SIAM J. SCI. COMPUT.
Vol. 33, No. 5, pp. 2382–2401

ALGORITHMS FOR AREA PRESERVING FLOWS*

CATHERINE KUBLIK†, SELIM ESEDOĞLU‡, AND JEFFREY A. FESSLER‡

Abstract. We propose efficient and accurate algorithms for computing certain area preserving geometric motions of curves in the plane, such as area preserving motion by curvature. These schemes are based on a new class of diffusion generated motion algorithms using signed distance functions. In particular, they alternate two very simple and fast operations, namely convolution with the Gaussian kernel and construction of the distance function, to generate the desired geometric flow in an unconditionally stable manner. We present applications of these area preserving flows to large scale simulations of coarsening.

Key words. diffusion generated motion, level set, area preserving mean curvature motion, coarsening

AMS subject classification. 65C20

DOI. 10.1137/100815542

1. Introduction.

Computing the motion of interfaces, e.g., curves in the plane or surfaces in space, is an essential component of many applications. For instance, in image processing and computer vision, many popular variational models for segmentation and reconstruction involve initializing a curve or surface and then evolving it towards features (e.g., edges) that are of interest. Typically, the resulting motion is by a normal speed that includes geometric terms such as curvature of the interface and its derivatives. There are many numerical schemes for simulating geometric motions of interfaces, including front tracking, phase-field, and level set methods. Front tracking methods [4] involve an explicit discretization of the interface and approximate its motion by moving marker particles located on the interface. These methods are well suited for two dimensional motions of curves that do not cross, but become very difficult to implement whenever topological changes occur, particularly in dimensions higher than two. On the other hand, phase-field [23, 26] and level set [27] methods represent the interface implicitly and are therefore able to handle topological changes naturally.

Motion by mean curvature has been extensively studied in the mathematics literature [15, 17, 19] and in applications such as crystal growth and image processing [1, 24]. Under this geometric flow each point x on a curve Γ moves with normal velocity $v_N = \kappa(x)$, where $\kappa(x)$ is the mean curvature of the curve at $x \in \Gamma$. It is also called the Euclidean curve shortening flow since the Euclidean perimeter of a curve shrinks as quickly as possible when evolving according to this motion. A variant of this motion is the geometric flow that decreases the total perimeter of a collection of curves as quickly as possible while preserving the total enclosed area: it is referred to as area (or volume) preserving mean curvature motion. This motion finds

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 22, 2010; accepted for publication (in revised form) July 12, 2011; published electronically September 27, 2011.

http://www.siam.org/journals/sisc/33-5/81554.html

†Department of Mathematics, University of Texas at Austin, Austin, TX 78712 (kublik@math.utexas.edu). This author's research was supported by NSF grants DMS-0748333 and DMS-0713767.

‡Department of Mathematics, University of Michigan, Ann Arbor, MI 48109 (esedoglu@umich.edu, fessler@umich.edu). The second author's research was supported by NSF grants DMS-0748333 and DMS-0713767 and an Alfred P. Sloan Foundation fellowship. The third author's research was supported by NIH grant R01 HL 098686.

applications in image processing [10, 16] and arises physically as a limit of a nonlocal model describing the phase separation of binary alloys [3, 18, 29], and as a limit of the Lifshitz–Slyozov–Wagner (LSW) mean field model with kinetic drag when the size of the particles is small compared to the value of the kinetic drag [8]. The LSW model was introduced by Lifshitz, Slyozov, and Wagner in [20, 42] to study the general phenomenon of Ostwald ripening [25], which describes the dissolution/precipitation dynamics of precipitate crystals in a saturated solution: small precipitate crystals dissolve and redeposit themselves on the surface of larger crystals, thus creating a coarsening of the system size since small particles shrink and eventually vanish while larger ones grow. The area preserving mean curvature flow exhibits a similar coarsening behavior (see section 4). Mathematically, the area preserving mean curvature motion is characterized by its normal velocity $v_N = \kappa - \bar{\kappa}$, where κ is the mean curvature and $\bar{\kappa}$ denotes the average mean curvature of the collection of curves evolving under this motion.

Several schemes based on the level set method [10, 28, 44] and diffusion generated motion [37] have been proposed for approximating this motion. In this paper, we describe new schemes for area preserving flows based on a new class of algorithms that generate the desired interfacial motion by alternating two simple and efficient steps: construction of the signed distance function, and convolution with a kernel (usually a Gaussian kernel). The resulting schemes are unconditionally stable, and have low, namely $O(N \log N)$, per time step cost, where N is the total number of grid points. In addition, we present applications of these efficient area preserving algorithms to large scale simulations of area preserving curvature motion. The paper is organized as follows. In section 2, we give a brief description of previous diffusion generated motion based algorithms, and we describe our new area preserving schemes in section 3. We present applications of our new algorithms in section 4 and conclude in section 5.

2. Diffusion generated motion. The algorithms proposed in this paper have been motivated by the threshold dynamics idea of Merriman, Bence, and Osher [22, 23]. The Merriman–Bence–Osher (MBO) algorithm is obtained by time splitting the well-known Allen–Cahn phase-field equation for motion by mean curvature. The resulting scheme alternates two steps, namely convolution and thresholding. More precisely, consider a set Σ in the domain $\Omega \subset \mathbb{R}^2$, with boundary $\partial\Sigma$ evolving with normal speed $v_N = \kappa$, where $\kappa(x)$ is the mean curvature of the interface $\partial\Sigma$ at the point $x \in \partial\Sigma$. Then $\partial\Sigma$ is the $\frac{1}{2}$ level set of the characteristic function χ_Σ of the set Σ:

$$\partial\Sigma =: \left\{ x : \chi_\Sigma(x) = \frac{1}{2} \right\}.$$

Given an initial set Σ_0 defined through its characteristic function χ_0, and a time step size $\delta t > 0$, the MBO scheme generates a discrete sequence $\{\Sigma_j\}_{j\in\mathbb{N}}$ at subsequent times $j(\delta t)$ in the following way: from the set Σ_{j-1}, obtain the set Σ_j by alternating between the following two operations:

1. *Diffusion step.* From the set Σ_{j-1} defined through its characteristic function χ_{j-1} at $(j-1)\delta t$, solve the following initial value problem for a length of time δt:

$$\begin{cases} u_t = \Delta u, \\ u(x,0) = \chi_{j-1}(x), \end{cases}$$

which is equivalent to forming the function

$$\mathcal{L}(x) = (G_t * \chi_{j-1})(x),$$

where G_t is the n-dimensional Gaussian kernel given by

$$G_t(x) = \frac{1}{(4\pi t)^{\frac{n}{2}}} e^{\frac{-|x|^2}{4t}}. \tag{2.1}$$

2. *Sharpening step.*

$$\chi_j(x) = \begin{cases} 0 & \text{if } \mathcal{L}(x) < \frac{1}{2}, \\ 1 & \text{else.} \end{cases}$$

The location of the interface $\partial\Sigma_j$ is given by the level set $\{x \in \Omega : \chi_j(x) = \frac{1}{2}\}$. The time discretization $\{\partial\Sigma_j\}_{j\in\mathbb{N}}$ has been proven to converge to motion by mean curvature in the limit $\delta t \to 0^+$ (see [2, 13, 21]). One of the main advantages of this algorithm is its unconditional stability. In fact, the scheme remains monotone (preserves the order of sets) for all choices of δt, independent of the spatial resolution. In addition, its computational complexity is low ($O(N \log N)$ due to the FFT used in the convolution step), which makes it computationally more attractive than standard level set techniques that involve the solution of a nonlinear and degenerate PDE [27]. Nevertheless, there exists some semi-implicit schemes for level set methods described by Smereka in [39]. Several generalizations of the basic MBO scheme have been proposed for generating more complicated interfacial motions, including anisotropic curvature motion, motion by curvature plus constant, and motion of multiple junctions [12, 21, 31, 32, 33, 34, 35, 36].

Despite its computational advantages, the MBO scheme inherits a major drawback from its construction using characteristic functions: inaccuracy on uniform grids. Indeed, characteristic functions cannot resolve the location of the interface better than the spatial grid size. Consequently, unless the grid size is refined concurrently with the time step size, the approximate motion generated by the scheme gets stuck. It is therefore necessary to discretize the scheme with methods that can provide subgrid accuracy in the location of the interface. This was done by Ruuth in [32] through the efficient use of an unequally spaced FFT. Such an adaptive strategy is especially needed for simulating high order motions.

To address this issue, a variant of the MBO scheme has recently been proposed by Esedoḡlu, Ruuth, and Tsai in [11]. The new algorithms differ from the original by representing the interface through its signed distance function instead of its characteristic function, thereby replacing the thresholding step with a *redistancing step*. The new schemes therefore simulate the motion of an interface by alternately diffusing and redistancing the signed distance function to the interface. The diffusion step consists of convolving the distance function with an appropriate kernel, usually chosen to be the Gaussian kernel, and the redistancing step simply consists of constructing the signed distance function to the interface from the previously diffused distance function. In these algorithms, the computationally very efficient thresholding step is replaced by the construction of the signed distance function, for which there are also very efficient ($O(N \log N)$) algorithms (e.g., fast marching, fast sweeping, etc. [6, 30, 38, 40, 41]), hence maintaining the complexity of the original diffusion generated motion scheme. This new class of diffusion generated motion algorithms thus provides an efficient and highly accurate technique for simulating a wide range of interfacial motions.

3. Proposed schemes. In this section we introduce our area preserving schemes, and present systematic studies of their numerical convergence and accuracy.

3.1. Diffusion generated motion using signed distance functions. For a given open set $\Sigma \in \mathbb{R}^2$, we denote by d the signed distance function to its boundary $\partial\Sigma$:

$$d(x) := \begin{cases} \inf_{y\in\Sigma^c} |x-y| & \text{if } x \in \Sigma, \\ -\inf_{y\in\Sigma} |x-y| & \text{if } x \in \bar{\Sigma}^c, \end{cases}$$

where d is positive in the interior of Σ and negative outside its closure. The signed distance function dynamics [11] for motions with normal speed of the form

$$v_N = \kappa + S,$$

where κ is the curvature and S is a given function mapping $\mathbb{R}^2$ to $\mathbb{R}$, now reads as follows: from an initial set Σ_0 defined through its signed distance function $d_0(x)$ and a time step size $\delta t > 0$, generate a time discrete approximation $\{\partial\Sigma_j\}_{j\in\mathbb{N}}$ at times $j\,(\delta t)$ by alternating between the following two operations:

1. *Diffusion step.* From the set Σ_{j-1}, defined through its signed distance function d_{j-1} at time $(j-1)\delta t$, form the function

$$\mathcal{L}(x) = (G_t * d_{j-1})\,(x) + S_{j-1}(x)\,(\delta t)\,,$$

where G_t is the n-dimensional Gaussian kernel given in (2.1).

2. *Redistancing step.* Construct d_j, the signed distance function to the zero level set of $\mathcal{L}$, defining the new set Σ_j (and thus its boundary $\partial\Sigma_j$)

$$d_j(x) = \mathbf{Redist}(\mathcal{L}(x)),$$

where **Redist** denotes the construction of the signed distance function to the zero level set of $\mathcal{L}$; this operation is commonly used in level set techniques [27], and unlike thresholding does not require making a binary decision at each grid point about whether the point is inside or outside the zero level set, thus allowing subgrid resolution to be maintained.

3.2. Algorithms for area preserving flows. Building on the distance function dynamics for curvature motions, we propose new and efficient algorithms for area preserving flows in two dimensions. These algorithms generate interfacial motions with normal velocities

$$v_N = \kappa - \bar{\kappa} + S, \tag{3.1}$$

where κ denotes the curvature, $\bar{\kappa}$ is the average curvature, and $S = S(x,t)$ is an additional normal speed term that may depend on space and time. Such terms arise, for example, in computer vision applications from data fitting terms in variational models [10]. The core of our algorithm is the scheme for area preserving curvature motion which evolves interfaces with normal velocity $v_N = \kappa - \bar{\kappa}$. Under this motion, n disjoint curves Γ_i will evolve to decrease their total length while maintaining the total enclosed area constant. In our algorithms, we use the fact that both κ and $\bar{\kappa}$ can be calculated very easily using the signed distance function to the interface.

Let us now recall a few well-known properties of the signed distance function that hold more generally in $\mathbb{R}^n$ (see, e.g., [9, 14]). Consider the set $\Sigma \subset \Omega$ defined through

its signed distance function d, and let $\partial\Sigma$ be its boundary. The first property of d is based on the fact that the normals to a smooth interface do not focus immediately, so that the signed distance function is smooth in a tubular neighborhood T of $\partial\Sigma$, and linear with slope one along the normals:

$$|\nabla d| = 1 \text{ for all } x \in T, \text{ with boundary condition } d|_{x\in\partial\Sigma} = 0. \tag{3.2}$$

The second property is that the Laplacian of the signed distance function d at a point x gives, up to a multiplicative constant depending on the dimension, the mean curvature of the isosurface of d passing through x:

$$\Delta d(x) = (n-1)H(x), \tag{3.3}$$

where $H(x)$ denotes the curvature of the level set $\{\xi : d(\xi) = d(x)\}$, and n is the dimension. In two dimensions, we will denote by $\kappa(x)$ the curvature of the curve $\partial\Sigma = \{\xi : d(\xi) = 0\}$, so that (3.3) simplifies to

$$\Delta d(x) = \kappa(x). \tag{3.4}$$

Before moving on to the expression of the average curvature $\bar{\kappa}$ in terms of the signed distance function d, we need to recall some simple definitions and properties. The average curvature of a curve C is defined as

$$\bar{\kappa} = \frac{1}{|C|}\int_C \kappa ds, \tag{3.5}$$

where $|C|$ denotes the length of the curve C. For a two dimensional simply connected set Σ of genus p, the average curvature of its boundary $\partial\Sigma$ can be expressed as

$$\bar{\kappa} = \frac{2\pi\,(1-p)}{|\partial\Sigma|},$$

where $|\partial\Sigma|$ is the total length of the boundary of Σ. The genus number p can be interpreted as the number of "holes" in the set Σ. If the set Σ is made up of K connected components $\Sigma = \bigcup_{j=1}^{K}\Sigma_j$, with Σ_j being a set of genus p_j, then the average curvature of $\partial\Sigma$ becomes

$$\bar{\kappa} = \frac{2\pi\left(K - \sum_{j=1}^{K} p_j\right)}{|\partial\Sigma|}. \tag{3.6}$$

From expression (3.6), we see that the only quantity left to compute is the perimeter of the set Σ. Using the divergence theorem we notice that

$$\begin{aligned}
\int_\Sigma \Delta d\; dx &= \int_{\partial\Sigma} \nabla d\cdot\nu\; ds \\
&= \int_{\partial\Sigma} |\nabla d|^2 ds \qquad \text{(since } \nu = |\nabla d| \text{ is the outward unit normal)} \\
&= \int_{\partial\Sigma} ds \qquad \text{(since by definition of the signed distance function, } |\nabla d| = 1) \\
&= |\partial\Sigma|.
\end{aligned}$$

Thus we have

$$|\partial\Sigma| = \int_\Sigma \Delta d\; dx, \tag{3.7}$$

ALGORITHM 1. *Given the initial set Σ_0 defined through its signed distance function $d_0(x)$ and a time step $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(x)$ at the subsequent discrete times $t = j(\delta t)$ by alternating between the following two steps:*

1. *Using G_t in (2.1), form*

$$\mathcal{L}(x) = (d_{j-1} * G_{\delta t})(x) - \frac{2\pi\ (\#\{d_{j-1} > 0\} - \#\{d_{j-1} < 0\} + 1)}{\int_{d_{j-1}>0} \Delta d_{j-1}(x)dx}(\delta t) + S_{j-1}(x)(\delta t).$$

2. *Construct the signed distance function d_j using*

$$d_j(x) = \mathbf{Redist}(\mathcal{L}(x)).$$

which provides a simple relation between the length of the boundary of a set and its signed distance function. Note that (3.7) is valid even when d is not the signed distance function to the interface. In fact, (3.7) remains true if d is a function the gradient of which is the unit normal to the interface. Since construction of the signed distance function is needed for generating curvature motion in our algorithm, we use the signed distance function in (3.7). In our computations, we use (3.6) and (3.7) to compute the average curvature of a set. Notice that all the computations are done using only the signed distance function to the interface. Note also that under area preserving curvature motion, the boundaries of sets will evolve to decrease their total length while maintaining the total area at its initial value. Consequently, the final state of the evolution is a disk with area equal to the initial total area. In particular, disks preserve their circular symmetry under this motion.

The complete algorithm for the general motion with normal speed $v_N = \kappa - \bar{\kappa} + S$ builds on the area preserving curvature motion scheme by simply shifting the location of the redistancing process by a constant determined by $S(x,t)$. For clarity in the description of the algorithms, we define

$$\#\{d > 0\} := \text{ number of connected components of } \{x : d(x) > 0\}.$$

The first-order in time version of the algorithm is given in Algorithm 1 and can be improved to be second-order accurate in time in Algorithm 2.

The overall computational complexity of the general algorithm is $O(N \log N)$. Indeed, counting connected components can be performed in $O(N)$ operations, where N is the total number of grid points. The convolution is done in $O(N \log N)$ operations using the FFT. Also there exist algorithms, such as fast marching and fast sweeping, that construct the signed distance function in $O(N \log N)$ operations [6, 30, 38, 40, 41]. All the other terms used in the scheme, including the integral, can be done in $O(N)$ operations. Thus, we see that the per time step cost of the complete algorithm is $O(N \log N)$. In addition, due to its unconditional stability, there is no restriction on the time step size; in other words, the time step size is restricted only by accuracy considerations. If high accuracy is needed, however, it may be necessary to take very small time steps. Nevertheless the convergence studies carried out in the next section show that high accuracy can be achieved even with large time steps (see Tables 3 and 4). Note that the number of reinitialization operations involved in Algorithm 2 can be kept at one per time step.

ALGORITHM 2. ***Multistep, second-order in time version.*** *Given the initial set Σ_0 defined through its signed distance function $d_0(x)$ and a time step $\delta t > 0$, generate the sets Σ_j via their signed distance function $d_j(x)$ at the subsequent discrete times $t = j(\delta t)$ by alternating between the following two steps:*

1. *Using G_t in (2.1), form*

$$\mathcal{L}_1(x) = (d_{j-2} * G_{2\delta t})(x) + S_{j-2}(x)(2\delta t) - \frac{2\pi\ (\#\{d_{j-2} > 0\} - \#\{d_{j-2} < 0\} + 1)}{\int_{d_{j-2}>0} \Delta d_{j-2}(x)dx}(2\delta t),$$

$$\mathcal{L}_2(x) = (d_{j-1} * G_{\delta t})(x) + S_{j-1}(x)(\delta t) - \frac{2\pi\ (\#\{d_{j-1} > 0\} - \#\{d_{j-1} < 0\} + 1)}{\int_{d_{j-1}>0} \Delta d_{j-1}(x)dx}(\delta t).$$

2. *Construct the signed distance function d_j using*

$$d_j(x) = \mathbf{Redist}\left(\frac{1}{3}\left(4\mathcal{L}_2(x) - \mathcal{L}_1(x)\right)\right).$$

In the case of pure area preserving curvature motion (i.e., for $S = 0$), in the interest of even better accuracy, we propose slightly modified versions of Algorithms 1 and 2 that consist of exactly matching, at each time step $j\,(\delta t)$, the current area with the initial area. This strategy is similar to the one used by Ruuth and Wetton [37] in the case of threshold dynamics for area preserving curvature motion. For clarity of the exposition, we define the area operator $\mathcal{A} : C^2(\mathbb{R}^2, \mathbb{R}) \to \mathbb{R}$ applied to a smooth function ϕ to be

$$\mathcal{A}(\phi) := \int_{\phi(x)>0} dx = |\{x : \phi(x) > 0\}|,$$

which computes the area of the zero superlevel set of ϕ. We modify Algorithms 1 and 2 to use Newton's method at each time step to find λ_j^* the zero of the function $\lambda_j \mapsto a(\lambda_j) - a_0$, where a_0 is the initial area, and $a(\lambda_j)$ the current area at time step j defined as

$$a(\lambda_j) = \mathcal{A}(d_{j-1} * G_{\delta t} - \lambda_j).$$

Our expression for the average curvature used in Algorithms 1 and 2 is taken here to be the initial guess for the Newton iteration:

$$\lambda_j^0 = \frac{2\pi\ (\#\{d_{j-1} > 0\} - \#\{d_{j-1} < 0\} + 1)}{\int_{d_{j-1}>0} \Delta d_{j-1}(x)dx}(\delta t).$$

The signed distance function d_j is then constructed using

$$d_j = \mathbf{Redist}\left(d_{j-1} * G_{\delta t} - \lambda_j^*\right).$$

To modify Algorithm 2, we proceed in the same way; namely, at each time step $j\,(\delta t)$, we compute $\lambda_{j,1}^*$ and $\lambda_{j,2}^*$ coming from the result of a Newton iteration using d_{j-2}

ALGORITHM 3. ***Area Preserving Curvature Motion.*** *Given the initial set Σ_0 with area a_0 defined through its signed distance function $d_0(x)$, a time step $\delta t > 0$, and a tolerance $\eta > 0$, generate the sets Σ_j via their signed distance function $d_j(x)$ at the subsequent discrete times $t = j(\delta t)$ by alternating between the following two steps:*

1. *Using Newton's method with initial guess*

$$\lambda_j^0 = \frac{2\pi\ \left(\#\{d_{j-1} > 0\} - \#\{d_{j-1} < 0\} + 1\right)}{\int_{d_{j-1}>0} \Delta d_{j-1}(x)dx}(\delta t),$$

find λ_j^* such that $\left|\mathcal{A}\left(d_{j-1} * G_{\delta t} - \lambda_j^*\right) - a_0\right| < \eta$ and form

$$\mathcal{L}(x) = \left(d_{j-1} * G_{\delta t}\right)(x) - \lambda_j^*.$$

2. *Construct the signed distance function d_j using*

$$d_j(x) = \mathbf{Redist}(\mathcal{L}(x)).$$

with time step $2\delta t$ and d_{j-1} with time step δt, respectively. The final update in this case becomes

$$d_j = \mathbf{Redist}\left(\frac{1}{3}\left(4\left(d_{j-1} * G_{\delta t} - \lambda_{j,2}^*\right) - \left(d_{j-2} * G_{\delta t} - \lambda_{j,1}^*\right)\right)\right). \tag{3.8}$$

The more accurate version of Algorithm 1 in the case of pure area preserving curvature motion is given in Algorithm 3, and a multistep version of Algorithm 3 is provided in Algorithm 4. This multistep version follows the same pattern as Algorithm 2 and uses the update (3.8). Note that although Algorithms 3 and 4 preserve the area more accurately than Algorithms 1 and 2, they will be inapplicable in situations where the total area may change due to the presence of a nonzero forcing term S.

Remark. The algorithms for area preserving curvature motion presented in this paper are specific to two dimensions. Indeed, unlike curvature for curves, the integral of mean curvature for surfaces does not have a simple topological interpretation that would simplify into simple formulas like (3.6) and (3.7). Consequently, although a version of our area preserving curvature motion algorithm using signed distance function can be obtained in three dimensions and higher, the form of such an algorithm wouldn't be as simple as Algorithms 1, 2, 3, and 4 described in this paper.

3.3. Numerical convergence study. In this section we describe some convergence studies done with the algorithms for area preserving curvature motion introduced in the previous section. In the computations presented below, we used a second-order accurate procedure to construct the signed distance function in a tubular neighborhood of the interface. For details on more sophisticated algorithms for constructing signed distance functions, we refer the reader to [5, 6, 30, 38, 40, 43]. We emphasize again here that even though d is only a signed distance function in a tubular neighborhood of the interface, (3.7) remains true. Whether the function d has kinks in the interior of the domain or not is immaterial since the behavior of d inside the domain (away from the interface) has no effect on the result. In particular, this is true for any grid function, whether it is obtained from sampling a smooth or singular function.

ALGORITHM 4. ***Area Preserving Curvature Motion: Second-order in time version.*** *Given Σ_0 with area a_0 having distance function $d_0(x)$, $\delta t > 0$, and $\eta > 0$, generate Σ_j at times $t = j(\delta t)$ by alternating between the following two steps:*

1. *Using Newton's method with initial guesses*

$$\lambda_{j,1}^0 = \frac{2\pi \ (\#\{d_{j-2} > 0\} - \#\{d_{j-2} < 0\} + 1)}{\int_{d_{j-2}>0} \Delta d_{j-2}(x)dx}(\delta t)$$

and

$$\lambda_{j,2}^0 = \frac{2\pi \ (\#\{d_{j-1} > 0\} - \#\{d_{j-1} < 0\} + 1)}{\int_{d_{j-1}>0} \Delta d_{j-1}(x)dx}(\delta t),$$

find $\lambda_{j,1}^*$ and $\lambda_{j,2}^*$ such that $\left|\mathcal{A}\left(d_{j-2} * G_{\delta t} - \lambda_{j,1}^*\right) - a_0\right| < \eta$ and $\left|\mathcal{A}\left(d_{j-1} * G_{\delta t} - \lambda_{j,2}^*\right) - a_0\right| < \eta$, and form

$$\mathcal{L}_1(x) = \left(d_{j-2} * G_{\delta t}\right)(x) - \lambda_{j,1}^*,$$
$$\mathcal{L}_2(x) = \left(d_{j-1} * G_{\delta t}\right)(x) - \lambda_{j,2}^*.$$

2. *Construct the signed distance function d_j using*

$$d_j(x) = \textbf{Redist}\left(\frac{1}{3}\left(4\mathcal{L}_2(x) - \mathcal{L}_1(x)\right)\right).$$

We start by investigating the convergence of Algorithms 1 and 2 in the case $S = 0$, namely for pure area preserving curvature motion. The results are displayed in Tables 1 and 2. The initial condition for these tests is an ellipse $\mathcal{E}$ with major axis $a = 0.45$ and minor axis $b = 0.2$, and the computational domain is $[0, 1]^2$. The evolution is computed over the time interval $[0, 0.01]$. The major and minor axes of the final curve (which is no longer an ellipse) are 0.4 and 0.22, respectively. At the final time $T = 0.01$ we measure the quantity $\int_{\Sigma(T)}(x^2 + y^2)d\Omega$, where $\Sigma(t) \in \mathbb{R}^2$ evolves under area preserving curvature motion with initial condition $\Sigma(0) = \mathcal{E}$. This integral quantity is used since it corresponds to the second moment of the evolving shape. We compare the computed integral with the exact one $\int_{\Sigma_e(T)}(x^2 + y^2)d\Omega$, where the exact evolution $\Sigma_e(t)$ is computed using a front tracking technique with a very fine discretization of the parameterized curve. We also display the error in area and its associated convergence rate.

As can be seen in Table 1, Algorithm 1 settles into a clearly first-order convergence rate. Table 2 shows the convergence rate for Algorithm 2, which in this example turns out to be significantly higher than second order, perhaps due to some cancellation of errors. In any case, Algorithm 2 achieves very high accuracy even on very modest sized grids.

We now present the convergence of Algorithms 3 and 4 with the same initial condition (same ellipse). The results are displayed in Tables 3 and 4, respectively. We repeat that these algorithms can be used only to simulate pure area preserving motion by curvature. In both convergence tests, the area was preserved up to an error of 10^{-14} or less. Table 3 displays the convergence of Algorithm 3 and Table 4

TABLE 1

Convergence of Algorithm 1 *for* $S = 0$. *The initial condition is an ellipse with major axis* $a = 0.45$ *and minor axis* $b = 0.2$ *on* $[0,1]^2$. *The evolution was computed for* $t \in [0, 0.01]$.

Resolution	# of time steps	Relative error in $\int_{\Sigma(T)} (x^2+y^2)\, d\Omega$ (in %)	Order	Error in area	Order
33×33	20	0.0167	–	0.000921	–
65×65	40	0.3218	−4.27	0.000561	0.72
129×129	80	0.2046	0.65	0.000271	1.05
257×257	160	0.1002	1.03	0.00012	1.18
513×513	320	0.0487	1.04	0.000055	1.12
1025×1025	640	0.0240	1.02	0.0000263	1.06
2049×2049	1280	0.0119	1.01	0.0000128	1.03

TABLE 2

Convergence of Algorithm 2 *for* $S = 0$. *The initial condition is an ellipse with major axis* $a = 0.45$ *and minor axis* $b = 0.2$ *on* $[0,1]^2$. *The evolution was computed for* $t \in [0, 0.01]$.

Resolution	# of time steps	Relative error in $\int_{\Sigma(T)} (x^2+y^2)\, d\Omega$ (in %)	Order	Error in area	Order
33×33	20	0.8098	–	0.00164	–
65×65	40	0.4919	0.72	0.000710	1.20
129×129	80	0.0951	2.37	0.000150	2.24
257×257	160	0.00738	3.69	0.0000210	2.84
513×513	320	0.000928	2.99	0.00000434	2.27
1025×1025	640	0.000196	2.25	0.00000109	1.99
2049×2049	1280	0.00000915	4.42	0.000000265	2.04

TABLE 3

Convergence of Algorithm 3. *The initial condition is an ellipse with major axis* $a = 0.45$ *and minor axis* $b = 0.2$ *on* $[0,1]^2$. *The evolution was computed for* $t \in [0, 0.01]$.

Resolution	# of time steps	Relative error in $\int_{\Sigma(T)} (x^2+y^2)\, d\Omega$ (in %)	Order
33×33	20	0.6256	–
65×65	40	0.0588	3.41
129×129	80	0.0213	1.47
257×257	160	0.0194	0.13
513×513	320	0.0117	0.73
1025×1025	640	0.00631	0.89
2049×2049	1280	0.00327	0.95

the convergence of Algorithm 4. Note that Algorithms 3 and 4 achieve very good accuracy even with quite few large time steps (i.e., wide Gaussians).

In addition to the convergence studies described above, we tested our area preserving curvature motion algorithm on an initial configuration containing three circles with radii 0.15, 0.2, and 0.22 on $[0,1]^2$. Since circles remain circles under this motion, we monitored the evolution of each of the three radii for $t \in [0, 0.1]$ and compared it with the exact evolution obtained by numerical integration of the coupled ODEs for the radii. The circles were placed far apart initially so that no collision occurred during the evolution. The result is shown in Figure 1. Figure 2 shows the three circles in the initial condition (thick line) and the final curves at time $t = 0.1$ (fine line). Note that area preserving curvature motion is global and hence couples the evolving circles to each other.

Finally, we carried out a convergence study where we measured the decay of error in the position of two circles (radius of the circles). The results are displayed in Ta-

TABLE 4

Convergence of Algorithm 4. The initial condition is an ellipse with major axis $a = 0.45$ and minor axis $b = 0.2$ on $[0,1]^2$. The evolution was computed for $t \in [0, 0.01]$.

Resolution	# of time steps	Relative error in $\int_{\Sigma(T)} (x^2 + y^2)\, d\Omega$ (in %)	Order
33×33	20	0.3040	–
65×65	40	0.0112	4.76
129×129	80	0.00735	0.61
257×257	160	0.00689	0.09
513×513	320	0.00197	1.81
1025×1025	640	0.000488	2.01
2049×2049	1280	0.000117	2.06

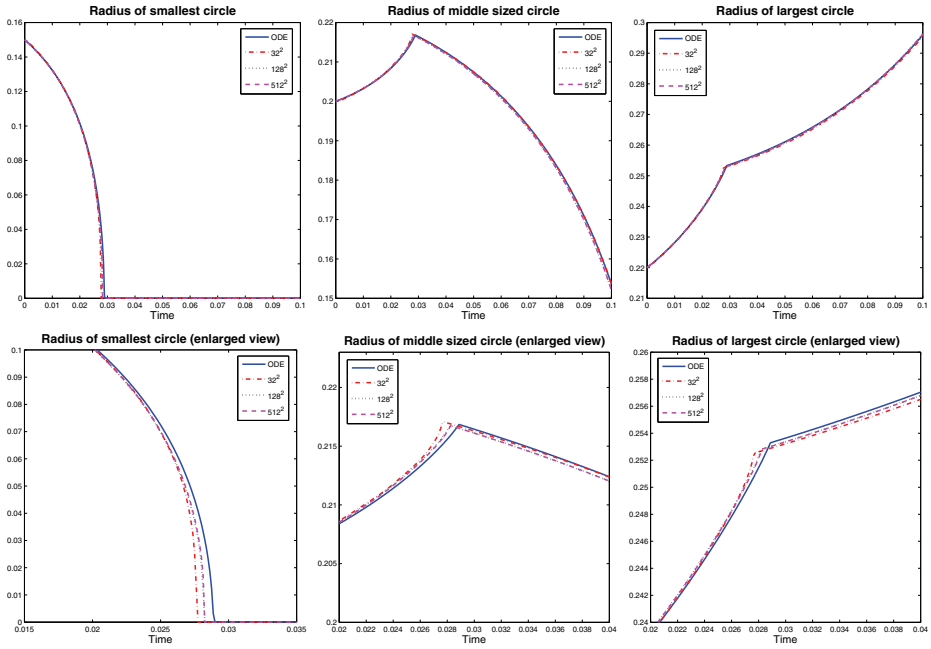


FIG. 1. *Comparison between the evolutions of the exact and computed radii obtained from three initial circles taken as the initial condition and evolved under area preserving curvature motion. The initial three circles have the following radii:* 0.15, 0.2, *and* 0.22. *In each of the plots, we superimpose the exact evolution (in bold) and the evolution of the radii computed on a* 32^2, *a* 128^2, *and a* 512^2 *grid. The top row shows the entire evolution, while the bottom row shows an enlarged view of the plots where the graph is not differentiable (corresponding to the disappearance of the smallest circle). This computation was performed with Algorithm* 4.

bles 5 and 6 obtained with Algorithms 2 and 4, respectively. As in the previous ellipse example, Table 5 shows that Algorithm 2 settles into a clear first-order convergence rate, whereas Table 6 shows a convergence rate for Algorithm 4 that turns out to be a little bit higher than second order, probably because of some error cancellation.

Finally we present the convergence of Algorithms 1 and 2 in a case where the forcing term S is nonzero. In these tests, a circle with radius 0.25 on $[0,1]^2$ taken as the initial condition is evolved with normal velocity $v_N = \kappa - \bar{\kappa} + S$, where $S = \frac{1}{r(t)^2}$, and $r(t)$ is the radius of the circle at time t. Under this motion, the circle will expand throughout the evolution. This simple case allows us to obtain the exact evolution

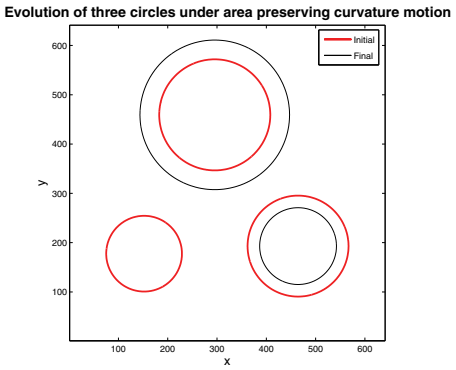


FIG. 2. *Evolution of three circles under area preserving curvature motion for* $t \in [0, 0.1]$. *The initial condition is shown by the thick curves, and the final curves at* $t = 0.1$ *are displayed by the thin curves. In this configuration, the largest circle grew while the other two shrunk (the smallest one actually disappeared).*

for the radius of the circle, which reads as $r_e(t) = \left(3t + r_0^2\right)^{\frac{1}{3}}$. We computed the evolution from $t = 0$ to $t = 0.0038$, i.e., from an initial radius of 0.25 until a radius of 0.3. The error is measured in the relative difference between the exact area of the circle at the final time of the evolution and the computed area at the final time. The results are displayed in Tables 7 and 8.

TABLE 5

Convergence of Algorithm 2 *in an example where the initial condition consists of two circles on* $[0, 1]^2$ *with radius* 0.15 *and* 0.24, *respectively. The area preserving evolution was computed for* $t \in [0, 0.01]$.

Resolution	# of time steps	Rel. error in radius 1 (in %)	Order	Rel. error in radius 2 (in %)	Order
33×33	5	3.7017	–	1.1133	–
65×65	10	1.7716	1.06	0.4756	1.23
129×129	20	0.7408	1.26	0.2002	1.25
257×257	40	0.3401	1.12	0.0921	1.12
513×513	80	0.1634	1.06	0.0443	1.06
1025×1025	160	0.0802	1.03	0.0217	1.03
2049×2049	320	0.0397	1.01	0.0108	1.01

TABLE 6

Convergence of Algorithm 4 *in an example where the initial condition consists of two circles on* $[0, 1]^2$ *with radius* 0.15 *and* 0.24, *respectively. The area preserving evolution was computed for* $t \in [0, 0.01]$.

Resolution	# of time steps	Rel. error in radius 1 (in %)	Order	Rel. error in radius 2 (in %)	Order
33×33	5	2.2671	–	0.6135	–
65×65	10	0.1300	4.12	0.0352	4.12
129×129	20	0.0550	1.24	0.0150	1.23
257×257	40	0.0115	2.26	0.0031	2.25
513×513	80	0.0023	2.33	0.000653	2.27
1025×1025	160	0.000395	2.53	0.000142	2.20
2049×2049	320	0.0000139	4.83	0.0000388	1.87

TABLE 7

Convergence of Algorithm 1 *with a radially symmetric forcing term* $S = \frac{1}{r^2}$. *The initial condition is a circle with radius* $r_0 = 0.25$ *on* $[0,1]^2$.

Resolution	# of time steps	Relative error in area (in %)	Order
33×33	10	0.4742	–
65×65	20	0.2708	0.81
129×129	40	0.1472	0.88
257×257	80	0.0764	0.95
513×513	160	0.0389	0.97
1025×1025	320	0.0196	0.99
2049×2049	640	0.00985	0.99

TABLE 8

Convergence of Algorithm 2 *with a radially symmetric forcing term* $S = \frac{1}{r^2}$. *The initial condition is a circle with radius* $r_0 = 0.25$ *on* $[0,1]^2$.

Resolution	# of time steps	Relative error in area (in %)	Order
33×33	10	0.1853	–
65×65	20	0.0525	1.82
129×129	40	0.01282	2.03
257×257	80	0.003146	2.03
513×513	160	0.000800	1.97
1025×1025	320	0.000191	2.07
2049×2049	640	0.0000398	2.26

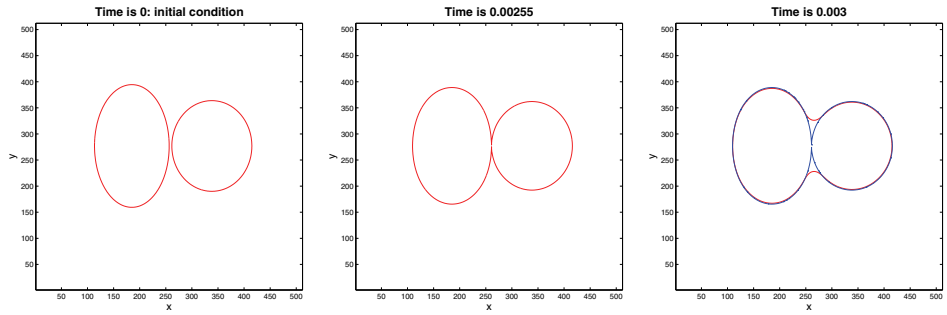
4. Application: Large scale simulations of area preserving curvature motion. In this section, we demonstrate the capacity of our proposed algorithms to handle large scale simulations with very good accuracy.

4.1. Curve shortening at various area fractions. Geometrically in two dimensions, the area preserving curvature flow describes the shortening of a curve (or interface) separating two phases, while maintaining the area of each phase equal to their respective initial area. A natural question therefore arises: at what rate does the total length of the curve decrease? Scaling arguments [7] suggest that the total length L decreases as a power law in time according to

$$L(t) \sim t^{-\frac{1}{2}}.$$

In a recent paper, Dai [7] obtained a rigorous result for the rate of decrease of L in the case of a dilute mixture. Specifically, he showed that for a collection of nonintersecting and convex plane curves, the total length $L(t)$ cannot decrease faster than $t^{-\frac{1}{2}}$ in a time average sense. In this simplified case, there is no coalescence and the only singularity is the disappearance of curves. In the general case, however, collisions of curves will occur causing singularities in the curvature to appear at the times of first intersections. In fact, at the points of intersections, the curvature will be infinite, leading to an immediate smoothing and a fast decay of the sum of the lengths pertaining to the merging curves. Figure 3 illustrates this point in a simple example of two curves intersecting each other during their evolution under area preserving curvature motion.

In this context, we refer to the phase enclosed by the curves as Phase 1. Phase 2 denotes its complement. As a demonstration of the proposed algorithms, we present some simulations of area preserving curvature motion on very large collections of



(a) (b) (c)

FIG. 3. *Coalescence of two curves during their evolution under area preserving curvature motion.* (a) *shows the initial condition and* (b) *displays the new curve just as the two previous ones collided. The curvature of the new curve is very large at the point of intersection.* (c) *shows the new curve shortly after the collision occurred superimposed with the curve shown in* (b).

closed curves (or droplets). We also measure certain statistics related to the configurations of droplets during their evolution. Our algorithms achieve good accuracy with droplets as small as 15 pixels in diameter, allowing us to carry out simulations with a very large number of initial droplets. Indeed, we present simulations with initial conditions containing up to 25000 droplets on a 4097^2 grid. Our computations considered various area fractions of Phase 1 ranging from 10% to 50%, which equivalently considered area fractions of Phase 2 ranging from 50% to 90%.

4.2. Numerical results. In this section, we present the results of our simulations. We construct the initial data by generating random sets of points from a uniform distribution and placing a disk centered at each of the points with radius r_d chosen from a uniform distribution. The droplets are obtained by taking the union of the disks (some of which may be overlapping). From such initial configurations (i.e., randomly generated droplet configurations), it would be reasonable to expect a certain collision rate during the evolution under area preserving curvature motion. Figure 4 illustrates the area preserving motion by curvature evolution on initial data with 10% and 40% area fraction. To avoid boundary effects, the computations are done on a domain slightly larger than $[0,1]^2$. Additionally, as an extra precaution to prevent premature mergings of nearby droplets, at each time step we divide the droplets into subsets containing only droplets that are far enough apart and update the signed distance function of each subset separately. This allows individual grains to evolve independently all the way up to the time of overlap with another evolving droplet that may be as close as a single grid point. Finally, since the average size of the droplets increases during the evolution (slowing down the dynamics), we perform our computations using an adaptive time step regulated by the average size of the droplets; namely, the time step size increases with the average size of the droplets as the evolution progresses. This adaptivity is made possible by the unconditional stability of our algorithms and is necessary to perform computations across such a large difference in scales. In Figure 5, we demonstrate that taking larger time steps once the average grain size is large enough does not significantly change the configurations of droplets.

Figure 6 compares the rate of decrease of the total length L (which is also the energy dissipated by the evolution) with the theoretical bound $t^{-\frac{1}{2}}$ for various area frac-

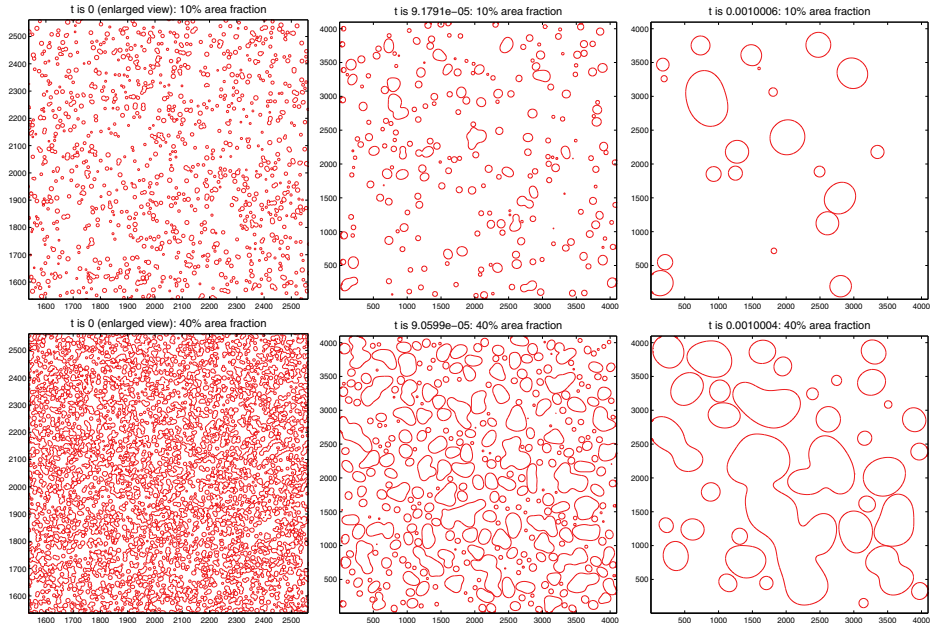
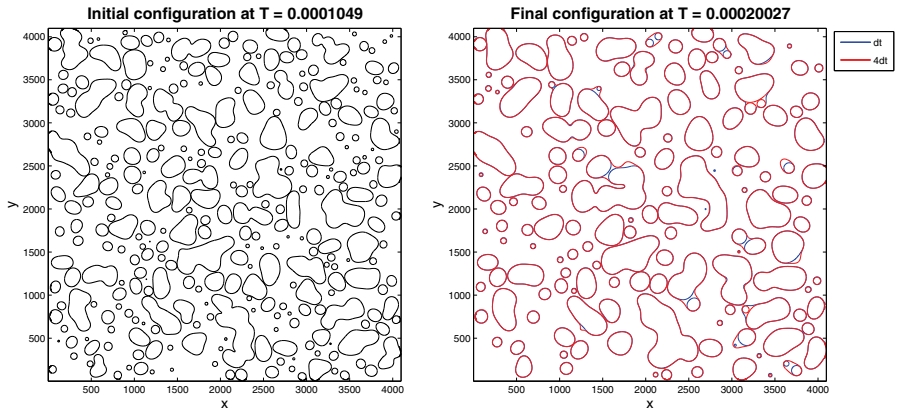


FIG. 4. *Evolution of two configurations of droplets under area preserving curvature motion. The top row displays the evolution of a configuration of droplets with* 10% *area fraction. The bottom row shows the evolution for a configuration with* 40% *area fraction. Because of the very large number of droplets in the early configurations, we show only a subset of these configurations in the two plots of the first column. These subsets have been enlarged for a better view.*

tions. Note that this bound has been rigorously established [7] for convex curves that never collide. Moreover, it is known [8] that it is possible to construct configurations of droplets that coarsen arbitrarily fast under this geometric motion. Nevertheless, for randomly generated initial conditions, the rate of decrease of L, obtained from our computations, closely follows the theoretical bound despite frequent collisions between droplets.

Another quantity that we study is the number of connected components K in the configurations of droplets. Based on the rate of decay of L, and using a simple heuristic argument on a uniform configuration of disks, we can show that the number of connected components should essentially decay as $\frac{1}{t}$. In Figure 7 we compare the numerically observed rate of decrease of K for various area fractions with the bound $\frac{1}{t}$. For the randomly generated initial conditions chosen in our simulations, the plots show very good agreement between the computed rate and the bound $\frac{1}{t}$.

Intuition also suggests that at any given time, a certain population of droplets will be just about to collide, generating configurations at which the energy decrease rate must be elevated (the proportion of droplets just about to collide to all droplets would of course depend on the area fraction). However, it appears that even if collisions between droplets cause a deviation in the coarsening rate of randomly generated initial configurations, our numerical experiments, despite their large size, are still not large enough to discern such a difference—perhaps the effect is very small. Nevertheless, as one would expect, we observe that *at any time* during the evolution (outside of the transient initial period and the final stage where only a few droplets remain), there is a constant proportion of eccentric droplets (i.e., droplets that are the result of a recent collision). This observation agrees with the expectation that collisions occur



(a) (b)

FIG. 5. *Comparison of the effect of two different time step sizes on the computed solution: fixed time step size of δt, and the adaptive time step size that reaches $4\delta t$ at the initial condition shown in* (a). *In this example, $\delta t = 4.7710^{-7}$.* (a) *shows the initial configuration and* (b) *displays the configurations obtained from computations using δt and $4\delta t$. Except for minor differences, both computations are able to resolve the dynamics, thus exonerating the use of larger time steps as the coarsening proceeds.*

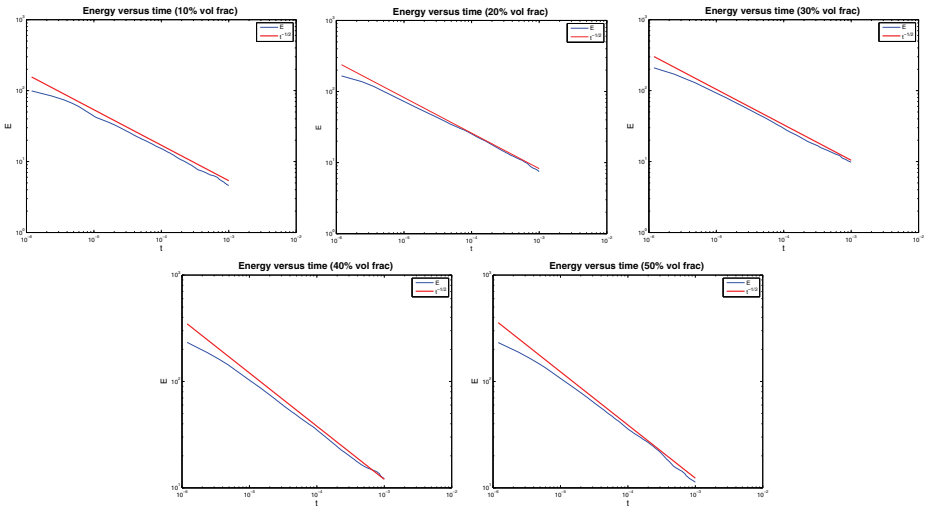


FIG. 6. *Loglog plot of the total length L of the boundary of all droplets (also the energy being decreased by the evolution) versus time for various area fractions. The thick line corresponds to the theoretical bound $t^{-\frac{1}{2}}$. From top to bottom, and left to right, the plots correspond to area fractions ranging from* 10% *to* 50%. *The plots acknowledge good agreement between the theoretical bound $t^{-\frac{1}{2}}$ and the numerically observed rates (fine line).*

at a definite rate in proportion to the number of droplets. To exhibit this behavior, we measure the isoperimetric ratio

$$I(C) := \frac{P^2}{A} \tag{4.1}$$

of each grain in order to characterize their shape. In (4.1), C is a closed curve, P is

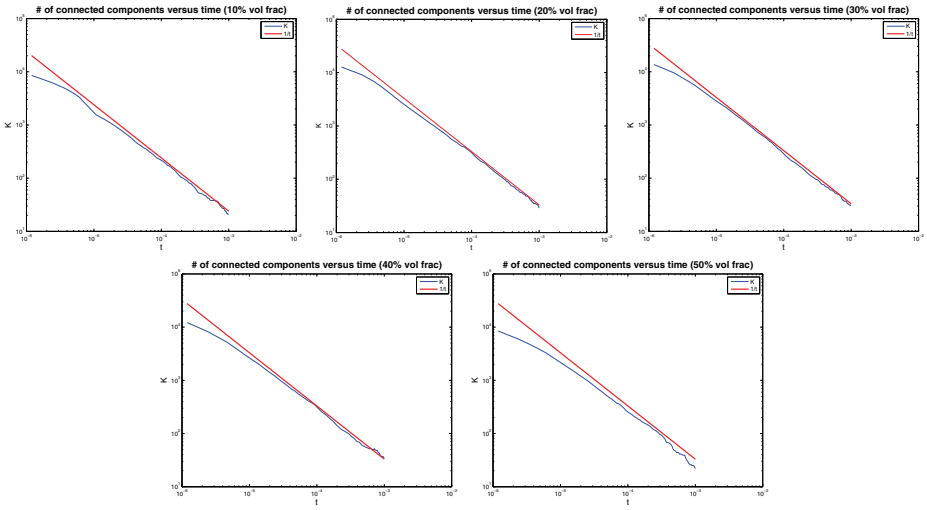


FIG. 7. *Loglog plot of the number of connected components K versus time for various area fractions. From top to bottom, and left to right, the plots correspond to area fractions ranging from* 10% *to* 50%. *The plots acknowledge good agreement between the bound* $\frac{1}{t}$ *(thick line) and the numerically observed rates (fine line).*

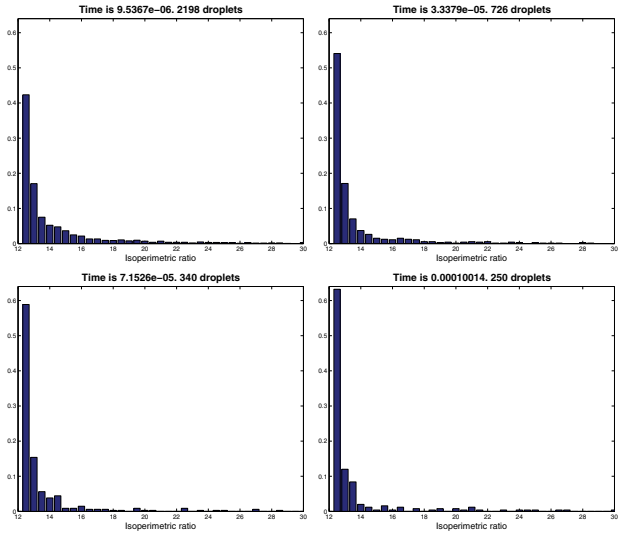


FIG. 8. *Time evolution of the distribution of isoperimetric ratios computed from the evolution of an initial configuration of droplets with* 50% *area fraction under area preserving curvature motion. During the evolution, the distribution of isoperimetric ratios remains quite wide. This width underlines the existence, at all times, of a certain proportion of droplets that resulted from collisions.*

its perimeter, and A is its area. Since the isoperimetric ratio is minimized by a circle, we have that for any closed curve

$$I \geq 4\pi \sim 12.57,$$

where $I = 4\pi$ when C is a circle. For an ellipse with minor axis b and major axis $a = 3b$, the isoperimetric ratio is approximately $I \sim 18.95$. For a more elongated ellipse

TABLE 9

Proportion of eccentric droplets (the isoperimetric ratio of which satisfies $I > 20$) for various area fractions. The proportion of eccentric droplets increases with the area fraction, as one would expect.

Area fraction	Proportion of eccentric droplets	# of droplets at onset of constant proportion	# of droplets at the end of constant proportion
10%	0.12%	5054	193
20%	0.46%	2623	78
30%	1.18%	2932	91
40%	2.49%	2766	34
50%	3.81%	3885	49

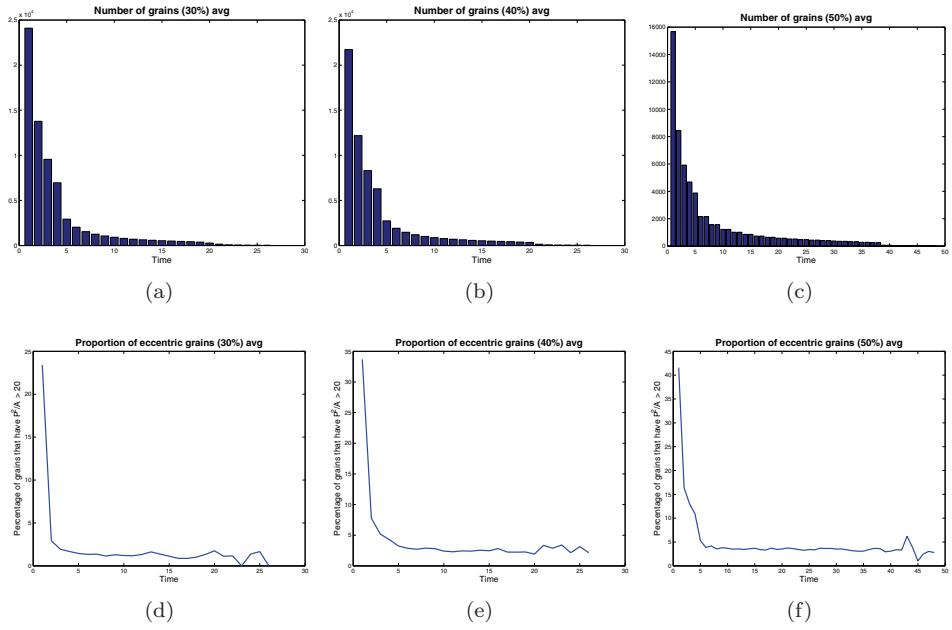


FIG. 9. *Proportions of eccentric droplets (i.e., with isoperimetric ratio $I > 20$) in the case of* 30%, 40%, *and* 50% *area fraction.* (d), (e), *and* (f) *clearly show that after a transient initial time, the proportion of eccentric droplets seems to stabilize around a constant value.* (a), (b), *and* (c) *corroborate the fact that the proportion of eccentric droplets remains constant from the time when the total number of droplets is on average* 3000 *until the time when it is around* 50.

with minor axis b and major axis $a = 4b$, the isoperimetric ratio is approximately $I \sim 23.42$. From these references, we look at the proportion of droplets with isoperimetric ratio $I > 20$. Figure 8 shows the distribution of isoperimetric ratios at different times throughout the evolution for a configuration of droplets with 50% area fraction. Table 9 displays the proportion of eccentric droplets for the various area fractions studied in our computations. Figure 9 shows the proportion of eccentric droplets and the total number of droplets for configurations with area fractions 30%, 40%, and 50%. In each case, the proportion of eccentric droplets decreases very quickly during the transient initial phase (as is the total number of droplets) and then stabilizes itself around a constant value. The proportion of eccentric droplets remains constant until the total number of droplets becomes too small.

5. Conclusion. We described efficient and highly accurate algorithms for simulating certain area preserving curvature flows in the plane, with normal speeds of the form $v_N = \kappa - \bar{\kappa} + S$. Our schemes are based on a diffusion generated motion approach using signed distance functions, thus making them more accurate than the standard MBO (threshold dynamics) schemes on uniform grids. We proposed first-order and second-order in time versions of our algorithms and carried out numerical studies to verify their convergence and accuracy. These schemes are also relevant for image processing applications where the additional term S in the velocity $\kappa - \bar{\kappa} + S$ comes from data fitting terms in variational models.

In addition, we presented an application of these new schemes to large scale simulations of coarsening. This application demonstrated the ability of our algorithms to handle large scale computations due to their high accuracy and computational efficiency. In particular, we were able to simulate the coarsening of large numbers of droplets under area preserving curvature motion. The large number of droplets allowed us to obtain reliable statistical measurements, which are reported.

REFERENCES

[1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, *Axioms and fundamental equations of image processing*, Arch. Rational Mech. Anal., 123 (1993), pp. 199–257.

[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 (1995), pp. 484–500.

[3] L. Bronsard and B. Stoth, *Volume-preserving mean curvature flow as a limit of a nonlocal Ginzburg–Landau equation*, SIAM J. Math. Anal., 28 (1997), pp. 769–807.

[4] L. Bronsard and B. Wetton, *A numerical method for tracking curve networks moving with curvature motion*, J. Comput. Phys., 120 (1993), pp. 66–87.

[5] A. Chambolle and M. Novaga, *Approximation of the anisotropic mean curvature flow*, Math. Models Methods Appl. Sci., 17 (2007), pp. 833–844.

[6] L.-T. Cheng and Y.-H. Tsai, *Redistancing by flow time dependent Eikonal equation*, J. Comput. Phys., 227 (2008), pp. 4002–4017.

[7] S. Dai, *On the shortening rate of collections of plane convex curves by the area-preserving mean curvature flow*, SIAM J. Math. Anal., 42 (2010), pp. 323–333.

[8] S. Dai, B. Niethammer, and R. Pego, *Crossover in coarsening rates for the monopole approximation of the Mullins-Sekerka model with kinetic drag*, Proc. Roy. Soc. Edinburgh Sect. A, 140 (2010), pp. 553–571.

[9] M. C. Delfour and J.-P. Zolesio, *Shapes and Geometries: Analysis, Differential Calculus and Optimization*, Adv. Des. Control 4, SIAM, Philadelphia, 2001.

[10] I. C. Dolcetta, S. F. Vita, and R. March, *Area preserving curve shortening flows: From phase separation to image processing*, Interfaces Free Bound., 4 (2002), pp. 325–343.

[11] S. Esedoḡlu, S. Ruuth, and R. Tsai, *Diffusion generated motion using signed distance functions*, J. Comput. Phys., 229 (2010), pp. 1017–1042.

[12] S. Esedoḡlu, S. J. Ruuth, and R. Tsai, *Threshold dynamics for high order geometric motions*, Interfaces Free Bound., 10 (2008), pp. 263–282.

[13] L. Evans, *Convergence of an algorithm for mean curvature motion*, Indiana Univ. Math. J., 42 (1993), pp. 553–557.

[14] H. Federer, *Curvature measures*, Trans. Amer. Math. Soc., 93 (1959), pp. 418–491.

[15] M. Gage, *Curve shortening makes convex curves circular*, Invent. Math., 76 (1984), pp. 357–364.

[16] M. Gage, *On an area-preserving evolution equation for plane curves*, in Nonlinear Problems in Geometry, Contemp. Math. 51, AMS, Providence, RI, 1986, pp. 51–62.

[17] M. Gage and R. Hamilton, *The heat equation shrinking convex plane curves*, J. Differential Geom., 23 (1986), pp. 69–96.

[18] D. Golovaty, *The volume preserving motion by mean curvature as an asymptotic limit of reaction-diffusion equations*, Quart. Appl. Math., 55 (1997), pp. 243–298.

[19] G. Huisken, *Flow by mean curvature of convex surfaces into spheres*, J. Differential Geom., 20 (1984), pp. 237–266.

[20] I. M. Lifshitz and V. V. Slyozov, *The kinetics of precipitation from supersaturated solid*

solutions, J. Phys. Chem. Solids, 19 (1961), pp. 35–50.
[21] P. Mascarenhas, *Diffusion Generated Motion by Mean Curvature*, Technical report, UCLA CAM Report 92-33, UCLA, Los Angeles, CA, 1992.
[22] B. Merriman, J. K. Bence, and S. Osher, *Diffusion generated motion by mean curvature*, in Proceedings of the Computational Crystal Growers Workshop, J. E. Taylor, ed., AMS, Providence, RI, 1992, pp. 73–83.
[23] B. Merriman, J. K. Bence, and S. Osher, *Motion of multiple junctions: A level set approach*, J. Comput. Phys., 112 (1994), pp. 334–363.
[24] W. W. Mullins, *Two-dimensional motion of idealized grain boundaries*, J. Appl. Phys., 27 (1956), pp. 900–904.
[25] B. Niethammer, *The mathematics of Ostwald ripening*, in Geometric Analysis and Nonlinear Differential Equations, S. Hildebrandt and H. Karcher, eds., Springer, Berlin, 2002, pp. 649–663.
[26] R. H. Nochetto, M. Paolini, and C. Verdi, *A dynamic mesh algorithm for curvature dependent evolving interfaces*, J. Comput. Phys., 123 (1996), pp. 296–310.
[27] S. Osher and J. A. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[28] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. Kang, *A PDE-based fast local level set method*, J. Comput. Phys., 155 (1999), pp. 410–438.
[29] J. Rubinstein and P. Sternberg, *Nonlocal reaction-diffusion equations and nucleation*, IMA J. Appl. Math., 48 (1992), pp. 248–264.
[30] G. Russo and P. Smereka, *A remark on computing distance functions*, J. Comput. Phys., 163 (2000), pp. 51–67.
[31] S. J. Ruuth, *A diffusion generated approach to multiphase motion*, J. Comput. Phys., 145 (1998), pp. 166–192.
[32] S. J. Ruuth, *Efficient algorithms for diffusion-generated motion by mean curvature*, J. Comput. Phys., 144 (1998), pp. 603–625.
[33] S. J. Ruuth and B. Merriman, *Convolution-generated motion and generalized Huygens' principles for interface motion*, SIAM J. Appl. Math., 60 (2000), pp. 868–890.
[34] S. J. Ruuth and B. Merriman, *Convolution-thresholding methods for interface motion*, J. Comput. Phys., 169 (2001), pp. 678–707.
[35] S. J. Ruuth, B. Merriman, and S. Osher, *Convolution generated motion as a link between cellular automata and continuum pattern dynamics*, J. Comput. Phys., 151 (1999), pp. 836–861.
[36] S. J. Ruuth, B. Merriman, J. Xin, and S. Osher, *Diffusion-generated motion by mean curvature for filaments*, J. Nonlinear Sci., 11 (2001), pp. 473–493.
[37] S. J. Ruuth and B. T. R. Wetton, *A simple scheme for volume-preserving motion by mean curvature*, J. Sci. Comput., 19 (2003), pp. 373–384.
[38] J. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Nat. Acad. Sci. U.S.A., 93 (1996), pp. 1591–1595.
[39] P. Smereka, *Semi-implicit level set methods for curvature and surface diffusion motion*, J. Sci. Comput., 19 (2003), pp. 439–456.
[40] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, *Fast sweeping methods for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694.
[41] J. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.
[42] C. Wagner, *Theorie der Alterung von Niederschlägen durch Umlösen*, Z. Elecktrochemie, 65 (1961), pp. 581–594.
[43] H. Zhao, *A fast sweeping method for Eikonal equations*, Math. Comp., 74 (2005), pp. 603–627.
[44] H.-K. Zhao, B. Merriman, S. Osher, and L. Wang, *Capturing the behavior of bubbles and drops using the variational level set approach*, J. Comput. Phys., 143 (1998), pp. 495–518.