SIAM J. APPL. MATH.
Vol. 64, No. 2, pp. 442–467

A SECOND ORDER SHAPE OPTIMIZATION APPROACH FOR IMAGE SEGMENTATION*

MICHAEL HINTERMÜLLER† AND WOLFGANG RING†

Abstract. The problem of segmentation of a given image using the active contour technique is considered. An abstract calculus to find appropriate speed functions for active contour models in image segmentation or related problems based on variational principles is presented. The speed method from shape sensitivity analysis is used to derive speed functions which correspond to gradient or Newton-type directions for the underlying optimization problem. The Newton-type speed function is found by solving an elliptic problem on the current active contour in every time step. Numerical experiments comparing the classical gradient method with Newton's method are presented.

Key words. segmentation, active contours, level set method, shape sensitivity analysis, Newton's method

AMS subject classifications. 68U10, 49Q10, 49Q12

DOI. 10.1137/S0036139902403901

1. Introduction.

Identifying curve-like objects in images is one of the fundamental tasks in image analysis. In image segmentation we are interested in finding boundary curves for regions with approximately constant color or gray values. These curves usually represent boundaries of objects in the image. Image segmentation is therefore often the starting point for the treatment of other, more involved problems in image analysis such as automatic object recognition or image registration.

In recent years, image segmentation has been greatly influenced by two different ideas. On the one hand, global energy principles which should be satisfied for the optimal contour have been introduced and successfully applied. On the other hand, deformable (active) contours, which are represented as zero level sets of a time-dependent function $u : \mathbb{R}^2 \times [0,T] \to \mathbb{R}$, have been used to describe the geometric variable. Kass, Witkin, and Terzopoulos [22] introduced parametrized curves (now referred to as classical snakes) which evolve in such a way that the sum of an internal energy, comprising an elasticity and a rigidity term, and an external energy, indicating the presence of edges in the image, is minimized. Caselles, et al. [6] introduced a geometrically intrinsic (parametrization-independent) formulation of active contours, treating the propagating curve as the zero level set of a function $u : \mathbb{R}^2 \times [0,T] \to \mathbb{R}$. The propagation of the level set function u is driven by an appropriate speed function $F : \mathbb{R}^2 \to \mathbb{R}$, which occurs in the level set equation

$$u_t + F\,|\nabla u| = 0 \text{ on } \mathbb{R}^2. \tag{1.1}$$

The speed function F proposed in [6] is given by

$$F = g\left(\operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) + \nu\right), \tag{1.2}$$

*Received by the editors March 11, 2002; accepted for publication (in revised form) December 20, 2002; published electronically December 31, 2003.
http://www.siam.org/journals/siap/64-2/40390.html

†Special Research Center on Optimization and Control, Institute of Mathematics, University of Graz, Heinrichstrasse 36, A-8010 Graz, Austria (michael.hintermueller@uni-graz.at, wolfgang.ring@kfunigraz.ac.at).

where $g : \mathbb{R}^2 \to \mathbb{R}$ is an edge detector and ν is a constant. The edge detector is chosen in such a way that $g = 0$ at ideal edges of the image and $g > 0$ otherwise. The construction of the speed function F is such that the active contour propagates according to a curve-shortening mean curvature flow (see, e.g., [19]) with an additional constant deflation velocity ν. The motion of the curve is stopped at points which are located on (strong enough) edges where $g \sim 0$. Thus, g functions as a stopping criterion.

Several authors (see, e.g., [8, 26, 27, 37, 30]) have observed that a similar speed function, given by

$$F = \operatorname{div}\left(g\frac{\nabla u}{|\nabla u|}\right) = g \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) + \frac{1}{|\nabla u|}\langle \nabla g, \nabla u\rangle, \tag{1.3}$$

can be interpreted as the gradient direction for the cost functional

$$J(\Gamma) = \int_\Gamma g\, dS \tag{1.4}$$

with respect to the contour Γ, where S denotes the arclength measure on Γ. The flow in the negative gradient direction with respect to the cost functional (1.4) can therefore be considered as a geodesic flow with respect to the Riemannian metric $g : \mathbb{R}^2 \to \mathbb{R}$. Thus, the intrinsic curve propagation (1.1) with speed function (1.3) can also be derived from variational principles. In fact, it can be proved (see [8, 4]) that minimizing the classical snake model and the geodesic model (1.4) are (in some sense) equivalent.

It has turned out to be useful to add a domain integral term to the cost functional (1.4) to speed up the propagation. A corresponding cost functional has the form

$$J(\Gamma) = \int_\Gamma g\, dS + \nu \int_\Omega g\, d\mathbf{x}. \tag{1.5}$$

Many variants of the speed functions (1.2) and (1.3) or their corresponding variational principle have been considered in the literature, including affine invariant geodesic flow [30, 29], generalizations to three-dimensional situations [25, 9], region-based active contours [31, 20, 21], segmentation of moving objects [7], and active contour models based on the Mumford–Shah functional [13, 11, 12, 10]. We also refer to the recent monographs [5, 32], which treat the image segmentation problem extensively and provide numerous references to further literature on the subject.

Usually, in the image processing literature, parametrized contours and methods from classical calculus of variations (see [8, 37, 4]) are used to derive the Euler–Lagrange equation for a cost functional of type (1.4), even if the propagation of the contour is treated in the intrinsic level set formulation. We propose (and advertise) the use of an alternative technique with which we can calculate sensitivities with respect to geometric variables on a purely intrinsic basis. We shall use the speed method, which is commonly applied for the sensitivity analysis of shape optimization problems but, as it seems, is not very well known in the image processing community. The speed method has several advantages over the use of parametrized curves (or surfaces). It is intrinsic, i.e., independent of the chosen parametrization, it can treat the case where the current contour consists of several disjoint closed curves in a unified way, and it has (via the Hadamard–Zolésio structure theorem [15, sect. 3.3, p. 348]) a very natural link to the level set formulation of curve propagation. We also stress

the fact that the Euler–Lagrange equations for many of the cost functionals discussed in the references, which we listed in the previous paragraph, can be easily derived using the speed method. Application of Lemmas 1 and 3 will do the job for most cases. Most of the results related to shape sensitivity calculus (with the exception of the usage of the shape derivative of the signed distance function) can be found in the book by Sokołowski and Zolésio [36] and in the new book by Delfour and Zolésio [15]. The advantage of utilizing shape sensitivity analysis in combination with the level set method as motivated above was previously observed in [24] in the context of inverse problems.

We also want to stress the nature of the segmentation problem as a (nonlinear) optimization problem. It is our goal to find the optimal contour in the least possible number of time steps and to achieve maximal descent in each individual step. This objective is quite different from aiming for a smooth propagation of a contour, which is often the focus of attention for level set–based propagating interface problems. For this reason, we propose applying and adapting ideas from nonlinear programming to active contour propagation. In the following we shall employ line search methods and preconditioning of the gradient direction. To realize the latter idea, we calculate a Newton-type speed function for the level set formulation of the variational problem (1.4). It turns out that the calculation of the Newton-type speed function involves the solution of an elliptic equation on the active contour Γ. That is, we have to track the zero level set at every step of the propagation, and we have to assemble appropriate (geometry-dependent) stiffness and mass matrices. This implementational and computational effort is repaid by a significant reduction of the number of iterations.

The structure of the paper is the following. In section 2, we recall basic facts and formulas from shape sensitivity analysis. Section 3 deals with the calculation of certain useful identities concerning the shape derivative of the signed distance function of a smooth open domain. These identities will prove to be very helpful in section 5. In section 4, we explain how the gradient of a shape functional of the form (1.4) can be interpreted as a normal vector field to the boundary of the current shape via the Hadamard–Zolésio structure theorem and how a connection to the level set formulation can be drawn. In this section, we also introduce the concept of second order (Newton-type) preconditioning of the shape gradient. Section 5 deals with the calculation of the Newton direction for a shape functional of type (1.4). In this context it turns out that, if we restrict our consideration to a certain class of possible speed functions, we get a symmetric shape Hessian, which depends only on intrinsic properties of the current contour. Moreover, this restricted class of speed functions has desirable properties for the stable propagation of the level set function according to the level set equation (1.1). In section 6, we derive the elliptic equation on the actual contour which defines the Newton-type speed function. This equation involves the Laplace–Beltrami operator. The Newton-type algorithm and its numerical realization are the subject of section 7. Also, a numerical technique for relaxing the CFL-condition for the time step size in the level set equation is considered. Finally, in section 8, a report on numerical test runs of the new algorithm and a comparison with the method based on the negative gradient as the speed function are given.

2. Shape sensitivity analysis via the speed method. We briefly recall the speed method from shape optimization, which can be used to calculate sensitivities of a functional with respect to a geometric variable such as a domain or the boundary of an open domain. Our main references for this section are the books [15, 36]. This

section also contains basic facts from tangential calculus on smooth boundaries of open sets.

In image analysis, sensitivities with respect to geometric objects such as contours are usually calculated using parametrized curves and techniques from classical calculus of variations as, e.g., in [8, 37]. We now present a technique for sensitivity analysis which works on boundaries of open sets instead of parametrized curves.

Let $\Gamma = \partial\Omega$ be the boundary of an open set $\Omega \subset \mathbb{R}^2$. We call such a boundary Γ a *contour* in $\mathbb{R}^2$. Suppose $V : \mathbb{R}^2 \to \mathbb{R}^2$ is a given smooth vector field with compact support in $\mathbb{R}^2$. We consider the initial value problem

$$\begin{cases} \mathbf{X}'(t) = V\big(\mathbf{X}(t)\big), \\ \mathbf{X}(0) = \mathbf{x}, \end{cases} \tag{2.1}$$

with $\mathbf{x} \in \mathbb{R}^2$ given. The flow (or time-t map) with respect to V is defined as the mapping $T_t : \mathbb{R}^2 \to \mathbb{R}^2$, with

$$T_t(\mathbf{x}) = \mathbf{X}(t), \tag{2.2}$$

where $\mathbf{X}(t)$ is the solution to (2.1) at time t. If Γ is a contour, we define

$$\Gamma_t = \{T_t(\mathbf{x}) \,:\, \mathbf{x} \in \Gamma\} = T_t(\Gamma). \tag{2.3}$$

In an analogous way, we define $\Omega_t = T_t(\Omega)$ for an arbitrary open set Ω. Note that, if $V \in \mathcal{C}_0^k(\mathbb{R}^2, \mathbb{R}^2)$, then $T_t \in \mathcal{C}^k(\mathbb{R}^2, \mathbb{R}^2)$; thus, smoothness properties of Γ are inherited by Γ_t, provided that the vector field V is smooth enough.

Suppose we are given a functional $J : G \to \mathbb{R}$, where G is an appropriate set of contours. We define the *Eulerian derivative* of J at a contour Γ in the direction of a perturbation vector field V by

$$dJ(\Gamma; V) = \lim_{t \downarrow 0} \frac{1}{t}\Big(J(\Gamma_t) - J(\Gamma)\Big). \tag{2.4}$$

Let B be a Banach space of perturbation vector fields. We say that the functional J is shape differentiable at Γ in B if $dJ(\Gamma; V)$ exists for all $V \in B$ and the mapping $V \mapsto dJ(\Gamma; V)$ is linear and continuous on B. We use the analogous definition for functionals $J(\Omega)$ which depend on an open set Ω as an independent variable instead of on a contour Γ.

We now present a series of lemmas which cover some results from shape calculus which will become useful later on. We start with the Eulerian derivative of a domain integral.

LEMMA 1. *Suppose $\phi \in W^{1,1}_{\mathrm{loc}}(\mathbb{R}^2)$ and $\Omega \subset \mathbb{R}^2$ is open and bounded. Then, the functional*

$$J(\Omega) = \int_\Omega \phi \, d\mathbf{x}$$

is shape differentiable for perturbation vector fields $V \in \mathcal{C}_0^1(\mathbb{R}^2; \mathbb{R}^2)$. The Eulerian derivative of J is given by

$$dJ(\Omega; V) = \int_\Omega \operatorname{div}(\phi\, V)\, d\mathbf{x}. \tag{2.5a}$$

If $\Gamma = \partial\Omega$ is of class $\mathcal{C}^1$, then

$$dJ(\Omega; V) = \int_\Gamma \phi \, \langle V, \mathbf{n}\rangle \, dS, \tag{2.5b}$$

where $\mathbf{n}$ denotes the exterior unit normal vector to Ω, $\langle\cdot,\cdot\rangle$ the inner product on $\mathbb{R}^2$, and dS the arclength measure on Γ.

Proof. See Propositions 2.45 and 2.46 in [36, p. 77]. ∎

For a vector field $V \in \mathcal{C}_0^1(\mathbb{R}^2;\mathbb{R}^2)$ and an open set of class $\mathcal{C}^2$ with boundary Γ, we define the tangential divergence of V by

$$\operatorname{div}_\Gamma V = (\operatorname{div} V - \langle DV \cdot \mathbf{n}, \mathbf{n}\rangle)\big|_\Gamma, \tag{2.6}$$

where DV denotes the Jacobian matrix of V. If the vector field V is defined only on Γ, we can still define the tangential divergence of V as the tangential divergence of an arbitrary extension of V. It can be shown (cf. [36, Prop. 2.51, p. 82]) that the definition does not depend on the particular choice of the extension. With this, we are able to state the following result on boundary integrals.

LEMMA 2. *Suppose $\phi \in W^{2,1}_{\mathrm{loc}}(\mathbb{R}^2)$ and Γ is a contour of class $\mathcal{C}^1$. Then, the functional*

$$J(\Gamma) = \int_\Gamma \phi \, dS \tag{2.7}$$

is shape differentiable for perturbation vector fields $V \in \mathcal{C}_0^1(\mathbb{R}^2;\mathbb{R}^2)$ with

$$dJ(\Gamma; V) = \int_\Gamma \big(\langle \nabla\phi, V\rangle + \phi \operatorname{div}_\Gamma V\big)\, dS. \tag{2.8}$$

Proof. See sections 2.18 and 2.19 in [36]. ∎

Using tangential calculus (see sections 2.19 and 2.20 in [36] or the results in [16]), we can simplify the expression (2.8). We define the tangential gradient of a function $h \in \mathcal{C}^2(\Gamma)$ as

$$\nabla_\Gamma h = \nabla \tilde{h}\big|_\Gamma - \frac{\partial \tilde{h}}{\partial n}\mathbf{n} \tag{2.9}$$

on Γ, where $\tilde{h}$ denotes an arbitrary smooth extension of h. It can be shown that the definition (2.9) does not depend on the specific choice of the extension. We have the following Green's formula on Γ.

PROPOSITION 1 (Green's theorem on Γ). *Suppose Γ is a contour of class $\mathcal{C}^2$, $h \in \mathcal{C}^2(\Gamma)$, and $V \in \mathcal{C}_0^1(\mathbb{R}^2;\mathbb{R}^2)$ with $\langle V, \mathbf{n}\rangle = 0$ for every point $\mathbf{x} \in \Gamma$. Then, we have*

$$\int_\Gamma \langle \nabla_\Gamma h, V\rangle \, dS = -\int_\Gamma h \operatorname{div}_\Gamma V \, dS. \tag{2.10}$$

Remark 1. Green's formula also holds for functions h in the Sobolev space $H^1(\Gamma)$. In this case, (2.10) acts as a definition for the tangential gradient $\nabla_\Gamma h$.

Suppose we are given a smooth vector field V. We set $V_\tau = V - \langle V, \mathcal{N}\rangle \mathcal{N}$ as the tangential component of V with respect to Γ. Here $\mathcal{N}$ denotes an extension of the normal vector field $\mathbf{n}$ on Γ. We have

$$\begin{aligned}
\operatorname{div}_\Gamma V &= \operatorname{div}_\Gamma V_\tau + \operatorname{div}_\Gamma\big(\langle V, \mathcal{N}\rangle \mathcal{N}\big) \\
&= \operatorname{div}_\Gamma V_\tau + \big(\operatorname{div}\big(\langle V, \mathcal{N}\rangle \mathcal{N}\big) - \big\langle D\big(\langle V, \mathcal{N}\rangle \mathcal{N}\big)\cdot \mathcal{N}, \mathcal{N}\big\rangle\big)\big|_\Gamma \\
&= \operatorname{div}_\Gamma V_\tau + \big(\langle V, \mathcal{N}\rangle \big(\operatorname{div}\mathcal{N} - \langle D\mathcal{N}\cdot\mathcal{N}, \mathcal{N}\rangle\big)\big)\big|_\Gamma \\
&= \operatorname{div}_\Gamma V_\tau + \langle V, \mathcal{N}\rangle \operatorname{div}_\Gamma \mathcal{N}\big|_\Gamma.
\end{aligned}$$

The term $\operatorname{div}_\Gamma \mathcal{N}|_\Gamma$ is usually denoted by κ and is called the *curvature* of Γ. Thus, we find

$$\operatorname{div}_\Gamma V = \operatorname{div}_\Gamma V_\tau + \kappa \langle V, \mathbf{n} \rangle$$

on Γ. We thus obtain an equivalent expression for the Eulerian derivative of the cost functional (2.7). We have

$$\begin{aligned} dJ(\Gamma; V) &= \int_\Gamma \left(\langle \nabla \phi, V \rangle + \phi \operatorname{div}_\Gamma V \right) dS \\ &= \int_\Gamma \left(\left\langle \nabla_\Gamma \phi + \frac{\partial \phi}{\partial n} \mathbf{n}, V \right\rangle + \phi \operatorname{div}_\Gamma (V_\tau) + \phi \kappa \langle V, \mathbf{n} \rangle \right) dS \\ &= \int_\Gamma \left(\frac{\partial \phi}{\partial n} + \phi \kappa \right) \langle V, \mathbf{n} \rangle \, dS + \int_\Gamma \left(\langle \nabla_\Gamma \phi, V_\tau \rangle + \phi \operatorname{div}_\Gamma (V_\tau) \right) dS. \end{aligned}$$

The last integral is zero due to Proposition 1. We therefore obtain the following lemma.

LEMMA 3. *Under the assumptions of Lemma* 2 *the Eulerian derivative of the cost functional* (2.7) *is equivalently given by*

$$dJ(\Gamma; V) = \int_\Gamma \left(\frac{\partial \phi}{\partial n} + \phi \kappa \right) \langle V, \mathbf{n} \rangle \, dS. \tag{2.11}$$

It is also useful to be able to calculate sensitivities for more general functionals of the form

$$J(\Omega) = \int_\Omega \phi(\Omega, \mathbf{x}) \, d\mathbf{x} \tag{2.12}$$

or

$$J(\Gamma) = \int_\Gamma \psi(\Gamma, \mathbf{x}) \, dS(\mathbf{x}), \tag{2.13}$$

where the functions $\phi(\Omega) : \Omega \to \mathbb{R}$ and $\psi(\Gamma) : \Gamma \to \mathbb{R}$ themselves depend on the geometric variables Ω and Γ, respectively. In this case, formulas (2.5) and (2.11) have to be corrected by terms which take care of the derivatives of ϕ and ψ with respect to Ω or Γ. We define the following two variants of derivatives of a geometry-dependent function with respect to the geometry.

DEFINITION 1. *Suppose $\psi(\Gamma) \in B(\Gamma)$ for all $\Gamma \in G$, where $B(\Gamma)$ is some appropriate Banach space of functions on Γ, and let $V \in \mathcal{C}_0^1(\mathbb{R}^2, \mathbb{R}^2)$. We set $\psi^t = \psi(\Gamma_t) \circ T_t(V)$ and $\psi^0 = \psi(\Gamma)$, and we assume that $\psi^t \in B(\Gamma_t)$ for all $0 < t < T$ with some $T > 0$. If the limit*

$$\dot{\psi}(\Gamma; V) = \lim_{t \downarrow 0} \frac{1}{t} \left(\psi^t - \psi^0 \right) \tag{2.14}$$

exists in the strong (weak) topology on $B(\Gamma)$, then $\dot{\psi}(\Gamma; V)$ is called the strong (weak) material derivative *of ψ at Γ in direction V.*

The analogous definition holds for functions $\phi(\Omega)$ which are defined on open sets and not on contours.

The material derivative is the derivative of ϕ (or ψ) with respect to the geometry for a moving (Lagrangian) coordinate system. Let us first consider the case of a

domain function $\phi : \Omega \to \mathbb{R}$. It is easily seen that, for the special case where ϕ is independent of Ω, we find

$$\dot{\phi}(\Omega; V) = \dot{\phi}(V) = \langle \nabla \phi, V \rangle.$$

For a function which does not depend on Ω, any reasonable derivative with respect to Ω in a fixed (Eulerian) coordinate system must be 0. It is therefore natural to subtract the term $\langle \nabla \phi, V \rangle$ from $\dot{\phi}$ to define a derivative of ϕ with respect to Ω in a stationary coordinate system. This is the idea of the following definition.

DEFINITION 2. *Suppose that the weak material derivative $\dot{\phi}(\Omega; V)$ and the expression $\langle \nabla \phi(\Omega), V \rangle$ exist in $B(\Omega)$. Then, we set*

$$\phi'(\Omega; V) = \dot{\phi}(\Omega; V) - \langle \nabla \phi, V \rangle \tag{2.15}$$

and we call $\phi'(\Omega; V)$ the shape derivative *of ϕ at Ω in direction V.*

Note that

$$\phi'(\Omega; V) = \phi'(V) = 0$$

for any function ϕ which does not depend on Ω.

For boundary functions $\psi(\Gamma) : \Gamma \to \mathbb{R}$, the expression $\langle \nabla \psi, V \rangle$ does not make sense. In this case, we define the shape derivative as

$$\psi'(\Gamma; V) = \dot{\psi}(\Gamma; V) - \langle \nabla_\Gamma \psi, V \rangle\big|_\Gamma. \tag{2.16}$$

With these definitions we are able to calculate the Eulerian derivatives for the shape functionals (2.12) and (2.13).

PROPOSITION 2. *Suppose $\phi = \phi(\Omega)$ is given such that the weak L^1-material derivative $\dot{\phi}(\Omega; V)$ and the shape derivative $\phi'(\Omega; V) \in L^1(\Omega)$ exist. Then, the cost functional* (2.12) *is shape differentiable and we have*

$$dJ(\Omega; V) = \int_\Omega \phi'(\Omega; V)\, d\mathbf{x} + \int_\Gamma \phi \, \langle V, \mathbf{n} \rangle \, dS. \tag{2.17}$$

For boundary functions $\psi(\Gamma)$ we get, under the same technical assumptions for the cost functional (2.13)*,*

$$dJ(\Gamma; V) = \int_\Gamma \psi'(\Gamma; V)\, dS + \int_\Gamma \kappa \, \psi \, \langle V, \mathbf{n} \rangle \, dS. \tag{2.18}$$

If $\psi(\Gamma) = \phi(\Omega)\big|_\Gamma$, then we have

$$dJ(\Gamma; V) = \int_\Gamma \phi'(\Omega; V)\big|_\Gamma \, dS + \int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \kappa \, \phi \Big) \, \langle V, \mathbf{n} \rangle \, dS. \tag{2.19}$$

Suppose that $\phi(\Omega)$ satisfies $\phi(\Omega)|_\Gamma = 0$ for all (admissible) domains Ω, and let $\vartheta \in \mathcal{D}(\mathbb{R}^2)$ be given. We define the cost functional

$$J_0(\Gamma) = \int_\Gamma \vartheta \, \phi(\Omega) \, dS = 0$$

for arbitrary Γ. Thus,

$$0 = dJ_0(\Gamma; V) = \int_\Gamma \vartheta \, \phi'(\Omega, V)\, dS + \int_\Gamma \frac{\partial}{\partial n} \Big(\vartheta \, \phi(\Omega) \Big) \, \langle V, \mathbf{n} \rangle \, dS.$$

If we choose ϑ such that

$$\frac{\partial \vartheta}{\partial n} = 0 \quad \text{on } \Gamma \tag{2.20}$$

and if we use the fact that the set of test functions which satisfy (2.20) is dense in $L^2(\Gamma)$, we get

$$\phi'(\Omega; V)\big|_\Gamma = -\frac{\partial \phi}{\partial n} \langle V, \mathbf{n}\rangle\big|_\Gamma$$

on Γ. We have therefore proved the following lemma.

LEMMA 4. *Suppose that $\phi(\Omega) \in H^{\frac{3}{2}+\epsilon}(\Omega)$ satisfies $\phi(\Omega)|_\Gamma = 0$ for all (admissible) domains Ω and that the shape derivative $\phi'(\Omega; V)$ exists in $H^{\frac{1}{2}+\epsilon}(\Omega)$ for some $\epsilon > 0$. Then, we have*

$$\phi'(\Omega; V)\big|_\Gamma = -\frac{\partial \phi}{\partial n} \langle V, \mathbf{n}\rangle\big|_\Gamma. \tag{2.21}$$

Remark 2. The Hadamard–Zolésio structure theorem [15, Thm. 3.6 and Cor. 1, p. 348f] states that the Eulerian derivative of a domain or boundary functional always has a representation of the form

$$dJ(\Omega; V) = \big\langle G, \langle V, \mathbf{n}\rangle \big\rangle_{\mathcal{C}^{-k}(\Gamma), \mathcal{C}^k(\Gamma)} = \big\langle G\,\mathbf{n}, V \big\rangle_{\mathcal{C}_2^{-k}(\Gamma), \mathcal{C}_2^k(\Gamma)}; \tag{2.22}$$

that is, the Eulerian derivative is concentrated on Γ and can be identified with the *normal vector field* $G\,\mathbf{n}$ on Γ. We set

$$D_\Gamma J(\Omega) = G\,\mathbf{n}, \tag{2.23}$$

and we call this expression the *shape gradient* of J at Ω.

3. Shape derivative of the signed distance function. The signed (or oriented) distance function is a useful tool in shape analysis. Many differential geometric quantities such as the normal vector field of a contour Γ or its curvature can be easily expressed in terms of the signed distance function b_Γ of Γ. We shall now apply the techniques introduced in the previous section to calculate the shape derivative of the signed distance function of a given (open, bounded) set Ω. This will be helpful later on when we have to calculate Eulerian derivatives of functionals which depend also on geometric properties of Γ such as normal direction or curvature. The following definitions and facts are taken from [15, Chap. 5]. The *distance function* d_A of a subset $A \subset \mathbb{R}^2$ is defined as

$$d_A(\mathbf{x}) = \inf_{\mathbf{y}\in A} |\mathbf{y} - \mathbf{x}|. \tag{3.1}$$

The *signed distance function* b_Ω of a bounded open set $\Omega \subset \mathbb{R}^2$ is defined as

$$b_\Omega(\mathbf{x}) = d_\Omega(\mathbf{x}) - d_{\mathbb{R}^2\setminus\Omega}(\mathbf{x}). \tag{3.2}$$

If we set $\Gamma = \partial\Omega$, we can express d_Ω in terms of Γ. We have

$$b_\Omega(\mathbf{x}) = \begin{cases} d_\Gamma(\mathbf{x}) & \text{for } \mathbf{x} \in \operatorname{int}(\mathbb{R}^2 \setminus \Omega), \\ 0 & \text{for } \mathbf{x} \in \Gamma, \\ -d_\Gamma(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega. \end{cases} \tag{3.3}$$

We shall use the notation $b_\Gamma = b_\Omega$. Note in particular that

$$b_\Gamma\big|_\Gamma = 0. \tag{3.4}$$

It can be shown that b_Γ is uniformly Lipschitz continuous on $\mathbb{R}^2$ and hence, by Rademacher's theorem, differentiable a.e. on $\mathbb{R}^2$ with $|\nabla b_\Gamma| = 1$ a.e. on $\mathbb{R}^2 \setminus \Gamma$. If $\mathrm{meas}(\Gamma) = 0$, then we have

$$|\nabla b_\Gamma|^2 = 1 \quad \text{a.e. on } \mathbb{R}^2. \tag{3.5}$$

If Γ is smooth and compact ($C^{1,1}$ is enough), then ∇b_Γ is Lipschitz continuous, and we have $\nabla b_\Gamma(\mathbf{x}) = \mathbf{n}(p_\Gamma(\mathbf{x}))$ for all $\mathbf{x}$ in some neighborhood of Γ, where p_Γ denotes the projection onto Γ. Thus, ∇b_Γ can be considered as an extension of the unit normal vector field $\mathbf{n}$ onto a neighborhood of Γ, and we have

$$\nabla b_\Gamma\big|_\Gamma = \mathbf{n}. \tag{3.6}$$

Moreover, the second fundamental form of Γ can be expressed in terms of b_Γ. For a C^2-submanifold $\Gamma \subset \mathbb{R}^2$ we have

$$\Delta b_\Gamma\big|_\Gamma = \kappa. \tag{3.7}$$

See [15, p. 369] for the last relation. Taking the gradient on both sides of the Eikonal equation (3.5) yields

$$D^2 b_\Gamma \cdot \nabla b_\Gamma = 0 \quad \text{on } \Gamma. \tag{3.8}$$

Let $W \in \mathcal{C}_0^1(\mathbb{R}^2, \mathbb{R}^2)$ be a given perturbation vector field. We shall derive certain properties of the shape derivative $b'_\Gamma = b'_\Gamma(\Gamma; W)$ of the signed distance function. The signed distance function satisfies the Eikonal equation (3.5) together with the boundary condition (3.4). The weak form of (3.5) is given by

$$\int_{\mathbb{R}^2} |\nabla b_\Gamma|^2 \, \psi \, d\mathbf{x} = \int_{\mathbb{R}^2} \psi \, d\mathbf{x} \tag{3.9}$$

for all test functions $\psi \in \mathcal{D}(\mathbb{R}^2)$. Taking the Eulerian derivative on both sides of (3.9) and using (2.17), we get

$$2 \int_{\mathbb{R}^2} \langle \nabla b'_\Gamma, \nabla b_\Gamma \rangle \, \psi \, d\mathbf{x} = 0 \tag{3.10}$$

for all $\psi \in \mathcal{D}(\mathbb{R}^2)$. Note that, since the functional (3.9) is defined on a fixed domain and depends on Γ only via b_Γ in the integral, the boundary term in (2.17) vanishes. This can be see by writing

$$\int_{\mathbb{R}^2} |\nabla b_\Gamma|^2 \, \psi \, d\mathbf{x} = \int_\Omega |\nabla b_\Gamma|^2 \, \psi \, d\mathbf{x} + \int_{\mathbb{R}^2 \setminus \Omega} |\nabla b_\Gamma|^2 \, \psi \, d\mathbf{x}$$

and applying (2.17) to both terms on the right-hand side. The boundary integrals from both contributions sum up to zero. Equation (3.10) implies that

$$\langle \nabla b'_\Gamma, \nabla b_\Gamma \rangle = 0. \tag{3.11}$$

Equation (3.11) holds at least on some neighborhood of Γ on which b_Γ is smooth enough to guarantee the existence of a (weak) material derivative and hence the applicability of (2.17).

If we apply Lemma 4 to b_Γ and use (3.5), we get $b'_\Gamma|_\Gamma = -\frac{\partial b_\Gamma}{\partial n}\langle V, \mathbf{n}\rangle|_\Gamma = -\langle \nabla b_\Gamma, \nabla b_\Gamma\rangle \cdot \langle V, \mathbf{n}\rangle = -\langle V, \mathbf{n}\rangle$ by (3.5). With $v_n = \langle V, \mathbf{n}\rangle$, we obtain

$$b'_\Gamma\big|_\Gamma = -v_n. \tag{3.12}$$

Since $\nabla b'_\Gamma$ is orthogonal to $\mathbf{n}$ by (3.11), we have

$$\nabla b'_\Gamma = \nabla_\Gamma b'_\Gamma = -\nabla_\Gamma v_n \text{ on } \Gamma. \tag{3.13}$$

Moreover, we have $\Delta b'_\Gamma\big|_\Gamma = \operatorname{div}(\nabla b'_\Gamma)\big|_\Gamma = \operatorname{div}_\Gamma(\nabla_\Gamma b'_\Gamma) + \langle D^2 b'_\Gamma \cdot \nabla b_\Gamma, \nabla b_\Gamma\rangle\big|_\Gamma$. Because $0 = \nabla\langle \nabla b'_\Gamma, \nabla b_\Gamma\rangle = D^2 b'_\Gamma \cdot \nabla b_\Gamma + D^2 b_\Gamma \cdot \nabla b'_\Gamma$ and since $D^2 b_\Gamma$ is symmetric, we can conclude that $\langle D^2 b'_\Gamma \cdot \nabla b_\Gamma, \nabla b_\Gamma\rangle = -\langle D^2 b_\Gamma \cdot \nabla b'_\Gamma, \nabla b_\Gamma\rangle = \langle \nabla b'_\Gamma, D^2 b_\Gamma \cdot \nabla b_\Gamma\rangle = 0$ due to (3.8). Therefore, we obtain

$$\Delta b'_\Gamma\big|_\Gamma = -\Delta_\Gamma v_n. \tag{3.14}$$

4. Gradient and Newton-type level set flow for a shape optimization problem. For the numerical solution of a shape optimization problem one can use shape sensitivity information to move the geometric variable step by step in the direction of the negative gradient. Alternatively, one can use some other descent direction, which can be obtained from the gradient by applying a Newton-type preconditioner to the negative gradient. Like the shape gradient, the descent direction should have the form of a normal vector field on Γ (see Remark 2). Suppose $F\,\mathbf{n}$ is such a descent direction, where $F : \Gamma \to \mathbb{R}$ is a scalar function which depends on Γ. If we embed the discrete iterative optimization procedure in a continuous flow $\Gamma(t)$ which propagates in direction $F\,\mathbf{n}$, we get the following propagating front formulation for $\Gamma(t)$:

$$\dot{\mathbf{x}}(t) = F\big(\mathbf{x}(t), \Gamma(t)\big)\,\mathbf{n}(\mathbf{x}(t)) \quad \text{for } \mathbf{x}(t) \in \Gamma(t). \tag{4.1}$$

An equivalent formulation is given by the level set equation

$$u_t + \tilde{F}|\nabla u| = 0 \quad \text{on } \mathbb{R}^2 \times (0, T), \tag{4.2}$$

where the propagating front is the zero level set of the function u, i.e.,

$$\Gamma(t) = \{\mathbf{x} \in \mathbb{R}^2 \,:\, u(\mathbf{x}, t) = 0\}. \tag{4.3}$$

In (4.2), the scalar function $\tilde{F} : \mathbb{R}^2 \times [0, T) \to \mathbb{R}$ is chosen such that $\tilde{F}|_{\Gamma(t)} = F(\Gamma(t))$. See [35] for an extensive exposition of propagating front problems and their analytical and numerical treatment in the level set context. Note that by the Hadamard–Zolésio structure theorem (see Remark 2), the shape gradient $D_\Gamma J$ can always be interpreted as a scalar speed function G on Γ, which can be used in the level set formulation. Thus, shape sensitivity analysis and the level set method can be combined in a very natural way.

We want to use a speed function $F : \Gamma \to \mathbf{R}$, which represents a Newton-type descent direction for the shape optimization problem (1.4). This function is determined in the following way. Let $F : \Gamma \to \mathbb{R}$ and $G : \Gamma \to \mathbb{R}$ be given functions. We now establish a one-to-one correspondence between scalar speed functions and a

certain class of perturbation vector fields. Let $\tilde{F}$ and $\tilde{G}$ denote extensions of F and G, respectively, which are constructed as solutions to the transport equations

$$\langle \nabla \tilde{F}, \nabla b_\Gamma \rangle = 0 \quad \text{on } \mathbb{R}^2, \quad \tilde{F}\big|_\Gamma = F, \tag{4.4}$$

and

$$\langle \nabla \tilde{G}, \nabla b_\Gamma \rangle = 0 \quad \text{on } \mathbb{R}^2, \quad \tilde{G}\big|_\Gamma = G. \tag{4.5}$$

Note that Γ is noncharacteristic with respect to the transport equation; thus, (4.4) and (4.5) have unique solutions, at least locally in some neighborhood of Γ, which is small enough such that the characteristics of (4.4) (which are straight lines) do not intersect. With these solutions, we define the vector fields

$$V_F = \tilde{F}\,\nabla b_\Gamma \quad \text{and} \quad V_G = \tilde{G}\,\nabla b_\Gamma \tag{4.6}$$

on some neighborhood of Γ on which $\tilde{F}$, $\tilde{G}$, and ∇b_Γ are smooth. Outside this neighborhood we assume that V_F and V_G are extended in some smooth way. Note that the construction of V_F and V_G is such that

$$\langle V_F, \mathbf{n} \rangle = F \text{ and } \langle V_G, \mathbf{n} \rangle = G \quad \text{on } \Gamma. \tag{4.7}$$

Now we consider a cost functional of type (1.4). Let $d^2J(\Gamma;V;W) = d(dJ(\Omega;V))(\Omega;W)$ be the second Eulerian derivative of the cost functional (1.4). In general, the second Eulerian derivative is not symmetric in the two arguments V and W and does not depend only on $V|_\Gamma$ and $W|_\Gamma$. From the subsequent computation we shall see, however, that for perturbation vector fields of the form (4.6), the second Eulerian derivative is symmetric in (V_F, V_G) and depends only on F and G.

We propose the following optimization algorithm. We define a Newton-type speed function $F : \Gamma \to \mathbf{R}$ as the solution to

$$d^2J(\Gamma; V_F; V_G) = -dJ(\Gamma; V_G) \quad \text{for all } G : \Gamma \to \mathbb{R}. \tag{4.8}$$

We then find the extension $\tilde{F}$ of F onto some neighborhood of Γ by solving the transport equation (4.4). Finally, we use $\tilde{F}$ as speed function for one time step in the level set equation

$$u_t + \tilde{F}|\nabla u| = 0. \tag{4.9}$$

The step size is chosen such that some line search criterion is satisfied. With the updated geometry, we start the procedure over again until some stopping criterion is reached.

5. Calculation of the Newton-type speed function. In this section, we calculate the second Eulerian derivative for shape functionals of the form

$$J_1(\Gamma) = \int_\Gamma \phi \, dS \tag{5.1}$$

and

$$J_2(\Omega) = \int_\Omega \phi \, d\mathbf{x} \tag{5.2}$$

for some fixed $\phi \in W^{2,1}_{loc}(\mathbb{R}^2)$. For the following calculations we assume that Γ is of class $\mathcal{C}^2$, which implies that $b_\Gamma \in \mathcal{C}^2$ on some neighborhood of Γ (see [15, Thm. 4.3, p. 219]).

When it is clear from the context, we omit "$|_\Gamma$." We start with the calculation for J_1. Using (2.11), (3.6), and (3.7), we obtain

$$\begin{aligned} dJ_1(\Gamma, V_F) &= \int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \phi\,\kappa\Big)\,\langle V_F, \mathbf{n}\rangle\,dS \\ &= \int_\Gamma \big(\langle \nabla\phi, \nabla b_\Gamma\rangle + \phi\,\Delta b_\Gamma\big)\,\langle V_F, \nabla b_\Gamma\rangle\,dS. \end{aligned} \tag{5.3}$$

In this section, we consider only perturbation vector fields V_F, which satisfy (4.6). Note that $dJ_1(\Gamma; V_F)$ (for fixed V_F) is a shape functional of type (2.13). Therefore, the second Eulerian derivative can be calculated by applying formula (2.19) to $dJ_1(\Gamma; V_F)$. With $b'_\Gamma = b'_\Gamma(\Gamma; V_G)$ we get

$$\begin{aligned} d^2J_1(\Gamma; V_F; V_G) &= \int_\Gamma \frac{\partial}{\partial n}\Big[\big(\langle \nabla\phi, \nabla b_\Gamma\rangle + \phi\,\Delta b_\Gamma\big)\,\langle V_F, \nabla b_\Gamma\rangle\Big]\,\langle V_G, \mathbf{n}\rangle\,dS \\ &\quad + \int_\Gamma \kappa\,\Big(\frac{\partial \phi}{\partial n} + \phi\,\kappa\Big)\,\langle V_F, \mathbf{n}\rangle\,\langle V_G, \mathbf{n}\rangle\,dS \\ &\quad + \int_\Gamma \big(\langle \nabla\phi, \nabla b'_\Gamma\rangle + \phi\,\Delta b'_\Gamma\big)\,\langle V_F, \mathbf{n}\rangle\,dS \\ &\quad + \int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \phi\,\kappa\Big)\,\langle V_F, \nabla b'_\Gamma\rangle\,dS \\ &= I_1 + I_2 + I_3 + I_4. \end{aligned}$$

Using (4.7), (3.13), (3.14), and Green's formula (2.10), the integral I_3 simplifies to

$$\begin{aligned} I_3 &= -\int_\Gamma \big(\langle \nabla\phi, \nabla_\Gamma G\rangle + \phi\Delta_\Gamma G\big)\,F\,dS \\ &= -\int_\Gamma \big(\langle \nabla\phi, \nabla_\Gamma G\rangle\,F - \langle \nabla_\Gamma(\phi\,F), \nabla_\Gamma G\rangle\big)\,dS \\ &= \int_\Gamma \phi\,\langle \nabla_\Gamma F, \nabla_\Gamma G\rangle\,dS. \end{aligned} \tag{5.4}$$

Now let us consider I_1. We have

$$\begin{aligned} I_1 &= \int_\Gamma \Big(\frac{\partial^2 \phi}{\partial n^2} + \frac{\partial \phi}{\partial n}\,\kappa + \phi\langle \nabla(\Delta b_\Gamma), \nabla b_\Gamma\rangle\Big)\,F\,G\,dS \\ &\quad + \int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \kappa\,\phi\Big)\,\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle\,G\,dS \\ &= K_1 + K_2. \end{aligned}$$

From (3.5), we conclude

$$0 = \Delta\langle \nabla b_\Gamma, \nabla b_\Gamma\rangle = 2\langle \nabla(\Delta b_\Gamma), \nabla b_\Gamma\rangle + 2D^2 b_\Gamma : D^2 b_\Gamma,$$

where $A : B = \sum_{i,j} a_{i,j}\,b_{i,j}$ denotes the tensor product of matrices $A = (a_{i,j})$ and $B = (b_{i,j})$. Thus,

$$\langle \nabla(\Delta b_\Gamma), \nabla b_\Gamma\rangle = -\|D^2 b_\Gamma\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix.

In I_2 we find a term of the form $\phi\,\kappa^2\,F\,G$. We have $\kappa^2 = (\mathrm{trace}(D^2 b_\Gamma))^2$. In two dimensions, the relation

$$(\mathrm{trace} D^2 b_\Gamma)^2 - \|D^2 b_\Gamma\|_F^2 = 2(b_\Gamma)_{x_1,x_1}(b_\Gamma)_{x_2,x_2} - 2(b_\Gamma)_{x_1,x_2}^2 = 2\det(D^2 b_\Gamma) = 0$$

holds due to (3.8). With this, we obtain

$$I_2 + K_1 = \int_\Gamma \Big(\frac{\partial^2\phi}{\partial n^2} + 2\frac{\partial\phi}{\partial n}\,\kappa\Big)\,F\,G\,dS. \tag{5.5}$$

The remaining term is $K_2 + I_4$. We find

$$\begin{aligned} K_2 + I_4 &= \int_\Gamma \Big(\frac{\partial\phi}{\partial n} + \kappa\,\phi\Big)\,\Big(\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle\,G - \langle V_F, \nabla_\Gamma G\rangle\Big)\,dS \\ &= \int_\Gamma \Big(\frac{\partial\phi}{\partial n} + \kappa\,\phi\Big)\,\Big(\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle\,G - \langle V_F, \nabla_\Gamma G\rangle\Big)\,dS. \end{aligned}$$

For the second expression in the above integral, we obtain, using (4.7) and definition (2.9),

$$\langle V_F, \nabla_\Gamma G\rangle = \Big\langle V_F, \nabla\langle V_G, \nabla b_\Gamma\rangle - \frac{\partial}{\partial n}\langle V_G, \nabla b_\Gamma\rangle\,\mathbf{n}\Big\rangle.$$

Thus, we get

$$\begin{aligned} &K_2 + I_4 \\ &\quad = \int_\Gamma \Big(\frac{\partial\phi}{\partial n} + \kappa\,\phi\Big)\,\Big(\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle\,G + \frac{\partial}{\partial n}\langle V_G, \nabla b_\Gamma\rangle\,F - \big\langle \nabla\langle V_G, \nabla b_\Gamma\rangle, V_F\big\rangle\Big)\,dS. \end{aligned} \tag{5.6}$$

Note that the first two terms $\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle\,G + \frac{\partial}{\partial n}\langle V_G, \nabla b_\Gamma\rangle\,F$ in (5.6) are symmetric in V_F and V_G, but they cannot be determined just from the restrictions $F = \langle V_F, \mathbf{n}\rangle|_\Gamma$ and $G = \langle V_G, \mathbf{n}\rangle|_\Gamma$. The term $\langle \nabla\langle V_G, \nabla b_\Gamma\rangle, V_F\rangle$ has the same nonintrinsic behavior and is not even symmetric in V_F and V_G. Now let us assume that V_F and V_G satisfy (4.6) on some neighborhood of Γ. Using this assumption, together with (3.8), we get

$$\begin{aligned} \nabla\langle V_F, \nabla b_\Gamma\rangle &= DV_F\cdot\nabla b_\Gamma + D^2 b_\Gamma\cdot V_F = D(\tilde F\,\nabla b_\Gamma)\cdot\nabla b_\Gamma + \tilde F D^2 b_\Gamma\cdot\nabla b_\Gamma \\ &= \langle\nabla\tilde F, \nabla b_\Gamma\rangle\cdot\nabla b_\Gamma + 2\langle V_F, D^2\nabla b_\Gamma\cdot\nabla b_\Gamma\rangle = 0, \end{aligned}$$

hence

$$\frac{\partial}{\partial n}\langle V_F, \nabla b_\Gamma\rangle = \big\langle\nabla\langle V_F, \nabla b_\Gamma\rangle, \nabla b_\Gamma\big\rangle = 0,$$

and, with the same reasoning,

$$\frac{\partial}{\partial n}\langle V_G, \nabla b_\Gamma\rangle = 0.$$

Thus, if we restrict our attention to perturbation vector fields of the form (4.4)–(4.6), the nonintrinsic and asymmetric terms in $d^2 J_1(\Gamma; V_F, V_G)$ vanish.

Taking all intermediate results together, we obtain the following expression for the second Eulerian derivative of J_1:

$$d^2 J_1(\Gamma; V_F; V_G) = \int_\Gamma \Big[\Big(\frac{\partial^2\phi}{\partial n^2} + 2\frac{\partial\phi}{\partial n}\,\kappa\Big)\,F\,G + \phi\langle\nabla_\Gamma F, \nabla_\Gamma G\rangle\Big]\,dS. \tag{5.7}$$

For J_2 we obtain, using Lemma 1,

$$dJ_2(\Omega; V_F) = \int_\Gamma \phi \, \langle V_F, \nabla b_\Gamma \rangle \, dS. \tag{5.8}$$

If we apply (2.19) in Proposition 2 and Lemma 4, we get

$$\begin{aligned} d^2 J_2(\Omega; V_F, V_G) = &\int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \kappa \phi \Big) \, \langle V_F, \mathbf{n} \rangle \, \langle V_G, \mathbf{n} \rangle \, dS \\ &+ \int_\Gamma \phi \, \Big(\frac{\partial}{\partial n} \langle V_F, \mathbf{n} \rangle \, G - \langle V_F, \nabla_\Gamma G \rangle \Big) \, dS. \end{aligned}$$

As in the discussion of expression (5.6), we find that the second integral is zero if V_F and V_G satisfy (4.6). We therefore get

$$d^2 J_2(\Omega; V_F; V_G) = \int_\Gamma \Big(\frac{\partial \phi}{\partial n} + \kappa \phi \Big) \, F \, G \, dS. \tag{5.9}$$

6. Gradient and Newton-type flow for variational image segmentation. In this section, we apply the results of sections 2 and 4 to cost functionals of the form (1.5). We consider a grayscale image given by its intensity map $I : \mathbb{R}^2 \to \mathbb{R}$, which assigns each point $\mathbf{x}$ its gray value $I(\mathbf{x}) \in \mathbb{R}$. For simplicity (to avoid special treatment of the boundary), we assume that the image is defined on all of $\mathbb{R}^2$. Let $\tilde{g} : [0, \infty) \to (0, \infty)$ be a given decreasing function which satisfies $\tilde{g}(r) \to 0$ as $r \to \infty$. The function $g_I(\mathbf{x}) = \tilde{g}(|\nabla I|(\mathbf{x}))$ acts as an edge detector in the sense that $g_I(\mathbf{x}) = 0$ if $\mathbf{x}$ lies on an ideal edge of I. In this paper, we use

$$g_I(\mathbf{x}) = \frac{1}{1 + \big(|\nabla I|(\mathbf{x})\big)^k} \text{ with } k = 1, 2. \tag{6.1}$$

To suppress the influence of noise, we replace ∇I in the above expression by a smoothed version $\nabla \hat{I}$. For the sake of simplicity, we use Gaussian smoothing, but other, more effective geometric smoothers (see, e.g., [3]) can be used as well. The method we propose also works for other edge detectors of the form $g_I : \mathbb{R}^2 \to \mathbb{R}$ for which $g_I \sim 0$ on edges and $g \sim c$ with $c > 0$ otherwise, provided that they satisfy the necessary smoothness requirements for performing shape sensitivity analysis of a functional of type (6.2) as exposed in sections 2–5.

Segmentation of an image is the task of partitioning a given image into disjoint parts of approximately constant gray value. Let Γ be the union of the boundaries of these homogeneous regions. Since homogeneous regions with different gray values are separated by edges, it is likely that the boundary Γ is located at points where the edge detector g_I has small values. In Figure 1 it is seen that the edges of the image coincide with the deep valley in the edge detector. This motivates the following variational approach. We seek the final segmenting contour Γ as the minimizer of the functional

$$J(\Gamma) = \int_\Gamma g_I \, dS + \nu \int_\Omega g_I \, d\mathbf{x}, \tag{6.2}$$

where $\Gamma = \partial\Omega$ and $\nu > 0$. We find the minimizer of (6.2) as the steady state of a family of propagating contours $\Gamma(t)$ which approach the minimal contour from the outside. Note that a contour of length zero (a point) is a global minimizer for (6.2). This,

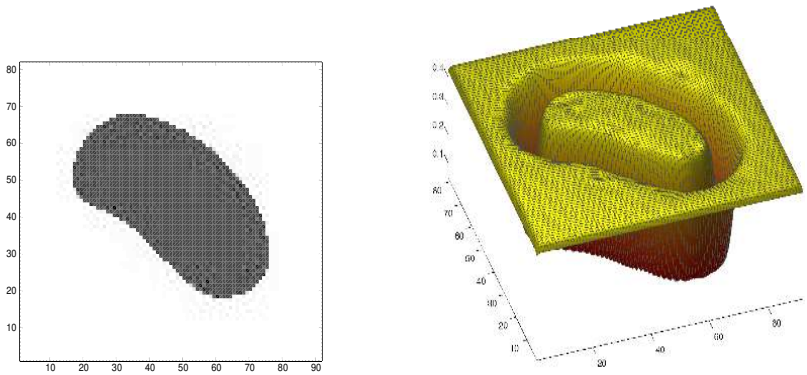

FIG. 1. *Grayscale image I and corresponding edge-map g.*

however, is not the desired segmenting contour. Rather, we want the propagation of the active contour to get stuck at the bottom of the valley of the edge detector, which is a *local minimizer* for (6.2). The second term in (6.2) is a regularization term, which helps shrink the active contour in the homogeneous regions where the influence of the edges is not very strong. The parameter ν must not be chosen too large, because otherwise the algorithm might overshoot the edge and end up at the global minimizer.

The Euler–Lagrange equation for the cost functional (6.2) with respect to the geometrical variable Γ was derived, e.g., in [8, 37] using parametrized curves $\Gamma = \Gamma(s,t)$, where s denotes the curve parameter and t is a time variable describing the movement of the curve. We derive the same result applying the speed method described in section 2. Applying Lemma 3 and (2.5b) in Lemma 1 immediately yields

$$dJ(\Gamma;V) = \langle D_\Gamma J, V\rangle = \int_\Gamma \left\langle \left(\frac{\partial g_I}{\partial n} + g_I\,(\kappa+\nu)\right)\mathbf{n}, V\right\rangle dS.$$

Thus, the flow for a contour $\Gamma(t)$ which propagates in the direction of the negative gradient with respect to the functional (6.2) is given by

$$\Gamma_t = -\big(g_I\,(\kappa+\nu) + \langle \nabla g_I, \mathbf{n}\rangle\big)\,\mathbf{n}. \tag{6.3}$$

Note that in our case $\mathbf{n}$ denotes the *exterior* normal vector to the region enclosed by Γ, so we have different signs in the expression (6.3) as, e.g., in [8, 37]. The level set formulation corresponding to (6.3) (see [35]) is given by

$$u_t = g_I\left(\operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) + \nu|\nabla u|\right) + \langle \nabla g_I, \nabla u\rangle = \left(\operatorname{div}\left(g_I\,\frac{\nabla u}{|\nabla u|}\right) + \nu\right)|\nabla u|. \tag{6.4}$$

For $\nu = 0$, (6.3) or (6.4) can be interpreted as geodesic curve-shortening flow with respect to an image-dependent metric g_I (cf. [8, 30]).

We now use the results from section 5 for the setup of a Newton-type algorithm as described in section 4, specifically in (4.8), (4.6), (4.5), and (4.4) for variational image segmentation. Using (5.7) and (5.9), we find that (4.8) for the Newton-type

speed function F has the form of an elliptic equation on Γ: Find $F : \Gamma \to \mathbb{R}$ such that

$$\int_\Gamma \Big[\Big(\frac{\partial^2 g_I}{\partial n^2} + (2\kappa + \nu) \frac{\partial g_I}{\partial n} + \nu \kappa g_I \Big) F G + g_I \langle \nabla_\Gamma F, \nabla_\Gamma G \rangle \Big] dS = - \int_\Gamma \Big(\frac{\partial g_I}{\partial n} + (\kappa + \nu) g_I \Big) G \, dS \tag{6.5}$$

for all test functions $G : \Gamma \to \mathbb{R}$.

The elliptic problem (6.5) has a unique solution, and the solution to (6.5) is a descent direction with respect to (6.2) if the bilinear form on the left-hand side is coercive on $H^1(\Gamma)$. Since $g_I > 0$ on $\mathbb{R}^2$, this is the case if

$$\Big(\frac{\partial^2 g_I}{\partial n^2} + (2\kappa + \nu) \frac{\partial g_I}{\partial n} + \nu \kappa g_I \Big) > 0 \quad \text{on } \Gamma. \tag{6.6}$$

See [36, section 2.21] for a comprehensive treatment of elliptic problems on contours. We give some heuristic arguments why condition (6.6) is likely to be satisfied in a neighborhood of the optimal contour. Let us consider the case $\nu = 0$. If the optimal contour is located at the bottom of a valley for the edge detector g_I and if the contour is approximately aligned with the direction of the valley, we have $\frac{\partial g_I}{\partial n} \sim 0$ and g_I is convex in the direction normal to the contour, i.e., $\frac{\partial^2 g_I}{\partial n^2} > 0$. Thus, (6.6) is satisfied for such a contour.

The positive definiteness of (6.5) is an important computational issue. For the actual computations, the Hessian has to be modified such that positive definiteness is maintained also for contours outside the (possibly very small) neighborhood of the optimal contour, where the convexity of g_I in the normal direction is strong enough to guarantee coercivity. This topic (among others) is addressed in the next section.

7. Implementation of an active contour algorithm for image segmentation based on the Newton-type speed function. It is often said that the speed function

$$F = -\big(g_I \kappa + \langle \nabla g_I, \mathbf{n} \rangle \big)$$

corresponds to the negative gradient direction with respect to the cost functional (6.2) with $\nu = 0$ and, therefore, propagation with this speed function decreases the cost functional as fast as possible. On the other hand, it is observed that the decrease of J along the propagation of the level set function u is not very fast and that the time steps in the numerical implementation of the level set algorithm must be chosen relatively small. Otherwise, zig-zagging trajectories for the points on the contour are observed. In the worst case, the time-stepping procedure can even become unstable. In other words, the numerical realization of the front-propagation problem (6.4) suffers from many drawbacks which are well known for gradient-based algorithms in nonlinear programming. Usually, the constant (expanding or shrinking) term, i.e., $\nu > 0$ in (6.2), is added to the speed function F to speed up the propagation. This procedure has the disadvantage that an additional parameter (the constant deflation or inflation speed) is introduced into the algorithm. It is a difficult task to choose this parameter in a reasonable way. If it is too large, it is possible that weak edges in the image are not recognized and the propagation of the contour does not stop at the edge. If it is chosen too small, the desired speed-up cannot be achieved (see Table 2).

We propose a speed-up method for the propagating interface problem which—in ideal cases—is even parameter free, i.e., $\nu = 0$ is set in all iterations. The method

can be considered as a preconditioned gradient method or, alternatively, as a Newton-type technique. As speed function F in the level set equation (1.1), we choose the Newton-type direction with respect to the cost functional (6.2) as calculated in section 5. Additionally, we use a line search technique in order to relax the restriction on the time step size in the discretization of the level set equation given by the CFL-condition. We consider the following algorithm.

ALGORITHM 1.

(1) **Initialization.** *Choose an initial (closed) contour* Γ_0. *Initialize the level set function* u^0 *such that* Γ_0 *is the zero level set of* u^0; *set* $k = 0$. *Choose a bandwidth* $w \in \mathbb{N}$ *and* $\nu \in \mathbb{R}$.

(2) **Newton direction.** *Find the zero level set* Γ_k *of the actual level set function* u^k. *Solve* (6.5) *to obtain the Newton-type direction* F^k.

(3) **Extension.** *Extend* F^k *to a band around the actual zero level set* Γ_k *with bandwidth* w *yielding* F^k_{ext}.

(4) **Update.** *Perform a time step in the level set equation with speed function* F^k_{ext} *to update* u^k *on the band. Let* $\hat{u}^{k+1}$ *denote this update.*

(5) **Reinitialization.** *Reinitialize* $\hat{u}^{k+1}$ *in order to obtain a signed distance function* u^{k+1} *with zero level set given by the zero level set of* $\hat{u}^{k+1}$. *Set* $k = k+1$ *and go to* (2).

Before we discuss steps (1)–(5) of Algorithm 1 in detail, we note that the algorithm operates only on a band around the actual zero level set (or contour) Γ_k. This so-called narrow band approach was introduced by Chopp [14]. The key aspect is the fact that typically only knowledge around the actual contour is of importance in the propagation of the contour through the level set equation (which, then, is also considered only on the band). Clearly, in our situation the shape gradient and the shape Hessian used in (6.5) are both defined only on Γ_k. Thus, the narrow band approach is appropriate. In the discrete setting, the restriction to a band around the actual zero level set reduces the computational time and the memory requirement. In [1] a fixed band is chosen with respect to the contour and then, as soon as the propagated contour approaches the boundary of the band, the band is reinitialized with respect to the actual contour. In contrast to this technique, we allow a continuously moving band, i.e., the band is moved together with the contour. This enables us to take larger time steps while preserving a low computational cost. For more details on the narrow band approach in level set methods we refer to [14, 1, 35].

Now let us discuss the steps of Algorithm 1 and the respective numerical realization.

Initialization. In the literature there exist different characteristic choices with respect to ν in the cost functional (6.2). In the following discussion, we decide to consider deflation, i.e., we choose $\nu \geq 0$ and Γ_0 such that the objects which should be segmented are within the area enclosed by Γ_0. Depending on Γ_0, a signed distance function u^0 is computed with Γ_0 as zero level set. This is done by utilizing the fast marching technique [33, 34] on the band around Γ_0 for solving the Eikonal equation

$$|\nabla u^0| = 1 \quad \text{with} \quad u^0 = 0 \text{ on } \Gamma_0.$$

Unless it is chosen too small, the algorithm is not sensitive (except for computational time) with respect to the bandwidth w.

Newton direction. This step is the core part of the new algorithm. As already mentioned in the previous section, the coercivity in $H^1(\Gamma)$ of the bilinear form in (6.5) is essential for having a well-defined Newton-type descent direction. Typically, in the

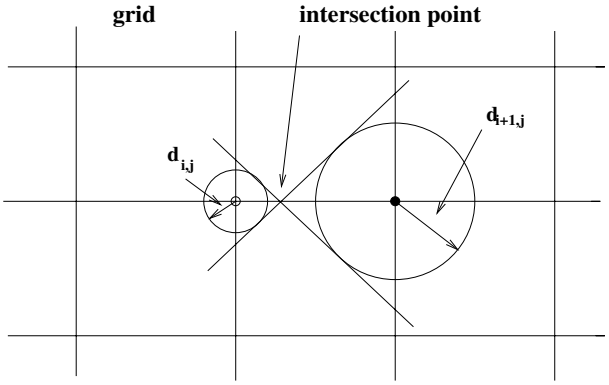

FIG. 2. *Computation of additional intersection points (on the discrete contour).*

course of the iteration it happens that

$$\frac{\partial^2 g_I}{\partial n} + (2\kappa + \nu)\frac{\partial g_I}{\partial n} + \nu\kappa g_I \leq 0 \quad \text{on some parts of } \Gamma_k.$$

Thus, the coercivity of the corresponding bilinear form is lost. To circumvent these difficulties, we incorporate the following modification of (6.5). Find $F : \Gamma \to \mathbb{R}$ such that

$$\int_\Gamma \Big[\Big(\frac{\partial^2 g_I}{\partial n^2} + (2\kappa+\nu)\,\frac{\partial g_I}{\partial n} + \nu\,\kappa\,g_I\Big)_+ F\,G + g_I\langle\nabla_\Gamma F, \nabla_\Gamma G\rangle\Big]\,dS = -\int_\Gamma \Big(\frac{\partial g_I}{\partial n} + (\kappa+\nu)\,g_I\Big)\,G\,dS \tag{7.1}$$

for all test functions $G : \Gamma \to \mathbb{R}$. Above we use $(\cdot)_+ = \max(\cdot, \epsilon)$ for $0 < \epsilon \ll 1$. In our numerical tests it turns out that frequently $\epsilon = 0$ can be set, i.e., we basically cut off the nonconvex part of the shape Hessian. For $\epsilon > 0$, equation (7.1) realizes a small correction of the Newton direction towards the steepest descent direction, i.e., the negative shape gradient.

The discretization of (7.1) is a rather delicate issue. This is due to the fact that $\langle\nabla_\Gamma F, \nabla_\Gamma G\rangle$ corresponds to the Laplace–Beltrami operator on $\Gamma = \Gamma_k$. We first need a discrete model Γ_k^h of Γ_k. For this purpose, we recall that one of the advantages of the level set method is the fact that it operates on a fixed (Cartesian) grid. In our case, the nodes are given by the pixels of the image. The information on the contour Γ_k is included in u^k, i.e., it is the zero level set of u^k. In order to get Γ_k^h we compute additional points on the grid lines (which are the lines joining the pixels of the given image) representing the discrete contour. Let us assume that $x_{i,j}$, $i = 1, \ldots, M$ and $j = 1, \ldots, N$, denote the grid points (nodes) and u_h^k is the signed distance function defined on the nodes. Whenever it is observed that $u_h^k(x_{i,j})$ and $u_h^k(x_{i+1,j})$ change sign, then the interface obviously passes through the grid line connecting $x_{i,j}$ and $x_{i+1,j}$. An analogous observation is true for $x_{i,j}$ and $x_{i,j+1}$. Since $d_{i,j} = |u_h^k(x_{i,j})|$ and $d_{i+1,j} = |u_h^k(x_{i+1,j})|$ give the distances to the contour, we compute an additional intersection point z_l^k as outlined in the graphic in Figure 2. The black node and the white node indicated different signs of u_h^k at these points. The discrete contour is given by the piecewise linear approximation joining the intersection points. For

simplicity, we temporarily assume that $\partial\Gamma_k^h = 0$ for all k and that $N_{\Gamma_k^h}$ represents the number of intersection points. Thus, Γ_k^h is a polygon. Let n_h^k denote the normal to Γ_h^k, and let $[z_l^k, z_{l+1}^k]$ represent a linear piece of Γ_k^h. For the discretization of F^k we use the ansatz

$$F_h^k(z) = \sum_{l=1}^{N_{\Gamma_k^h}} F_l^k \phi_{h,l}(z)$$

with $\phi_{h,l}$ the linear functions on Γ_k^h, which are globally continuous and satisfy $\phi_{h,l}(z_j^k) = \delta_{lj}$ for $l, j = 1, \dots, N_{\Gamma_k^h}$. We define the discretized tangential gradient $\nabla_{\Gamma_k^h}$ by

$$\nabla_{\Gamma_k^h} F_h^k(z) = \sum_{l=1}^{N_{\Gamma_k^h}} F_l^k \nabla_{\Gamma_k^h} \phi_{h,l}(z)$$

with

$$\nabla_{\Gamma_k^h} \phi_{h,l}(z) = \begin{cases} h_l^{-1} & \text{if} \quad z \in [z_{l-1}^k, z_l^k], \\ h_{l+1}^{-1} & \text{if} \quad z \in [z_l^k, z_{l+1}^k], \\ 0 & \text{else.} \end{cases}$$

Here h_l is the length of $[z_{l-1}^k, z_l^k]$; this is analogous for h_{l+1}. Note that $\nabla_{\Gamma_k^h} F_h^k$ is constant on each linear piece of Γ_k^h.

Let a_h^k denote the piecewise constant approximation of

$$a(z) = \left(\left(\frac{\partial^2 g_I}{\partial n^2} \right) + (2\kappa + \nu) \left(\frac{\partial g_I}{\partial n} \right) + \nu\, \kappa\, g_I \right)_+ (z) \quad \text{for } z \in \Gamma_k$$

with

$$a_h^k(z_l^k) = \left(\left(\frac{\partial^2 g_I}{\partial n^2} \right)_h + (2\kappa_h + \nu) \left(\frac{\partial g_I}{\partial n} \right)_h + \nu\, \kappa_h\, g_I \right)_+ (z_l^k) \quad \text{for } z_l^k \in \Gamma_k^h.$$

The approximation of the normal derivatives and the mean curvature in z_l^h, $l = 1, \dots, N_{\Gamma_k^h}$, are discussed below. Let b_h^k denote the piecewise constant approximations of g_I, which are defined as

$$b_h^k(z) = \frac{1}{2}(g_{I,h}(z_l^k) + g_{I,h}(z_{l+1}^k)) \quad \text{for } z \in [z_l^k, z_{l+1}^k].$$

For the discretization of the first term under the integral in (7.1), we use a mass lumping technique which yields a positive definite diagonal matrix. The right-hand side is approximated by utilizing the trapezoidal rule on each linear piece of Γ_k^h. The discretization of (7.1) is then given by

$$\sum_{l=1}^{N_{\Gamma_k^h}} F_l \left(a_h^k(z_l^k) h_l + \int_{\Gamma_k^h} b_h^k \langle \nabla_{\Gamma_k^h} \phi_{h,l} \nabla_{\Gamma_k^h} \phi_{h,j} \rangle \right) = \sum_{l=1}^{N_{\Gamma_k^h}} c_h^k(z_l^k) \hat{h}_l \tag{7.2}$$

for $j = 1, \dots, N_{\Gamma_k^h}$. Above, $\hat{h}_l$ is given by $\hat{h}_l = \frac{1}{2}(h_l + h_{l+1})$. In the case where Γ_k^h contains nonclosed components, $\hat{h}_l$ has to be modified on terminal linear pieces of these components.

When assembling the system matrix in (7.2) one has to be careful in order to produce a tridiagonal band matrix. In order to obtain this structure we employ the following technique. We compute a list containing all intersection points. First, we check whether one of the intersection points in the list is located on the grid lines joining the boundary pixels of the image. If this is the case, then we start with an intersection point on the boundary. In any case, we take z_l with the minimal l and follow the corresponding piece of the contour, compute the respective entries in the stiffness matrix corresponding to the actual intersection point, and delete this point from the list. If we have finished the piece of the discrete contour and the list is not empty, we repeat this procedure by checking intersection points on the boundary. If it turns out that there are no intersection points located on a boundary grid line, then we take z_l with minimal l. The final stiffness matrix is tridiagonal allowing efficient solutions of the discretization of (7.2).

For details concerning asymptotic error estimates for the finite element discretization of the elliptic equation (7.1) as described above, we refer to [17, 18].

The discretization of κ on the (fixed) grid points is based on finite differences like those in [35]. To obtain an approximation of $\frac{\partial g_I}{\partial n}$, we evaluate g_I in the grid points, compute $\nabla_h g_I$ by central differences, and compute $n_h = \frac{\nabla_h u_h^k}{|\nabla_h u_h^k|}$ as an approximation to the normal derivative in all grid points. Then

$$\left(\frac{\partial g_I}{\partial n}\right)_h (x_{i,j}) = \nabla_h g_I(x_{i,j})^T n_h(x_{i,j}).$$

Values for κ_h and $(\frac{\partial g_I}{\partial n})_h$ at intersection points are obtained as weighted averages of the respective quantity at neighboring grid points.

Extension. Since $dJ(\Gamma_k; V_G)$, $d^2J(\Gamma_k; V_F; V_G)$, and, thus, the Newton-type direction F^k are defined only on Γ_k, but the level set equation is defined on Ω (or at least on a band around Γ_k), an extension of F^k to Ω (or the band) must be computed. There exist many possible ways to extend F^k. According to (4.4), F^k_{ext} in step 3 of Algorithm 1 must satisfy

$$\langle \nabla F^k_{ext}, \nabla u^k \rangle = 0, \quad F^k_{ext}|_{\Gamma_k} = F^k. \tag{7.3}$$

On the discrete level, we realize (7.3) by employing the technique of [2]. Again, the fast marching method is used on the narrow band only. For more details we refer to [2].

Update. The discretization of the level set equation follows the standard suggestions in, e.g., [35]; i.e., the time stepping is done by using an explicit Euler scheme combined with an ENO-scheme for the term involving the spatial derivatives. Usually, the CFL-condition gives a link between the step size of the time and the spatial discretization such that the difference scheme is stable [23]. In our situation, the CFL-condition yields

$$\|F^k_{ext,h}\|_\infty \Delta t^k \leq \Delta x,$$

where Δt^k denotes the time step size in iteration k and Δx is the mesh size of the spatial discretization. Obviously, this might lead to very small time step sizes. This is especially true at early stages of the iteration process where the shape gradient is still large. Close to the discrete solution the CFL-condition becomes less stringent since $F^k_{ext,h}$ becomes "smaller."

In contrast to the requirement induced by the CFL-condition, we determine Δt^k based on considerations coming from optimization concepts. First, we relax Δt^k_{CFL}, the time step size required by the CFL-condition, by choosing a threshold $T^k := \ell \Delta t^k_{CFL}$ with $\ell > 1$. Due to our modification of the shape Hessian (its discretization induces a positive definite matrix), we expect that F^k is a local descent direction; i.e., for sufficiently small time step sizes the cost functional J is reduced by propagating Γ_k through the level set equation. A so-called sufficient decrease condition, well known from nonlinear programming [28], is given by the Armijo-condition. In our context this condition becomes

$$J(\Gamma_{k+1}) - J(\Gamma_k) \leq \mu \Delta t^k \langle F^k, dJ(\Gamma_k; F^k) \rangle < 0$$

with a fixed parameter $\mu \in (0,1)$. Numerically we realize the Armijo-condition in the following way: Let $\Gamma^h_k(\Delta t)$ denote the zero level of $u^k_h(\Delta t)$, the result of a time step with Δt in the discretized level set equation with speed function given by $F^k_{ext,h}$. At every iteration level k, we utilize the following algorithm.

ALGORITHM 2.

(1) *Set* $a_0 = \Delta t^k_{CFL}$, $b_0 = T^k$, $r_0 = b_0 - a_0$, $0 < \xi \ll \frac{1}{2}$. *Choose* $\Delta t_0 \in (a_0 + \xi r_0, b_0 - \xi r_0)$ *and* $0 < \mu_1 < \mu_2 < 1$*; set* $l = 0$. *Choose the maximal number of cycles* $L \in \mathbb{N}$.

(2) *Perform a time step in the level set equation with time step size* Δt_l *and speed function* $F^k_{ext,h}$, *and compute* $u^k_h(\Delta t_l)$.

(3) *Compute the zero level set* $\Gamma^h_k(\Delta t_l)$ *of* $u^k_h(\Delta t_l)$. *If* $l = L$, *then* $\Delta t^k = \Delta t_l$, $\hat{u}^{k+1}_h := u^k_h(\Delta t^k)$ *and RETURN to Algorithm* 1. *If* $\Gamma^h_k(\Delta t_l)$ *satisfies*

$$J_h(\Gamma^h_k(\Delta t_l)) - J_h(\Gamma^h_k) \leq \mu_2 \Delta t_l \langle F^k_h, dJ(\Gamma_k; F^k)_h \rangle < 0, \tag{7.4}$$

then $a_{l+1} = \Delta t_l$, $b_{l+1} = b_l$, $r_{l+1} = b_{l+1} - a_{l+1}$, *and compute* $\Delta t_{l+1} \in (a_{l+1} + \xi r_{l+1}, b_{l+1} - \xi r_{l+1})$. *If* (7.4) *is satisfied with* μ_2 *replaced by* μ_1 *and*

$$J_h(\Gamma^h_k(\Delta t_l)) - J_h(\Gamma^h_k) > \mu_2 \Delta t_l \langle F^k_h, dJ(\Gamma_k; F^k)_h \rangle,$$

then $\Delta t^k = \Delta t_l$, $\hat{u}^{k+1}_h := u^k_h(\Delta t^k)$ *and RETURN to Algorithm* 1. *If*

$$J_h(\Gamma^h_k(\Delta t_l)) - J_h(\Gamma^h_k) > \mu_1 \Delta t_l \langle F^k_h, dJ(\Gamma_k; F^k)_h \rangle, \tag{7.5}$$

then $b_{l+1} = \Delta t_l$, $a_{l+1} = a_l$, $r_{l+1} = b_{l+1} - a_{l+1}$, *and compute* $\Delta t_{l+1} \in (a_{l+1} + \xi r_{l+1}, b_{l+1} - \xi r_{l+1})$. *Set* $l = l + 1$ *and go to step* (2).

This step size strategy takes place in step 4 of the discrete analogue of Algorithm 1. The output (of Algorithm 2) is $\hat{u}^{k+1}_h = u^k_h(\Delta t^k)$.

Relaxing the CFL-condition significantly and computing time steps by Algorithm 2 is substantiated by the fact that we are not as interested in tracking an interface as we are in finding—as quickly as possible—a contour which locally minimizes the cost functional.

Reinitialization. Since we allow larger time steps compared to the typical choices for level set–based front propagation, the reinitialization is of importance. Like in the initialization phase, we use a fast marching technique for solving the Eikonal equation

$$|\nabla u| = 1 \quad \text{with} \quad u = 0 \quad \text{on} \quad \Gamma^{k+1}$$

numerically. Here, Γ^{k+1} is the zero level set of $\hat{u}^{k+1}$.

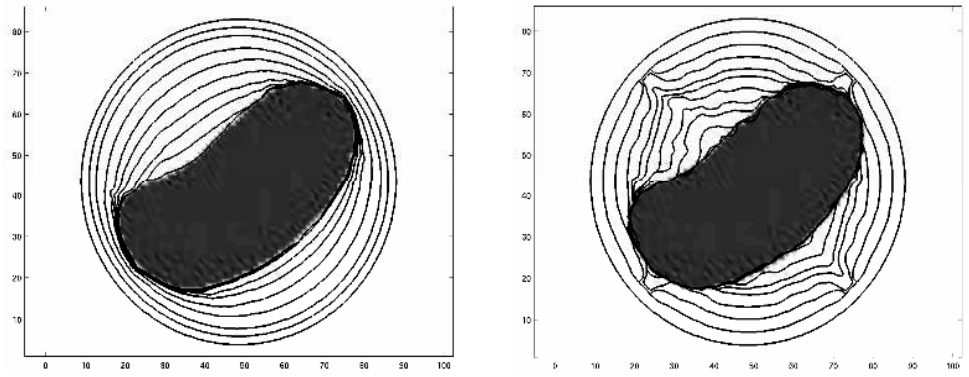

FIG. 3. *Image with zero level sets of u_h^k based on the parameter free ($\nu = 0$) Newton-type direction (left) and on the gradient-based direction with parameter $\nu = 1$ (right).*

TABLE 1
Time step sizes and cost functional values for the left graph of Figure 3.

k	Δt^k	Δt^k_{CFL}	J^k_h	$J^k_{h,r}$
1	0.00027	0.00014	67.71894	67.73983
2	0.00916	0.00458	63.62859	63.58714
3	0.05119	0.01462	55.69355	55.30486
4	0.07655	0.02187	45.59301	45.34222
5	0.11608	0.03317	37.06772	36.81020
6	0.16018	0.04577	28.19008	27.54977
7	0.20494	0.05856	16.41064	15.95286
8	0.31020	0.08862	9.73240	9.92598
9	0.34469	0.09848	4.01012	3.83231

8. Numerical results. In this section we report on numerical tests attained by Algorithm 1 for the discretization described in the previous section. With respect to the grayscales contained in the image data, the first two examples represent the ideal situation. We use these examples to demonstrate the advantages of the Newton-type direction compared to the gradient direction with deflation or inflation. Also comparisons with a method based on the negative gradient are given. The third example is related to the task of segmenting a contrast agent–based image of a kidney. Here we show that the new algorithm can handle inflation, i.e., $\nu < 0$, efficiently.

Let us start by reporting on the results for the image in Figure 3. Table 1 displays the time step sizes Δt^k accepted by Algorithm 2, the corresponding CFL-based time step Δt^k_{CFL}, the cost value J^k_h prior to the reinitialization, and $J^k_{h,r}$ after the reinitialization for Algorithm 1. Moreover, $\nu = 0$ is chosen. From Table 1 we can see that the Newton-type method stops after 9 iterations. The time step sizes Δt^k and Δt^k_{CFL} are increasing, which is expected since F^k_h should ideally vanish at a local solution. Also, our step size rule (Algorithm 2) yields significantly larger time steps than obtained from the CFL-condition. The cost functional is monotonically decreasing, and the reinitialization has only a slight influence on the cost functional value. If the speed function is changed from the Newton-type direction to the gradient direction with $\nu = 0$, then the algorithm (with step size strategy) needs 327 iterations (instead of 9 iterations for the Newton-type direction) to reduce the objective value from $J^1_h = 66.8179$ to $J^{327}_h = 3.6318$. If we allow a constant deflation by choosing

TABLE 2
Comparison of algorithms.

	Newton $\nu = 1$ with Alg. 2	Newton $\nu = 0$ with Alg. 2	Gradient $\nu = 1$ with Alg. 2	Gradient $\nu = 1$ no Alg. 2	Gradient $\nu = 0$ with Alg. 2
# it.	8	9	13	31	327

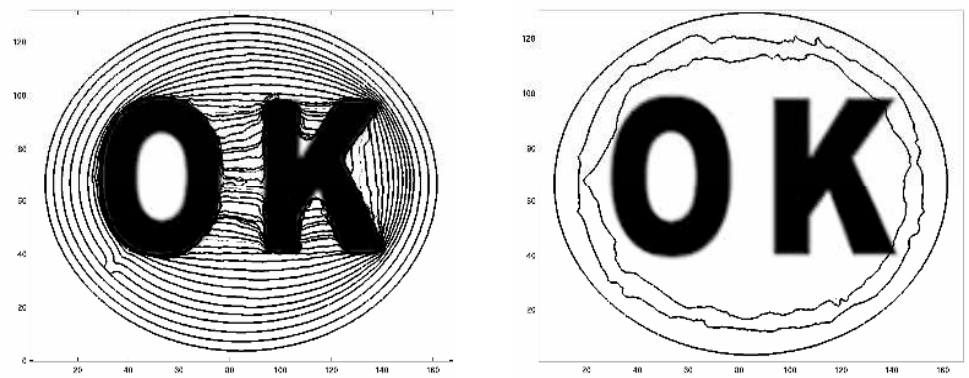

FIG. 4. *Image with zero level sets of u_h^k based on the Newton-type direction with $\nu = 0.01$ (left) and on the gradient-based direction with parameter $\nu = 1$ (right).*

$\nu = 1$ in the gradient-based method, then 13 iterations are needed. The fact that ν has to be chosen appropriately in order to avoid overshooting the desired contour or a slowly converging algorithm is a clear disadvantage. We also ran the algorithm with the gradient-based speed function with $\nu = 1$ and no step size strategy; i.e., $\Delta t^k = \Delta t^k_{CFL}$ was chosen. Then 31 iterations are needed for finding the local minimum numerically. In our test runs we also observe that the Newton-type direction acts more globally than the gradient direction. In fact, in the right graph of Figure 3 we can observe that the gradient direction detects certain parts of the contour rather quickly, while it takes some time to correctly detect the nonconvex part of the contour. The Newton-type direction yields a rather global propagation of the zero level set towards the desired contour; i.e., in the detection process (evolution of the zero level set of u_h^k) the zero level sets approach the desired contour more uniformly; see the left graph of Figure 3. Also, the contours based on the gradient-type propagation are less regular than the contours obtained from the Newton-type propagation. This behavior does not depend on our preference for employing Algorithm 2. It merely reflects our theoretical findings, i.e., computing F^k as the solution of the elliptic equation (7.1) induces additional smoothness properties of F^k. In Table 2 we summarize the convergence behavior.

Our second example is concerned with the segmentation of the letters "O" and "K" as displayed in Figure 4. Besides the aspect that our initial contour has to split into two disjoint contours, it is interesting to investigate how the Newton direction copes with the contours of "K" which involve, e.g., rather acute angles and specific nonconvexities. We shall also see that the appropriate choice of ν is a delicate issue for the gradient method. This is due to the fact that we have to balance the two objectives of fast progress and accurate segmentation. For the Newton method, on the other hand, a rather small value for ν already gives good progress without degrading the

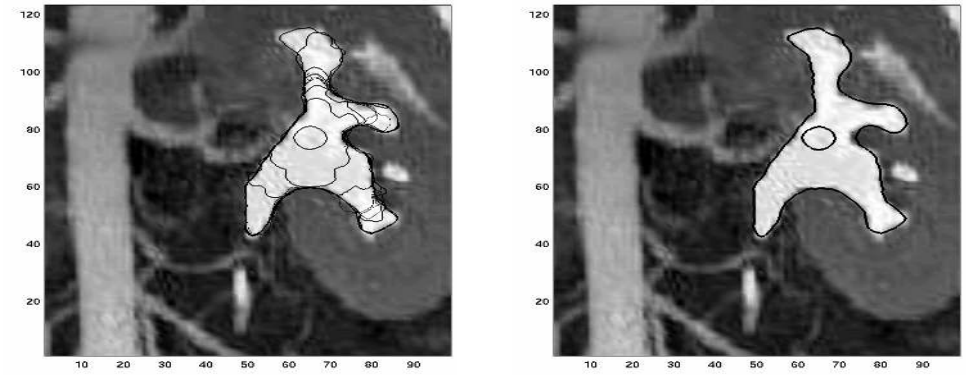

FIG. 5. *Image with zero level sets of u_h^k based on the Newton-type direction with $\nu = -0.3$ (left) and the corresponding initial and final contours (right).*

segmentation behavior.

The gradient-based direction with constant deflation ($\nu = 1$) with or without Algorithm 2 typically overshoots the right uppermost and lowermost corners. As a consequence, the segmentation misses the "K." For smaller values for ν the convergence speed of the gradient-based algorithm is significantly reduced, and too small ν eventually prevents the algorithm from convergence. From the right graph in Figure 4 we can observe that, like in the previous example, the contours for the gradient-based speed function are quite irregular. This prevents the algorithm from taking larger time step sizes.

For the Newton-type speed function, we choose $\nu = 0.01$ and still get reasonable progress in every iteration (successful termination after 33 iterations) but avoid the overshooting of the corners of "K." The evolution of the contours is displayed in the left graph of Figure 4. We initialize the algorithm with the outermost ellipse which shrinks towards the convex hull of the two letters and finally collapses onto two separate contours. With the same value for ν, the gradient-based algorithm with a step size strategy needs more then 100 iterations.

The final example is concerned with the task of segmenting a contrast agent–based image of a kidney. We initialize the algorithm by choosing a small circle inside the part of the image which we aim to segment. Thus, ν has to be assigned a negative value in order to allow inflation of the initial contour. In the left graph of Figure 5 we display some of the iterates of the Newton-type method with $\nu = -0.3$. The algorithm detects the correct contour after 47 iterations. The right graph shows the initial and the final contours for the Newton-type method.

REFERENCES

[1] D. ADALSTEINSSON AND J. A. SETHIAN, *A fast level set method for propagating interfaces*, J. Comput. Phys., 118 (1995), pp. 269–277.

[2] D. ADALSTEINSSON AND J. A. SETHIAN, *The fast construction of extension velocities in level set methods*, J. Comput. Phys., 148 (1999), pp. 2–22.

[3] L. ALVAREZ, P.-L. LIONS, AND J.-M. MOREL, *Image selective smoothing and edge detection by nonlinear diffusion*. II, SIAM J. Numer. Anal., 29 (1992), pp. 845–866.

[4] G. AUBERT AND L. BLANC-FÉRAUD, *Some remarks on the equivalence between 2d and 3d classical snakes and geodesic active contours*, Int. J. Comput. Vision, 34 (1999), pp. 19–28.

[5] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing*, Springer-Verlag, New York, 2002.
[6] V. Caselles, F. Catté, T. Coll, and F. Dibos, *A geometric model for active contours in image processing*, Numer. Math., 66 (1993), pp. 1–31.
[7] V. Caselles and B. Coll, *Snakes in movement*, SIAM J. Numer. Anal., 33 (1996), pp. 2445–2456.
[8] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, Int. J. Comput. Vision, 22 (1997), pp. 61–79.
[9] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, *Minimal surfaces based object segmentation*, IEEE Trans. Pattern Anal. Machine Intell., 19 (1997), pp. 394–398.
[10] T. F. Chan, B. Y. Sandberg, and L. A. Vese, *Active contours without edges for vector-valued images*, J. Visual Commun. Image Representation, 11 (2000), pp. 130–141.
[11] T. F. Chan and L. A. Vese, *Image Segmentation Using Level Sets and the Piecewise Constant Mumford-Shah Model*, UCLA CAM report 00-14, University of California, Los Angeles, 2000.
[12] T. F. Chan and L. A. Vese, *A level set algorithm for minimizing the Mumford-Shah functional in image processing*, UCLA CAM Report 00-13, University of California, Los Angeles, 2000.
[13] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Trans. Image Process., 10 (2001), pp. 266–277.
[14] D. L. Chopp, *Computing minimal surfaces via level set curvature flow*, J. Comput. Phys., 106 (1993), pp. 77–91.
[15] M. C. Delfour and J.-P. Zolésio, *Shapes and Geometries: Analysis, Differential Calculus, and Optimization*, Adv. Des. Control 4, SIAM, Philadelphia, 2001.
[16] M. C. Delfour and J.-P. Zolésio, *Tangential calculus and shape derivatives*, in Shape Optimization and Optimal Design (Cambridge, 1999), Dekker, New York, 2001, pp. 37–60.
[17] G. Dziuk, *Finite elements for the Beltrami operator on arbitrary surfaces*, in Partial Differential Equations and Calculus of Variations, Lecture Notes in Math. 1357, S. Hildebrandt and R. Leis, eds., Springer-Verlag, Berlin, 1988, pp. 142–155.
[18] G. Dziuk, *An algorithm for evolutionary surfaces*, Numer. Math., 58 (1991), pp. 603–611.
[19] L. C. Evans and J. Spruck, *Motion of level sets by mean curvature*, I, J. Differential Geom., 33 (1991), pp. 635–681.
[20] S. Jehan-Besson, M. Barlaud, and G. Aubert, *DREAM^{2}S: Deformable regions driven by an Eulerian accurate minimization method for image and video segmentation*, Int. J. Comput. Vision, 53 (2003), pp. 45–70.
[21] S. Jehan-Besson, M. Barlaud, and G. Aubert, *Video object segmentation using Eulerian region-based active contours*, in Proceedings of the Eighth IEEE International Conference on Computer Vision, IEEE Press, Piscataway, NJ, pp. 353–361.
[22] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes; active contour models*, Int. J. Comput. Vision, 1 (1987), pp. 321–331.
[23] R. J. LeVeque, *Numerical Methods for Conservation Laws*, 2nd ed., Birkhäuser-Verlag, Basel, 1992.
[24] A. Litman, D. Lesselier, and F. Santosa, *Reconstruction of a two-dimensional binary obstacle by controlled evolution of a level-set*, Inverse Problems, 14 (1998), pp. 685–706.
[25] R. Malladi, R. Kimmel, D. Adalsteinsson, G. Sapiro, V. Caselles, and J.A. Sethian, *A geometric approach to segmentation and analysis of* 3*d medical images*, in Proceedings of the Mathematical Methods in Biomedical Image Analysis Workshop, San Francisco, CA, IEEE Press, Piscataway, NJ, 1996.
[26] R. Malladi, J. Sethian, and B. C. Vemuri, *Evolutionary fronts for topology independent shape modeling and recovery*, in ECCV—'94, Vol. 1, Lecture Notes in Comput. Sci. 800, Springer-Verlag, Berlin, 1994, pp. 3–13.
[27] R. Malladi, J. Sethian, and B. C. Vemuri, *Shape modeling with front propagation: A level set approach*, IEEE Trans. Pattern Anal. Machine Intell., 13 (1995), pp. 158–175.
[28] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[29] P. J. Olver, G. Sapiro, and A. Tannenbaum, *Invariant geometric evolutions of surfaces and volumetric smoothing*, SIAM J. Appl. Math., 57 (1997), pp. 176–194.
[30] P. J. Olver, G. Sapiro, and A. Tannenbaum, *Affine invariant detection: Edge maps, anisotropic diffusion, and active contours*, Acta Appl. Math., 59 (1999), pp. 45–77.
[31] N. Paragios and R. Deriche, *Geodesic active regions: A new paradigm to deal with frame partition problems in computer vision*, Int. J. Visual Commun. Image Representation, 13 (2002), pp. 249–268.
[32] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, Cambridge, UK, 2001.

[33] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Nat. Acad. Sci. U.S.A., 93 (1996), pp. 1591–1595.
[34] J. A. Sethian, *Fast marching methods*, SIAM Rev., 41 (1999), pp. 199–235.
[35] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.
[36] J. Sokołowski and J.-P. Zolésio, *Introduction to Shape Optimization*, Springer-Verlag, Berlin, 1992.
[37] A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, *A geometric snake model for segmentation of medical imagery*, IEEE Trans. Med. Imaging, 16 (1997), pp. 199–209.