To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–24

Self-Consistent Gradient Flow for Shape Optimisation

D. Kraft*

University of Graz, Institute of Mathematics, NAWI Graz,
Universitätsplatz 3, 8010 Graz, Austria

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

We present a model for image segmentation and describe a gradient-descent method for level-set based shape optimisation. It is commonly known that gradient-descent methods converge slowly due to zig-zag movement. This can also be observed for our problem, especially when sharp edges are present in the image. We interpret this in our specific context to gain a better understanding of the involved difficulties. One way to overcome slow convergence is the use of second-order methods. For our situation, they require derivatives of the potentially noisy image data and are thus undesirable. Hence, we propose a new method that can be interpreted as a self-consistent gradient flow and does not need any derivatives of the image data. It works very well in practice and leads to a far more efficient optimisation algorithm. A related idea can also be used to describe the mean-curvature flow of a mean-convex surface. For this, we formulate a mean-curvature Eikonal equation, which allows a numerical propagation of the mean-curvature flow of a surface without explicit time stepping.

Keywords: Level-Set Method; Shape Optimisation; Image Segmentation; Gradient Flow; Topological Derivative; Mean-Curvature Flow

AMS Subject Classification: 49Q10; 65D18; 65J22; 65K10

1. Introduction

Since its introduction in [22], the *level-set method* of Osher and Sethian has become very popular for describing evolving geometries for shape optimisation and free-boundary problems. The basic idea is the following:

DEFINITION 1 Let a continuous *level-set function* $\phi\colon \mathbb{R}^n \times [0,\infty) \to \mathbb{R}$ be given. For any time $t \geq 0$, this function describes the evolving sets

$$\Omega_t = \{x \in \mathbb{R}^n \mid \phi(x,t) < 0\}\,, \quad \Gamma_t = \{x \in \mathbb{R}^n \mid \phi(x,t) = 0\}\,.$$

Note that the level-set method does not require *any* regularity of the domains at all: A possible level-set function for an arbitrary open set $\Omega \subset \mathbb{R}^n$ can be constructed by using, for instance, its *signed distance function*

$$\mathrm{sd}_\Omega(x) = \begin{cases} \operatorname{dist}(x, \partial\Omega) & \text{for } x \notin \Omega, \\ -\operatorname{dist}(x, \partial\Omega) & \text{for } x \in \Omega. \end{cases}$$

*Corresponding author. Email: daniel.kraft@uni-graz.at

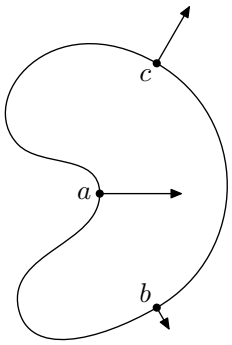


Figure 1.: The scalar speed method in normal direction. In the shown situation, the signs relate as $F(a) < 0 < F(b) < F(c)$.

Here, we use the notation

$$\operatorname{dist}(x, \partial\Omega) = \inf_{y \in \partial\Omega} |x - y| = \min_{y \in \partial\Omega} |x - y|$$

and assume that $\partial\Omega \neq \emptyset$. It is well-known that the signed distance function has Lipschitz constant one (see, for instance, Theorem 2.1 on page 268 of [8]). Hence, even requiring Lipschitz continuity of the level-set function ϕ does not imply any regularity of Ω_t. This makes the level-set method attractive for situations where it is advantageous to allow non-smooth domains.

In the present work, we want to describe and analyse methods for *shape optimisation* based on level sets. Let us refer to [1] for a general functional-analytic framework for shape optimisation with level sets and to [24] for an early work that captures already the basic ideas in a non-rigorous fashion. In order to perform optimisation, we need, of course, a way to describe *changes* to shapes. We do this with a *scalar speed method*: Over time, the boundary of the evolving domain Ω_t is transformed *in normal direction* according to a given speed field $F\colon \mathbb{R}^n \to \mathbb{R}$. Positive speed corresponds to outward movement and a growing domain, while a negative value of F leads to local shrinking of the domain. This is illustrated in Figure 1. Note that this approach is in contrast to the classical methods described, for instance, in [8] and [26]. There, one usually considers a vector-valued *velocity field*. We, on the other hand, fix the direction of movement as the normal to the boundary because a tangential movement has no geometrical meaning. This speed method complements the level-set method very well, as it allows quite irregular evolutions. It is, in particular, possible to describe changes in topology (as we will see later in Figure 3). The effect of this transformation on the level-set function can be described by propagating ϕ in time with the *level-set equation*

$$\dot{\phi} + F(x)\,|\nabla\phi| = 0, \quad \phi(\cdot, 0) = \phi_0. \tag{1}$$

Here, the initial level-set function ϕ_0 describes the initial domain Ω_0. Let us assume throughout this work that $\overline{\Omega_0} = \Gamma_0 \cup \Omega_0$, i. e., that Γ_0 has empty interior. We assume, furthermore, that ϕ_0 and F are Lipschitz continuous. This is not restrictive in practice. Under these assumptions, one can establish the existence of a unique *viscosity solution* for the level-set equation (1). This is well-known, let us just refer to [5], [6] and [11]. Furthermore, a thorough discussion of the underlying concepts for the specific situation of (1) can also be found in Chapter 2 of [19].

For our discussion below, we follow the approach described in [16] and Chapter 3 of [19]. In particular, it was shown there that the evolving sets Ω_t and Γ_t can be expressed in terms of the *F-induced distance* to the initial geometry:

DEFINITION 2 For $x, y \in \mathbb{R}^n$ and a continuous path $\xi \in W^{1,\infty}([0,1], \mathbb{R}^n)$ with $\xi(0) = x$ and $\xi(1) = y$, we define its length as

$$l(\xi) = \int_0^1 \frac{|\xi'(t)|}{F(\xi(t))}\, dt.$$

The set of all such paths is denoted by $X_{\text{ad}}(x, y)$. Furthermore, we define the *F-induced distance* by

$$d(x, y) = \inf_{\xi \in X_{\text{ad}}(x,y)} l(\xi).$$

The distance to the initial domain Ω_0 is then given by

$$D(x) = \begin{cases} \inf_{y \in \Gamma_0 \cup \Omega_0} d(x, y) & \text{if } F(x) > 0, \\ -\inf_{y \in \mathbb{R}^n \setminus \Omega_0} d(x, y) & \text{if } F(x) < 0. \end{cases}$$

We leave $D(x)$ undefined if $F(x) = 0$.

With these notions, one can interpret the level-set equation (1) as the Hamilton-Jacobi-Bellman equation of a control problem. This allows to find a Hopf-Lax representation formula for ϕ and the evolving domains. Intuitively speaking (and ignoring the sign for a moment), $D(x)$ is the time it takes the evolving boundary Γ_t to arrive at some point $x \in \mathbb{R}^n$. Thus, it does not come as a surprise that one can show the following formulas:

THEOREM 1 *Let F be Lipschitz continuous and have compact support. Assume that $\overline{\Omega_0} = \Gamma_0 \cup \Omega_0$. Then*

$$\begin{aligned} \Omega_t &= \left\{ x \in \Omega^+ \;\middle|\; D(x) < t \right\} \cup (\Omega_0 \cap \Omega^z) \cup \left\{ x \in \Omega^- \;\middle|\; D(x) < -t \right\}, \\ \Gamma_t &= \left\{ x \in \Omega^+ \;\middle|\; D(x) = t \right\} \cup (\Gamma_0 \cap \Omega^z) \cup \left\{ x \in \Omega^- \;\middle|\; D(x) = -t \right\}, \\ \Gamma_t \cup \Omega_t &= \left\{ x \in \Omega^+ \;\middle|\; D(x) \le t \right\} \cup ((\Gamma_0 \cup \Omega_0) \cap \Omega^z) \cup \left\{ x \in \Omega^- \;\middle|\; D(x) \le -t \right\} \end{aligned}$$

for all $t > 0$. Here, $\Omega^\pm$ are the (open) sets where F is positive and negative, respectively, and $\Omega^z = \{x \in \mathbb{R}^n \mid F(x) = 0\}$.

Proof. See Corollary 5 in [19]. ■

An important consequence is that the evolving domain Ω_t grows where F is positive, shrinks where F is negative and is stationary where $F = 0$. This was already mentioned above as intuitive expectation. Theorem 1 shows now that this follows, indeed, rigorously for the time evolution of the level-set equation (1). Furthermore, one can also apply the representation formula of Theorem 1 for *shape-sensitivity analysis* of domain functionals. In particular, let $f \in L^1_{\text{loc}}(\mathbb{R}^n)$ and consider

$$j(\Omega) = \int_\Omega f\, dx. \tag{2}$$

The co-area formula together with Theorem 1 implies then the following statement:

THEOREM 2 *Assume that Theorem 1 holds for Ω_t and consider j as per (2). Then*

$$j(\Omega_t) = j(\Omega_0) + \int_0^t \int_{\Gamma_s} Ff \, d\sigma \, ds$$

holds for all $t \geq 0$.

Proof. See Theorem 10 in [19]. ■

It follows immediately from Theorem 2 that $t \mapsto j(\Omega_t)$ is absolutely continuous. By the Lebesgue differentiation theorem, this function is differentiable for almost all $t \geq 0$ with the *Eulerian shape derivative*

$$j'(t) = \lim_{h \to 0^+} \frac{j(\Omega_{t+h}) - j(\Omega_t)}{h} = \int_{\Gamma_t} Ff \, d\sigma.$$

Note that this shape derivative matches the classical results (see, for instance, Section 2.11 of [26]). It can, however, be derived without requiring a smooth boundary of the domain Ω_t. One can even extend this result to *shape-dependent integrands*: Assume that our functional has now the form

$$J(\Omega) = \int_\Omega f(x, \Omega) \, dx, \tag{3}$$

where f itself depends on Ω in some way. Let us assume that $t \mapsto f(x, \Omega_t)$ has a shape derivative $f'(x, \Omega_t)$ for any fixed x. In this situation, we can follow Corollary 7 in [19] to conclude that the *total shape differential* of J, corresponding to the shape derivative in direction of the speed field F, is

$$J'(t) = dJ(\Omega_t; F) = \int_{\Gamma_t} Ff(x, \Omega_t) \, d\sigma + \int_{\Omega_t} f'(x, \Omega_t) \, dx. \tag{4}$$

If f depends on Ω via one or several shape-dependent quantities, one can use this formula to justify a chain rule.

In this paper, we develop numerical methods for shape optimisation based on these results. Section 2 introduces a model for image segmentation that we use for analysing and demonstrating our methods. We discuss a simple gradient-descent method in Section 3. Since this method is inefficient due to the well-known "zig-zag behaviour" of gradient methods, we introduce a new idea in Section 4 and Section 5. This leads to a method that is inherently based on the level-set framework described above. It can be interpreted as a *self-consistent gradient flow* and is much more efficient than gradient descent for our model problem. One can also include information from topological derivatives into this method, which we do in Section 6. In the final Section 7, we discuss how the same approach could be extended to functionals with boundary terms and describe, in particular, the relation of our ideas to the *mean-curvature flow* of some initial domain Ω_0. Let us also remark that our numerical code related to the described level-set framework (and, in particular, shape evolutions based on Theorem 1) is released as free software in an extension package [18] for GNU Octave [9].

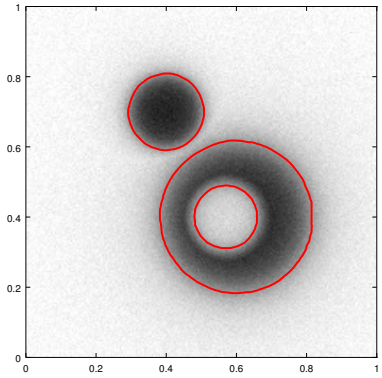


Figure 2.: Our goal of the example optimisation problem considered throughout this paper is to find a shape matching one segment of a given grey-scale image. Here, the solution is shown with the red contour. It consists of the dark ring and disc.

2. A Model for Image Segmentation

Throughout this paper, we will discuss the developed optimisation methods based on a model for image segmentation. Our goal with this optimisation problem is shown in Figure 2: Given a grey-scale image, we want to identify the shape of a *segment*, i. e., a region with approximately homogeneous intensity, of the image. We do this by minimising the following shape functional:

$$J(\Omega) = \int_\Omega (u(x) - \overline{u})^2 \, dx - 2\gamma \cdot \sigma \cdot \operatorname{vol}(\Omega) \tag{5}$$

Here, $u\colon D \to I \subset \mathbb{R}$ is the grey-scale image. Typically, we consider $D = [0,1]^2$ and $I = [0,1]$. The set $\Omega \subset D$ is the shape we are looking for, which should be a part of the domain D on which the image u attains an approximately constant intensity. The quantities $\overline{u}$ and σ in (5) are the mean intensity and the standard deviation of the image over Ω, i. e.,

$$\overline{u} = \frac{1}{\operatorname{vol}(\Omega)} \int_\Omega u(x)\, dx, \quad \sigma^2 = \frac{1}{\operatorname{vol}(\Omega)} \int_\Omega (u(x) - \overline{u})^2 \, dx.$$

The variable $\gamma > 0$ is a constant parameter. Thus, our goal is to minimise (5) over all possible open sets $\Omega \subset D$. We call D the *hold-all domain* and assume it to be bounded.

Note that this approach is slightly different from the usual meaning of *image segmentation*. We are only trying to identify a single segment's shape, which is a problem suited very well for demonstrating our general optimisation framework. In the literature, image segmentation usually means to find *all* segments of the image. In other words, one looks not just for $\Omega \subset D$, but instead for a disjoint decomposition of D into $\Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_N$. One can, however, extend the model in (5) to include corresponding terms for $D \setminus \Omega$ as well. This leads to a so-called *two-phase image segmentation*. It is straight-forward to adapt the techniques developed below to this situation. Furthermore, using *two* sets Ω_1 and Ω_2 as optimisation variables, one can even characterise *four* different phases. With the Four-Colour Theorem, this is enough to describe all possible (regular) image

segmentations. Again, our theory and methods can be extended to this case as well. Such *multi-phase* image-segmentation approaches were introduced in [27].

We would also like to point out that our model (5) is not meant to be a state-of-the-art method for image segmentation. Instead, it is just a convenient example problem for the discussion of the general optimisation framework we want to present. Let us now briefly interpret the chosen cost functional J: The first term in (5) is a data-fitting term that penalises segments which do not have a nearly uniform intensity. This term is already present in the classical Chan-Vese model for image segmentation described in [3]. Instead of approximating the segment with a constant intensity $\overline{u}$, one can also use higher-order polynomials. See, for instance, [4]. This is, again, a straight-forward extension of the method discussed in the following. Probably more interesting (and non-standard), however, is the second term in (5): Since it prefers Ω to become larger due to the negative sign, it creates a *balloon force* that prevents the segment Ω from collapsing to the empty set. The weight of this force is given by the parameter γ and, more importantly, by the standard deviation σ of the image over the segment. In other words, if the constant intensity $\overline{u}$ is a poor fit for the actual image over Ω, we increase the force. This may happen, for instance, if the image is very noisy. In these cases, the stronger balloon force is needed to overcome the penalisation imposed by the data-fitting term. When we have derived the shape derivative of J below, we will see much better how these two competing forces interact.

Our model (5) is covered completely by the shape calculus described above. Thus, we can calculate the shape derivatives according to (4). Let F be some fixed speed field. For simplicity, we denote the shape derivative in direction F again just by a prime. Recall our shape-dependent quantities:

$$\operatorname{vol}(\Omega) = \int_\Omega dx, \quad \overline{u} = \frac{1}{\operatorname{vol}(\Omega)} \int_\Omega u\,dx, \quad \operatorname{Var}(u) = \frac{1}{\operatorname{vol}(\Omega)} \int_\Omega (u-\overline{u})^2\,dx, \quad \sigma = \sqrt{\operatorname{Var}(u)}$$

Thus, the shape derivatives are:

$$\begin{aligned}
\operatorname{vol}(\Omega)' &= \int_\Gamma F\,d\sigma, \\
\overline{u}' &= \frac{1}{\operatorname{vol}(\Omega)} \int_\Gamma uF\,d\sigma - \frac{\operatorname{vol}(\Omega)'}{\operatorname{vol}(\Omega)^2} \int_\Omega u\,dx = \frac{1}{\operatorname{vol}(\Omega)} \int_\Gamma (u-\overline{u})F\,d\sigma, \\
\left(\int_\Omega (u-\overline{u})^2\,dx\right)' &= \int_\Gamma (u-\overline{u})^2 F\,d\sigma + 2\overline{u}' \int_\Omega (\overline{u}-u)\,dx = \int_\Gamma (u-\overline{u})^2 F\,d\sigma, \\
\operatorname{Var}(u)' &= \frac{1}{\operatorname{vol}(\Omega)} \int_\Gamma (u-\overline{u})^2 F\,d\sigma - \frac{\operatorname{vol}(\Omega)'}{\operatorname{vol}(\Omega)^2} \int_\Omega (u-\overline{u})^2\,dx \\
&= \frac{1}{\operatorname{vol}(\Omega)} \int_\Gamma \left((u-\overline{u})^2 - \operatorname{Var}(u)\right) F\,d\sigma, \\
\sigma' &= \frac{\operatorname{Var}(u)'}{2\sqrt{\operatorname{Var}(u)}} = \frac{\operatorname{Var}(u)'}{2\sigma} = \frac{1}{2\cdot\operatorname{vol}(\Omega)} \int_\Gamma \left(\frac{(u-\overline{u})^2}{\sigma} - \sigma\right) F\,d\sigma
\end{aligned}$$

Using these results, we can now also compute the shape derivative of J in direction F:

$$\begin{aligned} dJ(\Omega; F) &= \int_\Gamma (u-\overline{u})^2 F \, d\sigma - 2\gamma\sigma \int_\Gamma F \, d\sigma - \gamma \int_\Gamma \left(\frac{(u-\overline{u})^2}{\sigma} - \sigma \right) F \, d\sigma \\ &= \int_\Gamma \left((u-\overline{u})^2 \left(1 - \frac{\gamma}{\sigma}\right) - \gamma\sigma \right) F \, d\sigma \end{aligned} \tag{6}$$

Note that $dJ(\Omega; \cdot)$, interpreted as functional operating on the speed field F, is supported on the *boundary* Γ of our domain Ω. This corresponds to the well-known *Hadamard-Zolésio structure theorem*, see Theorem 3.6 on page 479 of [8]. This interpretation of the shape derivative as a functional on speed fields will be important for the gradient-descent method described in the next Section 3.

Before we come to this point, let us now also give a rough interpretation of (6): Assume that $x \in \Gamma$ and that $F(x) > 0$. This means that propagation of Ω in direction F adds x to the domain. For this situation, the integrand's value at x tells us how J changes when a neighbourhood of x is included into the image segment. (This is related to *topological derivatives*, which appear naturally for our example in the shape differential (6). See the discussion in Section 6.) In particular, the *sign* of the integrand at x tells us whether or not x "should" be included in the segment. This leads to the *inclusion criterion*

$$(u(x)-\overline{u})^2 \left(1 - \frac{\gamma}{\sigma}\right) < \gamma\sigma. \tag{7}$$

In the case $\gamma < \sigma$, this condition is fulfilled if and only if $u(x)$ is "close enough" to the current segment intensity $\overline{u}$. The right-hand side (proportional to σ) defines the threshold that tells us *how close* that actually is. The larger the variance of pixels in the segment already is, the higher is also the tolerance for adding new pixels. This corresponds to our earlier interpretation of the balloon force generated by the second term in (5). If $\gamma \geq \sigma$, the inclusion criterion (7) is always satisfied. In this case, the balloon force is so strong that the tracking term in (5) can never counterbalance it, no matter in how bad a way the pixel $u(x)$ fits to the current segment.

Note that the criterion (7) really compares the *standard deviation* to the *quadratic* error term $(u(x)-\overline{u})^2$. Intuitively, it would make much more sense to compare the quadratic error to, say, the *variance*. One can also construct cost functionals whose derivatives produce such a criterion. They are, however, less interesting to consider as example models. Furthermore, the draft paper [13] also proposes a pixelwise condition similar to (7) in an ad-hoc way to define a post-processing step. It was found there empirically that the standard deviation gives, indeed, much better results than using the variance in the threshold condition.

3. The Gradient-Descent Method

As we have just seen, we can compute the directional derivative $dJ(\Omega; F)$ of the cost functional (5) in direction of some speed field F. We would like to use this derivative to define a *steepest-descent direction*. When this is done, one can implement a standard gradient-descent method. See, for instance, Chapter 3 of [21] for a general discussion in the finite-dimensional situation. A remaining difficulty, however, is that the shape derivative (6) is supported on the boundary Γ. We need, on the other hand, a speed field defined on the *hold-all domain* D in order to define a descent direction. Thus, one needs

a suitable method to extend the integrand in (6) from Γ onto D. This can be done in various ways, let us refer to the discussion given in Section 6.2 of [19] for more details. For the purpose of this paper, we focus on the following method: One can interpret $dJ(\Omega;\cdot)$ as a continuous linear functional operating on speed fields F from some Hilbert space $\mathcal{H}$. Using the Riesz representation theorem, we can then construct a speed field $G \in \mathcal{H}$ that corresponds to the shape derivative. We call this G *shape gradient* (in contrast to the *shape derivative* $dJ(\Omega;\cdot)$) and use $-G$ as descent speed field. (This distinction between derivative and gradient is not made by all authors, but we think it helps to clarify the situation. See also part (iv) of Remark 2.34 in [14].) The choice of the concrete function space $\mathcal{H}$ influences the resulting descent method, since it can encode information about desired smoothing and even things like geometric constraints. We refer to [15] for a comparison of various spaces. In the following, we always choose $\mathcal{H} = H^1(D)$, as this space performs well in practice. In particular, the shape gradient G can then be computed by solving the variational problem

$$\int_D \left(FG + \beta \langle \nabla F, \nabla G \rangle\right) dx = dJ(\Omega; F), \tag{8}$$

which must be satisfied for all $F \in \mathcal{H} = H^1(D)$. The parameter $\beta > 0$ is a weighting factor that can be used to tune the smoothing properties of the inner product we use. It is often beneficial for the descent method to choose $\beta \ll 1$, as we have demonstrated in [17]. We use $\beta = 10^{-2}$ for the numerical results in this paper. The variational problem (8) is solved numerically with standard finite-element techniques.

Based on descent directions constructed in this way, it is now straight-forward to implement a gradient-descent method. We employ a standard line search based on the Armijo rule, see Section 3.1 of [21]. More details can be found in Section 6.3 of [19]. Numerical examples of this gradient-descent method applied to our image-segmentation model are shown in Figure 3. The input images are artificially created and have noise added. They are plotted together with the initial (blue curve) and final (red) shapes in the left column of Figure 3. Note that a change in topology happens when the segment forms the ring in Figure 3a. This does not disturb the method at all. The plots on the right show how the cost and gradient norm (blue and red) decrease with the descent iterations. The cost is relative to an "exact" value that was computed for the same noisy images by using the gradient descent starting from an "informed guess" for the initial shape. In particular, the initial segment was chosen as the shape used when generating the image itself. The gradient norm is the H^1-norm of the shape gradient G that solves (8). The green dots show the accepted step length t satisfying the Armijo rule at each iteration. We enforce a minimum step length of $t_{\min} = 10^{-3}$. Around iteration 50, this limit becomes active in Figure 3b since the final shape is already attained and the cost and gradient norm no longer decrease significantly.

While the method works well for the image in the upper row, note that this is no longer the case for the image with *sharp edges* shown in Figure 3c. Even though the descent in Figure 3d runs over many more iterations than the one of Figure 3b, the descent gets stuck very quickly and hits the step-size limit at $t = t_{\min}$ without actually reaching the final solution. The lower line of green dots in Figure 3d is $t_{\min}$, as before. The second line above it is at $2t_{\min}$, which is the first step size tried after a step with $t_{\min}$ has been taken. In many iterations, the enforced step length makes the situation worse, so that the following step is immediately accepted and "corrects" this again. This leads to these distinct *two* lines of green dots. In order to understand why the gradient descent performs so poorly with sharp edges, it is useful to consider the shape gradients

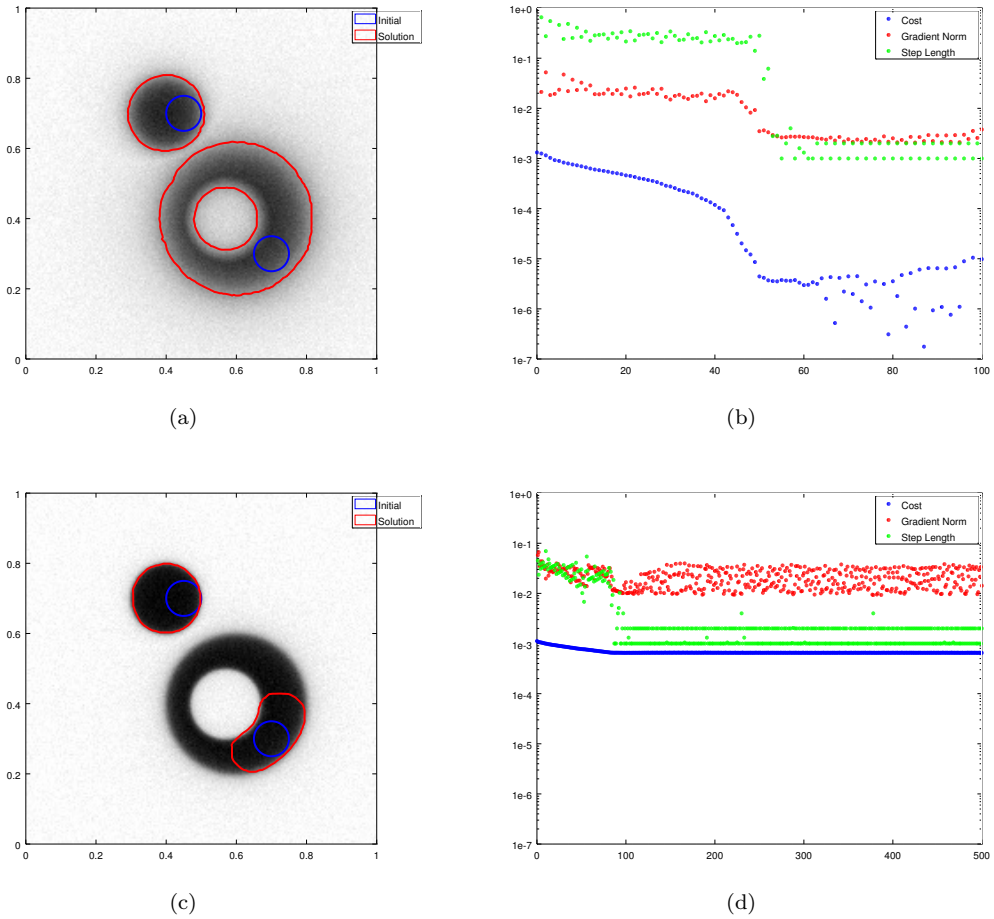
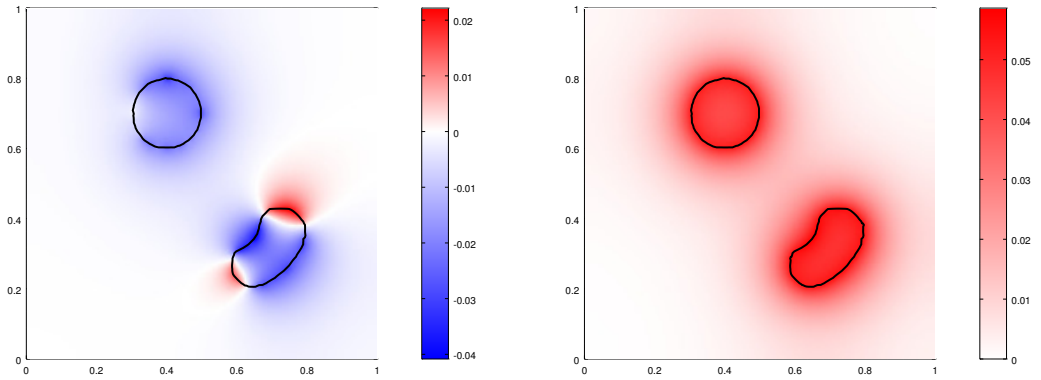


Figure 3.: Gradient descent for image segmentation. The left column shows the input images overlaid by the initial (blue) and resulting (red) shapes. The evolution of the cost (blue), gradient norm (red) and line-search step length (green) is shown on the right.

that occur when the descent is stuck. The speed fields computed for two consecutive iterations are shown in Figure 4. If one looks at even more consecutive steps, one finds that the whole iteration sequence is a back-and-forth between situations similar to these two. In particular, where the current shape is already at the image's edge, the boundary is repeatedly moved over the edge and back. This can be seen clearly in the figure: The speed is negative (blue) in all these regions in Figure 4a and positive (red) in Figure 4b. The front is only moved consistently forward where it is still in the process of closing around the hole. A consequence of this behaviour is that each step is forced to be very short by the line search, so that the descent is not able to progress any further.

In fact, this kind of *zig-zag movement* is a general (and well-known) "feature" of gradient-descent methods. A possible solution for this problem is the use of *second-order methods*. In our particular situation, we would like to weight the speed such that it is decreased accordingly at image edges. This is precisely what a Newton-type method would do, since the Hessian would, roughly speaking, consist of the normal derivatives of the image data. This means that the gradient would be divided by a large value where the shape is close to an edge and by a small value elsewhere. The drawback of such a



(a) Iteration 400 (b) Iteration 401

Figure 4.: Speed fields (as heat maps) at two consecutive iteration steps in the stuck descent of Figure 3d. The current shape is plotted with the black line.

method, however, is that derivatives of the image data are required. For a noisy image, they are not easy to evaluate in a robust way. Because of this, we propose an alternative idea to improve the gradient descent in the following.

4. Self-Consistent Speed Fields

Let us now describe an alternative strategy for the shape optimisation of (5) that avoids the inefficient zig-zag movement of the gradient descent discussed above in Section 3. This is a completely new idea, which can be interpreted as a *gradient flow*. At the heart of this approach, there are two crucial observations about the shape evolution of some initial geometry Ω_0 along a given speed field F:

- Recall that the value $F(x)$ of the speed field at some position $x \in \mathbb{R}^n$ defines the normal speed of movement of the boundary Γ at this point x. This, however, means that the value of $F(x)$ is only significant at the instant t in time when $x \in \Gamma_t$.
- If $F > 0$, then the evolution of Ω is monotone. In particular, our domain always grows. This implies that each point $x \notin \Omega_0$ is reached by the advancing front at a precisely defined, *unique arrival time*. In other words, for each such point there is a unique $t \geq 0$ such that $x \in \Gamma_t$. For all $\tau < t$, it follows that $x \notin \Gamma_\tau \cup \Omega_\tau$. If, on the other hand, $\tau > t$, then $x \in \Omega_\tau$. This t is, in fact, given by the distance $D(x)$ of Definition 2.

Particularly interesting is the following conclusion, which can be drawn by combining both observations: *If we are given some monotone shape evolution, then a single speed field is enough to encode the propagation for all times.* This is even true if the shape evolution is defined in terms of multiple speed fields (e. g., descent steps) or with a time-dependent speed. (The latter situation has not been discussed above, but appears sometimes in a different context.)

Assume that we have an optimisation problem whose shape derivative has the form

$$dJ(\Omega; F) = \int_\Gamma F f(x, \Omega) \, d\sigma \tag{9}$$

for some *shape-dependent* function f. We have already seen in (6) that our image-segmentation problem is of this type. Note, however, that we will have to make some assumptions on f later on for Theorem 3. These assumptions are usually not fulfilled for the image-segmentation problem. Nevertheless, the method still works very well in practice, as we demonstrate in Section 5 below. For the following discussion, let us assume at least $f < 0$ already now. Because we choose the speed field as $-f$ in Definition 3 below, this ensures that the resulting shape evolution is always monotonically growing. The idea can be adapted in a straight-forward way for $f > 0$ and a monotonically shrinking domain as well. If the sign of f is not fixed (which is usually the case in optimisation since we are looking for a zero of the gradient), the approach still works in practice (see Section 5). Now, in order to solve the optimisation problem, we are interested in a speed field that has the following property:

DEFINITION 3 The speed field F is called a *self-consistent gradient flow* if

$$F(x) = -f(x, \Omega_t) \tag{10}$$

for all $t \geq 0$ and $x \in \Gamma_t$. Here, Ω_t is the time evolution of Ω_0 with respect to the speed field F itself.

Clearly, F defined in this way is a descent direction not only at $\Omega = \Omega_0$ but all along the time evolution, i. e., for all $\Omega = \Omega_t$ with $t \geq 0$, as the shape derivative (9) is negative by definition. In fact, we have (in some sense) chosen F to be the negative shape derivative. This is the same idea that was already used in [24]. It motivates the claim that this speed field corresponds, somehow, to a gradient flow. It has to be noted, however, that it does, in general, *not* correspond to the shape gradient introduced in Section 3 as the Riesz representative of the shape derivative. Another important issue is the following: In order to make sense of the condition (10), we already have to know some shape evolution on which Ω_t and Γ_t can be based. In other words, one *already needs a speed field* in order to apply (10)! In fact, for Definition 3 to be fulfilled, this condition needs to hold assuming the shape evolution induced by the speed field F itself. This is the reason for calling it "self-consistent". Thus, (10) can not be used directly to compute such a self-consistent gradient flow. What we can do, however, is to define an *iteration*: Given an initial speed field F_0, we can, indeed, use (10) to define another speed field $F = \psi(F_0)$. The self-consistent gradient flow that we are looking for is then a *fixed point of* ψ. With certain assumptions, we will see in Theorem 3 that such a fixed point exists and the iteration converges to it. The computation of $F = \psi(F_0)$ is depicted in Figure 5: For some fixed $x \notin \Omega_0$, we compute $t = D(x)$. Next, the evolved shape $\Omega(x) = \Omega_t$ at this time is found. It corresponds to the snapshot in the shape evolution when x lies precisely on the advancing front Γ_t. As discussed above, the speed field F_0 is used to compute D and the shape evolution. When this is done, we set

$$\psi(F_0)(x) = F(x) = -f(x, \Omega(x)) \tag{11}$$

according to (10). This means that we evaluate the shape dependence of f for the domain $\Omega(x)$. (Note that the value of F does not matter for $x \in \Omega_0$. Since we assumed a monotonically growing shape, changes to the speed field in Ω_0 will not influence the shape evolution at all.)

Of course, it is not possible in practice to compute all of the evolved shapes $\Omega(x)$ for $x \in \mathbb{R}^n$. Doing so would, roughly speaking, correspond to performing a gradient descent

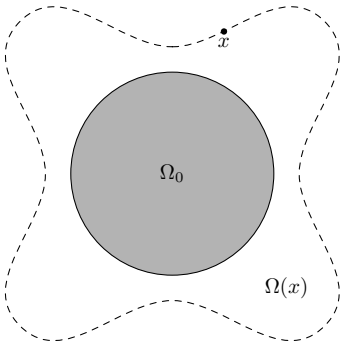
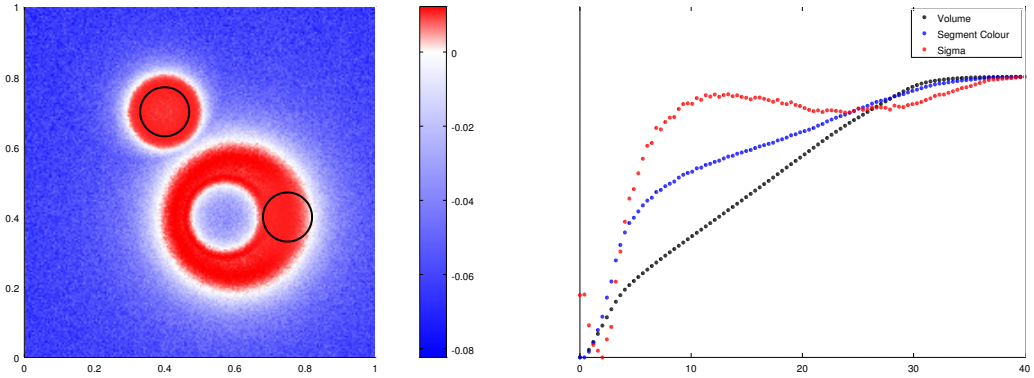


Figure 5.: Computation of $F = \psi(F_0)$ at some point x. The shape $\Omega(x)$, on which the shape-dependent quantities are based, is indicated with the dashed outline.



(a) (b)

Figure 6.: Behaviour of the shape-dependent quantities that appear in $dJ(\Omega; F)$ of the image-segmentation model (see (6)) during a sample time evolution. The initial domain and the applied speed field is shown on the left. An affine transformation has been applied to make the values on the right comparable to each other.

with infinitesimally small time steps. Even the computation for all points of a discrete grid has a prohibitive computational cost and cannot be done practically. One can, however, compute just a handful of snapshots together with the corresponding shape-dependent quantities. The shape dependence at some point x can then be found by interpolating these values for the desired time $D(x)$. The qualitative behaviour of the important shape-dependent quantities in our case is plotted in Figure 6b for a sample time evolution. This shows that they behave "nicely", which means that it is, indeed, justified to apply some suitable interpolation method in practice. Note that this is the case even though neither the speed field nor the image are actually continuous for the example (due to the added noise). In fact, one can show that certain assumptions ensure that the shape-dependent quantities are actually Lipschitz continuous with respect to the evolution time t. See Subsection 7.2.3 of [19] for more details. Thus, there exists also a theoretical justification for the suggested interpolation approach. Hence, we are perfectly able not only to *define* $\psi(F_0)$, but also to *compute* it without too much difficulty.

The proof for existence of a fixed point of ψ is quite technical, so that we refer to Section 7.2 of [19]. The main idea is the following: One can derive a Lipschitz estimate on $\|\psi(F) - \psi(G)\|_\infty$ in terms of the difference $\|F - G\|_\infty$. Furthermore, the Lipschitz constant can be made arbitrarily small by restricting the domain to a band

$$E_t = \{x \in \mathbb{R}^n \mid 0 \leq \mathrm{sd}_{\Omega_0}(x) \leq \underline{F}t\}$$

around the initial geometry Ω_0. Here, $\underline{F} > 0$ is a minimum speed value we enforce on f (see below). Thus, by choosing $t > 0$ sufficiently small, one can always guarantee that ψ is a contraction mapping on a suitable subset of $C(E_t)$. This yields existence of a fixed point with *Banach's fixed-point theorem*. Before we can state this as a theorem, we have to define a certain notion of geometric regularity which is inspired by our work in [20]:

Definition 4 We say that the initial set Ω_0 has *uniform lower density* if there exist $c \in (0, 1)$ and $t_0 > 0$ such that

$$0 < c \leq \frac{\mathrm{vol}\left(B_t\left(x\right) \cap \Omega_0\right)}{\mathrm{vol}\left(B_t\left(x\right)\right)}$$

holds for all $t \in (0, t_0)$ and $x \in \Gamma_0$.

This property forbids, roughly speaking, outward-pointing kinks of the domain Ω_0. The main theorem about the existence of a self-consistent gradient flow is then Theorem 25 in [19], which we state here for convenience:

Theorem 3 *Let the integrand f in (9) depend on Ω only via shape-dependent quantities that can be expressed as domain functionals of the form (3). Furthermore, assume that*

- *f is Lipschitz continuous and $0 < \underline{F} \leq -f \leq \overline{F}$,*
- *Ω_0 has uniform lower density, and*
- *the perimeters of evolved sets Ω_t can be bounded uniformly if t is small enough.*

Then there exists a time $t > 0$ such that ψ has a unique fixed point F^* on E_t. Iteration with ψ converges towards F^* starting from, in particular, any constant speed field with value in $\left[\underline{F}, \overline{F}\right]$. All iterates and F^* itself are Lipschitz continuous. The speed field F^* is a self-consistent gradient flow according to Definition 3 for times up to t.

5. Numerical Realisation as a Multi-Step Procedure

Following up on the description in Section 4, let us now discuss numerical computations based on the idea of self-consistent gradient flows. The numerical evaluation of ψ for a single fixed-point iteration was already described above. In order to build a complete optimisation procedure out of it, let us recall two important difficulties: First, the result of Theorem 3 is only local in nature. To work around this issue, one can, however, apply the theorem repeatedly to layer *multiple* bands around the initial geometry and each other. This allows one to extend the domain where a fixed-point exists, as described in Section 7.3 of [19]. Instead of trying to really find *a single* self-consistent speed field for the whole descent, we perform multiple steps:

(1) Choose a *step length* t_0 that is small enough. For our purposes, it was always enough to consider t_0 simply as a fixed parameter in the algorithm. It could, however, also be chosen based on some convergence criteria if necessary.

(2) Choose some initial speed field F_0. A possible choice is the steepest-descent speed under the assumption that all shape-dependent quantities are constant, i. e.,

$$F_0(x) = -f(x, \Omega_0).$$

(3) Evaluate $\psi(F_0)$ numerically on a suitable neighbourhood of Ω_0. Iterate ψ a few times until a fixed point F^* is found approximately.
(4) Propagate Ω_0 along F^* up to time t_0. Repeat the procedure with the new set as Ω_0.

The second difficulty to keep in mind is that one has to assumed $0 < \underline{F} \leq F \leq \overline{F}$ in order to show Theorem 3. In the context of an optimisation algorithm, we actually expect the speed field to converge to zero when the shape approaches a critical point. Furthermore, it is very restrictive to assume that the shape only grows throughout the optimisation descent. This assumption can be violated even if the initial domain is a subset of the final optimal shape. We will demonstrate this situation on a simple example in Subsection 5.1. In this example, the real gradient flow is such that certain points $x \notin \Omega_0$ come to lie *twice* on the boundary of the evolving domain during the whole propagation: Once when they are temporarily added to the current shape, and once when they are removed from it again later on. It will turn out, however, that these situations pose no problem to the *multi-step optimisation strategy* described above. Each step is, of course, limited to monotone behaviour in the sense that each point can, at most, either be added to or removed from Ω_0 at a single instant in time. During the course of multiple steps, however, the "direction of monotonicity" may change. This is another reason in favour of a multi-step approach with a finite step length t_0 for each iteration. We will see below that it does, indeed, work very well for practical shape optimisation.

5.1 *Demonstration in 1D*

Before we turn our attention back to the image-segmentation problem of Section 3, let us first consider a modified version in only one dimension. This simplified problem allows us to demonstrate some basic properties of our multi-step gradient-flow algorithm. In particular, we want to minimise the cost function

$$J(\Omega) = \int_\Omega (u(x) - u_0)^2 \, dx + \frac{\gamma}{\operatorname{vol}(\Omega)}. \tag{12}$$

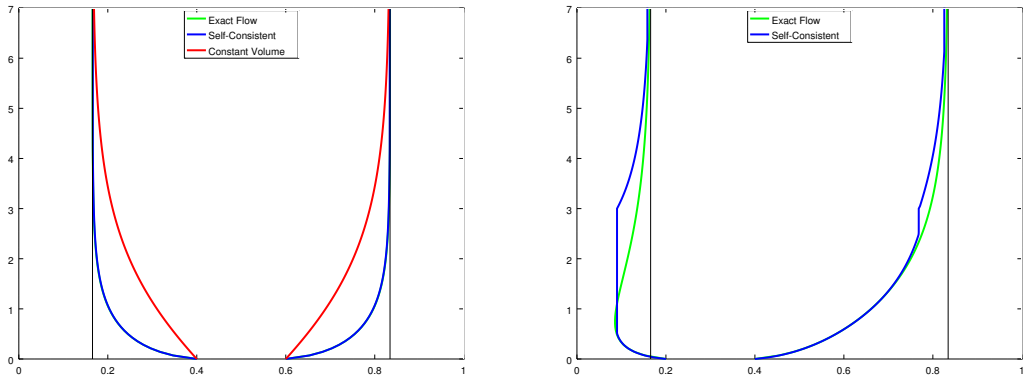
This is similar to (5) of our image-segmentation model, but note that u_0 is assumed to be independent of the shape Ω here. Also the form of the balloon force with $\operatorname{vol}(\Omega)$ is changed. Similar to Section 2, we can easily compute the shape derivative of (12) to be

$$dJ(\Omega; F) = \int_\Gamma \left((u - u_0)^2 - \frac{\gamma}{\operatorname{vol}(\Omega)^2} \right) F \, d\sigma. \tag{13}$$

In other words, moving Γ across some point x is beneficial if and only if

$$|u(x) - u_0|^2 < \frac{\gamma}{\operatorname{vol}(\Omega)^2}.$$

From this equation, we can clearly see that this tolerance for adding points to the domain decreases if the volume of Ω grows. This is a main feature of the new balloon force in comparison to the one used in the original image-segmentation model (5).



(a) (b)

Figure 7.: Shape evolution for the 1D problem (12) according to the exact gradient flow (green), the self-consistent speed field (blue) and the steepest-descent speed under the assumption of a constant volume $b^* - a^*$ (red). The vertical lines indicate the boundaries a^* and b^* of the optimal shape. Note that the blue and green lines overlap on the left. The right plot shows the result of a two-step procedure for a non-monotone situation.

Shape optimisation itself simplifies a lot if only one space dimension is used. If, in particular, the initial domain $\Omega_0 = (a_0, b_0)$ is an interval, it is enough to track the movement of the interval's boundary points in time to describe the shape evolution. In other words, we only have to solve a two-dimensional ODE instead of the level-set equation (1) to compute the domain's time propagation:

$$\begin{aligned} \dot{a}(t) &= -F(a(t), \Omega_t) = -F(a(t), a(t), b(t)), \\ \dot{b}(t) &= F(b(t), \Omega_t) = F(b(t), a(t), b(t)) \end{aligned}$$

with initial values $a(0) = a_0$ and $b(0) = b_0$. This is, of course, much easier. It even allows us to compute the *exact gradient flow* just by using an ODE solver. The speed field corresponding to the steepest descent depends only on the current shape's volume $\mathrm{vol}\,(\Omega_t) = b(t) - a(t)$ and follows from (13). In particular, we have

$$F(x, a, b) = \frac{\gamma}{(b-a)^2} - (u(x) - u_0)^2. \tag{14}$$

Let us simplify the problem even further: We assume that the "image" u is fixed as $u(x) = x$. For this situation, we expect that the optimal shape is given by an interval $(a^*, b^*) = (u_0 - \delta, u_0 + \delta)$ symmetric around u_0. The width 2δ depends on the parameter γ and arises when the error term $(u - u_0)^2 = \delta^2$ and the balloon force $\gamma/(b-a)^2 = \gamma/(2\delta)^2$ are in equilibrium. This is the case for

$$\delta = \sqrt[4]{\frac{\gamma}{4}} \quad \Rightarrow \quad a^* = u_0 - \delta, \; b^* = u_0 + \delta.$$

The numerical results for the choice of $\gamma = 5 \cdot 10^{-2}$ and the symmetric initial shape $(a_0, b_0) = (0.4, 0.6)$ are shown in Figure 7a. In this situation, the shape grows mono-

tonically until it reaches the optimal interval indicated by the vertical lines. The blue line shows the shape evolution according to the self-consistent gradient flow. It matches the exact gradient flow (green line) almost perfectly. For this situation with monotonic growth, only a single step of the procedure outlined above is necessary. The evolving shape converges to the optimal interval for $t \to \infty$. Let us also show that self-consistency really makes a difference: The red line in Figure 7a corresponds to the speed field chosen as the steepest descent (14), but with the shape dependence (i. e., the volume $b-a$) *fixed* at the optimal shape. In this case, we still see convergence towards the optimal shape, but in a clear contrast to the desired gradient flow. Also note that convergence to the optimal shape itself stems only from the fact that we have used precisely the *optimal shape's* volume. This is something that can, of course, not be done in a real computation when the solution is not already known a-priori.

It is also interesting to consider the same example with an initial interval that is not symmetric around u_0. The result for the choice $(a_0, b_0) = (0.2, 0.4)$ is shown in Figure 7b. As before, the exact gradient flow is depicted in green. One can clearly see that the lower boundary $a(t)$ behaves *non-monotonically* in this situation: When Ω_t is small at the beginning, the balloon force is strong enough to grow the interval at both boundaries. This is even true after the lower boundary crosses over the equilibrium value a^*. However, at some point, the expansion due to the upper boundary decreases the force until the lower boundary starts to *increase* again. For $t \to \infty$, both boundaries converge towards their equilibrium values from below. Note that this phenomenon occurs even though the initial interval is a strict subset of the optimal shape. From a naive point of view, one might have expected a monotonic growth towards the optimal shape. To handle this situation with our method, we need at least two steps. The blue curve shows the result with a length of $t_0 = 3$ for the first step. One can nicely see that the evolution according to the self-consistent speed field matches the exact gradient flow initially as before. However, as soon as the lower boundary would have to turn around, it gets stuck instead. With the second step, the direction of the lower boundary is reversed and it converges, as expected, to a^* from below. This shows that the multi-step procedure does, indeed, work well also for difficult situations with non-monotonic behaviour.

5.2 *The Image-Segmentation Problem*

Next, we apply the multi-step self-consistent gradient flow to the full image-segmentation problem of Section 3. For the examples in this subsection, we always use a fixed step length of $t_0 = 10$. The results corresponding to Figure 3 are shown in Figure 8. The first five steps of the evolution for an image without sharp edges are illustrated in Figure 8a. The result is already very close to the final shape. Compare also the cost decrease in Figure 8b to Figure 3b: For the gradient descent, roughly 50 steps are required to bring the cost down to 10^{-5}. The same decrease is realised by the gradient-flow method in only a tenth of this number of steps! Of course, each step for the gradient flow is more expensive as it involves multiple fixed-point iterations with ψ. For this example, however, four iterations per step are enough to reach a point where an increase in the number of fixed-point iterations does not cause any noticeable difference in the result anymore. Thus, even if we take the fixed-point iterations themselves into account as well, it still requires only 20 "operations" to converge to the final shape. The gradient flow is also clearly more efficient in terms of computation time.

If one compares the resulting shapes between Figure 3a and Figure 8a, a striking difference lies in the regularity of the boundary. The gradient descent produces a much

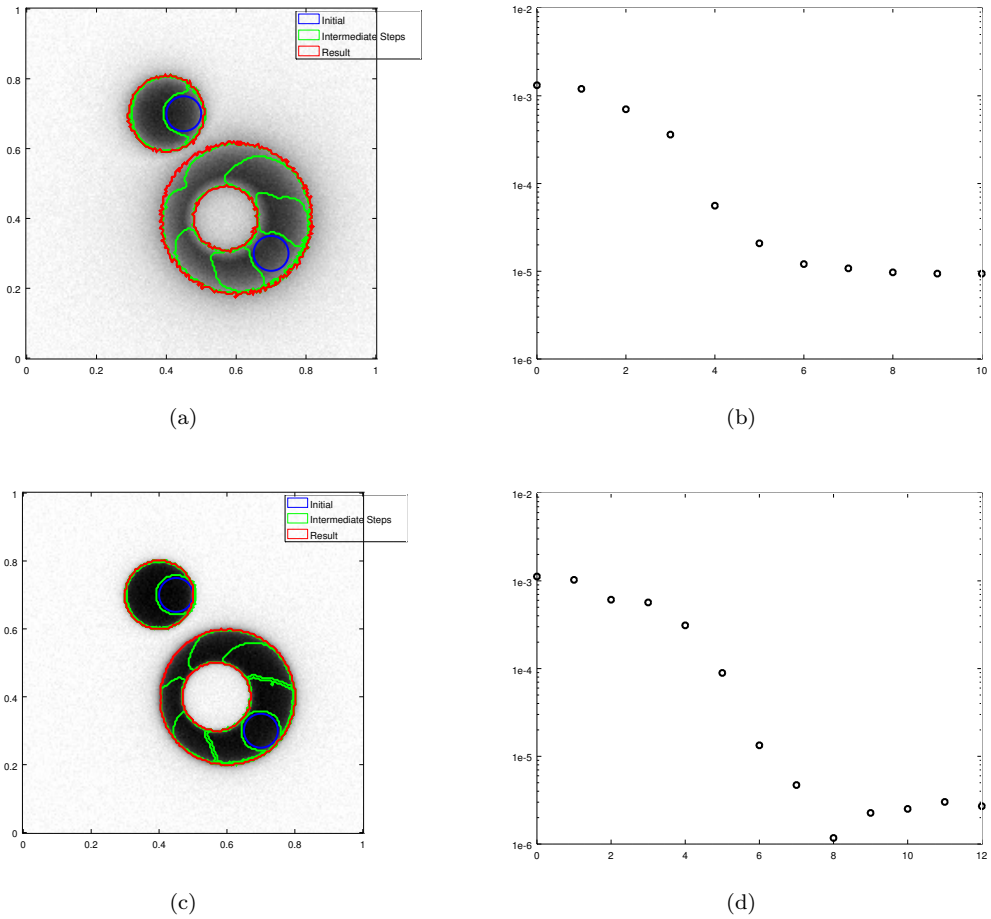


Figure 8.: Application of the multi-step self-consistent gradient flow to our image-segmentation problem. The evolution of the domain is shown on the left for the first few iterations with the green lines. The corresponding evolution of the cost is on the right. Compare also Figure 3.

smoother boundary than the gradient flow. To understand this effect, note that the problem itself (i. e., the cost function in (5)) does not include any regularisation for the boundary at all. Since our input images contain noise, the natural result of the optimisation procedure is thus an irregular boundary as seen in Figure 8a. The smoother boundary of Figure 3a, on the other hand, is a result of the computation of the shape gradient by solving the regularising equation (8).

Let us now do the same comparison for the image of Figure 3c, which has sharper edges. The gradient-descent method does not work for this image, and the desire to find an alternative method was our main motivation for the work on the gradient-flow idea after all. This, of course, leads to the question whether the gradient-flow method performs better for this image. The answer can be seen in Figure 8c: It does. It requires more descent steps (the first six are shown in green), but it still shows convergence to the desired shape. The situation is also slightly more delicate with respect to the fixed-point steps, since the sharp edges lead to larger Lipschitz constants for the shape-dependent quantities. For the shown result, we used ten fixed-point iterations for each descent step.

Nevertheless, it is clear that the method works quite well, while we have seen in Figure 3d that gradient descent simply fails for this image due to excessive "zig-zaging".

6. Incorporating Topological Derivatives

One of the major advantages that level-set methods have over other strategies is their flexibility with respect to topological changes. This was already discussed above and can be seen, for instance, in Figure 3a. However, even though our speed method is, in theory, able to change the topology, there are often situations where these changes are not actually performed in practice. Often, the gradient descent or the gradient flow discussed above avoid topology changes and converge instead to a local minimum with the same topology as the initial geometry. A main issue here is that the speed method is based on transforming the *boundary* of the current domain. Thus, it is neither able to create holes in the interior of Ω nor new components of the domain in $\mathbb{R}^n \setminus \overline{\Omega}$. A common method to overcome this difficulty is to use an initial geometry with lots of components and holes. During the optimisation process, unnecessary holes and components can then disappear or join together. See, for instance, [3] and [28].

For more systematic approaches, a concept called the *topological derivative* can be used. See, for instance, the methods described in [2] and [23]. In particular, this quantity characterises how the cost changes when new components of the domain are created. (It is also often defined as the change of the cost when holes are punched into Ω, but the basic idea is the same. For our discussion, we will concentrate, for simplicity, on the situation of new components.) This is in contrast to the shape derivative discussed so far, which describes the change of the cost when the boundary of Ω is moved. For some $x \in \mathbb{R}^n \setminus \overline{\Omega}$, a simple version of the topological derivative can be formally defined as

$$d_T J(\Omega; x) = \lim_{\rho \to 0^+} \frac{J(\Omega \cup B_\rho(x)) - J(\Omega)}{\operatorname{vol}(B_\rho(x))}.$$

The definition for $x \in \Omega$ and the creation of holes can be done in a similar way. While we do not want to discuss technical details here, let us still describe, at least in an informal way, how topological derivatives can be incorporated into our gradient flow. This results quite naturally in a hybrid method that performs much better for some problems.

For our situation (with only domain functionals), the topological derivative actually matches the shape derivative in some sense. In particular, if the shape derivative is given by (9), then the topological derivative at some point x is

$$d_T J(\Omega; x) = f(x, \Omega).$$

A heuristic argument to see why this is the case is the following: Consider the situation that $x \in \Gamma$ and $F(x) = 1$. The contribution of the particular point x to the shape derivative (9) is then precisely $f(x, \Omega)$. Furthermore, $F(x) > 0$ with $x \in \Gamma$ means that, from a local point of view, x is being added to Ω. This, in turn, is roughly just the meaning of the topological derivative at x.

Now, assume that we have found a self-consistent gradient flow F according to Definition 3. In this case, $F(x) = -f(x, \Omega(x)) = -d_T J(\Omega(x); x)$. Instead of evolving Ω_0 along F to get the next iterate Ω_1 of the geometry, we can instead use the following idea: Whenever $F(x) > 0$, this means that the topological derivative is negative at x. Consequently, it is beneficial for decreasing the cost to include those x into the domain.

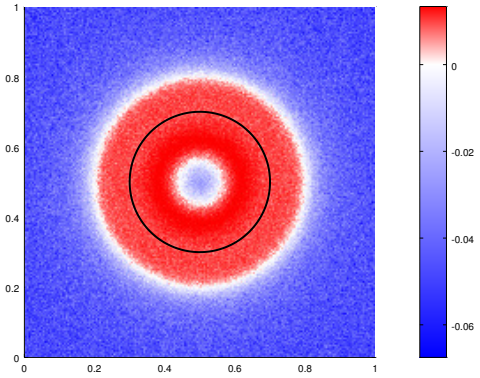
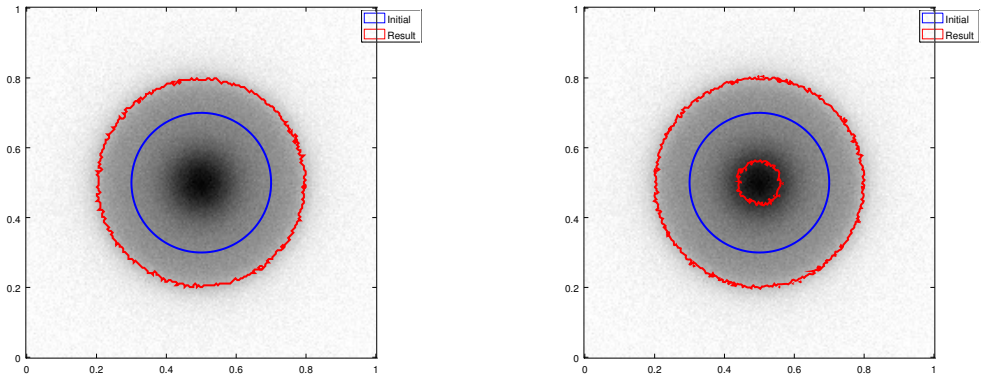


Figure 9.: The self-consistent speed field and the initial geometry for the image-segmentation example discussed in Section 6.



(a) (b)

Figure 10.: The shapes that result after taking one step based on the speed field in Figure 9. We use a shape evolution of Ω_0 according to the speed field on the left, the new shape in the right plot is defined via (15).

Similarly, we want to exclude all x where the topological derivative is positive. Thus, we can simply *define*

$$\Omega_1 = \{x \in \mathbb{R}^n \mid F(x) > 0\}\,. \tag{15}$$

This can be achieved by just interpreting $-F$ as the level-set function for Ω_1. Depending on the problem at hand, one can build a suitable heuristic to decide when to use (15) instead of a shape evolution. For instance, whenever the descent seems to be converged, one can try a step with (15) to check whether this is really the case or it is just stuck in a local minimum. This approach matches the suggestion made in Subsection 3.4 of [23].

To conclude this section, let us demonstrate this idea on a simple example. We use the image-segmentation problem depicted in Figure 10, with the initial geometry Ω_0 shown in blue. For this situation, both the gradient-descent and the gradient-flow method lead to a suboptimal shape that does not exclude the centre. The corresponding speed field

resulting from our fixed-point iteration is plotted in Figure 9. On can clearly see that it (and thus the topological derivatives) are negative where the hole needs to be created. The geometry that results by evolving Ω_0 along the speed field is shown in Figure 10a. In contrast, the domain according to (15) can be seen in Figure 10b. For the situation at hand, the latter is the superior solution.

7. Boundary Length and Mean-Curvature Flow

In the last section of our paper, we would like to discuss how the idea of "a single speed field for the whole evolution" can also be applied to other problems. In particular, an important generalisation of the model problem (5) includes the perimeter of Ω as an additional regularisation term. This prevents the effect of an irregular boundary that we have seen for the examples in Figure 8. Going even further, if the cost function is *just* the perimeter, it is well-known that the shape evolution corresponding to the gradient flow is the so-called *mean-curvature flow*. It has been widely studied both theoretically and from the point of view of applications. See, for instance, [12]. The description of mean-curvature flow with level-set methods is discussed, among others, in [11] and [22]. In particular, the normal speed F is given by the (negative) *mean curvature* κ of the evolving surface. The mean curvature can be computed from a sufficiently smooth level-set function ϕ by the expression

$$\kappa(x) = \operatorname{div}\left(\frac{\nabla\phi(x)}{|\nabla\phi(x)|}\right). \tag{16}$$

(For a derivation, see (1.4.9) on page 29 of [11].) Thus, we can modify our level-set equation (1) to read:

$$0 = \dot{\phi} + F(x)\,|\nabla\phi| = \dot{\phi} - \kappa\,|\nabla\phi| = \dot{\phi} - \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\cdot|\nabla\phi| \tag{17}$$

While this equation has a singularity at $\nabla\phi = 0$, one can still establish a suitable solution theory based on viscosity solutions. This was pioneered by Evans and Spruck in a series of papers starting with [10]. It is also discussed in great detail in [11]. For numerical methods, we refer to [7].

Our discussion is focused on a new approach instead: We would like to compute the mean-curvature flow based on an Eikonal equation similar to Sethian's Fast Marching Method [25] and the approach taken in [16]. This method, of course, has a major drawback: Since it is based on the idea described in Section 4, it only works as long as the evolution of the shape is *monotone*. This is, for instance, the case for mean-convex geometries (meaning that κ is positive all over the surface). However, also many interesting cases are excluded by that requirement. On the other hand, we believe that our new approach also has advantages that make it, nevertheless, interesting to study: First, it may lead to more efficient methods for the computation of the mean-curvature flow if more algorithmic research is devoted to this topic. Second, a description of the mean-curvature flow in terms of an Eikonal equation allows to apply new analytical tools similar to the new results that could be derived in Chapter 4 of [19] based on the Hopf-Lax formula given in Theorem 1. And third, this can be used as a starting point to apply self-consistent gradient-flow methods to problems which have regularisation terms based on the boundary length. Here, one can hope that the restriction to monotone evolutions

can be circumvented with a multi-step procedure similar to the one demonstrated in Section 5. This idea will be discussed briefly at the end of this section and is also an interesting area for further research.

The basic idea of our method is the same as described in Section 4: We want to define a speed field corresponding to the steepest descent and make sure that it is self-consistent with respect to shape-dependent quantities. For the case of mean-curvature flow, we have to set $F(x) = -\kappa(x)$. As before, this value depends on the shape $\Omega(x)$. In contrast to the situation analysed above, however, the shape dependence is *local* this time: The curvature depends only on the shape of the boundary in a neighbourhood of x. It does not depend on things like the volume of Ω or other global quantities. Let us now assume that some speed field F is fixed. We use Theorem 1 to describe the shape evolution. Excluding the stationary case $F(x) = 0$, this implies that a level-set function for the domain evolution is given by

$$\phi(x,t) = D(x) \pm t. \tag{18}$$

Note specifically that the spatial dependence of ϕ is precisely the term $D(x)$. Thus, *the mean curvature $\kappa(x)$ can be computed from D alone without any time-dependent terms.* In particular, (16) implies

$$\kappa(x) = \operatorname{div}\left(\frac{\nabla\phi(x,t)}{|\nabla\phi(x,t)|}\right) = \operatorname{div}\left(\frac{\nabla D(x)}{|\nabla D(x)|}\right).$$

One can also show (compare Lemma 36 in [19]) that the distance D depends on the speed field F via the Eikonal equation

$$F(x)\,|\nabla D(x)| = 1. \tag{19}$$

Consequently, we can conclude for a self-consistent situation:

$$-\kappa\,|\nabla D| = -\operatorname{div}\left(\frac{\nabla D}{|\nabla D|}\right)\cdot|\nabla D| = 1 \tag{20}$$

In other words, the distance that defines the shape evolution for the mean-curvature flow solves the *mean-curvature Eikonal equation* (20).

A possible way to solve (20) numerically is to introduce a pseudotime and evolve towards a stationary situation. For this, we introduce an artificial time dependence of D and propagate the parabolic equation

$$\frac{D_\tau}{|\nabla D|} = \operatorname{div}\left(\frac{\nabla D}{|\nabla D|}\right) + \frac{1}{|\nabla D|}$$

in time (denoted by the variable τ). We can hope that it converges towards a stationary state D_∞ for $\tau \to \infty$, which then solves (20). Note that this equation is the same as the mean-curvature level-set equation (17) except for the additional forcing term. Thus, we can apply the methods of [7] to solve it: As a first step, let us introduce $\epsilon > 0$ as a small regularisation parameter. In order to avoid singularities for $\nabla D = 0$, we consider

the regularised equation

$$\frac{D_\tau}{\sqrt{|\nabla D|^2+\epsilon^2}} = \operatorname{div}\left(\frac{\nabla D}{\sqrt{|\nabla D|^2+\epsilon^2}}\right) + \frac{1}{\sqrt{|\nabla D|^2+\epsilon^2}}. \tag{21}$$

We solve this equation with standard finite elements and a semi-implicit time-stepping scheme. In particular, let $D^i = \sum_k c_k^i u_k$ be the finite-element discretisation of D at some time t_i. Here, $(u_k)_k$ is the finite-element basis and $\left(c_k^i\right)_{ik}$ are the time-dependent coefficients. With

$$D_\tau \approx \frac{D^{i+1}-D^i}{\Delta\tau} = \frac{1}{\Delta\tau}\sum_k \left(c_k^{i+1}-c_k^i\right) u_k,$$

we see that the discrete version

$$\begin{aligned}\frac{1}{\Delta\tau}\sum_k \left(c_k^{i+1}-c_k^i\right) &\int \frac{u_k v}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx \\ &= -\sum_k c_k^{i+1}\int \frac{\langle \nabla u_k, \nabla v\rangle}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx + \int \frac{v}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx\end{aligned}$$

of the weak form of (21) must hold for all test functions v. To simplify notation, let us assemble all coefficients $\left(c_k^i\right)_k$ to a vector c^i and define

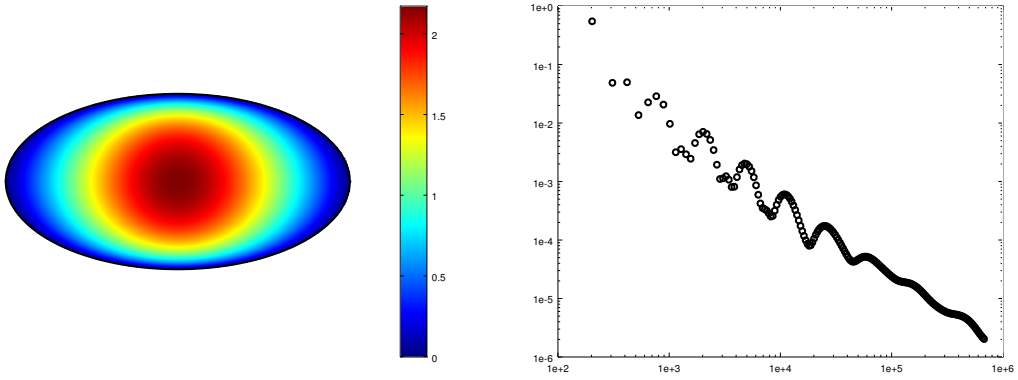
$$M_{kl}^i = \int \frac{u_k u_l}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx, \quad K_{kl}^i = \int \frac{\langle \nabla u_k, \nabla u_l\rangle}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx, \quad f_k^i = \int \frac{u_k}{\sqrt{|\nabla D^i|^2+\epsilon^2}}\,dx.$$

The matrices M and K can be computed by scaling the standard mass and stiffness matrices accordingly. For linear elements, ∇D^i and thus also the weights are constant on each mesh triangle. Then, each time step corresponds to solving the linear system

$$\left(M^i + \Delta\tau \cdot K^i\right) c^{i+1} = M^i c^i + \Delta\tau \cdot f^i.$$

Let us demonstrate the viability of this method: We use an elliptic initial domain Ω_0 and solve (20) on it with the method described above. The resulting distance D, which is the stationary state of (21), is shown (inside of Ω_0) in Figure 11a. Its contour lines correspond to the shape at various times (as per (18)). One can clearly see that the shape first turns into a circle and later vanishes. This is the expected behaviour according to [12]. The relative difference in L^2-norm between consecutive pseudotime steps is plotted in Figure 11b. This clearly shows that the pseudotime iteration converges, indeed, to a stationary state and thus a solution of (20).

To conclude this discussion, let us give a brief outlook how the shape-optimisation idea of Section 4 can be combined with the mean-curvature approach: A combined method promises a way to solve shape-optimisation problems that include perimeter terms as regularisation. Let us, again, consider a base problem whose shape derivative is of the



(a) (b)

Figure 11.: Solution of the mean-curvature Eikonal equation (20) for an elliptic initial geometry. The resulting distance D is shown on the left. The right plot shows the relative L^2-difference between consecutive time steps during the pseudotime evolution.

form (9). If we add $P(\Omega)$ as additional term to the cost, the new shape derivative is

$$dJ(\Omega; F) = \int_\Gamma F(f(x,\Omega) + \kappa(x))\, d\sigma.$$

Thus, the steepest-descent direction is

$$F(x) = -f(x, \Omega(x)) - \kappa(x).$$

As before, κ depends on the distance D, which, in turn, can be computed from the initial speed field F_0 by solving (19). In order to find a self-consistent gradient flow, we now have to find a solution (F, D) to the mixed algebraic-differential equation

$$F = \psi(F) - \operatorname{div}\left(\frac{\nabla D}{|\nabla D|}\right), \quad F\,|\nabla D| = 1. \tag{22}$$

Here, ψ denotes the fixed-point iteration mapping without perimeter regularisation as it was discussed above.

Acknowledgements

The author would like to thank Wolfgang Ring of the University of Graz for thorough proofreading of the manuscript. This work is supported by the Austrian Science Fund (FWF) and the International Research Training Group IGDK 1754.

References

[1] M. Burger, *A Framework for the Construction of Level Set Methods for Shape Optimization and Reconstruction*, Interfaces and Free Boundaries 5 (2003), pp. 301–329.

[2] M. Burger, B. Hackl, and W. Ring, *Incorporating Topological Derivatives into Level Set Methods*, Journal of Computational Physics 194 (2004), pp. 344–362.
[3] T.F. Chan and L.A. Vese, *Image Segmentation Using Level Sets and the Piecewise-Constant Mumford-Shah Model*, Tech. Rep. 00-14, UCLA CAM, 2000.
[4] C. Chen, J. Leng, and G. Xu, *A General Framework of Piecewise-Polynomial Mumford-Shah Model for Image Segmentation*, Tech. Rep. 13-50, UCLA CAM, 2013.
[5] M.G. Crandall, *Viscosity Solutions: a Primer*, in *Viscosity Solutions and Applications*, Lecture Notes in Mathematics, Vol. 1660, Springer, 1997, pp. 1–43.
[6] M.G. Crandall, H. Ishii, and P.L. Lions, *User's Guide to Viscosity Solutions of Second Order Partial Differential Equations*, Bulletin of the American Mathematical Society 27 (1992), pp. 1–67.
[7] K. Deckelnick and G. Dzuik, *Numerical Approximation of Mean Curvature Flow of Graphs and Level Sets*, in *Mathematical Aspects of Evolving Interfaces*, P. Colli and J.F. Rodrigues, eds., Lecture Notes in Mathematics, Vol. 1812, Springer, 2003, pp. 53–87.
[8] M.C. Delfour and J.P. Zolésio, *Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization*, 2nd ed., Advances in Design and Control, SIAM, 2011.
[9] J.W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations* (2015), `https://www.gnu.org/software/octave/doc/interpreter/`.
[10] L.C. Evans and J. Spruck, *Motion of Level Sets by Mean Curvature, I*, Journal of Differential Geometry 33 (1991), pp. 635–681.
[11] Y. Giga, *Surface Evolution Equations: A Level Set Approach*, Monographs in Mathematics, Vol. 99, Birkhäuser, 2006.
[12] M.A. Grayson, *The Heat Equation Shrinks Embedded Plane Curves to Round Points*, Journal of Differential Geometry 26 (1987), pp. 285–314.
[13] S.L. Keeling and L. Knittelfelder, *Image Segmentation Based upon Piecewise Polynomials on Connected Sets*, 2015. Draft, University of Graz.
[14] M. Keuthen, *Second Order Shape Optimization with Geometric Constraints*, Ph.D. diss., Technical University of Munich, 2015.
[15] M. Keuthen and D. Kraft, *Shape Optimization of a Breakwater*, Inverse Problems in Science and Engineering (2015).
[16] D. Kraft, *A Hopf-Lax Formula for the Time Evolution of the Level-Set Equation and a New Approach to Shape Sensitivity Analysis*, preprint IGDK-2015-18, `https://igdk1754.ma.tum.de/foswiki/pub/IGDK1754/Preprints/Kraft_2015A.pdf`.
[17] D. Kraft, *A Hopf-Lax Formula for the Level-Set Equation and Applications to PDE-Constrained Shape Optimisation*, in *Proceedings of the 19th International Conference on Methods and Models in Automation and Robotics*, IEEE Xplore, 2014, pp. 498–503.
[18] D. Kraft, *The `level-set` Package for GNU Octave*, Octave Forge (2014–2015), `http://octave.sourceforge.net/level-set/`.
[19] D. Kraft, *A Level-Set Framework for Shape Optimisation*, Ph.D. diss., University of Graz, 2015.
[20] D. Kraft, *Measure-Theoretic Properties of Level Sets of Distance Functions*, Journal of Geometric Analysis (accepted) (2015).
[21] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer Series in Operation Research and Financial Engineering, Springer, 2006.
[22] S.J. Osher and J.A. Sethian, *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations*, Journal of Computational Physics 79 (1988), pp. 12–49.
[23] R. Ramlau and W. Ring, *A Mumford-Shah Level-Set Approach for the Inversion and Segmentation of X-Ray Tomography Data*, Journal of Computational Physics 221 (2007), pp. 539–557.
[24] F. Santosa, *A Level-Set Approach for Inverse Problems Involving Obstacles*, ESIAM: Control, Optimisation and Calculus of Variations 1 (1996), pp. 17–33.
[25] J.A. Sethian, *A Fast Marching Level Set Method for Monotonically Advancing Fronts*, Proceedings of the National Academy of Sciences 93 (1996), pp. 1591–1595.
[26] J. Sokolowski and J.P. Zolésio, *Introduction to Shape Optimization: Shape Sensitivity Analysis*, Springer Series in Computational Mathematics, Vol. 16, Springer, 1992.
[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 (2002), pp. 271–293.
[28] M.Y. Wang, X. Wang, and D. Guo, *A Level Set Method for Structural Topology Optimization*, Computer Methods in Applied Mechanics and Engineering 192 (2003), pp. 227–246.