases}$$

where λ represents the threshold value.

More generally, each vote can be assigned some weight. Letting $N \subset Z^2$ be the neighborhood of interest and W be the matrix of weights, we obtain the update rule for threshold dynamics,

$$C_{jk}^{n+1} = \begin{cases} 1 & \text{if } \sum_{j',k'\in N} W_{j',k'} C_{j-j',k-k'}^{n} \ge \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

2.2. *Threshold Dynamics as a Method for Interface Motion*

In order to model a desired front motion using the update rule (1), an appropriate neighborhood and weight values must be chosen.

In early cellular automata, a neighborhood of nearest neighbors was selected. This choice has the advantages of speed and simplicity but is inadequate for modeling many interesting natural phenomena. In particular, rules which use these small neighborhoods are unable to model the effects of curvature on the speed of propagation [13, 33]. These simple automata also add grid-based anisotropy to the front motion [25]. This effect is easily seen for a 2D regular lattice since a front oriented at 45° from the horizontal travels at $\sqrt{2}$ times the speed of a vertical or horizontal front.

In an attempt to reduce grid effects and to obtain an approximation to the curvature contribution of the motion, several authors have considered refining the lattice and taking larger neighborhoods [11, 13, 30, 31, 10, 8, 9, 7, 16]. In carrying out this procedure, several interesting questions arise. In particular:

1. What is the limiting motion of the automata model as the mesh is refined and the neighborhoods increased?
2. How well does a particular automaton approximate its continuum limit? In particular, how much refinement is required to faithfully reproduce curvature effects and to eliminate unwanted grid effects?
3. If it is impractical to approximate the limiting motion using an automaton, how should the limiting motion be computed?

We now review recent results which provide a partial answer to the first of these questions. Answers to the remaining questions will be the focus of the remainder of the paper.

3. CONVOLUTION-GENERATED MOTION

The procedure of forming a weighted sum over some neighborhood, followed by thresholding at some value, is reminiscent of convolution-generated motion [24]. This is a procedure for defining a time-discrete evolution of an interface that is the boundary of a region, as follows: Given an initial region Ω in R^d, let

$$\chi\colon R^d \to R$$

be the characteristic function of Ω, i.e.,

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

We define the updated region Ω^{new} to be the set

$$\Omega^{\text{new}} = \{\mathbf{x}: \chi * K(\mathbf{x}) > \lambda\}$$

for some threshold value $\lambda \in R$ and some convolution kernel function, $K: R^d \to R$, where $*$ denotes the convolution

$$\chi * K(\mathbf{x}) = \int_{R^d} K(\mathbf{x} - \mathbf{y}) \chi(\mathbf{y})\, d\mathbf{y}.$$

Or, in terms of characteristic functions, we define the updated characteristic function $\chi^{\text{new}}(\mathbf{x})$ by

$$\chi^{\text{new}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \chi * K(\mathbf{x}) > \lambda \\ 0 & \text{if } \chi * K(\mathbf{x}) \le \lambda. \end{cases}$$

Convolution-generated motion can be used to produce a great variety of interface motions, which are amenable to both theoretical analysis and computation. On the theoretical side, the resulting analytic velocity of the interface (in the proper continuous time limit) can be obtained explicitly for a large class of convolution kernels, and general convolution-generated motions can also be described in terms of a geometric Huygens' principle. On the computational side, accurate and efficient discretizations can be designed for computing convolution-generated motions. These aspects of the method are described in more detail in the following sections.

First, we recall a special case of convolution-generated motion which produces a classical "motion by mean curvature." This simple algorithm highlights the elegance and power of the convolution formulation. Also, the discovery and investigation of this particular algorithm was our original motivation for considering convolutions as a means of generating interface motion.

3.1. *Diffusion-Generated Motion by Mean Curvature*

A particularly simple convolution-based algorithm exists for moving an interface with a normal speed equal to its mean curvature [18, 19]. If the initial region has characteristic function χ, the updated region at a time Δt is

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

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

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

"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 [17–19] shows that this should result in precisely motion by mean curvature. Indeed, a variety of rigorous proofs have been given to show that this simple algorithm converges to motion by mean curvature in any number of dimensions as the time step size goes to zero [6, 2, 15]. Note also that this procedure has a direct extension to the motion of multiple regions, i.e., interfaces which intersect to form triple points or other multiple junctions. See [18, 17, 19, 22] for details.

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 [18] and proven rigorously in [14]. 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,

$$K(\mathbf{x}) = \begin{cases} \frac{1}{\pi r^2} & \text{if } |\mathbf{x}| < r \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

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

3.2. *General Convolution-Generated Motion*

Based on the update rule (1), it is clear that we are interested in more general forms of convolution-generated motion. In particular, it is natural to consider the following generalizations of (2):

1. Allow different convolution kernel functions, K. The method formally allows arbitrary kernel functions, and asymmetrical kernels can be used to produce anisotropic motion laws, as originally suggested in [18]. Without loss of generality, we shall assume that the kernel has been normalized to satisfy $\int K(\mathbf{x})\, d\mathbf{x} = 1$.

2. Allow a general threshold, λ, in $\{\mathbf{x}\colon \chi * K(\mathbf{x}) > \lambda\}$. This provides a continuum of convolution-generated methods parameterized by $\lambda \in [0, 1)$ with $\lambda = 0$ corresponding to the standard Huygens' principle for constant motion (see [24]), 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 $\lambda = \frac{1}{2} + c\sqrt{\Delta t}$ [15, 17, 22], so $\lambda = \lambda(\Delta t)$ is a useful form. More generally, λ may be selected locally as a function of the normal direction defined by the level sets of $K * \chi$ to achieve an interesting variety of anisotropic motions [24].

These generalizations produce a semi-discrete version of threshold dynamics—i.e., continuous in space but discrete in "time." To obtain an approximation of the continuous dynamics (assuming such a limit exists), we must somehow introduce a time step and clarify what it means to take the small time step limit. Intuitively, the effective time step is determined by the size of the support of K, since the larger the support of K, the further its convolution will move the set boundary. Thus the small time step limit is obtained by scaling down the support of K in a suitable fashion. More precisely, let us scale the fixed kernel $K(\mathbf{x})$ by the mass preserving form $K(\mathbf{x}/r)/r^d$, so that the typical radius of its support scales like $r \ll 1$. By convolving this scaled kernel with χ and thresholding the result at λ, the set boundary is 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 time step of some limiting motion

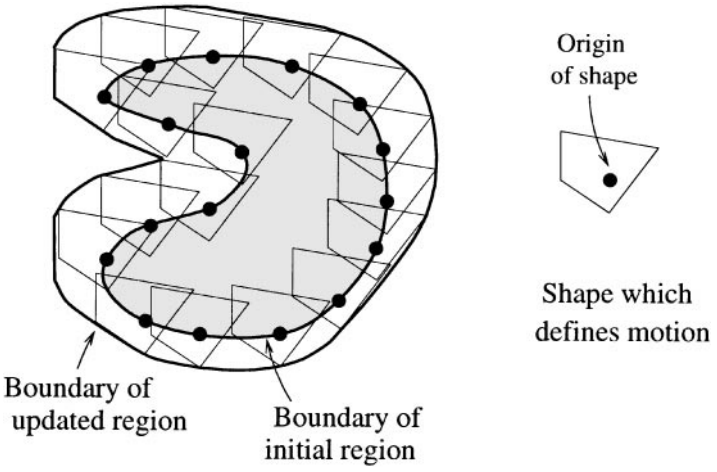

FIG. 2. When K is non-negative and $\lambda = 0$, Huygens' principle for constant motion is produced.

law, $s(r) = v_n \Delta t$, this fixes the relation between the size of the kernel, r, and the effective time step, Δt. Note, in particular, that if K is a Gaussian kernel with support of size r, this general procedure yields $\Delta t \sim r^2$. This is precisely the scaling relation between kernel size and time step used in the diffusion-generated case discussed above, although there it can also be motivated by the simple fact that diffusion for a time Δt will smear (and thus move) the set boundary over a distance $r \sim \sqrt{\Delta t}$.

3.3. *Obtainable Motion Laws*

It is natural to ask what motion laws arise from convolution-generated motion and how the radius of the kernel scales with Δt.

In the case where K is non-negative and $\lambda = 0$, convolution-generated motion reduces to Huygens' principle for "constant motion." Specifically, the updated set is given *exactly* as follows (see Fig. 2):

> Set $\mathcal{N}$ equal to the shape consisting of all points where K is non-zero. Using only translations, place copies of $\mathcal{N}$ so that its origin is positioned at the boundary of the initial set. The forward envelope of shapes forms the boundary of the updated set.

If we assume that each update corresponds to one time step of length Δt, then $\mathcal{N}$ and hence K have radii which scale like Δt.

Another interesting case occurs in two dimensions when $\lambda = 1/2$ and K is the scaled characteristic function of a symmetric region $\mathcal{N}$ (i.e., $\mathcal{N} = -\mathcal{N}$). If we define $r(\theta)$ to be the polar representation of the boundary of $\mathcal{N}$, then it is easy to show that a leading order approximation of the displacement of a smooth initial boundary is $r^2(\theta)\kappa/6$ [24]. Thus general "anisotropic curvature motions" of the form

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

are obtained simply by taking

$$r(\theta) = \sqrt{6b(\theta)\Delta t}.$$

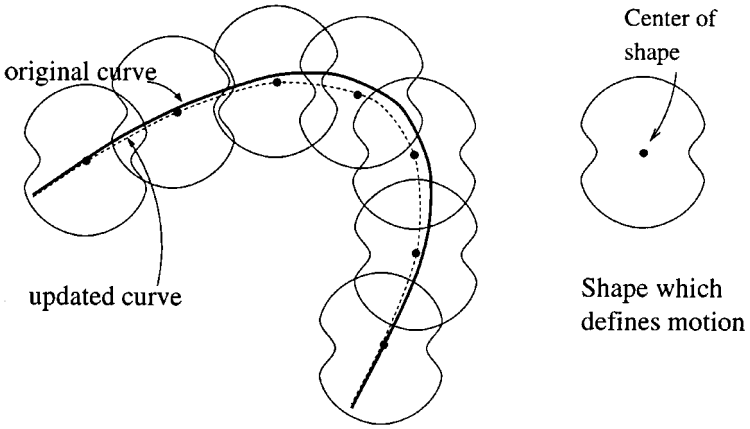

FIG. 3. When $\mathcal{N}$ is symmetric and $\lambda = 1/2$, a geometric algorithm for anisotropic curvature motion is produced by placing shapes $1/2$ inside the original region by area. The boundary of the updated region is given by the locus of shape centers.

Similar to the case of constant motion, this algorithm also has a simple geometric version [24] (see Fig. 3):

> Using only translations, place copies of $\mathcal{N}$ so that exactly half of their area lies inside the original region. The locus of shape centers forms the boundary of the updated set.

A combination of these two types of motion can be obtained by varying the threshold, λ. This class of methods has been studied in the recent and comprehensive work of Ishii, *et al.* [15] for the case where λ is 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 $\Delta t \to 0$.

To produce more general motions, λ may be allowed to depend on other quantities. For example, in [24] λ is defined locally as a function of the normal direction to obtain motions of the form

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

where b is non-negative and continuous. Nonlocal choices for λ also produce interesting flows. For example, volume preserving motion by mean curvature [4, 20], i.e., $v_n = \kappa - \bar{\kappa}$, where $\bar{\kappa}$ is the surface average of the mean curvature, is realized by selecting the level that encloses the same volume as the original set in diffusion-generated motion, instead of the $\frac{1}{2}$ level [21]. Convergence of these last two procedures has been demonstrated numerically, but not proven analytically.

4. NUMERICAL APPROXIMATION

We now discuss the numerical approximation of convolution-generated motion. For illustrative purposes, our examples will focus on the fundamental, but simple, case of anisotropic curvature-dependent motion (4). Subsequent sections will build upon these results to approximate models arising in developmental biology and excitable media.

4.1. *Overview*

Perhaps the most obvious method for approximating convolution-generated motion is with threshold dynamics. If small neighborhoods are used, however, these automata cannot model the effects of curvature on the speed of propagation. They also add unwanted anisotropy to the front motion. Thus, large neighborhoods typically must be used if even a qualitative agreement with the model is sought.

Since large neighborhoods are inefficient to treat using a direct evaluation of the automata rule (1), it is natural to evaluate the sum pseudospectrally. Alternatively, the summation step may be approximated by a number of one-dimensional convolutions [13, 30, 31, 10, 8, 9]. This approach is rather limited, however, because it accommodates only a very small class of sums.

4.2. *Pseudospectral Approximation*

An obvious way of approximating convolution-generated motion is with a regular lattice of grid points, $\{\mathbf{x}_{jk}\}$. Using this approach, functions are represented by their values at grid points. This makes the thresholding step trivial since it can be carried out pointwise. The convolution step is also straightforward since it reduces to a summation (1) which can be treated efficiently in Fourier space using fast Fourier transform (FFT) methods.

Thus a simple (but naive) discretization for convolution-generated motion is as follows:

ALGORITHM RegularLattice.

GIVEN: A kernel, K, an initial region, R, and a regular lattice of points, $\{\mathbf{x}_{jk}\}$.
BEGIN
(1) Set $C_{jk} = \begin{cases} 1 & \text{if } \mathbf{x}_{jk} \in R \\ 0 & \text{otherwise} \end{cases}$
(2) Set $[\hat{K}_{jk}]$ equal to the discrete Fourier transform of $[K(\mathbf{x}_{jk})]$.
(3) REPEAT for all steps:
BEGIN
(a) Set $[\hat{C}_{jk}]$ equal to the discrete Fourier transform of $[C_{jk}]$.
(b) Convolve by replacing each $\hat{C}_{jk}$ with $\hat{C}_{jk}\hat{K}_{jk}$.
(c) Set $[C_{jk}]$ equal to the inverse Fourier transform of $[\hat{C}_{jk}]$.
(d) Threshold each C_{jk} by setting $C_{jk} = \begin{cases} 1 & \text{if } C_{jk} > \lambda \\ 0 & \text{otherwise.} \end{cases}$
END
END

Unfortunately, the algorithm RegularLattice as well as other grid-based techniques [13, 30, 31, 10, 8, 9, 7,16] have several deficiencies. These include:

1. The convolution step must propagate the level set λ at least one grid point, otherwise the thresholding step will keep the front stationary [19]. Thus, the grid spacing must be much less than the speed of propagation of the front times the time step size:

$$\Delta x \ll v_n \Delta t. \tag{5}$$

Note that this requirement can be especially severe for the fundamental case of curvature motion since it applies even if the local curvature of the interface is small. See Fig. 4 for an illustration of a shape which has "frozen" in place because the mesh spacing is too large.

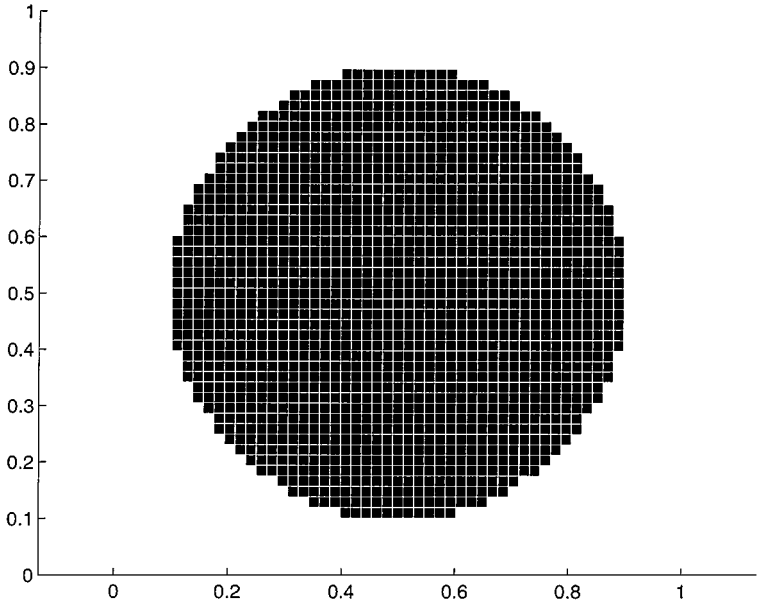

FIG. 4. The motion cannot be resolved by the algorithm RegularLattice and the interface remains stationary. Here, diffusion-generated motion is used with a time step of $\Delta t = 0.001$ and a mesh spacing of $\Delta x = 1/54$.

2. The underlying lattice will contribute an unwanted anisotropy to the motion since the front must travel an integer number of cells per time step in the local normal direction. This grid-based anisotropy is often apparent even when large neighborhoods are used (see, e.g., Fig. 5 of [31]). To reduce these effects, Markus and Hess [16] considered a randomized lattice. Unfortunately, this approach does not provide a means of generating a desired anisotropy. It also has the disadvantage that the summation step is carried out directly, since simple transform methods cannot be used on an irregular lattice. See [25] for a study of randomized rules for reducing grid-based anisotropy.

3. In many convolution-based algorithms, Richardson extrapolation in the time step size can be used to obtain an accelerated convergence to the limiting motion law [23]. Using a

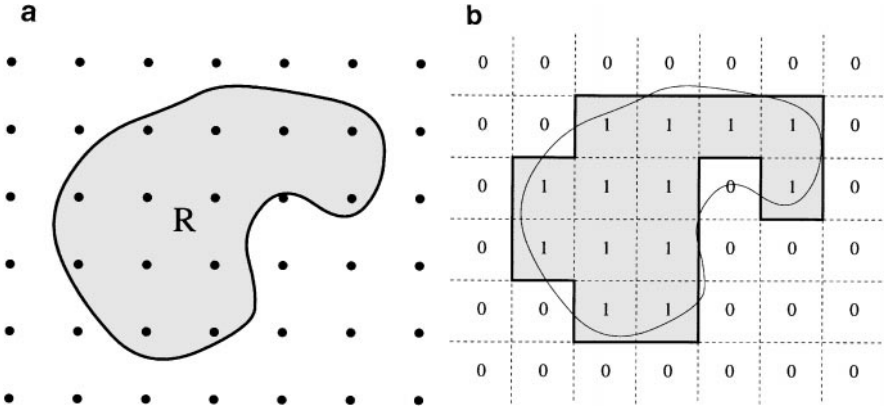

FIG. 5. An automata representation of the original region, R, using a uniform mesh. (a) Original region. (b) CA representation.

lattice-based discretization, however, the front must travel an integer number of cells per time step. Thus, very small differences in the level set λ can produce jumps (of size Δx) in the front location after thresholding. This type of irregular error is undesirable because it makes the construction of higher order accurate, extrapolated results impractical [23].

4. A further difficulty with the lattice-based approach is that each thresholding step produces an $O(\Delta x)$ displacement in the front position (see, e.g., Fig. 5). To reduce this error to an acceptable level often necessitates a prohibitively large number of grid points whenever an accurate representation of the limiting motion is desired. For example, consider a diffusion-generated approximation to curvature motion. Here, each step of the algorithm produces an $O(\Delta t^2)$ error in the position of the front [21]. To preserve this overall accuracy, grid points must be a distance $O(\Delta t^2)$ apart since each thresholding step produces an error which is comparable to the mesh spacing. Thus, $\Omega(1/\Delta t^4)$ grid points[4] (and $\Omega(1/\Delta t^4)$ operations per step) are required whenever a uniform mesh is used. Even with a banded method, $\Omega(1/\Delta t^3)$ operations per step are needed [23], which can be impractical when accurate results are sought.

In many problems, these limitations can be overcome by using discretizations based on fast Fourier transforms on adaptively refined grids [23]. A discussion of these methods follows.

4.3. *Convolution-Generated Motion Algorithm*

As we saw in the previous section, a pointwise thresholding displaces the front a distance which is comparable to the mesh spacing. To eliminate this effect, and to obtain a more efficient computation, an improved approximation to the thresholding step must be derived.

It is easily seen that the Fourier coefficients of the characteristic function for a region R are given *exactly* by

$$\hat{\chi}_{jk} = \iint_R \exp(-2\pi i j x) \exp(-2\pi i k y)\, dA$$

for integers j and k. Using approximations of these integrals to derive the Fourier coefficients gives a greatly improved discretization of convolution-generated motion [23, 24]:

ALGORITHM CG-Motion.

GIVEN: A kernel, K, and an initial region, R.
BEGIN
(1) Set $\hat{\chi}_{jk} = \iint_R \exp(-2\pi i j x) \exp(-2\pi i k y)\, dA$, $-N \le j, k \le N$.
(2) Set $\hat{K}_{jk} = \iint K(x, y) \exp(-2\pi i j x) \exp(-2\pi i k y)\, dA$, $-N \le j, k \le N$.
(3) REPEAT for all steps:
BEGIN
(a) Convolve by replacing each $\hat{\chi}_{jk}$ with $\hat{\chi}_{jk}\hat{K}_{jk}$.
(b) Threshold by setting $\hat{\chi}_{jk} = \iint_{R(t)} \exp(-2\pi i j x) \exp(-2\pi i k y)\, dA$, $-N \le j, k \le N$, where $R(t) = \{(x, y) : \sum_{-N \le j,k \le N} \hat{\chi}_{jk} \exp(-2\pi i j x) \exp(-2\pi i k y) \ge \lambda\}$.
END
END

[4] $F(n) = \Omega(f(n))$ if there exist positive constants c and n_0 such that $F(n) \ge c f(n)$ whenever $n \ge n_0$.

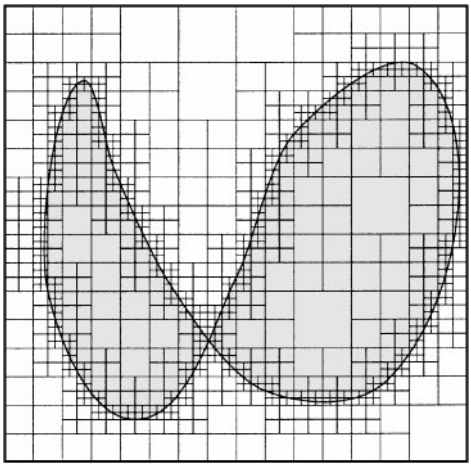

FIG. 6. Integration is carried out by dividing the domain into squares. Contributions from all but the finest regions can be evaluated exactly.

Note that the convolution step acts as a filter, removing high-frequency components. Since this convolution step is linear, the different Fourier modes do not interact and there is never a need to treat the highest frequency components. Thus, an excellent approximation is obtained using fewer Fourier modes than might otherwise be expected. Indeed, a very high degree of accuracy is possible for the fundamental case of a Gaussian kernel since the neglected terms are all exponentially small whenever $N \gg 1/\sqrt{2\pi\,\Delta t}$.

To complete the discretization, the integrals arising in the algorithm CG-Motion must be evaluated. These are accurately and efficiently treated using the quadrature methods described in [23, 24]. Briefly,

- If $R(t)$ is a square, the integration step is carried out exactly. More general regions are treated by dividing the domain into small squares (see, e.g., Fig. 6) and summing the contributions from each. At the finest level, the contributions to the Fourier coefficients are approximated using a quadrature over triangles.
- During mesh refinement, a large number of unequally spaced function evaluations are required (see, e.g., Fig. 6). Because the FFT requires an equally spaced grid, our implementations use a recent unequally spaced fast Fourier transform method [3]. This method is also used for the rapid evaluation of the Fourier sums that arise in the quadrature steps of the algorithm.

We now consider how our proposed discretization compares with grid-based methods.

4.4. *Comparison to Lattice-Based Methods*

There are several reasons why the algorithm CG-Motion is preferred over automata-based discretizations of convolution-generated motion. These reasons are outlined below.

1. A lattice-based method must satisfy (5) globally, or part of the front may erroneously remain stationary. By recursively refining near the interface and interpolating at the finest cell level, our approach eliminates this restriction.

2. An unwanted anisotropic component to the motion is generated whenever a regular lattice is used since the front must travel an integer number of cells per time step. No such restriction occurs with the algorithm CG-Motion since interpolation is used to locate the front at the finest cell level. Thus, the interpolation step eliminates unwanted anisotropy and naturally allows for desired anisotropies through the use of appropriate kernels.

3. A lattice-based method produces an irregular error which makes the construction of higher order accurate, extrapolated results impractical. Because the algorithm CG-Motion uses interpolation to locate the front at the finest cell level, the error arising from the thresholding step is relatively small. In many instances, this makes an accelerated convergence to the limiting motion law possible using Richardson extrapolation in the time step size. See [23] for further details.

4. Far fewer operations are required to obtain an accurate representation of the front using the algorithm CG-Motion. Consider, for example, the fundamental case of diffusion-generated motion in two dimensions. Here, the proposed method requires only

$$O\left(\frac{1}{\Delta t}\log^2(\Delta t)\right)$$

operations per step to preserve the overall accuracy of the method [23]. This compares very favorably to the idealized finite difference result for smooth curves $O(1/\Delta t^3)$ which was derived in [23].

The last advantage, in particular, makes it practical to determine the limiting motion of automata which are impractical to obtain using a lattice-based approach.

Consider, for example, the case $\lambda = 1/2$ with a kernel which is the scaled characteristic function of a square. Based on the considerations of Section 3.2, we know that a well-defined limiting motion is produced if the width of the kernel scales like $\sqrt{\Delta t}$. For this reason, we take our test kernel to be the scaled characteristic function for the region bounded by

$$r(\theta) = \sqrt{\Delta t}/\max(|\sin(\theta)|, |\cos(\theta)|)$$

in polar coordinates.

To test the performance of the algorithm RegularLattice for this family of kernel functions, we compute the area change for an initially square region (see Fig. 7) using a time step Δt and $2N \times 2N$ basis functions. Each solution, $A_N^{\Delta t}$, is then compared with the semi-discrete result, $A^{\Delta t}$ (i.e., the limit as $N \to \infty$ with a fixed $\Delta t > 0$), and the continuous result, A. Based on the corresponding errors given in Table I, it is clear that the lattice-based approach becomes impractical whenever accurate solutions are required. (The value of N is selected to be the smallest power of 2 which satisfies the constraint $|A^{\Delta t} - A_N^{\Delta t}| \leq |A^{\Delta t} - A|$.)

TABLE I
Errors and CPU Times for the Algorithm RegularLattice

Δt	N	Relative error (semi-discrete)	Relative error (limiting motion)	CPU
1/192	128	3.1%	7.1%	34 s
1/384	1024	1.3%	3.6%	6848 s

Note. All computations reported were carried out on a 50-MHz SPARC Sun-4.

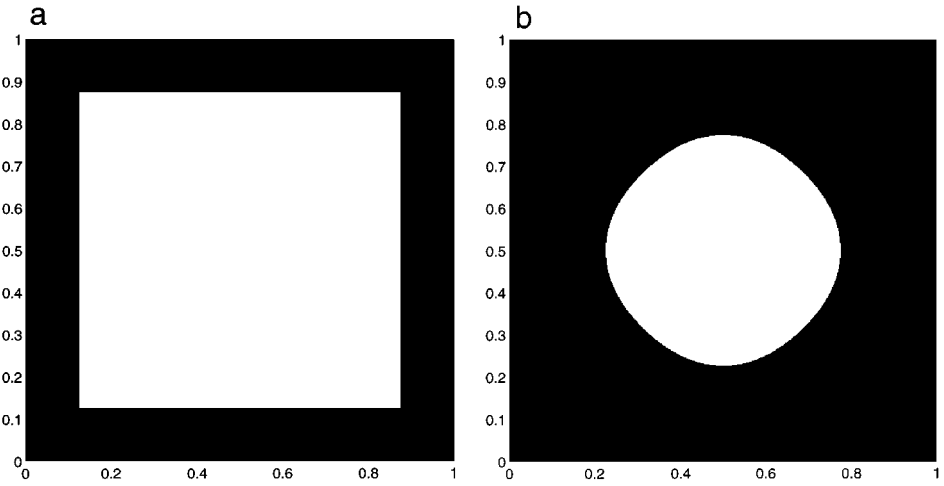

FIG. 7. The anisotropic curvature-dependent motion of a square region over a time $T = 1/24$. Here, $v_n = \kappa/(6\max(\sin^2(\theta), \cos^2(\theta)))$. (a) Original region. (b) Final region.

The algorithm CG-Motion, however, obtains accurate, efficient approximations to the limiting motion. See Table II. Even more accurate results can be obtained by increasing the number of basis functions and by using a stricter tolerance for the quadrature step. See Table III for a demonstration of this property for the semi-discrete problem.

Of course, the algorithm CG-Motion can also be used to obtain improved discretizations of other well-known models which have traditionally been treated using automata. The remainder of this article will focus on models arising in two specific application areas: developmental biology and excitable media.

5. APPLICATION TO DEVELOPMENTAL BIOLOGY

Cellular automata of the form (1) have arisen in a variety of disciplines in developmental biology [5]. We now focus on an interesting model of vertebrate skin patterns which is based on local activation and inhibition [37].

5.1. *The Model*

Young's model for vertebrate skin patterns [37] assumes that cells are in one of two states—differentiated (colored) and undifferentiated. Each differentiated cell produces two

TABLE II
Errors and CPU Times for the Algorithm CG-Motion

Δt	N	Relative error (semi-discrete)	Relative error (limiting motion)	CPU
1/192	32	0.4%	4.3%	3 s
1/384	32	0.7%	3.0%	6 s
1/768	32	0.4%	0.7%	12 s

TABLE III
Errors and CPU Times for the Algorithm CG-Motion

Δt	N	Relative error (semi-discrete)	CPU
1/192	32	0.4%	3 s
1/192	64	0.06%	14 s
1/192	128	0.01%	65 s

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. See Fig. 8.

To discretize this continuum model, Young uses an automaton. The initial state is given by

$$C_{jk}^0 = \begin{cases} 1 & \text{if cell } (j,k) \text{ is differentiated} \\ 0 & \text{otherwise.} \end{cases}$$

Updated states are found by spatially averaging over a circular neighborhood, $\mathcal{N}$, and applying a threshold function,

$$C_{jk}^{n+1} = \begin{cases} 1 & \text{if } \sum_{j',k' \in \mathcal{N}} W_{j',k'} C_{j-j',k-k'}^n \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where W is a matrix derived from the weighted difference of chemical concentrations. Although a variety of weights produce interesting steady-state patterns [26, 37], we shall study the particularly natural choice of setting W equal to the difference of two Gaussians (see Fig. 8c) [26]. Alternative choices include setting W to an approximation of the difference of two scaled characteristic functions [37].

5.2. *A Convolution-Based Discretization*

The convolutional form of Young's automaton is easily derived. Simply set

$$\chi : R^d \to R$$

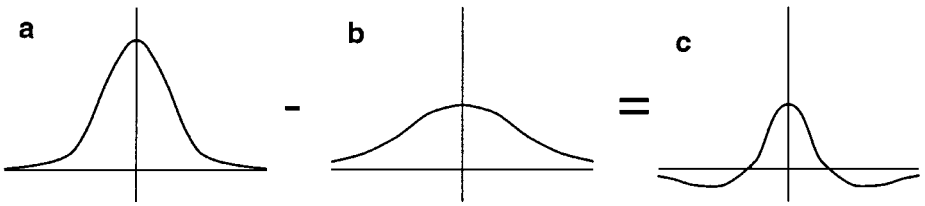

FIG. 8. The steady-state activator (a) and inhibitor (b) concentrations about a differentiated cell are modeled using Gaussian distributions. The inhibitor has a longer range than the activator. The combined effect (c) of activator and inhibitor is modeled using the difference of the two Gaussian distributions.

equal to the characteristic function for the differentiated region, Ω, and define the updated region, Ω^{new}, to be the set

$$\Omega^{\text{new}} = \{\mathbf{x}\colon \chi * K(\mathbf{x}) > 0\}$$

for the kernel function, K. The kernel K is not refined with Δt since the time evolution of the model is naturally discrete. (If such a refinement were carried out it would change the scaling of the final patterns and would not produce a well-defined motion law.)

Solutions to this convolution-based model are efficiently obtained using the algorithm CG-Motion. For example, the steady patterns given in Fig. 9 were obtained using $N = 128$. Note that these computations are fully resolved since further refinement in N leaves the results unchanged. Automata-based solutions, however, are inefficient because interfaces are poorly represented. Even using a lattice of 2048×2048 grid points, an incorrect solution is obtained using the algorithm RegularLattice for the case $K = K_A - \frac{4}{5}K_I$.

Of course, the algorithm CG-Motion also efficiently (and trivially) treats more general kernels. Asymmetric kernels are of particular interest because they can generate striped patterns [37]. See Fig. 10 for an example.

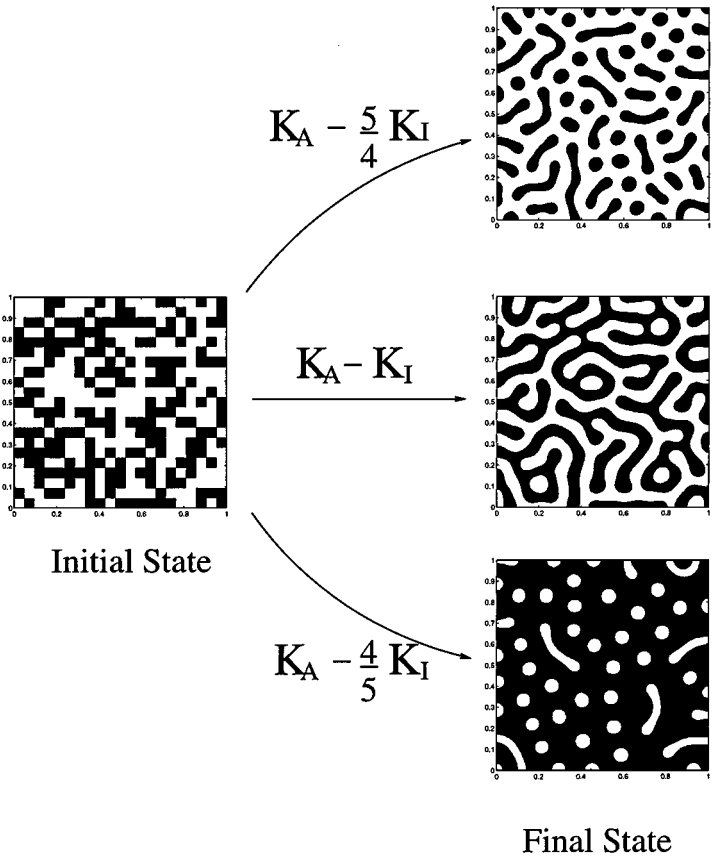

FIG. 9. Isotropic pattern formation after 100 steps starting from a random checkerboard pattern. In each case the kernel is the difference of two Gaussian distributions, $K_A(\mathbf{x}) = \frac{2500}{\pi}\exp(-2500|\mathbf{x}|^2)$ and $K_I(\mathbf{x}) = \frac{1250}{3\pi}\exp(-\frac{1250}{3}|\mathbf{x}|^2)$.

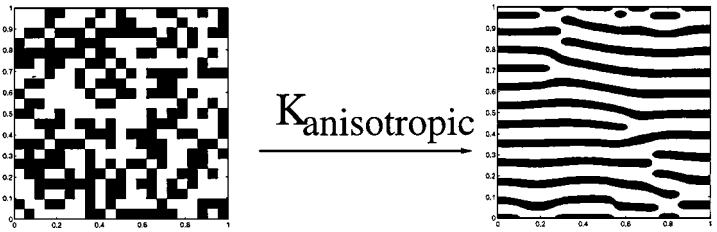

FIG. 10. Anisotropic pattern formation. In this case, the kernel is asymmetric: $K_{\text{anisotropic}}(x, y) = \frac{2500}{\pi}\exp(-1250(2x^2 + y^2)) - \frac{1250}{3\pi}\exp(-\frac{625}{3}(x^2 + 2y^2))$.

6. APPLICATION TO EXCITABLE MEDIA

In the biological and physiological literature, the best-known examples of cellular automata are the excitable media [5]. We now give a brief overview of cellular automata models for excitable media and show that convolution-generated motion is useful for producing improved approximations to certain features that are essential to the system.

6.1. *Overview*

In an excitable system, a sufficient stimulus (i.e., above some threshold value) leads to a large response followed by a period of recovery to a stable rest state. An *excitable medium* is a spatially distributed excitable system coupled in such a way that excitation can provoke excitation in neighboring regions. Note that these systems often experience a recovery or *refractory* period during which the medium is unable to be re-excited regardless of the size of the stimulus. Examples of excitable media arise in diverse physical, chemical, and biological systems, including models for nerve cells, muscle cells, cardiac function, developmental biology, chemical reaction, and star formation. See [32, 5, 30, 34] for further details and references.

In early cellular automata models for excitable media, update rules were based on the values in a neighborhood of nearest neighbors. Because this choice produces waves which propagate at a speed of one cell per time step, several serious shortcomings occur [33, 8]. The most serious of these are [31, 13]:

1. The speed of propagation does not depend on the wavefront curvature.
2. The speed of propagation does not depend on the extent of recovery of the medium.
3. Unwanted anisotropy is added to the front motion.

In order to include the effects of dispersion,[5] more recent automata select threshold values according to the recovery of the medium [13, 30, 31, 10, 8, 9, 7, 16]. Averages over large neighborhoods are used in an attempt to reduce unwanted anisotropy and to obtain an approximation to the curvature component of the motion [13, 30, 31, 10, 8, 9, 7, 16]. Typically, this averaging step is carried out either directly [7, 16] (which is slow but general), using a number of one-dimensional convolutions [13, 30, 31, 10, 8, 9] (which is efficient but specialized), or pseudospectrally (which is efficient *and* general—see Section 4).

[5] The dependency of wave speed on the extent of recovery of the medium is known as *dispersion*.

Although these changes give a significant improvement over early cellular automata models, several serious objections remain:

1. Similar to the case of convolution-generated motion (see Section 4.2), the grid spacing must be much less than the speed of propagation of the front times the time step size (see Eq. (5)) or the thresholding step may keep the front stationary. This requirement can be especially severe for excitable media simulations in 3D since important features such as "vortex filaments" and "scroll rings" move with speed proportional to their local curvature (which can be small) [29].
2. Wave fronts travel an integer number of cells per time step. Reducing these discontinuities to an acceptable level can require a prohibitively large number of operations per step (cf. Section 4.2). Gerhardt *et al.* explain this effect (for their automaton) as follows [8]: "Waves tend to travel at integer speeds ... these discontinuities in wave speed cannot be eliminated but they can be reduced by increasing r (the radius of the neighborhood). However, as r increases, the number of cells per unit length must increase, so computation time increases."
3. Similar to the case of convolution-generated motion (see Section 4.2), an unwanted, grid-based anisotropy is added to the motion.
4. In three dimensions, the simulation of "vortex filaments" is of great current interest. Unfortunately current automata do not seem to be able to model the motion of these structures. For example, Henze and Tyson state that in their simulations "updates in the cellular automata may be too infrequent to resolve accurately the intraperiod fluctuations that determine net filament displacement in the local normal and binormal directions" [13]. Indeed, for their automaton (or others [30, 31, 10, 8, 9]), "it is hard to address this potentially paramount issue directly ... since it is impossible to change the time step independently of the other parameters" [13].
5. Often, it is of great theoretical interest to determine geometric properties defined on the interface (see, e.g., [29]). Because cellular automata use a crude approximation of the front (see, e.g., Fig. 1), they often produce a very poor representation of such geometric quantities. For example, Henze and Tyson [13] note that in 3D, "values of filament geometry, twist, arclength derivative of twist and displacements based on cellular automata pivot cores are very noisy compared with their PDE analogues."
6. Finally, there is a possibility that certain dynamics of the media may not be captured using automata which limit the excitation variable to only two states. Gerhardt *et al.* explain this effect (for their automaton) as follows [8]: "The dynamics of the core of a spiral wave can never be fully satisfactory in our cellular automaton because we limit the excitation variable to two states only. Thus, we can never describe subtle differences in the excitation variable, such as those present in the core of a spiral wave where criss crossing concentration gradients seem to be important."

As we shall see in the next section, many of these objections are simply artifacts of using a lattice-based model and thus are naturally eliminated using convolution-based methods.

6.2. *A Convolution-Based Model*

Convolution-generated motion naturally eliminates many of the objections found in automata-based models for excitable media. Furthermore, it is typically straightforward to derive the convolutional form based on an automata model.

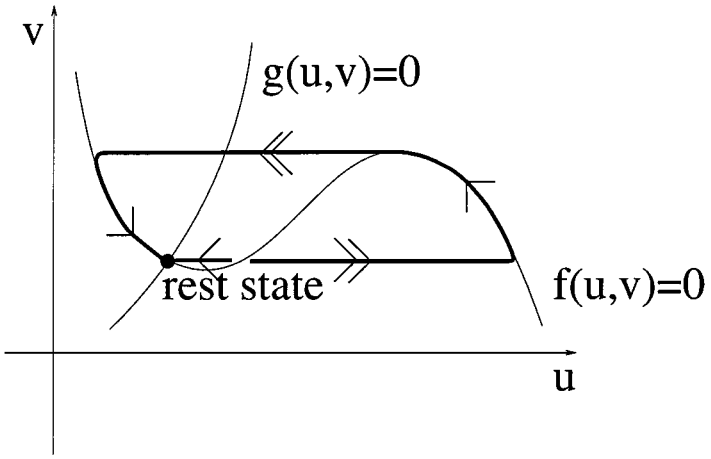

FIG. 11. Typical phase plane diagram for an excitable medium. The local kinetics of the excitation variable u and the recovery variable v are determined by the nullclines of the systems: $f(u, v) = 0$ and $g(u, v) = 0$. Small perturbations from the unique rest state are damped out, while large perturbations cause the system to undergo a large excursion before returning to the rest state.

Consider, for example, the excitable automata introduced in Gerhardt *et al.* [10, 8, 9], Weimar *et al.* [30, 31], and Henze and Tyson [13]. In these automata, update rules are chosen to mimic the dynamics of a two-variable system of reaction–diffusion equations,

$$\frac{\partial u}{\partial t} = \frac{1}{\epsilon} f(u, v) + D_u \nabla^2 u$$
$$\frac{\partial v}{\partial t} = g(u, v) + D_v \nabla^2 v,$$

where $f(u, v)$ and $g(u, v)$ specify the local kinetics of the system (see Fig. 11). Note that the scalar u (the excitation variable) changes on a time scale which is much faster than the scalar v (the recovery variable). To derive the corresponding automaton model, the reaction–diffusion system is "split" in a non-convergent way into four steps which are carried out sequentially (see the Appendix for details):

1. The excitation variable is diffused.
2. The diffused excitation variable is thresholded to 0 (resting) or 1 (excited) according to the value of the recovery variable, i.e., $\lambda = \lambda(v)$.
3. The recovery variable is evolved according to the local kinetics.
4. The result from Step 3 is diffused to give the updated recovery variable.

Finally, the discretization is completed by representing u and v by their pointwise values on a regular lattice.

The advantages of this automaton over earlier models are clear. Since large neighborhoods are used, the motion of the wave front will exhibit fewer grid effects (i.e., reduced anisotropy) and will have an improved dependence on curvature. Furthermore, dispersion is included in the model since thresholding is carried out according to the value of the recovery variable. Indeed, simulation results reported for FitzHugh-Nagumo kinetics [13] and the Oregonator model [31] agree well with PDE simulations for the period, wavelength,

and meander patterns[6] of spiral waves for a wide range of parameters. When compared to PDE simulations, the automata model has the practical advantage that it ignores the details of the fast kinetics so that "the time step in the cellular automaton can exceed that in PDE simulations by 1 or 2 orders of magnitude" [13].

Unfortunately, a variety of serious objections arise whenever a lattice-based spatial discretization is used. As discussed in the previous section, discontinuities in the motion and unwanted anisotropy occur because the front travels an integer number of cells per time step. A prohibitively fine mesh may be required in order to resolve the motion of important features such as vortex filaments. Also, geometric quantities defined on the interface are often impractical to estimate using an automaton because a very crude representation of the front is used. Finally, we would like to be able to refine the time step independent of other parameters in order to optimize the discretization and to determine the strengths and limitations of the model. Unfortunately, this type of refinement is impossible whenever the averaging step is carried out using the one-dimensional convolutions proposed in [13, 30, 31, 10, 8, 9].

Similar to the case of convolution-generated motion, we want to avoid discretizations which use a pointwise thresholding because this type of thresholding displaces the front a distance which is comparable to the mesh spacing. Fortunately, an improved discretization is easily obtained. Steps 1 and 2 above are trivially treated using the algorithm CG-Motion. The evolution of the recovery variable is similar, except that we must use Gaussian quadrature rather than exact integration to evaluate the Fourier coefficients.

Similar to automata-based discretizations, this convolution-generated approach allows for very large time steps (relative to PDE simulations) since it ignores the details of the fast dynamics.[7] Relative to automata-based discretizations, however, it is clear that our approach gives a much more accurate treatment of the front since it recursively refines near the interface and interpolates at the finest cell level. This allows for accurate estimates of quantities defined on the interface and even captures the motion of features which depend on curvature. Furthermore, discontinuities and unwanted anisotropy in the front motion are eliminated since interpolation is used to locate the front at the finest cell level. Finally, our proposed discretization has the benefit that Δt can be selected independently of other parameters, unlike the methods proposed in [13, 30, 31, 10, 8, 9].

We now give a number of numerical experiments which were derived using our proposed discretization. More detailed studies will be the focus of a future report.

6.3. *Numerical Experiments*

We now apply our algorithms to the problem of simulating an excitable medium. For simplicity, we consider the convolution-based model that corresponds to Barkley's local kinetics [1],

$$f(u, v) = u(1 - u)(1 - u_{th}(v))$$
$$g(u, v) = u - v,$$

where $u_{th}(v) = (v - a)/b$ and a and b are parameters. (See the Appendix for full details.)

[6] The *meander pattern* is the path traced out by the tip of a spiral wave. See [34] for further details.

[7] Since the details of the fast dynamics are ignored, this approach may be inadequate for modeling the dynamics of the spiral core in certain problems (see previous section).

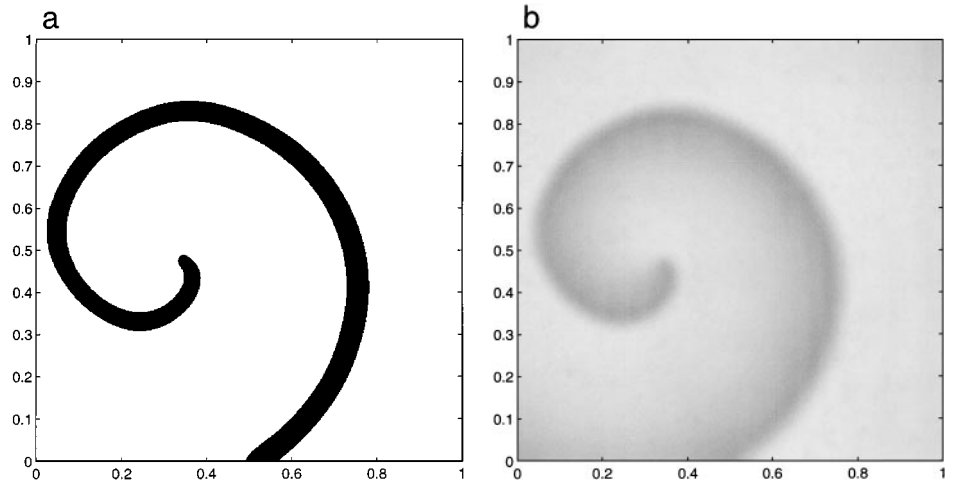

FIG. 12. An isolated spiral wave. This simulation used Barkley's dynamics with $a = 0.3$, $b = 0.01$, $\epsilon = 0.005$, and $D_u = 0.000625 = D_v$. The discretization parameters were $N = 64$ and $\Delta t = 1/16$.

If we apply the algorithm CG-Motion to this model, a variety of interesting spiral waves are obtained. Figure 12, for example, gives an isolated spiral produced by this model. Notice that the wave front is smooth and isotropic, even though only 128×128 basis functions are used. (Lattice-based discretizations for similar spirals exhibit considerable grid-based anisotropy. See, e.g., Fig. 5 of [31].) Figure 13 demonstrates that colliding spirals can be generated using the same model with different initial conditions. Note that all shape changes are handled naturally by the method—no special treatment of topological mergings or breakings is needed.

Of course, non-Gaussian kernels may also be used in the convolution step to produce spiral waves with a desired anisotropy. The use of such asymmetric kernels will be considered as part of a future report.

7. CONCLUSION

In this work, we have presented convolution-generated motion both as a means for describing the limiting pattern dynamics of a class of cellular automata models and as an independent, alternative foundation for expressing pattern producing models.

Conceptually, convolution-generated motion can be thought of as a class of models intermediate between cellular automata and continuum PDEs, in that it eliminates the fine scales of automata on the one hand, but it naturally describes perfect interfaces, which are often obtained as singular limits of reaction–diffusion/Ginzburg–Landau type PDEs, at the other extreme. Thus it simultaneously achieves the long length scale limit of automata and the short length scale limit of reaction–diffusion PDEs, both of which are difficult limits to analyze theoretically or simulate numerically. Convolution-generated motion also makes accessible a large class of motions that cannot be conveniently described or approximated by traditional PDE or automata methods.

From a computational standpoint, convolution-generated motion is easy to implement since convolutions can be carried out as multiplications in Fourier space and thresholding can be carried out using a simple quadrature. These methods are also efficient and

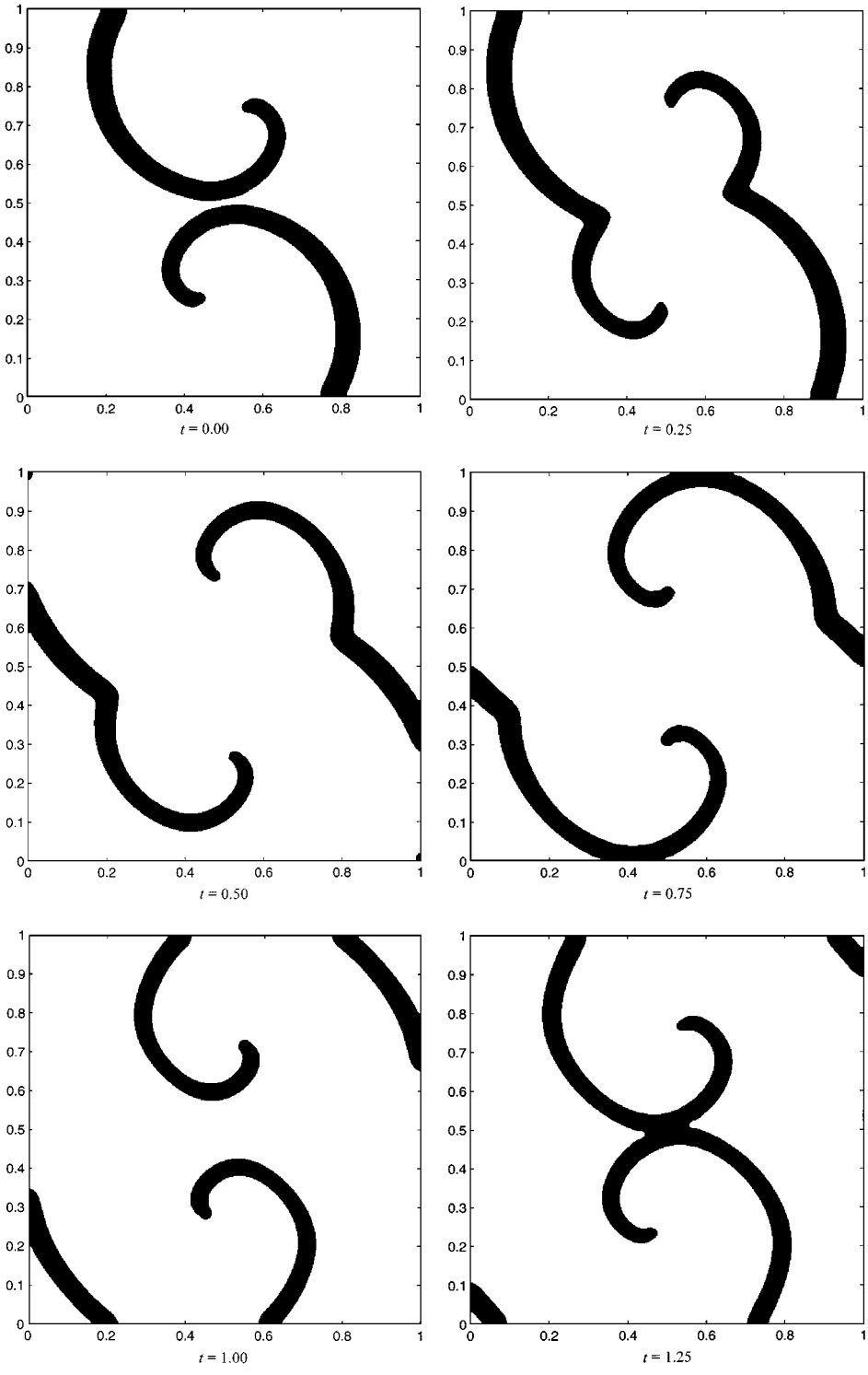

FIG. 13. Excitation variable for two interacting spiral waves. All parameters are set according to Fig. 12.

accurate because they can be implemented using adaptively refined grids with fast Fourier transform methods. These properties lead to a variety of practical advantages. In particular, these methods do not exhibit unwanted anisotropy and they capture curvature-dependent motions that are impractical to study using automata-based discretizations. Furthermore, convolution-based discretizations are much faster (e.g., 1000 times faster) when accurate solutions are required. In addition, convolution-based algorithms produce smooth interfaces, thus allowing geometric quantities such as the interface normal and curvature to be accurately evaluated. Motion parameters such as Δt are easily varied independent of other parameters in the model, as well.

Theoretically, convolution-generated motion has the desirable property that it is often straightforward to derive the limiting motion laws for smooth interfaces [24]. Furthermore, rigorous derivations of limiting motion laws have been obtained for an interesting variety of convolution kernels and threshold values [15].

Based on these computational and theoretical advantages as well as our studies of models arising in developmental biology and excitable media, we believe that convolution-generated motion represents a useful, novel approach for modeling natural phenomena.

In future work, we shall carry out a detailed comparison of the convolution-based model for excitable media with its reaction–diffusion PDE counterpart. In particular, it will be of great interest to compare the motion of vortex filaments in three dimensions with asymptotically derived motion law [29] and with PDE simulations. Also, we plan a general investigation of the connection between convolution-generated motion and interface dynamics described by the singular limit of reaction–diffusion/Ginzburg–Landau PDEs, similar to the investigation of its connection with automata carried out in this paper.

APPENDIX

A Convolution-Based Model for Excitable Media

Gerhardt *et al.* [10, 8, 9], Weimar *et al.* [30, 31], and Henze and Tyson [13] devised a convolution-based model for excitable media which mimics the dynamics of a two-variable system of reaction–diffusion equations,

$$\frac{\partial u}{\partial t} = \frac{1}{\epsilon} f(u, v) + D_u \nabla^2 u$$
$$\frac{\partial v}{\partial t} = g(u, v) + D_v \nabla^2 v.$$

A variety of reaction dynamics may be simulated using their model [31, 13]. For simplicity, this section considers the special case of Barkley's local kinetics [1] (see Fig. 14),

$$f(u, v) = u(1 - u)(1 - u_{th}(v))$$
$$g(u, v) = u - v,$$

where $u_{th}(v) = (v - a)/b$ and a and b are parameters.

For these kinetics, the convolution-based approach derives estimates of u and v at time $t + \Delta t$ from the values at time t by "splitting" the reaction–diffusion system into four parts. Each of these parts is treated sequentially as follows:

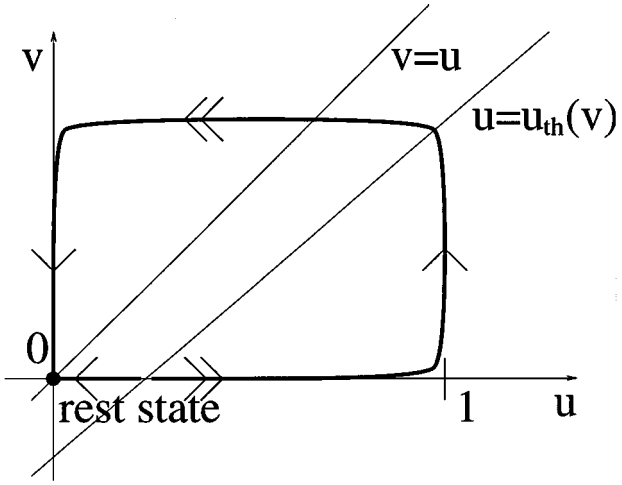

FIG. 14. Phase plane diagram for Barkley's local dynamics. Here, the nullcline for u is the line $v = u$, and the nullcline for v consists of three lines: $u = 0$, $u = u_{th}(v)$ and $u = 1$.

1. Evaluate the convolution of $u(t)$ with the heat kernel,

$$\bar{u} = u(t) * K_{D_u},$$

where

$$K_{D_u}(\mathbf{x}) \equiv \frac{1}{4\pi D_u \Delta t} \exp\left(-\frac{1}{4D_u \Delta t}|\mathbf{x}|^2\right).$$

2. Threshold $\bar{u}$ by setting

$$\bar{u} = \begin{cases} 1 & \text{if } \bar{u} > \lambda(v) \\ 0 & \text{otherwise.} \end{cases}$$

The threshold value, $\lambda(v)$, can be determined by demanding that the dependence of wave speed on the state of recovery match that of the PDE [13]. For Barkley's dynamics, the speed of a planar wave can be derived analytically [28],

$$c(v) = (1 - 2u_{th}s(v))\sqrt{\frac{D_u}{2\epsilon}}.$$

To preserve this relationship between wave speed and recovery, it is easy to show (using the 1D analytical solution to the heat equation) that

$$\lambda(v) = \frac{1}{2} - \frac{1}{2}\mathrm{erf}\left(c(v)\sqrt{\frac{\Delta t}{D_u}}\right).$$

3. Evolve the excitation and recovery variables according to the local dynamics for a time Δt. Specifically,

(a) Find the solution of $\bar{v}_t = g(\bar{u}, \bar{v})$ at time $t + \Delta t$ starting from $\bar{v}(t) = v(t)$.

(b) If $\bar{v}(t+\Delta t) > v_{\mathrm{MAX}}$ then the excitation variable changes in the interval and we must:

- Find the time when the excitation variable changes; i.e., find t^* so that $\bar{v}(t^*) = v_{\mathrm{MAX}}$.
- Set $\bar{u} = 0$.
- Find the solution of $\bar{v}_t = g(0, \bar{v})$ at time $t + \Delta t$ starting from $\bar{v}(t^*) = v_{\mathrm{MAX}}$.

(c) Update the excitation variable: $u(t+\Delta t) = \bar{u}$.

4. Evaluate the convolution of $\bar{v}$ with the heat kernel to obtain the updated value of the recovery variable,

$$v(t+\Delta t) = \bar{v} * K_{D_v}$$

where

$$K_{D_v}(\mathbf{x}) \equiv \frac{1}{4\pi D_v \Delta t} \exp\left(-\frac{1}{4D_v \Delta t}|\mathbf{x}|^2\right).$$

Note that Step 1 simply evolves the excitation variable according to the heat equation. Steps 2 and 3 update each variable according to the local dynamics. Finally, Step 4 evolves the recovery variable according to the heat equation.

REFERENCES

1. D. Barkley, A model for fast computer simulation of waves in excitable media, *Physica D* **49**, 75 (1991).
2. G. Barles and C. Georgelin, A simple proof of convergence for an approximation scheme for computing motions by mean curvature, *SIAM J. Numer. Anal*. **32**(2), 484 (1995).
3. G. Beylkin, On the fast Fourier transform of functions with singularities, *Appl. Comput. Harmonic Anal*. **2**, 363 (1995).
4. 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).
5. G. B. Ermentrout and L. Edelstein-Keshet, Cellular automata approaches to biological modeling, *J. Theor. Biol*. **160**, 97 (1993).
6. L. C. Evans, Convergence of an algorithm for mean curvature motion, *Indiana Univ. Math. J*. **42**, 553 (1993).
7. V. G. Fast and I. G. Efimov, Stability of vortex rotation in an excitable cellular medium, *Physica D* **49**, 75 (1991).
8. M. Gerhardt, H. Schuster, and J. J. Tyson, A cellular automata model of excitable media. II. Curvature, dispersion, rotating waves and meandering waves, *Physica D* **46**, 392 (1990).
9. M. Gerhardt, H. Schuster, and J. J. Tyson, A cellular automata model of excitable media. III. Fitting the Belousov–Zhabotinsky reaction, *Physica D* **46**, 416 (1990).
10. M. Gerhardt, H. Schuster, and J. J. Tyson, A cellular automata model of excitable media including curvature and dispersion, *Science* **247**, 416 (1990).
11. J. Gravner and D. Griffeath, Threshold growth dynamics, *Trans. Am. Math. Soc*. **340**(2), 837 (1993).
12. H. Gutowitz (Ed.), *Cellular Automata: Theory and Experiment* (MIT Press, Cambridge, MA, 1991).
13. C. Henze and J. Tyson, Cellular automaton model of three-dimensional excitable media, *J. Chem. Soc., Faraday Trans*. **92**(16), 2883 (1996).
14. H. Ishii, A generalization of the Bence, Merriman and Osher algorithm for motion by mean curvature, in *Curvature Flows and Related Topics*, edited by A. Damlamian, J. Spruck, and A. Visintin (Gakkôtosho, Tokyo, 1995), pp. 111–127.

15. H. Ishii, G. E. Pires, and P. E. Souganidis, Threshold dynamics type schemes for propagating fronts, *TMU Math. Preprint Ser*. **4** (1996).
16. M. Markus and B. Hess, Isotropic cellular automaton for modeling excitable media, *Nature* **347**, 56 (1990).
17. P. Mascarenhas, *Diffusion Generated Motion by Mean Curvature*, CAM Report 92-33, University of California, Department of Math, Los Angeles (1992).
18. B. Merriman, J. Bence, and S. Osher, Diffusion generated motion by mean curvature, in *Computational Crystal Growers Workshop*, edited by J. E. Taylor (American Mathematical Society, Providence, RI, 1992), pp. 73–83. Also available as UCLA CAM Report 92-18 (April 1992).
19. B. Merriman, J. Bence, and S. Osher, Motion of multiple junctions: A level set approach, *J. Comput. Phys*. **112**(2), 334 (1994).
20. J. Rubinstein and P. Sternberg, Nonlocal reaction-diffusion equations and nucleation, *IMA J. Appl. Math*. **48**, 248 (1992).
21. S. J. Ruuth, *Efficient Algorithms for Diffusion-Generated Motion by Mean Curvature*, Ph.D. thesis, University of British Columbia, Vancouver, Canada (1996).
22. S. J. Ruuth, A diffusion-generated approach to multiphase motion, *J. Comput. Phys*. **145**, 166 (1998).
23. S. J. Ruuth, Efficient algorithms for diffusion-generated motion by mean curvature, *J. Comput. Phys*. **144**, 603 (1998).
24. S. J. Ruuth and B. Merriman, *Convolution Generated Motion and Generalized Huygens' Principles for Interface Motion*, CAM Report 98-04, University of California, Los Angeles (1998).
25. B. Schönfisch, Anisotropy in cellular automata, *Biosystems* **41**, 29 (1997).
26. N. V. Swindale, A model for the formation of ocular dominance stripes, *Proc. R. Soc. London B* **208**, 243 (1980).
27. T. Toffoli and N. Margolus, *Cellular Automata Machines* (MIT Press, Cambridge, MA, 1987).
28. H. Tuckwell, *Introduction to Theoretical Neurobiology, Volume 2, Nonlinear and Stochastic Theories* (Cambridge Univ. Press, Cambridge, UK, 1988).
29. J. J. Tyson and J. P. Keener, Singular perturbation theory of traveling waves in excitable media, *Physica D* **32**, 327 (1988).
30. J. R. Weimar, J. J. Tyson, and L. T. Watson, Diffusion and wave propagation in cellular automata models of excitable media, *Physica D* **55**, 309 (1992).
31. J. R. Weimar, J. J. Tyson, and L. T. Watson, Third generation cellular automata for modeling excitable media, *Physica D* **55**, 328 (1992).
32. N. Wiener and A. Rosenblueth, The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle, *Arch. Inst. Cardiol. Mexico* **16**, 205 (1946).
33. A. T. Winfree, *When Time Breaks Down* (Princeton Univ. Press, Princeton, NJ, 1987).
34. A. T. Winfree, Stable particle-like solutions to the nonlinear wave equations of three-dimensional excitable media, *SIAM Rev*. **32**(1), 1 (1990).
35. S. Wolfram, Universality and complexity in cellular automata, *Physica D* **10**, 1 (1984).
36. S. Wolfram, *Cellular Automata and Complexity: Collected Papers* (Addison–Wesley, Reading, MA, 1994).
37. D. A. Young, A local activator-inhibitor model of vertebrate skin patterns, *Math. Biosci*. **72**, 51 (1984).