Fast Minimization of Region-based Active Contours using the Shape Hessian of the Energy

Günay Doğan

Theiss Research,
National Institute of Standards and Technology,
Gaithersburg, MD 20899-8990, USA. **gunay.dogan@nist.gov**

Abstract. We propose a novel shape optimization algorithm for region-based active contour models. Region-based active contours are preferred for many segmentation problems, because they incorporate more global information by aggregating cues or statistics over the distinct regions defined by the contour configuration. This makes them effective in a diverse array of segmentation scenarios, also more robust to contour initializations, Unfortunately they are also more expensive computationally, because a significant part of the optimization involves repeated integrations of the image features over the regions through the many iterations of the contour updates. Accordingly, we aim to decrease the overall computational cost of region-based active contours by reducing the cost of an individual iteration, and the total number of iterations. To this end, we first develop a Lagrangian curve representation that is spatially adaptive and economical in terms of the number of nodes used. Then we perform the shape sensitivity analysis of the general form of the region-based segmentation energy. In particular, we compute the second variation or the shape Hessian of the energy, and we use this to compute fast descent directions for the contours to significantly reduce the computational cost. Our implementation builds on a finite element discretization of the whole framework, including the contours. This results in efficient velocity computations in linear time with respect to the number of contour nodes.

1 Introduction

Image segmentation is the problem of partitioning a given image into distinct homogeneous regions with respect to image values or statistics. Thus, it is often formulated as the problem of finding the boundaries that define the best such partition: we would like each region to be as uniform as possible within itself, but as different as possible from its neighbors across the boundaries. For 2d images, this leads to a shape optimization problem, in which we try to compute the set of optimal curves minimizing a shape energy encoding this expectation of a good partition. Typically, one solves such problems by starting with a set of suboptimal initial curves (or guesses), and updating or deforming the curves until they reach an optimal configuration. In this paper, we propose to use efficient Lagrangian representations for the curves, and building on these representations, we develop a novel optimization algorithm that reaches the minima fast by exploiting the second order sensitivity (Hessian) of the shape energy.

Our target is region-based shape energies used for image segmentation. Note that the characteristics of each region in a segmentation formulation can be conveniently

quantified by the statistics of the image features in the region [4]. The statistics computations can often be expressed as various integrals over the regions. This results in shape energies composed of such integrals. Often the integrands of these integrals vary spatially, but also depend on other integrals over these regions. An example is the following energy (1) proposed by Chan and Vese in [2]. It aims to find a partitioning of the image domain into a foreground region Ω_1 and a background region Ω_2 (inside and outside the boundary curve Γ respectively), each with distinct averages c_1, c_2 of the image intensity $I(x)$ respectively.

$$J(\Gamma) = \mu \int_\Gamma d\Gamma + \frac{1}{2} \sum_{i=1,2} \int_{\Omega_i} (I(x) - c_i)^2 dx, \quad \mu > 0, \quad c_i = \frac{1}{|\Omega_i|} \int_{\Omega_i} I(x) dx. \tag{1}$$

More general approaches to incorporating statistics into the shape optimization formulation are described in [4, 10]. One can develop a more general statistical formulation, by considering a Bayesian interpretation of the estimation problem and trying to maximize the a posteriori probability $p(\{\Omega_1, \Omega_2\}|I)$, namely the likelihood of having a certain partitioning $\{\Omega_1, \Omega_2\}$ given the image I (multiple phases or regions $\{\Omega_i\}_{i=1}^m$ are possible, but not considered in this paper to simplify the presentation). We can write $p(\{\Omega_1, \Omega_2\}|I) \propto p(I|\{\Omega_1, \Omega_2\})\ p(\{\Omega_1, \Omega_2\})$ and separate the a priori shape information $p(\{\Omega_1, \Omega_2\})$ from image-based cues encoded in $p(I|\{\Omega_1, \Omega_2\})$. A common example of the a priori shape term would be $p(\{\Omega_1, \Omega_2\}) \propto e^{-\mu|\Gamma|}$. Assuming no correlation between labelings of regions and parametric probability distributions with parameters θ_1, θ_2, one can simplify the conditional probability:
$p(I|\{\Omega_1, \Omega_2\}) = p(I|\Omega_1)p(I|\Omega_2) = p(I|\theta_1)p(I|\theta_2), \quad \theta_1 = \theta(\Omega_1),\ \theta_2 = \theta(\Omega_2).$
Maximizing the probability $p(\{\Omega_1, \Omega_2\}|I)$ is equivalent to minimizing its negative logarithm. Thus we end up with the following energy

$$J(\Gamma) = \mu \int_\Gamma d\Gamma - \int_{\Omega_1} \log p(I(x)|\theta_1) dx - \int_{\Omega_2} \log p(I(x)|\theta_2) dx. \tag{2}$$

The parameters θ_i depend on the form of the probability density function and often involve integrals over the regions Ω_i. For example, the Gaussian probability density function has the form $p_i(s) = p(s|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(s-c_i)^2}{2\sigma_i^2}\right)$, where the parameters c_i, σ_i are computed by the integrals $c_i = \frac{1}{|\Omega_i|} \int_{\Omega_i} I(x) dx, \quad \sigma_i^2 = \frac{1}{|\Omega_i|} \int_{\Omega_i} (I(x) - c_i)^2 dx$. It is not hard to see that we can concoct a diverse collection of statistical formulations where shape energies with region integrals play a central role. Thus, shape energies with integral parameters have significant use for image segmentation, and the prototype form for energies with integrals is

$$E(\Omega) = \int_\Omega g(x, \theta_w(\Omega)) dx, \qquad \theta_w(\Omega) = \int_\Omega w(x) dx. \tag{3}$$

or one whose weight function g may depend on multiple such integrals

$$E(\Omega) = \int_\Omega g(x, \theta_{w_1}(\Omega), \ldots, \theta_{w_m}(\Omega)) dx, \qquad \theta_{w_i}(\Omega) = \int_\Omega w_i(x) dx. \tag{4}$$

Combining (3), (4) for multiple regions leads to

$$J(\Gamma) = \mu \int_\Gamma d\Gamma + E(\Omega_1) + E(\Omega_2), \qquad \left(\text{or} \quad J(\Gamma) = \mu \int_\Gamma d\Gamma + \sum_{i=1}^{n_\Omega} E(\Omega_i) \right) \quad (5)$$

for background/foreground (or multi-region) segmentation energies. The building blocks (3), (4) and the resulting energy (5) will be the main focus of this paper. We will perform the shape sensitivity analysis for (3), (4), (5) and use it to develop a fast shape optimization algorithm to compute the optimal boundaries. Our first contribution in the shape sensitivity analysis is the computation of the shape Hessian of region energies (3), (4), (5) (the first shape derivative was computed in [1]). The explicit formula for the shape Hessian enables us to compute superior descent directions to perform the shape optimization effectively. Our second contribution is the efficient discretization of the velocity equations and descent process on a foundation of spatially adaptive Lagrangian curves using the finite element method. This results in linear time complexity with respect to the number of curve nodes, except for the region integrals, which require traversing the image pixels. We demonstrate the effectiveness of our algorithm using the Chan-Vese model (1) as a test case.

2 Differential Geometry and Shape Derivatives

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 $\boldsymbol{V}$ affects the energy, we will be able to choose from the space of admissible velocities the particular velocity that decreases the energy for a given curve Γ. We define the *shape derivative* of an energy $J(\Gamma)$ at Γ with respect to 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 [5] for more information on shape derivatives.

For a domain-dependent function $\varphi(\Omega)$, the *material derivative* $\dot{\varphi}(\Omega; \boldsymbol{V})$ and the *shape derivative* $\varphi'(\Omega; \boldsymbol{V})$ at Ω in direction $\boldsymbol{V}$ are defined as follows [11, Def. 2.85,2.88]
$\dot{\varphi}(\Omega; \boldsymbol{V}) = \lim_{t\to 0} \frac{1}{t}\big(\varphi(x(t,\cdot), \Omega_t) - \varphi(\cdot, \Omega_0)\big), \quad \varphi'(\Omega; \boldsymbol{V}) = \dot{\varphi}(\Omega; \boldsymbol{V}) - \nabla\varphi \cdot \boldsymbol{V}.$

Before we start deriving the shape derivatives of the energies (3), (4), (5), we need 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 n, κ, $\boldsymbol{\kappa}(:= \kappa n)$ respectively. For a given function $f \in C^2(\mathcal{D})$ on the image domain $\mathcal{D}$, we define tangential gradient $\nabla_\Gamma f$ and tangential Laplacian $\Delta_\Gamma f$:
$\nabla_\Gamma f = (\nabla f - \partial_n f n)|_\Gamma, \quad \Delta_\Gamma f = \big(\Delta f - n \cdot D^2 f \cdot n - \kappa \partial_n f\big)\big|_\Gamma.$
Now we can pursue the shape derivatives of the shape energies (1), (3), (4), (5).

Lemma 1 ([5]). *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 $V = \boldsymbol{V} \cdot n$ is the normal component of the vector velocity.*

Theorem 1 ([11, Sect. 2.31, 2.33]). *Let $\phi = \phi(x, \Omega), \psi = \psi(x, \Gamma)$ be given so that the shape derivatives $\phi' = \phi'(\Omega; \boldsymbol{V}), \psi' = \psi'(\Gamma; \boldsymbol{V})$ exist. Then the shape energies $J_1(\Omega) = \int_\Omega \phi(x, \Omega)dx, J_2(\Gamma) = \int_\Gamma \psi(x, \Gamma)d\Gamma$ are shape differentiable and we have*

$$dJ_1(\Omega; \boldsymbol{V}) = \int_\Omega \phi' dx + \int_\Gamma \phi V d\Gamma, \quad dJ_2(\Gamma; \boldsymbol{V}) = \int_\Gamma (\psi' + (\psi\kappa + \partial_n\psi)V)d\Gamma. \quad (6)$$

As the shape derivative $dJ(\Gamma; \boldsymbol{V})$ depends only on $V = \boldsymbol{V} \cdot n$, the normal component of the velocity [5], we use V (to imply $\boldsymbol{V} = Vn$) in our derivations from now on and assume a normal extension, i.e. $\frac{\partial V}{\partial n} = 0$, if needed.

Now using scalar velocity fields V, W (recall $\boldsymbol{V} = Vn, \boldsymbol{W} = Wn$) to perturb Ω and we define the second shape derivative of $J(\Omega), \varphi(\Omega)$ as follows [5]
$d^2J(\Omega; V, W) = d\left(dJ(\Omega; V)\right)(\Omega; W), \quad \varphi''(\Omega; V, W) = (\varphi(\Omega; V))'(\Omega; W).$

Lemma 2 ([5, 8]). *The second shape derivative of curve length $J(\Gamma) = |\Gamma| = \int_\Gamma d\Gamma$ with respect to velocity fields V, W is $d^2J(\Gamma; V, W) = \int_\Gamma \nabla_\Gamma V \cdot \nabla_\Gamma W d\Gamma$.*

Theorem 2. *Let $\phi = \phi(x, \Omega)$ be given so that the first and the second shape derivatives $\phi'_V = \phi'(\Omega; V)$, $\phi'' = \phi''(\Omega; V, W)$ exist. Then the second shape derivative of the domain energy $J(\Omega) = \int_\Omega \phi(x, \Omega)dx$ with respect to velocities V, W is given by*

$$d^2J(\Omega; V, W) = \int_\Omega \phi'' dx + \int_\Gamma \left(\phi'_W V + \phi'_V W\right) dS + \int_\Gamma \left(\partial_n\phi + \kappa\phi\right) VW d\Gamma. \quad (7)$$

Proof. For $J(\Omega) = \int_\Omega \phi(x, \Omega)dx$, the first shape derivative at Ω in direction V is
$dJ(\Omega; V) = J_1(\Omega) + J_2(\Gamma) = \int_\Omega \phi'(\Omega; V)dx + \int_\Gamma \phi V d\Gamma$, using Theorem 1.
We can continue to compute
$dJ_2(\Gamma; W) = \int_\Gamma \phi'(\Omega; W)V d\Gamma + \int_\Gamma(\partial_n\phi V + \phi V\kappa)W d\Gamma,$
$dJ_1(\Omega; W) = \int_\Gamma \phi''(\Omega; V, W)dx + \int_\Gamma \phi'(\Omega; V)W d\Gamma.$
which can be summed into $d^2J(\Omega; V, W) = dJ_1(\Omega; W) + dJ_2(\Gamma; W)$.

Proposition 1. *The first shape derivative of the shape energy* (3) *at Ω with respect to velocity V is given by $dJ(\Omega; V) = \int_\Gamma \left(g(x, \theta_w(\Omega)) + \theta_{g_p}(\Omega)w(x)\right) V d\Gamma$ [1].*
The second shape derivative of (3) *with respect to velocities V, W is given by*

$$\begin{aligned} d^2J(\Omega; V, W) = &\int_\Gamma \left(\partial_n g + \theta_{g_p}\partial_n w + \kappa(g + \theta_{g_p}w)\right) VW d\Gamma \\ &+ \int_\Gamma g_p V d\Gamma \int_\Gamma wW d\Gamma + \int_\Gamma wV d\Gamma \int_\Gamma g_p W d\Gamma + \theta_{g_{pp}} \int_\Gamma wV d\Gamma \int_\Gamma wW d\Gamma, \end{aligned}$$

We use the short notation for the functions $g = g(x, \theta_w(\Omega))$, $w = w(x)$, and g_p, g_{pp} denote the derivatives of $g(x, \theta_w(\Omega))$ with respect to its second variable. $\theta_{g_p}, \theta_{g_{pp}}$ are the integrals of g_p, g_{pp} over the domain Ω.

Proof. Let $\phi(x, \Omega) = g(x, \theta_w(\Omega))$ in Theorem 1, and calculate $\phi'(\Omega; V)$.
$\phi'(\Omega; V) = g_p(x, \theta_w)\theta'_w = g_p(x, \theta_w) \int_\Gamma w(y)V d\Gamma(y).$ (using Thm.1 on θ_w)
We substitute $\phi'(\Omega; V)$ in the form (6) for $dJ(\Omega; V)$

$dJ(\Omega;V) = \int_\Omega g_p(x,\theta_w)\left(\int_\Gamma w(y)Vd\Gamma(y)\right)dx + \int_\Gamma g(x,\theta_w)Vd\Gamma.$
Since $\theta_{g_p} = \int_\Omega g_p(x,\theta_w)dx$, we can exchange the order of integration and obtain
$dJ(\Omega;V) = \int_\Gamma \left(g(x,\theta_w) + \theta_{g_p} w(x)\right) Vd\Gamma.$
For the second shape derivative, we use Theorem 2. Compute $\phi'' = \phi''(\Omega;V,W)$,
$\phi'' = (g_p)' \int_\Gamma w(y)Vd\Gamma + g_p\left(\int_\Gamma w(y)Vd\Gamma\right)' \qquad$ (we use Thm.1)
$\phi'' = g_{pp}\int_\Gamma w(z)Vd\Gamma(z)\int_\Gamma w(y)Wd\Gamma(y) + g_p\int_\Omega \left(\partial_n w(y) + \kappa w(y)\right)VWd\Gamma.$
We substitute in (7) and reorganize the various terms.

We do not derive the second shape derivative of the energy (4) because of limited space in this paper. It follows the same steps of the derivation of the energy (3). Its first derivative can be found in [1]. The first shape derivative of the Chan-Vese energy (1) is given by the following proposition ([2]).

Proposition 2. *The first shape derivative of the energy* (1) *at Γ with respect to V is*

$$dJ(\Gamma;V) = \int_\Gamma (\mu\kappa + f(\Gamma))Vd\Gamma = \int_\Gamma (\mu\kappa + (c_2-c_1)(I(x) - \frac{1}{2}(c_1+c_2)))Vd\Gamma. \quad (8)$$

Proposition 3. *The second shape derivative of energy* (1) *at Γ with respect to V, W is*

$$\begin{aligned} d^2J(\Gamma;V,W) =& \mu\int_\Gamma \nabla_\Gamma V\cdot\nabla_\Gamma W d\Gamma + \int_\Gamma \beta(x,\Gamma)VWd\Gamma \\ &- \sum_{i=1,2}\frac{1}{|\Omega_i|}\int_\Gamma (I(x)-c_i)Vd\Gamma\int_\Gamma (I(x)-c_i)Wd\Gamma, \end{aligned}$$

where $\beta(x,\Gamma) = \frac{1}{2}(c_2-c_1)\left((I - \frac{c_1+c_2}{2})\kappa + \partial_n I\right)$. Or more concisely:

$$d^2J(\Gamma;V,W) = \mu\langle\nabla_\Gamma V, \nabla_\Gamma W\rangle + \langle\beta V, W\rangle - \mathcal{S}_1(V)\mathcal{S}_1(W) - \mathcal{S}_2(V)\mathcal{S}_2(W), \quad (9)$$

where $\langle\cdot,\cdot\rangle$ denotes the L^2 scalar product $\langle f,g\rangle = \int_\Gamma fgd\Gamma$, and $\mathcal{S}_i(V)$, is the boundary integral $\mathcal{S}_i(V) = |\Omega_i|^{-1/2}\int_\Gamma (I(x)-c_i)Vd\Gamma, \quad i=1,2.$

Proof. The proof follows from Lemma 2 & Prop.1 with $g(x,p_1,p_2) = \frac{1}{2}\left(I(x) - \frac{p_1}{p_2}\right)^2$. substituted in $J(\Gamma) = \mu\int_\Gamma d\Gamma + \sum_{i=1,2}\int_{\Omega_i} g(x,\theta_I(\Omega_i),\theta_{\mathbb{1}}(\Omega_i))dx$.

3 The Minimization Algorithm

In this section, we use the shape derivatives to devise an iterative minimization algorithm for the energies (1), (5). The algorithm starts with a given initial curve Γ_0, computes a descent velocity $\boldsymbol{V}$, deforms the curve with the descent velocity $\boldsymbol{V}$, repeats this process until convergence, at which point the curve should be at a minimum of the energy, and a valid segmentation should be attained. Thus the energy minimization procedure involves updating the current curve Γ^k at each iteration to obtain the next curve Γ^{k+1} with the descent velocity $\boldsymbol{V}$ $(= Vn)$ by

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \tau\boldsymbol{V}, \quad \forall\boldsymbol{X}^k \in \Gamma^k, \quad (10)$$

where $\tau > 0$ is a step size parameter (or time step for the curve evolution). Computing the descent velocity requires knowing the normal n and the curvature κ of the curve. Thus we use three additional identities $\boldsymbol{\kappa} = -\Delta_\Gamma \boldsymbol{X},\ \kappa = \boldsymbol{\kappa} \cdot n,\ \boldsymbol{V} = Vn$, to relate the curve points $\boldsymbol{X}$, the mean curvature κ, the vector curvature $\boldsymbol{\kappa}$, the scalar velocity V, finally the vector velocity $\boldsymbol{V}$. Let us now state the outline of the minimization algorithm, which will consist of the following steps:

choose an initial curve Γ_0
repeat
  mark the pixels in the domains Ω_1, Ω_2
  compute integrals $\theta_I(\Omega_i) = \int_{\Omega_i} I,\ \theta_{\mathbb{1}}(\Omega_i) = \int_{\Omega_i} \mathbb{1}$ and then the energy $J(\Gamma)$
  compute the descent velocity $\boldsymbol{V}$
  compute step size τ ensuring energy decrease
  update Γ^k to obtain Γ^{k+1} using (10)
until stopping criterion is satisfied

The three critical components to effectiveness of this algorithm are: 1) the choice of the descent velocity, 2) the choice of step size, and 3) the stopping criterion. Some descent velocities, which we elaborate below, yield more stable evolutions and faster convergence. Moreover, for successful convergence, the step size should be matched to the velocity at each iteration (not set to a fixed value like many implementations in literature). Finally, the stopping criterion should be set appropriately depending on the image and the curves, in order to avoid early termination or unnecessary extra iterations that do not improve segmentation. We describe these components next.

Gradient descent velocities. The first shape derivative already gives us a gradient descent velocity, by setting the velocity equal to negative shape gradient: $V = -G = -(\mu\kappa + f(\Gamma))$, (because this implies $dJ(\Gamma; V) = -\int_\Gamma G^2 d\Gamma \leqslant 0$). The velocity $V = -G$ was used in the original paper [2] by Chan and Vese as part of a level set implementation.

A more general and flexible way to define a gradient descent velocity is to choose a scalar product $b(\cdot,\cdot)$ that induces a Hilbert space $B(\Gamma)$ on the curve Γ [3, 12], and use the following equation to compute the associated gradient descent velocity V:

$$b(V, W) = -dJ(\Gamma; W) = -\langle G, W\rangle, \qquad \forall W \in B(\Gamma). \tag{11}$$

The velocity V computed in this way is also a gradient descent velocity: $dJ(\Gamma; V) = -b(V, V) \leqslant 0$. Examples of possible scalar products are $b(\cdot,\cdot)$ the L^2 scalar product and the weighted H^1 scalar product

$$\langle V, W\rangle_{L^2} = \langle V, W\rangle = \int_\Gamma VW d\Gamma, \quad \langle V, W\rangle_{H^1} = \langle \alpha\nabla_\Gamma V, \nabla_\Gamma W\rangle + \langle \beta V, W\rangle, \tag{12}$$

where $\alpha = \alpha(x, \Gamma) > 0$ and $\beta = \beta(x, \Gamma) > 0$. The gradient descent velocity computed with the L^2 scalar product turns out to be $V = -G$ mentioned above, and is the most common choice in the literature for active contours problem.

In practice, the L^2 gradient velocity works reasonably, but it is not very efficient. The gradient descent method is known to converge slowly in the optimization literature [9]. The gradient depends on the scaling of the problem, and the choice of the right step size is not straight-forward. We propose to remedy this by normalizing the L^2 gradient

velocity with its norm, and by using a standard line search algorithm for step size. This is explained in detail below. Another important problem with the velocity $V = -G$ is that the curvature term in $G = \mu\kappa + f$ makes the iterations unstable, and one has to take small steps to evolve the curve Γ in a stable manner. To address this, we use a semi-implicit time step discretization of the velocity and curve evolution, which is unconditionably stable. We should note that using an H^1 scalar product also stabilizes the velocity computation, and one can continue with the simpler and more efficient explicit stepping scheme for curve evolution.

Newton's method for minimization. Having the explicit formula for the second shape derivative or the shape Hessian (9) opens up opportunities to reduce the number of iterations or energy evaluations by pursuing a Newton-type optimization algorithm. One can compute a Newton velocity by setting $b(V, W) = d^2J(\Gamma; V, W)$ in the velocity equation (11). This was proposed in the recent works [6–8]. Note that the expression (9) is symmetric and resembles an H^1 scalar product (12). However, it is not positive definite because of the arbitrary range of the coefficients $\beta(x, \Gamma)$ and the last two terms in (9), which are negative semi-definite. Therefore, using $b(V, W) = d^2J(\Gamma; V, W)$ in (11) is not guaranteed to produce a descent direction. We would like to work with a positive definite Hessian and retain some structure of the second shape derivative $d^2J(\Gamma; \cdot, \cdot)$ to help improve convergence. For this, we replace β in (9) with a thresholded version $\beta_+(x, \Gamma) = \max(\beta(x, \Gamma), \beta_M)$ ($\beta_M > 0$) and omit the last two terms. Then the sum of the first and second integrals in (9) is positive definite. The choice of β_M needs care. One is inclined to choose β_M as small as practically possible. But the corresponding discrete Hessian matrix gets more and more ill-conditioned as β_M gets smaller; therefore, in practice, we need to choose a value that is not very small as an acceptable trade-off. For energy (1), we set $\beta_M = 0.1|c_2 - c_1| \max |I|$, a threshold that is automatically adjusted to the image intensity and contrast between regions.

Explicit vs semi-implicit stepping At this point, we should take a closer look at the interaction of the descent velocity $\boldsymbol{V}$ and the step size τ. Although the crucial criterion of energy reduction for step size selection is encoded in the line search routine, described in the next subsection, the choice of velocity might impose more conditions on the step size. Given the curve points $\boldsymbol{X}$ of the curve, a natural sequence to compute the descent velocity and to update the curve is the following: 1) compute $f(\Gamma^k), n^k, \kappa^k$ from $\boldsymbol{X}$, 2) compute scalar velocity V^k using one of the methods described above, 3) compute $\boldsymbol{V}^k = Vn^k$, 4) update $\boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \tau\boldsymbol{V}^k$. These steps constitute the explicit stepping scheme. This works well with H^1 gradient descent, Newton's method, and does not require any additional conditions on τ. If we use L^2 gradient descent, as mostly done for this problem, the iterations are not very stable and one needs to take small steps τ, consequently many more iterations.

An alternative stepping scheme in the case of L^2 gradient descent would be a semi-implicit stepping scheme [6]. In this approach, we take $\boldsymbol{X}^k, n^k$ as known quantities and $\kappa^{k+1}, \boldsymbol{\kappa}^{k+1}, V^{k+1}, \boldsymbol{V}^{k+1}$ as unknowns at each iteration. We write $\boldsymbol{\kappa}^{k+1} = -\Delta_\Gamma \boldsymbol{X}^{k+1} = -\Delta_\Gamma \boldsymbol{X}^k + \tau\Delta_\Gamma \boldsymbol{V}^{k+1}$, and obtain the following system of equations

$$\begin{aligned} \boldsymbol{\kappa}^{k+1} - \tau\Delta_\Gamma \boldsymbol{V}^{k+1} = -\Delta_\Gamma \boldsymbol{X}^k, \quad \kappa^{k+1} - \boldsymbol{\kappa}^k \cdot n^{k+1} = 0, \\ V^{k+1} + \mu\kappa^{k+1} = -f(\Gamma^k), \quad \boldsymbol{V}^{k+1} - V^{k+1}n^k = 0, \end{aligned}$$

which we need to solve simultaneously for the unknowns. This semi-implicit scheme is unconditionally stable, i.e. it does not impose any conditions on the step size τ. Its disadvantage is that we now need to solve a system of equations, and handle a larger number of unknowns at each iteration. We use the semi-implicit stepping scheme only for L^2 gradient descent in our experiments, and the explicit scheme for Newton velocity.

Line search for the step size. For many implementations of active contour models, the step size τ in the curve update (10) is fixed, and a preset number of iterations are taken with the fixed τ. This is not a very effective approach, as one might miss (overshoot) the minimum with a large τ or take too many iterations with a small τ. A better approach is to use a line search that chooses a step τ guaranteeing energy decrease from a range of τ values. For example, one can select τ that satisfies the Armijo criterion [9]

$$J(\Gamma^{k+1}) < J(\Gamma^k) + \alpha\tau dJ(\Gamma^k; V^k). \tag{13}$$

This criterion is mostly effective in ensuring energy decrease at each step and attaining convergence. But sometimes this monotone line search for the step size might fail; even the smallest τ may appear not to yield energy decrease. This is due to the inconsistency between the computed energy (obtained by summing up pixels) and the computed shape derivative (using the discretized curve and the interpolation of the image). To circumvent this, we pursued a nonmonotone line search strategy [13], which does not require strict energy decrease at each iteration, but rather decrease on average in the recent iterations. We used the following criterion

$$J(\Gamma^{k+1}) < C^k + \alpha\tau^k dJ(\Gamma^k; V^k), \tag{14}$$

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 $\alpha = 10^{-3}$, $\eta = 0.2$ in our experiments. Compared to the monotone strategy (13), the nonmonotone line search sometimes took more iterations to converge, but mostly resulted in fewer energy evaluations and curve updates, hence less computation. Moreover it converged with fewer line search failures.

The stopping criterion. The conventional approach to stop iterations in optimization literature is to track the norm of the gradient and to stop when it is small [9]. This is not as straightforward for a shape optimization implementation involving discretized geometries and real data. One needs to account for imperfections in the data, such as noise, lack of contrast, and limited resolution of the numerical model. Even the choice of the norm for the gradient requires care. Simply setting the stopping tolerance to a fixed value is not successful for this particular problem. Thus we tried to derive a robust stopping criterion starting from the following pointwise condition on the shape gradient $|G(x, \Gamma)| < \varepsilon \delta G(\Gamma)$, where $\varepsilon > 0$ is an absolute tolerance, $\delta G(\Gamma)$ is the difference between the max and min values of the shape gradient among distinct regions and provides a relative scaling. We chose to track the L^2 norm of the shape gradient, for which the threshold tolerance can be derived by integrating the pointwise condition over Γ

$$\|G\|_{L^2(\Gamma)} = \left(\int_\Gamma |G|^2 d\Gamma \right)^{\frac{1}{2}} < \varepsilon \delta G(\Gamma) |\Gamma|^{\frac{1}{2}}. \tag{15}$$

This criterion can be elaborated further for specific region-based energies. For example, for the Chan-Vese energy (1), we have $G(x, \Gamma) = (c_2 - c_1)(I(x) - \frac{1}{2}(c_1 + c_2))$, and

we used $\delta G(\Gamma) \approx \frac{1}{2}|c_2 - c_1|^2$. Figure 2 shows the evolution of the shape gradient and stopping tolerance for a typical segmentation example. The norm of the shape gradient is small at the beginning, large in the middle, small at the end. If we used a small fixed tolerance, the iterations would stop immediately at the beginning of the minimization iterations. If we used a large fixed tolerance, the iterations would not stop for many more unnecessary iterations. Our tolerance (15) is dynamic; it stays small initially, then grows gradually through the iterations.

4 Geometry and Numerical Discretization

For efficiency, we build our numerical framework on a Lagrangian representation of the curves Γ. We find that this enables a faithful discretization of our shape optimization formulation, in particular, of the shape sensitivity relationships and the velocity (from Sections 2 and 3). Most calculations take place on the curves Γ, and we refer to the domains inside and outside Γ, only when we need to compute the region integrals for the shape energies (1), (5). We represent a curve as a list of elements $\{\Gamma_i\}_{i=1}^m$ such that $\Gamma = \bigcup_i \Gamma_i$. An element Γ_i is defined as the line segment connecting node $\boldsymbol{X}_i$ to node $\boldsymbol{X}_{i+1}$. *We do not use a parameterization of the curves.*

The curves can undergo large deformations between iterations, and this can cause the nodes to be scattered unevenly. Still it is crucial to maintain a *good* distribution of the nodes. We do not want the node distribution to be uniform, but rather to be spatially adaptive. We want fewer nodes for flat curves in flat image regions and more nodes in high curvature portions of the curves and in highly varying regions of the image (measured by comparing high and low order quadratures of the image intensity function on the curve elements). The adaptation of node distribution is done via refinement and coarsening procedures and ensures computational efficiency together with good resolution and accuracy. Moreover, to enable topological adaptivity of curves (like a level set model), we implemented intersection detection and curve surgery procedures for splitting and merging of curves. The implementation details of these adaptation procedures are beyond the scope of this paper, and will be elaborated in a separate report.

Discretizing velocity computations. We propose a finite element discretization of the continuous velocity equations from Section 3. For this, we introduce a set of piecewise linear basis functions $\{\phi_i\}_{i=1}^m$ satisfying $\phi_i(X_j) = \delta_{ij}$ (also vector version $\boldsymbol{\phi}_i = (\phi_i, \phi_i)^T$). We expand the variables, such as the velocity V and the geometric quantities $\boldsymbol{X}, \boldsymbol{\kappa}$, in terms of the basis functions: $V = V_j\phi_j$ $(:= \Sigma_j V_j \phi_j)$, $\boldsymbol{X} = \mathbf{X}_j\boldsymbol{\phi}_j$, $\boldsymbol{\kappa} = \mathbf{K}_j\boldsymbol{\phi}_j$, and we work with the corresponding coefficient vectors $\mathbf{X}, \mathbf{V}, \mathbf{K}$ respectively. This enables us to write the discretized versions of the key integral equations:

$$\begin{aligned}
&\langle \boldsymbol{\kappa}, \boldsymbol{\phi}\rangle = \langle \nabla_\Gamma \boldsymbol{X}, \nabla_\Gamma \boldsymbol{\phi}\rangle, && \Rightarrow \boldsymbol{\kappa}_j \langle \boldsymbol{\phi}_j, \boldsymbol{\phi}_i\rangle = \boldsymbol{X}_j \langle \nabla_\Gamma \boldsymbol{\phi}_j, \nabla_\Gamma \boldsymbol{\phi}_i\rangle, && \Rightarrow \mathbf{MK} = \mathbf{AX},\\
&b(V, \phi) = -\langle G(\Gamma), \phi\rangle, && \Rightarrow V_j\, b(\phi_j, \phi_i) = -\langle G(\Gamma), \phi_i\rangle, && \Rightarrow B\mathbf{V} = -\mathbf{G},
\end{aligned}$$

where we have introduced the following matrices and vectors

$$\begin{gathered}
A_{ij} := \langle \nabla_\Gamma \phi_i, \nabla_\Gamma \phi_j\rangle, \quad \boldsymbol{A}_{ij} := A_{ij}\boldsymbol{Id}, \quad M_{ij} := \langle \phi_i, \phi_j\rangle, \quad \boldsymbol{M}_{ij} := M_{ij}\boldsymbol{Id},\\
M_{\beta,ij} := \langle \beta\phi_i, \phi_j\rangle, \quad B_{ij} := b(\phi_i, \phi_j), \quad \mathbf{G}_i := \langle G(\Gamma), \phi_i\rangle.
\end{gathered}$$

We have $B = M$ for L^2 velocity, and $B = \mu A + M_\beta$ for Newton velocity. We emphasize that all the relevant operations with these matrices (matrix inversion, matrix-vector products) have linear time complexity. These matrices consist of circular tridiagonal blocks (corresponding to individual curves), and they can be inverted using the Thomas algorithm, e.g. for computing velocity $\mathbf{V} = -B^{-1}\mathbf{G}$ or curvature $\mathbf{K} = \mathbf{M}^{-1}\mathbf{A}\mathbf{X}$.

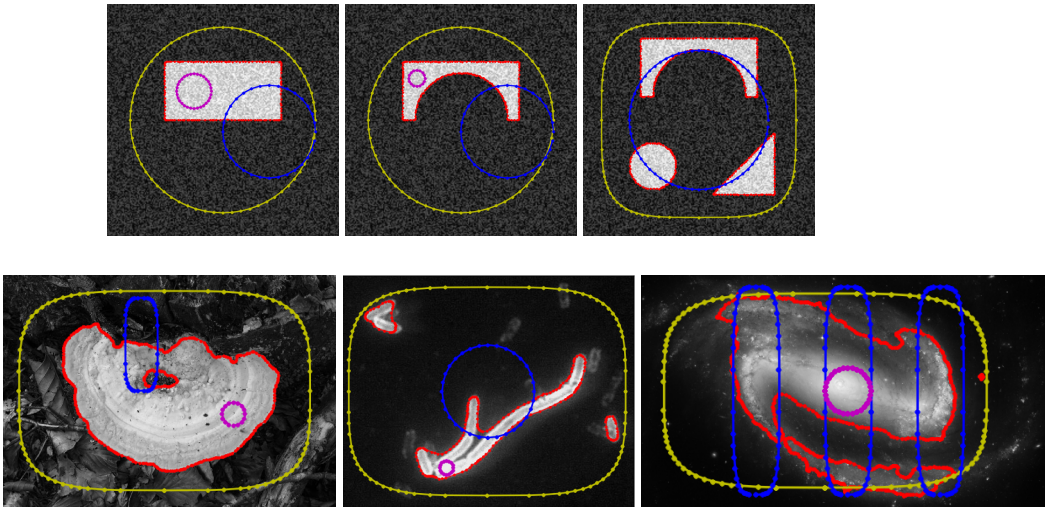

Fig. 1. Example images and initial curves used for testing. Top row: synthetic images (rectangle, concave, multiple). Bottom row: real images (fungus, bacteria, galaxy). Three different starting curves Γ_{in} (magenta), Γ_{out} (yellow), Γ_{part} (blue) used for each image.

5 Numerical Examples

In this section, we demonstrate that the Newton's method, which takes advantage of the shape Hessian, produces the segmentation in significantly less computation than the L^2 gradient descent. For this, we used a set of synthetic image examples (with increasing complexity of the regions), and a set of real images (see Figure 1). We set the length penalty in the energy (1) to $\mu = 10^{-3}$. Also to discourage curves extending beyond the image domain D, we added another weighted length term $\int_\Gamma (1 - \mathbb{1}_D) d\Gamma$, which has an effect only on the parts Γ outside D. We started the minimization iterations with different initial curve configurations: $\Gamma^0_{in}, \Gamma^0_{out}, \Gamma^0_{part}$, curves that were inside, outside or partially overlapping the regions of interest respectively. The results of these experiments are given in Table 1, where we report the number of energy evaluations and the number of iterations. As we used line search to determine the right step size for an iteration in all methods, the number of energy evaluations was a better measure of computational cost than the number of iterations, because it also included steps or iterations that were tested by line search, but were not taken. We stopped the iterations

when the L^2 norm of the shape gradient was less than the scaled tolerance (15) (with $\varepsilon = 0.5$) or when line search failed, namely, it returned the smallest step size, which did not give apparent energy decrease. In most of these cases, the segmentation was already complete when line search failure happened. We made the following observations:

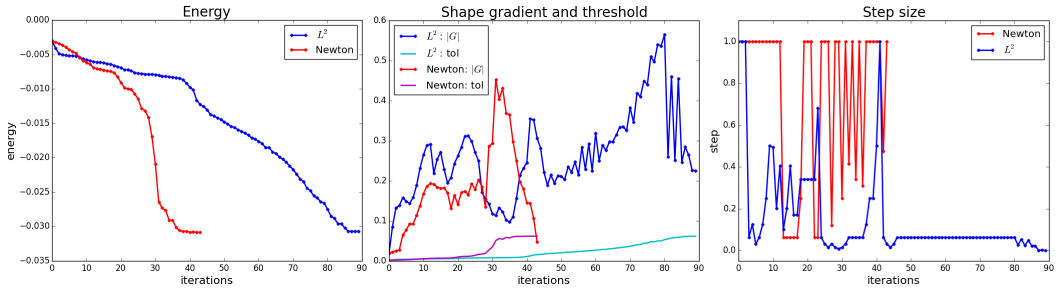

Fig. 2. The energy, shape gradient, scaled stopping tolerance and step size values for all the iterations of minimization (synthetic example with concave region).

- L^2 *velocity:* Normalizing the L^2 velocity (dividing by $\|V\|_2$) before line search improved its performance, especially in cases of poor contrast or low intensity when the L^2 velocity was slow (recall $V = -(c_2 - c_1)(I(x) - 0.5(c_1 + c_2))$).
- *Test step size:* We used $\tau = 1$ as an initial step size for line search in the Newton's method, whereas using twice the previous step size worked best for the L^2 gradient descent. The former takes full steps as much as it can all along, whereas the latter reduces τ when the curve is close to the minimum.
- *Monotone vs Nonmonotone:* The choice of nonmonotone line search using criterion (14) did not make a big difference for the L^2 gradient descent. For Newton's velocity, it reduced the number of energy evaluations on average and nearly eliminated line search failures. This is due to the fact the nonmonotone criterion does not impose strict energy decrease.
- *Newton velocity:* Required fewer energy evaluations than the L^2 velocity to reach the minimum. Especially, if some part of the curve is already on the region boundary, the Newton's method quickly pulls the rest of the curve to the region boundary.

Overall we observed that the Newton velocity consistently outperformed L^2 gradient descent. We should nonetheless add that, although our results were in favor of the Newton's method, one can concoct examples of images and initial curves, for which the performance difference is not as apparent.

Acknowledgment

This work was supported by the NIST grant 70NANB13H018.

	rectangle			concave			multiple	
	Γ_{in}	Γ_{out}	Γ_{part}	Γ_{in}	Γ_{out}	Γ_{part}	Γ_{out}	Γ_{part}
L^2 (m)	35 (16)	47 (23!)	33 (19!)	35 (18)	99 (58)	40 (19)	54 (35)	54 (26!)
L^2 (n)	46 (20!)	46 (24)	40 (19)	51 (31)	83 (51!)	45 (20!)	59 (33!)	56 (27!)
Nwt (m)	26 (19!)	**23 (17)**	**23 (17)**	40 (19!)	38 (32)	**26 (17)**	43 (36)	32 (22)
Nwt (n)	**22 (20)**	25 (19)	28 (21)	**27 (19)**	**37 (34)**	27 (18)	**40 (33)**	**28 (22)**

	fungus			bacteria			galaxy		
	Γ_{in}	Γ_{out}	Γ_{part}	Γ_{in}	Γ_{out}	Γ_{part}	Γ_{in}	Γ_{out}	Γ_{part}
L^2 (m)	44 (21)	63 (34!X)	49 (24!)	74 (40)	117 (60!X)	86 (47)	46 (26)	33 (20!X)	60 (38)
L^2 (n)	52 (29!)	54 (31!X)	35 (21!)	64 (35)	MAX!	64 (35)	54 (29!)	83 (46!)	56 (36)
Nwt (m)	**17 (14)**	**22 (21)**	**36 (15!)**	**27 (25)**	**33 (28)**	**12 (11)**	**26 (18)**	32 (25!)	**26 (22)**
Nwt (n)	**17 (14)**	**22 (21)**	60 (58)	**27 (25)**	**33 (28)**	**12 (11)**	34 (26)	**29 (28)**	37 (34)

Table 1. Performance of L^2 vs Newton velocity. The test images were segmented using both velocities with monotone (m) line search criterion (13) and nonmonotone (n) criterion (14). Some of these experiments terminated due to line search failures (rather than fulfilment of the stopping criterion (15)), albeit with successful segmentation (marked with !). A few terminated prematurely with line search failure (marked with !X). One did not finish in the maximum (100) number of iterations (marked with MAX). We report the number of energy evaluations for each experiment (and the iteration numbers in parantheses). The best results are highlighted in bold.

References

1. G. Aubert, M. Barlaud, O. Faugeras, and S. Jehan-Besson. Image segmentation using active contours: calculus of variations or shape gradients? *SIAM J. Appl. Math.*, 63(6), 2003.
2. T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.
3. G. Charpiat, P. Maurel, J.-P. Pons, R. Keriven, and O. Faugeras. Generalized gradients: Priors on minimization flows. *International Journal of Computer Vision*, 73:325–344, 2007.
4. D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *IJCV*, 72:195–215, 2007.
5. 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.
6. G. Doğan, P. Morin, and R. H. Nochetto. A variational shape optimization approach for image segmentation with a Mumford-Shah functional. *SIAM J. Sci. Comp.*, 30(6), 2008.
7. M. Hintermüller and W. Ring. A second order shape optimization approach for image segmentation. *SIAM J. Appl. Math.*, 64(2):442–467, 2003/04.
8. 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), 2004.
9. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.
10. N. Paragios and R. Deriche. Geodesic active regions: A new framework to deal with frame partition problems in computer vision. *J. Vis. Commun. Image Represent.*, 13(1-2), 2002.
11. J. Sokołowski and J.-P. Zolésio. *Introduction to Shape Optimization*, volume 16 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1992.
12. G. Sundaramoorthi, A. Yezzi, and A. Mennucci. Sobolev active contours. *International Journal of Computer Vision*, 73:345–366, 2007. 10.1007/s11263-006-0635-2.
13. 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.