Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.OA-2017-0074

Vol. **11**, No. 2, pp. 211-234
May 2018

Selection of Regularization Parameter in the Ambrosio-Tortorelli Approximation of the Mumford-Shah Functional for Image Segmentation

Yufei Yu[1] and Weizhang Huang[1,*]

[1] *Department of Mathematics, the University of Kansas, Lawrence, Kansas 66045, U.S.A*

Received 20 June 2017; Accepted (in revised version) 26 October 2017

Abstract. The Ambrosio-Tortorelli functional is a phase-field approximation of the Mumford-Shah functional that has been widely used for image segmentation. It has the advantages of being easy to implement, maintaining the segmentation ability, and Γ-converging to the Mumford-Shah functional as the regularization parameter goes to zero. However, it has been observed in actual computation that the segmentation ability of the Ambrosio-Tortorelli functional varies significantly with different values of the parameter and it even fails to Γ-converge to the original functional for some cases. In this paper we present an asymptotic analysis on the gradient flow equation of the Ambrosio-Tortorelli functional and show that the functional can have different segmentation behavior for small but finite values of the regularization parameter and eventually loses its segmentation ability as the parameter goes to zero when the input image is treatd as a continuous function. This is consistent with the existing observation as well as the numerical examples presented in this work. A selection strategy for the regularization parameter and a scaling procedure for the solution are devised based on the analysis. Numerical results show that they lead to good segmentation of the Ambrosio-Tortorelli functional for real images.

AMS subject classifications: 65M50, 65M60, 94A08, 35K55
Key words: Regularization, image segmentation, phase-field model, moving mesh, mesh adaptation, finite element method.

1. Introduction

Segmentation for a given image is a process to find the edges of objects and partitions the image into separate parts that are relatively smooth. It has been achieved in mathematics by minimizing functionals and multiple theories have been developed. One of the most commonly used functionals, proposed by Mumford and Shah [28], takes the form

$$E[u,\Gamma]=\frac{\alpha}{2}\int_{\Omega\setminus\Gamma}|\nabla u|^2dx+\beta H^1(\Gamma)+\frac{\gamma}{2}\int_{\Omega}(u-g)^2dx, \tag{1.1}$$

*Corresponding author. *Email addresses:* y920y782@ku.edu (Y. F. Yu), whuang@ku.edu (W. Z. Huang)

where Ω is a rectangular domain, α, β, and γ are positive parameters, g is the grey level of the input image, u is the target image, Γ denotes the edges of the objects in the image, and $H^1(\Gamma)$ is the one-dimensional Hausdorff measure. Upon minimization, u is close to g, ∇u is small on $\Omega \setminus \Gamma$, and Γ is as short as possible. An optimal image is thus close to the original one and almost piecewise constant. Moreover, the terms in (1.1) represent different and often conflicting objectives, making its minimization and thus image segmentation an interesting and challenging topic to study.

To avoid mathematical difficulties caused by the $H^1(\Gamma)$ term, De Giorgi et al. [8] propose an alternative functional as

$$F[u] = \frac{\alpha}{2}\int_\Omega |\nabla u|^2 dx + \beta H^1(S_u) + \frac{\gamma}{2}\int_\Omega |u-g|^2 dx, \tag{1.2}$$

where S_u is the jump set of u. They show that (1.2) has minimizers in $SBV(\Omega)$ (the space of special functions of bounded variation) and is equivalent to (1.1) in the sense that if $u \in \Omega$ is a minimizer of (1.2), then $(u, \overline{S_u})$ is a minimizer of (1.1), where $\overline{S_u}$ is the closure of S_u.

Although it is a perfectly fine functional to study in mathematics, (1.2) is not easy to implement in actual computation due to the fact that the jump set of the unknown function and its Hausdorff measure are extremely difficult, if not impossible, to compute. To avoid this difficulty, Ambrosio and Tortorelli [1] propose a regularized version as

$$AT_\epsilon[u,\phi] = \frac{\alpha}{2}\int_\Omega (\phi^2 + k_\epsilon)|\nabla u|^2 dx + \beta\int_\Omega \left(\epsilon|\nabla\phi|^2 + \frac{1}{4\epsilon}(1-\phi)^2\right) dx + \frac{\gamma}{2}\int_\Omega |u-g|^2 dx, \tag{1.3}$$

where $\epsilon > 0$ is the regularization parameter, $k_\epsilon = o(\epsilon)$ is a parameter used to prevent the functional from becoming degenerate, and ϕ is a new unknown variable which ideally is an approximation of the characteristic function for the complement of the jump set of u, i.e.,

$$\phi(x) \approx \chi_u(x) \equiv \begin{cases} 0, & \text{if } x \in S_u, \\ 1, & \text{if } x \notin S_u. \end{cases} \tag{1.4}$$

They show that AT_ϵ has minimizers $u \in SBV(\Omega)$ and $\phi \in L^2(\Omega)$ and Γ-converges to $F(u)$. Γ-convergence, first introduced by Ennio de Giorgi, is a concept that guarantees the minimizer of a regularized functional converges to that of the original functional as the regularization parameter goes to 0.

The first finite element approximation for the functional AT_ϵ is given by Bellettini and Coscia [4]. They seek linear finite element approximations u_h and ϕ_h to minimize

$$\begin{aligned} AT_{\epsilon,h}[u_h,\phi_h] = \frac{\alpha}{2}\int_\Omega (\phi_h^2 + k_\epsilon)|\nabla u_h|^2 dx + \beta\int_\Omega \left(\epsilon|\nabla\phi_h|^2 + \frac{1}{4\epsilon}\pi_h((1-\phi_h)^2)\right) dx \\ + \frac{\gamma}{2}\int_\Omega \pi_h((u_h - g_\epsilon)^2) dx, \end{aligned} \tag{1.5}$$

where π_h is the linear Lagrange interpolation operator and g_ϵ is a smooth function which converges to g in the L^2 norm as $\epsilon \to 0$. They show that $AT_{\epsilon,h}$ Γ-converges to $F(u)$ when

the maximum element diameter is chosen as $h = o(\epsilon)$. A finite element discretization of AT_ϵ was first applied to image segmentation by Bourdin [5]. It should be pointed out that Feng and Prohl [13] have established the existence and uniqueness of the solution to an initial-boundary value problem (IBVP) of the gradient flow equation of (1.3) and proven that a finite element approximation of the IBVP converges to the continuous solution as the mesh is refined.

It is noted that the Ambrosio-Tortorelli functional (1.3) is actually a phase-field approximation of the Mumford-Shah functional (1.1). Phase-field modeling has been used widely in science and engineering to handle sharp interfaces, boundaries, and cracks in numerical simulation of problems such as dendritic crystal growth [23, 36], multiple-fluid hydrodynamics [24, 31, 32, 37], and brittle fracture [6, 14, 27]. It employs a phase-field variable ϕ, which depends on a regularization parameter ϵ describing the actual width of the smeared interfaces, to indicate the location of the interfaces. Phase-field modeling has the advantage of being able to handle complex interfaces without relying on their explicit description. Mathematically, phase-field models such as (1.3) have been studied extensively (e.g., see [1]) for Γ-convergence. However, few studies have been published for the role of the regularization parameter in actual simulation. It is a common practice that a specific value of ϵ is used without discussion or explanation in phase-field modeling. Even worse, it has been observed [26, 29, 34] that a phase-field model for brittle fracture simulation does not Γ-converge as $\epsilon \to 0$ and ϵ can be interpreted as a material parameter since its choice influences the "critical stress". More recently, ϵ has been chosen as a material parameter based on theoretical and experimental analyses of a simplified phase-field model [11].

The objective of this paper is to study the effects of the regularization parameter in (1.3), a special example of phase-field modeling in image segmentation. We consider the gradient flow equation of the functional AT_ϵ subject to a homogeneous Neumann boundary condition and carry out an asymptotic analysis for the solution of the corresponding IBVP as $\epsilon \to 0$. We show that, when g is continuous, the functional can have different segmentation behavior for small but finite ϵ and eventually loses its segmentation ability for infinitesimal ϵ. This is consistent with the existing observation in phase-field modeling and with the numerical examples to be presented. The analysis is also used to devise a selection strategy for ϵ and a scaling for u and g. Numerical results with real images confirm that the strategies can lead to good segmentation of AT_ϵ in the sense that ϕ is close to the characteristic function of the jump set of the approximation u of g (cf. (1.4)).

An outline of the paper is as follows. The asymptotic analysis is given in Section 2, followed by a moving mesh finite element method in Section 3. Illustrative numerical examples are given in Section 4. A selection strategy for ϵ and scaling of u and g as well as examples with several real input images are presented in Section 5. Finally, Section 6 contains conclusions.

2. Behavior of the minimizer of AT_ϵ as $\epsilon \to 0$ for continuous g

We first explain why we consider g as a continuous function. In image segmentation, the function g represents an image and is given the grey-level values at the pixels. Gener-

ally speaking, the values of g at points other than the pixels are needed in finite element computation. These values are computed commonly through (linear) interpolation of the values at the pixels. This means that g is treated as a continuous function in finite element computation and such a treatment is independent of the regularization parameter. Thus we consider g as a continuous function and want to study how the minimizer of AT_ϵ behaves as $\epsilon \to 0$.

To this end, we consider the gradient flow equation of functional (1.3),

$$\begin{cases} u_t = \alpha\nabla\cdot((k_\epsilon+\phi^2)\nabla u) - \gamma(u-g), \\ \phi_t = 2\beta\epsilon\Delta\phi - \alpha|\nabla u|^2\phi + \frac{\beta}{2\epsilon}(1-\phi), \end{cases} \qquad t>0, \quad x\in\Omega \tag{2.1}$$

subject to the homogeneous Neumann boundary condition

$$\frac{\partial u}{\partial n} = \frac{\partial \phi}{\partial n} = 0 \quad \text{for} \quad x\in\partial\Omega, \tag{2.2}$$

and the initial condition

$$u(x,0) = u^0(x), \quad \phi(x,0) = \phi^0(x), \quad x\in\Omega. \tag{2.3}$$

This IBVP has been studied and used to find a local minimizer of (1.3) (as a steady-state solution) by a number of researchers. For example, Feng and Prohl [13] have established the existence and uniqueness of the solution of the IBVP and proven that a finite element approximation converges to the continuous solution as the mesh is refined.

By assumption, $g \in C^0(\Omega)$. Then we can expect that the solution u and ϕ of the IBVP is smooth. To see the behavior of the solution as $\epsilon \to 0$, we consider the asymptotic expansion of u and ϕ as

$$u = u^{(0)} + \epsilon u^{(1)} + \epsilon^2 u^{(2)} + \cdots, \tag{2.4a}$$

$$\phi = \phi^{(0)} + \epsilon\phi^{(1)} + \epsilon^2\phi^{(2)} + \cdots. \tag{2.4b}$$

Inserting these into (2.1), we get

$$\begin{aligned} u_t^{(0)} + \epsilon u_t^{(1)} + \mathcal{O}(\epsilon^2) = \alpha\nabla\cdot\Big[\Big(o(\epsilon) + (\phi^{(0)} + \epsilon\phi^{(1)} + o(\epsilon))^2\Big)\nabla(u^{(0)} \\ + \epsilon u^{(1)} + \mathcal{O}(\epsilon^2))\Big] - \gamma\left(u^{(0)} + \epsilon u^{(1)} + \mathcal{O}(\epsilon^2) - g\right), \end{aligned} \tag{2.5a}$$

$$\begin{aligned} \phi_t^{(0)} + \epsilon\phi_t^{(1)} + \mathcal{O}(\epsilon^2) = 2\beta\epsilon(\Delta\phi^{(0)} + \epsilon\Delta\phi^{(1)} + \mathcal{O}(\epsilon^2)) - \alpha\Big|\nabla u^{(0)} + \epsilon\nabla u^{(1)} \\ + \mathcal{O}(\epsilon^2)\Big|^2 d\cdot\left(\phi^{(0)} + \epsilon\phi^{(1)} + \mathcal{O}(\epsilon^2)\right) + \frac{\beta}{2\epsilon}\left(1 - \phi^{(0)} - \epsilon\phi^{(1)} - \mathcal{O}(\epsilon^2)\right), \end{aligned} \tag{2.5b}$$

where we have used $k_\epsilon = o(\epsilon)$. Collecting the $\mathcal{O}(1)$ terms in (2.5a), we have

$$u_t^{(0)} = \alpha\Delta u^{(0)} - \gamma(u^{(0)} - g), \quad \text{in } \Omega. \tag{2.6}$$

Similarly, collecting the $\mathcal{O}(1/\epsilon)$ terms and $\mathcal{O}(1)$ terms in (2.5b) we get

$$\frac{\beta}{2}\left(1-\phi^{(0)}\right) = 0, \qquad \phi_t^{(0)} = -\alpha\Big|\nabla u^{(0)}\Big|^2\phi^{(0)} - \frac{\beta}{2}\phi^{(1)}.$$

From these we obtain

$$\phi = 1 - \epsilon \frac{2\alpha}{\beta} \left| \nabla u^{(0)} \right|^2 + \mathcal{O}(\epsilon^2). \tag{2.7}$$

Like u, $u^{(0)}$ also satisfies the homogeneous Neumann boundary condition. Since $g \in C^0(\Omega)$, it can be shown (e.g., see [12]) that $u^{(0)}$ has continuous second-order derivatives and thus $\nabla u^{(0)}$ is bounded. Combining this with (2.7) we conclude that $\phi \to 1$ as $\epsilon \to 0$. Since the boundaries between different objects in u are indicated by $\phi = 0$, this implies that u is a single object and there is no segmentation as $\epsilon \to 0$ when g is continuous. Moreover, u and thus $u^{(0)}$ are kept close to g and we can expect $\nabla u^{(0)}$ to be large in the places where ∇g is large. From (2.7) we can see that, for small but not infinitesimal ϵ, ϕ can become zero at places where ∇g is large. In this case, the functional will have good segmentation (cf. the numerical examples in Section 4). We have seen so far that the choice of the regularization parameter in (1.3) can be crucial for image segmentation: *different values of ϵ can lead to very different segmentation behavior of the functional and its segmentation ability will disappear as $\epsilon \to 0$.*

It should be emphasized that the above observation is not in contradiction with the theoretical analysis made in [1] for the Γ-convergence and segmentation ability of the functional (1.3). In [1], these properties are analyzed for $u \in SBV(\Omega)$, implicitly implying that u is discontinuous in general. The above analysis has been made under the assumption that g, and thus u are continuous although they may have large gradient from place to place.

It is instructive to see some transient behavior of the solution to the gradient flow equation. To simplify, we drop the diffusion term in the second equation in (2.1) and get

$$\phi_t = -\alpha |\nabla u|^2 \phi + \frac{\beta}{2\epsilon}(1 - \phi). \tag{2.8}$$

It has been proven in [13] that the solution of (2.1) satisfies $0 \le \phi \le 1$. From this we see that the first term on the right-hand side of (2.8) is nonpositive, which will make ϕ decrease, and the second term is nonnegative, making ϕ increase. These two terms will compete and reach an equilibrium state. Moreover, if $\phi = 1$, we have $\phi_t = -\alpha |\nabla u|^2 \le 0$, meaning that as long as $|\nabla u| \neq 0$, the first term will decrease ϕ until $\phi_t = 0$ is reached. Similarly, if $\phi = 0$, we have $\phi_t = \frac{\beta}{2\epsilon} > 0$, which means ϕ will increase until the system reaches its equilibrium. The equilibrium values of ϕ can be obtained by setting the right-hand side of (2.8) to be zero, i.e.,

$$\phi \approx \frac{\beta}{\beta + 2\epsilon\alpha |\nabla u|^2}. \tag{2.9}$$

Thus, the equilibrium value of ϕ is around 1 for smooth regions where ∇u is small and around 0 on edges where ∇u is large.

3. The adaptive moving mesh finite element method

In this section we describe an adaptive moving mesh finite element method for solving

the gradient flow equation (2.1). Recall that a crucial requirement for the Ambrosio-Tortorelli Approximation (1.3) of the Mumford-Shah functional is that ϵ must be small. Since the width of object edges is of the same order of ϵ, the size of the mesh elements around the edges should be of the same order of ϵ or smaller for any finite element approximation to be meaningful. On the other hand, the mesh elements do not have to be that small within each object where u and ϕ are smooth. Thus, mesh adaptation is necessary for the efficiency of the finite element computation. A numerical example illustrating the efficiency of mesh adaptation will be presented in Section 5. We use here the MMPDE moving mesh method [21, 22] that has been specially designed for time dependent problems.

It should be pointed out that a number of other moving mesh methods have been developed in the past and there is a vast literature in the area. The interested reader is referred to the books or review articles [2, 3, 7, 22, 33] and references therein. Moreover, moving mesh methods have been successfully applied to phase-field models, e.g., see [10, 25, 30, 35, 38–40].

It is remarked that the spatial domain Ω is typically a rectangular domain for image segmentation. However, finite element computation is not subject to this restriction. Moreover, we will consider examples in both one and two dimensions for illustrative purpose in the next section. For these reasons, we consider Ω as a general polygonal domain in d-dimensions ($d = 1$ and 2).

3.1. Finite element discretization

We now consider the integration of (2.1) up to a finite time $t = T$. Denote the time instants by

$$0 = t_0 < t_1 < \cdots < t_{n_f} = T.$$

For the moment, we assume that a simplicial mesh for Ω is given at these time instants, i.e., $\mathcal{T}_h^n$, $n = 0, \cdots, n_f$, which are considered as the deformation from each other and have the same number of the elements (N) and the vertices (N_v) and the same connectivity. Such a mesh is generated using the MMPDE moving mesh strategy to be described in Section 3.2.

For the finite element discretization of (2.1), the mesh is considered to change linearly between t_n and t_{n+1}, i.e.,

$$x_j(t) = \frac{t - t_n}{t_{n+1} - t_n} x_j^{n+1} + \frac{t_{n+1} - t}{t_{n+1} - t_n} x_j^n, \quad j = 1, \cdots, N_v, \quad t \in (t_n, t_{n+1}),$$

where $x_j(t)$, x_j^n, and x_j^{n+1} ($j = 1, \cdots, N_v$) denote the coordinates of the vertices of $\mathcal{T}_h(t)$, $\mathcal{T}_h^n$, and $\mathcal{T}_h^{n+1}$, respectively. Denote the linear basis function associated with the j-th vertex by $\psi_j(\cdot, t)$ and let $V_h(t) = \text{span}\{\psi_1, \cdots, \psi_{N_v}\}$. Then, the weak formulation for the linear finite element approximation for (2.1) is to find $u_h(\cdot, t)$, $\phi_h(\cdot, t) \in V^h(t)$, $0 < t \leq T$ such

that

$$\int_\Omega \frac{\partial u_h}{\partial t} v dx = -\alpha \int_\Omega (k_\epsilon + \phi_h^2) \nabla u_h \cdot \nabla v dx - \gamma \int_\Omega (u_h - g) v dx = 0, \quad \forall v \in V^h(t), \tag{3.1a}$$

$$\int_\Omega \frac{\partial \phi_h}{\partial t} v dx = -2\beta\epsilon \int_\Omega \nabla \phi_h \cdot \nabla v dx - \alpha \int_\Omega |\nabla u_h|^2 \phi_h v dx + \frac{\beta}{2\epsilon} \int_\Omega (1 - \phi_h) v dx, \quad \forall v \in V^h(t). \tag{3.1b}$$

This is almost the same as that for the finite element approximation on a fixed mesh. The main difference lies in the time differentiation. To see this, expressing u_h into

$$u_h(x,t) = \sum_{i=1}^{N_v} u_i(t) \psi_i(x,t), \tag{3.2}$$

and differentiating it with respect to time, we get

$$\frac{\partial u_h(x,t)}{\partial t} = \sum_{i=1}^{Nv} \frac{du_i}{dt} \psi_i(x,t) + \sum_{i=1}^{Nv} \frac{\partial \psi_i(x,t)}{\partial t} u_i(t).$$

It is known (e.g., see [22]) that

$$\frac{\partial \psi_i}{\partial t} = -\nabla \psi_i \cdot \dot{X}, \quad \text{a.e. in } \Omega,$$

where

$$\dot{X} = \sum_{i=1}^{N_v} \dot{x}_i \psi_i(x,t),$$

and $\dot{x}_i$'s denote the nodal mesh speed. Combining the above results, we obtain

$$\frac{\partial u_h}{\partial t} = \sum_{i=1}^{N_v} \frac{du_i}{dt} \psi_i - \nabla u_h \cdot \dot{X}.$$

Similarly,

$$\phi_h(x,t) = \sum_{i=1}^{Nv} \phi_i(t) \psi_i(x,t), \qquad \frac{\partial \phi_h}{\partial t} = \sum_{i=1}^{N_v} \frac{d\phi_i}{dt} \psi_i - \nabla \phi_h \cdot \dot{X}.$$

From these we can see that mesh movement introduces an extra convection term. Inserting these into (3.1) and taking $v = \psi_j$ successively, we can rewrite (3.1) into an ODE system in the form

$$\begin{cases} M(X)\dot{U} = F(\dot{X}, X, \Phi, U, X), \\ M(X)\dot{\Phi} = G(\dot{X}, X, \Phi, U, X), \end{cases} \tag{3.3}$$

where $M(X)$ is the mass matrix. This system for U and Φ is integrated from t_n to t_{n+1} using the fifth-order Radau IIA method (see, e.g., Hairer and Wanner [16]), with a variable time step being selected based on a two-step error estimator [15].

3.2. The MMPDE moving mesh strategy

We now describe the generation of $\mathscr{T}_h^{n+1}$ using the MMPDE moving mesh strategy [22]. For this purpose, we denote the physical mesh by $\mathscr{T}_h = \{x_1, \cdots, x_{N_v}\}$, the reference computational mesh by $\hat{\mathscr{T}}_{c,h} = \{\hat{\xi}_1, \cdots, \hat{\xi}_{N_v}\}$ (which is chosen as the very initial physical mesh in our computation), and the computational mesh $\mathscr{T}_{c,h} = \{\xi_1, \cdots, \xi_{N_v}\}$. We assume that all of these meshes have the same number of elements and vertices and the same connectivity. Then, for any element $K \in \mathscr{T}_h$ there exists a corresponding element $K_c \in \mathscr{T}_{c,h}$. We denote the affine mapping between K_c and K by F_K and its Jacobian matrix by F_K'.

A main idea of the MMPDE moving mesh strategy is to view any adaptive mesh as a uniform one in the metric specified by a certain tensor. A metric tensor (denoted by $\mathbb{M}$) is a symmetric and uniformly positive definite matrix-valued function defined on Ω. In our computation, we choose $\mathbb{M}$ to be a piecewise constant function depending on u_h as

$$\mathbb{M}_K = \det(|H_K|)^{-\frac{1}{d+4}}|H_K|, \quad \forall K \in \mathscr{T}_h, \tag{3.4}$$

where H_K is a recovered Hessian of u_h on element K, $|H_K| = Q\mathrm{diag}(|\lambda_1|, \cdots, |\lambda_d|)Q^T$, assuming that the eigen-decomposition of H_K is $Q\mathrm{diag}(\lambda_1, \cdots, \lambda_d)Q^T$, and $\det(|H_K|)$ is the determinant of $|H_K|$. The recovered Hessian in K is obtained by twice differentiating a local quadratic polynomial fitting in the least-squares sense to the nodal values of u_h at the neighboring vertices of the element. The form of (3.4) is known [18] optimal with respect to the L^2 norm of linear interpolation error. With this choice of $\mathbb{M}$, we hope that the mesh elements are concentrated in the regions of object edges where the curvature of u is large.

The mesh $\mathscr{T}_h$ being uniform in metric $\mathbb{M}$ will mean that the volume of K in $\mathbb{M}$ is proportional to the volume of K_c with the same proportional constant for all $K \in \mathscr{T}_h$ and K measured in $\mathbb{M}$ is similar to K_c. These requirements can be expressed mathematically as the equidistribution and alignment conditions (e.g., see [22]),

$$|K|\sqrt{\det(\mathbb{M}_K)} = \frac{\sigma_h|K_c|}{|\Omega_c|}, \qquad \forall K \in \mathscr{T}_h, \tag{3.5a}$$

$$\frac{1}{d}\mathrm{tr}\left((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T}\right) = \det\left((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T}\right)^{\frac{1}{d}}, \qquad \forall K \in \mathscr{T}_h, \tag{3.5b}$$

where $|K|$ and $|K_c|$ denote the volume of K and K_c, respectively, d is the dimension of Ω, $\mathrm{tr}(\cdot)$ denotes the trace of a matrix, and

$$|\Omega_c| = \sum_{K_c \in \mathscr{T}_{c,h}} |K_c|, \qquad \sigma_h = \sum_{K \in \mathscr{T}_h} |K|\sqrt{\det(\mathbb{M}_K)}.$$

An energy functional associated with these conditions has been proposed in [17] as

$$\begin{aligned} I_h(\mathscr{T}_h, \mathscr{T}_{c,h}) = \theta \sum_{K \in \mathscr{T}_h} |K|\sqrt{\det(\mathbb{M}_K)}\left(\mathrm{tr}((F_k')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T})\right)^{\frac{dp}{2}} \\ + (1-2\theta)d^{\frac{dp}{2}} \sum_{K \in \mathscr{T}_h} |K|\sqrt{\det(\mathbb{M}_K)}\left(\frac{|K_c|}{|K|\sqrt{\det(\mathbb{M}_K)}}\right)^p, \end{aligned} \tag{3.6}$$

where $\theta \in (0, 0.5]$ and $p > 1$ are two dimensionless parameters. In our computation, we take $\theta = 1/3$ and $p = 1.5$ which are known from experience to work well for most problems.

Notice that I_h is a function of $\mathscr{T}_h$ and $\mathscr{T}_{c,h}$. We can take $\mathscr{T}_{c,h}$ as the reference computational mesh $\hat{\mathscr{T}}_{c,h}$ and minimize I_h with respect to $\mathscr{T}_h$. With the MMPDE strategy, the minimization is carried out by integrating a modified gradient system of I_h,

$$\frac{\partial x_i}{\partial t} = -\frac{P_i}{\tau}\left(\frac{\partial I_h}{\partial x_i}\right)^T, \quad i = 1, \cdots, N_v, \quad t \in (t_n, t_{n+1}], \tag{3.7}$$

where $\frac{\partial I_h}{\partial x_i}$ is a row vector, $P_i = \det(\mathbb{M}(x_i))^{\frac{p-1}{2}}$ is a positive function chosen to make (3.7) invariant under the scaling transformation of $\mathbb{M}$, and $\tau > 0$ is a positive parameter used to adjust the time scale of mesh movement. Starting from $\mathscr{T}_h^n$, we can integrate (3.7) (with proper modifications for the boundary vertices to allow them to slide on the boundary) from t_n to t_{n+1} to obtain $\mathscr{T}_h^{n+1}$. Special attention may be needed for the computation of the metric tensor that is typically available only at $\mathscr{T}_h^n$ (the mesh at $t = t_n$). During the integration of (3.7), the location of the physical vertices changes, and the values of $\mathbb{M}$ at these vertices should be updated via interpolation of its values on the vertices of $\mathscr{T}_h^n$. It is also worth mentioning that the mesh governed by (3.7) is known [20] to stay nonsingular if it is nonsingular initially.

To avoid the need of constantly updating the metric tensor $\mathbb{M}$ during the integration of the mesh equation, we now consider an indirect approach of minimizing I_h. In this approach, we choose $\mathscr{T}_h = \mathscr{T}_h^n$ and minimize I_h with respect to $\mathscr{T}_{c,h}$. Then the MMPDE for the computational vertices reads as

$$\frac{\partial \xi_i}{\partial t} = -\frac{P_i}{\tau}\left(\frac{\partial I_h}{\partial \xi_i}\right)^T, \quad i = 1, \cdots, N_v, \quad t \in (t_n, t_{n+1}]. \tag{3.8}$$

Starting from $\hat{\mathscr{T}}_{c,h}$, this equation can be integrated from t_n to t_{n+1} to obtain a new computational mesh $\mathscr{T}_{c,h}^{n+1}$. In our computation, we use Matlab® function *ode15s*, a Numerical Differentiation Formula based integrator, for this purpose. Note that $\mathscr{T}_h^n$ and $\mathbb{M} = \mathbb{M}^n$ are fixed during the integration and $\mathscr{T}_h^n$ and $\mathscr{T}_{c,h}^{n+1}$ form a correspondence. Denote the correspondence by Ψ_h, i.e., $\mathscr{T}_h^n = \Psi_h(\mathscr{T}_{c,h}^{n+1})$. Then, the new physical mesh is defined as

$$\mathscr{T}_h^{n+1} = \Psi_h(\hat{\mathscr{T}}_c), \tag{3.9}$$

which can be readily computed using linear interpolation.

A benefit of the above ξ-formulation is that the derivative $\partial I_h/\partial \xi_i$ in (3.8) can be found analytically using the notion of scalar-by-matrix differentiation [19] and has a relatively simple, compact matrix form. Using this, we can rewrite (3.8) into

$$\frac{\partial \xi_i}{\partial t} = \frac{P_i}{\tau}\sum_{K \in \omega_i} |K| v_{i_K}^K, \quad i = 1, \cdots, N_v, \tag{3.10}$$

where ω_i is the patch of the elements containing x_i as a vertex, the index i_K denotes the local index of x_i in K, and $v_{i_K}^K$ is the local velocity contributed by the element K to the partial derivative $\frac{\partial I_h}{\partial x_i}$. The local velocities on element K are given by

$$\begin{bmatrix} (v_1^K)^T \\ \vdots \\ (v_d^K)^T \end{bmatrix} = -E_K^{-1} \frac{\partial G}{\partial \det(\mathbb{J})} - \frac{\partial G}{\partial \det(\mathbb{J})} \frac{\det(\hat{E}_K)}{\det(E_K)} \hat{E}_K^{-1}, \qquad v_0^K = -\sum_{i=1}^{d} v_d^K, \tag{3.11}$$

where $E_K = [x_1^K - x_0^K, \cdots, x_d^K - x_0^K]$ and $\hat{E}_K = [\xi_1^K - \xi_0^K, \cdots, \xi_d^K - \xi_0^K]$ are the edge matrices of K and K_c, respectively, $\mathbb{J} = (F_K)^{-1} = \hat{E}_K {E_K}^{-1}$, $G = G(\mathbb{J}, \det(\mathbb{J}), \mathbb{M}_K)$ is a function associated with the meshing energy functional, and $\partial G/\partial \mathbb{J}$ and $\partial G/\partial \det(\mathbb{J})$ are the partial derivatives of G with respect to the first and second arguments, respectively. For the meshing energy functional (3.6), we have

$$\begin{aligned} G &= \theta \sqrt{\det(\mathbb{M}_K)} (\mathrm{tr}(\mathbb{J}\mathbb{M}_K^{-1}\mathbb{J}^T))^{\frac{dp}{2}} + (1-2\theta) d^{\frac{dp}{2}} \sqrt{\det(\mathbb{M}_K)} \left(\frac{\det(\mathbb{J})}{\sqrt{\det(\mathbb{M}_K)}} \right)^p, \\ \frac{\partial G}{\partial \mathbb{J}} &= dp\theta \sqrt{\det(\mathbb{M}_K)} \big(\mathrm{tr}(\mathbb{J}\mathbb{M}_K^{-1}\mathbb{J}^T)\big)^{\frac{dp}{2}-1} \mathbb{M}_K^{-1}\mathbb{J}^T, \\ \frac{\partial G}{\partial \det(\mathbb{J})} &= p(1-2\theta) d^{\frac{dp}{2}} \det(\mathbb{M}_K)^{\frac{1-p}{2}} \det(\mathbb{J})^{p-1}. \end{aligned}$$

4. Numerical results: behavior of (u, ϕ) as $\epsilon \to 0$

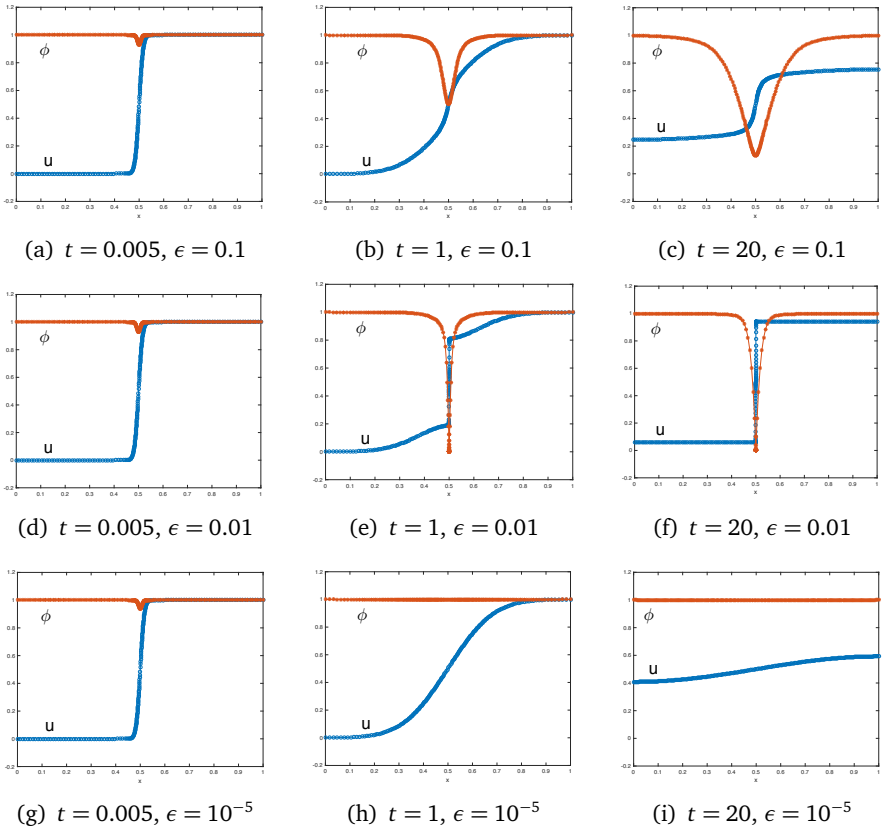
In this section we present numerical results obtained with the moving mesh finite element method described in the previous section to illustrate the analysis in Section 2. We choose two analytical functions for g, with one each in one dimension and two dimensions, to simulate the grey-level values of images. In particular, the sharp jumps in g model the object edges in the image.

Example 4.1 (1D hyperbolic tangent)**.** In this example, we take

$$g = 0.5(1 + \tanh(100(x - 0.5))), \quad x \in (0, 1), \tag{4.1}$$

which has a sharp jump at $x = 0.5$. The initial conditions are taken as $u^0 = g$ and $\phi^0 = 1$. We take $N = 200$, $\alpha = 0.01$, $\beta = 10^{-3}$, $\gamma = 10^{-3}$, and $k_\epsilon = 10^{-9}$. The computed solution at three time instants for $\epsilon = 0.1$, 0.01, and 10^{-5} is shown in Fig. 1. It can be seen that the mesh concentrates around and follows the sharp jumps in the solution. This demonstrates the mesh adaptation ability of the MMPDE moving mesh strategy.

Recall that the jump in the solution simulates object edges in a real image and an ideal segmentation should sharpen this jump while smoothing out the regions divided by the jump. The first row of Fig. 1 shows the evolution of u and ϕ for $\epsilon = 0.1$. One can see that the jump is not sharpened and u is smoothed out on the whole domain as time evolves.



(a) $t = 0.005$, $\epsilon = 0.1$ (b) $t = 1$, $\epsilon = 0.1$ (c) $t = 20$, $\epsilon = 0.1$

(d) $t = 0.005$, $\epsilon = 0.01$ (e) $t = 1$, $\epsilon = 0.01$ (f) $t = 20$, $\epsilon = 0.01$

(g) $t = 0.005$, $\epsilon = 10^{-5}$ (h) $t = 1$, $\epsilon = 10^{-5}$ (i) $t = 20$, $\epsilon = 10^{-5}$

Figure 1: Example 4.1. The computed solution u_h and ϕ_h at three time instants for various values of ϵ. No scaling has been used on g and u.

This indicates that the Ambrosio-Tortorelli functional with $\epsilon = 0.1$ does not provide a good segmentation. The result is shown for a smaller $\epsilon = 0.01$ on the second row of the figure. As time evolves, the jump is getting sharper and u becomes piecewise constant essentially, an indication for good image segmentation. However, when ϵ continues to decrease, as shown on the last row ($\epsilon = 10^{-5}$) of Fig. 1, the jump disappears for the time being, ϕ approaches to 1, and u becomes smooth over the whole domain. This implies that the Ambrosio-Tortorelli functional loses its segmentation ability for very small ϵ, consistent with the analysis in Section 2.

It is interesting to see the transient behavior of ϕ. From the simplified equation (2.8), we have $\phi_t = -\alpha|\nabla u|^2\phi$ initially due to the initial condition $\phi = 1$. Thus, we expect that ϕ decreases initially and this decrease is more significant in the regions where ∇u is larger. This is confirmed in the numerical results; see Fig. 1(a,d,g). As time evolves, the system reaches an equilibrium state and ϕ is approximately given by (2.9). When ϵ is not too small and ∇u is sufficiently large at some places, then ϕ can become close to zero at the places and this yields a good segmentation; see the second row of Fig. 1. However, when ϵ is too small, ϕ will essentially become 1 everywhere and the functional loses its segmentation ability (cf. the third row of Fig. 1).

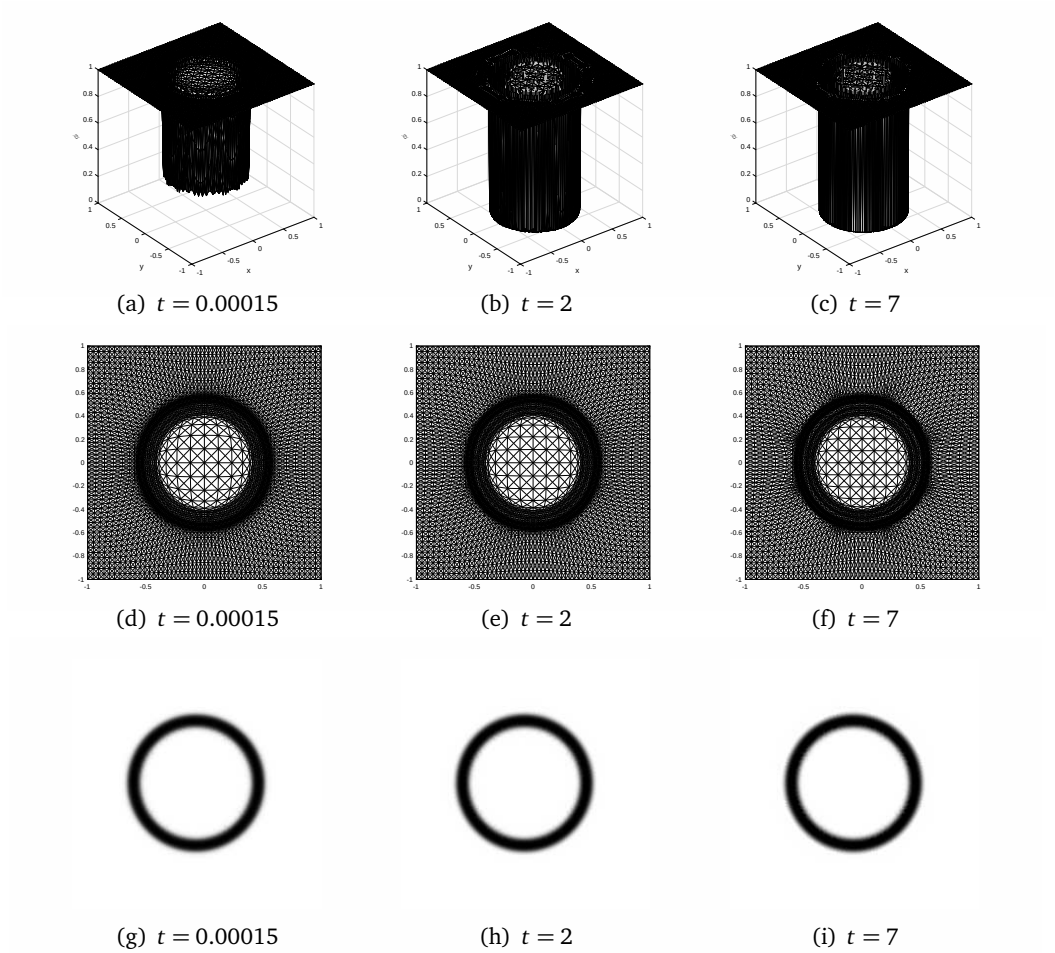
Example 4.2 (2D hyperbolic tangent)**.** In this example, we choose

$$g = 0.49\left[2 + \tanh(50(\sqrt{(x-0.5)^2+(y-0.5)^2} - 0.05)) \right.$$
$$\left. - \tanh(50(\sqrt{(x-0.5)^2+(y-0.5)^2} + 0.05))\right], \quad (x,y) \in (0,1)\times(0,1),$$

which models a circle, being close to 0 on the circle and approximately 1 elsewhere. For the reasons to be explained in Section 5, u and g in the IBVP (2.1) are scaled in this example according to (5.4).

We take

$$u^0 = g, \qquad \phi^0 = 1, \qquad N = 2\times 50\times 50,$$
$$\alpha = 10^{-3}, \quad \gamma = 10^{-5}, \quad \beta = 10^{-2}, \quad k_\epsilon = 10^{-10}.$$

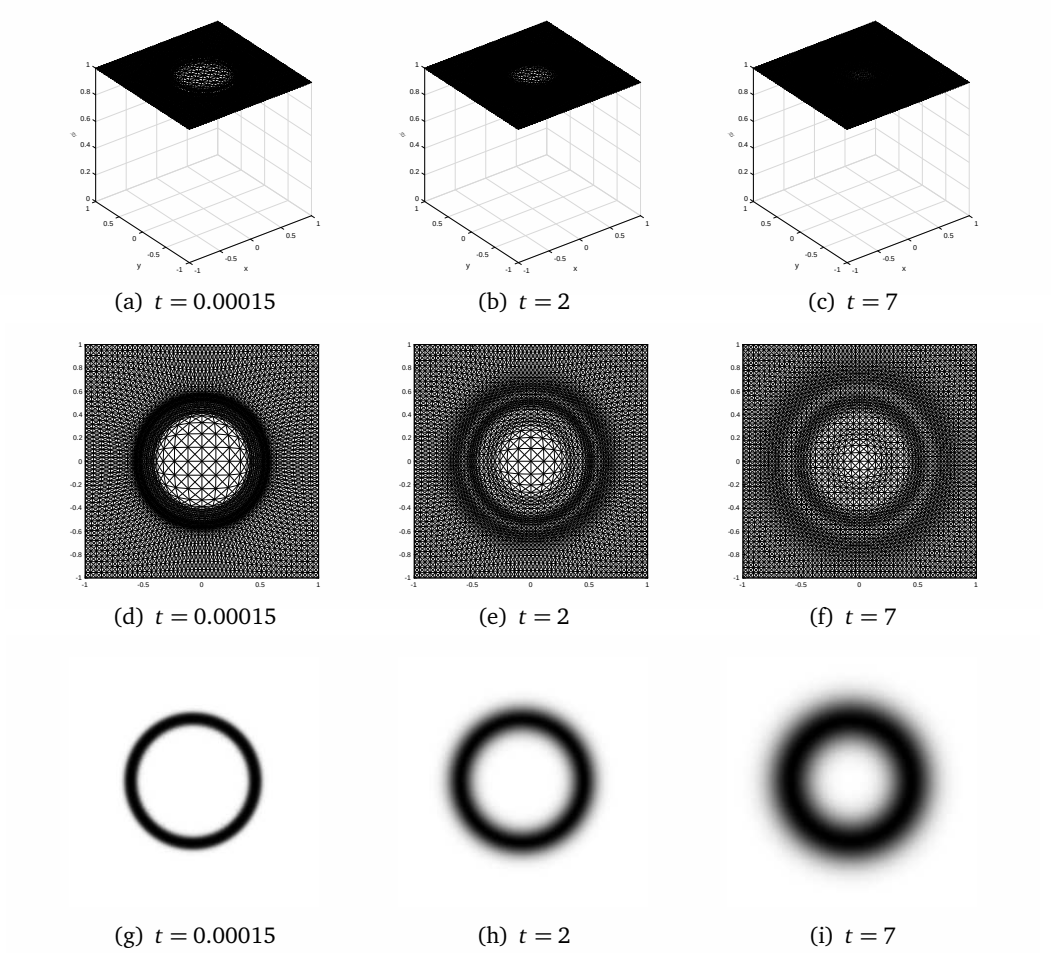


(a) $t = 0.00015$ (b) $t = 2$ (c) $t = 7$

(d) $t = 0.00015$ (e) $t = 2$ (f) $t = 7$

(g) $t = 0.00015$ (h) $t = 2$ (i) $t = 7$

Figure 2: Example 4.2. Evolution of the solution for $\epsilon = 10^{-3}$. The first, second, and third rows show the evolution of ϕ, the moving mesh, and the image of u, respectively.



(a) $t = 0.00015$ (b) $t = 2$ (c) $t = 7$

(d) $t = 0.00015$ (e) $t = 2$ (f) $t = 7$

(g) $t = 0.00015$ (h) $t = 2$ (i) $t = 7$

Figure 3: Example 4.2. Evolution of the solution for $\epsilon = 10^{-7}$. The first, second, and third rows show the evolution of ϕ, the moving mesh, and the image of u, respectively.

The numerical results obtained with $\epsilon = 10^{-3}$ and $\epsilon = 10^{-7}$ are shown in Figs. 2 and 3, respectively. They show that the mesh concentrates around the jump (the circle) very well, which, once again, demonstrates the mesh adaptation ability of the MMPDE moving mesh method.

Fig. 2 shows that the Ambrosio-Tortorelli functional with $\epsilon = 10^{-3}$ makes a good segmentation. The evolution of ϕ is given on the first row, and ϕ deceases rapidly to 0 along the circle at $t = 7$. The image of the circle is clear as shown on the third row. However, the situation changes when a smaller ϵ is used. As shown in Fig. 3 with $\epsilon = 10^{-7}$, the segmentation ability disappears. As t increases, ϕ becomes close to 1 in the whole domain, failing to identify the circle. In the same time, the image of u blurs out. As for Example 4.1, the above observation is consistent with the analysis in Section 2, that is, when g is continuous, the segmentation ability of the Ambrosio-Tortorelli functional varies for small but finite ϵ and disappears as $\epsilon \to 0$.

5. Selection of the regularization parameter and scaling of g and u

5.1. Selection of the regularization parameter

From the analysis in Section 2 and the examples in the previous section, we have seen that it is crucial to choose a proper ϵ for the Ambrosio-Tortorelli functional to produce a good segmentation when g is continuous. To see how to choose ϵ properly, we recall that ϕ is given in (2.7) for small ϵ. We want to have $\phi = 0$ on object edges. Taking $\phi = 0$ in (2.7) we get

$$\epsilon = \frac{\beta}{2\alpha|\nabla u^{(0)}|^2}, \tag{5.1}$$

where $u^{(0)}$ is the solution of (2.6) subject to a homogeneous Neumann boundary condition. Since $u^{(0)}$ is completely determined by its initial value g and an objective of the Ambrosio-Tortorelli functional is to make u (and thus $u^{(0)}$) close to g, it is reasonable to replace $u^{(0)}$ by g in the above formula, i.e.,

$$\epsilon = \frac{\beta}{2\alpha|\nabla g|^2}. \tag{5.2}$$

Since $|\nabla g|$ varies from place to place and ϵ is a constant, in our computation we replace the former with $(|\nabla g|_{\max} + |\nabla g|_{\min})/2$ and have

$$\epsilon = \frac{\beta}{2\alpha\left((|\nabla g|_{\max} + |\nabla g|_{\min})/2\right)^2}. \tag{5.3}$$

Notice that we could have replaced $|\nabla g|$ by $|\nabla g|_{max}$. In this case we have

$$\epsilon = \frac{\beta}{2\alpha|\nabla g|^2_{max}}.$$

Then from (2.7) we have

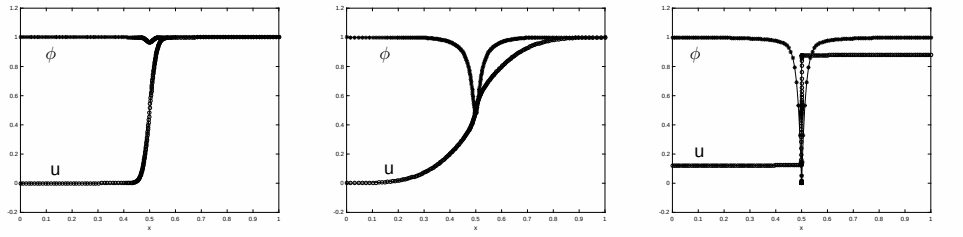
$$\phi = 1 - \frac{|\nabla u^{(0)}|^2}{|\nabla g|^2_{max}} + \mathcal{O}(\epsilon^2).$$

This implies that the boundaries (where $\phi \approx 0$) can be formed at places where $|\nabla u^{(0)}| \approx |\nabla g|_{max}$. But this can hardly happen since $u^{(0)}$ is much smoother than g in general and its gradient will not reach the maximal gradient of g. On the other hand, for the choice (5.3) we have

$$\phi = 1 - \frac{|\nabla u^{(0)}|^2}{\left((|\nabla g|_{\max} + |\nabla g|_{\min})/2\right)^2} + \mathcal{O}(\epsilon^2).$$

Then, the boundaries will be formed at places where $|\nabla u^{(0)}|$ is greater than or close to $(|\nabla g|_{\max} + |\nabla g|_{\min})/2$, which is more likely to happen.

To demonstrate this choice of ϵ, we apply it to Example 4.1 and obtain $\epsilon = 0.008$. The numerical result obtained with the same initial condition and parameters (other than ϵ) is shown in Fig. 4. One can see that this value of ϵ leads to a good segmentation of the image.



(a) $t = 0.005$ (b) $t = 1$ (c) $t = 20$

Figure 4: Example 4.1. The evolution of u and ϕ for $\epsilon = 0.008$ (determined by (5.3)). No scaling has been used on u and g.

It is noted that the choice for ϵ given in (5.3) depends on the gradient of the image datum g that can be sensitive to noise present in the image. Although the choice seems to work for the tested images with noise (see Section 5.3), it may be better to use a regularized version of g in practice. For example, a five-step Gaussian filter with radius 1 has been applied to g in (5.3). A comparison with and without regularization is demonstrated in Fig. 16. The results are very comparable for this example.

5.2. Scaling of g and u

Our experience shows that (5.3) works well when the difference in ∇g between the objects and their edges is sufficiently large. However, when the change of ∇g is small, the Ambrosio-Tortorelli functional can still fail to produce a segmentation of good quality. To avoid this difficulty, we propose to scale u and g in (2.1), i.e., $u \to Lu$ and $g \to Lg$ for some parameter $L \geq 1$. This will make the change of ∇g from place to place more significant. Moreover, the first equation of (2.1) will stay invariant. The second equation becomes

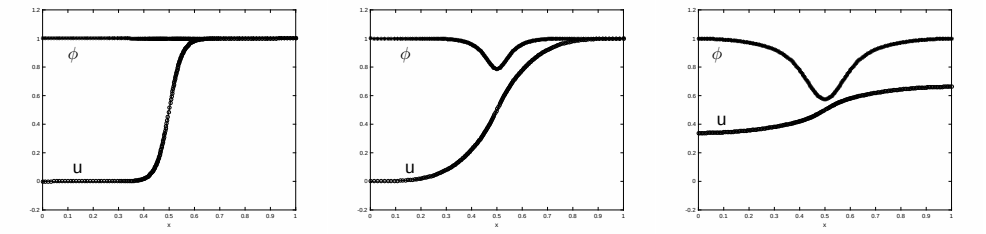
$$\phi_t = 2\beta\epsilon\Delta\phi - L^2\alpha|\nabla u|^2\phi + \frac{\beta}{2\epsilon}(1-\phi),$$

where the second term on the right-hand side is made larger, helping decrease ϕ. We choose

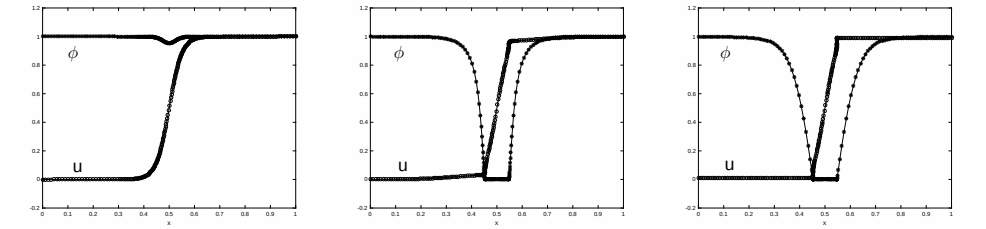
$$L = \max\left\{1, \frac{|\nabla g|_{\text{cr}}}{|\nabla g|_{\max}}\right\}, \tag{5.4}$$

where $|\nabla g|_{\text{cr}}$ is a parameter. Generally speaking, the larger $|\nabla g|_{\text{cr}}$ (and L) is, the more likely the segmentation works, but this will also make (2.1) harder to integrate. We take $|\nabla g|_{\text{cr}} = 3 \times 10^3$ (by trial and error) in our computation, unless stated otherwise.

To demonstrate the effects of the scaling, we recompute Example 4.1 with $u^0 = g = 0.5(1+\tanh(20(x-0.5)))$, which has a less steep jump at $x = 0.5$ than the function (4.1). Results with and without scaling are shown in Fig. 5. It can be seen that scaling improves the segmentation ability of the Ambrosio-Tortorelli functional.

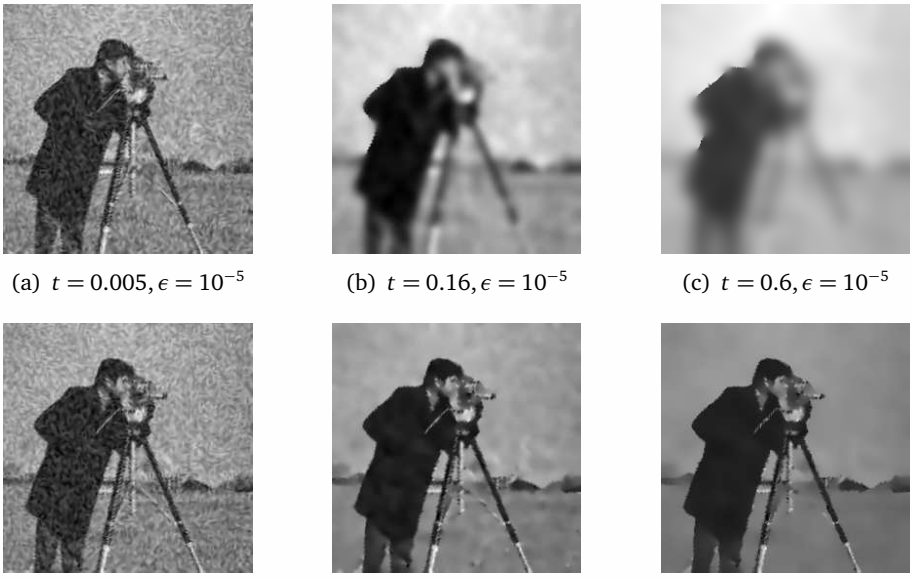


(a) $t = 0.005$ (b) $t = 1$ (c) $t = 20$



(d) $t = 0.005$ (e) $t = 1$ (f) $t = 20$

Figure 5: Example 4.1 with $g = 0.5(1 + \tanh(20(x - 0.5)))$ and $u^0 = g$. $\epsilon = 0.05$, which is chosen as in (5.3) and other parameters are the same as in Example 4.1. No scaling is used for the top row while the scaling with (5.4) for u and g is used for the bottom row.



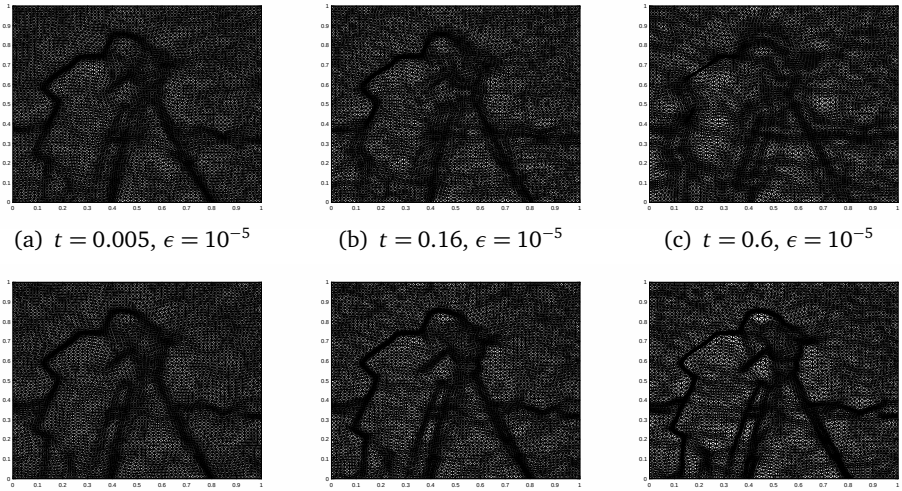
(a) $t = 0.005, \epsilon = 10^{-5}$ (b) $t = 0.16, \epsilon = 10^{-5}$ (c) $t = 0.6, \epsilon = 10^{-5}$

(d) $t = 0.005, \epsilon = 0.00044$ (e) $t = 0.16, \epsilon = 0.00044$ (f) $t = 0.6, \epsilon = 0.00044$

Figure 6: A comparison of the image segmentation with different ϵ values. The value of ϵ in (d,e,f) is obtained using (5.3).

5.3. Segmentation for real images

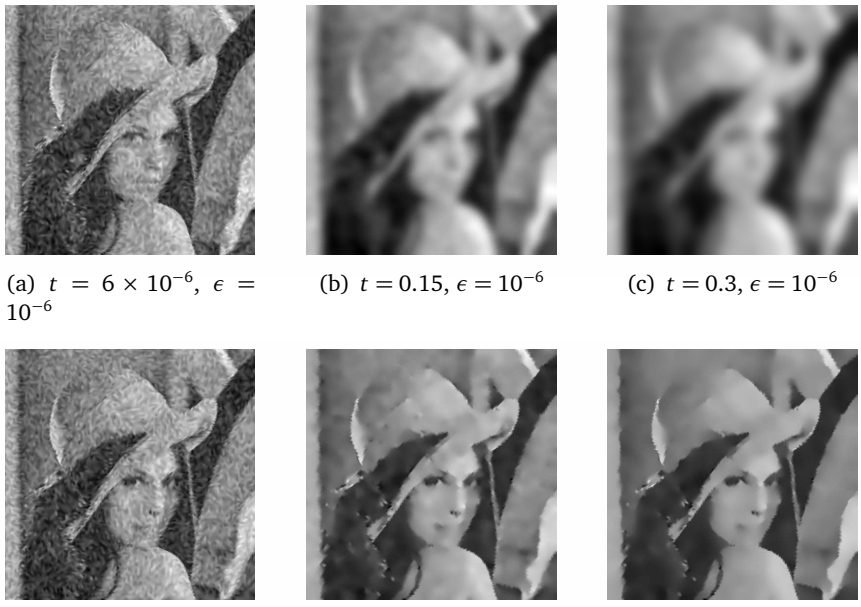
To further demonstrate the effects of the selection strategy (5.3) and the scaling (5.4) we present results obtained for four real images. The results are shown in Figs. 6, 8, 10,



(a) $t = 0.005$, $\epsilon = 10^{-5}$ (b) $t = 0.16$, $\epsilon = 10^{-5}$ (c) $t = 0.6$, $\epsilon = 10^{-5}$

(d) $t = 0.005$, $\epsilon = 0.00044$ (e) $t = 0.16$,$\epsilon = 0.00044$ (f) $t = 0.6$, $\epsilon = 0.00044$

Figure 7: The meshes corresponding to Fig. 6. The value of ϵ in (d,e,f) is obtained using (5.3).

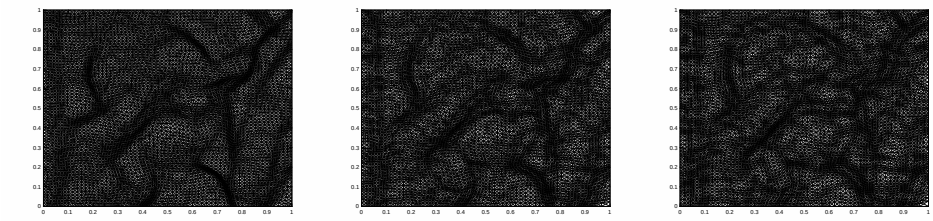


(a) $t = 6 \times 10^{-6}$, $\epsilon = 10^{-6}$ (b) $t = 0.15$, $\epsilon = 10^{-6}$ (c) $t = 0.3$, $\epsilon = 10^{-6}$

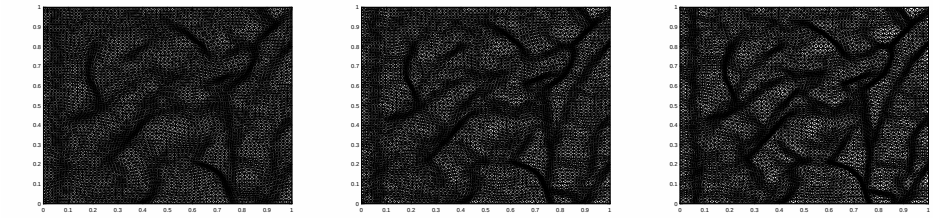
(d) $t = 6 \times 10^{-6}$, $\epsilon = 0.0004$ (e) $t = 0.15$, $\epsilon = 0.0004$ (f) $t = 0.3$, $\epsilon = 0.0004$

Figure 8: Evolution of the image. The value of ϵ in (d,e,f) is obtained using (5.3).

and 12 and the corresponding meshes are shown in Figs. 7, 9, 11, and 13, respectively. The original resolution (the number of pixels) for these four images are 256×256, 512×512, 756×800, and 480×640, respetively. In these four experiments, $N = 2 \times 70 \times 70$, $\alpha = 10^{-3}$, $\gamma = 10^{-5}$, $\beta = 10^{-2}$, and $k_\epsilon = 10^{-10}$ are used. The parameters are chosen (with



(a) $t = 6\times10^{-6}$, $\epsilon = 10^{-6}$ (b) $t = 0.15$, $\epsilon = 10^{-6}$ (c) $t = 0.3$, $\epsilon = 10^{-6}$



(d) $t = 6\times10^{-6}$, $\epsilon = 0.0004$ (e) $t = 0.15$, $\epsilon = 0.0004$ (f) $t = 0.3$, $\epsilon = 0.0004$

Figure 9: The meshes corresponding to Fig. 8. The value of ϵ in (d,e,f) is obtained using (5.3).

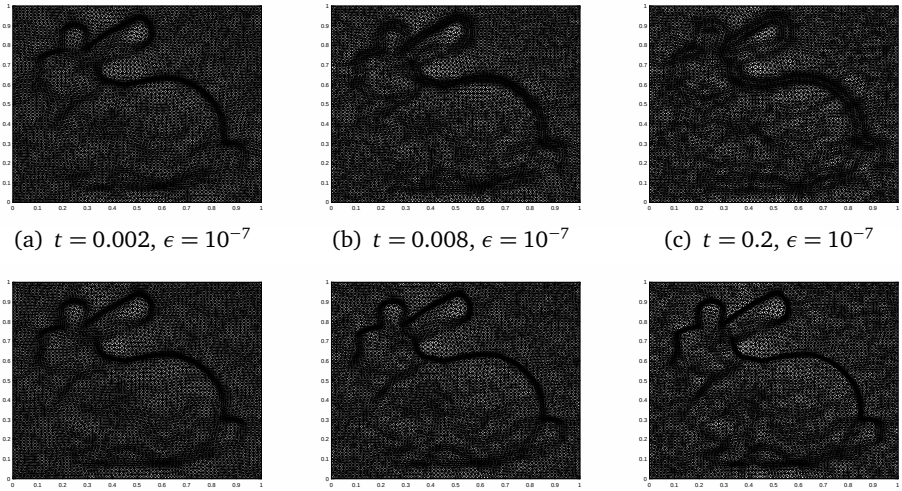
(a) $t = 0.002$, $\epsilon = 10^{-7}$ (b) $t = 0.008$, $\epsilon = 10^{-7}$ (c) $t = 0.2$, $\epsilon = 10^{-7}$

(d) $t = 0.002$, $\epsilon = 0.00016$ (e) $t = 0.008$, $\epsilon = 0.00016$ (f) $t = 0.2$, $\epsilon = 0.00016$

Figure 10: Evolution of the image. The value of ϵ in (d,e,f) is obtained using (5.3).

some adjustment) according to [13]. A random field in the range $(-0.25, 0.25)$ is added to g as well as u^0. One can observe that the selection strategy (5.3) for the regularization parameter significantly improves segmentation for all cases.

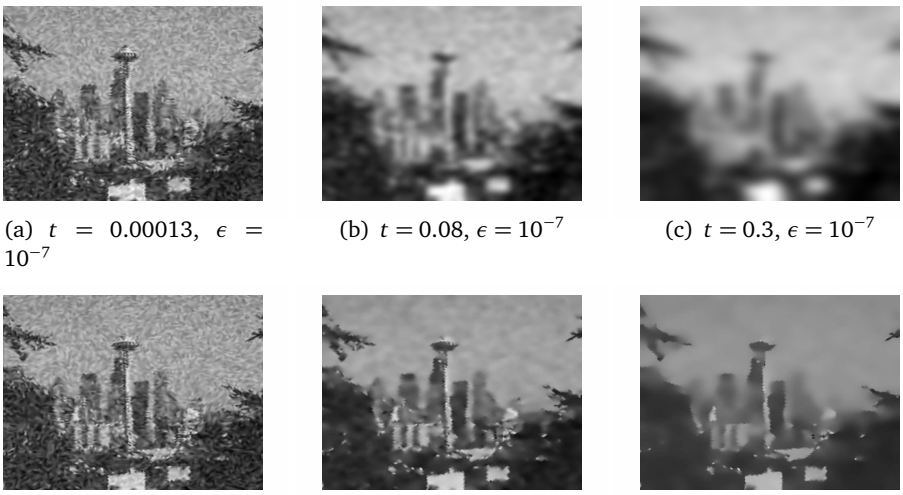
We now consider the efficiency of the adaptive moving mesh method. This issue has been studied extensively for test examples with exact solutions (e.g., see [22]) where the



(a) $t = 0.002$, $\epsilon = 10^{-7}$ (b) $t = 0.008$, $\epsilon = 10^{-7}$ (c) $t = 0.2$, $\epsilon = 10^{-7}$

(d) $t = 0.002$, $\epsilon = 0.00016$ (e) $t = 0.008$, $\epsilon = 0.00016$ (f) $t = 0.2$, $\epsilon = 0.00016$

Figure 11: The meshes corresponding to Fig. 10. The value of ϵ in (d,e,f) is obtained using (5.3).

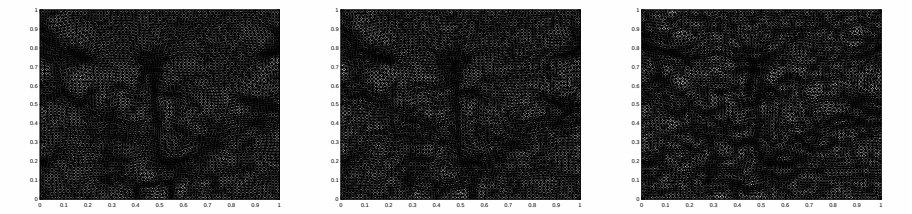


(a) $t = 0.00013$, $\epsilon = 10^{-7}$ (b) $t = 0.08$, $\epsilon = 10^{-7}$ (c) $t = 0.3$, $\epsilon = 10^{-7}$

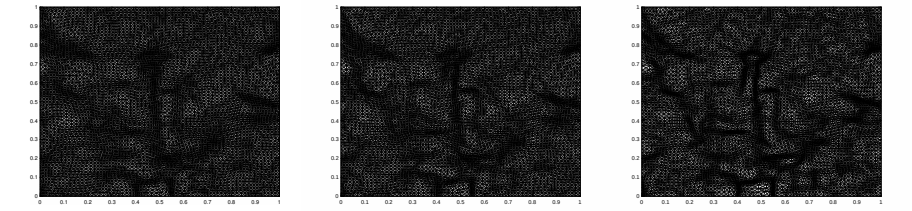
(d) $t = 0.00013$, $\epsilon = 0.00025$ (e) $t = 0.08$, $\epsilon = 0.00025$ (f) $t = 0.3$, $\epsilon = 0.00025$

Figure 12: Evolution of the image. The value of ϵ in (d,e,f) is obtained using (5.3).

error is plotted as a function of CPU time. In the current situation, the exact solution is unavailable and the error cannot be computed. Nevertheless, we can have a rough idea by comparing the segmentation quality. Two sets of results are shown in Fig. 14, one obtained with a fixed mesh and the other obtained with an adaptive mesh. The same parameters, $N = 2 \times 70 \times 70$, $\alpha = 10^{-3}$, $\gamma = 10^{-5}$, $\beta = 10^{-2}$, $k_\epsilon = 10^{-10}$ and $\epsilon = 0.00044$ are used in both. One can see that the moving mesh method leads to much better segmentation quality. The CPU time (on a linux machine with a single AMD Opteron 6386 SE 2.8 GHz



(a) $t = 0.00013$, $\epsilon = 10^{-7}$ (b) $t = 0.08$, $\epsilon = 10^{-7}$ (c) $t = 0.3$, $\epsilon = 10^{-7}$



(d) $t = 0.00013$, $\epsilon = 0.00025$ (e) $t = 0.08$, $\epsilon = 0.00025$ (f) $t = 0.3$, $\epsilon = 0.00025$

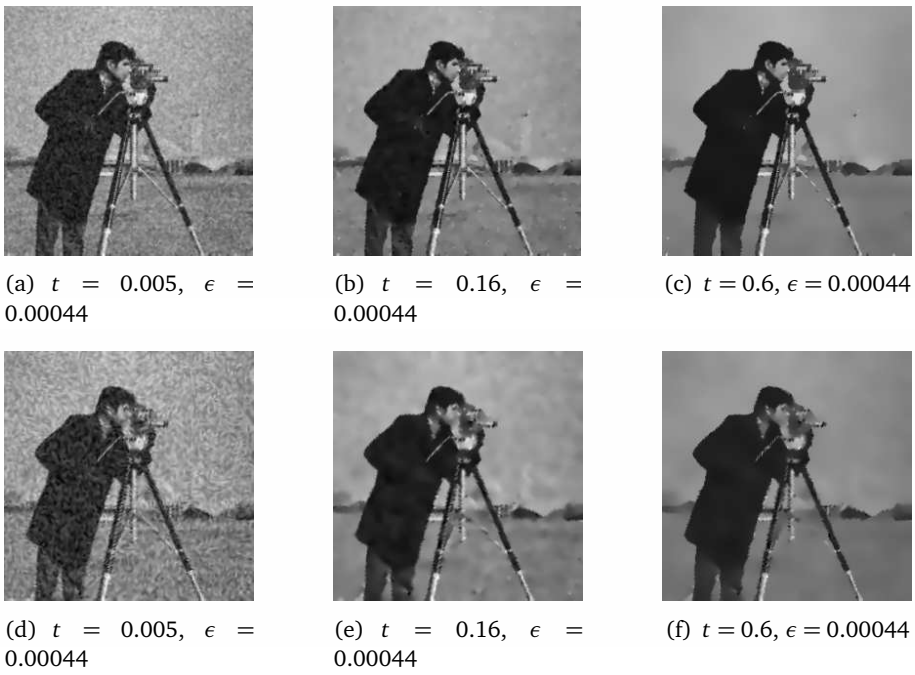
Figure 13: The meshes corresponding to Fig. 12. The value of ϵ in (d,e,f) is obtained using (5.3).

(a) $t = 0.005$, $\epsilon = 0.00044$ (b) $t = 0.16$, $\epsilon = 0.00044$ (c) $t = 0.6$, $\epsilon = 0.00044$

(d) $t = 0.005$, $\epsilon = 0.00044$ (e) $t = 0.16$, $\epsilon = 0.00044$ (f) $t = 0.6$, $\epsilon = 0.00044$

Figure 14: Comparison of image segmentation with fixed and adaptive meshes. (a,b,c): fixed mesh of $N = 2 \times 70 \times 70$; (d,e,f): adaptive mesh of $N = 2 \times 70 \times 70$.

processor, coded in Matlab) is 12,849 seconds and 12,498 seconds for the moving and fixed mesh cases, respectively. Most of CPU time has been spent on the computation of



(a) $t = 0.005$, $\epsilon = 0.00044$ (b) $t = 0.16$, $\epsilon = 0.00044$ (c) $t = 0.6$, $\epsilon = 0.00044$

(d) $t = 0.005$, $\epsilon = 0.00044$ (e) $t = 0.16$, $\epsilon = 0.00044$ (f) $t = 0.6$, $\epsilon = 0.00044$

Figure 15: Comparison of image segmentation with fixed and adaptive meshes. (a,b,c): fixed mesh of $N = 2 \times 150 \times 150$; (d,e,f): adaptive mesh of $N = 2 \times 70 \times 70$.

the gradient flow equation (2.1). The segmentation quality improves as the fixed mesh is refined. Fig. 15 shows that a finer fixed mesh of $N = 2 \times 150 \times 150$ gives almost the same level of segmentation quality as an adaptive mesh of $N = 2 \times 70 \times 70$. But the former takes 69,263 seconds, about six times of that used for the latter.

6. Conclusions

The Mumford-Shah functional has been widely used for image segmentation. Its Ambrosio-Tortorelli Approximation has been known for its relative ease in implementation, segmentation ability, and Γ-convergence to the Mumford-Shah functional as the regularization parameter ϵ goes to zero. The segmentation ability is based on the assumption that the input image g is discontinuous across the boundaries between different objects, and this discontinuity must be maintained in the limit of $\epsilon \to 0$ during numerical computation to retain the Γ-convergence and the segmentation ability for infinitesimal ϵ (e.g., see [4]). However, the maintenance of discontinuity in g is often forgotten and g is treated implicitly as a continuous function in actual computation. As a consequence, it has been observed that the segmentation ability of the Ambrosio-Tortorelli functional varies significantly with different values of ϵ and the functional can even fail to Γ-converge to the original functional for some cases. Moreover, there exist very few published numerical studies on the behavior of the functional as $\epsilon \to 0$.

(a) $t = 0.005$, $\epsilon = 0.00044$

(b) $t = 0.16$, $\epsilon = 0.00044$

(c) $t = 0.6$, $\epsilon = 0.00044$

(d) $t = 0.005$, $\epsilon = 0.00048$

(e) $t = 0.16$, $\epsilon = 0.00048$

(f) $t = 0.16$, $\epsilon = 0.00048$

Figure 16: A comparison of the image segmentation with ϵ chosen by g with and without regularization. (a,b,c): without regularization; (d,e,f): a five-step Gaussian filter has been applied to g in (5.3).

We have presented in Section 2 an asymptotic analysis on the gradient flow equation of the Ambrosio-Tortorelli functional as $\epsilon \to 0$ for continuous g. The analysis shows that the functional can have different segmentation behavior for small but finite ϵ and eventually loses its segmentation ability for infinitesimal ϵ. This is consistent with the existing observations in the literature and the numerical examples in one and two spatial dimensions presented in Section 4. Based on the analysis, we have proposed a selection strategy for ϵ and a scaling procedure for u and g in Section 5. Numerical results with real images show that they lead to a good segmentation of the Ambrosio-Tortorelli functional.

Finally, we recall that the Ambrosio-Tortorelli functional is a special example of phase-field modeling for image segmentation. We hope that the analysis and the selection strategy for the regularization parameter presented in this work can also apply to other phase-field models. We are specially interested in the phase-field modeling of brittle fracture (e.g., see [6, 14, 27]). Investigations in this direction are currently underway.

Acknowledgments The authors would like to thank the anonymous referees for their valuable comments.

References

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

[2] M. J. Baines, Moving Finite Elements, Oxford University Press, Oxford, 1994.
[3] M. J. Baines, M. E. Hubbard and P. K. Jimack, *Velocity-based moving mesh methods for nonlinear partial differential equations*, Commun. Comput. Phys., 10 (2011), pp. 509–576.
[4] G. Bellettini and A. Coscia, *Discrete approximation of a free discontinuity problem*, Numer. Func. Anal. Opt., 15 (1994), pp. 201–224.
[5] B. Bourdin, *Image segmentation with a finite element method*, Esaim-Math. Model. Num., 33 (1999), pp. 220–244.
[6] B. Bourdin, G. A. Francfort and J. J. Marigo, *Numerical experiments in revisited brittle fracture*, J. Mech. Phys. Solids, 48 (2000), pp. 797–826.
[7] C. J. Budd, W. Huang and R. D. Russell, *Adaptivity with moving grids*, Acta Numerica, 18 (2009), pp. 111–241.
[8] E. De Giorgi, M. Carriero and A. Leaci, *Existence theorem for a minimum problem with free discontinuity set*, Arch. Ration. Mech. An., 108 (1989), pp. 195–218.
[9] E. De Giorgi and T. Franzoni, *Su un tipo di convergenza variazionale*, Atti Accad. Naz. Lincei Rend. Cl. Sci. Fis. Mat. Natur., 58 (1975), pp. 842–850.
[10] Q. Du and J. Zhang, *Adaptive finite element method for a phase field bending elasticity model of vesicle membrane deformations*, SIAM J. Sci. Comput., 30 (2008), pp. 1634–1657.
[11] T. T. Nguyen, J. Yvonnet, M. Bornert, C. Chateau, K. Sab, R. Romani and R. Le Roy, *On the choice of parameters in the phase field method for simulating crack initiation with experimental validation*, Int. J. Fract, 197 (2016), pp. 213–226.
[12] L. C. Evans, Partial Differential Equations, American Mathematical Society, Providence, Rhode Island, Graduate Studies in Mathematics, 19 (1998).
[13] X. Feng and A. Prohl, *Analysis of gradient flow of a regularized Mumford-Shah functional for image segmentation and image inpainting*, M2AN Math. Model. Numer. Anal., 38 (2004), pp. 291–320.
[14] G. A. Francfort and J. J. Marigo, *Revisiting brittle fracture as an energy minimization problem*, J. Mech. Phys. Solids, 46 (1998), pp. 1319–1342.
[15] S. González-Pinto, J. I. Montijano and S. Pérez-Rodríguez, *Two-step error estimators for implicit Runge-Kutta methods applied to stiff systems*, ACM Trans. Math. Software, 30 (2004), pp. 1–18.
[16] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II, Stiff and differential-algebraic problems, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2nd Ed., 14(1996).
[17] W. Huang, *Variational mesh adaptation: isotropy and equidistribution*, J. Comput. Phys., 174 (2001), pp. 903–924.
[18] W. Huang, *Metric tensors for anisotropic mesh generation*, J. Comput. Phys., 2005, pp. 633–665.
[19] W. Huang and L. Kamenski, *A geometric discretization and a simple implementation for variational mesh generation and adaptation*, J. Comput. Phys., 301 (2015), pp. 322–337.
[20] W. Huang and L. Kamenski, *On the mesh nonsingularity of the moving mesh PDE method*, Math. Comp., to appear, (DOI: 10.1090/mcom/3271).
[21] W. Huang, Y. Ren and R. D. Russell, *Moving mesh partial differential equations (MMPDEs) based upon the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730.
[22] W. Huang and R. D. Russell, Adaptive Moving Mesh Methods, Springer, Applied Mathematical Sciences Series, 174 (2011).
[23] R. Kobayashi, *Modeling and numerical simulations of dendritic crystal growth*, Physica D, 63 (1993), pp. 410–423.
[24] C. Liu and J. Shen, *A phase field model for the mixture of two incompressible fluids and its*

approximation by a Fourier-spectral method, Phys. D, 179 (2003), pp. 211–228.
[25] J. A. Mackenzie and M. L. Robertson, *A moving mesh method for the solution of the one-dimensional phase-field equations*, J. Comput. Phys., 181 (2002), pp. 526–544.
[26] S. May, J. Vignollet and R. de Borst, *A numerical assessment of phase-field models for brittle and cohesive fracture: Γ-covergence and stress oscillations*, European J. Mech. A/Solids., 52 (2015), pp. 72–84.
[27] C. Miehe, F. Welschinger and M. Hofacker, *Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations*, Int. J. Numer. Meth. Eng., 83 (2010), pp. 1273–1311.
[28] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and associated variational problems*, Commun. Pur. Appl. Math., 42 (1989), pp. 577–685.
[29] K. Pham, H. Amor, J. J. Marigo and C. Maurini, *Gradient damage models and their use to approximate brittle fracture*, Int. J. Damage Mech., 20 (2011), pp. 618–652.
[30] J. Shen and X. Yang, *An efficient moving mesh spectral method for the phase-field model of two-phase flows*, J. Comput. Phys., 228 (2009), pp. 2978–2992.
[31] J. Shen and X. Yang, *Decoupled energy stable schemes for phase-field models of two-phase complex fluids*, SIAM J. Sci. Comput., 36 (2014), pp. B122–B145.
[32] J. Shen, X. Yang and H. Yu, *Efficient energy stable numerical schemes for a phase field moving contact line model*, J. Comput. Phys., 284 (2015), pp. 617–630.
[33] T. Tang, *Moving mesh methods for computational fluid dynamics flow and transport*, Contem Math, 383(8) (2004), pp. 141–173.
[34] J. Vignollet, S. May, R. de Borst and C. V. Verhoosel, *Phase-field models for brittle and cohesive fracture*, Meccanica, 49 (2014), pp. 2587–2601.
[35] H. Wang, R. Li and T. Tang, *Efficient computation of dendritic growth with r-adaptive finite element methods*, J. Comput. Phys., 227 (2008), pp. 5984–6000.
[36] A. A. Wheeler, B. T. Murray and R. J. Schaefer, *Computation of dendrites using a phase field model*, Physica D, 66 (1993), pp. 243–262.
[37] X. Yang, J. J. Feng, C. Liu and J. Shen, *Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method*, J. Comput. Phys., 218 (2006), pp. 417–428.
[38] X. Yang, J. J. Feng, C. Liu and J. Shen, *Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method*, J. Comput. Phys., 218 (2006), pp. 417–428.
[39] P. Yu, L. Q. Chen and Q. Du, *Applications of moving mesh methods to the Fourier spectral approximations of phase-field equations*, Recent Advances in Computational Sciences, World Sci. Publ., Hackensack, NJ, 2008, pp. 80–99.
[40] F. Zhang, W. Huang, X. Li and S. Zhang, *Moving mesh finite element simulation of brittle fracture and convergence of Newton's iteration*, J. Comput. Phys., 356 (2018), pp.127-149.