An efficient curve evolution algorithm for multiphase image segmentation

Günay Doğan

Theiss Research,
National Institute of Standards and Technology,
100 Bureau Dr., Stop 8910,
Gaithersburg, MD 20899-8910, USA
gunay.dogan@nist.gov

Abstract. We propose a novel iterative algorithm for multiphase image segmentation by curve evolution. Specifically, we address a multiphase version of the Chan-Vese piecewise constant segmentation energy. Our algorithm is efficient: it is based on an explicit Lagrangian representation of the curves and it converges in a relatively small number of iterations. We devise a stable curvature-free semi-implicit velocity computation scheme. This enables us to take large steps to achieve sharp decreases in the multiphase segmentation energy when possible. The velocity and curve computations are linear with respect to the number of nodes on the curves, thanks to a finite element discretization of the curve and the gradient descent equations, yielding essentially tridiagonal linear systems. The step size at each iteration is selected using a non-monotone line search algorithm ensuring rapid progress and convergence. Thus, the user does not need to specify fixed step sizes or iteration numbers. We also introduce a novel dynamic stopping criterion, robust to various imaging conditions, to decide when to stop the iterations. Our implementation can handle topological changes of curves, such as merging and splitting as well. This is a distinct advantage of our approach, because we do not need to know the number of phases in advance. The curves can merge and split during the evolution to detect the correct regions, especially the number of phases.

1 Introduction

The goal of this work is to devise an efficient algorithm for segmentation of approximately piecewise constant images. We are given a possibly noisy and degraded image I with domain composed of distinct regions $\{\Omega_l\}_{l=0}^{n_\Omega}$ each with homogeneous image intensity, i.e., $I|_{\Omega_l} \approx c_l$ (see Figure 1), and we would like to extract the boundaries of all the regions in the image and the average values $\{c_l\}$ of image intensity for all regions. We express this problem as an energy minimization problem, in which a curve $\Gamma = \bigcup_{l=1}^{n_\Omega} \Gamma_l$, which is the union of a set of simple closed curves, and a set of region averages $\{c_l\}_{l=0}^{n_\Omega}$, also the number of regions n_Ω are the unknowns. This can be considered as a more general version of two-phase segmentation problem solved by Chan and Vese in [5]. Given an

image $I : D \subset \mathbb{R}^2 \to \mathbb{R}$ defined on a bounded image domain D, we seek to minimize the following energy

$$J(\Gamma, \{c_l\}) = \frac{1}{2} \sum_{l=0}^{n_\Omega} \int_{\Omega_l} \chi_D(x)(I(x) - c_l)^2 dx + \mu \int_\Gamma d\Gamma, \quad \mu > 0 \tag{1}$$

where χ_D is the indicator function for the image domain D; it is included to account for the situations when the curves (and the enclosed regions) in the geometric model extend beyond D, but the image data is available only on D. The first term in the energy (1) is a data fidelity term so that the optimal curves match the boundaries of the homogeneous regions in the given image. The second term is a length penalty and favors shorter and smoother curves, so that the optimal curves do not fit insignificant variations or noise in the image.

Many different approaches have been proposed to address the problem of multiphase image segmentation. Recent notable approaches are based on graph formulations [1, 3, 16, 23], convex relaxations [2, 4, 17, 21], variational formulations [15, 22] and level sets [5, 7, 11, 27, 26]. Our algorithm is closer to the level set approach, but it is Lagrangian and offers several advantages that we explain below.

To develop our segmentation algorithm, we study the multiphase energy (1), derive its shape derivative (or first variation) and propose a semi-implicit gradient descent algorithm (in Section 2) that enables large steps (hence fewer iterations) in the minimization. Then we propose a numerical realization (in Section 3) using explicit (but nonparametric) Lagrangian curves to represent the region boundaries, and the finite element method to compute the gradient descent velocity in linear time with respect to the number of nodes on the curves, thereby to perform the curve updates very efficiently at each iteration of the minimization. We also introduce numerical procedures to ensure robustness and reliability in execution, and address issues of practical importance, such as automation of step size and stopping criterion, ensuring adequate distribution of nodes in curve representation, topological changes, i.e., merging and splitting of curves during the curve evolution.

We emphasize the following main contributions of our work:

- Formulation and implementation of the multiphase segmentation problem with only *a set of explicit polygonal curves*, in a way that does not require several level set functions, label grids or indicator functions to represent multiple regions. Moreover we do not need to know the number of regions or phases a priori, this number can change during the minimization.
- A *curvature-free semi-implicit gradient descent scheme* that is unconditionally stable with respect to step size, so that we can take *large steps* at each iteration and converge to the minimum of the energy (1) faster. Curvature is a second order differential geometric quantity and is difficult to handle in both parametric and level set approaches; therefore, not having to compute the curvature to obtain the gradient descent velocity is a critical ingredient in the efficiency and stability of our algorithm.

– A *linear time algorithm for velocity computation* through finite element discretization. This way we obtain sparse matrices that we need to invert at each iteration to compute the gradient descent velocity. The sparse matrices consist of circulant tridiagonal blocks on the diagonal and are inverted in $O(n)$ time where n is the number of curve nodes.

The latter two features of our approach are central to its efficiency. The memory footprint of the curves and the associated data structures, such as the vectors and matrices, is proportional to the number of nodes, and the time complexity consists of pixel summations and other curve procedures, with cost linearly proportional to the number of nodes.

Additionally, our algorithm requires minimal input from the user for execution. It is fully automatic; the user only sets the smoothing parameter μ in (1) and specifies the initial curve(s) and our algorithm performs the minimization without requiring the step size, the number of iterations or other stopping parameters from the user. We use line search to choose the right step size and follow the norm of the shape gradient with a novel dynamic tolerance formula to determine convergence (see Section 2).

Our algorithm currently has two limitations:

– It does not handle junctions. We assume that the boundaries of the homogeneous regions are simple curves that do not intersect with each other. This is not true for some images. Handling the cases with junctions requires some changes to our model and we will address this in future work.
– It does not guarantee convergence to a global minimum, and it can converge to a local minimum. But this can be alleviated with a good initialization scheme, such as topology optimization [12].

2 Gradient Descent Algorithm

The geometry of the problem is specified by a set of disjoint domains $\{\Omega_l\}_{l=0}^{n_\Omega}$ that are used to cover the image domain D, namely, we have $D \subset \bigcup_{l=0}^{n_\Omega} \Omega_l$ (note that a domain Ω_l may extend beyond D, as illustrated in Figure 1). The boundaries $\{\partial\Omega_l\}_{l=0}^{n_\Omega}$ of the domains make up the curve Γ, which is the free variable in this problem. The curve Γ is the union of a set of simple (non-intersecting) closed curves $\{\Gamma_l\}_{l=1}^{n_\Omega}$. The numbering or indexing of the domains and curves is such that a simple curve Γ_l gives the outer boundary of domain Ω_l. We distinguish between two cases:

– If Ω_l has no interior boundary due to a hole (i.e. another domain inside Ω_l), then $\partial\Omega_l = \Gamma_l$, namely, the boundary $\partial\Omega_l$ of Ω_l is equal to Γ_l.
– If Ω_l encloses some other domains $\{\Omega_k\}$ inside, then the boundary $\partial\Omega_l$ of Ω_l includes the outer boundaries $\{\Gamma_k\}$ of $\{\Omega_k\}$, namely $\partial\Omega_l = \Gamma_l \cup \bigcup_k \Gamma_k$.

See Figure 1 for an illustration of the domains, curves and their numbering. Note that the previous work on variational multiphase segmentation required multiple

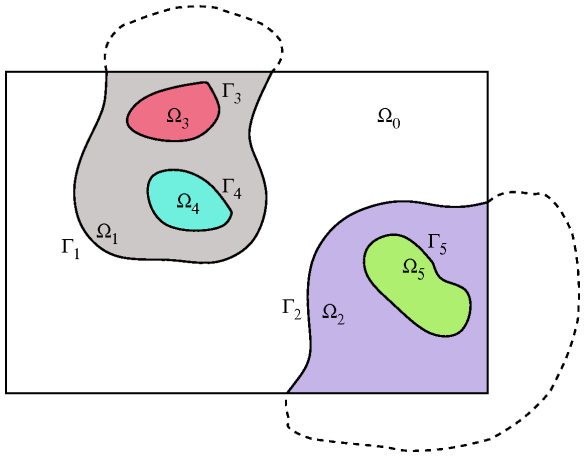

Fig. 1. Illustration of domains, boundaries and their numbering. The image domain D is covered by the regions $\{\Omega_l\}_{l=0}^{n_\Omega}$, specified by the curves $\Gamma = \bigcup_{l=0}^{n_\Omega} \Gamma_l$

level set functions, labeled grids or indicator functions in 2d to represent multi-phase partitioning of the image domain. This is in contrast with our approach, which is built on a set of 1d curves. Moreover, although the number of distinct regions n_Ω is a constant in our formulation (1), in the implementation it need not be known a priori, and it changes during the minimization process as the curves merge and split.

Before we continue to develop the gradient descent algorithm for the energy (1), we review some definitions and concepts from differential geometry. We denote the outer unit normal, the scalar curvature and the curvature vector of a curve $\Gamma \in C^2$ by $\boldsymbol{n}$, κ, $\boldsymbol{\kappa}(:= \kappa \boldsymbol{n})$ respectively. For a given function $f \in C^2(D)$, we define tangential gradient $\nabla_\Gamma f$ and tangential Laplacian $\Delta_\Gamma f$:

$$\nabla_\Gamma f = \left(\nabla f - \frac{\partial f}{\partial \boldsymbol{n}} \boldsymbol{n} \right) \Big|_\Gamma , \qquad \Delta_\Gamma f = \left(\Delta f - \boldsymbol{n} \cdot D^2 f \cdot \boldsymbol{n} - \kappa \frac{\partial f}{\partial \boldsymbol{n}} \right) \Big|_\Gamma .$$

If the function f is defined on Γ only, then we consider a normal extension of f and use the same definitions for the tangential derivatives.

Shape derivative of the energy. We use the concept of shape derivatives to understand the change in the energy induced by a given velocity field $\boldsymbol{V}$. Once we have the means to evaluate how any given velocity affects the energy, we can choose from the space of admissible velocities the particular velocity that decreases the energy (1) for a given Γ. We define the shape derivative of an energy $J(\Gamma)$ at Γ with respect to a velocity field $\boldsymbol{V}$ as the limit

$$dJ(\Gamma; \boldsymbol{V}) = \lim_{t \to 0} \frac{1}{t} (J(\Gamma_t) - J(\Gamma)),$$

where $\Gamma_t = \{x(t, X) : X \in \Gamma\}$ is the deformation of Γ by $\boldsymbol{V}$ via the ordinary differential equation $\frac{dx}{dt} = \boldsymbol{V}(x(t))$, $x(0) = X$. Shape derivative of energies $J(\Omega)$ depending on domains or regions Ω are defined similarly. We refer to the book [8] for more information on shape derivatives.

Lemma 1 ([8]). *The shape derivative of curve length $J(\Gamma) = |\Gamma| = \int_\Gamma d\Gamma$ with respect to velocity field $\boldsymbol{V}$ is $dJ(\Gamma; \boldsymbol{V}) = \int_\Gamma \kappa V d\Gamma$, where $V = \boldsymbol{V} \cdot \boldsymbol{n}$ is the normal component of the vector velocity.*

Lemma 2 ([5, 8, 12]). *The shape derivative of the data fidelity term*

$$J(\Omega_l) = \frac{1}{2} \int_{\Omega_l} \chi_D(x)(I(x) - c_l)^2 dx$$

from energy (1) *for domain Ω_l with respect to velocity field $\boldsymbol{V}$ is*

$$dJ(\Omega_l; \boldsymbol{V}) = \frac{1}{2} \int_{\partial\Omega_l} \chi_D(x)(I(x) - c_l)^2 V dx, \tag{2}$$

where $V = \boldsymbol{V} \cdot \boldsymbol{n}$ is the normal component of the vector velocity.

We will use Lemma 1 and Lemma 2 next to write the shape derivative for our energy (1) that is based on multiple curves and multiple regions.

Theorem 1. *The shape derivative of the energy* (1) *for $\Gamma = \cup_{l=1}^{n_\Omega} \Gamma_l \in C^2$ with respect to a given velocity field $\boldsymbol{V}$ is*

$$dJ(\Gamma; \boldsymbol{V}) = \int_\Gamma GV d\Gamma, \qquad G = \mu\kappa + f(\Gamma), \tag{3}$$

where G is the shape gradient, $V = \boldsymbol{V} \cdot \boldsymbol{n}$ is the normal component of the velocity and the image-based force term f is defined by

$$f|_{\Gamma_l} = (c_l^{out} - c_l^{in})\chi_D(x) \left(I(x) - \frac{1}{2}(c_l^{in} + c_l^{out}) \right), \tag{4}$$

in which $c_l^{in} = c_l = \frac{1}{|\Omega_l \cap D|} \int_{\Omega_l \cap D} I(x)dx$ is the average image intensity in the region $\Omega_l \cap D$ enclosed by Γ_l, and $c_l^{out} = c_k = \frac{1}{|\Omega_k \cap D|} \int_{\Omega_k \cap D} I(x)dx$ is the average over the outer region Ω_k enclosing both Ω_l and Γ_l. We can write the shape derivative more explicitly as

$$dJ(\Gamma; \boldsymbol{V}) = \mu \int_\Gamma \kappa V d\Gamma + \sum_{l=1}^{n_\Omega} (c_l^{out} - c_l^{in}) \int_{\Gamma_l} \chi_D(x) \left(I(x) - \frac{1}{2}(c_l^{in} + c_l^{out}) \right) V d\Gamma$$

Proof. The Euler-Lagrange equation of the energy (1) with respect to the unknown c_l gives $c_l = \frac{1}{|\Omega_l \cap D|} \int_{\Omega_l \cap D} I(x)dx$. We use Lemmas 1, 2 and write the shape derivative of the energy (1) as

$$dJ(\Gamma; \boldsymbol{V}) = \mu \int_\Gamma \kappa V d\Gamma + \frac{1}{2} \sum_{l=0}^{n_\Omega} \int_{\partial\Omega_l} \chi_D(x)(I(x) - c_l)^2 V d\Gamma. \tag{5}$$

Note that the boundary Ω_l consists of the curves Γ_l and $\{\Gamma_k : k \in IN(l)\}$, where $IN(l)$ is the set of indices of the curves immediately inside Γ_l. The domain Ω_0

does not have an outer boundary, it has only interior boundaries given by the curves $\{\Gamma_k : k \in IN(0)\}$. Thus we rewrite the shape derivative as follows

$$\begin{aligned} dJ(\Gamma;\boldsymbol{V}) = \mu\int_\Gamma \kappa V d\Gamma + \frac{1}{2}\sum_{k\in IN(0)}\int_{\Gamma_k}\chi_D(x)(I(x)-c_k)^2\boldsymbol{V}\cdot(-\boldsymbol{n}_k)d\Gamma \\ +\frac{1}{2}\sum_{l=1}^{n_\Omega}\Big(\int_{\Gamma_l}\chi_D(x)(I(x)-c_l)^2\boldsymbol{V}\cdot\boldsymbol{n}_l d\Gamma \\ +\sum_{k\in IN(l)}\int_{\Gamma_k}\chi_D(x)(I(x)-c_k)^2\boldsymbol{V}\cdot(-\boldsymbol{n}_k)d\Gamma\Big). \end{aligned} \tag{6}$$

Each simple curve Γ_l appears only twice in the expression (6), because Γ_l is the outer boundary of the domain Ω_l and it is an inner boundary of the domain Ω_m, namely $l \in IN(m)$. So we can collect all the integrals of Γ_l together and reorganize (6) as

$$\begin{aligned} dJ(\Gamma;\boldsymbol{V}) &= \mu\int_\Gamma \kappa V d\Gamma + \frac{1}{2}\sum_{l=1}^{n_\Omega}\int_{\Gamma_l}\chi_D(x)\left((I(x)-c_l^{out})^2-(I(x)-c_l^{in})^2\right)\boldsymbol{V}\cdot\boldsymbol{n}_l d\Gamma \\ &= \mu\int_\Gamma \kappa V d\Gamma + \sum_{l=1}^{n_\Omega}(c_l^{out}-c_l^{in})\int_{\Gamma_l}\chi_D(x)\left(I(x)-\frac{1}{2}(c_l^{in}+c_l^{out})\right)\boldsymbol{V}\cdot\boldsymbol{n}d\Gamma, \end{aligned}$$

where $c_l^{in} = c_l$ is the constant value for the domain Ω_l, for which $\partial\Omega_l$ is the outer boundary, and c_l^{out} is the constant value for the enclosing domain Ω_m, for which Γ_l is an inner boundary. This concludes our proof.

Gradient descent for minimization. Theorem 1 enables us to derive a gradient descent velocity, so that we can evolve a given initial curve Γ^0 continuously in a manner that decreases its energy $J(\Gamma)$ and drives it to a minimum of the energy (1). For this, we simply set $\boldsymbol{V} = -(\mu\boldsymbol{\kappa} + f(\Gamma)\boldsymbol{n})$ following the shape derivative equations (3), (4). By substituting this velocity in $V = \boldsymbol{V}\cdot\boldsymbol{n}$ and then in (3), we can verify that

$$dJ(\Gamma;\boldsymbol{V}) = \int_\Gamma(\mu\kappa+f)V d\Gamma = -\int_\Gamma(\mu\kappa+f)^2 d\Gamma \leqslant 0,$$

by recalling $\boldsymbol{\kappa} = \kappa\boldsymbol{n}$. Thus, $\boldsymbol{V}$ is indeed a gradient descent velocity, and to pursue the minimization, one can conceive a curve evolution scheme as follows: Start with initial curve Γ^0 and update the curve iteratively by

$$\boldsymbol{V}^k = -\mu\boldsymbol{\kappa}^k - f(\Gamma^k)\boldsymbol{n}^k, \quad \boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \tau^k\boldsymbol{V}^k,\ \forall\boldsymbol{X}^k \in \Gamma^k.$$

This update scheme can be viewed as an explicit time discretization of the evolution equation $\frac{dx}{dt} = \boldsymbol{V}(x) = -\mu\boldsymbol{\kappa} - f(\Gamma)\boldsymbol{n}$; we use the curve Γ^k at the current step to compute the geometric quantities $\boldsymbol{n}, \boldsymbol{\kappa}$ and the data term $f(\Gamma)$; we then compute the velocity $\boldsymbol{V}$, and finally update Γ^k with $\boldsymbol{V}$ to obtain Γ^{k+1}. The fact

that the scheme is explicit and contains the curvature term creates stability issues resulting in spurious oscillations on the curve as we iterate. This is a feature of this geometric update scheme and it is there regardless of the representation, whether it is parametric curves or level sets. This instability can be prevented by taking small steps τ, but this leads to too many iterations and increased computation time. To circumvent these difficulties, we propose a semi-implicit update scheme:

$$\boldsymbol{V}^{k+1} + \mu\boldsymbol{\kappa}^{k+1} = -f(\Gamma^k)\boldsymbol{n}^k, \quad \boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \tau^k\boldsymbol{V}^{k+1}, \quad \forall \boldsymbol{X}^k \in \Gamma^k,$$

where we choose to evaluate the curvature term $\boldsymbol{\kappa}$ in the next iteration rather than the current iteration. We recall $\boldsymbol{\kappa} = -\Delta_\Gamma \boldsymbol{X}$, so we can write $\boldsymbol{\kappa}^{k+1} = -\Delta_\Gamma \boldsymbol{X}^{k+1} = -\tau^k \Delta_\Gamma \boldsymbol{V}^{k+1} + \Delta_\Gamma \boldsymbol{X}^k$. Hence we obtain the following form of the semi-implicit update, in which the curvature does not appear explicitly

$$\begin{aligned} (Id - \mu\tau^k\Delta_\Gamma)\boldsymbol{V}^{k+1} &= -\mu\Delta_\Gamma \boldsymbol{X}^k - f(\Gamma^k)\boldsymbol{n}^k, \\ \boldsymbol{X}^{k+1} &= \boldsymbol{X}^k + \tau^k\boldsymbol{V}^{k+1}, \quad \forall \boldsymbol{X}^k \in \Gamma^k. \end{aligned} \tag{7}$$

Now computing the velocity $\boldsymbol{V}^{k+1}$ at each iteration requires inverting a second order tangential differential operator, namely computing

$$-(Id - \mu\tau^k\Delta_\Gamma)^{-1}(\mu\Delta_\Gamma \boldsymbol{X}^k - f(\Gamma^k)\boldsymbol{n}^k). \tag{8}$$

We will see that this operation can be done in linear time with respect to the number points on the curve; therefore, the semi-implicit step is asymptotically as efficient as the explicit step. Moreover, it is unconditionally stable with respect to step size τ^k, so we can take steps as large as we need in order to achieve significant energy decrease and to approach the minimum in few iterations.

Equation (8) can also viewed as preconditioning the gradient descent velocity with a smoothing operator, and this is helpful in understanding the stability of this scheme. In other words, we are solving for a gradient descent velocity in $H^1(\Gamma)$ space and this has been shown to have favorable properties in recent works [6, 24, 25]. Moreover, an H^1 gradient descent scheme seems to appear naturally when we use the second shape derivative to derive gradient descent velocities for curve integrals [13, 14]. Using the semi-implicit updating scheme (7), we devise the Algorithm 1 for iterative minimization.

At this point, we should point out two important pieces of Algorithm 1: the stopping criterion and step size selection. In many implementations of the Chan-Vese segmentation algorithm [5] with curve evolution and its variants, users fix the step size τ and take a fixed number of iterations. This is not a good approach, because sometimes the number of iterations may not be enough to reach the minimum, or sometimes one keeps taking many unnecessary iterations even when the curve is already at the minimum. The solution implemented in [10, 14] is to select the step that satisfies the Armijo energy decrease criterion [19] at each iteration and to stop when the norm of the shape gradient falls below a stopping tolerance. At each iteration, we choose a step size τ^k that satisfies

Algorithm 1 Gradient Descent Algorithm

set initial curve Γ^0 and $k = 0$
repeat
  mark domains $\{\Omega_l\}$ on pixels
  sum up pixels in Ω_l as $\mathrm{SUM_l}$ and compute average $c_l = \frac{1}{|\Omega_l \cap D|}\mathrm{SUM_l}$
  compute energy $J^k = J(\Gamma^k)$
  solve $(Id - \mu\tau^k \Delta_\Gamma)\boldsymbol{V}^{k+1} = -\mu\Delta_\Gamma \boldsymbol{X}^k - f(\Gamma^k)\boldsymbol{n}^k$
  test step size τ^k, modify if necessary to ensure energy decrease
  update $\boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \tau^k \boldsymbol{V}^{k+1}, \quad \forall \boldsymbol{X}^k \in \Gamma^k$
until $\|G^{k+1}\|_{L^2} < tol(\Gamma^k, \{c_l^k\})$

the nonmonotone energy decrease condition proposed by Zhang and Hager in [28] for continuous optimization

$$J(\Gamma^{k+1}) < C^k + \alpha\tau^k dJ(\Gamma^k; \boldsymbol{V}^k), \tag{9}$$

where $C^k = (\eta Q^{k-1} C^{k-1} + J(\Gamma^k))/Q^k,\ C^0 = J(\Gamma^0),\ Q^k = \eta Q^{k-1} + 1,\ Q^0 = 0$. We set $\alpha = 10^{-4}$, $\eta = 0.2$ in our experiments. We found that the nonmonotone energy decrease condition (9) is more robust and more efficient than the monotone counterpart. It allows energy increases in some iterations, but ensures good progress towards the minimum.

We found that a fixed tolerance on the norm of the shape gradient was not a robust stopping criterion across various image examples, e.g., poor contrast, unbalanced region sizes, etc. Thus we derived our novel dynamic cutoff tolerance. We impose the following relative tolerance on pointwise values of the data fidelity term (setting $\mu = 0$ in the shape gradient G (3)),

$$\left| I(x) - \frac{1}{2}(c_{in} + c_{out}) \right| < \varepsilon \frac{1}{2}|c_{in} - c_{out}|. \tag{10}$$

The term $\frac{1}{2}|c_{in} - c_{out}|$ gives a measure of contrast between neighboring regions. Noting $G|_{\Gamma_l} = (c_{in}^l - c_{out}^l)(I(x) - \frac{1}{2}(c_{in}^l + c_{out}^l))$, we multiply (10) by $(c_{in}^l - c_{out}^l)$ and integrate over Γ to obtain the L^2 norm of the shape gradient, thereby getting the following condition as a stopping criterion:

$$\|G\|_{L^2} = \left(\int_\Gamma |G|^2 d\Gamma \right)^{1/2} < tol(\Gamma^k, \{c_l^k\}) = \frac{1}{2} \min_l (|c_{in}^l - c_{out}^l)^2 |\Gamma|^{1/2} \varepsilon. \tag{11}$$

Figure 2 illustrates the effectiveness of the dynamic cutoff tolerance in identifying the dip in $\|G\|_{L^2}$ in different versions of the same segmentation problem under varying conditions. A fixed tolerance would give either premature termination or no termination for these images.

3 Discretization and Numerical Solution

The works building on the Chan-Vese approach [5] to image segmentation represent the curves as level set functions. The main advantage of the level set

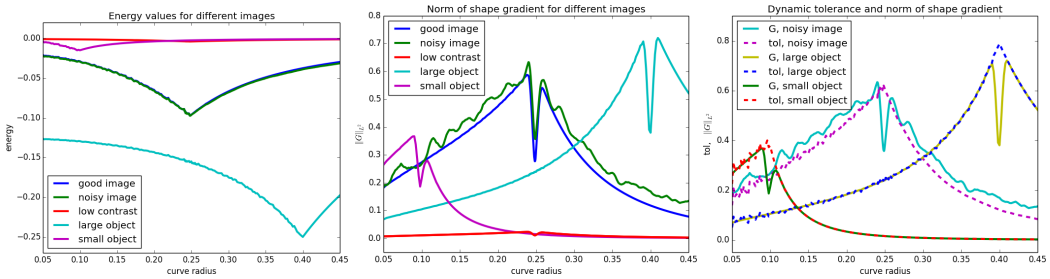

Fig. 2. Illustration of dynamic cutoff tolerance for different versions of a simple image example, a filled white circle centered at (0.5,0.5) in the foreground on a square image domain $[0,1]^2$ with zero background. Good, noisy and low contrast images have a circle of radius 0.25, large object has radius 0.40, small object has radius 0.10. Low contrast has gray-scale value 0.1 in the circle, whereas the others have gray-scale value 1.0. The noisy image has uniform noise added to all pixels with values between [-0.5,0.5]. The plots show the energy values, the L^2 norm of the shape gradient, the dynamic cutoff tolerance for circle curves centered at (0.5,0.5) on these images. The dip in the shape gradient norms signal the optimal circle radius in the minimization and cannot be captured with a fixed tolerance or criterion. On the other, the dynamic cutoff tolerance (shown with dotted curves on the right) tracks the norm of shape gradient very well. We can stop our iterative algorithm when the norm of the shape gradient falls significantly below the dynamic tolerance.

approach is that it can handle topological changes, such as splitting and merging of curves, easily without additional work. The main disadvantage of the level set approach is the high computational cost of introducing an additional dimension to represent a 1d object, often using the image grid itself as the basis of a 2d array representation for the curve. The level set representation can be difficult to maintain through the evolution and may require costly reinitialization or other regularization schemes [20]. Moreover, representing more than two phases with level sets requires using more than one level set function, thus increasing the computational cost further.

We choose to work with explicit Lagrangian representations of curves, because it is much more efficient with respect to memory use and running time than the level set approach; *all our velocity computations and curve updates have linear time complexity with respect to the number of points on the curve,* and the number of points used to represent the curves is much fewer than the number of pixels that the image contains; therefore, the number of variables that we need to deal with is much lower than it would be in the case of a level set approach (including narrow-band level set representation).

Curve representation and adaptivity. We approximate a continuous curve Γ as a polygonal curve Γ^h that consists of linear curve elements $\{\Gamma_i^h\}_{i=1}^m$. Thus the curve approximation is piecewise linear. The curve element Γ_i^h is a segment connecting a point $(x_{1,i}, x_{2,i})$ to the next point $(x_{1,i+1}, x_{2,i+1})$. This way the discrete curve Γ^h is stored as an ordered list of curve nodes $\{(x_{1,i}, x_{2,i})\}_{i=1}^m$ and *this representation, nor the velocity computations described below, does not*

require a parameterization. Therefore our approach is explicit Lagrangian, but not parametric.

An important issue in realizing the Lagrangian curve representation is where to put the nodes and how to distribute them, especially after the curve is deformed by the iterations in unpredictable ways. A reasonable strategy is to equidistribute the nodes yielding uniform element length, but this approach is suboptimal. Ideally we would like to distribute the nodes in an economical way, just enough the capture the geometry and the image faithfully, but not use more nodes than necessary in order to control the computational cost; namely we put more nodes where the geometry and image vary more and few nodes in flat regions. Thus, the curve representation is spatially adaptive and it changes dynamically during the gradient descent evolution following our adaptivity criteria. We realize this through two atomic operations on the curve:

- *Coarsening*: Combines two consecutive elements Γ_i^h, Γ_{i+1}^h into one by removing the shared node.
- *Refinement*: Splits an element Γ_i^h into two elements by adding a new node $(x_{1,i+\frac{1}{2}}, x_{2,i+\frac{1}{2}})$ in the middle. The new node is displaced in the normal direction to match the average of the curvatures at the two nodes of Γ_i^h. (Curvature at a node is estimated by fitting an osculating circle to the node and its two neighbors.)

The decision to refine or coarsen an element is based on the following two criteria:

- *Geometric criterion*: The error on an element Γ_i^h for piecewise linear approximation is bounded by $\max_{\Gamma_i^h} \kappa |\Gamma_i^h|$. If this error estimate is high, we refine. If it is very small, then we coarsen.
- *Data criterion*: We aim to evaluate if the element resolves the underlying image data. For this, we compare two numerical approximations of the integral $\int_{\Gamma_i^h} I(x) d\Gamma$ by a low order and a high order numerical quadrature rule. If the difference is large, we refine. If it is too small, we coarsen.

Since we have two criteria to guide adaptivity, we refine if one of the rules mark the element for refinement, and coarsen only if both rules mark the element and its neighbor for coarsening. We have observed that our approach to curve adaptivity works very well in a diverse set of scenarios. The curve dynamically adjusts the number of nodes during the minimization process capturing complicated geometries and images in a efficient and reliable manner. In addition to curve adaptivity, we have implemented topological changes for curves; we detect curve intersections and split or merge the curves if needed. Explaining the details of topological changes is beyond the scope of this paper and this will be pursued in a separate paper. Descriptions of other methods for topological changes of Lagrangian curves can be found in [9, 18]. We perform the procedures for curve adaptivity and topological changes at each iteration after the curve has been moved by the new velocity.

Finite element method for velocity. Computing the update velocity using our semi-implicit scheme at each iteration requires solving the following

velocity PDE for $\boldsymbol{V}^{k+1}$

$$-\mu\tau^k \Delta_\Gamma \boldsymbol{V}^{k+1} + \boldsymbol{V}^{k+1} = -\mu \Delta_\Gamma \boldsymbol{X}^k - f(\Gamma^k)\boldsymbol{n}^k \quad \text{on } \Gamma. \tag{12}$$

Note that since $\boldsymbol{V}, \boldsymbol{X}, \boldsymbol{n}$ are vectors, Equation (12) is actually two PDEs to be solved on Γ^k, one for each component of these vector functions. To solve the PDE (12), we discretize it using the finite element method (FEM) on curves. For this, we first write its weak form: we multiply the PDE (12) with suitable test functions $\boldsymbol{\phi}$, integrate over Γ^k, then integrate the tangential Laplacian term $\Delta_\Gamma \boldsymbol{V}$ by parts

$$\langle \boldsymbol{V}^{k+1}, \boldsymbol{\phi} \rangle + \mu\tau^{k+1} \langle \nabla_\Gamma \boldsymbol{V}^{k+1}, \nabla_\Gamma \boldsymbol{\phi} \rangle = \mu \langle \nabla_\Gamma \boldsymbol{X}, \nabla_\Gamma \boldsymbol{\phi} \rangle - \langle f(\Gamma^k)\boldsymbol{n}^k, \boldsymbol{\phi} \rangle. \tag{13}$$

The brackets $\langle \cdot, \cdot \rangle$ denote integrals on the curve Γ^k, i.e. $\langle f, g \rangle = \int_{\Gamma^k} f(x)g(x)d\Gamma$ and $\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \begin{pmatrix} \langle f_1, g_1 \rangle \\ \langle f_2, g_2 \rangle \end{pmatrix}$. Next we choose a finite set of nodal basis functions $\{\boldsymbol{\phi}_i\}$ to discretize the weak form of the velocity PDE (13). We use piecewise linear vector functions $\boldsymbol{\phi}_i = (\phi_i, \phi_i)$, such that ϕ_i is nonzero only on the elements $\Gamma^h_{i-1}, \Gamma^h_i$, but zero on the other elements, and satisfies

$$\phi_i(X_i) = 1, \quad \phi_i(X_{i-1}) = \phi_i(X_{i+1}) = 0.$$

We expand the unknown velocity in terms of the basis functions: $\boldsymbol{V}(X) = \Sigma_i \mathbf{V}_i \boldsymbol{\phi}_i(X)$ (omit the iteration index k to simplify notation), so that our new unknown becomes a finite vector of nodal velocity coefficients $\mathbf{V} = \{\mathbf{V}_i\}_{i=1}^m$. We expand the position vector $\mathbf{X} = \Sigma_i \mathbf{X}_i \boldsymbol{\phi}_i$ as well and obtain

$$\sum_j \boldsymbol{V}_j (\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle + \mu\tau \langle \nabla_\Gamma \boldsymbol{\phi}_i, \nabla_\Gamma \boldsymbol{\phi}_j \rangle) = -\langle \boldsymbol{\phi}_i, f(\Gamma^h)\boldsymbol{n} \rangle + \mu \sum_j \boldsymbol{X}_j \langle \nabla_\Gamma \boldsymbol{\phi}_i, \nabla_\Gamma \boldsymbol{\phi}_j \rangle.$$

We define the corresponding matrices $\mathbf{M}_{ij} = \langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle$, $\mathbf{A}_{ij} = \langle \nabla_\Gamma \boldsymbol{\phi}_i, \nabla_\Gamma \boldsymbol{\phi}_j \rangle$ and vector $\mathbf{f}_i = \langle \boldsymbol{\phi}_i, f(\Gamma^h)\boldsymbol{n} \rangle$, and obtain the compact linear system that we need to solve to compute velocity at each iteration:

$$(\mathbf{M} + \mu\tau\mathbf{A})\mathbf{V} = \mu\mathbf{A}\mathbf{X} - \mathbf{f}.$$

Note that we actually have $\mathbf{M} = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}$, $\mathbf{f} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}$. Since the basis functions are piecewise linear, the entries of the matrices M, A can be computed easily and are given by the following expressions:

$$M_{ij}, A_{ij} = \begin{cases} d_i/3, & 2/d_i, & \text{if } i = j, \\ d_i/6, & -1/d_i, & \text{if } i = j-1, j+1 \bmod m, \\ 0, & 0, & \text{otherwise,} \end{cases}$$

where $d_i = |\Gamma^h_i|$ is the length of the i^{th} curve element and m is the number of nodes on the curve. The entries of the vector $\mathbf{f}$, on the other hand, depend on the image and can be computed with numerical quadrature by interpolating

the image on each element Γ_i^h. *The matrices M, A are circulant tridiagonal for a simple curve; therefore, they can be inverted in linear time with respect to the number of nodes on the curve using the Thomas algorithm* . In the case of multiple simple curves, the matrices M, A consist of circulant triadiagonal blocks on their diagonals and zero elsewhere. The inversion is still linear time as each block is inverted independently.

4 Experiments

In this section, we present several numerical experiments demonstrating important aspects of our algorithm. We have confirmed the convergence and reliability of our algorithm in many experiments. Figure 3 shows the computed energy values, step sizes, the norm of the shape gradient and the dynamic cutoff criterion from one of our experiments (segmentation of the galaxy image). The energy goes down steadily on average, but the nonmonotone step size criterion sometimes allows increases in the energy. We see the step sizes are kept large as long as possible, but they decrease as we get close the minimum to ensure a good fit in the final positioning of the curves at the minimum. The L^2 norm of the shape gradient fluctuates through iterations. Our dynamic cutoff tolerance is small at the beginning and prevents premature termination. It increases gradually and helps detect the right dip in the norm of the shape gradient.

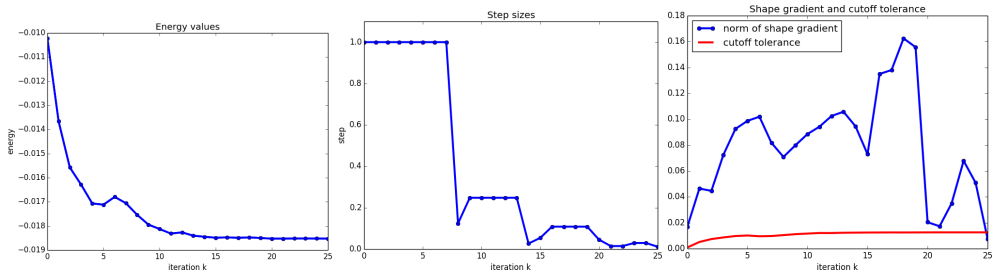

Fig. 3. The minimization dashboard. The shape energy (left), the step sizes (middle), L^2 norm of shape gradient (right, blue curve), the values of the dynamic cutoff tolerance (right, red curve) from the iterations of a segmentation example with galaxy image (Figure 4). The nonmonotone step size criterion allows some increases in energy to encourage large steps as much as possible. The dynamic cutoff tolerance tracking the L^2 norm of the shape gradient signals when to stop the iterations.

We also show two segmentation examples. One is a two-phase segmentation of the galaxy image shown in Figure 4. This example shows the judicious progress of the curve evolution. The curves take large steps in the first few iterations and achieve rapid energy decreases. The steps get smaller gradually to ensure convergence and a good fit at the minimum. The step sizes are determined by the nonmonotone step size selection criterion. The number of nodes on the curves changes at each iteration to ensure good representation of the geometry and

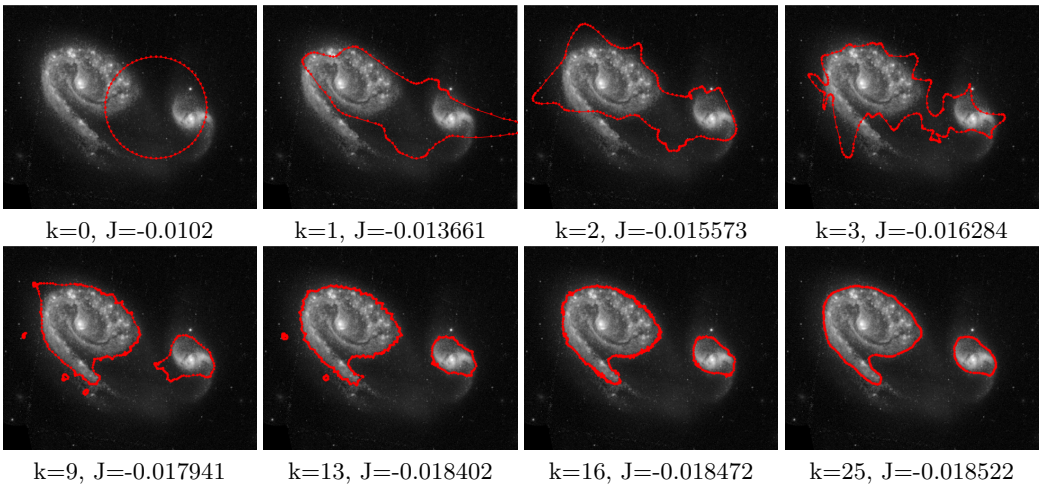

Fig. 4. Segmentation of galaxy image. We observe large displacements in the first iterations to achieve sharp decreases in the energy and small displacements in the final iterations to ensure a good fit. The curves adapt to the new configurations easily by adding and subtracting nodes, always ensuring a good representation of the geometry.

resolving the image features. The curves have high quality representations at all times. They do not suffer from entanglements or other problems typically expected in front-tracking techniques. Moreover, topological changes, such as merging and splitting, are handled in a graceful manner by our topology surgery routines.

The second segmentation example is a noisy synthetic multiphase image, shown in Figure 5. The behaviour of the curve evolution is similar to that we observe for the galaxy image. In this image, we additionally observe adaptation of the phases. We start with two phases, increase to three phases right away in the first iteration, then increase to five, back to four, and finally settle at five phases when we terminate at the minimum. We see that the algorithm finds the correct number of phases without the user specifying this number a priori.

References

1. E. Bae and X.-C. Tai. Graph cut optimization for the piecewise constant level set method applied to multiphase image segmentation. In *Scale space and variational methods in computer vision*, pages 1–13. Springer, 2009.
2. E. Bae, J. Yuan, and X.-C. Tai. Global minimization for continuous multiphase partitioning problems using a dual approach. *International journal of computer vision*, 92(1):112–129, 2011.
3. Y. Boykov and G. Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.
4. T. F. Chan, S. Esedoglu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

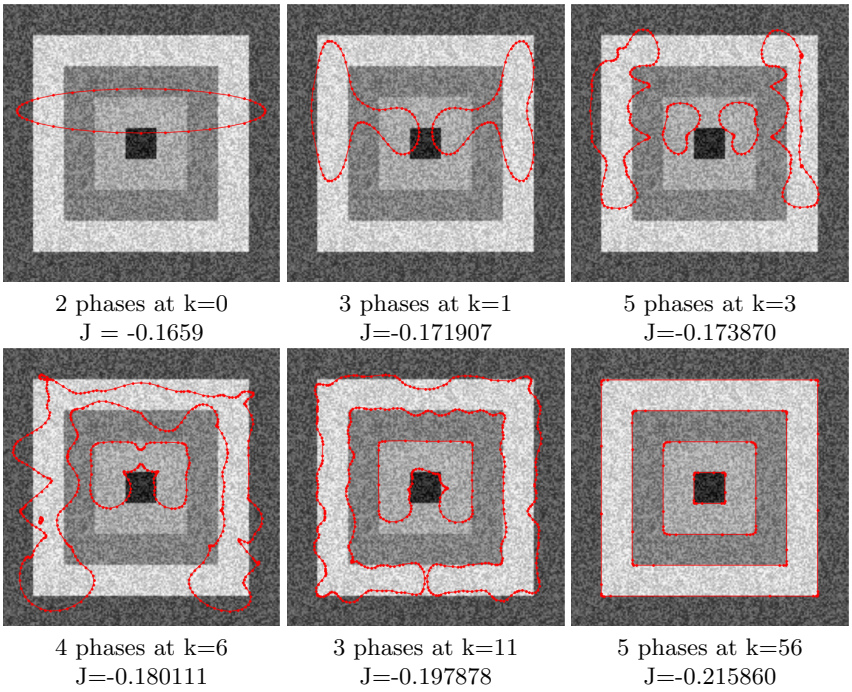

Fig. 5. Segmentation of synthetic multiphase image. This example shows segmentation of a noisy multiphase image. The curves take large steps and achieve large energy decreases at the beginning. They take small conservative steps close to the minimum to ensure a good fit and convergence. The number of phases changes during the evolution and finalizes in five phases. The number of nodes on the curves changes adaptively as well, increasing or decreasing as needed. The final curves have few nodes, because the region boundaries are straight.

5. T. F. Chan and L. A. Vese. Active contours without edges. *Image Processing, IEEE Transactions on*, 10(2):266–277, 2001.
6. G. Charpiat, P. Maurel, J.-P. Pons, R. Keriven, and O. Faugeras. Generalized gradients: Priors on minimization flows. *IJCV*, 73(3):325–344, 2007.
7. G. Chung and L. A. Vese. Image segmentation using a multilayer level-set approach. *Computing and Visualization in Science*, 12(6):267–285, 2009.
8. M. C. Delfour and J.-P. Zolésio. *Shapes and Geometries*. Advances in Design and Control. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.
9. H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, 83(2):140–171, 2001.
10. G. Doğan, P. Morin, and R. H. Nochetto. A variational shape optimization approach for image segmentation with a Mumford-Shah functional. *SIAM J. Sci. Comput.*, 30(6):3028–3049, 2008.
11. A. Dubrovina, G. Rosman, and R. Kimmel. *Active contours for multi-region image segmentation with a single level set function*. Springer, 2013.

12. M. Hintermüller and A. Laurain. Multiphase image segmentation and modulation recovery based on shape and topological sensitivity. *J. Math. Imaging Vision*, 35(1):1–22, 2009.
13. M. Hintermüller and W. Ring. A second order shape optimization approach for image segmentation. *SIAM J. Appl. Math.*, 64(2):442–467, 2003/04.
14. 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(1-2):19–42, 2004. Special issue on mathematics and image analysis.
15. Y. M. Jung, S. H. Kang, and J. Shen. Multiphase image segmentation via modica-mortola phase transition. *SIAM Journal on Applied Mathematics*, 67(5):1213–1232, 2007.
16. V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.
17. J. Lellmann and C. Schnörr. Continuous multiclass labeling approaches and algorithms. *SIAM Journal on Imaging Sciences*, 4(4):1049–1096, 2011.
18. T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4(2):73–91, 2000.
19. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.
20. S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.
21. T. Pock, A. Chambolle, D. Cremers, and H. Bischof. A convex relaxation approach for computing minimal partitions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 810–817. IEEE, 2009.
22. B. Sandberg, S. H. Kang, and T. F. Chan. Unsupervised multiphase segmentation: A phase balancing model. *Image Processing, IEEE Transactions on*, 19(1):119–130, 2010.
23. J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.
24. G. Sundaramoorthi, A. Yezzi, and A. C. Mennucci. Sobolev active contours. *IJCV*, 73(3):345–366, 2007.
25. G. Sundaramoorthi, A. Yezzi, A. C. Mennucci, and G. Sapiro. New possibilities with Sobolev active contours. In *Proceedings of the 1st International Conference on Scale Space Methods and Variational Methods in Computer Vision*, 2007.
26. X.-C. Tai, O. Christiansen, P. Lin, and I. Skjælaaen. Image segmentation using some piecewise constant level set methods with mbo type of projection. *International Journal of Computer Vision*, 73(1):61–76, 2007.
27. L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International journal of computer vision*, 50(3):271–293, 2002.
28. H. Zhang and W. W. Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM Journal on Optimization*, 14(4):1043–1056, 2004.