A Variational Shape Optimization Approach for Image Segmentation with a Mumford-Shah Functional

Günay Doğam
University of Pennsylvania, gunayd@gmail.com

Pedro Morin
Universidad Nacional del Litoral

Ricardo H. Nochetto
University of Maryland - College Park

A Variational Shape Optimization Approach for Image Segmentation with a Mumford-Shah Functional

Abstract
We introduce a novel computational method for a Mumford–Shah functional, which decomposes a given image into smooth regions separated by closed curves. Casting this as a shape optimization problem, we develop a gradient descent approach at the continuous level that yields nonlinear PDE flows. We propose time discretizations that linearize the problem and space discretization by continuous piecewise linear finite elements. The method incorporates topological changes, such as splitting and merging for detection of multiple objects, space–time adaptivity, and a coarse-to-fine approach to process large images efficiently. We present several simulations that illustrate the performance of the method and investigate the model sensitivity to various parameters.

Keywords
image segmentation, Mumford-Shah, shape optimization, finite element method

SIAM J. SCI. COMPUT.
Vol. 30, No. 6, pp. 3028–3049

A VARIATIONAL SHAPE OPTIMIZATION APPROACH FOR IMAGE SEGMENTATION WITH A MUMFORD–SHAH FUNCTIONAL[*]

GÜNAY DOĞAN[†], PEDRO MORIN[‡], AND RICARDO H. NOCHETTO[§]

Abstract. We introduce a novel computational method for a Mumford–Shah functional, which decomposes a given image into smooth regions separated by closed curves. Casting this as a shape optimization problem, we develop a gradient descent approach at the continuous level that yields nonlinear PDE flows. We propose time discretizations that linearize the problem and space discretization by continuous piecewise linear finite elements. The method incorporates topological changes, such as splitting and merging for detection of multiple objects, space–time adaptivity, and a coarse-to-fine approach to process large images efficiently. We present several simulations that illustrate the performance of the method and investigate the model sensitivity to various parameters.

Key words. image segmentation, Mumford–Shah, shape optimization, finite element method

AMS subject classifications. 49M15, 49M25, 65D15, 65K10, 68T45, 90C99

DOI. 10.1137/070692066

1. Introduction.

Image segmentation and image smoothing are two fundamental tasks in image processing. The goal of image segmentation is to extract uniform regions and their boundaries from given images, where uniformity is defined with respect to some image features, such as image intensity. Image smoothing, on the other hand, is the process of reducing the variation in image intensity. For example, we can perform image smoothing to reduce the noise content of a given degraded image. In general we would like to apply a *selective* version of image smoothing, only within uniform regions but not across the boundaries of the regions. The intrinsic connection between the image segmentation and denoising is thus apparent. In fact, one reasonable strategy would be to couple both problems and to pursue them simultaneously. This is exactly what the Mumford–Shah model does, and the main theme of this paper is a novel computational method to solve a certain variant of the Mumford–Shah model; see (1.2)–(1.3) and Figure 1.1.

Mumford and Shah propose the following optimization problem in [19]:

$$\min_{u,K}\left\{\frac{1}{2}\int_D (u-I)^2dx+\frac{\mu}{2}\int_{D\setminus K}|\nabla u|^2dx+\gamma\,\mathrm{length(K)}\right\}. \tag{1.1}$$

The goal of this minimization is to find the set of discontinuities K, or the edge set, and a piecewise smooth approximation u of a given image intensity function $I : D \subset \mathbb{R}^2 \to \mathbb{R}$. However, the problem is hard in this form, as the two variables

[*]Received by the editors May 15, 2007; accepted for publication (in revised form) April 23, 2008; published electronically October 13, 2008. This work was partially supported by NSF grants DMS-0505454 and INT-0126272.

http://www.siam.org/journals/sisc/30-6/69206.html

[†]School of Engineering and Applied Sciences, University of Pennsylvania, Philadelphia, PA 19104 (gdogan@seas.upenn.edu).

[‡]Departamento de Matemática, Facultad de Ingenieria Química, Universidad Nacional del Litoral, Instituto de Matemática Aplicada del Litoral, CONICET, Güemes 3450, S3000GLN Santa Fe, Argentina (pmorin@santafe-conicet.gov.ar, http://math.unl.edu.ar/~pmorin).

[§]Department of Mathematics and Institute for Physical Science and Technology, University of Maryland, College Park, MD 20742 (rhn@math.umd.edu, http://www.math.umd.edu/~rhn).

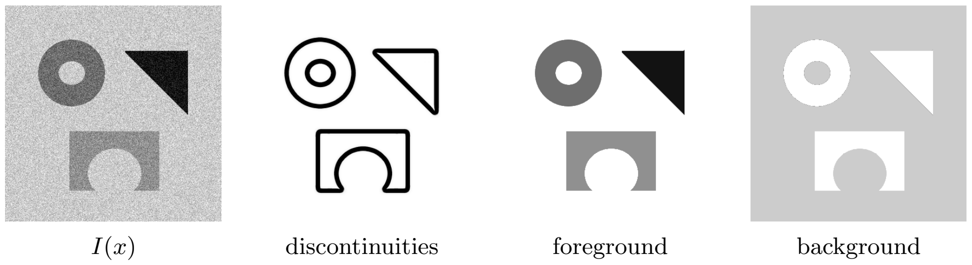


FIG. 1.1. *The Mumford–Shah model performs simultaneous segmentation and denoising of images. From left to right, the figures show the original noisy synthetic image, the set of discontinuities obtained, the denoised foreground, and the denoised background.*

u, K of the optimization are of very different natures; the edge set K may be quite general and may exhibit singularities. So a method that approximates (1.1) directly is yet to be found. Instead several attempts have been made to substitute (1.1) by close problems that are more suitable for computation. One idea, studied by Ambrosio and Tortorelli [1], is to approximate the edge set K with a diffuse edge set function v and use Γ-convergence to examine their relation. Finite element solutions for this approach have been investigated in [4], [13]. We stress that the diffuse interface approach focuses mostly on image smoothing because a decomposition of the image into distinct regions and region boundaries is not directly available. An alternative approach, proposed by Vese and Chan [24] and by Tsai, Yezzi, and Willsky [23], minimizes the following energy with respect to closed curves Γ:

$$J(\Gamma) = \sum_{i=1}^{2} \frac{1}{2} \left(\int_{\Omega_i} (u_i - I)^2 + \mu |\nabla u_i|^2 \right) dx + \gamma \int_{\Gamma} dS, \tag{1.2}$$

where u_i are obtained from

$$\begin{cases} -\mu \Delta u_i + u_i = I & \text{in} \quad \Omega_i, \\ \frac{\partial u_i}{\partial \nu_i} = 0 & \text{on} \quad \partial\Omega_i \end{cases} \tag{1.3}$$

with $i = 1, 2$. The curves Γ partition the image domain D into a foreground Ω_1 (inside the curves) and a background $\Omega_2 = D \backslash \Omega_1$ (outside the curves). Then the piecewise smooth approximation u is given by $u = u_1 \chi_{\Omega_1} + u_2 \chi_{\Omega_2}$.

With the Vese–Chan approach, we obtain a segmentation of the image into Γ, Ω_1, Ω_2. A simple example is depicted in Figure 1.1. Image smoothing happens as a by-product but is not the main emphasis. Moreover, the set of discontinuities is constrained to be simple closed curves; therefore, in its basic form, the method always yields a two-region or foreground–background segmentation. Vese and Chan [24] and Tsai, Yezzi, and Willsky [23] propose enhancements to handle cases with triple junctions and multiple regions. They represent Γ as a *level set* function and thus convert the problem into a curve evolution problem. Each iteration consists of solving the PDE (1.3) by finite differences and next moving the curve Γ in the L^2 gradient descent direction. This process is, however, computationally intensive. Hintermüller and Ring propose an inexact Newton-CG-type optimization scheme [18], resulting in significantly fewer iterations than the L^2 gradient descent, at the expense of additional computational cost per iteration.

In this paper, as in [18], [23], [24], we use the Mumford–Shah model (1.2)–(1.3) for image segmentation and also pursue a shape optimization approach. The main difference of our approach and those in [18], [23], [24] is that rather than using a level set function to represent Γ, Ω_1, Ω_2 implicitly, we use a polygon for Γ and conforming triangulations for Ω_1 and Ω_2 to represent them *explicitly.* This results in a Lagrangian shape optimization scheme. We formulate the problem in an infinite-dimensional setting, which yields nonlinear PDE flows. Motivated by computational efficiency, we discretize in time, thereby linearizing the problem, and then discretize in space by the finite element method (FEM) for additional flexibility. In the level set approach to Mumford–Shah, one performs the optimization on a uniform mesh, which typically coincides with the image grid. In contrast, we decouple the computational meshes from the image grid, which allows us to generate and modify meshes at our convenience. Typically our method deals with at most several thousands of unknowns per iteration. This translates into significant reduction in computation for large images. Furthermore, we incorporate a number of additional features into our algorithm, very beneficial for practical examples, resulting in an efficient and robust segmentation algorithm. These are the following:

- *Choice of descent directions.* Our method allows for easy incorporation of different descent directions, in addition to the L^2 descent. In particular an inexact Newton descent direction based on [18] can be adopted for faster convergence. These variations are typically major for the level set approach, in terms of both implementation and computation. Interestingly, for our method, a single inexact Newton step turns out to require less computation than the L^2 gradient step. This is in contrast with [18].
- *Space adaptivity.* As we do not discretize directly on the image grid, we need to avoid missing important image features. This is accomplished through image-based adaptivity. The mesh is refined where the image varies more but is kept coarse elsewhere. Furthermore, geometric adaptivity ensures that curves are represented accurately by concentrating nodes in regions of large curvature. Space adaptivity is crucial for balancing accuracy and computational cost.
- *Topological changes.* General-purpose image segmentation algorithms do not require a priori knowledge of the number or topology of objects in the image. Therefore, during the optimization process, curves should be able to merge or split to recover arbitrary optimal configurations. This is handled automatically by the level set method but not by explicit Lagrangian approaches. We implement a suitable topology surgery.
- *Multilevel optimization.* Two major issues in practice are computational cost and local minima. To address both of them at once, we introduce a heuristic multilevel strategy: we start with a relatively coarse discretization and tighten the error tolerances in space–time adaptivity as we approach a minimum. We found this strategy advantageous for both computational cost and avoiding local minima.

In addition to its algorithmic advantages, the explicit Lagrangian representation is by itself useful. It gives us direct access to the individual components forming the background and the foreground, and we can easily examine properties, such as shape, area, and other possible statistics. In the level set case, they are embedded in the level set function and we need to extract them first to do further processing. When the optimization terminates, we immediately have all the individual objects and the boundaries explicitly without any need for postprocessing. As a by-product, this also yields a sparse representation for the image: the curves, triangulations, and approximation u require significantly less storage than the input image.

The rest of the paper is organized as follows. In section 2, we review the basic shape differential calculus to deal with the Mumford–Shah functional (1.2). In section 3, we introduce the system of nonlinear PDEs that enables us to compute gradient descent directions for (1.2). We also describe the L^2 and weighted H^1 descent directions and their relative merits. In section 4, we discretize the system of PDEs from section 3, first in time and next in space. We investigate stability properties and describe the resulting linear systems and how to solve them. In section 5, we introduce procedures for time-step selection, topology surgery, space adaptivity, and the multilevel algorithm. We conclude our paper with several experiments in section 6. They document properties and performance of our variational method, as well as explore the model sensitivity to parameters μ and γ.

2. Shape sensitivity analysis.

We recall some definitions and results that will help us derive the gradient descent flows to implement the minimization of (1.2). Our main references for these are [9] and [22]. Let D be the image domain, an open and bounded set in $\mathbb{R}^2$. We denote by Γ the union of a finite set of simple closed curves of class C^2 and by Ω the domain enclosed by Γ (another possibility is to let Ω be the complement of the enclosed region). We define the *tangential gradient* $\nabla_\Gamma h$ of a given function $h \in C^2(D)$ as

$$\nabla_\Gamma h = \big(\nabla h - \partial_\nu h\, \nu\big)|_\Gamma,$$

where ν denotes the unit normal vector to Γ, pointing outward Ω. For $\vec{W} \in [C^1(\mathcal{D})]^2$, we define the *tangential divergence* of $\vec{W}$ by

$$\text{div}_\Gamma \vec{W} = \big(\text{div}\vec{W} - \nu \cdot D\vec{W} \cdot \nu\big)|_\Gamma, \tag{2.1}$$

where $D\vec{W}$ denotes the Jacobian matrix of $\vec{W}$. Given the Hessian D^2h of h and curvature κ of Γ, the *Laplace–Beltrami operator* Δ_Γ on Γ is defined as follows:

$$\Delta_\Gamma h = \text{div}_\Gamma(\nabla_\Gamma h) = \big(\Delta h - \nu \cdot D^2 h \cdot \nu - \kappa\, \partial_\nu h\big)|_\Gamma. \tag{2.2}$$

The velocity method. To evaluate the effect of a particular choice of deformation or velocity $\vec{V}$ on the energy $J(\Gamma)$ of a curve Γ, we need to state the relationship between $\vec{V}$ and the sequence of curves $\{\Gamma_t\}_{t\geq 0}$ it induces. Given $\vec{V}$, each point $x \in \Omega_0$ (the domain enclosed by the initial curve Γ_0) is continuously deformed by the system of (autonomous) ODEs defined by $\vec{V}$:

$$\frac{dx}{dt} = \vec{V}(x(t)) \quad \forall t \in [0,T], \qquad x(0) = X, \tag{2.3}$$

where $X \in \Omega_0 = \Omega$. This defines the mapping $x(t,\cdot) : X \in \Omega \to x(t,X) \in \mathbb{R}^2$ and also the deformed curves and domains

$$\Gamma_t = \{x(t,X) : \quad X \in \Gamma_0\}, \qquad \Omega_t = \{x(t,X) : \quad X \in \Omega_0\}. \tag{2.4}$$

This formalization is the basis of the *velocity method* in shape optimization and it will allow us to examine the effect of $\vec{V}$ on $J(\Gamma)$. It turns out that the *normal velocity* $V := \nu \cdot \vec{V}$ is the key component of $\vec{V}$ in this context.

Shape calculus. Let $J(\Gamma)$ be a shape functional. The *shape derivative*, of the functional $J(\Gamma)$ at Γ, in the direction of the vector field $\vec{V}$ is defined as the limit

$$dJ(\Gamma;\vec{V}) = \lim_{t\to 0} \frac{1}{t}\big(J(\Gamma_t) - J(\Gamma)\big). \tag{2.5}$$

Let H be a Hilbert space of vector fields. The functional $J(\Gamma)$ is said to be shape differentiable at Γ in H if the shape derivative $dJ(\Gamma;\vec{V})$ exists for all $\vec{V} \in H$ and the mapping $\vec{V} \to dJ(\Gamma;\vec{V})$ is linear and continuous on H. We define $dJ(\Omega;\vec{V})$ similarly.

We define shape derivatives of functions as well. Given a function $\phi(\cdot,\Omega) : \Omega \to \mathbb{R}$ that itself depends on the geometric variable Ω, we can define its *material derivative* $\dot{\phi}(\Omega;\vec{V})$ at Ω in direction $\vec{V}$ as follows:

$$\dot{\phi}(\Omega;\vec{V}) = \lim_{t\to 0} \frac{1}{t}\big(\phi(x(t,\cdot),\Omega_t) - \phi(\cdot,\Omega_0)\big), \tag{2.6}$$

where the mapping $x(t,\cdot)$ is defined by (2.3). A similar definition holds for functions $\varphi(\cdot,\Gamma) : \Gamma \to \mathbb{R}$ that depend on boundaries Γ (see [22, Def. 2.71, Def. 2.85, Def. 2.88]). Then the *shape derivative* $\phi'(\Omega;\vec{V})$ of ϕ at Ω in direction $\vec{V}$ is defined to be

$$\phi'(\Omega;\vec{V}) = \dot{\phi}(\Omega;\vec{V}) - \nabla\phi \cdot \vec{V}. \tag{2.7}$$

For boundary functions $\varphi(\cdot,\Gamma)$, the shape derivative is defined to be

$$\varphi'(\Gamma;\vec{V}) = \dot{\varphi}(\Gamma;\vec{V}) - \nabla_\Gamma\varphi \cdot \vec{V}|_\Gamma. \tag{2.8}$$

For concreteness, let us give some examples of functions with dependence on geometric variables. These are $\phi = \phi(x,\Omega) = \phi(x,u_\Omega)$, where u_Ω is the solution of a PDE on Ω, and $\varphi = \varphi(x,\Gamma) = \varphi(x,\kappa)$, where κ is the curvature of Γ. Note that for functions $\phi = \phi(x)$, $\psi = \psi(x)$ that do not depend on the geometric variables Ω, Γ, the shape derivatives $\phi'(\Omega;\vec{V})$, $\psi'(\Gamma;\vec{V})$ are equal to zero.

Finally we define the second shape derivative $d^2J(\Gamma;\vec{V},\vec{W})$ of $J(\Gamma)$ at Γ with respect to vector fields $\vec{V},\vec{W}$ as

$$d^2J(\Gamma;\vec{V},\vec{W}) = d(dJ(\Gamma;\vec{V}))(\Gamma;\vec{W}). \tag{2.9}$$

This provides second order shape sensitivity information for shape functionals and it is useful in designing faster Newton-type optimization schemes. We refer to the books [9] and [22] for more information on tools of shape calculus and relevant results.

Shape derivative of length and PDE (1.3). If $J(\Gamma) = \int_\Gamma dS$, then

$$dJ(\Gamma;\vec{V}) = \int_\Gamma \kappa V dS \tag{2.10}$$

is its first shape derivative in the direction $\vec{V}$; recall that $V = \nu \cdot \vec{V}$. If $u_i = u_i(\Omega_i)$ satisfies (1.3), we can likewise compute their first shape derivative $u'_{i,V} = u'_i(\Omega_i;\vec{V})$ [9], [18], [22]:

$$\begin{cases} -\mu\Delta u'_{i,V} + u'_{i,V} = 0 \quad \text{in} \quad \Omega_i, \\ \frac{\partial u'_{i,V}}{\partial \nu_i} = \operatorname{div}_\Gamma(V\nabla_\Gamma u_i) + \frac{1}{\mu}(I - u_i)V \quad \text{on} \quad \partial\Omega_i. \end{cases} \tag{2.11}$$

Shape derivatives of the Mumford–Shah functional. Combining (2.10) and (2.11) it is possible to derive the following first derivative for (1.2)–(1.3):

$$dJ(\Gamma;\vec{V}) = \int_\Gamma \Big(\frac{1}{2}\,[\![|u-I|^2]\!] + \frac{\mu}{2}\,[\![|\nabla_\Gamma u|^2]\!] + \gamma\kappa\Big)V dS,$$

where $[\![f]\!] = f_1 - f_2$ stands for the jump of f across Γ. This was derived by Vese and Chan using a level set representation [24] and by Hintermüller and Ring [18] employing shape differential calculus. Computing the second shape derivative is more involved, and we refer to [18]:

$$\begin{aligned} d^2J(\Gamma;\vec{V},\vec{W}) = &\gamma \int_\Gamma \nabla_\Gamma V \cdot \nabla_\Gamma W dS + \int_\Gamma \beta V W dS \\ &+ \int_\Gamma \Big([\![(u-I)u'_W]\!] + \mu\, [\![\nabla u \cdot \nabla u'_W]\!] \Big) V dS, \end{aligned} \tag{2.12}$$

where $\beta = -\frac{\mu}{2}\, [\![|\nabla_\Gamma u|^2]\!] + \frac{1}{2}\left(\kappa\, [\![|u-I|^2]\!] + \partial_\nu\, [\![|u-I|^2]\!]\right)$ and $u'_{i,W} = u'_i(\Omega_i;\vec{W})$ satisfies (2.11).

Note that both first and second shape derivatives depend only on the normal components V, W of the velocity fields $\vec{V}$, $\vec{W}$. Therefore, from now on we prefer to work with the scalar velocity fields V, W and thereby write $dJ(\Gamma;V)$, $dJ^2(\Gamma;V,W)$.

3. Gradient descent flows.

Having just computed shape derivatives for (1.2)–(1.3), we can derive gradient descent velocities V as follows. First we introduce a Hilbert space $B(\Gamma)$, such as $L^2(\Gamma)$ or $H^1(\Gamma)$, and choose a continuous, coercive, and symmetric bilinear form $b(\cdot,\cdot)$ on $B(\Gamma)$. Next solve for V:

$$b(V,W) = -dJ(\Gamma;W) \quad \forall W \in B(\Gamma). \tag{3.1}$$

If $\mathcal{B}$ is the elliptic operator associated with b, namely, $\langle \mathcal{B}V, W\rangle = b(V,W)$, then (3.1) is equivalent to solving the following PDE on Γ:

$$\mathcal{B}V = -\gamma\kappa - f, \tag{3.2}$$

where $f = \frac{1}{2}\, [\![|u-I|^2]\!] + \frac{\mu}{2}\, [\![|\nabla_\Gamma u|^2]\!]$. If $\|\cdot\|_{B(\Gamma)}$ is the norm induced by $b(\cdot,\cdot)$, then it turns out that the velocity V computed this way decreases the energy

$$dJ(\Gamma;V) = -b(V,V) = -\|V\|^2_{B(\Gamma)} \leqslant 0. \tag{3.3}$$

An obvious choice for $b(\cdot,\cdot)$ is the $L^2(\Gamma)$ scalar product. This gives

$$\langle V, W\rangle = \int_\Gamma V W dS = -\int_\Gamma (\gamma\kappa + f) W dS \quad \forall W \in B(\Gamma),$$

which is the L^2 gradient flow of [23], [24]. Alternatively, we could choose $b(\cdot,\cdot)$ to coincide with a weighted $H^1(\Gamma)$ scalar product and thereby deduce

$$\langle V, W\rangle_{H^1(\Gamma)} = \langle \alpha\nabla_\Gamma V, \nabla_\Gamma W\rangle + \langle \beta V, W\rangle = -\int_\Gamma (\gamma\kappa + f)\, W dS \quad \forall W \in B(\Gamma),$$

where $\alpha = \alpha(x,\Gamma)$, $\beta = \beta(x,\Gamma)$ are some positive functions. The strong form of this equation is the PDE on Γ

$$-\mathrm{div}_\Gamma(\alpha\nabla_\Gamma V) + \beta V = -\gamma\kappa - f, \tag{3.4}$$

which enforces additional smoothness on velocity V. This could be used to relax the regularity requirement on the curve for velocity computation. While curvature κ needs to be L^2 for the L^2 flow, (3.4) allows us to compute velocity with less regular κ. Among the many possibilities to specify α and β, we are interested in one specific choice: the one coming from the second shape derivative (2.12). Observe that (2.12) has a structure similar to a weighted H^1 scalar product but it may not be positive definite in general. However, we can make use of a modified version upon choosing

$$\alpha = \gamma, \quad \beta = \frac{1}{2}\Big(\kappa \left[\!\left[|u-I|^2 \right]\!\right] + \partial_\nu \left[\!\left[|u-I|^2 \right]\!\right] - \mu \left[\!\left[|\nabla u|^2 \right]\!\right] \Big)_+ \tag{3.5}$$

with $(\cdot)_+ := \max(\cdot, \epsilon)$ for a given $\epsilon > 0$ and omitting the last term in (2.12) as it is known to be negative definite (see [18]). The resulting velocity V corresponds to an inexact Newton descent direction, an idea first proposed by Hintermüller and Ring [18]. We will use this bilinear form in our experiments to obtain faster convergence.

Given a curve Γ, (3.2) by itself is not enough. We need a way to compute the curvature κ and eventually the vector velocity $\vec{V}$ to move the curve. For this, we recall the differential geometry relation $\vec{\kappa} = -\Delta_\Gamma \vec{X}$, first proposed by Dziuk for computation (see [8]). This relation is just an equivalent formula for the usual definition $\vec{\kappa} = \vec{r}_{ss}(s)$ of the curvature vector given an arc-length periodic parametrization of the curve $\vec{r} : \mathbb{R} \to \mathbb{R}^2$. Instead of the parametrized representation, we work with the function $\vec{X} : \mathbb{R}^2 \to \mathbb{R}^2$ on the plane, which, when restricted to Γ, yields the (x, y) coordinates of the point on the curve. Just as we differentiate parametrized $\Gamma(s)$ twice to obtain $\vec{\kappa}$ (in effect computing tangential derivatives), we apply the operator $\Delta_\Gamma = \mathrm{div}_\Gamma \nabla_\Gamma$ to each component of $\vec{X}$ to compute $\vec{\kappa}$. We use the minus sign to be consistent with the convention that a circle with outward unit normal $\vec{\nu}$ has positive scalar curvature κ. Proceeding as in [3], [11], we introduce the following system of nonlinear PDEs on Γ:

$$\vec{\kappa} = -\Delta_\Gamma \vec{X}, \quad \kappa = \vec{\kappa} \cdot \vec{\nu}, \quad \mathcal{B}V = -\gamma\kappa - f, \quad \vec{V} = V\vec{\nu}. \tag{3.6}$$

4. Discrete gradient flows.

In this section we describe the discretization of (3.6), which replaces $\Gamma(t)$ by a sequence of polygonal curves $\{\Gamma_n\}_{n\geqslant 0}$ such that $J(\Gamma_{n+1}) \leqslant J(\Gamma_n)$. We first introduce the time discretization that linearizes (3.6). We then discuss the space discretization based on the FEM with linear polynomials. We also explain how to solve the resulting linear systems.

4.1. Time discretization.

To discretize the gradient flow (3.1) in time, we choose a time step $\tau > 0$ and update the current position vector $\vec{X}_n$ of Γ_n by

$$\vec{X}_{n+1} = \vec{X}_n + \tau \vec{V}_{n+1} \tag{4.1}$$

to determine Γ_{n+1}. Although we refer to τ as a time step and frequently call the process curve evolution, it is in fact an instance of a shape optimization problem and τ is a step-size in the descent direction $\vec{V}_{n+1}$. The first step is the computation of the descent direction $\vec{V}_{n+1}$. To this end, we propose the minimization problem

$$\vec{V}_{n+1} = \nu_{n+1} V_{n+1} := \mathrm{argmin}_{\vec{V}} \Big(J(\Gamma_n + \tau\vec{V}) + \frac{1}{2\tau} b(\tau\vec{V}, \tau\vec{V}) \Big). \tag{4.2}$$

The second term can also be viewed as regularization for the velocity. The following is the optimality condition for (4.2), which mimics (3.1):

$$b(V_{n+1}, W) = -dJ(\Gamma_{n+1}; W) \qquad \forall W \in B(\Gamma_n). \tag{4.3}$$

This equation reveals the highly nonlinear nature of this implicit discretization. In fact, we need to determine V_{n+1} upon solving a PDE on Γ_{n+1} which is unknown.

To circumvent this difficulty, we propose below either *explicit* or *semi-implicit* time-stepping schemes. In both cases, we compute $u_{i,n}$ from (1.3) in $\Omega_{i,n}$ and next V_{n+1} on the current polygonal Γ_n. This splitting leads to linear PDEs. A key issue is whether there are time-step restrictions due to stability. We show that the only restriction is due to handling geometry properly, but not to stability.

The explicit scheme. In view of (3.6) and (4.3), a simple approach to find $\vec{V}_{n+1}$ is the explicit scheme: find $u_{i,n}$ from the PDE (1.3) in domains $\Omega_{i,n}$ and next

$$\vec{\kappa}_n = -\Delta_{\Gamma_n}\vec{X}_n, \tag{4.4}$$

$$\kappa_n = \vec{\kappa}_n \cdot \vec{\nu}_n, \tag{4.5}$$

$$\mathcal{B}_n V_{n+1} = -\gamma\kappa_n - f_n, \tag{4.6}$$

$$\vec{V}_{n+1} = V_{n+1}\vec{\nu}_n. \tag{4.7}$$

While this is a relatively efficient way to compute $\vec{V}_{n+1}$ and thus is preferred in practice, it might be prone to instabilities. This is because the velocity equation (4.6) includes the curvature term, or *geometric diffusion* term.

Stability of the explicit scheme. This time discretization is definitely unstable for the L^2 gradient flow, but it is stable for the weighted $H^1(\Gamma)$ scalar product if $\alpha > 0$. To gain some insight, we take $\alpha, \beta > 0$ constant and consider the curve Γ_n to be a graph of a function U_n with a small variation. In this case, the new curve Γ_{n+1} can be described by a function U_{n+1} satisfying the following approximation of (4.6):

$$-\alpha\Delta\delta U_{n+1} + \beta\delta U_{n+1} = \gamma\Delta U_n - f_n \tag{4.8}$$

with periodic boundary conditions and $\delta U_{n+1} := \frac{1}{\tau_n}(U_{n+1} - U_n)$ being the velocity V_{n+1}. This is thus a time discretization of $-\alpha\Delta U_t + \beta U_t - \gamma\Delta U = -f$. We can now multiply (4.8) by $-\Delta U_{n+1}$, integrate by parts, and add over n to infer that

$$\begin{aligned}\alpha\|\Delta U_{N+1}\|^2 + \beta\|\nabla U_{N+1}\|^2 \le{}& \alpha\|\Delta U_0\|^2 + \beta\|\nabla U_0\|^2 \\ &+ 2\sum_{n=0}^{N}\tau_n\Big(\gamma\|\Delta U_{n+1}\|\|\Delta U_n\| + \|f_n\|\|\Delta U_{n+1}\|\Big).\end{aligned}$$

Young's inequality, together with the mild restriction $\tau_n/\tau_{n-1} \le \Lambda$ (see section 5.1), implies that

$$\begin{aligned}\alpha\|\Delta U_{N+1}\|^2 + \beta\|\nabla U_{N+1}\|^2 \le{}& (\alpha + \tau_0)\|\Delta U_0\|^2 + \beta\|\nabla U_0\|^2 + \sum_{n=0}^{N}\tau_n\|f_n\|^2 \\ &+ (1 + \gamma + \gamma\Lambda)\sum_{n=0}^{N}\tau_n\|\Delta U_{n+1}\|^2.\end{aligned}$$

Applying Gronwall's inequality we conclude that

$$\begin{aligned}&\alpha\|\Delta U_{N+1}\|^2 + \beta\|\nabla U_{N+1}\|^2 \\ &\quad\le \left((\alpha + \tau_0)\|\Delta U_0\|^2 + \beta\|\nabla U_0\|^2 + \sum_{n=0}^{N}\tau_n\|f_n\|^2\right)\exp\left((1 + \gamma + \gamma\Lambda)\sum_{n=0}^{N}\tau_n\right),\end{aligned}$$

and the discretization is thus stable irrespective of the choice of the time steps τ_n. This is in sharp contrast with the case $\alpha = 0$ (the L^2 flow), since explicit time stepping of the heat equation is unstable without space discretization. This argument provides insight into the stability of the weighted H^1 inner product and rules out the use of the L^2 scalar product in conjunction with (4.4)–(4.7).

The semi-implicit scheme. We now describe a time discretization that can be used for both the L^2 and H^1 gradient flows. The essence of the scheme is to compute curvature and velocity implicitly in (3.6) which, upon imposing (4.1), translates into the linear system of PDEs: find $(\vec{\kappa}_{n+1}, \kappa_{n+1}, V_{n+1}, \vec{V}_{n+1})$ such that

$$\vec{\kappa}_{n+1} + \tau_n \Delta_{\Gamma_n} \vec{V}_{n+1} = -\Delta_{\Gamma_n} \vec{X}_n, \tag{4.9}$$

$$\kappa_{n+1} - \vec{\kappa}_{n+1} \cdot \vec{\nu}_n = 0, \tag{4.10}$$

$$\mathcal{B}_n V_{n+1} + \gamma \kappa_{n+1} = -f_n, \tag{4.11}$$

$$\vec{V}_{n+1} - V_{n+1} \vec{\nu}_n = 0, \tag{4.12}$$

and f_n is computed from $u_{i,n}$ on $\Omega_{i,n}$. This semi-implicit scheme was employed in [2], [3], [11] to solve other problems, some in shape optimization.

Stability of the semi-implicit scheme. We prove unconditional stability for the L^2 flow. For this, we take $\alpha = 0, \beta = 1$ and proceed as in [2], [3], [12]. We start by multiplying (4.11) with κ_{n+1} and integrating to obtain

$$\langle V_{n+1}, \kappa_{n+1} \rangle + \gamma \|\kappa_{n+1}\|^2 = -\langle f_n, \kappa_{n+1} \rangle.$$

Multiplying (4.12) with $\vec{\kappa}_{n+1}$ and (4.10) with V_{n+1}, we easily arrive at

$$\langle \vec{V}_{n+1}, \vec{\kappa}_{n+1} \rangle = \langle V_{n+1}, \vec{\kappa}_{n+1} \cdot \vec{\nu} \rangle = \langle \kappa_{n+1}, V_{n+1} \rangle,$$

whence

$$\langle \vec{V}_{n+1}, \vec{\kappa}_{n+1} \rangle + \gamma \|\kappa_{n+1}\|^2 = -\langle f_n, \kappa_{n+1} \rangle. \tag{4.13}$$

On the other hand, multiplying (4.9) with $\tau_n \vec{V}_{n+1}$ and observing that $\tau_n \vec{V}_{n+1} = \vec{X}_{n+1} - \vec{X}_n$ yields

$$\tau_n \langle \vec{V}_{n+1}, \vec{\kappa}_{n+1} \rangle - \langle \nabla_\Gamma \vec{X}_{n+1}, \nabla_\Gamma (\vec{X}_{n+1} - \vec{X}_n) \rangle = 0. \tag{4.14}$$

Multiplying (4.13) by τ_n and substituting into (4.14) we infer that

$$\langle \nabla_\Gamma \vec{X}_{n+1}, \nabla_\Gamma (\vec{X}_{n+1} - \vec{X}_n) \rangle + \gamma \tau_n \langle \kappa_{n+1}, \kappa_{n+1} \rangle = -\tau_n \langle f_n, \kappa_{n+1} \rangle.$$

Applying the inequality $ab \leqslant \epsilon a^2 + \frac{1}{4\epsilon} b^2$ with $\epsilon = \frac{\gamma}{2}, \quad a = \kappa_{n+1}, \quad b = -f_n$,

$$\langle \nabla_\Gamma \vec{X}_{n+1}, \nabla_\Gamma (\vec{X}_{n+1} - \vec{X}_n) \rangle + \tau_n \gamma \|\kappa_{n+1}\|^2 \leqslant \tau_n \frac{\gamma}{2} \|\kappa_{n+1}\|^2 + \frac{\tau_n}{2\gamma} \|f_n\|^2,$$

whence, using the inequality $|\Gamma_{n+1}| - |\Gamma_n| \leqslant \langle \nabla_\Gamma \vec{X}_{n+1}, \nabla_\Gamma (\vec{X}_{n+1} - \vec{X}_n) \rangle$ [2],

$$|\Gamma_{n+1}| - |\Gamma_n| + \tau_n \frac{\gamma}{2} \|\kappa_{n+1}\|^2 \leqslant \frac{\tau_n}{2\gamma} \|f_n\|^2.$$

Summing up over n, from 0 to $m - 1$, yields the stability estimate

$$|\Gamma_m| + \frac{\gamma}{2} \sum_{n=0}^{m-1} \tau_n \|\kappa_{n+1}\|^2_{L^2(\Gamma_n)} \le |\Gamma_0| + \frac{1}{2\gamma} \sum_{n=0}^{m-1} \tau_n \|f_n\|^2_{L^2(\Gamma_n)}.$$

4.2. Finite element discretization. We assume that iteration n is fixed in this discussion and avoid writing it for convenience. The discretization of (1.3) is fairly standard. Let $\mathcal{T}^{\Omega_i} = \mathcal{T}_n^{\Omega_i}$ be a shape-regular but possibly graded mesh of triangular finite elements over the domain Ω_i and let $\{\psi_j\}_{j=1}^I$ be the set of canonical basis functions of the finite element space $\mathcal{V}(\Omega_i)$ of continuous polynomials over $\mathcal{T}^{\Omega_i}$. We thus have a conforming approximation $\mathcal{V}(\Omega_i)$ of $H^1(\Omega_i)$. If $\langle f, g\rangle_{\Omega_i} = \int_{\Omega_i} fg\,dx$, the discrete version of (1.3) reads as follows: find $u_i \in \mathcal{V}(\Omega_i)$ such that

$$\mu\langle\nabla u_i, \nabla\psi\rangle_{\Omega_i} + \langle u_i, \psi\rangle_{\Omega_i} = \langle I, \psi\rangle_{\Omega_i} \quad \forall\psi \in \mathcal{V}(\Omega_i). \tag{4.15}$$

We now focus on the finite element discretization of (4.4)–(4.7) and (4.9)–(4.12). Let $\Gamma := \Gamma_n$ be a polygonal curve, and let $\mathcal{T} = \mathcal{T}_n$ be a graded partition of Γ into line segments. Let $T \in \mathcal{T}$ be a generic finite element and let $\vec{\nu}_T = (\nu_T^j)_{j=1,2}$ be the unit normal to T pointing outward Ω_1. We denote by $\vec{\nu}$ the unit normal to Γ, defined locally by $\vec{\nu}|_T = \vec{\nu}_T$ for all $T \in \mathcal{T}$. Let $\{\phi_j\}_{j=1}^I$ be the canonical basis functions of the finite element space $\mathcal{V}(\Gamma)$ of continuous piecewise linear functions over $\mathcal{T}$, and set $\vec{\mathcal{V}}(\Gamma) := \mathcal{V}(\Gamma)^2$. We thus have a conforming approximation $\mathcal{V}(\Gamma)$ of $H^1(\Gamma)$.

The explicit scheme. We multiply (4.4)–(4.7) by test functions $\phi \in \mathcal{V}(\Gamma)$ and $\vec{\phi} \in \vec{\mathcal{V}}(\Gamma)$ and integrate by parts those terms involving Δ_Γ. We thus arrive at the fully discrete problem: seek $\vec{V}, \vec{\kappa} \in \vec{\mathcal{V}}(\Gamma)$, $V, \kappa \in \mathcal{V}(\Gamma)$, such that

$$\langle\vec{\kappa}, \vec{\phi}\rangle = \langle\nabla_\Gamma\vec{X}, \nabla_\Gamma\vec{\phi}\rangle \qquad \forall\vec{\phi} \in \vec{\mathcal{V}}(\Gamma), \tag{4.16}$$

$$\langle\kappa, \phi\rangle = \langle\vec{\kappa}\cdot\vec{\nu}, \phi\rangle \qquad \forall\phi \in \mathcal{V}(\Gamma), \tag{4.17}$$

$$\langle\alpha\nabla_\Gamma V, \nabla_\Gamma\phi\rangle + \langle\beta V, \phi\rangle = -\gamma\langle\kappa, \phi\rangle - \langle f, \phi\rangle \quad \forall\phi \in \mathcal{V}(\Gamma), \tag{4.18}$$

$$\langle\vec{V}, \vec{\phi}\rangle = \langle V, \vec{\phi}\cdot\vec{\nu}\rangle \qquad \forall\vec{\phi} \in \vec{\mathcal{V}}(\Gamma). \tag{4.19}$$

The semi-implicit scheme. We multiply (4.9)–(4.12) by test functions $\phi \in \mathcal{V}(\Gamma)$ and $\vec{\phi} \in \vec{\mathcal{V}}(\Gamma)$ and again integrate by parts. We obtain the fully discrete problem: seek $\vec{V}, \vec{\kappa} \in \vec{\mathcal{V}}(\Gamma)$, $V, \kappa \in \mathcal{V}(\Gamma)$, such that

$$\langle\vec{\kappa}, \vec{\phi}\rangle - \tau\langle\nabla_\Gamma\vec{V}, \nabla_\Gamma\vec{\phi}\rangle = \langle\nabla_\Gamma\vec{X}, \nabla_\Gamma\vec{\phi}\rangle \quad \forall\vec{\phi} \in \vec{\mathcal{V}}(\Gamma), \tag{4.20}$$

$$\langle\kappa, \phi\rangle - \langle\vec{\kappa}\cdot\vec{\nu}, \phi\rangle = 0 \qquad \forall\phi \in \mathcal{V}(\Gamma), \tag{4.21}$$

$$\langle\alpha\nabla_\Gamma V, \nabla_\Gamma\phi\rangle + \langle\beta V, \phi\rangle + \gamma\langle\kappa, \phi\rangle = -\langle f, \phi\rangle \qquad \forall\phi \in \mathcal{V}(\Gamma), \tag{4.22}$$

$$\langle\vec{V}, \vec{\phi}\rangle - \langle V, \vec{\phi}\cdot\vec{\nu}\rangle = 0 \qquad \forall\vec{\phi} \in \vec{\mathcal{V}}(\Gamma). \tag{4.23}$$

4.3. Matrix formulation. The matrix formulation for (4.15) is well known, so we skip it here. We turn our attention to the matrix formulation of the fully discrete problems (4.16)–(4.19) and (4.20)–(4.23). Given the matrix entries

$$M_{\beta_{i,j}} := \langle\beta\phi_i, \phi_j\rangle, \quad M_{i,j} := \langle\phi_i, \phi_j\rangle, \quad \vec{M}_{i,j} := M_{i,j}\vec{\mathrm{Id}},$$
$$\vec{N}_{i,j} := (N_{i,j}^k)_{k=1}^2 := \langle\phi_i, \phi_j\nu^k\rangle_{k=1}^2,$$
$$A_{i,j} := \langle\nabla_\Gamma\phi_i, \nabla_\Gamma\phi_j\rangle, \quad A_{\alpha i,j} := \langle\alpha\nabla_\Gamma\phi_i, \nabla_\Gamma\phi_j\rangle, \quad \vec{A}_{i,j} := A_{i,j}\vec{\mathrm{Id}},$$

with $\vec{\mathrm{Id}} \in \mathbb{R}^{2\times2}$ being the identity matrix, $(\vec{e}_k)_{k=1}^2$ the canonical basis of $\mathbb{R}^2$, and $\nu^k = \vec{\nu}\cdot\vec{e}_k$, the corresponding matrices are

$$M_\beta := (M_{\beta_{i,j}})_{i,j=1}^I, \quad M := (M_{i,j})_{i,j=1}^I, \quad \vec{M} := (\vec{M}_{i,j})_{i,j=1}^I, \quad \vec{N} := (\vec{N}_{i,j})_{i,j=1}^I,$$
$$A_\alpha := (A_{\alpha i,j})_{i,j=1}^I, \quad A := (A_{i,j})_{i,j=1}^I, \quad \vec{A} := (\vec{A}_{i,j})_{i,j=1}^I.$$

We use the convention that a vector of nodal values of a finite element function is written in boldface: $\mathbf{V} = (V_i)_{i=1}^{I} \in \mathbb{R}^I$ is equivalent to $V = \sum_{i=1}^{I} V_i\phi_i \in \mathcal{V}(\Gamma)$. We point out that $\vec{M}, \vec{A}$, and $\vec{N}$ possess matrix-valued entries and therefore the matrix-vector product is understood in the sense $\vec{M}\vec{\mathbf{V}} = \big(\sum_{j=1}^{I} \vec{M}_{i,j}\vec{V}_j\big)_{i=1}^{I}$, each component $\vec{V}_i$ of $\vec{\mathbf{V}}$, as well as each of $\vec{M}\vec{\mathbf{V}}$, is itself a vector in $\mathbb{R}^2$.

We are now in a position to write the desired matrix formulations. Upon expanding the unknown scalar functions $V, K \in \mathcal{V}(\Gamma)$ and vector functions $\vec{V}, \vec{K} \in \vec{\mathcal{V}}$ in terms of the basis functions and setting $\phi = \phi_i$ and $\vec{\phi}^k = \phi\vec{e}_k$, we easily arrive at the linear system of equations:

$$\begin{array}{cc}
\textit{explicit scheme} & \textit{semi-implicit scheme} \\
\vec{M}\vec{\mathbf{K}} = \vec{A}\vec{\mathbf{X}}, & -\tau\vec{A}\vec{\mathbf{V}} + \vec{M}\vec{\mathbf{K}} = \vec{A}\vec{\mathbf{X}}, \\
M\mathbf{K} = \vec{N}^T\vec{\mathbf{K}}, & M\mathbf{K} - \vec{N}^T\vec{\mathbf{K}} = \mathbf{0}, \\
(A_\alpha + M_\beta)\mathbf{V} = -\gamma M\mathbf{K} - \mathbf{f}, & (A_\alpha + M_\beta)\mathbf{V} + \gamma M\mathbf{K} = -\mathbf{f}, \\
\vec{M}\vec{\mathbf{V}} = \vec{N}\mathbf{V}, & \vec{M}\vec{\mathbf{V}} - \vec{N}\mathbf{V} = \vec{\mathbf{0}}.
\end{array}$$

4.4. Solving the linear system. The linear system corresponding to the finite element discretization of (4.15), the domain PDE, has been studied extensively in literature, and effective strategies exist. In our experiments, the size of the linear system was at most several thousands. So a direct linear solver was appropriate for our problem. We used UMFPACK [6] for this purpose.

We now describe how to solve the linear systems for the velocity equations. But first we make an important observation. If the order of the nodal values in the vectors and matrices is arranged following the connectivity pattern of the nodes, then the matrices have a particular form with circulant tridiagonal blocks on their diagonals (except for $\vec{N}$). Each block corresponds to a single closed curve and looks like

$$\begin{pmatrix}
a_1 & c_1 & & & b_1 \\
b_2 & a_2 & c_2 & & \\
 & b_3 & a_3 & \ddots & \\
 & & \ddots & \ddots & c_{m-1} \\
c_m & & & b_m & a_m
\end{pmatrix}.$$

If the block is not singular, we can invert it with $O(m)$ operations using a variant of the Gaussian elimination algorithm. Then the block diagonal matrix can also be inverted with linear time complexity. This property was also noted in [17] for a related image segmentation problem.

The explicit scheme. Now given the linear system for the explicit system (4.16)–(4.19), we can solve for the velocity $\vec{V}$ as follows:

$$\begin{aligned}
&\vec{\mathbf{K}} = \vec{M}^{-1}\vec{A}\vec{\mathbf{X}}, \\
&\mathbf{K} = M^{-1}\vec{N}^T\vec{\mathbf{K}}, \\
&\mathbf{V} = (A_\alpha + M_\beta)^{-1}(-\gamma M\mathbf{K} - \mathbf{f}), \\
&\vec{\mathbf{V}} = \vec{M}^{-1}\vec{N}\mathbf{V}.
\end{aligned} \tag{4.24}$$

Because of the special structure of the coefficient matrices $M, \vec{M}, A_\alpha + M_\beta$ as described above, we can invert these easily in linear time.

The semi-implicit scheme. The solution of the semi-implicit system (4.20)–(4.23) is a bit more involved compared to the explicit system. We first rewrite the system as

$$\begin{pmatrix} \vec{M} & 0 & 0 & -\vec{N} \\ 0 & M & -\vec{N}^T & 0 \\ -\tau\vec{A} & 0 & \vec{M} & 0 \\ 0 & \gamma M & 0 & A_\alpha + M_\beta \end{pmatrix} \begin{pmatrix} \vec{\mathbf{V}} \\ \mathbf{K} \\ \vec{\mathbf{K}} \\ \mathbf{V} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vec{A}\vec{\mathbf{X}} \\ -\mathbf{f} \end{pmatrix} \tag{4.25}$$

and next eliminate the variables to arrive at the Schur complement system

$$(\gamma\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + A_\alpha + M_\beta)\mathbf{V} = -\mathbf{f} - \gamma\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\mathbf{X}}. \tag{4.26}$$

This system is to be solved at each time step. Since (4.26) is symmetric and positive definite, we can solve it efficiently using the conjugate gradient (CG) method. A couple of improvements can accelerate the convergence of CG. The first is to use a good initial guess. Since the value of γ for our test problems is in general small, at most 10^{-2}, we can expect the following system to be close to (4.26):

$$(A_\alpha + M_\beta)\mathbf{V} = -\mathbf{f} - \gamma\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\mathbf{X}}.$$

The second improvement is to use a good preconditioner for CG. Since we need to approximate the inverse of $(\tau\gamma\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + A_\alpha + M_\beta)$, again $(A_\alpha + M_\beta)^{-1}$ is a natural candidate when γ is small. These two enhancements accelerate CG significantly because the action of $(A_\alpha + M_\beta)^{-1}$ can be computed with linear complexity.

5. The main algorithm.

The FEM of section 4 requires several computational enhancements to be effective on practical examples. We describe them in this section.

We stress that our method is driven by curve evolution. This is consistent with the fact that our primary goal is segmentation, namely, detection of the region boundaries. So we start each iteration with the curve mesh and generate the domain meshes inside and outside the curve, using TRIANGLE (see Shewchuk [21]). We first solve the domain PDE (1.3) for u_i, next solve (4.24)(c) or (4.26) for scalar velocity $\mathbf{V}$, and compute vector velocity $\vec{\mathbf{V}} = M^{-1}\vec{N}\mathbf{V}$. With $\vec{\mathbf{V}}$ at hand, we move the curve Γ according to $\vec{\mathbf{X}} + \tau\vec{\mathbf{V}}$. There are yet a number of issues that we need to address:

- choosing a suitable time step (or step-size) for each iteration of the optimization;
- handling curve and domain meshes efficiently and accurately;
- implementing topological changes for curves, such as merging and splitting.

The last item is crucial as we would like the capability to detect multiple objects in given images without a priori knowledge of their number or topology. We describe how to handle these three issues below. The resulting scheme is given in Algorithm 1.

5.1. Time adaptivity.

We consider two aspects of the choice of time step: first, its role in the optimization process, and second, its impact on the quality of the curve mesh. An effective time step should reduce the energy with the given velocity. To ensure this, we employ a backtracking strategy; we move the curve with the given velocity and current time step and compute the energy. If the energy is not reduced, we halve the time step and check again. We repeat this until we obtain a time step that gives energy decrease. Here a line search approach is also possible, for example, based on the Armijo conditions, as proposed in [17], [18].

Algorithm 1. The main algorithm.

```
choose an initial curve $\Gamma_0$
compute $dJ(\Gamma_0;\cdot)$
**repeat**
    adapt curve $\Gamma$ geometrically
    generate and adapt domain meshes $\Omega_i$
    solve domain PDE (1.3) in $\Omega_i$
    compute velocity $\vec{V}$ on $\Gamma$ with the explicit or the semi-implicit scheme
    use backtracking to choose time step $\tau$
    move the curve $\Gamma$ with $\vec{V}$ and $\tau$
    detect intersections, perform topological changes if necessary
    compute $dJ(\Gamma;\cdot)$
**until** $|dJ(\Gamma;\cdot)| \leqslant \delta_1 |dJ(\Gamma_0;\cdot)| + \delta_0$
```

On the other hand, we also need to be vigilant for the impact of the time step on the quality of the curve mesh. Large time steps may cause mesh distortion and mesh entanglement by neighboring nodes crossing each other. To avoid these, we utilize the time-step control scheme proposed by Bänsch, Morin, and Nochetto in [3] and also used in [11]. Its main idea is to prevent nodes of the same element a relative tangential displacement greater than the size of the element. If $\vec{V}_1, \vec{V}_2$ are the velocities at the two nodes of a line element, then the time step τ should satisfy $\tau|(\vec{V}_1 - \vec{V}_2)\cdot\vec{t}| \approx \tau h|\nabla_\Gamma \vec{V}| \leqslant \epsilon_\tau h$, where $\vec{t}$ is the unit tangent vector, h is the size of the element, and ϵ_τ is a specified tolerance.

5.2. Space adaptivity. The procedures described below help tune mesh resolution with respect to geometry and data. The goal is to balance accuracy and computational cost by mesh refinement where resolution is needed and mesh coarsening where resolution is unnecessary. We realize this as follows:

- geometry adaptivity for curves to represent shapes well;
- data-driven adaptivity for domains to obtain accurate smooth approximations u_i.

Geometry adaptivity for curves. This idea was introduced by Bänsch, Morin, and Nochetto in [3], was further used in [11], and consists of approximating the second fundamental form. This reduces to curvature κ for planar curves and suggests concentrating nodes where Γ varies most, and so κ is largest. Since the pointwise accuracy in representing a curve by a polygonal is proportional to $h_T^2|\kappa|$, and κ is accessible through our scheme, we impose $h_T^2|\kappa| \le \epsilon_{geom}$ for each element T on Γ. We then refine if $h_T^2|\kappa| > \epsilon_{geom}$ or coarsen if $h_T^2|\kappa| \ll \epsilon_{geom}$.

Data-driven adaptivity for domains. The accuracy of solution u_i of (1.3) in the domain Ω_i depends on sufficient resolution of the image I. The meshes generated with TRIANGLE can be properly graded to resolve the image well. For this purpose, we introduce some error indicators that help us evaluate the local mesh resolution. Since a posteriori error estimation is well developed for elliptic PDEs, one might be inclined to resort to a posteriori error estimators to guide adaptivity. However, it is questionable whether such an off-the-shelf approach would be adequate for shape optimization for the following reasons:

- It is possible to go through very *bad* domains in the intermediate stages of the optimization, which are unrelated to the final shape. One may thus waste unnecessary computational effort by concentrating on accurate recovery over such domains.

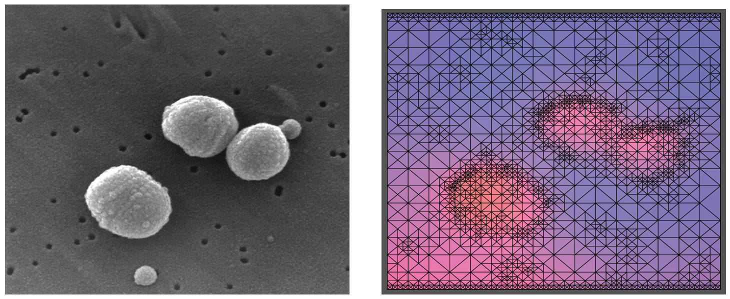


FIG. 5.1. *Input image (left) and computational mesh (right) created using the data-driven adaptivity algorithm of section* 5.2. *The mesh is finer where the image varies more and coarser in smooth regions, thereby allowing for an accurate representation of the image with a low computational cost.*

- Through the shape optimization process, we always have an approximation of the domain and no direct clue of the exact domain. Therefore, pretending to have the exact domain may be counter productive. Mumford and Shah show in [19] that the only singularities that the set K can contain are triple points and cracks. Since we restrict ourselves to the case where K is a union of closed curves in (1.2), these are guaranteed to be smooth by this result. So refining to capture possible reentrant corners in K is unjustified (in fact wasteful from the viewpoint of computation).

An alternative is to have adaptivity driven by data resolution. If $\Pi_h I$ denotes the piecewise polynomial interpolant of the given image I in the current finite element space, then we check the L^2 error between $\Pi_h I$ and I:

$$\|I - \Pi_h I\|_{L^2(\Omega_i)} \leqslant \epsilon_{data} \frac{|\Omega_i|}{|D|}.$$

If this is violated, we refine the elements with largest error. This is illustrated in Figure 5.1. Let us note that this will give finer triangulations than necessary since the solutions u_i in Ω_i are smooth. On the other hand, we reduce computational effort by not re-solving the PDE to calculate the error estimators for several iterations of an adaptivity loop. A similar idea has been used by Fried to obtain an initial finite element mesh in [14].

5.3. Topological changes. The level set formulation of Mumford–Shah has one important advantage [18], [23], [24]: the ability to carry out topological changes such as merging and splitting of curves automatically. This is not the case for a Lagrangian approach, such as ours, and so additional work is necessary. Nevertheless, we have also implemented this capability. The topology procedure consists of four steps executed at the end of each iteration of the evolution:

- *Detect.* The line sweeping algorithm is used to detect whether curves intersect. If there are intersections, then topological surgery needs to be carried out.
- *Adjust.* The local resolution of the curve is adjusted at intersection locations to account for degenerate cases, such as multiple elements intersecting the same element.
- *Reconnect.* The intersecting elements are reconnected to obtain the correct resulting curves.
- *Clean-up.* Possible superfluous curves created by the previous step are deleted.

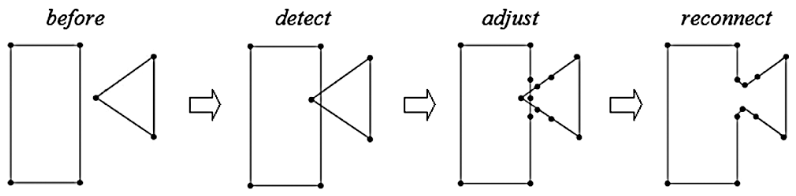


FIG. 5.2. *A simple example of a topological change: intersection is detected, local resolution is adjusted, intersecting elements are reconnected, and the new curve is obtained.*

The first step of the procedure has $O((n + I) \log n)$ time complexity where n is the number of elements and I is the number of intersections. (See [7] for more information on the line-sweeping procedure.) The remaining steps have linear time complexity. We illustrate the topology procedure with a simple example in Figure 5.2. We refer to [10] for more details on the implementation of the topology surgery procedure.

5.4. A multilevel algorithm. In image processing, a commonly used heuristic is to compute the solution on a coarsely sampled image, for example, one obtained by an image pyramid, and to use this as an initial guess for the computations on the higher-resolution image. We also found a multilevel approach to be very beneficial for this optimization problem. The main idea for this is to keep the computations as coarse as possible when we are far away from the optimum and to refine the computations as we get closer. However, in contrast to the practice in image processing, the basis of our approach is the discretization of the PDE, not the resolution of the image. The motivation of our multilevel algorithm is twofold. First from the time-step side: very careful time steps at the beginning of the optimization are not worth the effort. This is true in particular with backtracking as we may need to solve the domain PDE several times to compute the energy as part of backtracking. Thus we prefer to use N_{coarse} number of fixed time steps at the beginning. This hopefully brings us closer to the optimum. Also in this stage, space adaptivity in domain is switched off and space adaptivity on the curve is executed with $\epsilon_{coarse} > \epsilon_{geom}$. This results in coarser curves, which also implies coarser triangulations, so these initial iterations are cheap. Another very important benefit of this coarse stage is that it helps avoid local minima.

After the initial coarse stage, we take a continuation approach to tighten the tolerance parameters of the space adaptivity procedures. We enforce better resolution as we approach the minimum. Since the shape gradient becomes small close to the minimum, we can use this as a way to determine when we should enforce higher resolution in the following manner. We choose a sequence of gradient thresholds $\infty = d_n > d_{n-1} > \cdots > d_1 > d_0 = 0$ and corresponding geometric adaptivity tolerance factors $\epsilon_n \geqslant \epsilon_{n-1} \geqslant \cdots \geqslant \epsilon_1 \geqslant \epsilon_{geom}$ and corresponding time steps $\tau_n \geqslant \tau_{n-1} \geqslant \cdots \geqslant \tau_1$. Then at a given iteration k, if the magnitude of the shape gradient $|dJ(\Gamma_k; \cdot)|$ drops to the interval $[d_i, d_{i+1}]$, $i > 0$, the adaptivity tolerance is set to be ϵ_i and the time step is set to be τ_i. If $|dJ(\Gamma_k; \cdot)|$ drops to $[d_1, d_0]$, we use backtracking for time-step selection. In our experiments we found that the use of two levels (i.e., $n = 3$) gives satisfactory results.

6. Numerical experiments. In this section we present a number of simulations, which examine various properties of the model and the numerical scheme. We implemented the discrete gradient flows of section 4 and the computational procedures described in section 5 within the finite element toolbox ALBERTA [20]. The space adaptivity parameters were set to $\epsilon_{geom} = 0.03$, $\epsilon_{data} = 0.05$. For the multilevel strategy, we used two levels and an initial coarse stage. The coarse stage consisted of $N_{coarse} = 50$ coarse iterations with time step $\tau_{coarse} = 0.1$ and geometric adaptivity tolerance $\epsilon_{coarse} = 3\epsilon_{geom}$. Then at the first and second levels, the time steps and adaptivity tolerances were set to $\tau_1 = 0.02$, $\epsilon_1 = \epsilon_{geom}$ and $\tau_2 = 0.04$, $\epsilon_2 = 2\epsilon_{geom}$, respectively. We used the bacteria, the galaxy, and the dandelion images from Wikimedia Commons and the jet image from kiwiaircraftimages.com for our experiments. These images are of size 580×488, 2212×1263, 1024×768, and 600×380, respectively.

Space adaptivity. We tested the effectiveness of space adaptivity. Geometric adaptivity adjusts the curve resolution with respect to varying curvature. Data-driven adaptivity, on the other hand, refines the domain meshes to resolve the variations in the image function. This is illustrated with an example in Figure 5.1. Sample meshes generated during the optimization are also shown in Figure 6.4.

Convergence properties of $\boldsymbol{L^2}$ flow versus $\boldsymbol{H^1}$ flow. As these two flows yield different velocities, we used different time steps for each: $\tau_1 = 0.2$, $\tau_2 = 0.4$, $\tau_{coarse} = 1.0$ for the L^2 flow; $\tau_1 = 0.02$, $\tau_2 = 0.04$, $\tau_{coarse} = 0.1$ for the H^1 flow. We observed that the H^1 flow converged in fewer iterations than the L^2 flow and the evolution of the curves was smoother. Moreover, a single H^1 iteration was cheaper in terms of computation than an L^2 iteration. This is because for the H^1 flow we were able to use explicit time stepping. To be more specific, we were able to use (4.24c) to compute the H^1 velocity, which required a single matrix inversion with linear time complexity. On the other hand, to compute the L^2 velocity, we used (4.26) and solved for V iteratively at a cost of several matrix-vector products with the coefficient matrix $(\gamma\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + A_\alpha + M_\beta)$. This aspect of our method is in contrast with the level set implementation in [18], where a Newton-type step is more costly than an L^2 gradient descent step. The results of the experiments on a bacteria image are given in Figures 6.1, 6.2, and 6.3.

Denoising properties. The Mumford–Shah model (1.2) can be used for denoising as well as segmentation. Thus we tested the model on images with varying degrees of noise and obtained smoothed versions, such that the smoothing was confined to the uniform regions and did not take place across boundaries. We observed that the number of iterations required increased with the amount of noise. These results are shown in Figures 6.5 and 6.6. We also observed that the performance on this problem depended on the parameter μ. Smaller values of μ resulted in more iterations.

Dependence on model parameters. We examined the sensitivity of the results with respect to the model parameters μ and γ. The experiments were in agreement with their role as weights in the energy (1.2). Increasing μ resulted in smoother approximations, whereas increasing γ resulted in smoother curves. We observed, however, that depending on the input image, the segmentation result may be sensitive to the parameter values. See Figures 6.7, 6.8, and 6.9.

Choice of initial curves. In the final set of experiments, we tested the model sensitivity with respect to the choice of initial curves. In a number of examples, we observed that we were able to attain qualitatively similar results starting with significantly different initial curves; see Figure 6.10. It would nevertheless be misleading to think that the algorithm produces the sought segmentation regardless of the choice of initial curves. The energy (1.2) is *nonconvex* and at each step of the optimization we make use of only local information, i.e., the shape derivatives. Therefore, we cannot in general expect to obtain similar results starting with different initial curves, nor can we expect to end up with similar energy values. We can only expect to reach a local minimum; see Figure 6.10(d). An important related question that we do not explore in this paper is what would be a *good* starting configuration. Some ideas based on topological derivatives have been proposed for related problems in [15] and [16].

Computational cost. In most of the experiments we obtained the segmentation in less than a minute on a laptop computer with a P4 2.66 GHz processor and 512 MB memory. We observed that this result depended not on the image resolution but rather on the variation within the image. As our algorithm adapts the number of elements with respect to variation or detail in the image, a higher amount of variation results in more elements, thereby increasing the computational cost. This is why our experiments on the noisy jet images took up to 3.5 minutes although the jet image of Figure 6.6 has the smallest resolution. Through the optimization process, the cost of computation was dominated by the domain solves. Although the cost of a single domain solve and the curve computations in a single iteration were roughly comparable, one step of the optimization routine with backtracking required multiple domain solves, which added up. In the domain solves, about one-third of the cost came from adaptivity and assembly of the linear system and about two-thirds came from the solution of the linear system. Mesh generation introduced an additional 5% to 10% cost. On the other hand, curve computations were somewhat evenly distributed among mesh adaptivity, assembly of the linear system, solution of the linear system, topology corrections, and other overhead, with the linear solves having a slightly larger share.

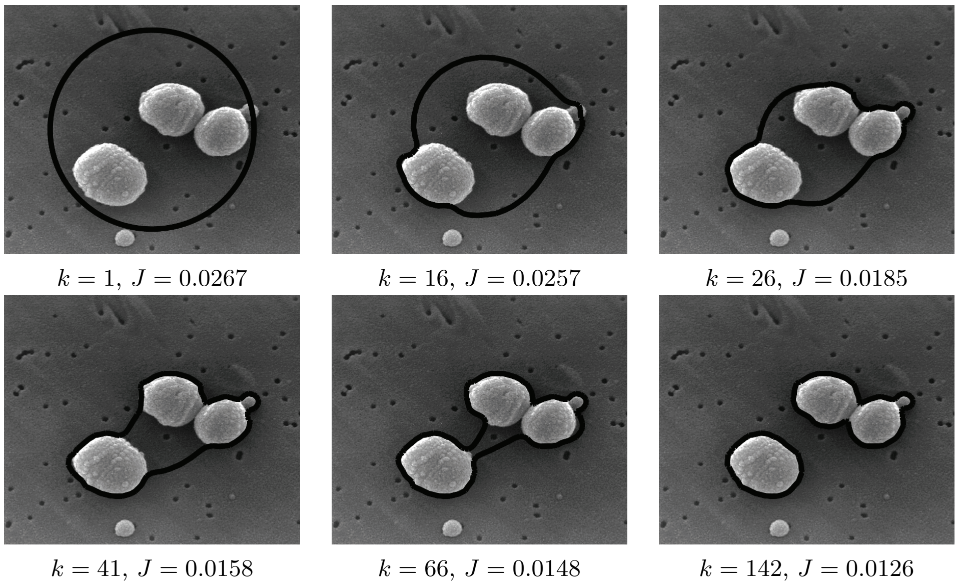


FIG. 6.1. **Curve evolution by H^1 flow.** *Segmentation of the bacteria image of size* 580×488 *using the* H^1 *flow with parameters* $\mu = 5 \times 10^{-3}$, $\gamma = 1.5 \times 10^{-3}$. *The* H^1 *flow produced the segmentation in* 142 *iterations and* 43*s. The curves* Γ *superimposed on the images, and energy values* J *for the first, some intermediate, and last iterations are presented. The corresponding piecewise smooth approximations* u_i *are given in Figure* 6.2.

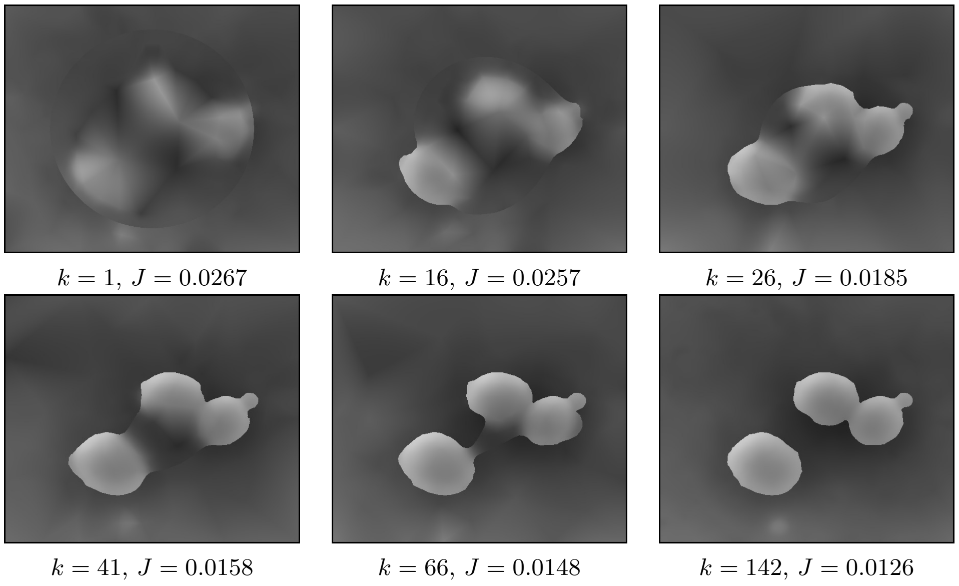


FIG. 6.2. **Texture evolution by H^1 flow.** *Evolution of the piecewise smooth approximation u for the bacteria image obtained with the H^1 flow and parameters $\mu = 5 \times 10^{-3}$, $\gamma = 1.5 \times 10^{-3}$. The corresponding curves are shown in Figure* 6.1.

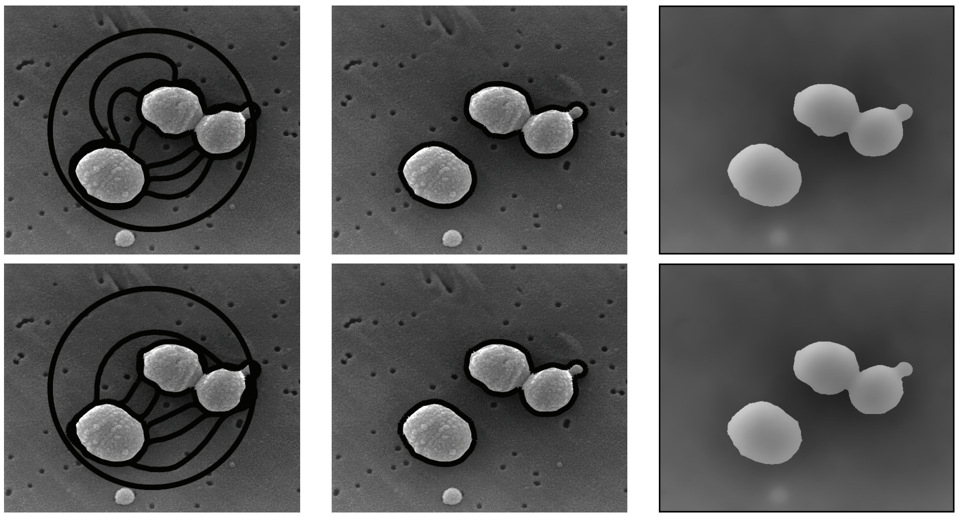


FIG. 6.3. **Comparison of L^2 flow and H^1 flow.** *Segmentation and approximation results for the L^2 flow (top row) and the H^1 flow (bottom row). The columns from left to right display the curve evolution, the final segmentation, and the final smooth approximation u to the image. The L^2 flow converged in* 586 *iterations and* $2m\,51s$. *The H^1 flow converged in* 142 *iterations, about one-fourth of the L^2 flow, and* $43s$. *This illustrates the benefits of H^1 flow with explicit time stepping over a semi-implicit L^2 flow, which are apparent although the L^2 flow uses* 400 *coarse iterations, as opposed to* 100 *for the H^1 flow, to speed up the early computations.*

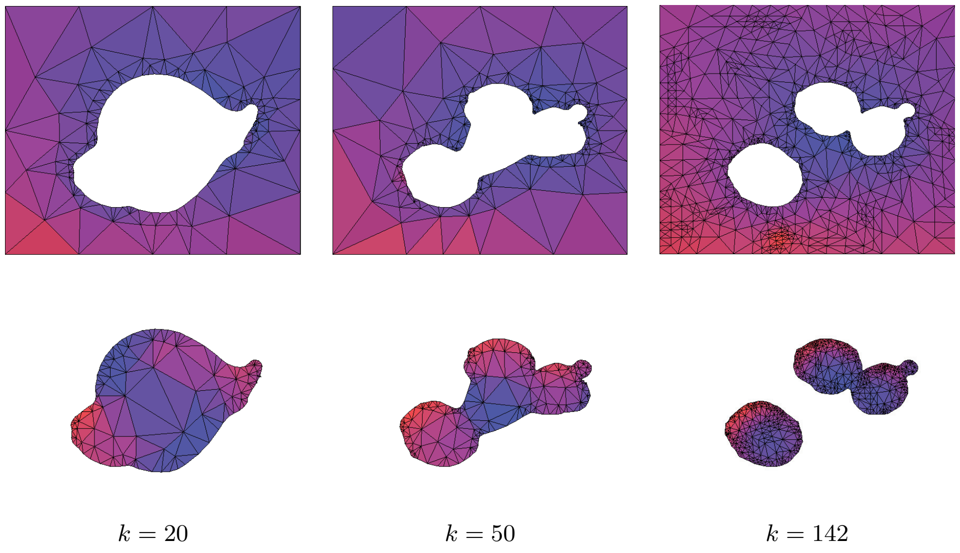


FIG. 6.4. **Background and foreground meshes.** *Snapshots of the finite element meshes for* Ω_2 *(top row) and* Ω_1 *(bottom row) corresponding to the* H^1 *flow. These meshes were generated by TRIANGLE* [21] *from the curves* Γ. *Note that the mesh is locally refined in the last image as data-driven adaptivity is switched on, after the initial coarse iterations, to resolve the variations in the image.*

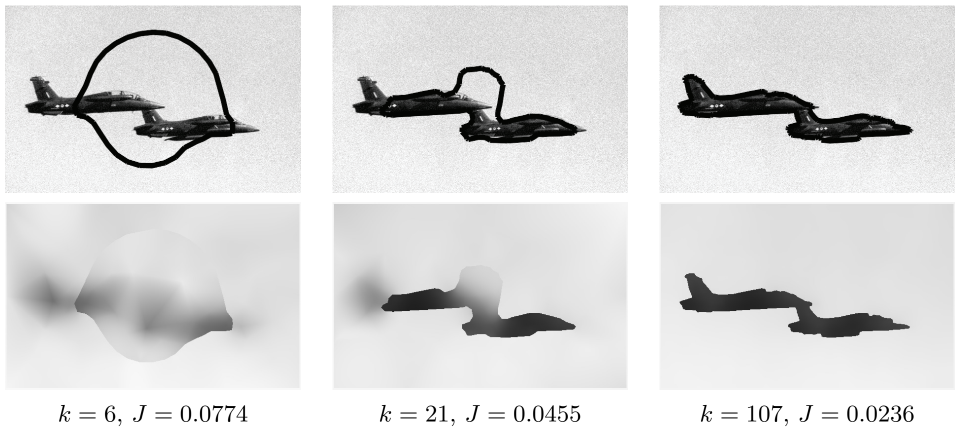


FIG. 6.5. **Denoising.** *Use of the Mumford–Shah model* (1.2) *for image denoising using the* H^1 *flow with parameters* $\mu = 2 \times 10^{-2}$, $\gamma = 2 \times 10^{-3}$. *We added a* 10% *noise level to the image intensity of a jet image of size* 600×380. *As the optimization process tries to position the curve at the object boundaries, the uniform regions are distinguished and the PDE* (1.3) *realizes the smoothing in these regions. Our algorithm terminated in* 107 *iterations and* 47*s. The iteration counter* k *of the* H^1 *flow and the corresponding energy values* J *are displayed. The top row shows the evolution of curve* Γ *superimposed on the images, and the bottom row shows the evolution of the approximation* u.

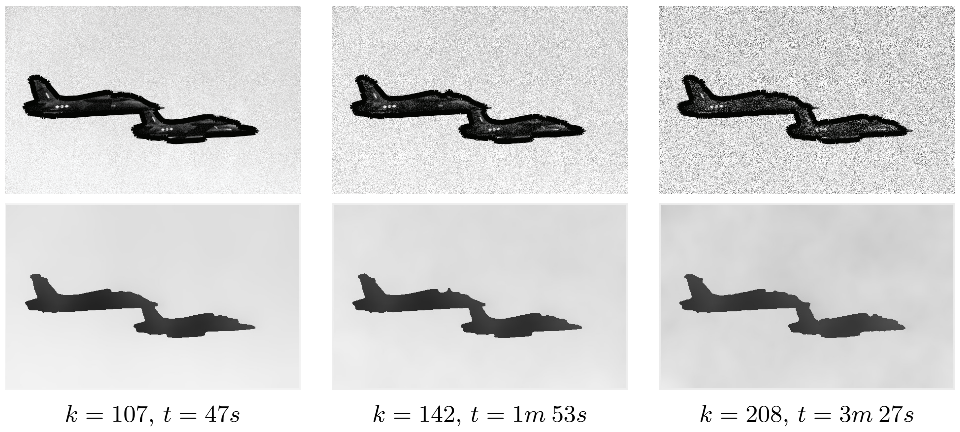


FIG. 6.6. **Sensitivity to noise.** *Study of the impact of noise on the optimization process. We added* 10%, 25%, *and* 50% *noise level to the jet image and computed the* H^1 *flow for the Mumford–Shah model* (1.2) *with parameters* $\mu = 2 \times 10^{-2}$, $\gamma = 2 \times 10^{-3}$. *The computational results are shown in the left, center, and right columns, respectively. The top row shows the computed edges superimposed on the noisy images, and the bottom row shows the denoised approximations. The number of iterations and computations increased with the amount of noise. In particular, for the last example, we chose* $N_{coarse} = 100$ *(instead of* $N_{coarse} = 50$*) to speed up the calculations. As the amount of noise increases, the background and the foreground become less and less dissimilar, thereby making it more difficult to obtain a proper segmentation. Moreover, as the variation in the image increases with noise, the data-driven adaptivity algorithm refines the mesh to improve the image representation, which in turn increases the computational cost. Other experiments also revealed the effect of the smoothing parameter* μ *on the performance: using smaller* μ *increased the number of iterations considerably for the noisier images.*

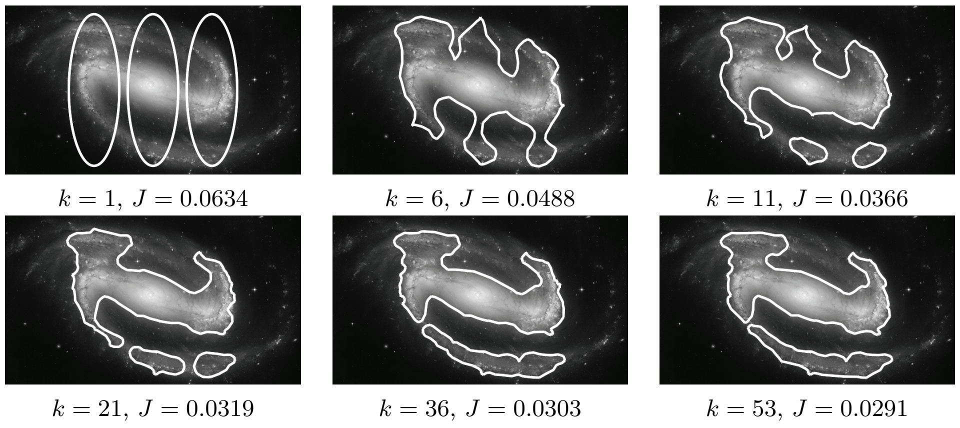
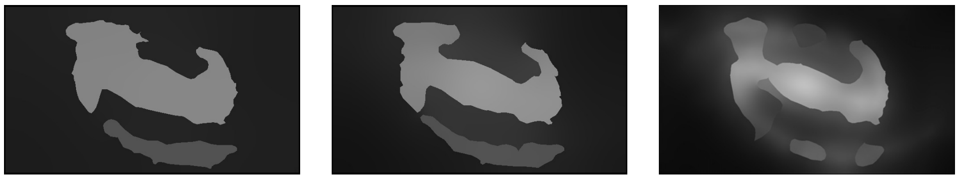
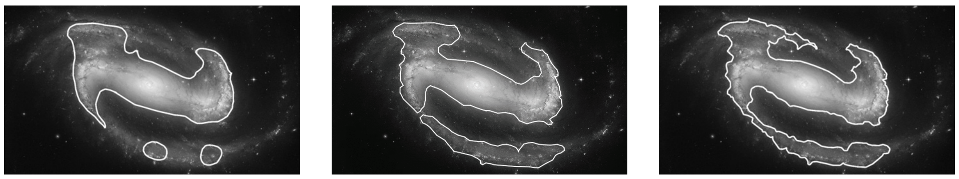


FIG. 6.7. **Segmentation of images without sharp edges.** *Segmentation of the galaxy image of size* 2212×1262, *using the Mumford–Shah model* (1.2) *with model parameters* $\mu = 0.25$, $\gamma = 5 \times 10^{-4}$. *We show the curves given by the* H^1 *flow and the energy values* J *at several iterations* k. *Although there are no sharp edges defined in this image, our algorithm gives a very satisfactory segmentation in* 53 *iterations and* 20*s.*



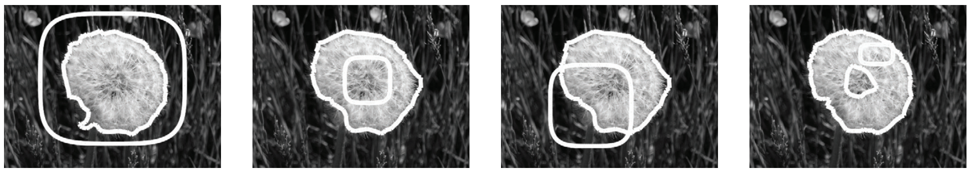
$\mu = 2.0$, $k = 66$, $t = 25s$ | $\mu = 0.25$, $k = 53$, $t = 20s$ | $\mu = 0.01$, $k = 120$, $t = 1m\,51s$

FIG. 6.8. **Sensitivity to μ.** *Segmentation results for different values of parameter μ in the Mumford–Shah energy* (1.2) *and $\gamma = 5 \times 10^{-4}$. The approximation u for the most satisfactory segmentation (visually) is depicted in the middle. When increasing μ up to* 2.0 *the regularized foreground and background given by u_1 and u_2, respectively, become constant. On the other hand, the approximation u gets somewhat less smooth as we decrease μ to* 0.01.



$\gamma = 2 \times 10^{-3}$, $k = 193$, $t = 1m\,55s$ | $\gamma = 5 \times 10^{-4}$, $k = 53$, $t = 20s$ | $\gamma = 1 \times 10^{-4}$, $k = 60$, $t = 37s$

FIG. 6.9. **Sensitivity to γ.** *Segmentation results for different values of parameter γ in the Mumford–Shah energy* (1.2) *and $\mu = 0.25$. This experiment clearly shows how the penalization of length in the Mumford–Shah model* (1.2) *constrains the curves. For the highest value of the parameter, $\gamma = 2 \times 10^{-3}$, we obtained shorter and smoother curves. For the lowest value, $\gamma = 1 \times 10^{-4}$, we obtained longer and much rougher curves.*



(a) $J = 0.0364$, $k = 74$, $t = 28s$ | (b) $J = 0.0379$, $k = 66$, $t = 35s$ | (c) $J = 0.0371$, $k = 62$, $t = 31s$ | (d) $J = 0.0369$, $k = 137$, $t = 56s$

FIG. 6.10. **Sensitivity to initial curves.** *Segmentation with the H^1 flow for the Mumford–Shah model* (1.2) *with parameters $\mu = 0.02$, $\gamma = 1 \times 10^{-3}$, starting from different initial curves Γ_0:* (a) Γ_0 *outside the object,* (b) Γ_0 *inside the object,* (c) Γ_0 *partially overlapping with the object,* (d) Γ_0 *small inside the object. The number of iterations k, the corresponding final energies J, and CPU times are presented. The first three examples show qualitatively similar segmentations, despite starting from quite different initial curves. However, the last example reveals the nonconvex nature of the energy* (1.2)*: the small initial curve (rounded rectangle), which is inside the dandelion, results in two final curves separating the dandelion into a darker center and a lighter periphery. This final configuration is a local minimum for this problem, which is to be expected for an algorithm based on local information such as shape derivatives.*

Acknowledgment. We are grateful to the anonymous referees for their valuable comments and suggestions that helped improve this article, particularly the experiments section.

REFERENCES

[1] L. Ambrosio and V. M. Tortorelli, *On the approximation of free discontinuity problems*, Boll. Un. Mat. Ital. B, 6 (1992), pp. 105–123.

[2] E. Bänsch, *Finite element discretization of the Navier-Stokes equations with a free capillary surface*, Numer. Math., 88 (2001), pp. 203–235.

[3] E. Bänsch, P. Morin, and R. H. Nochetto, *A finite element method for surface diffusion: The parametric case*, J. Comput. Phys., 203 (2005), pp. 321–343.

[4] B. Bourdin and A. Chambolle, *Implementation of an adaptive finite-element approximation of the Mumford-Shah functional*, Numer. Math., 85 (2000), pp. 609–646.

[5] T. Chan and L. Vese, *A level set algorithm for minimizing the Mumford-Shah functional in image processing*, in Proceedings of the First IEEE Workshop on Variational and Level Set Methods in Computer Vision, 2001, pp. 161–168.

[6] T. A. Davis, *Algorithm* 832 : *UMFPACK v*4.3*—an unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 196–199.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, 1997.

[8] K. Deckelnick, G. Dziuk, and C. M. Elliott, *Computation of geometric partial differential equations and mean curvature flow*, Acta Numer., 14 (2005), pp. 139–232.

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

[10] G. Doğan, *Topological changes and adaptivity for curve evolution*, in preparation.

[11] G. Doğan, P. Morin, R. H. Nochetto, and M. Verani, *Discrete gradient flows for shape optimization and applications*, Comput. Method Appl. M., 196 (2007), pp. 3898–3914.

[12] G. Doğan, *A Variational Shape Optimization Framework for Image Segmentation*, Ph.D. thesis, University of Maryland, College Park, 2006.

[13] X. Feng and A. Prohl, *Analysis of gradient flow of a regularized Mumford-Shah functional for image segmentation and image inpainting*, M2AN, 38 (2004), pp. 291–320.

[14] J. M. Fried, *Image Segmentation Using Adaptive Finite Elements*, Technical report, Universität Freiburg, Germany, 2003.

[15] L. He and S. J. Osher, *Solving the Chan–Vese model by a multiphase level set algorithm based on the topological derivative*, in Proceedings of the First International Conference on Scale Space Variational Methods in Computer Vision, 2007.

[16] M. Hintermüller, *Fast level set based algorithms using shape and topological sensitivity*, Control Cybernet., 34 (2005), pp. 305–324.

[17] M. Hintermüller and W. Ring, *A second order shape optimization approach for image segmentation*, SIAM J. Appl. Math., 64 (2003/04), pp. 442–467.

[18] M. Hintermüller and W. Ring, *An inexact Newton-CG-type active contour approach for the minimization of the Mumford-Shah functional*, J. Math. Imaging Vision, 20 (2004), pp. 19–42.

[19] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685.

[20] A. Schmidt and K. Siebert, *Design of Adaptive Finite Element Software*, Lecture Notes in Comput. Sci. and Engrg. 42, Springer-Verlag, Berlin, 2005.

[21] J. R. Shewchuk, *Triangle: Engineering a* 2*D Quality Mesh Generator and Delaunay Triangulator*, in Applied Computational Geometry: Towards Geometric Engineering, M. Lin and D. Manocha, eds., Lecture Notes in Comput. Sci. 1148, Springer-Verlag, Berlin, 1996.

[22] J. Sokołowski and J.-P. Zolésio, *Introduction to Shape Optimization*, Springer Ser. Comput. Math. 16, Springer-Verlag, Berlin, 1992.

[23] A. Tsai, A. Yezzi, and A. Willsky, *Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification*, IEEE Trans. Image Process., 10 (2001), pp. 1169–1186.

[24] L. A. Vese and T. F. Chan, *A multiphase level set framework for image segmentation using the Mumford and Shah model*, Internat. J. Computer Vision, 50 (2002), pp. 271–293.