Segmentation of Surfaces Using Active Contours

PhD thesis

by Matthias Krueger

Submitted in Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Department of Computer Science
The University of Auckland, February 2011.

Abstract

This thesis develops new techniques for segmentation of digital surfaces, based on the active contour concept. A special focus lies on 3D face data; in particular, to what extent 2D feature segmentation methods can be further developed by taking 3D information into account.

Two different approaches are pursued:

First, a fully 3D model for feature extraction from general surfaces is proposed, that generalises the standard geodesic active contour. It avoids some intrinsic limitations of existing works by employing the implicit, instead of the parametric surface representation. Experiments on synthetic and natural surfaces emphasise the significant gain in accuracy.

Second, the thesis demonstrates that 2D techniques can be adapted for the segmentation of surface data given in the form of range images. Well-known active contour schemes are incorporated into efficient and accurate algorithms for curvature-based 3D lip contour segmentation. Yet tests suggest that known methods, such as first-order ACs or graph cuts, are not suitable for 3D eye contour segmentation. To fill this gap, a novel AC algorithm is introduced that generalises existing shortest path type schemes to second-order energies while maintaining a low computational complexity. The comparison with state-of-the-art algorithms underlines that in many cases the approach yields equally good, or better results with less user interaction. The automatic variant allows for a robust and efficient segmentation of approximately circular or elliptic structures, even in noisy images with gaps in the object boundary.

Finally, the proposed algorithms are tested on a 3D face database. Curvature- and texture-based segmentation algorithms are evaluated in a comparative study on lip contours. The results clearly indicate that current 2D systems can benefit from 3D information: (i) incorporating the surface geometry into the active contour evolution improves the performance of the classic 2D GAC algorithm. This has been conjectured before, yet without empirical evidence; (ii) the curvature-based techniques provide more robust results than the texture-based one; (iii) the curvature-based methods deliver good results for about 50% of the faces where the texture-based method fails.

The thesis advances theoretical knowledge and new applications in equal measure. The results underline the versatility of the active contour concept.

'There are only two mistakes one can make along the road to truth -
not going all the way, and not starting.'

Buddha

Acknowledgments

Completing a PhD project has often been compared to a journey, and for me this holds true in the literal sense of the word. Almost five years have passed since I started working on my research proposal in Bonn, Germany. In the meantime I was lucky enough to spend about four years in beautiful New Zealand while pursuing my studies. Having returned to Europe recently I successfully defended my PhD via video link from Zurich, Switzerland.

My foremost thanks have to go to my supervisor, Dr Patrice Delmas, who has constantly accompanied, supported and encouraged me from the very beginning until the end of the process. I am also very grateful to my co-supervisor, Associate Professor Dr Georgy Gimel'farb, for always being there when needed, concerning matters as different as reviewing my manuscripts and organising the PhD defence. Both my supervisors have taught me many things about Computer Vision in general and writing down research results in particular.

All of this would not have been possible without appropriate funding in the first place. I am extremely grateful to *Education New Zealand* and the University of Auckland for trusting in my abilities and granting me doctoral scholarships. The *BuildIT* programme of the NZ Tertiary Education Commission as well as the CSGST scheme of the University of Auckland generously covered my overseas travel costs in 2008.

Many thanks to Alfonso Gastélum Strozzi for help in technical matters and just being a good mate, to Ralf Haeusler for his manifold support towards the end of my studies, to my former supervisor Prof. Dr. Harald Garcke for the hint how to tackle the proof of Theorem 14, and to the Bejing University of Technology (BJUT) for providing their 3D face database [bju05].

I also wish to thank my parents and family for their unconditional support during the last four years and at any stage of my life so far. And last but definitely not least I am deeply grateful to my wife Dessislava for her love and patient support throughout my PhD project. Without her encouragement I probably would not even have attempted to fulfill this long lasting dream of mine. Moreover, she essentially contributed to this thesis by untiring and relentless proofreading of the manuscript.

Contents

List of Figures

List of Tables

List of Symbols

Notation	Description
$\mathscr{D}$	Neighbourhood system in a graph. 29
∂_s	Arc length derivative. 12
∂	Partial derivative. xviii
ds	Infinitesimal line element. 14
f	Feature detector function. 35, 37, 38, 121, 122
$G = \langle \mathscr{V}, \mathscr{E} \rangle$	Graph with vertices $\mathscr{V}$ and edges $\mathscr{E}$. 29
Γ	Curve/contour. 11
Id	Unit matrix. 34
I	Image. 33
J	Parametrisation interval of a curve. 11
κ_g	Geodesic curvature. 13
κ	Curvature. 12
M	Surface. 12
N	Number of image pixels. 104
∇	Nabla (gradient) operator. xix
∇_M	Surface gradient. 13
$\hat{\nu}$	Curve unit co-normal. 13
ν	Curve/surface unit normal. 12
Ω	Rectangular image domain in $\mathbb{R}^2$. 110, 142
$P_w v$	Projection operator. 53
ϕ	Level set function. 16
$\mathbb{R}^+$	$\{x \in \mathbb{R} : x > 0\}$. 14
s	Arc length parameter. 11

Acronyms

Notation	Description
AC	Active Contour. x, 33, 40, 84, 94, 95, 131, 132, 158, 159, 166, 169
ACG	Globally Optimal Active Contour scheme. 157
ACGA	Adapted Globally Optimal Active Contour scheme. 158, 159
DA	Dijkstra's Algorithm. 29, 111
DP	Dynamic Programming. 28, 36, 42, 103, 104
ENO	Essentially Non Oscillating. 22, 57
FMM	Fast Marching Method. 23, 25, 26
GAC	Geodesic Active Contour. 34, 35, 38, 41, 47, 56, 75, 81, 90–92, 97, 109, 131, 132, 153, 156
HJ	Hamilton Jacoby. 17, 20, 23, 57
LF	Lax-Friedrich. 22, 58
PDE	Partial Differential Equation. 8, 17, 24, 38, 41, 43, 44, 53, 57, 62, 110
WENO	Weighted Essentially Non Oscillating. 22, 57, 81

Notation

A few notational conventions have to be clarified.

The Euclidean inner product in $\mathbb{R}^n$, with $\mathbb{R}$ the field of real numbers, is denoted by angle brackets, e.g.[1]

$$\langle v, w \rangle := \sum_{i=1}^{n} v_i w_i$$

for two vectors $v, w \in \mathbb{R}^n$ with components v_i and w_i, respectively. The Euclidean norm of a vector v is denoted by $|v|$. A $m \times n$-matrix A with entries $a_{ij}, i = 1, \dots m, j = 1, \dots, n$ is denoted by either of the equivalent symbols

$$A = [a_{ij}] .$$

The transpose of a matrix A is denoted by $A^\top$, the notation Id represents the unit matrix.

For convenience, the following short notation is used for partial derivatives:

$$\partial_{x_i} f := \frac{\partial f}{\partial x_i} ,$$

where $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ is a differentiable function of n variables. The same notation applies to vector valued functions, e.g. curves, and other variables, e.g. the time variable t:

$$\partial_t \Gamma := \frac{\partial \Gamma}{\partial t}$$

for a time-dependent, parametrised curve $\Gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^2, (r, t) \mapsto \Gamma(r, t)$. Higher order derivatives are written analogously, e.g.

$$\partial_{ss} \Gamma := \frac{\partial^2 \Gamma}{\partial s^2} .$$

Given a multidimensional scalar function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, the gradient- or nabla operator ∇ is defined by

$$\nabla f := (\partial_{x_1} f, \partial_{x_2} f, \dots, \partial_{x_n} f)^\top . \tag{1}$$

[1] The symbol ':=' means that the left- and right-hand notations are equivalent by definition.

Being applied to a vector field $F : \mathbb{R}^n \to \mathbb{R}^n$ with component functions F_i, the nabla operator computes the divergence, i.e.

$$\nabla \cdot F := \sum_{i=1}^{n} \partial_{x_i} F_i$$

The arc length of a regular curve $\Gamma : J \to \mathbb{R}^n$ is denoted by $\|\Gamma\|$ with

$$\|\Gamma\| := \int_J |\partial_r \Gamma| \, dr \ .$$

For convenience, the abbreviation nD is used in place of 'n-dimensional', e.g. 2D and 3D. Note, that the notation '(X.Y)' is equivalent to 'Equation X.Y'.

Chapter 1

Introduction

1.1 Computer Vision and Image Segmentation

1.1.1 Introduction

Computer Vision is a ”field of robotics in which programs attempt to identify objects represented in digitized images provided by video cameras, thus enabling robots to ’see’.” (from [EB10]).

Behind this concise definition lurks a huge area of research that has drawn considerable interest in the academic community. Partly, its attractiveness can be explained by the versatility of the applications in the digital age. Digital cameras have long replaced their analogue counterparts on the mass market, and even off-the-shelf cameras today provide images with a resolution of up to 12 megapixels. On the other end of the spectrum, high-end 3D devices, such as MRI- and CT-scanners, have brought forward the development of medical imaging into a very active Computer Vision subdomain. 3D object scanners have become affordable while achieving high planar- and depth resolution. This, in turn, has facilitated a rapid evolution of algorithms in the field of 3D human face analysis and recognition. The fact that security is a central issue of our time, has further boosted this development.

Another reason for the fascination of Computer Vision is its interdisciplinarity and the sheer diversity of mathematical techniques that can be applied. Significant developments in theoretical Computer Vision involve methods originating from differential geometry, partial differential equations, statistics, and combinatorics etc.

Typical tasks in Computer Vision include:

(a) (b) (c)

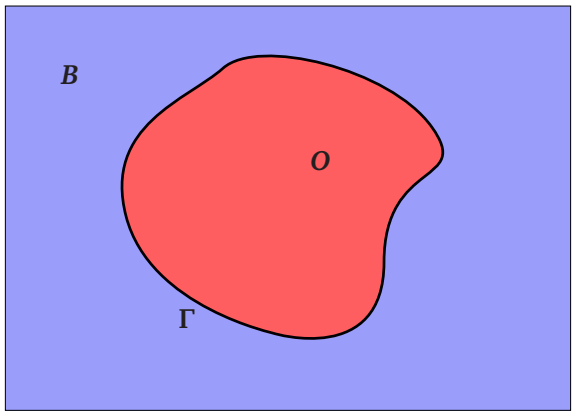
Figure 1.1: Segmentation of the image 'Damian' (a) using the graph cuts technique [BK03]: constraints (b) provided by the user, and the obtained segmentation (c); red and blue colours highlight the object and the background, respectively.

- Reconstruction of 3D data from one, two (stereovision), or several (multi-view stereo) images;
- Detection and recognition of particular objects in visual data;
- Motion estimation from a video sequence (object tracking, optical flow, etc.).

1.1.2 Image Segmentation

Segmentation algorithms play a crucial role in many Computer Vision systems. In fact, they often form the basis of the tasks listed above. The objective is to partition a visual object – a 2D- or 3D image, or surface/volume data – into multiple segments and to obtain higher level information from the raw image, such as shape, size, or position of objects. Typically, the object of interest has to be separated from the background; see e.g. Fig. 1.1. This example illustrates some key facts about image segmentation:

- The task of determining 'meaningful' image segments is ill-posed. Depending on the application, it might be desirable to segment Damian's eye- or mouth-contour instead of his overall body shape;
- Therefore, in most cases some level of user input is necessary to guide a segmentation algorithm towards the desired result (cf. Fig. 1.1(b));
- Image segmentation remains a difficult task. Despite user guidance, the result might still not be completely satisfactory (cf. Fig. 1.1(c)).



(a)

Figure 1.2: Segmentation of an image into: an object (O) and its background (B). The curve $\Gamma = \partial O$ is the object boundary.

1.2 Active Contours

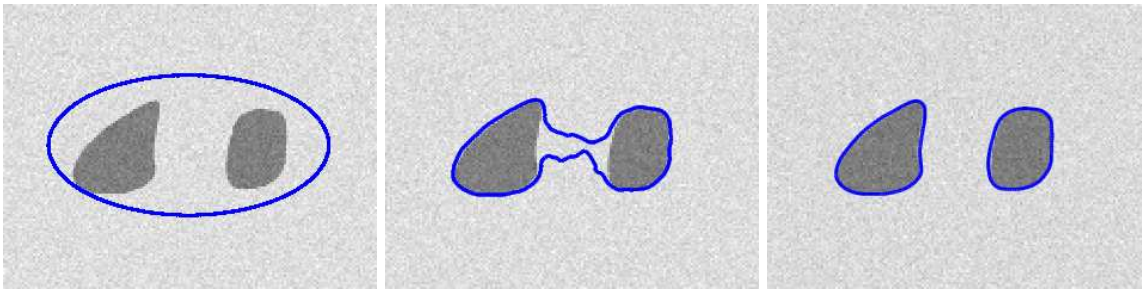
1.2.1 Active Contours for Image Segmentation

The task of image segmentation consists in assigning one of k labels to an image pixel (multiregion resp. multiphase segmentation). However, in many applications two ($k = 2$) labels are sufficient, i.e. the image has to be segmented into two phases. These phases are referred to as the object O (or foreground) and background B (cf. Fig. 1.1(c)). In the two-phase segmentation scenario there is a one-to-one correspondence between the image partition and the object boundary Γ (see Fig. 1.2(a)). Therefore, two-phase image segmentation is equivalent to the detection of the object boundary.

This notion leads to a concept that has proven highly successful: the active contour (2D)-, respectively active surface (3D) approach. Active contours (or 'snakes') were introduced by Kass et al. [KWT88]. The key idea is to evolve a curve in order to find local minima of a suitable energy functional, and thus localise the object of interest. Kass et al. [KWT88] suggested the functional

$$\mathscr{E}(\Gamma) = \int_{[0,1]} E_{int}(\Gamma) + E_{ext}(\Gamma)\,, \tag{1.1}$$

where the internal energy E_{int} imposes smoothness on the contour Γ, while the external energy E_{ext} attracts it to desired image features, such as lines or edges, and $[0,1]$ is the parametrisation interval of the curve. The model will be discussed in more detail in Chapter 3, yet the classic active contour energy (1.1) illustrates clearly the principle of active contours.



(a) (b) (c)

Figure 1.3: Segmentation of a synthetic image using geodesic active contours [CKS97]: the contour initially placed by the user (a) evolves (b) and converges after splitting in two (c).

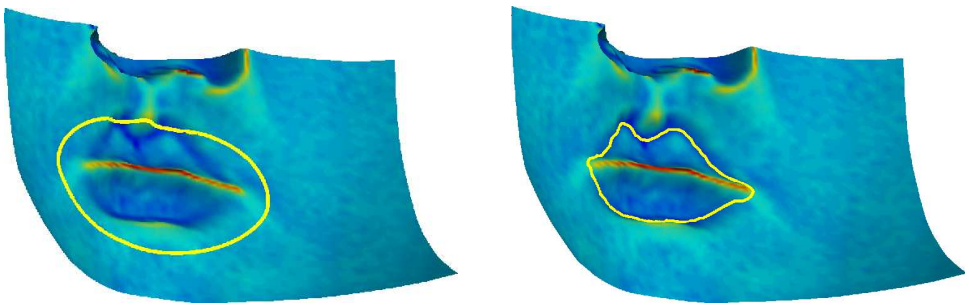
The energy is minimised by a contour that aligns well with the image features of interest while maintaining smoothness. The fine balance between the two energies is controlled by certain parameters.

Since the energy (1.1) is not convex and may possess several local minima, Kass et al. [KWT88] considered the gradient flow of (1.1). The user places an initial contour close to the object of interest, which then evolves until it converges towards a local minimum of (1.1). The concept of the classic active contours for image segmentation was very successful due to its simplicity and efficiency. However, the snakes have two significant disadvantages: the dependency on the chosen parametrisation, as well as the inability to change their topology. These aspects triggered the next major step in the development of the active contour theory: geodesic active contours (GAC) [CKS97]. The essential new idea was to represent the contour implicitly as the zero level set of a 2D function. The level set framework, developed in the pioneering work [OS88] by Osher and Sethian, allows for splitting and joining of curves (see Fig. 1.3), and flow equations stemming from geometric functionals can be easily implemented. Caselles et al. [CKS97] proposed the following energy functional:

$$\mathscr{E}(\Gamma) = \int_{\Gamma} f(s)\, ds\,, \tag{1.2}$$

where s denotes the arc length parameter and f is a stopping function similar to the external energy in (1.1). Note, that opposed to (1.1), Equation (1.2) represents a geometric functional, as the respective integral no longer depends on the curve parametrisation.

Subsequently, active contours theory remained an area of strong interest for the Computer Vision community. Important advances include the piecewise globally optimal active contours [CK97], Chan and Vese's active contours without edges [CV01], and graph cut based active



(a) (b)

Figure 1.4: Example of surface segmentation using GACs proposed in Chapter 4: initial (a) and final (b) contours; the face surface is colorised with its mean curvature.

contours [BK03]. A detailed review of active contours theory is given in Chapter 3.

1.2.2 Active Contours for Segmentation of Surfaces

With the growing affordability of 3D acquisition devices, such as 3D laser scanners, the field of 3D data processing has become very popular over the last decade. Considering the popularity of active contours for image segmentation, it is not surprising that several attempts have been made to generalise the method to surface objects (cf. Fig. 1.4). In the Computer Graphics community, an early snakes algorithm based on dynamic programming [WS92] was used on triangulated surfaces for extracting features from meshes [MBV97, JK04]. A more sophisticated approach, enabling snake evolution beyond mesh edges, was proposed by Bischoff et al. [BWK05]. Yet each snake node in this algorithm has to be traced separately, which leads to limitations for the evolution velocity field (see Section 4.2 for a detailed literature review). Later, Spira and Kimmel [SK07] generalised the GAC model to curved surfaces. Section 4.3 reviews and analyses this approach in depth.

1.3 Face Analysis

One of the central application areas for surface data processing is 3D face analysis, including face recognition and biometrics. Recent studies [CBF05, BCF06] have proven that the incorporation of 3D information significantly improves the performance of strictly 2D face recognition- and biometric systems. Nevertheless, it has not been researched yet, how face feature contour segmentation can benefit from the analysis of 3D data.

The following subsection gives a summary on the state of the art in feature segmentation from 2D- and 3D face data, respectively.

1.3.1 2D Face Feature Segmentation

AC methods have been used extensively for face feature contour segmentation from 2D images in the past (e.g. [DEL02, KCK07, SB07]). Potential applications include facial emotion recognition, audio-visual communication systems, and lip-reading. The approaches used can roughly be classified into three categories:

- Active contours (e.g. [DEL02, KCK07, SB07]);
- Active shape models [CTCG95] (e.g. [LTB96, WLN05]);
- Parametric models (e.g. [TKC00, HECC06]).

There are also hybrid approaches (e.g. [ECC04]) that blend techniques from multiple categories.

All three categories have strengths and limitations. Active contours are flexible and thus provide accurate and realistic results. However, in presence of noise and difficult illumination conditions they can lack the necessary robustness. On the other end of the spectrum, active shape based methods are efficient and robust, yet they require a comprehensive statistical training and often deliver only rough approximations of the actual contours. Parametric models can be considered as a compromise between active contours and statistical methods, since splines with a limited degree of freedom are optimised. A priori knowledge about the feature of interest is incorporated into the model. Promising results were obtained with this approach already [TKC00, ECC04, HECC06]. On the other hand, accurate localisation of several key points prior to the spline fitting step is crucial for satisfactory segmentation results. This is an important drawback for noisy data and images of poor quality.

1.3.2 3D Face Analysis

As opposed to 2D, the available 3D face data is mostly static. Usually, 3D face databases, such as the BJUT DB [bju05], contain one or several datasets per person, differing in pose or face expression.

Accordingly, the applications for 3D face feature segmentation differ from those for 2D:

- 3D face recognition and biometrics;
- Face anthropometry, i.e. the measurement of face features and related distances;
- 3D face modeling (e.g. [GDM$^+$08, GKM$^+$08]).

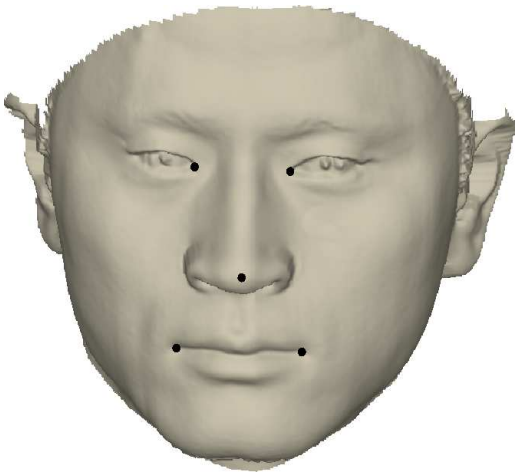


Figure 1.5: 3D face dataset (from the BJUT database [bju05]) with manually defined feature points (mouth corners, nosetip and inner eye corners).

Tracking feature contours in 3D data has a growing future potential (cf. [WDG08]). Yet known works (e.g. [CSJ05, CRAC06]) are restricted to the localisation of key points from 3D data (similar to Fig. 1.5).

1.4 Research Questions

Active contours are a focal area of research. Consequently, the thesis aims to advance the theoretical knowledge as well as new applications for this segmentation method. More specifically, I will investigate how active contour techniques can be used to segment surface data. Due to its key importance 3D face feature segmentation is the red thread throughout the thesis.

The following research questions are discussed:

- Is there a natural generalisation of the geodesic active contour model to curved surfaces, and if yes, which is it?
- Can face feature segmentation be improved by incorporating the face surface geometry into the evolution process?
- Is it feasible to design a face feature contour segmentation system based only on surface shape data and curvature characteristics?
- How does curvature-based segmentation perform compared to well-known texture-based approaches?

Additionally, the following question emerged in the course of research:

- Is there an efficient algorithm able to deliver at least approximately globally optimal second-order ACs?

1.5 Outline of the thesis

The remainder of the thesis is structured as follows: Chapter 2 reviews some important mathematical and algorithmical basics. These include numerical aspects of partial differential equations (PDE) necessary for treating level set equations, as well as an overview of some discrete optimisation algorithms, such as Dijkstra'a shortest path algorithm [Dij59].

In Chapter 3 the major contributions from the field of active contours are discussed in detail. Hereby, the focus lies on edge-based-, rather than on region-based techniques. It has to be pointed out that the state of the art in the areas of the main contributions is reviewed separately in the Chapters 4 and 6.

Chapter 4 focuses on generalised GAC models for curved surfaces. A thorough study of the existing approaches employing parametric surface representations [SK07, BBTV07] reveals an

intrinsic drawback of these methods. Namely, a balloon term cannot be incorporated due to its differing scale. The suggested remedy is to use the implicit surface representation which exploits the natural embedding of a surface into the surrounding Euclidean space. In order to reduce the increased computational complexity of the approach, a generalisation of a known 2D narrow band scheme is proposed. The experimental section underlines that a balloon term can be incorporated into the novel 3D GAC approach without the risk of obtaining diverging results. Furthermore, its applicability to a wide range of surfaces – from vascular surfaces in medical imaging to 3D head and face data – is emphasised.

The experiments with 3D face data provide promising results, in particular for lip contour segmentation. Despite the optimised algorithm, the 3D GAC method still involves considerable processing time power, which might be prohibitive for certain applications. Based on these findings, more efficient alternatives to the fully 3D approach are further investigated.

Chapter 5 illustrates that 3D surface segmentation does not necessarily require 3D techniques, such as the surface GAC introduced in Chapter 4. It is shown that well-known 2D active contour techniques can segment 3D data, when the surface is given as a range image. A new algorithm for lip contour segmentation, that incorporates parabolic shape priors into an efficient active contour scheme ([CK97]), is developed. Yet tests suggest that known methods, such as first-order ACs or graph cuts, are not suitable for 3D eye contour segmentation.

Motivated by the analysis in Chapter 5, a novel second-order AC framework for 2D images is designed in Chapter 6. It is based on the classic elastica that were first investigated by Euler [Eul44]. Two new segmentation energies are proposed, both can be considered as weighted elastica-, or second-order active contour energies. A proof is given that these energies – in contrast to the classic GAC energy (1.2) – have non-trivial global minima. Subsequently, an algorithm for the efficient approximation of the global minimiser is suggested, based on a new bidirectional Dijkstra search scheme for second-order energies. This algorithm can be used for either interactive-, or fully automatic segmentation. The comparison with state-of-the-art algorithms underlines that in many cases the approach yields equally good, or better results while less user interaction is required. The automatic variant allows for a robust and efficient segmentation of approximately circular or elliptic structures, even in very noisy images with gaps in the object boundary.

Chapter 7 evaluates the techniques developed thus far on a 3D face database. A classic 2D segmentation technique is compared with the proposed 2.5D- and fully 3D approaches.

Finally, in Chapter 8, the main conclusions are drawn, and future directions of this work are outlined.

1.6 Original contributions

To the best of the author's knowledge, the thesis presents the following original and significant contributions:

Theoretical and algorithmical aspects:

1. Geodesic active contours on curved surfaces are studied comprehensively in Chaper 4. A new approach is developed that overcomes the intrinsic drawbacks of previously published works. In addition, a well-known numerical optimisation scheme is extended to the 3D surface scenario. The proposed algorithm naturally generalises the successful GAC scheme to curved surfaces, since it is fully implicit, and a balloon force can be readily incorporated. Furthermore, the method is applicable to virtually all surfaces that can appear in applications;

2. A novel approach to 2D image segmentation is developed in Chapter 6. It is based on a new energy specification related to both, the classic GAC energy and Euler's elastica. It is proven that in contrast to the classic GAC the proposed second-order active contours possess a non-trivial global minimum in suitable function spaces. An efficient algorithm is designed to approximate the global minima. The new framework allows for both user-guided and automatic segmentation.

Applied aspects:

1. The thesis elaborates on face feature segmentation from 3D data. A fully 3D approach is presented in Chapter 4, and Chapter 5 proposes a more practical 2.5D approach, namely the segmentation of features from range images using well-known 2D techniques;

2. The algorithm for lip segmentation from Chapter 5, in conjunction with the novel second-order AC approach developed in Chapter 6, form a fully curvature-based automatic system for 3D face feature contour segmentation (see [KDG09]);

3. A thorough evaluation of 2D, 2.5D and 3D techniques is presented in Chapter 7. This includes a quantitative comparison of texture- and curvature-based approaches.

Chapter 2

Mathematical Foundations

2.1 Energy Minimisation and Gradient Flows

2.1.1 Some basics from differential geometry

Definition 1 (Regular curves)**.** *A differentiable curve* Γ *is given by a differentiable map* $\Gamma : J \to \mathbb{R}^n$, $r \mapsto \Gamma(r)$*, defined over the parameter interval* J*. It is further called regular if* $\forall r \in J : \partial_r \Gamma \neq 0$.

Definition 2 (Arc length)**.** *Let* $\Gamma : J \to \mathbb{R}^n$ *be a regular curve and* $J = [a, b]$*. The arc length* L *of the curve is given by* $L = \int_a^b |\partial_r \Gamma| \, dr$*. Further, the arc length map* $s : J \to [0, L]$ *is defined by*

$$s(r) = \int_a^r |\partial_r \Gamma(\tau)| \, d\tau \, . \tag{2.1}$$

Proposition 1 (Arc length parametrisation)**.** *Let the notation be as stated above. Define* $\widetilde{\Gamma} = \Gamma \circ s^{-1}$*. Then* $\widetilde{\Gamma} : [0, L] \to \mathbb{R}^n$, $s \mapsto \widetilde{\Gamma}(s)$ *describes the same curve as* Γ *and its tangent vector has unit length, i.e.* $|\partial_s \widetilde{\Gamma}| \equiv 1$*. Consequently, for every regular curve* Γ *there is an arc length parameter* s*.*

For the proof see e.g. [dC76].

Definition 3 (Arc length derivative)**.** *Let* $\Gamma : [0, L] \to \mathbb{R}^n$ *be a regular curve parametrised by its arc length and* $G : \Gamma \to \mathbb{R}^m$ *be a scalar* $(m = 1)$ *or vector-valued* $(m \in \mathbb{N}, m > 1)$ *function on* Γ*. Then the arc length derivative of* F *in* $s_0 = \Gamma(r_0)$ *is defined by*

$$\partial_s G(s_0) = \partial_r \left(G(\Gamma(r)) \right)_{|r=r_0} \, . \tag{2.2}$$

The following Proposition explains how the arc length derivative is computed, if Γ is given by an arbitrary parameter r.

Proposition 2 (Arc length transformation). *Let $\Gamma : J \to \mathbb{R}^n$ be a regular curve with parameter r and $\widetilde{\Gamma} : [0, L] \to \mathbb{R}^n$ be its arc length parametrisation. Then the following transformation rule holds for the parameter derivatives:*

$$\partial_s \quad = \quad \frac{\partial_r}{|\partial_r \Gamma|}\,, \tag{2.3}$$

i.e. the arc length derivative of a function $G : \Gamma \to \mathbb{R}^m$ in $s_0 = \Gamma(r_0)$ can be computed by

$$\partial_s G(s_0) = \left.\frac{\partial_r \left(G(\Gamma(r))\right)}{|\partial_r \left(G(\Gamma(r))\right)|}\right|_{r=r_0} \tag{2.4}$$

If further $|\partial_r \Gamma|$ is constant over J (not necessarily unit length), the following formula holds for the second-order derivative derivatives:

$$\partial_{ss} \quad = \quad \frac{\partial_{rr}}{|\partial_r \Gamma|^2}\,. \tag{2.5}$$

Proof. By the definition of the arc length it is $s = \int_a^r |\partial_r \Gamma(\tau)|\, d\tau$. Differentiating s by r yields $\partial_r s = \partial s / \partial r = |\partial_r \Gamma(r)|$ which includes (2.3) (note $\partial_s = \partial / \partial s$). If $|\partial_r \Gamma|$ is constant, a further iteration of this formula yields (2.5). □

A simple corollary of Proposition 2 is a formula for the unit tangent of a curve.

Corollary 3 (Unit tangent). *Let $\Gamma : J \to \mathbb{R}^n$ be a regular curve with parameter r. Then a unit tangent along Γ is given by*

$$\partial_s \Gamma = \frac{\partial_r \Gamma}{|\partial_r \Gamma|}\,. \tag{2.6}$$

Proof. Apply Proposition 2 with $F = Id$. □

Definition 4 (Curvature). *Let Γ be a regular curve in $\mathbb{R}^2$ and $\nu : \Gamma \to \mathbb{S}^1$ a unit normal field. Then the curvature of Γ is defined by $\kappa := \langle \partial_{ss} \Gamma, \nu \rangle$.*

For simplicity, the following definition of smooth surfaces comprises only the classic case of 2D surfaces in $\mathbb{R}^3$.

Definition 5 (Smooth surfaces). *A smooth surface M in $\mathbb{R}^3$ is given by a differentiable map $X : \Omega \to \mathbb{R}^3, (u_1, u_2) \mapsto X(u_1, u_2)$, where $\Omega \subset \mathbb{R}^2$ is the parameter domain. X is called a parametrisation of M.*

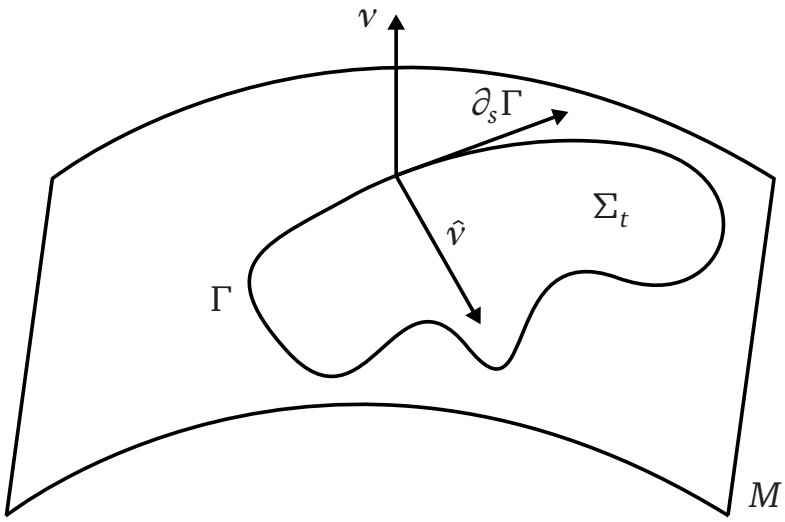


Figure 2.1: Geometry of a curve on a surface.

Definition 6 (Tangent space)**.** *Let $M \subset \mathbb{R}^3$ be a smooth surface and $x = X(u_1, u_2)$ be a point on M. Then, the vectors $\partial_{u_1} X(u), \partial_{u_2} X(u) \in \mathbb{R}^3$ span the tangent space $T_x M$ of M at x.*

Definition 7 (Surface gradient)**.** *Let $M \subset \mathbb{R}^3$ be a smooth surface and $g : M \to \mathbb{R}$ be a differentiable function on M. Then the surface gradient of g in a point $x \in M$ is defined as the tangent vector $\nabla_M g(x) \in T_x M$ that satisfies $\partial_r \left(g(\gamma(r))\right)_{|r=0} = \langle \nabla_M g(x), \partial_r \gamma(0) \rangle$ for all differentiable curves $\gamma : (-\varepsilon, \varepsilon) \to M$ with $\gamma(0) = x$.*

The following definition is essential for the analysis of curves on surfaces.

Definition 8 (Geodesic Curvature)**.** *Let $M \subset \mathbb{R}^3$ be a smooth hypersurface and $\Gamma : J \to M$ a regular curve on M. Further, let ν denote a unit normal field on M and $\hat{\nu}$ the co-normal to Γ on M (see Fig. 2.1). Then the geodesic curvature κ_g is defined as the length of the tangential component of the curvature vector $\partial_{ss}\Gamma$.*

Remark: Note, that $|\partial_s \Gamma|^2 = \langle \partial_s \Gamma, \partial_s \Gamma \rangle = 1$ implies $\langle \partial_s \Gamma, \partial_{ss} \Gamma \rangle = 0$, thus the curvature vector $\partial_{ss}\Gamma$ is orthogonal to Γ. The following decomposition holds (cf. Fig. 2.1):

$$\partial_{ss}\Gamma = \kappa_g \hat{\nu} + \kappa_n \nu \tag{2.7}$$

with κ_g the geodesic and κ_n the normal curvature.

2.1.2 Calculus of variations and gradient flows

Definition 9 (Families of curves)**.** *A map $\Gamma : J \times (-\epsilon, \epsilon) \to \mathbb{R}^2$, $(r, t) \mapsto \Gamma_t(r)$ is called a family of curves. The velocity field V is defined by $V = \partial_t \Gamma_t$.*

Equally, given a velocity field V and a curve Γ_0, a flow can be defined by the partial differential equation (PDE)

$$\partial_t \Gamma = V \quad \text{and} \tag{2.8}$$

$$\Gamma(.,0) = \Gamma_0 \tag{2.9}$$

Epstein and Gage [EG87] proved the following theorem:

Theorem 4 (Epstein/Gage)**.** *Consider the PDE (2.8) with the initial condition (2.9). Further, assume a decomposition of the velocity field into $V = V_n \cdot \nu + V_t \cdot \mathbf{t}$, where $\mathbf{t}$ is the unit tangent field along the curve. The flow (2.8) is then equivalent to $\partial_t \Gamma = V_n \cdot \nu$.*

Due to Theorem 4 such flows are completely determined by the normal component of the velocity field, whereas the tangential component corresponds to a mere change of parametrisation.

The following theorem is a generalisation of the arc length variation formula, see e.g. [KKO+96] for the proof.

Theorem 5 (First variation of the weighted arc length)**.** *Let Γ be a family of curves and $f : \mathbb{R}^2 \to \mathbb{R}^+$ be a strictly positive, scalar function. Consider the weighted arc length functional*

$$E(\Gamma) = \int_\Gamma f(s)\, ds\,. \tag{2.10}$$

Then the first variation of E is given by

$$\partial_t E(\Gamma_t)|_{t=0} = \int_{\Gamma_0} \left\langle V, \nabla f - \langle \nabla f, \mathbf{t}\rangle \mathbf{t} - f\kappa\nu \right\rangle ds \tag{2.11}$$

where $\mathbf{t} = \partial_s \Gamma$ denotes the unit tangent field along Γ.

Corollary 6. *The L^2-gradient flow equation of the functional (2.10) is given by the PDE*

$$\partial_t \Gamma = \left(f\kappa - \langle \nabla f, \nu\rangle\right)\nu \tag{2.12}$$

Proof. Theorem 5 implies that $E(\Gamma_t)$ decreases as fast as possible, if V is chosen as

$$V = -(\nabla f - \langle \nabla f, \mathbf{t}\rangle \mathbf{t} - f\kappa\nu)\,. \tag{2.13}$$

Due to Theorem 4, the tangential component of V can be omitted. Plugging $V_n \cdot \nu$ into (2.8) yields (2.12). □

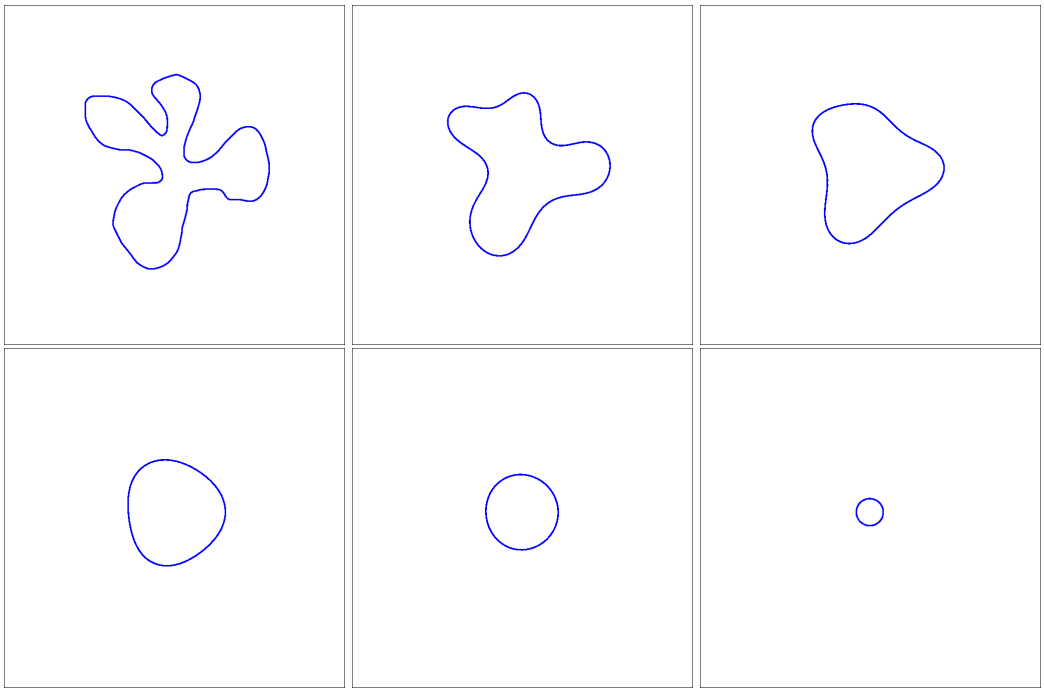


Figure 2.2: Flow of a curve by its mean curvature ((2.12) with $f \equiv 1$): the curve first becomes convex and then shrinks towards a circular point.

Plugging $f = 1$ in (2.12) yields the classic mean curvature flow (see Fig. 2.2). The following proposition is important for the motivation of balloon terms in active contour models.

Proposition 7 (Gradient flow of area functional, e.g. [SLTZ98])**.** *Let Σ_t be the respective domain enclosed by the family of curves Γ_t. Then the L^2-gradient flow equation of the area functional*

$$A(\Gamma_t) = \int_{\Sigma_t} dV \tag{2.14}$$

is given by

$$\partial_t \Gamma = \nu \,, \tag{2.15}$$

where ν is the inward-pointing unit normal field.

Proof. The divergence theorem yields

$$A(\Gamma_t) \;=\; \int_{\Sigma_t} dV = \frac{1}{2}\int_{\Sigma_t} \nabla \cdot \mathbf{x}\, dV = -\frac{1}{2}\int_{\Gamma_t} \langle \mathbf{x}, \nu \rangle \, ds \,, \tag{2.16}$$

where $\mathbf{x}$ is the identity vector field in $\mathbb{R}^2$. Plugging in $\Gamma_t = \begin{pmatrix} \Gamma_t^1 \\ \Gamma_t^2 \end{pmatrix}$ and $\nu = \begin{pmatrix} -\partial_r \Gamma_t^2 \\ \partial_r \Gamma_t^1 \end{pmatrix} / |\partial_r \Gamma_t|$,

the first variation of A is computed as:

$$\begin{aligned}
\partial_t A(\Gamma_t)|_{t=0} &= \partial_t \left(-\frac{1}{2} \int_J \left\langle \Gamma_t, \begin{pmatrix} -\partial_r \Gamma_t^2 \\ \partial_r \Gamma_t^1 \end{pmatrix} \right\rangle dr \right)_{|t=0} && (2.17)\\
&= \left(-\frac{1}{2} \int_J \left(\left\langle \partial_t \Gamma_t, \begin{pmatrix} -\partial_r \Gamma_t^2 \\ \partial_r \Gamma_t^1 \end{pmatrix} \right\rangle + \left\langle \Gamma_t, \begin{pmatrix} -\partial_{rt} \Gamma_t^2 \\ \partial_{rt} \Gamma_t^1 \end{pmatrix} \right\rangle \right) dr \right)_{|t=0} && (2.18)\\
&= -\frac{1}{2} \int_{\Gamma_0} \left(\langle V, \nu \rangle - \left\langle \partial_s \Gamma_t, \begin{pmatrix} -\partial_t \Gamma_t^2 \\ \partial_t \Gamma_t^1 \end{pmatrix} \right\rangle \right) ds && (2.19)\\
&= -\frac{1}{2} \int_{\Gamma_0} \left(\langle V, \nu \rangle - \langle \partial_t \Gamma_t, -\nu \rangle \right) ds && (2.20)\\
&= \int_{\Gamma_0} \langle V, -\nu \rangle \, ds \, . && (2.21)
\end{aligned}$$

Partial integration of the second summand in (2.18) yields (2.19). Rearranging the inner product yields (2.20), and the formula (2.15) for the gradient flow follows from (2.21). □

2.2 The Level Set Method

Due to its paramount importance for modern active contour approaches, the level set method [OS88] is reviewed here in some detail. For a comprehensive treatment of the topic see [OF03].

2.2.1 Derivation

In their pioneering work [OS88], Osher and Sethian proposed a new method for the numerical implementation of interface flow equations such as (2.8). In their *level set method* the interface (a curve or surface) is represented implicitly as the zero level set of a higher dimensional function. Thus, topological changes during the evolution process are handled in a natural and elegant manner.

The basic idea is the following: let Γ_t be a family of curves as previously defined in this chapter. Now, Γ_t is embedded into a higher dimensional scalar function $\phi : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ such that

$$\forall t : \Gamma_t = \{ x \in \mathbb{R}^2 : \phi(x,t) = 0 \} \, , \tag{2.22}$$

or equivalently

$$\forall t : \phi(\Gamma_t, t) = 0 \, . \tag{2.23}$$

Taking the time derivative of (2.23) yields

$$0 = \partial_t \left(\phi(\Gamma_t, t) \right) = \langle \nabla \phi, \partial_t \Gamma_t \rangle + \partial_t \phi \, , \tag{2.24}$$

which in turn implies the PDE for the level set function ϕ, if a velocity field V is given:

$$\partial_t \phi = -\langle V, \nabla\phi \rangle \,. \tag{2.25}$$

In terms of the scalar normal velocity V_n this equation can be rewritten as

$$\partial_t \phi = -\langle V_n \nu, \nabla\phi \rangle = -\left\langle V_n \frac{\nabla\phi}{|\nabla\phi|}, \nabla\phi \right\rangle = -V_n |\nabla\phi| \,. \tag{2.26}$$

2.2.2 Basic finite difference schemes

For simplicity, the case of evolving curves is considered here. This corresponds to a 2D level set function ϕ with an additional time variable. However, the techniques described in this section are generalised to higher dimensions in a straightforward way.

Let $\mathscr{R} = \left\{(x_i, y_j) : 1 \le i \le m, 1 \le j \le n\right\}$ be a uniform Cartesian grid, i.e. the subintervals $[x_i, x_{i+1}]$ and $[y_j, y_{j+1}]$ all have equal size Δx and Δy, respectively. As it is normally done in Computer Vision applications, the grid resolution $\Delta x = \Delta y = 1$ is assumed.

Setting $\phi_{ij} := \phi(x_i, y_j)$ the following finite difference schemes (see e.g. [OF03]) can be used to approximate the spatial derivatives of ϕ in a grid point (x_i, y_j):

$$D_{ij}^{-x}\phi = \phi_{ij} - \phi_{i-1,j} \tag{2.27}$$

$$D_{ij}^{+x}\phi = \phi_{i+1,j} - \phi_{ij} \tag{2.28}$$

$$D_{ij}^{ox}\phi = \phi_{i+1,j} - \phi_{i-1,j} \,. \tag{2.29}$$

The operators for derivatives in y-direction are defined analogously. D^{-x} and D^{+x} are called backward and forward difference operator, respectively, and – in the simple form given here – provide first-order accurate approximations of the actual derivative. The central difference operator D^{ox} has second-order accuracy. Higher-order accurate one-sided finite difference operators will be discussed in Sect. 2.2.6.

Time derivatives are usually approximated by simple Euler forward differences, for instance, (2.25) becomes

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \langle V, \nabla\phi^n \rangle = 0 \tag{2.30}$$

with $\phi^n := \phi(., t^n)$ at some time t^n and Δt the time step. Note, that (2.30) corresponds to an explicit update step, and ϕ^{n+1} is readily computed from ϕ^n.

2.2.3 Upwind differencing

Hamilton-Jacobi (HJ) equations, such as (2.25) and (2.26), are hyperbolic PDE and need to be treated numerically with great care. In the following, the principle of upwind differenc-

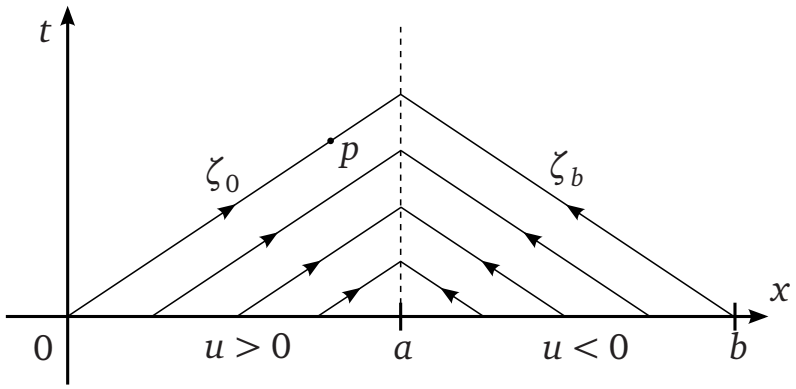


Figure 2.3: Propagation of information along characteristics.

ing is illustrated on a simple example. Then, the procedure for more general HJ equations is explained briefly.

The 1D case and characteristics Consider the 1D version of (2.25):

$$\partial_t \phi + u(x)\partial_x \phi = 0 \tag{2.31}$$

$$\text{with the initial condition:} \quad \phi(x,0) = f(x)\,. \tag{2.32}$$

Equation (2.31) is a 1D transport equation and the analytic solution can be computed readily by the method of characteristics (see e.g. [Joh91]). In fact, starting from a point x at $t = 0$ the information is transported along the time-space characteristics

$$\zeta_x(t) = \begin{pmatrix} x + u(x) \cdot t \\ t \end{pmatrix}, \tag{2.33}$$

i.e. the solution ϕ has the value $f(x)$ all along ζ_x. This is verified as follows: due to (2.33) ϕ is constructed to fulfil $\phi(x + u(x)t, t) = f(x)$. Taking the time derivative yields $\partial_x \phi \cdot u(x) + \partial_t \phi = 0$ which is (2.31). Condition (2.32) holds trivially.

Example Figure 2.3 illustrates the appearance of the characteristics for (2.31) for a simple two-valued coefficient function u given by

$$u(x) = \begin{cases} 3/2 & \text{if } x \le a \\ -3/2 & \text{if } x > a\,. \end{cases} \tag{2.34}$$

It becomes apparent that the sign of the coefficient function u determines the direction of the characteristics: a positive sign causes the information to propagate from left to right and vice versa for a negative sign. Further, the absolute value of u dictates the speed of the movement.

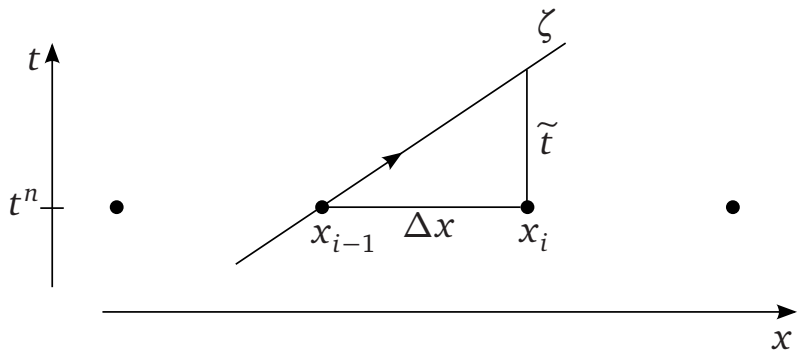


Figure 2.4: Illustration of the CFL condition: the time step Δt must not exceed $\widetilde{t}$.

Upwind derivatives The above example shows that the direction of the characteristics has to be taken into account when the derivatives are computed. Consider the discretisation of (2.31) given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + u \cdot \partial_x \phi^n = 0 \tag{2.35}$$

and assume further that ϕ^{n+1} shall be computed for a grid point x_i. This point in time and space corresponds to $p = (x_i, t^n)$ in Fig. 2.3.

If $u(x_i) > 0$ then the values of ϕ are moving from left to right and the backward difference operator D_i^{-x} has to be used to approximate $\partial_x \phi^n$ in (2.35). Likewise, if $u(x_i) < 0$ the operator D_i^{+x} has to be chosen. Picturing the characteristics running downwind, the finite differences are computed using grid points lying in upwind direction. Thus, the principle is named after this phenomenon.

Stability issues Figure 2.4 shows a window around point $p = (x_i, t^n)$ in Fig. 2.3. The variable $\widetilde{t}$ is the time that $\phi(x_{i-1})$ needs to propagate to x_i. Apparently, the time step Δt must not exceed $\widetilde{t}$: otherwise, information from grid points beyond x_{i-1} would be necessary to compute $\phi^{n+1}(x_i)$. For the 1D transport equation (2.35) this leads to the following Courant-Friedrichs-Levy (CFL) condition for the time step:

$$\Delta t < \alpha \cdot \frac{\Delta x}{\max\{|u|\}}\,, \tag{2.36}$$

where $\alpha \in (0, 1)$ is called the *CFL-number*.

Higher dimensions The method described above can easily be extended to higher dimensions. The upwind derivatives for (2.25) are then computed in a dimension-by-dimension manner. For the generalisation of the CFL condition (2.36) see e.g. [OF03].

2.2.4 Motion in normal direction

Consider Eq. (2.26) for general flows in normal direction. Solving this equation, both analytically and numerically, is more intricate than solving (2.25), since the PDE (2.26) is no longer linear.

Nonetheless, similar principles can be applied when discretising (2.26). In fact, it can be rewritten as

$$\partial_t \phi + V_n \left\langle \frac{\nabla \phi}{|\nabla \phi|}, \nabla \phi \right\rangle = 0 \,, \tag{2.37}$$

which may be perceived as an advection equation (2.25) with a velocity field pushing the curve in its normal direction. For simplicity, consider the equation in one spatial dimension:

$$\partial_t \phi + \underbrace{v_n \frac{\partial_x \phi}{|\partial_x \phi|}}_{\widetilde{u}} \cdot \partial_x \phi = 0 \,. \tag{2.38}$$

Similar reasoning as for (2.35) can be applied when deciding how $\partial_x \phi$ should be approximated in a grid point x_i. Assume $v_n > 0$, and let ϕ_x^- and ϕ_x^+ be the shorthand notation (as in [OF03]) for $D_i^{-x}\phi$ and $D_i^{+x}\phi$, respectively (see (2.27), (2.28)). If ϕ_x^- and ϕ_x^+ have the same sign, the term $\widetilde{u}$ in (2.38) inherits this sign, and ϕ_x^- or ϕ_x^+ can be chosen to approximate $\partial_x \phi$, following the *upwind* argumentation presented above. If ϕ_x^- and ϕ_x^+ have different signs, the situation becomes more complicated. This scenario corresponds to characteristics colliding ($\phi_x^- > 0$, $\phi_x^+ < 0$, see $x = a$ in Fig. 2.3) or expanding from a point ($\phi_x^- < 0$, $\phi_x^+ > 0$). Osher [OF03] suggests to use Godunov's method: this scheme sets $\partial_x \phi = 0$ for diverging characteristics, and, in the case of a collision, $\partial_x \phi$ is set to the larger in magnitude of ϕ_x^- and ϕ_x^+. This procedure is summarised by Rouy and Tourin's formula

$$\partial_x \phi^2 \approx \begin{cases} \left(\max\left\{\phi_x^-, -\phi_x^+, 0\right\}\right)^2 & \text{if } v_n \geq 0 \\ \left(\max\left\{-\phi_x^-, \phi_x^+, 0\right\}\right)^2 & \text{if } v_n < 0 \,. \end{cases} \tag{2.39}$$

Note, that the generalisation to higher dimensions is straightforward. The flow of a sample curve in normal direction is displayed in Fig. 2.5.

2.2.5 General Hamilton-Jacobi equations

The geodesic active contour model proposed in Chapter 4 includes terms that do not fit in the above schemes. Therefore, the numerical framework for general HJ equations proposed in [OS88, OF03] is briefly reviewed in this subsection. HJ equations have the form

$$\partial_t \phi + H(\nabla \phi) = 0 \,. \tag{2.40}$$

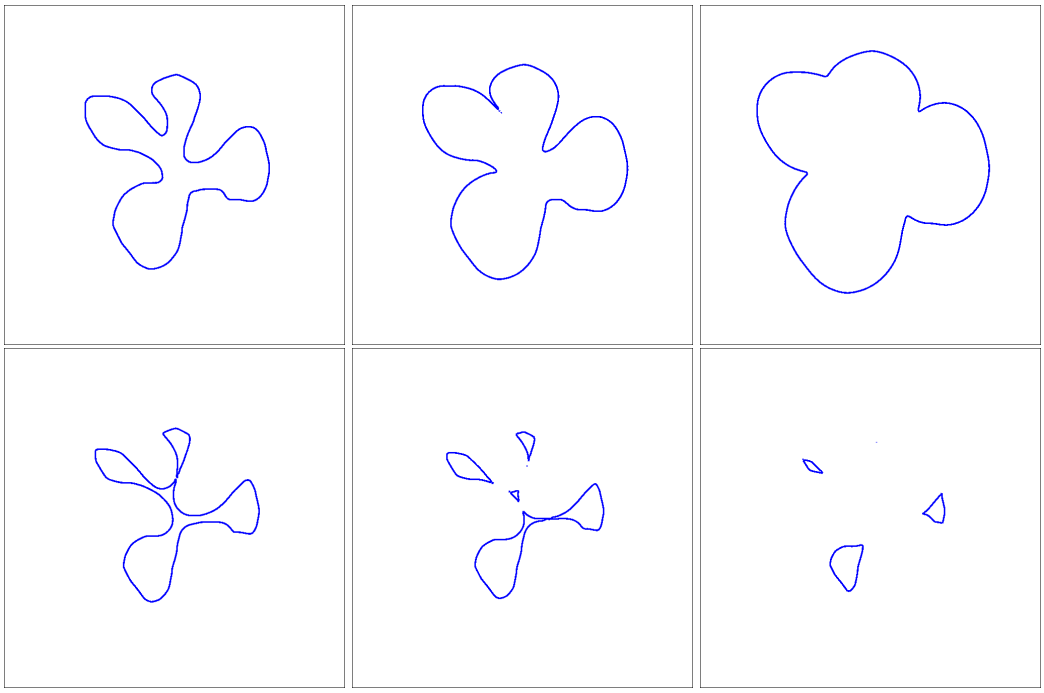


Figure 2.5: Flow of a curve in normal direction ((2.26) with $V_n = \pm 0.2$): original curve (upper left), curve moving outward (upper row), curve moving inward (lower row).

The function H, called *Hamiltonian*, may depend on x and t.

Relation to conservation laws Osher and Sethian [OS88] made the observation that the HJ equation (2.40) in one spatial dimension is closely related to hyperbolic conservation laws (see e.g. [Joh91]). Taking the spatial derivative of a 1D HJ equation, $\partial_t \phi + H(\partial_x \phi) = 0$, yields

$$\partial_t \partial_x \phi + \partial_x \left(H(\partial_x \phi) \right) = 0 \,. \tag{2.41}$$

Substituting $u = \partial_x \phi$, this equation becomes

$$\partial_t u + \partial_x \left(H(u) \right) = 0 \,, \tag{2.42}$$

which is a hyperbolic conservation equation. Performing the x derivative yields

$$\partial_t u + H'(u) \cdot \partial_x u = 0 \,, \tag{2.43}$$

which in turn is a quasilinear transport equation, similar to the linear equation (2.31). Equation (2.43) suggests that the derivatives of the Hamiltonian (such as H' in (2.43)) are crucial for the analysis of (2.40), since they determine speed and direction of the characteristics.

Numerical approximation Using an Euler forward scheme for the time step, (2.40) may be written as

$$\frac{\phi^{n+1}-\phi^n}{\Delta t}+\hat{H}^n(\phi_x^-,\phi_x^+,\phi_y^-,\phi_y^+)=0\,, \tag{2.44}$$

where $\hat{H}$ is the numerical Hamiltonian. Note, that (2.44) mirrors the 2D case and is generalised to higher dimensions in a straightforward manner.

A common approach for the numerical Hamiltonian is given by the *Lax-Friedrichs* (LF) scheme:

$$\hat{H}=\underbrace{H\left(\frac{\phi_x^+ + \phi_x^-}{2},\frac{\phi_y^+ + \phi_y^-}{2}\right)}_{I}\underbrace{-\alpha^x\left(\frac{\phi_x^+ - \phi_x^-}{2}\right)-\alpha^y\left(\frac{\phi_y^+ - \phi_y^-}{2}\right)}_{II} \tag{2.45}$$

The main idea is to use central differences to obtain an approximation of H (term I) and add numerical viscosity if the forward- and backward differences deliver differing values (term II). The amount of the added viscosity is controlled by the dissipation coefficients α^x and α^y which are defined by

$$\alpha^x = \max|H_1(\partial_x\phi,\partial_y\phi)| \tag{2.46}$$

$$\text{and}\quad \alpha^y = \max|H_2(\partial_x\phi,\partial_y\phi)| \tag{2.47}$$

with H_1 and H_2 the respective partial derivatives of H. A simple method to estimate α^x and α^y is to compute all possible values of ϕ_x^-,ϕ_x^+,ϕ_y^- and ϕ_y^+ on the grid at a certain time. Then the extremal values ϕ_x^{min},ϕ_x^{max}, ϕ_y^{min} and ϕ_y^{max} are derived. The alpha coefficients are determined by maximising $|H_1|$ and $|H_2|$ over the rectangle $[\phi_x^{min},\phi_x^{max}]\times[\phi_y^{min},\phi_y^{max}]$. Proceeding this way, uniform dissipation coefficients are estimated for the entire grid, giving the method the name *LF scheme with global flux*. The reader is referred to [OF03] for further approaches to define the dissipation coefficients and the numerical Hamiltonian.

2.2.6 Higher order ENO/WENO schemes

The forward and backward difference operators defined in (2.27), (2.28) possess only first-order accuracy. Shu and Osher [SO88, SO89] proposed higher-order ENO (essentially non oscillating) schemes. The main idea is to interpolate ϕ by higher order polynomials and to choose these with the smallest variation to compute the approximate derivatives, thus obtaining ENO schemes with second- and third-order accuracy.

Later, Jiang and Peng [JP99] proposed the WENO (weighted ENO) scheme, which picks

an optimal convex combination of the third-order ENO approximations and thus reaches fifth-order accuracy. See [OF03, SO88, SO89, JP99] for details.

2.2.7 Signed distance functions

In order to enable stable numerical finite difference approximations, it is desirable that the level set function is close to a signed distance function. There are several ways to obtain a signed distance function for a given interface: one is the fast marching method (FMM) [Set96] which is described in some detail in Section 2.3. If a level set function for the interface Γ is already given and needs to be *reinitialised* to a signed distance function, the following time-dependent HJ equation can be applied:

$$\partial_t \phi + \operatorname{sgn}(\phi(.,0)) \left(|\nabla \phi| - 1 \right) = 0 \,, \tag{2.48}$$

where $\Gamma = \{x \in \mathbb{R}^n : \phi(x,0) = 0\}$. Equation (2.48) keeps ϕ unchanged on Γ and enforces $|\nabla \phi| = 1$, the PDE for the distance function, elsewhere. Compared with the FMM, reinitialisation using (2.48) has the advantage of higher accuracy. The downside is the increased computational workload.

2.3 Fast Marching Method

Sethian [Set96] points out the close connection between the level set equation (2.26) and a certain stationary HJ equation, commonly known as Eikonal equation. Indeed, if the normal velocity is strictly positive – denoted by F instead of V_n in this special case – the propagation of an interface by the equation

$$\partial_t \phi + F|\nabla \phi| = 0 \tag{2.49}$$

can also be computed using an Eikonal equation. Let $T(x,y)$ denote the time at which the front reaches the point (x,y). Then T satisfies

$$|\nabla T| F = 1 \,. \tag{2.50}$$

Likewise, this equation can be used to compute the weighted distance function, given a weight (or cost-) function P. Then (2.50) becomes

$$|\nabla U| = P \,. \tag{2.51}$$

2.3.1 Analysis

As for the above time-dependent HJ equations, understanding the analytical properties of (2.51) is paramount for an appropriate numerical treatment. This section demonstrates how (2.51) is solved analytically in the easiest case, $P = 1$, and further how the solution behaves. Like for the level set equation (2.26), the method of characteristics is the key concept.

For simplicity, consider the prototypical Eikonal equation

$$|\nabla U(x)| = 1 \qquad \text{for } x \in \mathbb{R}^2, \tag{2.52}$$

$$U(x) = 0 \qquad \text{for } x \in \Gamma\,, \tag{2.53}$$

with Γ being a smooth curve in $\mathbb{R}^2$. By means of the Hamiltonian, $H(\nabla U) = |\nabla U| - 1$, Equation (2.52) can be written as

$$H(\nabla U) = 0\,. \tag{2.54}$$

This PDE is fully nonlinear and therefore difficult to solve. A smart approach (see e.g. [Joh91]) is to differentiate this equation, in order to obtain a quasilinear second-order equation which then can be solved by the method of characteristics. Substituting $G = \nabla U$ in (2.54) and taking the spatial derivative yields:

$$\langle \nabla_\xi H, \nabla_x G \rangle = 0\,, \tag{2.55}$$

where ξ denotes the variable of the Hamiltonian, i.e. $H = H(\xi)$. This is a quasilinear, first-order hyperbolic equation which can be solved as indicated for (2.31). For an initial point $x_0 \in \Gamma$ the following equations have to be solved:

$$\zeta'_{x_0}(s) = \nabla_\xi H(\xi_{x_0}(s)) = \frac{\xi_{x_0}(s)}{|\xi_{x_0}(s)|}, \qquad \zeta_{x_0}(0) = x_0 \tag{2.56}$$

$$\xi'_{x_0}(s) = 0, \qquad \xi_{x_0}(0) = \nabla U(x_0)\,. \tag{2.57}$$

Note, that (2.53) implies the gradient ∇U to stand perpendicular on Γ. Further, Eq. (2.52) determines its length, thus $\nabla U(x_0)$ might be either of the two possible unit normals $\nu_\Gamma(x_0)$ in x_0. One concludes $\xi_{x_0}(s) \equiv \nu_\Gamma(x_0)$ and $\zeta_{x_0}(s) = x_0 + s\nu_\Gamma(x_0)$. The vector field ξ_{x_0} constitutes the solution for G $(= \nabla U)$ along the characteristic ζ_{x_0}. The derivative of U along ζ_{x_0} is given by

$$\partial_s U(\zeta_{x_0}) = \langle \nabla U(\zeta_{x_0}), \zeta'_{x_0} \rangle = \langle \xi_{x_0}, \zeta'_{x_0} \rangle \tag{2.58}$$

$$= \langle \nu_\Gamma(x_0), \nu_\Gamma(x_0) \rangle = 1\,, \tag{2.59}$$

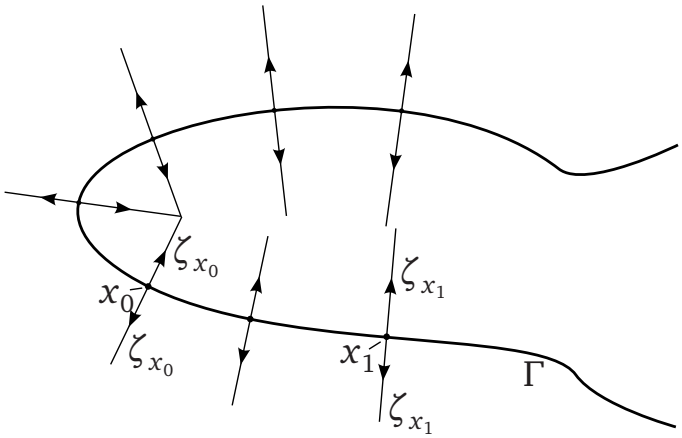


Figure 2.6: Characteristics of the Eikonal equation (2.52).

which implies $U(\zeta_{x_0}(s)) = s$. Putting all pieces together, the solution of (2.52),(2.53) is given by

$$U(x_0 + s\nu_\Gamma(x_0)) = s\,, \tag{2.60}$$

As expected, the solution U measures the distance to Γ along rays standing normal on Γ (see Fig. 2.6). Certainly, the solution in (2.60) is valid only locally in an open set around Γ. As it is common for hyperbolic PDE, the characteristics might collide and develop *caustics*.

2.3.2 Numerical scheme/FMM

The above analysis shows that the Eikonal equation – in analogy to the above level set equations – propagates information along the characteristics starting from the manifold of points with initially known values. As illustrated in Section 2.2.3, the numerical scheme for the gradient term in (2.51) has to take into account the direction of the characteristics. Again, a suitable discretisation scheme should only incorporate points lying in *upwind* direction. Due to these considerations, Sethian [Set96] proposed to use a Godunov scheme (see (2.39)) yielding

$$\begin{aligned} |\nabla U(x_i, y_j)|^2 \;\approx\; & \Big(\max\{D_{ij}^{-x}U, -D_{ij}^{+x}U, 0\}\Big)^2 \\ & + \Big(\max\{D_{ij}^{-y}U, -D_{ij}^{+y}U, 0\}\Big)^2 \,. \end{aligned} \tag{2.61}$$

To recall the basic finite difference operators used here, see Section 2.2.2.

Using the simple relation

$$\max\Big\{D_{ij}^{-x}U, -D_{ij}^{+x}U, 0\Big\} = \max\Big\{U_{ij} - \min\{U_{i-1,j}, U_{i+1,j}\}, 0\Big\}\,, \tag{2.62}$$

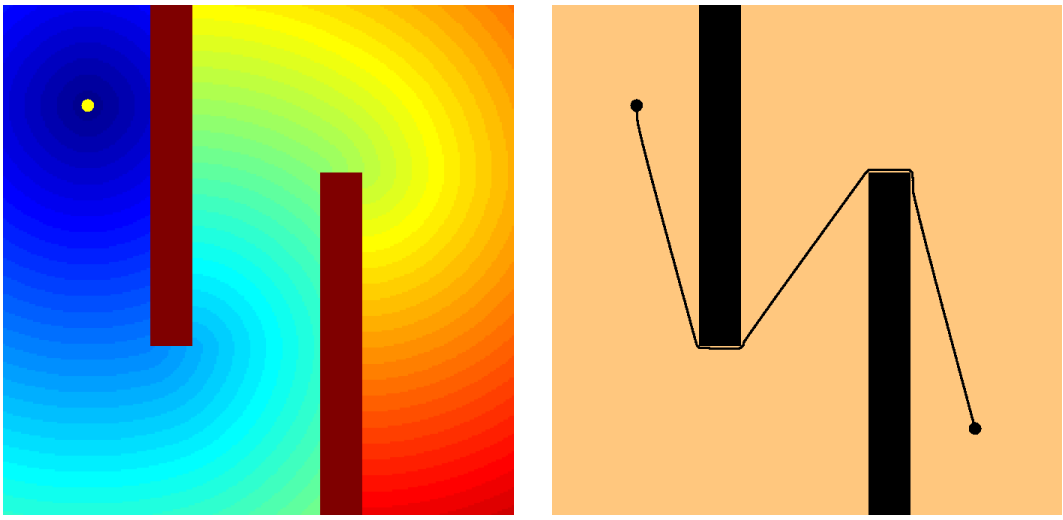


Figure 2.7: Application of the Eikonal equation (2.51) to an obstacle problem. The two bars have infinitely high cost $P = \infty$. Left: The weighted distance function U, computed using the FMM starting from the source in the upper left corner. Blue colour indicates values close to 0, red colour indicates larger values. Right: The shortest path to a destination point x_d, found by backtracking of $-\nabla U$ from x_d.

Eq. (2.51) can be transformed into its discrete counterpart

$$\big(\max\{u - U_1, 0\}\big)^2 + \big(\max\{u - U_2, 0\}\big)^2 = P^2 \,. \tag{2.63}$$

Here, $u = U_{ij}$, $U_1 = \min\{U_{i-1,j}, U_{i+1,j}\}$ and $U_2 = \min\{U_{i,j-1}, U_{i,j+1}\}$. Note, that in (2.63) the value of u at a given pixel (i, j) depends only on those neighbouring pixels that have a lower value than u. Thus, the discretisation scheme given in (2.61) uses indeed only function values in upwind direction. This observation, originally made by Sethian [Set96], allows for the efficient solution of (2.63) in one sweep, very similar to Dijkstra's algorithm [Dij59]. As in (2.53), let Γ be the set of pixels with initially known values, i.e. $\forall x \in \Gamma : U(x) = 0$. In the beginning, the pixels in Γ are entered to a priority queue Q. The key step of the algorithm is a loop: the pixel x_s with the lowest U value is extracted from Q and added to K, the set of pixels for which the U value is decided. All the hitherto undecided pixels adjacent to x_s are updated using (2.63). If they have not been in Q before, they are added. Otherwise their U value and position in Q is revised. Then the loop starts over, unless Q is empty. When Q is implemented by a suitable heap structure, the complexity of this algorithm is $O(N \log N)$ where N is the number of image pixels. This procedure leads to Algorithm 1 below.

Algorithm 1: Fast marching method (FMM).

Input: A set Γ with $\forall x \in \Gamma : U(x) = 0$, cost function P.

Output: Weighted distance function U.

```
// Initialisation
```

initialise priority queue Q

foreach $x \in \Gamma$ **do**
    $U(x) \leftarrow 0$
    label x as *Trial* and add to Q

foreach $x \notin \Gamma$ **do**
    $U(x) \leftarrow \infty$
    label x as *Far*

```
// Main loop
```

while Q *is not empty* **do**
    pop the point x_s with the smallest U-value from Q
    label x_s as *Known*
    foreach *neighbour* x_n *of* x_s **do**
        if x_n *is not Known* **then**
            compute $U(x_n)$ using (2.63)
            if x_n *is Far* **then** label x_n as *Trial*
            update Q

2.4 Discrete Optimisation

2.4.1 Dynamic Programming (DP)

The term *dynamic programming* goes back to Bellman, see e.g. [BD62] for a comprehensive treatise on the topic.

Let $f(x_1,\ldots,x_{N_v})$ be the *objective function* where the variables x_i can only take discrete values $0 \le x_i \le n_i$. The goal is to find the minimum (or maximum) value of f and a configuration of variables $(x_1^-,\ldots,x_{N_v}^-)$ for which this optimum is achieved. Often, the objective function can be written as a sum of functions, each depending only on a few of the variables. The easiest case is

$$f(x_1,\ldots,x_{N_v}) = f_1(x_1,x_2) + f_2(x_2,x_3) + \ldots + f_{N_v-1}(x_{N_v-1},x_{N_v}) \tag{2.64}$$

The number of operations to solve the minimum problem for (2.64) with a brute-force strategy grows exponentially with N_v. The complexity can be greatly reduced, if a *multistage optimisation process* is set up using the following recursive formula [BD62]:

$$h_1(x_1) = 0 \tag{2.65}$$

$$h_{k+1}(x_{k+1}) = \min_{0\le \mathbf{x}_k \le n_k} (f(x_k,x_{k+1}) + h_k(x_k)) \tag{2.66}$$

$$m_{k+1}(x_{k+1}) = \arg\min_{0\le \mathbf{x}_k \le n_k} (f(x_k,x_{k+1}) + h_k(x_k))\,, \tag{2.67}$$

with $k = 1,\ldots,N_v-1$. For every possible value of x_{k+1}, both the value of h_{k+1} and the value of m_{k+1} have to be stored, resulting in two tables with n_{k+1} entries each. Finally, the minimum of h_{N_v} has to be computed, and the minimiser is found by tracing back the m_{k+1}-tables of potentially optimum preceding values:

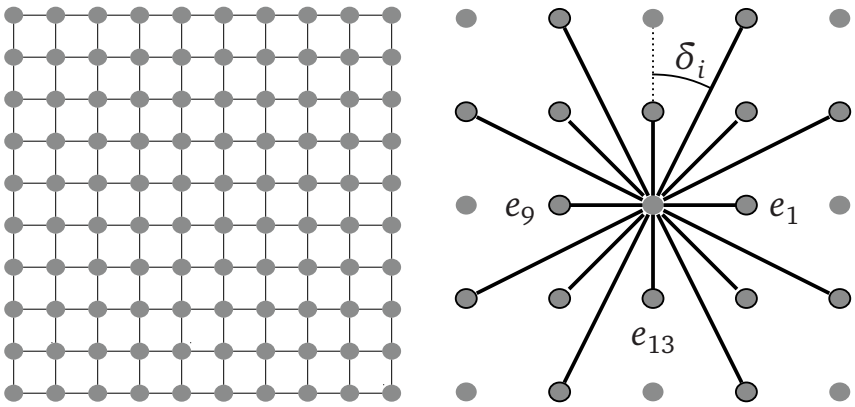
$$x_{N_v}^- = \arg\min_{0\le \mathbf{x}_{N_v} \le n_{N_v}} h_{N_v}(x_{N_v}) \tag{2.68}$$

$$x_k^- = m_{k+1}(x_{k+1}^-) \qquad k = 1,\ldots,N_v-1\,. \tag{2.69}$$

Using this scheme, the number of operations to minimise (2.64) is reduced to $O(n^2 N_v)$, where, for simplicity, $n_i = n$ for all i is assumed. Note however, that the complexity of the method grows with the number of variables interacting with each others. Consider, for instance, the objective function:

$$f(x_1,\ldots,x_{N_v}) = f_1(x_1,x_2,x_3) + f_2(x_2,x_3,x_4) + \ldots + f_{N_v-2}(x_{N_v-2},x_{N_v-1},x_{N_v})\,, \tag{2.70}$$

where three instead of two variables interact, respectively. Then, the complexity of the above multistage process increases to $O(n^3 N_v)$.



(a) (b)

Figure 2.8: An undirected 4-neighbourhood 2D grid graph (a) and a 16-neighbourhood system (b).

2.4.2 Dijkstra's Algorithm (DA)

An *undirected graph* is a pair $G = \langle \mathcal{V}, \mathcal{E} \rangle$ consisting of a set $\mathcal{V}$ of vertices (or nodes) and a set $\mathcal{E}$ of edges. Each edge is an element of $\mathcal{V} \times \mathcal{V}$.[1] If the edges are assumed to be *ordered pairs* of vertices instead, G is called a *directed graph*. A *weighted* graph is a graph with a weight function $\omega : \mathcal{E} \to \mathbb{R}$ that assigns a number (the weight) to each edge. In a *2D grid graph* the vertices are embedded into $\mathbb{R}^2$ in regular grid-like manner (see Fig. 2.8(a)). The set of edges can then be defined by assuming topologically identical neighbourhood systems $\mathcal{D}$ for all vertices with $\mathcal{D}$ given by a set $\mathcal{D} = \{e_i : i = 1, \ldots, |\mathcal{D}|\}$ of vectors (see Fig. 2.8(b)). The angular difference between adjacent directions e_i and e_{i+1} in a neighbourhood system is denoted by δ_i.

Dijkstra's algorithm (DA), proposed by Dijkstra [Dij59] in 1959, computes shortest paths in general graphs with non-negative edge weights. Its applications including route planning in navigation systems and computer networks, DA is a classic algorithm which remains relevant to this day.

DA is the archetype of the fast marching method, so it is not surprising that both algorithms are very similar. In fact, they differ only in the updating step (see Algorithm 2). Consider the problem of finding a minimal path from vertex v to vertex w. DA starts from v with distance value zero and then updates all neighbouring vertices with a tentative distance value. Since

[1] In the case of graphs without reflexive edges (as throughout this thesis), edges can also be characterised as two-element subsets of $\mathcal{V}$.

all edge weights are non-negative the smallest tentative value can be fixed and a new iteration of updating the neighbouring vertices begins. Like in the fast marching method, points with tentative distance values (labeled as *Trial*) are kept in a priority queue Q. Once Q is empty, the algorithm stops. Note that while the classic DA [Dij59] starts from a single vertex, Algorithm 2 displays the straightforward generalisation to a list of starting vertices.

Shortest paths from v to an arbitrary vertex w are computed by tracing back the shortest path with the predecessor function $pred$ defined by DA (see Algorithm 2). This simple routine is presented in Algorithm 3.

Complexity The computational complexity of DA – like that of the FMM – greatly depends on the implementation of the priority queue. Brute force implementations lead to a complexity of $O(|\mathcal{V}|^2)$. The usage of sophisticated data structures can accelerate the algorithm up to $O(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$, achieved by the *Fibonacci heap* [FT87].

Algorithm 2: Dijkstra's algorithm (DA), part one: Computation of the weighted distance function.

Input: Set of starting points S, weight function $\omega : \mathscr{E} \to \mathbb{R}^{>0}$.

Output: Weighted distance function $d : \mathscr{V} \to \mathbb{R}^{\geq 0}$,

predecessor function $pred : \mathscr{V} \setminus S \to \mathscr{V}$.

```
// Initialisation
```

initialise priority queue Q

foreach $x \in S$ **do**
    $d(x) \leftarrow 0$
    label x as *Trial* and add to Q

foreach $x \notin S$ **do**
    $d(x) \leftarrow \infty$
    label x as *Far*

```
// Main loop
```

while Q *is not empty* **do**
    pop the point x_s with the smallest d-value from Q
    label x_s as *Known*
    foreach *edge* (x_s, x_n) *starting from* x_s **do**
        if x_n *is not Known* **then**
            if $d(x_s) + \omega\left((x_s, x_n)\right) < d(x_n)$ **then** `// shorter path to` x_n`?`
                $d(x_n) \leftarrow d(x_s) + \omega\left((x_s, x_n)\right)$ `// update dist.value`
                $pred(x_n) \leftarrow x_s$ `// store predecessor vertex`
                if x_n *is Far* **then** `// update` Q
                    label x_n as *Trial*
                    add x_n to Q
                else update Q with new value $d(x_n)$

Algorithm 3: Dijkstra's algorithm (DA), part two: Tracking back the shortest path.

Input: Set of starting points S, endpoint $w \in \mathcal{V} \setminus S$,

predecessor function $pred : \mathcal{V} \setminus S \to \mathcal{V}$.

Output: List L, comprising the shortest path.

initialise empty list L

push w into L

$p \leftarrow w$

while $p \notin S$ **do** `//Starting point not reached yet`

  $p \leftarrow pred(p)$ `// progress to predecessor vertex`

  push p to the front of L

Chapter 3

A Review of Active Contours

This chapter gives a summary of the works that are broadly considered the cornerstones in the field of ACs, with main focus on edge-, rather than region-based methods. Two distinct approaches to ACs are pursued in Chapters 4 and 6, and the respective state of the art is discussed there in detail.

3.1 Edge based Active Contours

3.1.1 Snakes

Active Contours or *snakes*, first proposed by Kass et al. [KWT88], use explicitly parametrised splines to minimise the energy functional

$$E(\Gamma) = \int_{[0,1]} E_{int}(\Gamma(r)) + E_{ext}(\Gamma(r))\, dr\,, \tag{3.1}$$

and thus localise image features such as edges or lines. Here, $\Gamma : [0,1] \to \mathbb{R}^2$ is a parametrised curve, and E_{int} and E_{ext} denote the internal and the external energy, respectively. While the internal energy controls the regularity of the curve, the external energy attracts the curve to the desired image features. Typically, the latter relates to the image gradient, e.g. $E_{ext} = E_{edge} = -|\nabla I|^2$. Furthermore, user constraints for the curve can be incorporated in E_{ext}, see [KWT88] for details.

Summing up, the following functional was proposed for edge segmentation:

$$E(\Gamma) = \int_{[0,1]} \left(\frac{1}{2}\alpha(r)|\partial_r \Gamma(r)|^2 + \frac{1}{2}\beta(r)|\partial_{rr}\Gamma(r)|^2 + P(\Gamma(r)) \right) dr\,, \tag{3.2}$$

where α and β are the Tikhonov parameter functions and $P = -|\nabla I|^2$. Assuming constant α

and β, the following L^2-gradient flow is derived for (3.2):

$$\partial_t \Gamma = \alpha \partial_{rr} \Gamma - \beta \partial_{rrrr} \Gamma - \nabla P(\Gamma) \,. \tag{3.3}$$

Numerical implementation

The flow equation given by (3.3) is linear in the derivatives of Γ with a low-order nonlinearity given by the ∇P term. Thus, Kass et al. [KWT88] used a semi-implicit discretisation scheme for (3.3), yielding the following discrete equation:

$$\frac{\Gamma^{n+1} - \Gamma^n}{\Delta t} + A \cdot \Gamma^{n+1} = -\nabla P(\Gamma^n) \,, \tag{3.4}$$

where the vector Γ^n approximates Γ at a time t^n, i.e. Γ^n has the components $\Gamma_i^n = \Gamma(i\Delta x, t^n)$. Δt and Δx denote the time step and the spatial resolution, respectively. The stiffness matrix A, obtained by discretisation of the spatial derivatives using finite differences, is pentadiagonal. Consequently, the following linear system of equations has to be solved for Γ^{n+1}:

$$(Id + \Delta t \cdot A)\, \Gamma^{n+1} = \Gamma^n + \Delta t \cdot F(\Gamma^n) \tag{3.5}$$

with Id the unit matrix and $F = -\nabla P$ for convenience. The equation system (3.5) can be solved efficiently by LU decomposition of the matrix $(Id + \Delta t \cdot A)$. If the parameters α, β and Δt are kept constant, this has to be done only once.

Balloon force

Subsequently, Cohen [Coh91] observed that the external energy proposed by Kass et al. [KWT88] yields unsatisfactory results, when the snake is placed too far from the desired image features, or the noise is too large. As a remedy, Cohen proposed to incorporate a *balloon* force into the external force field, constantly pushing the contour in the normal direction:

$$F = k_1 v(r) - k \frac{\nabla P}{|\nabla P|} \,. \tag{3.6}$$

Here, v is a unit normal field, and the coefficients k_1 and k are chosen such that k is slightly bigger than k_1.

3.1.2 Geodesic Active Contours (GACs)

The classic snakes, though successful due to their simplicity and efficiency, have two major drawbacks: they dependend on the chosen parametrisation and are unable to change their topology. The latter limitation for snakes was overcome by McInerney and

Terzopoulos[MT95]. Independently, Caselles et al. [CCCD93] and Malladi et al. [MSV95] proposed the geometric active contours, given by:

$$\partial_t \Gamma = f(\Gamma)(\kappa + c)\nu \,, \tag{3.7}$$

where c is a constant, comparable to the balloon coefficient k_1 in (3.6), f is a positive edge indicator function with values close to 0 near edges and larger values elsewhere, and κ is the curvature of the contour. In [CCCD93, MSV95], f is chosen as

$$f(x) = \frac{1}{1 + |\nabla I|^p} \tag{3.8}$$

with $p = 1$ or $p = 2$. The flow equation (3.7) contains only geometric quantities that do not depend on any particular parametrisation of Γ. Due to the application of Osher and Sethian's level set framework [OS88], a parametrisation-free representation of the contour could be achieved, and changes in the topology of the contour were no longer a problem (see a review of the level set method in Section 2.2).

However, one major problem remained for the geometric ACs given by (3.7): the evolving curve can only stop if $f(\Gamma) = 0$. This is an ideal case that can never occur, since f is strictly positive by definition. Therefore, the curve might continue its propagation, even if the desirable object boundary is reached.

The geodesic active contours (GACs) proposed in [CKS97] and [KKO$^+$96] overcome this problem. They are based on the generalised arc length functional

$$E(\Gamma) = \int_\Gamma f(s)\, ds \,. \tag{3.9}$$

The L^2-gradient flow of (3.9) is

$$\partial_t \Gamma = \big(f(\Gamma)\kappa - \langle \nabla f, \nu \rangle\big)\, \nu \,, \tag{3.10}$$

see Chapter 2, Corollary 6. In both works, [CKS97] and [KKO$^+$96], an inflation term, similar to the one introduced in (3.7), was added, yielding the flow equation for the GAC:

$$\partial_t \Gamma = \big(f(\Gamma) \cdot (\kappa + c) - \langle \nabla f, \nu \rangle\big)\, \nu \,. \tag{3.11}$$

Siddiqi et al. [SLTZ98] refined the GAC equation (3.11) by replacing the constant inflation term c with a term originating from a simplified weighted area term. They started from the modified weighted area functional (cf. Chapter 2, Proposition 7)

$$A_f = -\frac{1}{2} \int_\Gamma f(s) \cdot \langle \mathbf{x}, \nu(s) \rangle \, ds \,, \tag{3.12}$$

where $\mathbf{x}$ is the identity vector field in $\mathbb{R}^2$. The gradient flow for A_f is derived as:

$$\partial_t \Gamma = \left(f + \frac{1}{2} \langle \Gamma, \nabla f \rangle \right) \nu \ . \tag{3.13}$$

The refined inflation term (3.13) in conjunction with (3.10) provides an additional attraction force in the vicinity of edges. It can prevent the contour from *overshooting* as a result of a balloon force that is weighted too high.

All methods reviewed in this subsection can be transferred to volumetric 3D images in a straightforward manner. Also it should be pointed out that this is one of the major achievements of the level set method.

3.1.3 ACs using Dynamic Programming

This section reviews active contour approaches using *dynamic programming* (DP) rather than variational techniques. For an explanation of the basic idea of DP see Section 2.4.1.

Shortly after ACs were introduced by Kass et al. [KWT88], Amini et al. [AWJ90] proposed the use of DP for the optimisation of ACs, in order to avoid numerical stability problems and incorporate hard constraints. Similar to the variational approach, an initial polygon close to the object of interest has to be provided by the user. Then in each iteration, the optimal movement of the nodes with respect to the AC energy (3.2) is computed with DP. Given n nodes of the contour and m admissible directions for each node to move, the computational complexity sums up to $O(nm^3)$ for one iteration. Amini et al.'s approach is clearly slower than the original approach in [KWT88], see [WS92] for an evaluation.

To speed up the DP-based active contours, Williams and Shah [WS92] suggested a greedy algorithm. Unlike in Amini et. al's approach [AWJ90], Williams and Shah proposed to optimise the movement of each node separately. As a result, the optimality of each iteration is sacrificed, but the complexity is significantly decreased to $O(nm)$, and a speed comparable to the variational approach is reached.

Chandran et al. [CMM91, CP98] proposed a DP method for the computation of globally optimal active contours with given endpoints. The authors split the line between the endpoints in equal size intervals. Assuming perpendicular lines passing through each of the nodes, the intersection points between the curve and these lines are then the stages of a multistage optimisation process, which is very similar to the procedure in e.g. [Mon71, Mar76]. Yet this technique functions only for curves passing through the stages monotonically. It cannot handle the case where a curve intersects one of the lines more than once, and therefore it is

quite restricted in its range of applications.

3.1.4 Globally Optimal GACs

All the AC models reviewed in Sections 3.1.1 and 3.1.2 are based on gradient flow techniques: the contours evolve and converge towards a local minimum of the respective energy. This implicates some serious drawbacks:

- The evolving contour might get trapped in *false* local minima caused by spurious image features;
- The result might depend strongly on the initial placement of the curve.

In the following, some techniques for computing globally optimal ACs are discussed.

Open contours

Cohen and Kimmel [CK97] proposed a method to compute globally optimal ACs, when their endpoints are given. They observed that such *open* ACs can be considered as weighted shortest paths. The fast marching method [Set96] – reviewed in Section 2.3 – provides the algorithmical framework for the very accurate and efficient computation of such paths.

Cohen and Kimmel [CK97] considered the energy

$$E(\Gamma) = \int_{\Gamma} \big(w + f(s)\big)\, ds \tag{3.14}$$

for open curves Γ. Here, f again notes a positive edge detector function, and w is a positive regularity parameter. Defining the potential $P := w + f$, Cohen and Kimmel then solved the Eikonal equation

$$|\nabla U| = P\,, \tag{3.15}$$

starting from one of the endpoints. Backtracking the shortest path delivers the global minimum of (3.14), see Fig. 2.7 for an example (p. 26).

This approach works very well for open contours, i.e. if the endpoints are given. Cohen and Kimmel [CK97] also proposed a method to compute a closed contour in the case where just one point of it is given. Yet this method includes the search for certain saddle points of the distance map which involves heuristics. User interaction is necessary to pick the correct saddle among a number of candidates defined by the algorithm. Apparently, closed contours found in this way are no longer global minimisers of the energy (3.14), but only piecewise globally optimal. In addition, closed contours comprising several partial contours, each of which is globally optimal, might not be smooth in the respective nodes.

Closed contours

Boykov and Kolmogorov [BK03] were the first to propose an approach to compute closed, globally optimal GACs. In fact, their method, called *Geocuts*, works for minimal surfaces just as well. In contrast to the methods reviewed so far, Boykov and Kolmogorov's method is not based on a variational or PDE approach, but on combinatorial optimisation using graph cuts [BVZ01]. They studied the Riemannian arc length functional

$$E(\Gamma) = \int_{\Gamma} \sqrt{\mathbf{t}(s)^{\top} \cdot D(s) \cdot \mathbf{t}(s)}\, ds \,, \tag{3.16}$$

where $\mathbf{t}$ is the unit tangent vector field along the curve and $D(.)$ is the Riemannian tensor field specifying the metric. As Boykov and Kolmogorov pointed out in [BK03], the classical GAC energy functional (3.9) is a special case of (3.16) obtained by setting

$$D = f(|\nabla I|) \cdot Id \,, \tag{3.17}$$

with Id the identity matrix. Further, in contrast to (3.9), Eq. (3.16) allows for the implementation of anisotropic Riemannian metrics. The key contribution of [BK03] is a formula to compute the edge weights in a graph in such a manner, that the cost of a cut approximates the Riemannian length (3.16) of the corresponding curve or surface. This result is obtained using the Cauchy-Crofton formula from integral geometry. Boykov and Kolmogorov [BK03] proposed the following Riemannian tensor for 2D/3D image segmentation:

$$D = f(|\nabla I|) \cdot Id + (1 - f(|\nabla I|)) \cdot \mathbf{u} \cdot \mathbf{u}^{\top} \,, \tag{3.18}$$

with the edge detector function

$$f(x) = \exp(-\frac{x^2}{2\sigma^2}) \tag{3.19}$$

and the normalised image gradient $\mathbf{u} = \frac{\nabla I}{|\nabla I|}$. Note, that the first summand in (3.18) attracts the cut/contour to areas with large image gradients and the second – anisotropic – summand aligns it with the edges. Minimisation of (3.16) is achieved by *min-cut/max-flow* optimisation techniques (see [BVZ01]).

The *Geocuts* approach [BK03] is powerful and efficient for segmenting 2D and 3D volumetric images. Like in the general graph cut segmentation framework [BFL06], user input for interactive segmentation is readily incorporated, and due to the efficient optimisation the method is very fast. These properties have made graph cut based methods extremely popular in recent years. Main drawbacks, compared to the level set and PDE based approaches,

remain the metrication error (see [BK03], Section 5.2) and the lack of smoothness in the contour.

Subsequently, an alternative approach to globally optimal GACs was suggested by Appleton and Talbot [AT05]. The authors succeeded in applying Cohen and Kimmel's herein before mentioned method [CK97] to form a novel technique for computing closed, globally optimal ACs. The key idea is the following: initially, the user has to provide a point within the object of interest. Then, an infinitesimal cut is made, connecting the internal point with the image boundary. Figuratively, a closed geodesic through one point on the cut can then be constructed by computing open geodesics between two points on either side of the cut, which are infinitesimally close but still separated by the cut. Thereby, the geodesic must not intersect the cut. In order to avoid a bias towards short curves Appleton and Talbot [AT05] used the edge detector function

$$\tilde{f} = \frac{1}{r} f \ , \tag{3.20}$$

where r is the distance to the inner point, and f is the standard GAC stopping function as in (3.8).

The above method works fine, if the contour passes through the cut only once. For the general case, Appleton and Talbot [AT05] proposed a more sophisticated refinement of the idea. They constructed a product space $S = S' \times \mathbb{Z}$, where S' is the image plane including a cut. $\mathbb{Z}$ codes the number of times the path has passed around the internal point in a certain direction. Different levels of this space are connected by a helical surface. Thus, admissible closed curves can be described as open curves with identical endpoints in S', yet exactly one layer apart in $\mathbb{Z}$. This procedure allows for computing an optimal closed geodesic through one point using Cohen and Kimmel's method [CK97] for open geodesics. The search for the optimal cut point is sped up by a branch-and-bound scheme, such that the computational complexity of the whole algorithm is similar to that of the fast marching method: about $O(N \log N)$ for an image with N pixels.

The method developed by Appleton and Talbot [AT05] provides an efficient way to segment salient contours from images. Only little user interaction is required, in the form of the position of the inner point – the seed. Yet this fact can also be interpreted as a major weakness of the method. Applied to complex or very noisy images the algorithm might not deliver the desired result. Unlike the Geocuts method [BK03], there is no means to specifically guide the

algorithm.

3.2 Region-based Active Contours

Since the region-based approach to image segmentation is not pursued in this thesis, this field will not be reviewed in detail. For the sake of completeness, Chan and Vese's ACs without edges [CV01] are covered here. For further approaches to statistical AC models see e.g. [Ron94, YTW99].

The central idea of region-based segmentation is to find statistically significant partitions of an image rather than edge-based contours. In applications where such results are desirable, region-based methods can overcome some of the problems intrinsic to the edge-based techniques. For instance, they exhibit less sensitivity to noise and the initial placement of the contour, as well as a better handling of object concavities.

Due to its simplicity and elegance, Chan and Vese's approach [CV01] is the most prominent in its field. They considered the functional

$$\begin{aligned} E(\Gamma, c_1, c_2) &= p_1\|\Gamma\| + p_2 A(\Gamma) \\ &\quad + \lambda_1 \int_{in(\Gamma)} |I(x) - c_1|^2 \, dx + \lambda_2 \int_{out(\Gamma)} |I(x) - c_2|^2 \, dx \,, \end{aligned} \tag{3.21}$$

where $\|.\|$ and $A(.)$ denote the arc length and the enclosed area, respectively. The area inside and outside the curve is denoted by $in(\Gamma)$ and $out(\Gamma)$, and $p_1, p_2, \lambda_1, \lambda_2 \geq 0$ are fixed parameters. While the terms in the first row of (3.21) are regularising and inflating terms, respectively, which are well-known from edge-based models, the area integrals in the second row of (3.21) are statistical terms penalising variance in the image intensity. Consequently, a minimising curve Γ partitions the image into two, not necessarily connected, regions with the smallest possible intensity variation in each of them. Chan and Vese [CV01] then derived the gradient flow of (3.21) and numerically implemented it with a level set function.

It is worth pointing out that no image gradient has to be computed in order to minimise (3.21), what makes the method suitable for very noisy data. Yet in many applications the objects of interest cannot be separated from the background solely by their intensity distribution. In such cases, edge characteristics remain an essential feature for the segmentation process.

Chapter 4

Geodesic Active Contours on Implicit Surfaces

4.1 Introduction

The generalisation of well-known methods from 2D image processing to images on curved surfaces has attracted considerable interest recently. Here, typical examples are isotropic and anisotropic diffusion [BMC+03, BBTV07], scale space analysis [BBTV07, Kim97], and image inpainting [WDZC05]. The objective is to adequately process data obtained from 3D scanners, or surface reconstruction algorithms.

As a consequence of Gauss' famous *theorema egregium* (see e.g. [dC76]), surfaces with a non-zero Gaussian curvature cannot be mapped onto the plane without metric distortions. Moreover, looking at particular applications, current research [CBF05, TMS04, YB07] shows that multi-modal approaches combining texture and shape information perform better in face biometrics than either 2D- or 3D techniques alone. Hence, there is a strong demand for object segmentation techniques, and in particular, active contour models that incorporate both 2D- and 3D information.

Object segmentation is one of the central problems in Computer Vision. Spira and Kimmel [SK07] first attempted to apply PDE based active contour models, namely the geodesic active contour (GAC), to images on parametric surfaces. Further, they conjectured that incorporating the surface geometry into the curve evolution process may improve face feature segmentation results. However, no evidence was given for this claim. Bogdanova et al. [BBTV07] provided an example where a surface GAC succeeds in segmenting the object of interest, whereas its

standard Euclidean counterpart fails. Yet this result was achieved on a synthetic surface and does not allow for conclusions concerning face data.

In this chapter, GAC segmentation on curved surfaces is studied in-depth. A model that avoids the intrinsic drawbacks of existing approaches using parametric surfaces is proposed. The chapter is organised as follows: the state of the art in the field is reviewed in Section 4.2. Section 4.3 gives a thorough analysis of the existing works employing parametric surfaces and reveals important limitations of such approaches. Section 4.4 suggests the application of the implicit-, instead of the parametric surface representation in order to avoid these issues. The numerical implementation of the corresponding PDE is explained in Section 4.5. Experimental results on several synthetic and real surfaces are presented in Section 4.6. Finally, Section 4.7 discusses the outcomes of the chapter and draws conclusions.

4.2 State of the Art

4.2.1 Approaches using Snakes

First attempts to generalise the classic active contour model to surfaces were undertaken in the Computer Graphics community on triangular meshes. Milroy et al. [MBV97] applied a greedy DP algorithm similar to that proposed for images by Williams and Shah [WS92].

Jung and Kim [JK04] chose a similar approach. However, they added some routines that check the snake evolution and handle collisions if necessary. Both, Milroy et al.'s and Jung and Kim's algorithms, deliver solutions that are, in general, not even local minima of the respective energy functional. Furthermore, the accuracy of these methods is limited, since the snake nodes have to coincide with surface vertices.

The latter restriction was tackled in [LL02, LLS+05] by evolving the snakes in local parametrisation areas. Hence, Lee and Lee's work can be considered as a 2.5D- rather than a fully 3D approach. It requires a covering of the surface mesh by several patches, which might be difficult to generate for intricate surfaces. In addition, it does not offer topological flexibility for the evolving contour.

Bischoff et al. [BWK05] proposed parametrisation free snakes evolving on meshes, thereby generalising their work for 2D images [BK04]. In this approach the movement of the snake nodes – called *snaxels* – is not restricted to mesh vertices, instead the snaxels are allowed to move on the triangle edges. Bischoff et al.'s method is parametrisation free, yet in respect to ACs it is an explicit- rather than an implicit approach. The movement of each snaxel has

to be traced 'manually'. Likewise, routines have to be provided to handle snaxel collisions and topology changes. While this algorithm features methodological improvements, such as topology control over the evolving contour, it is not as intuitive as implicit curve evolution. The authors point out the possible formation of certain artefacts due to restriction of the snaxel position to edges. Further, the method cannot process evolution speeds of mixed sign (i.e. positive and negative) within one time step.

4.2.2 Geometric (PDE) approaches

Geodesic active contours

Kimmel [Kim97] was the first to study geometric curve evolution on surfaces in the context of Computer Vision. He considered the geodesic curvature flow – the generalisation of the curvature flow to surfaces – on surfaces M that can be represented as graphs, i.e. $M = \{(x, y, z \in \mathbb{R}^3 : z = F(x, y)\}$. Subsequently, he applied this flow to images painted on surfaces (i.e. their intensity level lines) and showed that the evolving image creates a *scale space*.

Cheng et al. [CBMO02] analysed basic geometric curve evolution models on implicit surfaces, including the geodesic curvature-, advection-, and normal flows. They did not apply their framework to Computer Vision, and the suggested implementation was inefficient. Their approach was resumed and improved in terms of computational complexity in [KDG08a]. A detailed review of Cheng et al.'s technique is given in Section 4.4.2.

More recently, Spira and Kimmel [SK07] generalised GACs to segment images painted on parametric surfaces, thus extending Kimmel's work [Kim97] on the geodesic curvature flow. Spira and Kimmel projected the flow equations onto the parameterisation plane of the surface and solved the resulting 2D PDE by means of the level sets framework [OS88]. Yet they did not incorporate a balloon term in their model, therefore the method can only be applied to data involving no or very little noise. In fact, it was shown in [KDG08a] that balloon forces cannot be incorporated properly in GAC frameworks on parametric surfaces: the different scaling behaviour of the balloon term leads to undesirable results when large variations occur in the surface metric. A thorough analysis of Spira and Kimmel's algorithm is given in Section 4.3.

Bogdanova et al. [BBTV07] pursued a similar approach as Spira and Kimmel. They analysed geodesic active contours on omnidirectional images obtained from a catadioptric sensor, which is a combination of a curved mirror and a lens to form a projection onto the image

plane of a camera. Consequently, such images contain information of a scene in all possible directions. Since spherical-, hyperbolic- and parabolic mirrors are most common in practice, Bogdanova et al. [BBTV07] studied GAC on the respective surfaces: the sphere, the hyperboloid, and the paraboloid. The geometry of the surface is projected on the plane and incorporated into the image segmentation functional. Metric distortions stemming from the projection are equalised. In contrast to Spira and Kimmel's work [SK07], Bogdanova et al. [BBTV07] derived the flow equations directly from an adapted version of the GAC energy functional. Subsequently, they calculated the flow equations for the three studied surfaces. Their work is restricted to a few particular surfaces and, in principle, the intrinsic limitations of GAC models on parametric surfaces – as pointed out in [KDG08a] – remain valid for their approach.

Region-based approaches

The Chan Vese (CV) active contours without edges [CV01] were generalised to conformal surfaces in [LWC05]. A global conformal parametrisation is required to apply their method. In this case, the solution of the surface PDE is comparably simple, since conformal metrics differ only by a scalar factor from the Euclidean metric. Yet for complex surfaces appearing in practice such parametrisations cannot be obtained, unless the surface is cut into patches.

Subsequently, Hara et al. [HKIU07] studied the CV algorithm on polar meshes. The authors transformed the CV level set PDE to polar coordinates and used a finite difference scheme for the numerical implementation. In order to avoid numerical problems close to the poles, a special parametrisation of the sphere was employed.

Jin et al. [JYS07] studied segmentation on surfaces in a slightly different context. To reconstruct a 3D scene from multiple 2D views in a variational framework, they simultaneously evolved both a level set surface and a radiance pattern of the surface. Discontinuities of the radiance are modeled by curves on the surface. Jin et al. [JYS07] applied Cheng et al.'s [CBMO02] framework to evolve curves on the surface. Yet their model is only 2.5D, since the actual image segmentation according to a Mumford-Shah model takes place after projecting the surface to the original 2D views. Their approach is not applicable to images on implicit surfaces, as generally, 2D projections of the surface are not available.

Tian et al. [TMR09] employed the Closest Point Method (CPM) [RM08] in order to obtain a Chan Vese segmentation algorithm on arbitrary surfaces. The CPM is similar to Cheng's method [CBMO02], as surfaces have to be embedded in $\mathbb{R}^3$. Instead of the signed distance

transform of a given surface, the CPM requires the closest point transform. As opposed to Cheng's method [CBMO02], it allows to process open surfaces with boundary. The computational complexity is high since the PDE has to be solved in a neighbourhood of the surface.

Recently, Delaunoy et al. [DFPH09] introduced a multi-region segmentation method for triangulated meshes. They considered a convex functional, and their technique finds global, rather than local minimisers. Problems due to poor contour initialisation are avoided. However, common features of meshes, such as edges or ridges, cannot be segmented by this region-based method and therefore, the field of potential applications is limited.

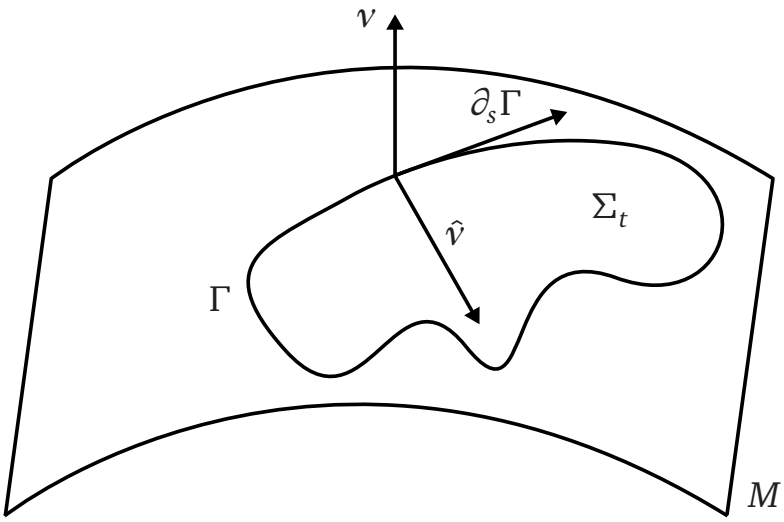


Figure 4.1: Geometry of a curve on a surface.

4.3 Analysis of GAC on parametric surfaces

The objective of this chapter is the generalisation of the 2D GAC scheme [CKS97, KKO+96] to curved surfaces. Including a balloon term (see Sections 3.1.1 and 3.1.2), the model is given by the evolution equation:

$$\partial_t \Gamma = \big(f \cdot (\kappa + c) - \langle \nabla f, \nu \rangle\big)\, \nu \,, \tag{4.1}$$

where Γ is a curve, and $\partial_t \Gamma := \frac{\partial \Gamma}{\partial t}$, κ, and ν are the time derivative, the curvature, and the unit normal field of the curve, respectively. The balloon term constant and the edge indicator function are denoted by c and f.

As stated above, Spira and Kimmel [SK07] were the first to introduce a GAC scheme for image data on surfaces. They considered the following, generalised GAC flow equation:

$$\partial_t \Gamma = \Big(\kappa_g f - \langle \nabla f, \hat{\nu} \rangle\Big)\, \hat{\nu} \,. \tag{4.2}$$

Here, κ_g denotes the geodesic curvature and $\hat{\nu}$ is the unit co-normal vector of the curve Γ on the surface M. The co-normal $\hat{\nu}$ is defined as the normalised cross product of the normal ν and the tangent vector field Γ_s (see Fig. 4.1). The geodesic curvature is the tangential component of the curvature, see e.g. [dC76] for an introduction to the differential geometry of curves (cf. also Section 2.1.1). On the example of Spira and Kimmel's approach [SK07], this section elaborates on the limitations of GAC models on parametric surfaces. Three major drawbacks are pointed out:

1. GAC models on parametric surfaces cannot properly incorporate a balloon term due to its differing scale;

2. Numerical problems arise due to parametrisation singularities;

3. There are topological restrictions for the evolving curve.

Point 1. was first addressed in [KDG08a], yet the stopping term was not included in the analysis. Section 4.3.3 performs a detailed investigation of the scaling behaviour of GACs on parametric surfaces. The further above stated issues are discussed in Section 4.3.4.

4.3.1 The Framework

Spira and Kimmel [SK07] studied geometric flows in the general case of parametric surfaces embedded in the Euclidean space $\mathbb{R}^n$. They considered surfaces M represented by a parametrisation $X : U \to \mathbb{R}^n$, where $U \subset \mathbb{R}^2$ is the parametrisation plane, i.e.

$$M = X(U) = \left\{ \left(x^1(u_1, u_2), x^2(u_1, u_2), \ldots, x^n(u_1, u_2) \right)^\top \right\} \subset \mathbb{R}^n. \tag{4.3}$$

If Γ is a curve on M, let $\tilde{\Gamma}$ be the preimage of Γ under X, i.e $\Gamma = X \circ \tilde{\Gamma}$. The respective arc length parameters are denoted by s and $\tilde{s}$. The local basis of the tangent spaces on M is given by the derivatives of X with respect to u_i referred to as $X_i = \partial_i := \partial_{u_i} X$. The elements of the metric tensor $[g_{ij}]$ on M are defined as

$$g_{ij} := \langle X_i, X_j \rangle \qquad i, j \in \{1, 2\}.$$

Its determinant $g := \det([g_{ij}]) = g_{11} g_{22} - g_{12}^2$ is called Gramian, its inverse is denoted by $[g^{ij}]$. The matrix of the metric tensor is denoted by $G = [g_{ij}]$.

4.3.2 Geometric Curve Evolution

Spira and Kimmel [SK07] employed the framework stated in Section 4.3.1 to transform geometric curve flows for Γ on M into flows for $\tilde{\Gamma}$ in U. This has the advantage of reducing a 3D- to a 2D problem. It will be highlighted in this section, however, that the approach has some intrinsic drawbacks.

In the following, the essential evolution equations from their framework are outlined.

Geodesic Active Contours Spira and Kimmel [SK07] transformed the flow equation (4.2) into the following level set PDE in the parametrisation plane U:

$$\partial_t \phi = \underbrace{f \left[\frac{(-1)^{i-j} \phi_i \phi_j \phi_{3-i,3-j}}{g|\nabla_M \phi|^2} + \frac{(-1)^{i-j} \Gamma_{ij}^k \phi_{3-i} \phi_{3-j} \phi_k}{g|\nabla_M \phi|^2} \right]}_{I} + \underbrace{(f_m g^{mm} + f_{3-m} g^{12}) \phi_m}_{II} . \tag{4.4}$$

For the sake of simplicity, the notation from [SK07] is adopted here: it is $\phi_i := \partial_i \phi$, $\phi_{i,j} := \partial_{ij}^2 \phi$ and equally $f_i := \partial_i f$. The Christoffel symbols of the metric are denoted by Γ_{ij}^k. In (4.4) and the following equation, Einstein's convention for summation over repeated indices is used. The squared norm of the surface gradient $\nabla_M \phi$ in local coordinates is given by

$$|\nabla_M \phi|^2 = g^{ij} \phi_i \phi_j . \tag{4.5}$$

Note, that term I in (4.4) is the weighted geodesic curvature flow, whereas term II is the geodesic advection (cf. (4.2)). The generalised stopping term is defined by

$$f = \frac{1}{1 + |\nabla_M I|^p} . \tag{4.6}$$

While in [SK07] shorthand notation is used, it must be emphasised that according to [Kim08] the surface gradient of I has to be taken. Moreover, a general exponent $p \in \{1, 2\}$ is included here, while Spira and Kimmel fix $p = 2$.

Geodesic Normal Flow The evolution equation for the geodesic normal flow in Spira and Kimmel's framework is given by

$$\partial_t \phi = -F |\nabla_M \phi| . \tag{4.7}$$

Note, that according to [Kim08], the formula given in [SK07] is erroneous.

4.3.3 Analysis of the Scaling Behaviour

The GAC model (see ((4.42) and (4.4)) proposed by Spira and Kimmel [SK07] does not include a balloon term, which pushes the contour over undesired local minima. Therefore, practical applications of the method are very limited.

In fact, it is shown in the following that using Spira and Kimmel's framework on parametric surfaces a balloon force cannot be properly incorporated into the GAC model, because it scales differently and thus leads to undesirable results.

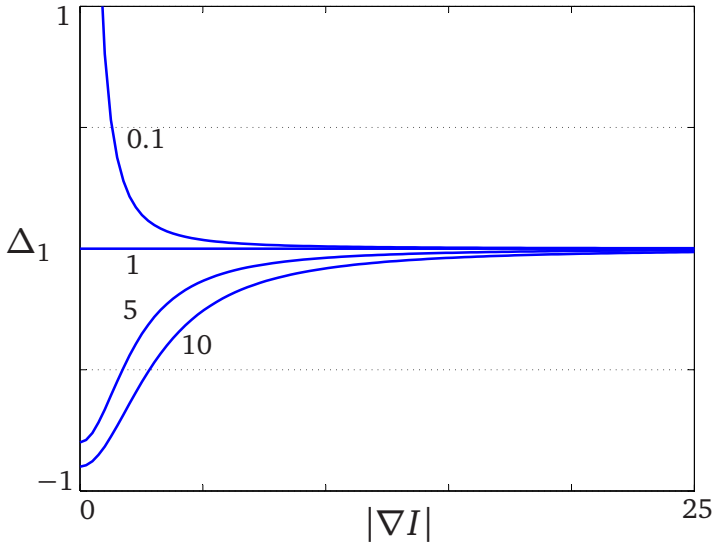


Figure 4.2: Error function Δ_1 for $p = 2$ and various values of k.

To study the scales of the different evolution terms, consider a simple model geometry: the scaling of a plane image by a constant factor $\sqrt{k}$. This corresponds to a parametrisation $X(u_1, u_2) = (\sqrt{k}u_1, \sqrt{k}u_2, 0)^\top$ and the metric tensor $[g_{ij}] = k \cdot [\delta_{ij}]$ with δ_{ij} the Kronecker delta. For convenience, this metric is named the *k-metric* here. The metric is conformally equivalent to the Euclidean metric (1-metric).

Note, that according to (4.6) the stopping term f depends on the metric as well. Let f^k denote the stopping term in the k-metric. The following Propositions on the scaling behaviour of the stopping term in the k-metric are proven in Appendix 4.B.1:

Proposition 8. *Let the function f^k be defined by (4.6) in the respective k-metric. Then*

$$f^k = (1 + \Delta_1) k^{\frac{p}{2}} f^1 , \tag{4.8}$$

where Δ_1 is a relative error which converges to 0 when $|\nabla_M I|$ becomes large (see Fig. 4.2). The gradients ∇f^k and ∇f^1 are collinear, and it holds:

$$\nabla f^k = (1 + \Delta_2) k^{\frac{p}{2}} \nabla f^1 . \tag{4.9}$$

The error terms are given by

$$\Delta_1 = \frac{k^{-\frac{p}{2}} - 1}{1 + k^{-\frac{p}{2}} |\nabla I|^p} \quad \text{and} \tag{4.10}$$

$$\Delta_2 = 2\Delta_1 + \Delta_1^2 . \tag{4.11}$$

Proposition 9. *The equation for GAC on parametric surfaces (4.4) with f defined by (4.6) scales with the factor $k^{\frac{p-2}{2}}$, while the weighted geodesic normal flow ((4.7) with $F = f$) scales with the factor $k^{\frac{p-1}{2}}$.*

The following Corollary presents the main result of this section.

Corollary 10. *Transferring the GAC equation (4.1), including a balloon force, to the framework on parametric surfaces results in an evolution equation that does not scale homogeneously.*

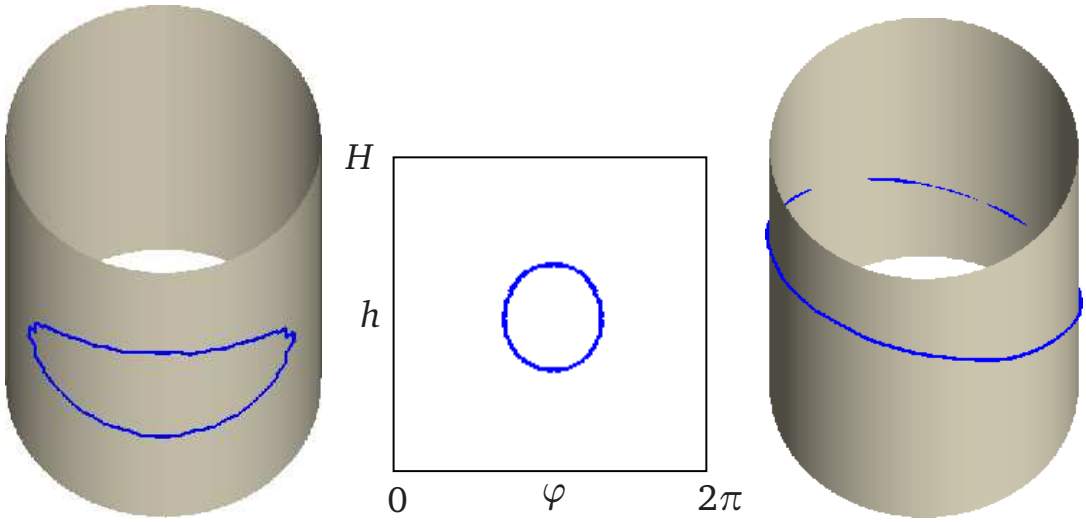
Consider GAC equation (4.4) with an additional weighted balloon term and let $p = 2$. This yields the following equation in the k-metric:

$$\begin{aligned} \partial_t \phi &= f^k \kappa_g^k + \nabla f^k \cdot (G^k)^{-1} \nabla \phi - c f^k |\nabla_M^k \phi| && (4.12) \\ &= f\kappa + (1 + \Delta_1)\langle \nabla f, \nabla \phi \rangle - c f k^{\frac{1}{2}} |\nabla \phi| \,, && (4.13) \end{aligned}$$

where a superscript k denotes terms with respect to the k-metric.

The essence of (4.13) is the following: consider a noisy image with the standard metric (1-metric). With a suitable balloon coefficient c, the GAC converges towards the desired object boundary. Yet, after rescaling the image with k larger than 1, the balloon term is weighted higher than the other terms in (4.13). Note that the error term $(1 + \Delta_1)$ is then close to 1, in particular near edges where $|\nabla_M I|$ takes large values. Consequently, the contour will miss the object edges, if k is chosen sufficiently large. If the metric is constant, this is not a problem, since the coefficients can be adapted to equalise the different scale factors. However, if the image is projected on a curved surface, the metric varies, and it is not clear how the weight of the balloon term can be controlled. The experiment shown in Fig. 4.8(g), Section 4.6.1, demonstrates that the effect caused by the balloon term depends on the *direction* of the contour evolution. Therefore, a scalar weighting function, varying inversely with the metric, cannot remedy the problem. At the very least, the surface gradient of the level set function, $\nabla_M \phi$, would have to be incorporated into a potential rectification term. In this case, however, the resulting term would no longer represent a standard geodesic normal flow term, numerics may become more difficult, and other issues may arise. Thus, it is far from obvious that a suitable rectification term exists at all.

Summing up these considerations, it is not possible to incorporate a balloon term properly into the known approaches [SK07, BBTV07], and moreover it is questionable if it is possible at all. In Section 4.6 the theoretical considerations of this section are supported with experimental evidence.



(a) (b) (c)

Figure 4.3: 0-homotopic curve on a parametric cylinder surface (a) and respective curve in the parametrisation plane (b). Non 0-homotopic curve on an implicit cylinder surface (c).

4.3.4 Further Issues

Topological Restrictions for the Curve

Implementing the evolution equation in the parametrisation domain involves topological limitations for the curve: it must be homotopic to zero, i.e. contractible to a point. As a simple example for evolution of a curve not homotopic to zero, consider the problem of computing nontrivial closed geodesics on a cylinder. Note, that this is essentially the same situation as in the computation of constrictions of vascular surfaces (see Section 4.6.3). A parametrisation of a cylinder of height H and radius r is given by

$$X_r(h,\varphi) = (r\cos\varphi, r\sin\varphi, h)^\top \tag{4.14}$$

with $h \in [0,H], \varphi \in [0,2\pi]$. On a parametric surface, curves must be representable as closed curves within the parametrisation domain (Fig. 4.3(a),(b)). Such limitations do not exist when evolving curves on implicit surfaces (Fig. 4.3(c)) using the framework discussed in Sect. 4.4.2.

Problems with Surface Parametrisations

A further limitation of Spira and Kimmel's [SK07] method is that it requires the surface to have a global, smooth parametrisation. However, such parametrisations in general do not exist for complex surfaces, see Appendix 4.B.2 for details.

4.4 Geodesic Active Contours on Implicit Surfaces

4.4.1 Motivation

As proven by Theorem 11 in Appendix 4.A, Spira and Kimmel's [SK07] GAC model (4.2) can be derived by variational principles. Like its 2D archetype (3.10), it is the L^2 gradient flow of a weighted arc length functional. In addition, Proposition 12 indicates that the successful balloon term concept similarly translates into the surface scenario. Hence, the full geodesic active contour model (4.1) can be generalised to surfaces. Yet the analysis in Section 4.3 has shown that the framework for GAC evolution, as proposed in [SK07], has considerable drawbacks. Undesirable results can occur, when a balloon term is incorporated into Spira and Kimmel's [SK07] GAC model (4.2).

There is an alternative way for representing curved surfaces where the above limitations do not arise. Due to their advantageous features, implicit surfaces have gained an increasing interest in the Computer Vision and Graphics communities in recent years. Radial basis functions (RBF) exhibit an elegant and efficient way to reconstruct surfaces from point clouds while filling holes [CBC+01]. Curless and Levoy [CL96] showed that the implicit surface representation can ease the merge of multiple range images stemming from a 3D scanner. Mauch [Mau00] suggested a method to efficiently compute the distance transform for triangulated surfaces.

Further, representing complex surfaces implicitly is a much easier and more intuitive process than parametrising them. The latter usually entails issues, such as discontinuities, singularities, and several parametrisation patches (see Appendix 4.B.2). In contrast to triangulated surfaces, geometric quantities – e.g. the normals and the curvature – can be computed very easily for implicit surfaces (cf. [BMC+03]). Last but not least, the level set method [OS88] continues to be very popular for surface reconstruction and segmentation. It automatically delivers implicit surfaces, particularly in applications related to medical imaging. Therefore, it is important to provide suitable tools for the processing of implicitly represented surfaces, and in particular segmentation algorithms.

Cheng et al. [CBMO02] introduced a framework for curve evolution on implicit surfaces. They tracked the evolving curve on a surface as the intersection with another evolving surface. All calculations are completed in the Euclidean embedding space $\mathbb{R}^3$. While this procedure avoids the above mentioned issues of the parametric approach, it results in an increased

computational workload. It will be shown in Section 4.5.2, however, that the complexity can be significantly reduced by a suitable localisation of the PDE.

Cheng et al.'s framework [CBMO02] is reviewed next. Afterwards, the PDE for GAC on implicit surfaces is presented in Section 4.4.3.

4.4.2 Geometric Curve Evolution on Implicit Surfaces

When a curve Γ is embedded as an isocontour into a 2D scalar function ϕ, the evolving curve can be tracked by solving a suitable PDE for ϕ (see Section 2.2.1 or [OS88, OF03]). The GAC in (4.1) corresponds to the following PDE in the level set formulation:

$$\partial_t \phi = \underbrace{cf|\nabla\phi|}_{I} + \underbrace{\langle \nabla f, \nabla\phi \rangle}_{II} + \underbrace{f\kappa|\nabla\phi|}_{III}. \tag{4.15}$$

Here, the term I is a motion in normal direction, II is an advection flow, and III is a weighted mean curvature flow. The generalised GAC model proposed in this section requires a framework that provides algorithms for the above three basic geometric evolution terms on implicit surfaces. It is assumed in the following that the implicit surface $M \subset \mathbb{R}^3$ is embedded into a cuboid $\Theta \subset \mathbb{R}^3$ and that M is given by the scalar function ψ on Θ, i.e. $M := \{x \in \Theta : \psi(x) = 0\}$.

Cheng et al. [CBMO02] introduced a method to evolve curves on such surfaces. A curve on a surface M is defined as the intersection of M with the zero level set S of a function ϕ on Θ (see Figs. 4.4 and 4.5). Essential for their method are orthogonal projection mappings onto a plane along a given vector $w \in \mathbb{R}^3$:

$$P_w v = v - \left\langle v, \frac{w}{|w|} \right\rangle \frac{w}{|w|} = v - \frac{1}{|w|^2}\langle v, w\rangle w. \tag{4.16}$$

For $x \in M$ and the normal vector ν of M at x, P_ν is the projection onto the tangent space $T_x M$ of M at x. Interpreting the gradient as a vector $\nabla = (\partial_{x_1}, \partial_{x_2}, \partial_{x_3})$, the differential operator $P_X \nabla$ can formally be defined by

$$(P_X \nabla)_i = \sum_{j=1}^{3} \left(\delta_{ij} - \frac{X_i X_j}{|X|^2} \right) \partial_{x_j},\ i = 1,2,3, \tag{4.17}$$

where $X = (X_1, X_2, X_3)$ is a vector field on Θ and δ_{ij} is the Kronecker symbol.

As M is the zero level set of ψ, the gradient $\nabla\psi$ is normal to M and $P_{\nabla\psi}$ projects vectors onto M. It follows from (4.17) that $(P_X \nabla)u = P_X \nabla u$ for a scalar function u. Note, that the left hand side of this equation is defined by (4.17), while the right hand side is defined by (4.16).

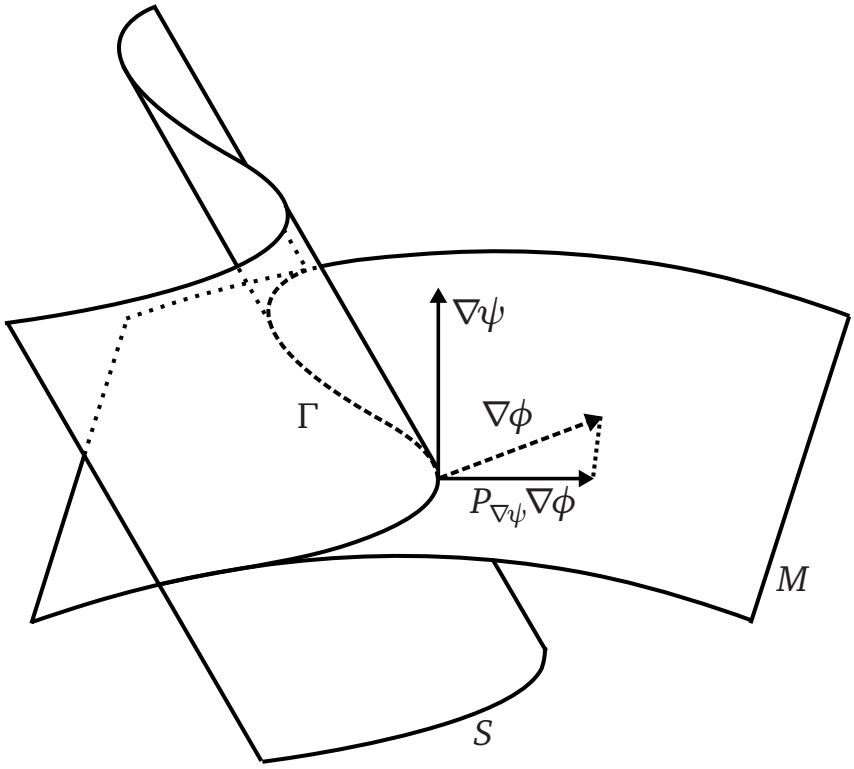


Figure 4.4: Implicitly describing a curve on a surface.

Consequently, the differential operator $P_{\nabla\psi}\nabla$ maps the function u to the tangential component of its gradient which is exactly the surface gradient. Equally, the surface divergence of a vector field X is given by $(P_{\nabla\psi}\nabla)\cdot X$ (cf. Eq. (1), p. xviii).

Geodesic curvature flow The curvature flow in $\mathbb{R}^2$ reads in level set formulation (see [OF03]):

$$\partial_t\phi = \left[\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\right]|\nabla\phi|.$$

According to (2.7), the geodesic curvature is the tangential component of the curvature. Denoting the gradient and the divergence operators on the surface by ∇_M and $\nabla_M\cdot$, respectively, the geodesic curvature flow on M is obtained after projection as:

$$\partial_t\phi = \left[\nabla_M\cdot\left(\frac{\nabla_M\phi}{|\nabla_M\phi|}\right)\right]|\nabla_M\phi|. \tag{4.18}$$

Note that $\frac{\nabla_M\phi}{|\nabla_M\phi|}$ is the co-normal field $\hat{\nu}$ in Section 4.A. Equation (4.18) can be rewritten as

$$\partial_t\phi = \underbrace{\left[\nabla\cdot\left(\frac{P_{\nabla\psi}\nabla\phi}{|P_{\nabla\psi}\nabla\phi|}|\nabla\psi|\right)\right]\frac{1}{|\nabla\psi|}}_{\kappa_g}|P_{\nabla\psi}\nabla\phi|\,, \tag{4.19}$$

Figure 4.5: Initialisation of a curve on an implicitly given face by intersecting the surface with a cylinder.

where κ_g is the geodesic curvature (cf. (2.7)).

Geodesic advection flow Let $V : M \to \mathbb{R}^3$ describe a tangential vector field on M, i.e. $\langle V, \nu \rangle = 0$ on M. If V is only defined on the surface M, it has to be extended to Θ (see Section 4.5.1). It follows that the corresponding geodesic advection flow is given by:

$$\partial_t \phi = -\langle P_{\nabla\psi} V, \nabla\phi \rangle. \tag{4.20}$$

Note, that $P_{\nabla\psi} V$ equals V on the surface M, but, in general, this is not valid for the extension of the vector field V.

Normal direction flow Analogously to the flows in Eqs. (4.19) and (4.20), Cheng et al. [CBMO02] obtained the following equation for curves evolving in direction of their co-normal:

$$\partial_t \phi = V_0 |P_{\nabla\psi} \nabla\phi| \,.$$

Here, V_0 is a scalar function.

4.4.3 The Equation for GAC on Implicit Surfaces

With Cheng et al.'s [CBMO02] framework at hand, the GAC model (4.15) can be generalised to implicit surfaces.

As in Section 4.4.2, let $M \subset \Theta$ be a surface embedded in the cuboid Θ and implicitly given by ψ. A greyscale image I on M is a mapping $I : M \to \mathbb{R}^+$. To apply Cheng et al.'s [CBMO02] framework, the image I has to be smoothly extended to at least a certain band surrounding M (see Section 4.5.1).

As in the classical GAC model, the image I should be smoothed before an edge indicator function is calculated. If a textured image is mapped onto the surface, this can be achieved by applying a low pass 2D filter to the image. Alternatively, images can be smoothed directly on an implicit surface [BMC$^+$03], or by applying a 3D filter to the volumetric image I.

If I_σ is a smoothed image, the edge indicator function is defined as:

$$f = \frac{1}{1 + |\nabla I_\sigma|^p}, \quad p \in \{1, 2\}.$$

Note, that f can be defined in terms of geometric quantities, such as the surface curvature, as well (see the experiments in Section 4.6.3). The proposed equation for GACs on implicit surfaces reads

$$\begin{aligned} \partial_t \phi &= \underbrace{cf\,|P_{\nabla\psi}\nabla\phi| + \langle P_{\nabla\psi}\nabla f, \nabla\phi\rangle}_{I} \\ &\quad + \underbrace{\mu f \kappa_g |P_{\nabla\psi}\nabla\phi|}_{II} . \end{aligned} \tag{4.21}$$

Here, the parameters c and μ determine the influence of the balloon force and the regularity terms, respectively, while the image gradient term has a normalised weight. The evolving contour on the surface M is given by the zero level set:

$$\Gamma(t) = \left\{ x \in \Theta : \phi(x, t) = \psi(x) = 0 \right\} .$$

The initial position of the curve is defined by $\phi(., 0)$. It is illustrated in Fig. 4.5 how the initial curve on a face surface is implemented.

With (4.21), GACs on implicit surfaces have been introduced. The next section looks more closely at their numerical implementation.

4.5 Numerical Implementation

4.5.1 General Issues

Common level set toolboxes such as [Mit08] are very useful for the implementation of curve evolution in $\mathbb{R}^2$ and surface evolution in $\mathbb{R}^3$. Yet they are not flexible enough to incorporate the non-standard terms in GAC equation (4.21), e.g. the geodesic curvature. Therefore I proceed along the lines of Cheng et al.'s [CBMO02] framework overviewed in Section 4.4.2 to implement PDE (4.21). Many details are omitted in [CBMO02], moreover, their scheme has to be adapted to the particular PDE for GACs. Thus, the numerical implementation of the model described in this subsection is original, unless stated otherwise. A new, optimised narrow band algorithm is presented in Section 4.5.2.

The PDE (4.21) for ϕ can be solved numerically on a uniform grid in $\Theta \subset \mathbb{R}^3$ by finite difference schemes. The reader is referred to Section 2.2 for a quick review of the level set method, and to [OF03] for a comprehensive introduction to the field and further reference.

For the geodesic curvature flow, Cheng et al. [CBMO02] showed that the resulting evolution equation (4.19) is degenerate second-order parabolic. Thus, central differences can be used to discretise the term II in (4.21). Term I is of HJ type, and the derivatives have to be computed using Essentially Non Oscillating (ENO)- or Weighted Essentially Non Oscillating (WENO) schemes to avoid numerical instabilities [OS88, OF03]. The experiments confirmed the results in [CBMO02]: fifth-order WENO schemes are necessary to accurately reinitialise the level set function (see Section 4.5.1). For the GAC equation (4.21), second-order ENO yields similar results as fifth-order WENO. As no significant difference in the processing times could be observed, fifth-order WENO was chosen for the implementation. The time derivative is approximated by a forward Euler discretisation.

The Curvature Term

Consider (4.21): as stated in (4.19), the geodesic curvature can be written as

$$\kappa_g = \left[\nabla \cdot \left(\frac{P_{\nabla\psi} \nabla\phi}{|P_{\nabla\psi} \nabla\phi|} |\nabla\psi| \right) \right] \frac{1}{|\nabla\psi|} \,. \tag{4.22}$$

The projected gradient term is

$$P_{\nabla\psi} \nabla\phi = \nabla\phi - \frac{1}{|\nabla\psi|^2} \langle \nabla\psi, \nabla\phi \rangle \nabla\psi. \tag{4.23}$$

One way to discretise κ_g is to combine Eqs. (4.22) and (4.23) and express the resulting equation in terms of the first and second derivatives of ϕ and ψ. While this may be the

most accurate way to calculate the geodesic curvature, the resulting term is computationally expensive. In the proposed algorithm, an alternative approach is pursued: the normalised surface gradient $\frac{P_{\nabla\psi}\nabla\phi}{|P_{\nabla\psi}\nabla\phi|}$ is discretised first, before discretising the divergence operator $\nabla\cdot$ in the second step, both by central differences.

The Hamilton-Jacobi Term

As stated previously, the second term (II) in (4.21) is of Hamilton-Jacobi type and therefore needs to be treated carefully in order to avoid numerical instabilities. The Hamiltonian H (see Section 2.2.5) is given by

$$H(\mathbf{x}) = cf\,|P_{\nabla\psi}\mathbf{x}| + \langle \mathbf{V}, \mathbf{x}\rangle\ . \tag{4.24}$$

Here, the following substitutions have been made: $\mathbf{x} = \nabla\phi$ and $\mathbf{V} = (v_1, v_2, v_3) = P_{\nabla\psi}\nabla f$. In order to approximate (4.24), a Lax-Friedrichs (LF) scheme with global flux (see [OF03]) is used. The partial derivatives of H are given by

$$\begin{aligned}
\partial_{x_i} H &= \partial_{x_i}\left(cf\,|\mathbf{x} - \frac{1}{|\nabla\psi|^2}\langle\nabla\psi, \mathbf{x}\rangle\nabla\psi| + \langle \mathbf{V}, \mathbf{x}\rangle \right) && (4.25)\\
&= \left\langle cf\,\frac{P_{\nabla\psi}\mathbf{x}}{|P_{\nabla\psi}\mathbf{x}|}, e_i - \frac{(\nabla\psi)_i}{|\nabla\psi|^2}\nabla\psi \right\rangle + v_i && (4.26)\\
&= cf\,\frac{(P_{\nabla\psi}\mathbf{x})_i}{|P_{\nabla\psi}\mathbf{x}|} - cf\,\frac{(\nabla\psi)_i}{|P_{\nabla\psi}\mathbf{x}||\nabla\psi|^2}\langle P_{\nabla\psi}\mathbf{x}, \nabla\psi\rangle + v_i && (4.27)\\
&= cf\,\frac{(P_{\nabla\psi}\mathbf{x})_i}{|P_{\nabla\psi}\mathbf{x}|} + v_i\ , && (4.28)
\end{aligned}$$

where e_i is the i-th canonical unity basis vector in $\mathbb{R}^3$ and $(.)_i$ denotes the i-th component of a vector. The chain rule is applied in (4.25), and the second summand in (4.27) vanishes as $\langle P_{\nabla\psi}\mathbf{x}, \nabla\psi\rangle = 0$ by the definition of the projection operator.

Note, that resuming (4.28) the dissipation coefficients in the LF scheme can be estimated as follows:

$$\alpha_i = \max|\partial_{x_i} H| \leq \max(|c|f + |v_i|). \tag{4.29}$$

Experiments have shown that this estimate speeds up the algorithm by about 15%, but the accuracy can suffer, since too much dissipation is added to the solution. Therefore, a pointwise estimation of the coefficients (global LF scheme) is used.

Reinitialisation

As proposed by Cheng et al. [CBMO02], the following sub-steps should be executed regularly to reinitialise the level set function ϕ and improve the numerical behaviour of the

equations:

1. Transforming ϕ to a signed distance function on the surface M;

2. Enforcing ϕ to have its level sets perpendicular to M.

The first sub-step is standard in the level set theory, yet it has to be adapted to the surface framework. A level set function ϕ_0 can be transformed to a signed distance function by solving the equation $\partial_t \phi = \text{sign}(\phi_0)(1-|\nabla\phi|)$ (see Section 2.2.7). Accordingly, in the surface framework the equation reads:

$$\partial_t \phi = \text{sign}(\phi_0)(1 - |P_{\nabla\psi}\nabla\phi|). \tag{4.30}$$

Problems that arise, when the level sets of ϕ off the surface become approximately tangential to the surface, are avoided with the second sub-step. Cheng et al. [CBMO02] proposed to iterate a few steps of the PDE

$$\partial_t \phi + \text{sign}(\psi)\left\langle \frac{\nabla\psi}{|\nabla\psi|}, \nabla\phi \right\rangle = 0 \tag{4.31}$$

at each step of the flow. Note, that on the surface M, where $\psi = 0$, the function ϕ is not changed. Elsewhere, ϕ evolves towards the steady-state solution, which implies $\langle\nabla\psi, \nabla\phi\rangle = 0$ and thus level sets of ϕ that are perpendicular to M.

Equation (4.31) can also be used to extend a scalar function u, defined only on the surface M, to a neighbourhood of M or even the whole set Θ. The function is extended in such a form, that it is constant in normal direction to the level sets of ψ, i.e. $\langle\nabla\psi, \nabla u\rangle = 0$.

4.5.2 The Narrow band Algorithm

One important drawback of level set approaches in general is the increased computational complexity. While a curve is a 1D object, its evolution is transformed to a 2D problem in the standard plane level set method. Thus, the computational complexity is increased by one order of magnitude to $O(n^2)$ for a $n \times n$ image. In the framework discussed here, the implications are even more disadvantageous: in order to evolve a curve on an implicit surface, the 3D function ϕ has to be updated in every time step, yielding a complexity of $O(n^3)$ for a surface embedded in a $n \times n \times n$-cube.

To speed up the computations, Cheng et al. [CBMO02] localised the numerical calculations to a neighbourhood of the surface, obtaining a computational complexity of $O(n^2)$ for

most surfaces. This algorithm is easy to implement, since the surface is static and the data structure holding the grid points being close to the surface has to be created only once.

Only information from a neighbourhood of the curve is required to implement (4.21), hence Cheng et al.'s [CBMO02] procedure is still not optimal for this purpose. For this reason, an optimised algorithm for the evolution of GACs on implicit surfaces is proposed here. Its complexity is linear in the length of the curve, i.e. $O(||\Gamma||)$. For contours appearing in practice (i.e. non-pathological curves), this corresponds to a complexity of $O(n)$, when M is embedded in a $n \times n \times n$-cube. Essentially, the approaches given in [CBMO02, PMZ$^+$99] are extended and generalised.

Let $\tilde{\psi}$ be a signed distance function for M, which can be computed by e.g. the fast marching method [Set96]. The evolving curve is represented by the scalar 3D function ϕ, i.e. $\Gamma(t) = \{x \in \Theta : \psi(x) = \phi(x,t) = 0\}$. It is assumed that ϕ is close to a signed distance function on M at a certain time t_0 (e.g. $t_0 = 0$), and $\tilde{\phi} := \phi(.,t_0)$ is retained (see below). The main idea is to define a function θ_p that approximates the distance of a grid point in space from the evolving curve at time t_0. Every p-norm of $\tilde{\psi}$ and $\tilde{\phi}$, i.e.

$$\theta_p(x) := \left(|\tilde{\psi}(x)|^p + |\tilde{\phi}(x)|^p\right)^{1/p}, \quad 1 \leq p < \infty \tag{4.32}$$

serves this purpose. The special case of the maximum norm is given for $p = \infty$ by

$$\theta_\infty(x) := \max\{|\tilde{\psi}(x)|, |\tilde{\phi}(x)|\} \tag{4.33}$$

As both $\tilde{\psi}$ and $\tilde{\phi}$ are close to signed distance functions in Θ and on the surface M, respectively, level sets of θ_p resemble circular or rectangular tubes along the curve Γ, if $p = 2$ or $p = \infty$ are chosen, respectively.

The smaller p is chosen, the smaller is the narrow band along the curve. This increases the efficiency of the algorithm while it can affect the accuracy and stability of the numerical solution. The most common cases – $p = 2$ and $p = \infty$ – deliver good results; $p = 2$ is chosen for higher efficiency throughout the experiments in this thesis. Tests showed that the choice of $p = 1$ results in large gradients at the tube boundary and thus poor numerical stability.

Having defined θ_p, one can follow [PMZ$^+$99] to localise level set equations. The following strategy is used: $\tilde{\phi} := \phi(.,t_0)$ is retained as the approximate signed distance function on the surface M. Based on the function θ_p, a narrow band T along the curve is computed. The extended narrow band N includes T, as well as all neighbouring grid points. The level set

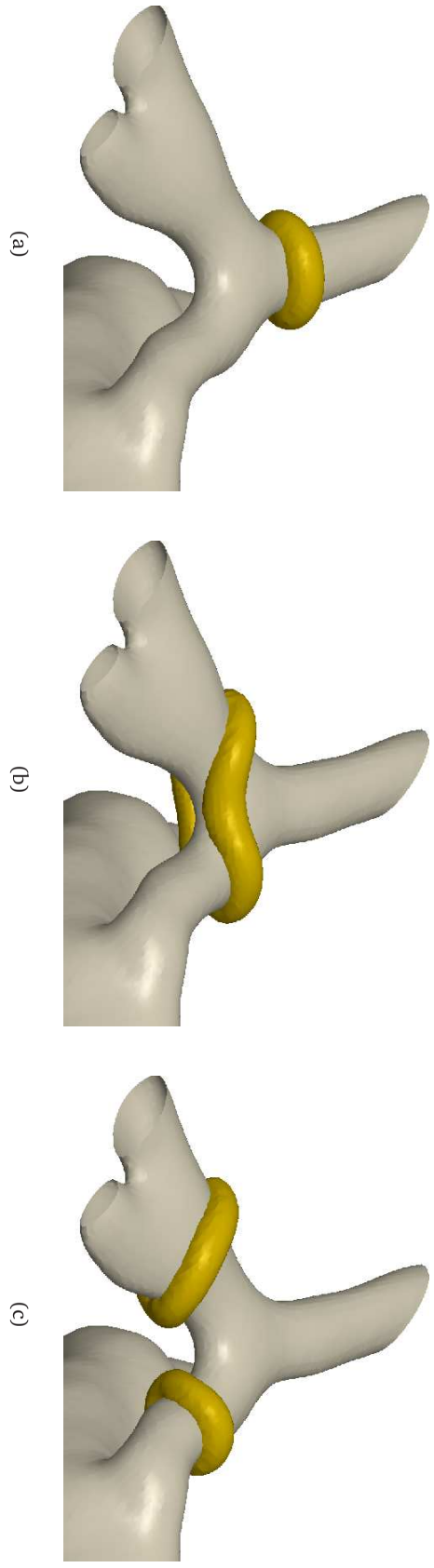


Figure 4.6: Evolving narrow band on a vascular surface.

function ϕ is updated in T by executing time steps for a modified level set PDE (see (4.70) in Appendix 4.C) until one of the following conditions is met:

1. The curve reaches a certain inner tube;
2. The surface gradient of ϕ diverges from unit length too much;
3. A maximum number of steps is reached, irrespective of the other two criteria.

Then, ϕ is reinitialised in N by iterating a few steps of the PDEs (4.30) and (4.31), and $\tilde{\phi} := \phi(., t_0)$ is retained. Now, the loop can be started over with an update of the narrow band. An evolving narrow band on a vascular surface is depicted in Fig. 4.6. Note, that it naturally splits at the junction.

The above algorithm is detailed in Appendix 4.C.

4.6 Experimental Results

For brevity, the proposed approach on implicit surfaces is denoted by *3D GAC*. It is closely related to the approaches in [SK07, BBTV07] where the GAC method is generalised using parametric surface representations. Yet only Spira and Kimmel's [SK07] scheme is applicable to a large class of surfaces, hence it is considered here as the state of the art. Bogdanova et al.'s [BBTV07] algorithm is restricted to three simple model surfaces (see Section 4.2.2), and Hara et al.'s work [HKIU07] is limited to surfaces close to a sphere. Tian et al.'s method [TMR09] generalises the Chan-Vese model [CV01] to surfaces and is applicable to a large class of them. Yet it appears to be unsuitable for feature segmentation, as ridges cannot be detected by a region-based model. The same limitations hold for the scheme recently suggested by Delaunoy et al. [DFPH09].

The 3D GAC algorithm was developed in MATLAB® and tested on an Intel Core 2 Duo 2.5 GHz machine. Since the code for Spira and Kimmel's algorithm [SK07] is not publicly available, the author coded it himself, also in MATLAB.

4.6.1 Comparison on Synthetic Surfaces

In Section 4.3 it was proven theoretically that a balloon force cannot be incorporated properly into GAC models on parametric surfaces. Experimental evidence for this effect is presented in this section.

For comparative purposes both Spira and Kimmel's algorithm [SK07] and the 3D GAC are

first applied to a plane. A square with some additive Gaussian noise (Fig. 4.7(a)) is used as texture. Table 4.1 shows that both approaches yield similar results in terms of accuracy. The parametric algorithm is about twice as fast, reflecting the lower intrinsic complexity of this approach.

In the next experiment a square without any noise is mapped on a pyramid surface (Fig. 4.7(b)). Subsequently, the performance of the 3D GAC, including a balloon term, is compared to Spira and Kimmel's scheme without a balloon term. As shown in Table 4.1 the implicit scheme is about twice as fast and displays a better accuracy. Goal of the next two experiments is to illustrate the scaling effect analysed in Section 4.3. For the sake of clarity, the surfaces considered here are simple function graphs over a plane and curved only in the direction of one axis. The parametrisations have the form (cf. Eq. (4.3)):

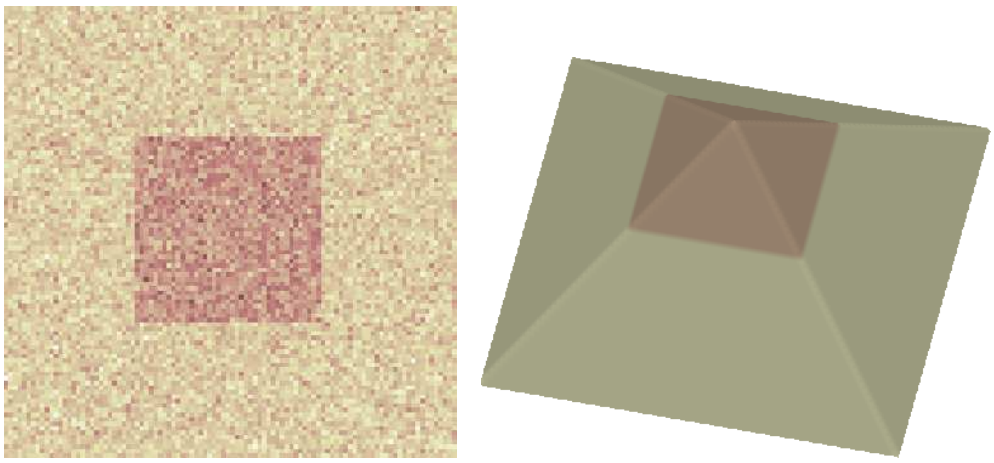
$$X(u_1, u_2) = \big(u_1, u_2, Z(u_1)\big)^\top \ . \tag{4.34}$$

Consequently, the local basis $X_i, i \in \{1, 2\}$ of the tangent space is given by

$$X_1 = \Big(1, 0, \partial_{u_1} Z\Big)^\top \qquad \text{and} \qquad X_2 = (0, 1, 0)^\top \ . \tag{4.35}$$

This yields the following metric tensor:

$$\big[g_{ij}\big] = \begin{pmatrix} 1 + (\partial_{u_1} Z)^2 & 0 \\ 0 & 1 \end{pmatrix} . \tag{4.36}$$



(a) (b)

Figure 4.7: Simple texture projected on a plane (a) and a pyramid (b).

			3D GAC		GAC Spira [SK07]		
Surface	Grid size	Noise	l^∞-err.	time (in s)	BF	l^∞-err.	time (in s)
Plane	$100 \times 100 \times 30$	0.003	1.78	125	yes	1.68	62
Pyramid	$100 \times 100 \times 65$	none	0.63	38	no	1.21	75
Halfpipe	$130 \times 140 \times 75$	0.003	2.75	329	yes/no	fails	
Notebook	120^3	0.001	1.52	100	yes/no	fails	
Sphere	120^3	0.001	1.52	241	yes/no	not applicable	

Table 4.1: Comparison of the 3D GAC with Spira and Kimmel's method [SK07] (parameters: $c = 0.4$ and $\mu = 1$, cf. (4.21)). The textures contain additive Gaussian noise with given variance values. 'BF' means Balloon force, the parameterisation plane had the size of the first two dimensions of the implicit grid: e.g. $100\times100\times30$ grid means 100×100 parameterisation plane. Time step for the parametric model: $\tau = 0.4$; for the implicit models: $\tau = 0.4$ for the plane and the pyramid, and $\tau = 1.0$ otherwise.

Equation (4.36) indicates that in regions of high surface slope - with respect to the parametrisation plane - the metric tensor deviates considerably from the unit matrix. According to the analysis in Section 4.3, the problem of different scaling factors has to become apparent in these regions. In order to test this empirically, two simple synthetic textures are mapped on a halfpipe-surface and a plane with a kink, respectively. For convenience, the latter is called a notebook-surface.

The first two rows of Fig. 4.8 show the experimental results: Spira and Kimmel's [SK07] model without a balloon force fails, even with very little added noise. After including a balloon term, the expected effect can be observed: in areas of high surface slope the contour does not converge, since the balloon force is weighted too high. More precisely, the contour passes the desired object boundary only in the direction in which the metric (4.36) varies. This is particularly apparent on the notebook-surface (Fig. 4.8(g)). Note, that the proposed algorithm, using the implicit surface representation, succeeds to segment the objects of interest (Figs. 4.8(d),(h)). Figures 4.8(i)-(k) show the segmentation process on a fully implicit sphere. The latter exemplifies a class of surfaces where Spira and Kimmel's [SK07] algorithm cannot be applied due to poles in the parametrisation. For many surfaces poles cannot be avoided, and the numerics of the parametric approach break down close to them (cf. the experiments in [SK07] where the poles of the sphere are cut).

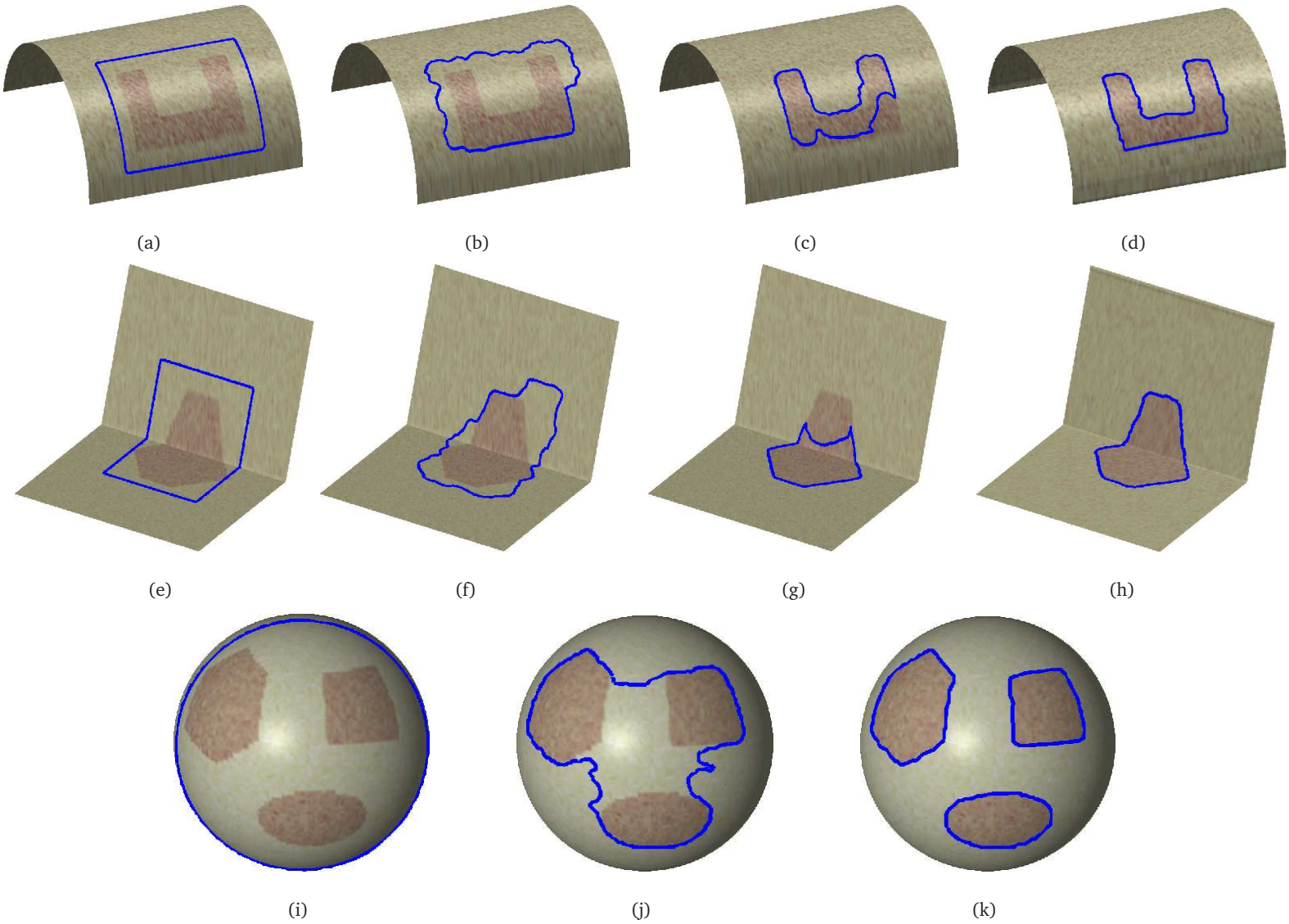


Figure 4.8: (a) and (e): initial contours on a halfpipe and a notebook surface, resulting contours using Spira's parametric approach without (b,f) and with (c,g) an added balloon force. (d) and (h): results of the implicit method, (i)-(k): GAC evolution on an implicit sphere.

4.6.2 Handling of Noise

The narrow band evolving along the contour depends directly on the surface level set function ψ. Hence, the impact of noise on the accuracy of the segmentation has to be studied. For the three scenarios shown in Fig. 4.8, different levels of additive Gaussian noise were applied to both, surface- and texture data (see Figs. 4.9 and 4.10). Afterwards, standard Gaussian filtering was applied. Figure 4.10 indicates that the segmentation accuracy is significantly influenced by the level of texture noise. For the three studied levels of surface noise, however, the segmentation accuracy is on a similar level. These experiments suggest that for levels of noise occurring in applications (e.g. data from a 3D scanner), the accuracy of the implicit algorithm is not significantly affected. Note, that all accuracy values were averaged over three runs.

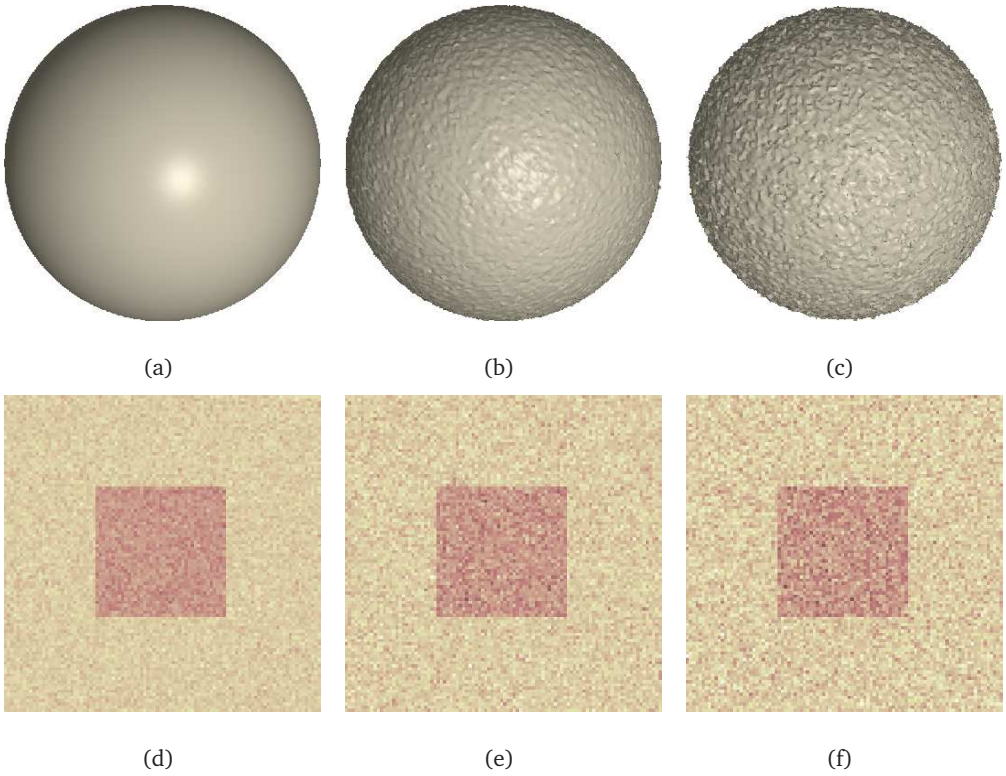


Figure 4.9: First row: Gaussian noise of different levels (variance $\mu = 0, 0.25, 0.5$) added to a sphere surface. Second row: Gaussian noise of different levels (variance $\mu = 0.001, 0.002, 0.003$) added to a synthetic image.

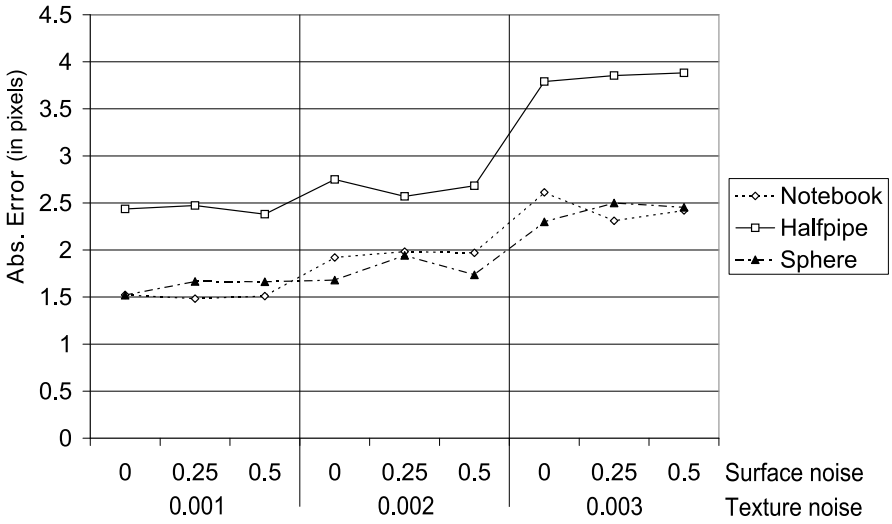


Figure 4.10: Error analysis of the 3D GAC model synthetic data under different forms of noise. Gaussian noise of given variance was added to both texture and surface data.

4.6.3 Applications

Experiments show that in contrast to approaches using surface parametrisations, the proposed GAC framework is applicable to a wide range of surfaces. Examples for applications of the proposed algorithm are given below.

3D Face Feature Segmentation The suggested algorithm was tested for 3D lip segmentation on 50 datasets randomly chosen from the BJUT database [bju05]. Quantitative comparisons were made with other techniques including Spira and Kimmel's [SK07] algorithm. The stopping term for the GAC was derived from mean curvature information of the face surface, see Section 5.5.1 for details.

The parameters were kept constant accross all datasets: $\mu = 0.2$ and $c = -0.025$[1]. In these experiments manually delineated lip contours, based on curvature data, served as a ground truth. A comparison with texture-based contours follows in Chapter 7. Table 4.6.3 shows the overall statistics for both the 3D GACs and Spira and Kimmel's method. The table indicates similar characteristics for the processing time and the median of the mean error distribution. Yet the arithmetic mean indicates that the accuracy of the 3D GAC is about 8% better than Spira and Kimmel's GAC. The cumulative distribution curves in Fig. 4.11 provide further insight into the performance of both methods. Up to the 80% mark both curves run in a similar manner. Then the curve of Spira's GAC flattens noticeably, indicating quickly deteriorating accuracy results for the remaining 20% of the datasets. By contrast, the curve of the 3D GACs suggests a significantly better performance on the remaining datasets.

[1]Empirical test showed that for both tested methods these parameters provided good results which worsened with slight variation of the parameters. Therefore, the parameters are, in a sense, locally optimal.

Method	Mean	Max.	Std.	Mean Error	Processing
	Error (pixels)			median (pixels)	time (in s)
Spira's GAC	0.94 ± 0.39	4.0 ± 2.7	0.9 ± 0.5	0.8	226 ± 84
3D GAC	0.86 ± 0.29	3.7 ± 1.7	0.8 ± 0.4	0.8	220 ± 74

Table 4.2: Accuracy evaluation for lip contour segmentation compared to curvature based ground truth.

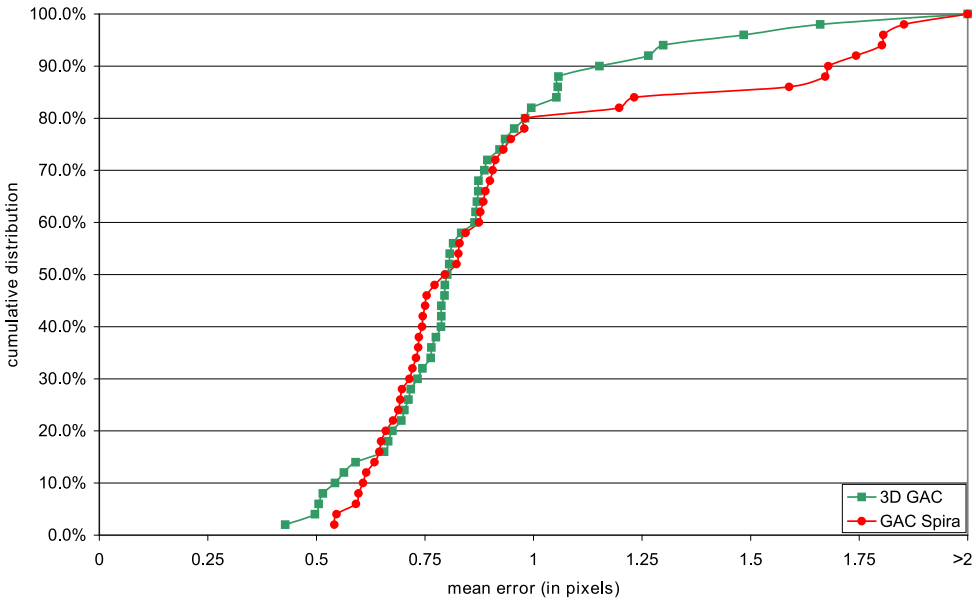


Figure 4.11: Evaluation of lip contour segmentation accuracy with respect to manually delineated ground truth based on curvature data.

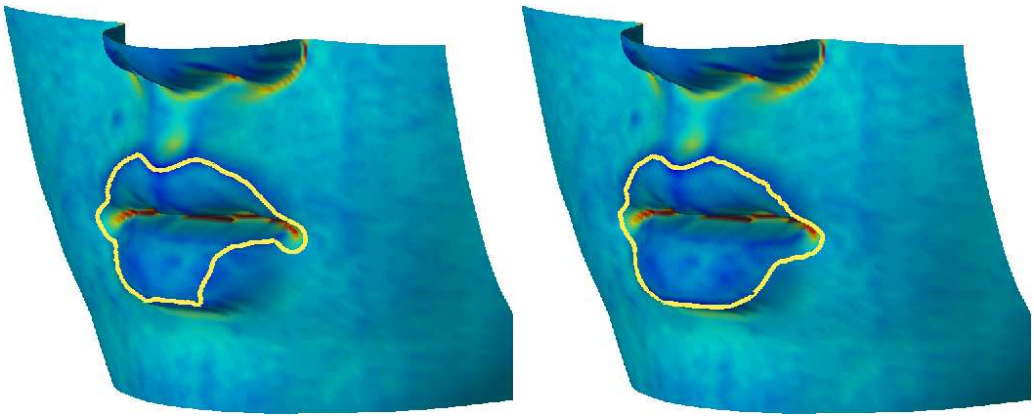


Figure 4.12: Sample dataset where Spira's GAC (left) fails, while the 3D GAC (right) converges to a useful solution.

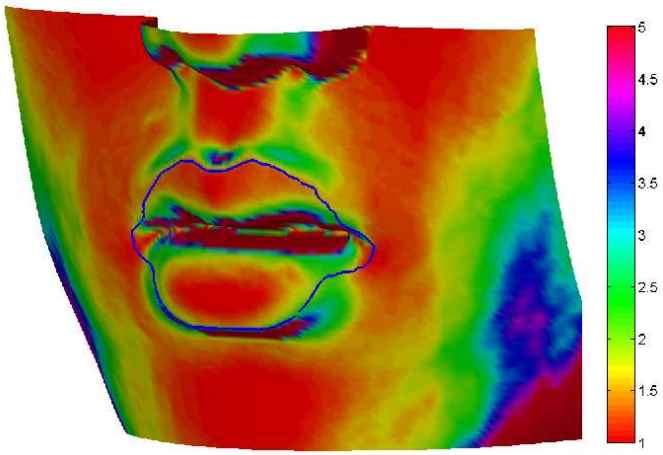
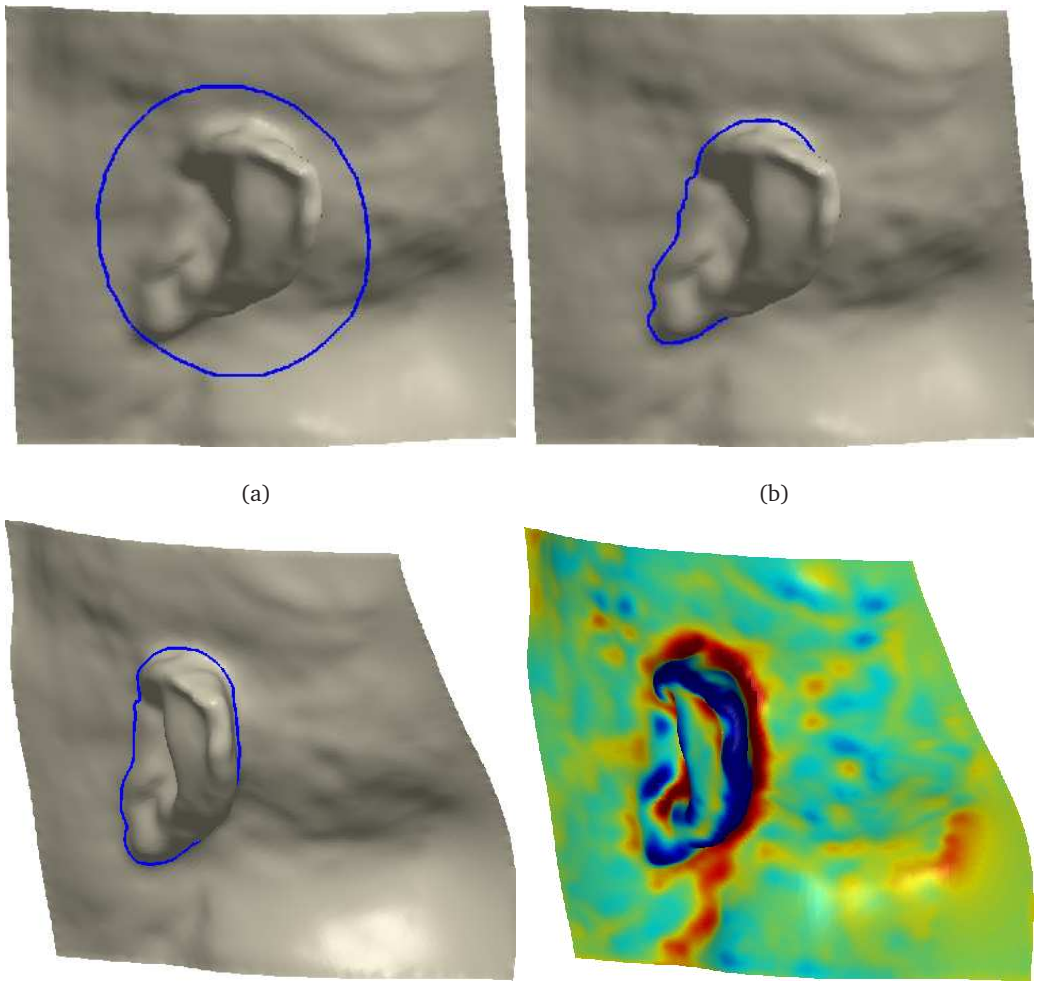


Figure 4.13: Mouth region colorised with the metric Gramian g and 3D lip contour obtained with the proposed algorithm.

Figure 4.12 shows an example where the 3D GAC succeeds to segment the lip contours while Spira and Kimmel's GAC fails. Note, that this occurs at a spot location with steep slope, and thus the metric tensor differs significantly from the unit matrix (Fig. 4.13). The scaling issue discussed in Section 4.3.3 causes the balloon term to be weighted too high, and the contour to pass the desired ridge.

Another potential application of 3D active contours is 3D ear segmentation. In Fig. 4.14, the implicit model is applied to the Max- Planck bust [pri]. The stopping function f used here is a slightly modified version of f_1 in Section 5.5.1, attracting the contour to areas of positive- (the yellow/red areas in Fig. 4.14 (d)), instead of negative mean curvature.



(a) (b)

(c) (d)

Figure 4.14: 3D Ear segmentation: Initial (a) and final (b) contour, different view (c), mean curvature colorised surface (d).

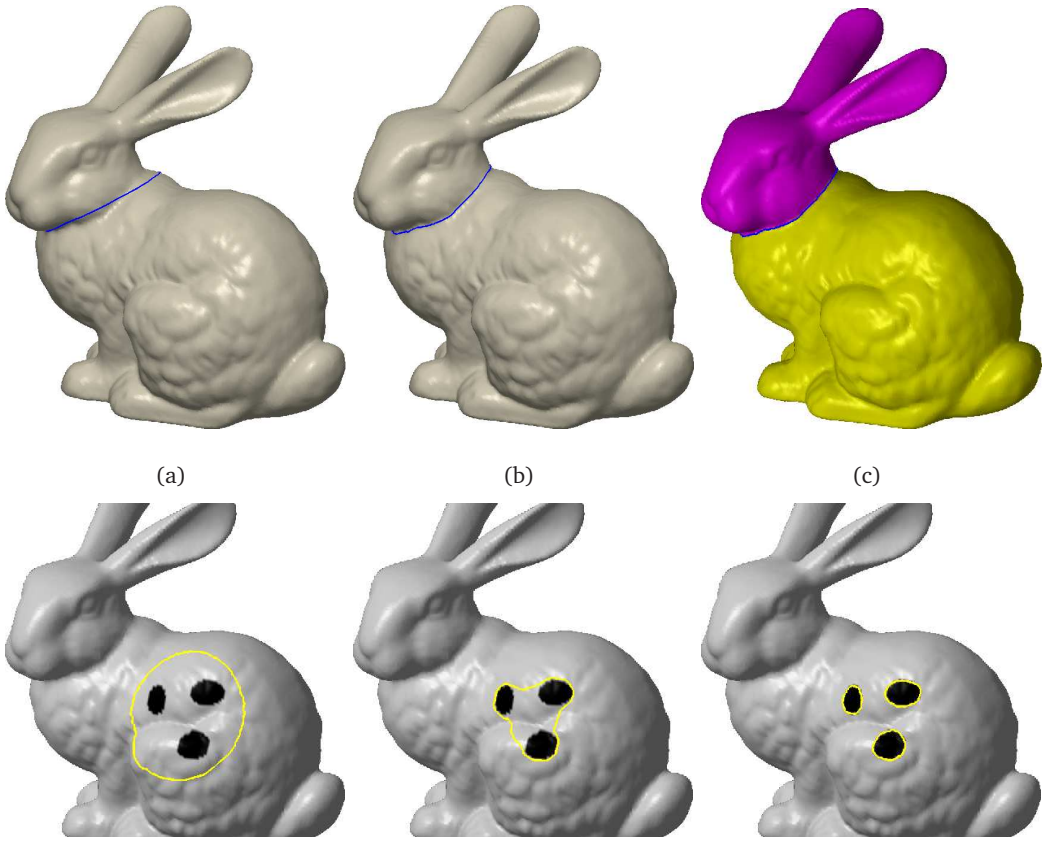
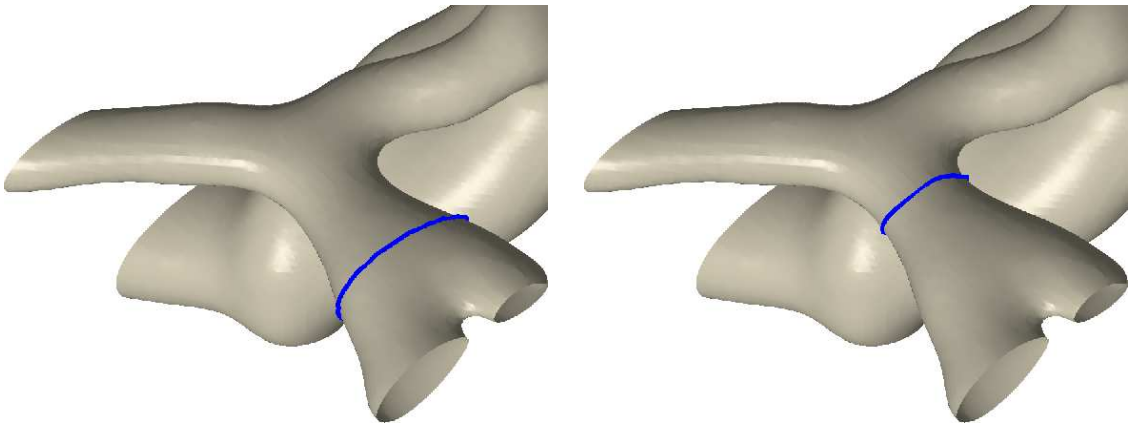


Figure 4.15: The proposed GAC model applied to the Stanford bunny. (a)-(c) shows segmentation of bunny's head by surface curvature data, (d)-(f) shows segmentation by texture.

		Implicit				Parametric			
Surface	Grid size	c	μ	τ	time (in s)	c	μ	τ	time
Lips (average)	varies	0.025	0.2	0.7	220	0.025	0.2	0.6	226 s
Ear	$123 \times 82 \times 114$	0.04	0.5	0.7	298	not applicable			
Blood vessel	$172 \times 120 \times 96$	0.3	1.0	1.0	79	not applicable			
Bunny dots	$141 \times 176 \times 175$	0.4	1.0	0.4	62	not applicable			
Bunny head	$141 \times 176 \times 175$	0.03	0.5	1.0	158	not applicable			

Table 4.3: Performance of the 3D GAC. Note, that only the 3D lips segmentation experiments could be compared with Spira's approach due to the issues discussed previously in Section 4.3.4. The parameters are denoted as in (4.21), τ is the timestep. In the lip segmentation tests a smaller timestep for Spira's algorithm had to be chosen in order to prevent numerical instabilities.



(a) (b)

Figure 4.16: Detection of the locally narrowest spot (constriction) on a vascular surface: Initial position (a) and position after convergence (b).

Mesh segmentation Segmentation tools for triangulated meshes [JK04, LL02, LLS$^+$05] commonly use intrinsic geometric information, such as the curvature, to partition a mesh surface into meaningful parts. It is demonstrated on the Stanford bunny (Fig. 4.15) that the proposed algorithm can be applied to surface segmentation by intrinsic- ((a)-(c)) and texture features ((d)-(f)). For the segmentation of bunny's head the same stopping function as for the 3D ear example was used. Note, that the bunny surface is topologically too complex to obtain a global parametrisation. Similarly, cuts and discontinuities occur when parametrising the 3D ear in Fig. 4.14 [SPR06, ZMT05]. Therefore no comparison to Spira and Kimmel [SK07] was tried.

Medical imaging A vascular constriction is the narrowing of a blood vessel. Mathematically speaking, a constriction is a part of a surface with locally minimal perimeter (cf. Fig. 4.16(b), [HA03, BWK05]). Figure 4.16 shows a vascular surface segmented from a volumetric image. The contour is initialised close to the constriction (Fig. 4.16(a)). Note, that the stopping term is set $f = 1$ in this experiment. In this case, PDE (4.21) essentially models a curve shortening flow (term II) with a supporting balloon force (first summand of term I) to accelerate the convergence. A linearly decreasing balloon term weight is chosen: close to the constriction the balloon force becomes zero, and the contour converges accurately towards the locally shortest curve.

4.7 Discussion and Conclusions

This chapter has proposed a new algorithm for active contour evolution on curved surfaces. It generalises the popular geodesic active contour model to non-Euclidean geometries. The problems intrinsic to existing approaches on parametric surfaces are avoided, when the surface is embedded implicitly into $\mathbb{R}^3$, and the Euclidean geometry of $\mathbb{R}^3$ is used to compute the contour evolution. This metric is constant on the surface, and the scaling problem for the balloon term does not occur. In order to reduce the increased computational workload, a new narrow band scheme, generalising this well-known technique to 3D curve evolution, has been designed. Examples on synthetic and natural surfaces have been provided, where the known works fail or are not applicable, while the 3D GAC delivers the expected results. Potential fields of application include 3D face/head analysis, medical imaging and mesh segmentation.

The theoretical, as well as the experimental results presented here, suggest that the proposed GAC model is the natural generalisation of the classic GACs to curved surfaces. It is fully implicit, and a balloon term can be readily incorporated. Yet despite the reduced computational complexity of the narrow band scheme, the processing times are still prohibitive for time sensitive applications. Note, that the algorithm has been tested in MATLAB so far, and an implementation of the code in C++ may further reduce the processing times.

Conclusions

1. *Integrating a balloon term into the GAC model on parametric surfaces results in a PDE that does not scale homogeneously. This intrinsic property can lead to unexpected and undesirable segmentation results;*

2. *Using the implicit surface representation avoids the above problem, as the curve evolution is computed with respect to the constant Euclidean metric in $\mathbb{R}^3$, instead of the variable metric in the parametrisation plane;*

3. *With the 3D narrow band optimisation scheme, the proposed algorithm has a computational complexity that is linear in the length of the evolving contour;*

4. *Yet processing times might be prohibitive for time sensitive applications.*

4.A Variational Foundations

In this appendix, the equation for the GAC on curved surfaces will be derived from the respective geometric energy functionals. To the best of the author's knowledge, those derivations cannot be found elsewhere in the literature. Spira et al. [SK07] did not study the variational background of GAC on surfaces. Bogdanova et al. [BBTV07] only considered GAC on three specific surfaces. Neither of these authors studied constant velocity (balloon) terms on surfaces, as they are required for the proposed model.

4.A.1 Arc length Minimisation

It is well-known that the L^2-gradient flow of the arc length functional $E(\Gamma) = \int_\Gamma ds$ is the flow of a curve by its curvature:

$$\partial_t \Gamma = \kappa \cdot \nu \,, \tag{4.37}$$

where Γ is a curve and $\partial_t \Gamma := \frac{\partial \Gamma}{\partial t}$, κ, and ν are the time derivative, the curvature, and the unit normal field of the curve, respectively. Hence, evolution of a curve under curvature flow locally decreases its length as fast as possible. The curvature flow was first studied in [GH86], and it was shown [Gra87] that it shrinks planar curves into round points in finite time. In [CCCD93] the curvature flow was applied to active contours for the first time. Later, the geodesic active contours were introduced in [CKS97, KKO+96]. They base on an energy functional that integrates a positive edge indicator function f along the contour Γ:

$$E(\Gamma) = \int_\Gamma f(s)\, ds \,. \tag{4.38}$$

This formalisation yields the evolution equation:

$$\partial_t \Gamma = \big(\kappa f - \langle \nabla f, \nu \rangle\big)\, \nu \,, \tag{4.39}$$

see Theorem 5 and Corollary 6 in Chapter 2. The generalisation of these ideas is the subject of the following theorem.

Theorem 11. *Let $M \subset \mathbb{R}^3$ be a smooth hypersurface, $f : M \to \mathbb{R}$ a smooth scalar function and Γ a closed curve on M. Then the L^2 gradient flow of the weighted arc length functional*

$$L_f = \int_\Gamma f(s)\, ds \tag{4.40}$$

on the surface M is given by the evolution equation

$$\partial_t \Gamma = f\, (\partial_{ss} \Gamma)^{\|} - \nabla_M f \tag{4.41}$$

where ∂_s is the arc length derivative, ∇_M the surface gradient on M, and $\|$ denotes the component of a vector that is tangential to the surface. An alternative formulation of the flow equation is

$$\partial_t \Gamma = \left(\kappa_g f - \langle \nabla_M f, \hat{\nu} \rangle\right) \hat{\nu} \,, \tag{4.42}$$

with κ_g the geodesic curvature and $\hat{\nu}$ the unit co-normal (cf. Fig. 4.1).

Proof. Let Γ be extended to a parameter family of curves $\Gamma : J \times (-\epsilon, \epsilon) \to M$, $(r,t) \mapsto \Gamma_t(r)$ with $\Gamma_0 = \Gamma$, and J the parametrisation interval. It can be assumed that the curve Γ_0 is parametrised by its arc length, i.e. $|\partial_r \Gamma_0| = 1$ or equivalently (cf. Proposition 2 in Chapter 2):

$$\partial_r \Gamma_0 = \partial_s \Gamma_0 \,. \tag{4.43}$$

Then, computing the time derivative of $L_f(t) := L_f(\Gamma(.,t))$ yields:

$$\begin{aligned} L_f'(t)|_{t=0} &= \partial_t \left(\int_J f(\Gamma_t) \sqrt{\langle \partial_r \Gamma_t, \partial_r \Gamma_t \rangle}\, dr \right) |_{t=0} \\ &= \left(\int_J \langle \nabla_M f, \partial_t \Gamma_t \rangle \sqrt{\langle \partial_r \Gamma_t, \partial_r \Gamma_t \rangle} \right. \\ &\quad \left. + f(\Gamma_t) \frac{\langle \partial_t \partial_r \Gamma_t, \partial_r \Gamma_t \rangle}{|\partial_r \Gamma_t|}\, dr \right) |_{t=0} \end{aligned} \tag{4.44}$$

Note, that $\partial_t(f(\Gamma)) = \langle \nabla_M f, \partial_t \Gamma \rangle$ by the very definition of the surface gradient (see Def. 7, p.13). The next step is to switch the order of derivation in the second summand and integrate by parts (cf. Theorem 5). No boundary term occurs as the curve is closed. Further, $t = 0$ is fixed in this step and Eq. (4.43) in conjunction with $\Gamma_0 = \Gamma$ is applied:

$$\begin{aligned} L_f'(t)|_{t=0} &= \int_\Gamma \big(\langle \nabla_M f, \partial_t \Gamma \rangle - \langle \nabla_M f, \partial_s \Gamma \rangle \langle \partial_t \Gamma, \partial_s \Gamma \rangle \\ &\quad - f(\Gamma) \langle \partial_t \Gamma, \partial_{ss} \Gamma \rangle \big)\, ds. \end{aligned} \tag{4.45}$$

The second summand in (4.45) vanishes since the velocity field of Γ can be assumed to be orthogonal to the curve, i.e. $\langle \partial_t \Gamma, \partial_s \Gamma \rangle = 0$. In fact, it is well-known that the evolution of a curve is governed only by its normal velocity, the tangential velocity can be neglected (see [EG87]). As the velocity field $\partial_t \Gamma$ is tangential to M, the claimed equation (4.41) follows directly from (4.45).

Further, taking the tangential component of the curvature vector yields $(\partial_{ss}\Gamma)^{\|} = \kappa_g \hat{\nu}$ (see (2.7)). The component of the gradient term tangential to Γ can be neglected, and one concludes (4.42). □

4.A.2 Area Minimisation

Balloon forces, pushing the contour with a constant velocity, were first introduced in [Coh91]. Subsequent works on geometric active contour models [CCCD93, CKS97, KKO+96] adopted the concept of a balloon term. In [CKS97] the following evolution equation was proposed:

$$\partial_t \Gamma = \big(f \cdot (\kappa + c) - \langle \nabla f, \nu \rangle\big)\, \nu \,. \tag{4.46}$$

Siddiqui et al. [SLTZ98] formalised the concept and introduced weighted area terms (see (3.12)). In the generalised model on surfaces however (see Section 4.4) only constant balloon forces are used.

With the following proposition the concept of a balloon force on surfaces is mathematically justified. It is a generalisation of the respective result in 2D (see [SLTZ98]), which has been reviewed as Proposition 7 in Chapter 2.

Proposition 12. *Let $M \subset \mathbb{R}^3$ be smooth hypersurface, Γ a closed smooth curve on M and $\Sigma \subset M$ the part of M that is enclosed by Γ. Then the L^2 gradient flow of the enclosed area functional*

$$A(\Gamma) = \int_\Sigma dS$$

on the surface M is given by the evolution equation

$$\partial_t \Gamma = \hat{\nu} \tag{4.47}$$

Remark 1. *The proof is not as straightforward as in the plane case: in $\mathbb{R}^2$ the area enclosed by a curve can easily be transformed to a line integral over the curve. In fact, Green's theorem yields*

$$A(\Omega) = \int_\Omega dV = \frac{1}{2} \int_\Omega \nabla \cdot \mathbf{x}\, dV = \frac{1}{2} \int_{\partial\Omega} \langle \mathbf{x}, \nu \rangle\, ds \tag{4.48}$$

with Ω the domain enclosed by $\Gamma = \partial\Omega$, the identity vector field $\mathbf{x}$ and the curve normal ν. However, this procedure is not possible on curved surfaces since partial integration in this generalised setting usually yields a new area integral including a mean curvature term (see e.g. [GM75, (2.11)]).

Remark 2. *In [CBMO02] the validity of Proposition 12 is assumed, but no proof is given.*

Proof. The techniques in the proof restrict its validity to the surfaces that can be described implicitly. These are virtually all surfaces, except pathological surfaces of mainly theoretical

interest such as self-intersecting surfaces. Due to the coarea formula (see e.g. [EG92]), the surface element of an implicit surface can be written as follows:

$$dS = \delta(\psi)|\nabla\psi|\,dV\ , \tag{4.49}$$

with δ the delta function. For simplicity, smoothed out versions of the delta- and the Heaviside function (cf. [OF03]) are denoted by δ and H below. By means of the coarea formula, the enclosed surface area can be expressed as a volume integral in terms of the level set functions:

$$A(\Gamma) = \int_{\mathbb{R}^3} (1 - H(\phi))\delta(\psi)|\nabla\psi|\,dV\ . \tag{4.50}$$

Now, the velocity field V is calculated which, applied to the curve Γ, minimises the functional A as fast as possible. Remember, that due to (4.20) evolution under an external vector field obeys the equation

$$\partial_t\phi = -\langle P_{\nabla\psi}V, \nabla\phi\rangle = -\langle V, P_{\nabla\psi}\nabla\phi\rangle\ . \tag{4.51}$$

Computing the time derivative of (4.50) yields

$$\begin{aligned}
\partial_t A(\Gamma) &= \int_{\mathbb{R}^3} -\delta(\phi)\,\partial_t\phi\ \delta(\psi)\,|\nabla\psi|\,dV && (4.52)\\
&\overset{(4.51)}{=} \int_{\mathbb{R}^3} \langle V, \delta(\phi)P_{\nabla\psi}\nabla\phi\rangle\ \delta(\psi)\,|\nabla\psi|\,dV && (4.53)\\
&= \int_M \langle V, \delta(\phi)P_{\nabla\psi}\nabla\phi\rangle\,dS. && (4.54)
\end{aligned}$$

Consequently, the negative gradient velocity field is $V = -\delta(\phi)P_{\nabla\psi}\nabla\phi = \delta(\phi)\hat{v}$ (cf. Fig. 4.4). Since $\delta(\phi)$ is constant on Γ, this term can be omitted, and Eq. (4.47) follows. □

Consequently, the incorporation of a balloon term in a GAC model on surfaces is justified from the variational point of view. This justifies the proposal of the following equation generalising (4.1):

$$\partial_t\Gamma = \Big(f(\kappa_g + c) - \langle\nabla_M f, \hat{v}\rangle\Big)\,\hat{v}\ . \tag{4.55}$$

The level set formulation of this equation is given in Section 4.4.3. Note that the balloon term has a different scale than the other two terms, since it stems from an area- and not from a line integral. As shown in Section 4.3, this causes problems when working with parametric surface representations.

4.B Issues of Parametric Surfaces

4.B.1 Scaling Behaviour/Proofs of Propositions

Proof of Proposition 8 One derives from (4.5) that $|\nabla_M I|^p = k^{-\frac{p}{2}}|\nabla I|^p$ in the k-metric. It follows

$$\frac{f^k}{f^1} = \frac{1+|\nabla I|^p}{1+k^{-\frac{p}{2}}|\nabla I|^p} = (1+\Delta_1)\frac{|\nabla I|^p}{k^{-\frac{p}{2}}|\nabla I|^p} = (1+\Delta_1)k^{\frac{p}{2}}\,. \tag{4.56}$$

Substituting $x = |\nabla I|^p$ and $r = k^{-\frac{p}{2}}$ yields

$$\begin{aligned} 1+\Delta_1 &= \frac{1+x}{x}\cdot\frac{rx}{1+rx} && (4.57)\\ &= \left(1+\frac{1}{x}\right)\cdot\left(1-\frac{1}{1+rx}\right)\,. && (4.58)\end{aligned}$$

Consequently, Δ_1 is given by

$$\begin{aligned} \Delta_1 &= \frac{1}{x}-\frac{1}{1+rx}-\frac{1}{x(1+rx)} && (4.59)\\ &= \frac{rx-x}{x(1+rx)} = \frac{r-1}{1+rx}\,. && (4.60)\end{aligned}$$

This concludes the proof of (4.8). In order to show (4.9), the gradient of f^k is computed as

$$\nabla f^k = \frac{-k^{-\frac{p}{2}}p\,|\nabla I|^{p-2}}{\left(1+k^{-\frac{p}{2}}|\nabla I|^p\right)^2}D^2I\cdot\nabla I\,, \tag{4.61}$$

where D^2I is the Hessian of I. As ∇f^k and ∇f^1 are collinear, the quotient can formally be computed:

$$\frac{\nabla f^k}{\nabla f^1} = k^{-\frac{p}{2}}\frac{(1+|\nabla I|^p)^2}{\left(1+k^{-\frac{p}{2}}|\nabla I|^p\right)^2} = (1+\Delta_2)k^{\frac{p}{2}}\,. \tag{4.62}$$

Due to $(1+\Delta_2) = (1+\Delta_1)^2$ one concludes (4.11). □

Proof of Proposition 9 Equation (4.4) in the k-metric can be written as

$$\partial_t\phi = f^k\kappa_g^k + \nabla f^k\cdot(G^k)^{-1}\nabla\phi\,, \tag{4.63}$$

where $(G^k)^{-1}$ denotes the inverse metric tensor and κ_g^k the geodesic curvature in the k-metric. Note, that $[g^{ij}] = \frac{1}{k}[\delta_{ij}]$ respectively $(G^k)^{-1} = \frac{1}{k}I$ with I the unit matrix. Using $\kappa_g^k = \frac{1}{k}\kappa_g^1$ and $(G^k)^{-1} = \frac{1}{k}(G^1)^{-1}$ and applying Proposition 8 in (4.63) yields

$$\begin{aligned} \partial_t\phi &= (1+\Delta_1)k^{\frac{p}{2}}f^1\frac{1}{k}\kappa_g^1 + (1+\Delta_2)k^{\frac{p}{2}}\nabla f^1\cdot\frac{1}{k}(G^1)^{-1}\nabla\phi && (4.64)\\ &= (1+\Delta_1)k^{\frac{p-2}{2}}[f^1\kappa_g^1 + (1+\Delta_1)\nabla f^1\cdot(G^1)^{-1}\nabla\phi]\,. && (4.65)\end{aligned}$$

Analogously, the weighted geodesic normal flow term is computed in the k-metric:

$$\partial_t \phi = -f^k |\nabla_M^k \phi| = -(1+\Delta_1) k^{\frac{p-1}{2}} f^1 |\nabla \phi| \,. \tag{4.66}$$

□

4.B.2 Problems with Surface Parametrisations

Computing parametrisations for mesh surfaces is a nontrivial task, and therefore, a large body of literature is dedicated to this topic. Refer to the recent survey papers [SPR06, FH05] for an introduction and further references.

As long as a surface is homeomorphic to a plane, the parametrisation process is relatively easy, and major distortions in the mapping can be avoided. For complex surfaces, e.g. closed surfaces and those of higher genus, the following issues are common:

1. The surface has to be cut open along seams. This procedure causes discontinuities and singularities in the parametrisation;

2. The surface has to be segmented into patches prior to parametrisation. This results in several local- instead of one global parametrisation;

3. Irregular (i.e. non-rectangular) parametrisation domains occur.

Issues 1) and 2) make the application of Spira's framework [SK07] intractable, because the numerics become unstable in the presence of singularities, and a smooth global parameterisation is required. Further, it is not obvious how to handle issue 3).

Consider the sphere as an easy example. For the standard spherical coordinates

$$X_r(\theta, \varphi) = (r \sin\theta \cos\varphi, r \sin\theta \sin\varphi, \cos\theta)^\top \,, \tag{4.67}$$

with $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$, the surface metric is given by

$$\left[g_{ij}\right] = \begin{pmatrix} r^2 & 0 \\ 0 & r^2 \sin^2\theta \end{pmatrix} \,. \tag{4.68}$$

Clearly, the metric degenerates towards the poles, i.e. when θ approaches 0 or π. This makes a stable numerical analysis on a full parametric sphere impossible. Note, that all experiments in [SK07] were carried out on a sphere with cut poles to enable stable numerical computations. Our experiments showed that even with the poles cut the numerics on the sphere

remain delicate. The choice of the time step has to very small in order to avoid instabilities, thus rendering the computations inefficient.

Altogether, the parametric framework [SK07] is not suitable for computing geometric curve evolution on complex surfaces.

4.C Detailed Description of the Narrow Band Algorithm

Let $0 < \beta < \gamma$ be two constants which determine the radii of the inner and the outer narrow band. In the experiments shown in this thesis (with unit grid size $\Delta x = 1$ and fifth-order WENO schemes), good results were obtained with $\beta = 2$ and $\gamma = 4$ (see Fig. 4.6 for an example). Only intricate surfaces with a comparably low grid resolution – such as the Stanford bunny in our experiment in Section 4.6.3 – required a larger narrow band with $\beta = 3$ and $\gamma = 6$, as suggested in [PMZ+99] for the use of fifth order WENO schemes. This corresponds to the width of the stencil of this finite difference scheme.

The proposed algorithm for the evolution of GAC on implicit surfaces consists of the following steps:

Step 0 *Initialise distance functions $\tilde{\psi}$ and $\tilde{\phi}$.*

Since M is static, this has to be done only once for $\tilde{\psi}$. However, the curve evolves, and therefore $\tilde{\phi}$ has to be recomputed regularly (see Step 3).

Step 1 *Compute the narrow band $T = \{x \in \Theta : \theta_p(x) \leq \gamma\}$ and the extended narrow band $N = \{x \in \Theta : \exists e \in E : \theta_p(x + \vec{e}) \leq \gamma\}$ with E the canonical basis of $\mathbb{R}^3$.*

θ_p only depends on $\tilde{\psi}$ and $\tilde{\phi}$ (see (4.33) and (4.32)). The narrow band has to be updated only if ϕ was reinitialised before and thus $\tilde{\phi}$ has changed (see Step 3).

Step 2 *Update level set function ϕ by solving the modified equation (4.70) in T.*

In order to avoid boundary stability issues, the equation is localised by multiplying with a cut-off function λ. λ equals one in the inner narrow band ($\theta_p \leq \beta$) and smoothly decreases to zero with $\theta_p \to \gamma$. A particular choice of λ was proposed in [PMZ+99] (see Fig. 4.17):

$$\lambda(\theta_p) := \begin{cases} 1, & \text{if} \quad \theta_p \leq \beta \\ \frac{(\theta_p - \gamma)^2 (2\theta_p + \gamma - 3\beta)}{(\gamma - \beta)^3}, & \text{if} \quad \beta < \theta_p \leq \gamma \\ 0, & \text{if} \quad \theta_p > \gamma. \end{cases} \tag{4.69}$$

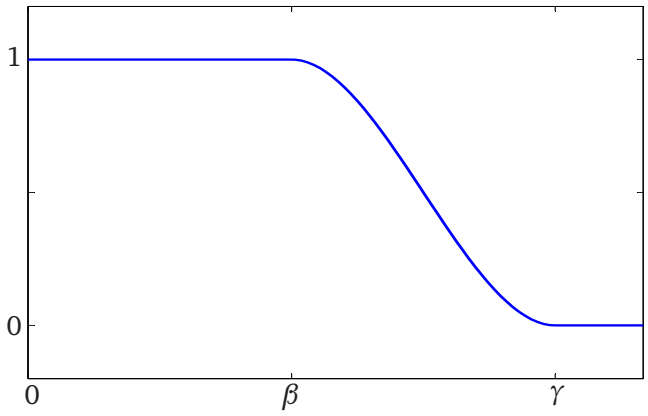


Figure 4.17: The cut-off function λ.

The modified equation then reads

$$\begin{aligned} \partial_t \phi &= \lambda(\theta_p)\Big(cf\,|P_{\nabla\psi}\nabla\phi| + \langle P_{\nabla\psi}\nabla f, \nabla\phi\rangle \\ &\quad + \mu f\,\kappa_g |P_{\nabla\psi}\nabla\phi|\Big). \end{aligned} \tag{4.70}$$

Step 3 *Reinitialise if necessary, then return to Step 1.*

The level set function ϕ is reinitialised if one of three criteria is met:

1. The evolving curve reaches the tube $\{x \in \Theta : \theta_p(x) = 1\}$ of radius 1 (this ensures the updated narrow band T does not outreach the current extended band N);
2. $|P_{\nabla\psi}\nabla\phi|$ differs from 1 significantly;
3. The maximum number of steps is reached.

Reinitialisation is achieved by iterating a few steps of (4.30) and (4.31) on the extended narrow band N. Both equations are hyperbolic, with characteristics spreading from the surfaces $\phi^{-1}(0)$ and $M = \psi^{-1}(0)$, respectively, in normal direction. No boundary conditions are needed. The necessary number of iterations is estimated by

$$n_{\text{iter}} = \left\lfloor \frac{\gamma + 1}{\alpha} \right\rfloor + 1\,, \tag{4.71}$$

where α denotes the CFL-number ($0 < \alpha < 1$) applied to compute the time step (see (2.36)). Thus, the information, which flows with velocity α, certainly reaches the boundary of N.

Afterwards, $\tilde{\phi} = \phi(.,t_0)$ is retained and the loop starts over at step 1.

Chapter 5

Segmentation of Face Feature Contours From Range Images

5.1 Introduction

The framework introduced in Chapter 4 enables the segmentation of a general class of surfaces, those that can be represented implicitly, while keeping up all the well-known attributes of the classic geodesic active contours. Yet in many applications, including 3D face recognition and analysis, the surface data is available in a more simple form, parametrised as the *graph of a function* Z over a domain Ω:

$$X(u_1, u_2) = \big(u_1, u_2, Z(u_1, u_2)\big)^\top \qquad (u_1, u_2) \in \Omega\,. \tag{5.1}$$

In differential geometry, such surfaces are called *Monge patches* (see e.g. [Gra97]). In the applied sciences, including Computer Vision, Monge patches usually occur on rectangular domains and are referred to as *range images* or *depth maps* (e.g. [SS01]). Range images are common in 3D face data processing due to their convenient form and little memory requirements. Furthermore, the significant part of the face, including the eyes, the nose and the mouth, can be well described by a range image (Fig. 5.1). This is, however, not possible for full 3D head models that include the ears.

The rest of the chapter is organised as follows: the state of the art in the field is summarised in Section 5.2. Section 5.3 reviews a simple algorithm for converting triangulated surfaces into range images. Section 5.4 explains how curvature images are derived from depth maps. The following two sections – 5.5 and 5.6 – discuss face feature contour segmen-

Figure 5.1: Dataset from the BJUT 3D face database [bju05]: original triangulated face surface (left), after conversion to a range map (middle), respective range image (right).

tation from range image data using efficient 2D techniques. More specifically, an algorithm for lip contour segmentation based on the globally optimal AC technique by Cohen and Kimmel [CK97] is proposed in Section 5.5. Subsequently, Section 5.6 illustrates that due to artefacts and noise in the respective region of common 3D scans, eye contour segmentation from 3D data appears a complex problem. This makes well-known techniques unsuitable for automatic segmentation of the eye contours. As a consequence, the need for strong regularity and shape priors demonstrated in this section leads to the study of second-order ACs in Chapter 6. Some preliminary conclusions are drawn in Section 5.7.

5.2 State of the Art

Several approaches for segmenting feature points from 3D face data were proposed in the past. Yet compared to its 2D counterpart, 3D face analysis is a relatively young area of research. Gordon's paper [Gor92] represents one of the pioneering works in the field, where characteristics of both the mean and the Gaussian curvature were used to localise feature points, lines, and regions. Thereby, the focus was on the eyes and nose. Lu et al. [LCJ04] and Colbry et al. [CSJ05] employed a shape index – based on the principal curvatures of the face surface – to extract 3D feature points. To increase the robustness, a face model was incorporated in [LCJ04], and a more sophisticated statistical approach for inter-point distances in [CSJ05]. By including texture data Lu and Jain [LJ06] developed this approach further. The three methods [LCJ04, CSJ05, LJ06] share the ability to handle pose variations of the face, yet the reported results show that detected feature points usually differ significantly

from the manually defined ground truth. Conde et al. [CRAC06] used spin images to localise the nose tip and inner eye points in poor quality face scans. Recently, Nair and Cavallaro [NC09] proposed a 3D face processing system based on a point distribution model, similar to active appearance models [CET01]. The system, that includes a landmark detection step, requires extensive training with manually labeled datasets.

Multimodal approaches, combining 3D shape with 2D texture information, were suggested in [WCH02, BR05]. Wang et al. [WCH02] employed Gabor filters and point signatures to extract feature points from 2D and 3D data, respectively. Boehnen and Russ [BR05] based their system for feature extraction on the texture image. The nose location, and the refinement of mouth- and eye location, are provided by the range data.

To the best of the author's knowledge, no work on the extraction of feature contours from 3D face data was published so far. Colbry et al. [CSJ05] attempted to segment the mouth line, yet their algorithm delivers only a rough approximation.

5.3 Converting Triangular Meshes into Range Maps

In order to obtain range image representations of triangulated surfaces, the *gridtrimesh* algorithm [Bri07] is employed in the thesis. It is succinctly reviewed below.

For simplicity, consider a single triangle T in $\mathbb{R}^3$ given by the three vertices v_1, v_2, v_3 (see Fig. 5.2). Then, T is mapped onto the x/y plane by the projection P, and the bounding box B is computed. Let the smallest and the largest x-grid value within the bounding box be denoted by w and e, respectively, and, analogously, the smallest and the largest y-grid value within the bounding box be denoted by s and n. Then, all grid points within the rectangle $[s,n] \times [w,e]$ are tested on whether they lie inside the current triangle. This is achieved by computing their respective barycentric coordinates with respect to the projected triangle. Define

$$A = \begin{pmatrix} P(v_1) & P(v_2) & P(v_3) \\ 1 & 1 & 1 \end{pmatrix} . \tag{5.2}$$

Note, that A is quadratic, since the $P(v_i)$ are vectors in $\mathbb{R}^2$. The 3D barycentric coordinate vector w of a grid point p is computed as

$$w = A^{-1} \cdot p \, . \tag{5.3}$$

Here, the x/y plane is identified with the plane in $\mathbb{R}^3$ given by $z = 1$. Then, the grid point p lies within $P(T)$, if and only if $0 < w_i < 1$ for all components of w. The interpolated z-value $Z(p)$ is

$$Z(p) := \begin{pmatrix} P_z(v_1) & P_z(v_2) & P_z(v_3) \end{pmatrix} \cdot w \, , \tag{5.4}$$

where $P_z : \mathbb{R}^3 \to \mathbb{R}$ is the projection onto the z-coordinate. If a grid point lies within more than one projected triangle, which may occur, for instance, for closed surfaces, the largest potential z-value is chosen. This procedure assumes that the observer looks at the surface from a point $(0,0,z_0)^\top \in \mathbb{R}^3$ with z_0 large and positive. Refer to [Bri07] for more details. Note, that texture values can be interpolated in the same manner as the z-values in (5.4).

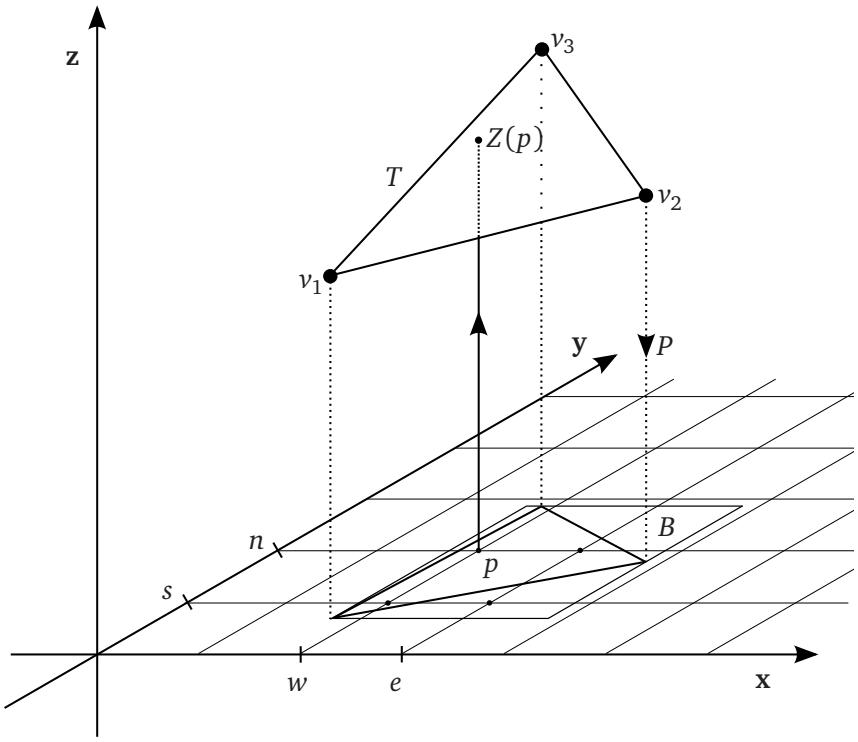


Figure 5.2: Projection of a 3D triangle onto the x/y plane. An interpolated Z value is computed for grid point p within the projected triangle.

5.4 Curvature Images

The most important geometric invariant of a surface is its curvature. The common curvature measures for classic 2D surfaces in $\mathbb{R}^3$ are the *Gaussian* and the *mean* curvature (see e.g. [dC76]).

Let M be a Monge patch given by a parametrisation as in (5.1). Then the Gaussian curvature K is

$$K = \frac{\partial_{u_1u_1} Z \cdot \partial_{u_2u_2} Z - (\partial_{u_1u_2} Z)^2}{\left(1 + |\nabla Z|^2\right)^2} , \tag{5.5}$$

and the mean curvature H is

$$H = \frac{(1 + (\partial_{u_2} Z)^2)\partial_{u_1u_1} Z - 2\partial_{u_1} Z \cdot \partial_{u_2} Z \cdot \partial_{u_1u_2} Z + (1 + (\partial_{u_1} Z)^2)\partial_{u_2u_2} Z}{\left(1 + |\nabla Z|^2\right)^{3/2}} . \tag{5.6}$$

See e.g. [Gra97] for both formulae. Figure 5.3 displays the respective curvature images for the range image in Fig. 5.1. This example suggests that the mean-, rather than the Gaussian curvature is suitable to describe feature contours of the mouth and the eyes, while the Gaussian curvature enables the localisation of certain feature points such as the mouth corners.

5.4.1 Laplacian of the curvature

When the surface data is of sufficient quality, utilising further derivatives of the curvature can be beneficial for the analyses. Consider, for instance, the image in Fig. 5.4(b) which displays the Laplacian of the mean curvature (Fig. 5.4(a)), i.e.

$$\Delta H := \nabla \cdot \nabla H = \partial_{u_1u_1} H + \partial_{u_2u_2} H . \tag{5.7}$$

Eye contour segmentation experiments suggested that Laplacian images often yield better results than curvature images. The explanation is that gaps in the ridges are not as distinct in Laplacian images (see Section 5.6). However, if the data is too noisy, it is not advisable to compute the Laplacian, since fourth-order derivatives of Z are involved (cf. Fig. 5.14(c), p.96).

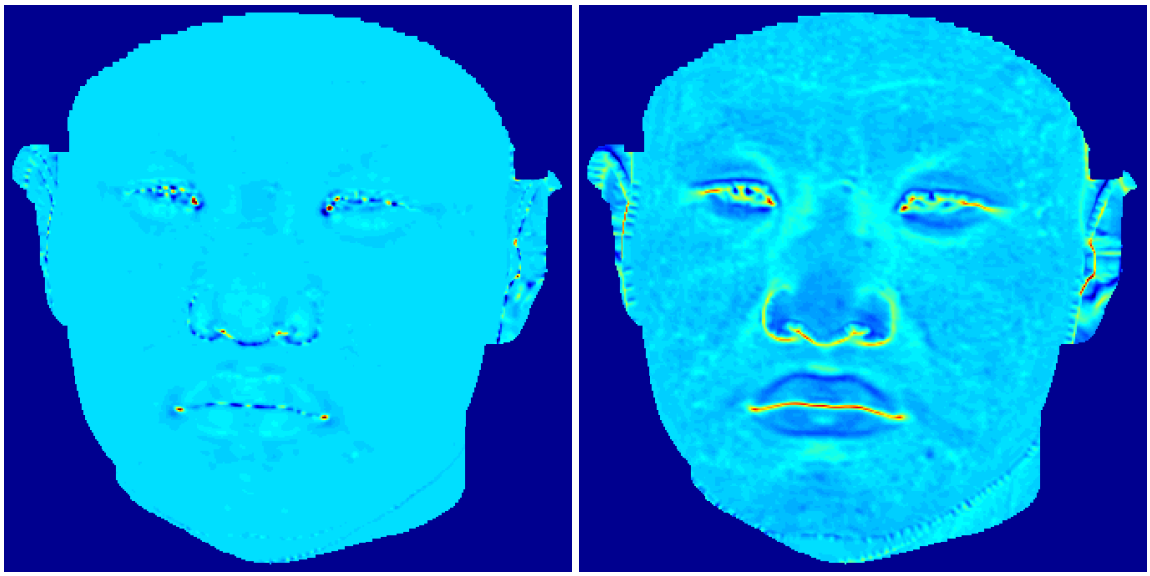
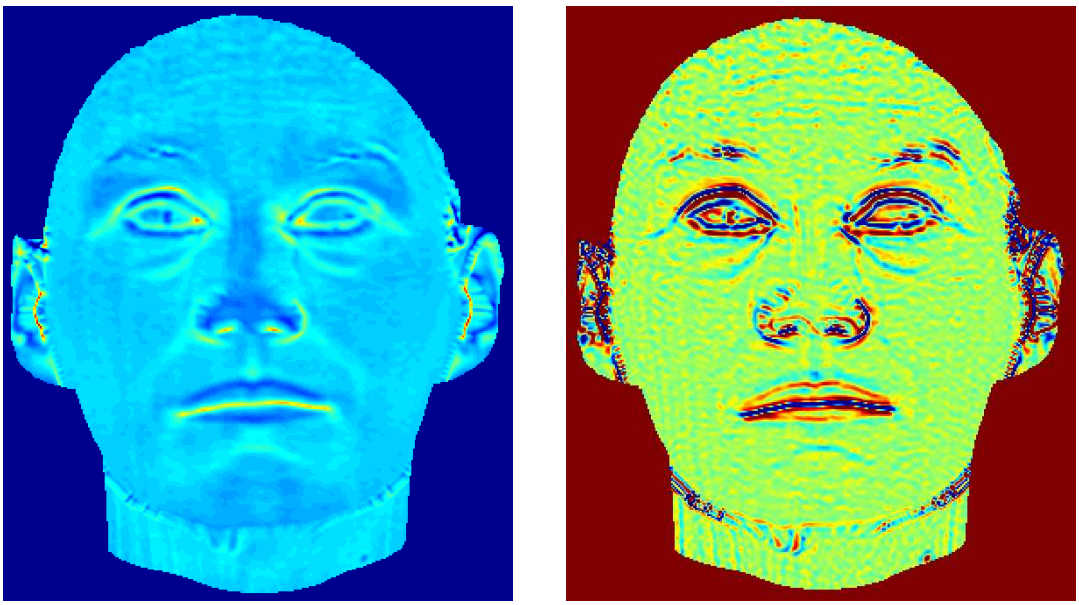


Figure 5.3: Curvature images computed from the range image in Fig. 5.1: the Gaussian (left) and mean curvature (right). Yellow/red colours mean positive values, dark blue colour means negative values.



(a) (b)

Figure 5.4: Mean curvature image (a) and derived Laplacian image (b). Note that outliers of the Laplacian were cut at the 2% and 98% percentiles.

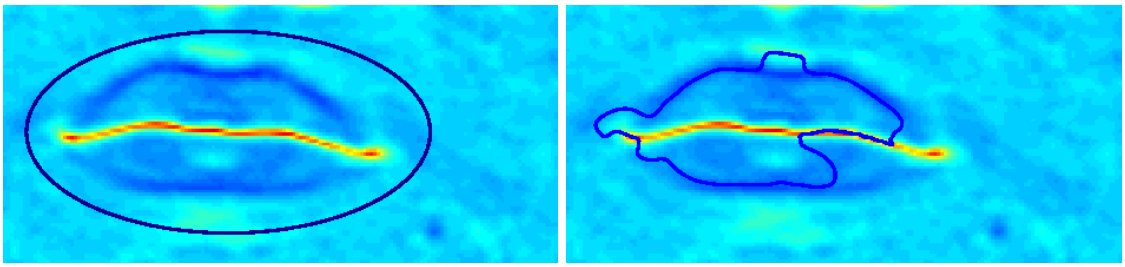


Figure 5.5: GAC with stopping term (5.8), $p = 1$: the initial contour (left) and the contour after passing the mouth corners (right).

5.5 Lip contours

Looking closer at the mouth region of the mean curvature image in Fig. 5.3, ridges delineating the lips from the surrounding areas can be observed. Yet these ridges tend to weaken close to the mouth corners. Thus, the lip segmentation cannot be achieved by a simple computation of ridge lines. The following section applies well-known AC schemes to design new algorithms for 3D lip contour segmentation.

5.5.1 Using closed contours

Initially, it is researched how the upper and lower lip contours can be segmented from range images by means of closed ACs. This technique was published in [KDG08b], using the 3D GACs introduced in Chapter 4. Nevertheless, it can be transferred straightforward to curvature images using standard 2D GACs [CKS97].

A naive approach is to define the stopping function as

$$f = \frac{1}{1+|H|^p}, \quad p \in \{1,2\}\,, \tag{5.8}$$

with H being the surface mean curvature. This stopping term would attract the contour to all areas of high absolute (positive or negative) curvature. Alternatively, the Gaussian curvature K could be used. While this may help extract ridges from curved surfaces, it is not suitable for 3D lip segmentation. Upon reaching the mouth corners, the contour usually continues to evolve (as seen in Fig. 5.5), since the region of the mouth corners displays only isolated points of high curvature.

Instead, as can be observed in Fig. 5.3, the separation between upper and lower lips has positive mean curvature, while the ridges encompassing the lips region have negative mean curvature. Therefore, a suitable stopping term f, that attracts the evolving curve to the ridges

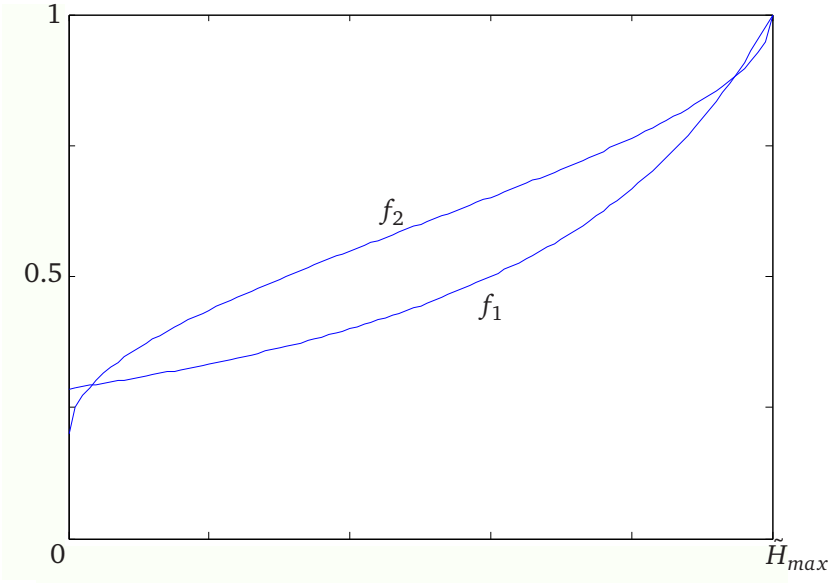


Figure 5.6: Two stopping functions f_1 and f_2.

and repels it in presence of positive mean curvature regions, needs to be designed. The next logical step is to consider the curvature sign in the stopping term:

$$f_1 = \frac{1}{1 + (\widetilde{H}_{max} - \widetilde{H})}, \tag{5.9}$$

where $\widetilde{H}$ is a rescaled version of H with values between 0 and $\widetilde{H}_{max}$[1]. Both, the negative and the positive sign ranges are separately rescaled so that

$$\widetilde{H}(x) \in \begin{cases} [0, \frac{\widetilde{H}_{max}}{2}) & \text{if} \quad H(x) < 0 \\ [\frac{\widetilde{H}_{max}}{2}, \widetilde{H}_{max}] & \text{if} \quad H(x) \geq 0 \end{cases}. \tag{5.10}$$

The graph of the function f_1 is shown in Fig. 5.6. Note, that it increases in the direction of positive mean curvature values. Hence, the advection term in the GAC equation pushes the contour into regions of negative curvature. Optimally, the contour converges to the upper and lower ridges, while assuming an equilibrium of the advection-, and the other two terms near the mouth corners. The experiments showed that the stopping function f_1 has an undesirable effect: since f_1 has a steep negative gradient for positive curvature values (see Fig. 5.6), the contour is repelled from the mouth corners too strongly (see Fig. 5.7). To counter this, a further stopping function f_2 with a more uniform negative slope (see Fig. 5.6) is proposed:

$$f_2 = \frac{0.8}{\pi} \cdot \arccos\left(-\frac{\widetilde{H} - \widetilde{H}_{max}/2}{\widetilde{H}_{max}/2}\right) + 0.2. \tag{5.11}$$

[1] $\widetilde{H}_{max}$ is set to 2.5 across all experiments reported in the thesis.

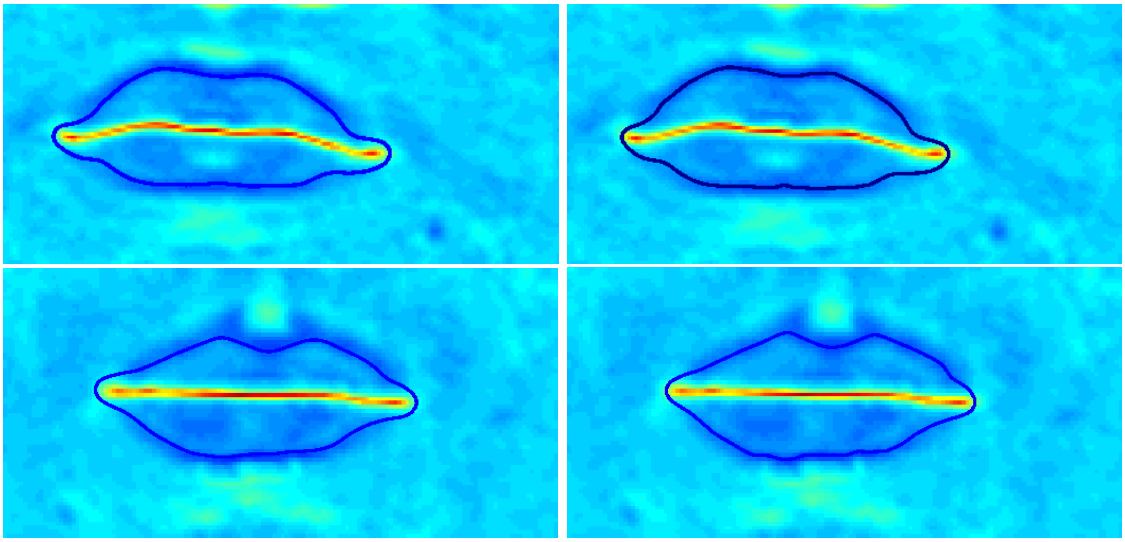


Figure 5.7: Comparison of stopping terms f_1 (left) and f_2 (right): for f_1 the GAC converges after 7200 respectively 6400 iterations, while it needs only 4600 respectively 4400 iterations using f_2. Further, the f_2 term converges closer to the mouth corners.

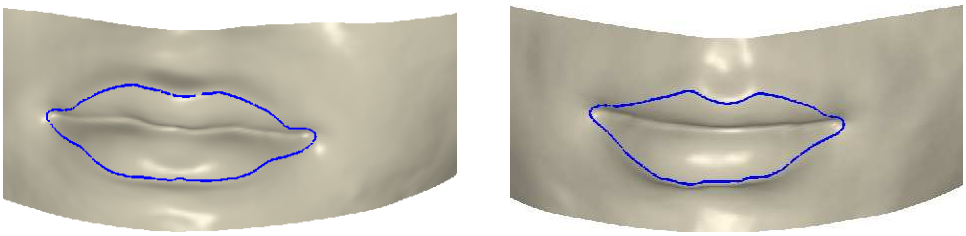
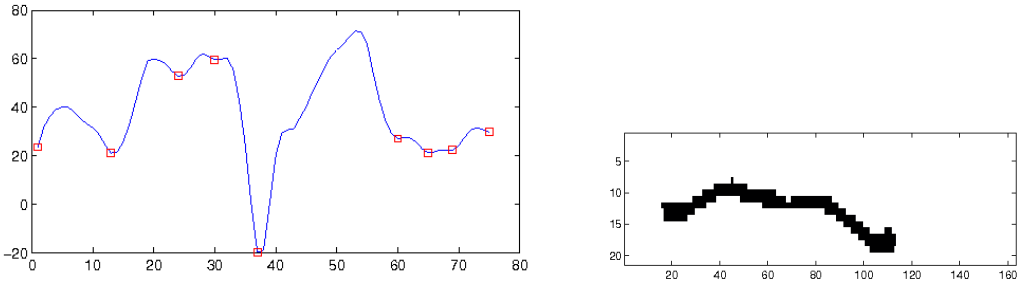


Figure 5.8: Sample results for lip contour segmentation from range images using GAC (stopping term: f_2).

Note, that the convergence of the GAC is accelerated due to the more uniform slope of f_2. The steeper gradient descent results in less necessary iterations. The 3D lip segmentation results for two sample datasets are shown in Fig. 5.8.



(a) (b)

Figure 5.9: Average mouth region curvature of (a), and the window around the mouth line after thresholding (b).

5.5.2 Using open contours

In the following, Cohen and Kimmel's method ([CK97], see also Sections 3.1.4 and 2.3), for computing globally optimal ACs with fixed endpoints is applied. Prerequisite for this approach is the prior detection of the mouth corners.

It has to be pointed out that in the literature sophisticated methods for mouth corner detection are described, e.g. incorporating statistics for inter-point distances (cf. [LJ06]). By contrast, the technique below is based on simple heuristics and is included here for completeness.

Mouth corner detection

The first step roughly localises the y coordinate of the mouth line. The curvature image, restricted to the mouth region in a preprocessing step, is analysed. After cutting off the curvature spectrum, using empirically chosen thresholds (1% and 99%), the average curvature for every y-value with respect to the x-axis (Fig. 5.9(a)) is computed. To determine the mouth line, the lowest local minimum which is in the middle 50% of the image is searched. Then a smaller window centered around the mouth line is considered, and the curvature value is thresholded (Fig. 5.9(b)). The mouth corners are then defined as the left- and rightmost pixels of the mouth line, that are vertically centred.

Segmentation of the mouth line

Once the mouth corners are fixed, the segmentation of the mouth line is straightforward using a specifically designed ridge detector function f:

$$f = \frac{1}{1 + |\max\{H, 0\}|}\,, \tag{5.12}$$

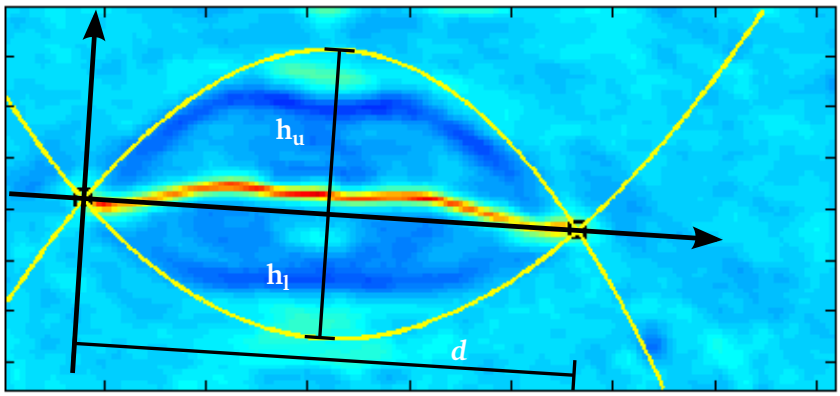


Figure 5.10: Defining parabolas in a rotated coordinate system.

where H is the mean curvature of the face surface. The function f takes small values close to 0 in areas of high positive mean curvature and values close to 1 elsewhere. Then the potential is defined as $P_m = f$, i.e. $w = 0$, and the minimising active contour is computed as described in Section 3.1.4.

Segmentation of the upper/lower lip contours

The segmentation of the upper and lower lip contours is more intricate, because the ridges are not as distinct as the mouth line. In addition, there are two lip contours and obviously only one minimal active contour. Hence, the AC algorithm has to be guided in order to segment both lip contours separately. Both issues suggest the aid of a shape prior in the segmentation process.

Parabolic shape priors The upper and lower lip contours roughly resemble parabolic shapes. Therefore, parabolas can be used to guide the AC algorithm towards the desired results. A coordinate system is rotated to align its x axis with the line running through the mouth corners (Fig. 5.10). Given the parameters d, h_u and h_l for the intersection with the x axis and the vertices, respectively, the equations for the upper and lower parabolas y_u and y_l are readily derived. The parameters h_u and h_l are empirically set to $h_u = d/3$ and $h_l = d/4$. Then, for each parabola an approximate distance transform $dist_{u|l}(x, y) = y - y_{u|l}(x)$ is computed.

Note, that the ridges are usually strong in the middle section, while they tend to weaken near the mouth corners. Hence, it is advisable to weight the ridge detector term high and the regularisation term low in the middle and vice versa towards the mouth corners. This was

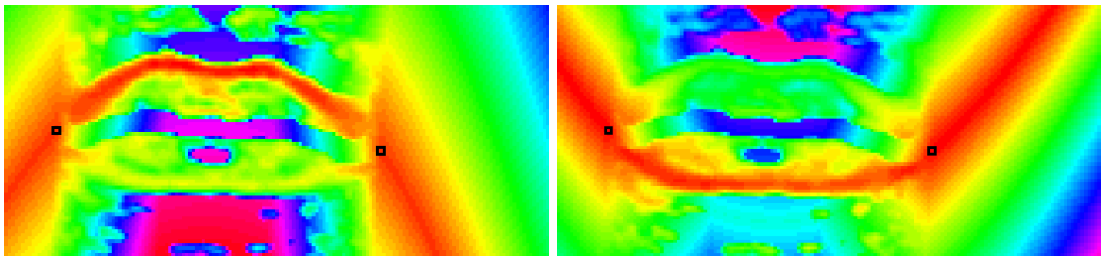


Figure 5.11: Potentials for the upper (left) and lower (right) lip contour. Red color means low values and green/blue color means larger values.

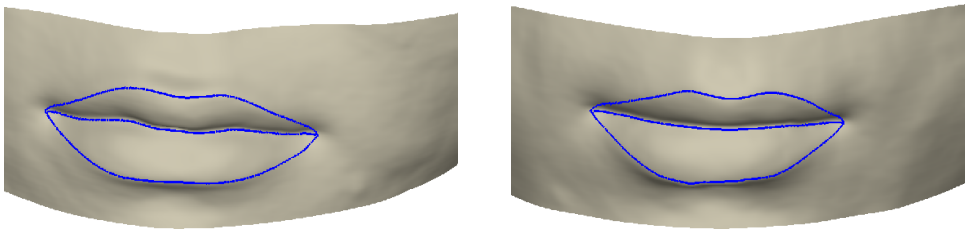


Figure 5.12: Sample results for lip contour segmentation from range images using globally optimal AC.

implemented on the rotated coordinate system by the following weight function: $w_{hor}(x,y) = \frac{-3}{d}|x - \frac{d}{2}| + \frac{3}{2}$. Afterwards, the function was cut off to take only values in $[0,1]$.

Defining the potentials P_u and P_l The potential function P_u and P_l for the upper and lower lip are now defined as:

$$P_{u|l} = \underbrace{\omega_{reg}(1 - w_{hor})}_{I} + \underbrace{\frac{w_{hor}}{a + b|\min\{H,0\}|}}_{II} + \underbrace{\omega_{prior} \cdot dist_{u|l}}_{III}, \tag{5.13}$$

where I is the regularity term, II denotes the ridge detector term, and III imposes a shape prior. The coefficients ω_{reg} and ω_{prior} control the weighting of the respective summands, while a and b are the parameters of the ridge detector function. The parameters for P_u were empirically chosen as $\omega_{reg} = \frac{1}{2}, \omega_{prior} = 0.01, a = \frac{1}{2}, b = 10$. For the potential of the lower lip the prior term was weighted slightly higher with $\omega_{prior} = 0.025$.

The potentials are visualised in Fig. 5.11: note that unlike the standard curvature image the upper and the lower lip contours can be distinguished and that the shape prior does not interfere with the ridge characteristics.

Sample results of the above approach are shown in Fig. 5.12, a quantitative evaluation will follow in Chapter 7.

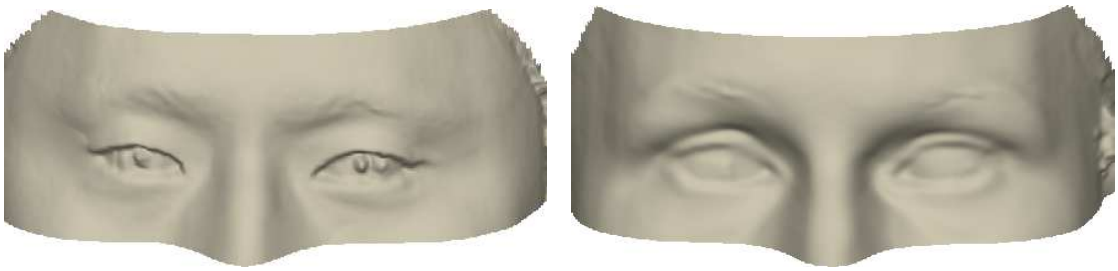
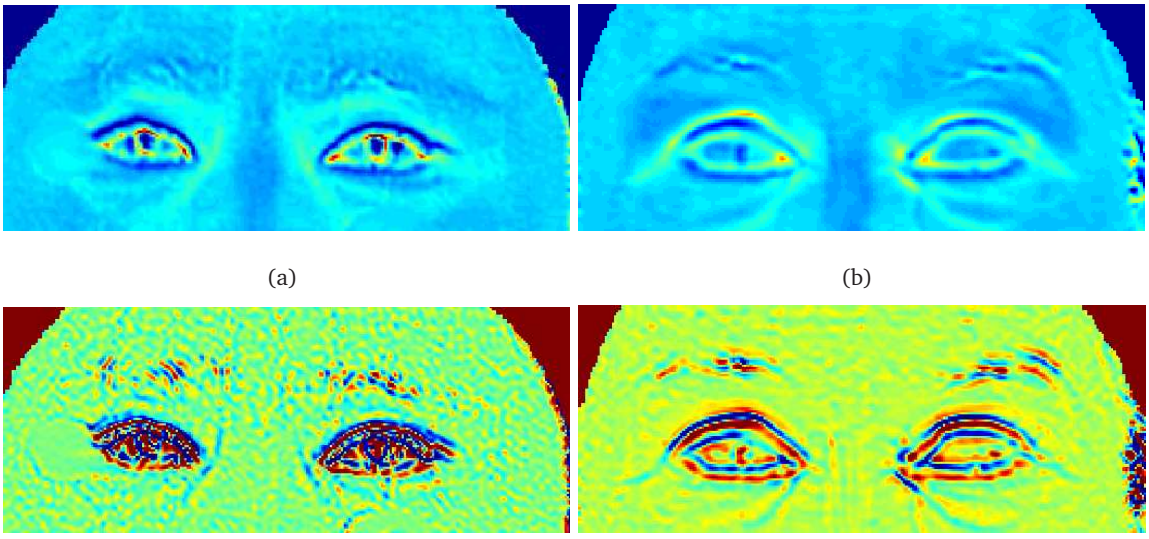


Figure 5.13: Eye region of sample 3D face surfaces.



(a) (b)

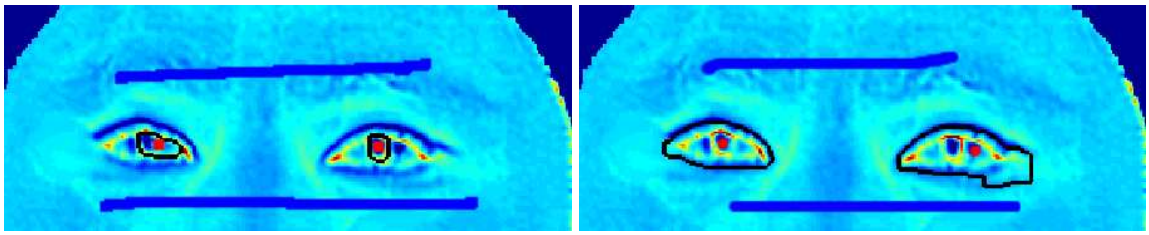
(c) (d)

Figure 5.14: Intrinsic geometric quantities for the surfaces in Fig. 5.13: mean curvature images (first row); the Laplacian of the mean curvature (second row). Darkblue colour means negative values, yellow/red colour means positive values.

5.6 Eye contours

This section elaborates on eye contour segmentation from 3D range images. It reveals that this problem cannot be solved efficiently by the well-known segmentation techniques.

Consider the surfaces in Fig. 5.13. In order to segment the eye contours from the surface, it is beneficial to analyse the intrinsic geometric quantities. Figures 5.14(a),(b) display the corresponding mean curvature images. In analogy to the lips (see Section 5.5), the outer eye contours can be described by ridges of high negative mean curvature. However, the ridges are less distinct and there are often large gaps around the eye corners.



(a) (b)

Figure 5.15: Attempts to segment the inner (a) and outer (b) eye contour using graph cut segmentation. While the obtained contours are too short in the first case, the problem in the latter case is that no stronger regularity can be enforced.

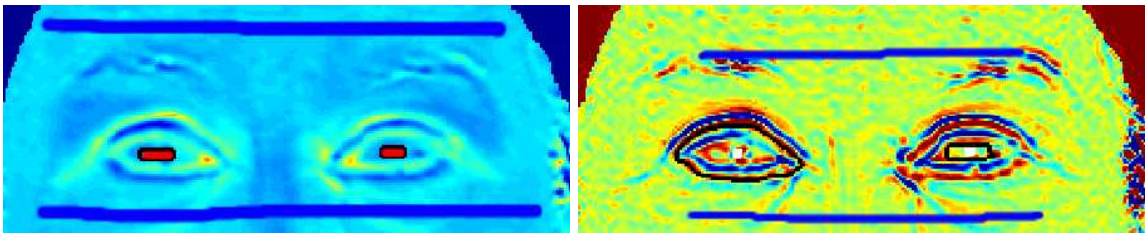
5.6.1 Results from known methods

Due to the round/elliptic form of the eyes, it appears natural to apply closed, rather than open contour segmentation methods. Furthermore, the heuristic determination of key or corner points – as in e.g. [LJ06, CSJ05, CRAC06] – can then be avoided.

Inner vs. outer eye contour Consider Figs. 5.14(a) and (b): in both images the outer eye ridges (dark blue) are quite distinct. While the inner eye contours (yellow/red) are clearly visible in Fig. 5.14(b), the inner eye in Fig. 5.14(a) possesses strong noise and artefacts, making the inner contour hardly distinguishable. This leads to poor segmentation results as in Fig. 5.15(a), obtained by the graph cut GACs [BK03]. The algorithm proposed in Chapter 6 can improve such outcomes to some extent, yet experiments showed that the outer eye ridges are less affected by artefacts and enable more reliable segmentation results.

Figure 5.15(b) illustrates the attempt to segment the outer eye contour using graph cut segmentation [BFL06]. Consider the right eye: since the ridge has a gap with strong positive curvature in the right corner, the graph cut contour makes a large loop around this heavily penalised area. Note, that while the popular graph cut algorithm is chosen for demonstration here, similar results can be expected for other globally optimal first-order methods, e.g. [AT05]. Enforcing strong regularity in such algorithms is delicate, as an increased arc length penalty term introduces a strong bias towards short curves, and might result in trivial contours around the seed (cf. Fig. 5.16(a)).

In simple cases, such as the face in Fig. 5.14(a), iterative methods like the classic GAC deliver the desired solution for the outer contour, yet long running times are prohibitive.



(a) (b)

Figure 5.16: Segmentation results for the outer eye contour obtained using graph cuts (the seeds are red in (a) and white in (b)). On the curvature image (a) trivial contours are obtained, on the Laplacian image (b) one eye contour is computed sucessfully.

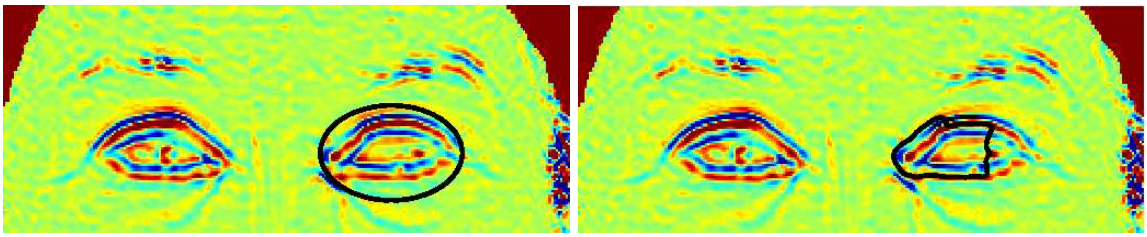


Figure 5.17: Failed attempts to segment the eye contour from a Laplacian image using the GAC: The initial contour (left) and the contour after 6000 iterations (right).

Consider now Fig. 5.14(b): this example is more challenging due to large gaps in the outer contour. Consequently, the graph cut algorithm delivers trivial contours, if the seed points are placed in the middle of the eye (Fig. 5.16(a)). For such rather clean data, the Laplacian of the mean curvature, shown in Fig. 5.14(d), can enable better results since the contours become more distinct. Yet the results obtained by the graph cuts technique remain unsatisfactory (Fig. 5.16(b)).

Standard 2D GACs do not succeed in the presence of large gaps either. Figure 5.17 clearly illustrates how the contour passes the corner with the gap and fails to converge.

5.6.2 Summary

The preceding section has illustrated that well-known techniques are not suitable for robust and efficient segmentation of eye contours from 3D face surfaces. Representatively, the standard GACs were studied for the class of deformable models and, respectively, the popular graph cut algorithm for the class of globally minimising first-order methods.

Main drawbacks of the GACs and similar methods:

- The contour has to be initialised, and the results may change with varying initial contours;
- The results are only locally optimal;
- A strong balloon force is necessary to guide the contour towards the desired outcome. As shown in Fig. 5.17, the contour might pass the object of interest;
- A large number of iterations is necessary, if the initial contour is placed at some distance from the desired location, making the approach computationally inefficient.

By comparison, the graph cut and other globally optimal first-order methods, such as [AT05], are very efficient. Yet the application to eye contour segmentation reveals the following limitations:

- Initialisation: a seed has to be placed inside the object of interest. It is error-prone to do this automatically, and heuristics are necessary;
- The obtained contours tend to be very short (Figs. 5.15(a) and 5.16(a),(b));
- The regularisation may be insufficient (Fig. 5.15(b), right eye).

The sample results in this section indicate that the segmentation of eye contours from 3D data exemplifies an application where second-order (i.e. curvature-), rather than mere first-order (i.e. length-) regularisation is desirable. Yet the existing approaches to automatic, globally optimal segmentation incorporating curvature regularity are quite inefficient. They feature complexities that are at least quadratic in the number of image pixels and involve large constants (see Section 6.2 for details).

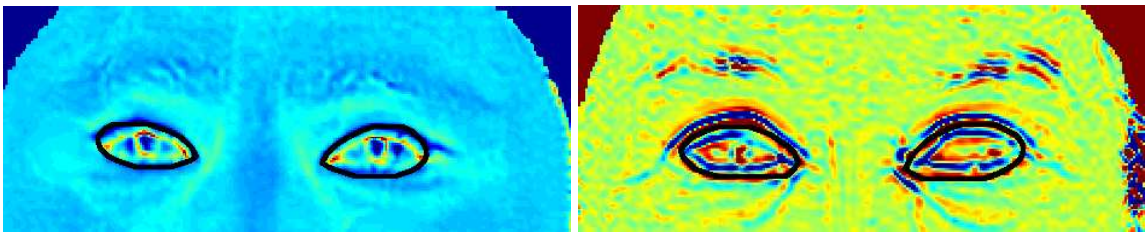


Figure 5.18: Segmentation results obtained automatically by the pseudo-elastica algorithm proposed in Chapter 6. Note, that the eye contours are computed separately. The computations take about 15-20 secs for each eye.

A second-order segmentation scheme is developed in Chapter 6 that can compute automatic segmentation results within a reasonable timeframe. In order to achieve acceptable efficiency, some heuristics are applied at the expense of exactness. Still, the algorithm approximates the global minimisers, i.e. it delivers pseudo-optimal contours. Its application to 3D face feature segmentation is evaluated in Chapter 7, sample results are displayed in Fig. 5.18.

The following section briefly states the outcomes of the chapter.

5.7 Conclusions

1. *Provided the surface data is given as a range image, efficient 2D techniques can be applied to 3D surface segmentation;*

2. *Algorithms for 3D lip segmentation can be designed from well-known AC schemes;*

3. *A globally optimising second-order segmentation algorithm is desirable for 3D eye contour segmentation. Yet existing approaches do not provide satisfactory efficiency.*

Chapter 6

Pseudo-Optimal Second-order Active Contours

In this chapter, a new segmentation method based on second-order energies is introduced. It has a significantly lower computational complexity – $O(N \log N)$ if user-input is provided – compared to the related works, such as [SC07]. The increased efficiency is achieved by integrating a novel second-order Dijkstra-type shortest path algorithm into a bidirectional search scheme. Some heuristics are incorporated into the algorithm at the cost of exact energy minimisation. In fact, the solutions found by the proposed algorithm are only approximately globally minimal. Due to its close relationship to the classic elastica ([Eul44], see e.g. [Mum94]), the method developed in this chapter is named *pseudo-elastica*. The pseudo-elastica core algorithm can be integrated into a completely automatic, or a user-guided segmentation scheme. The user-guided variant represents a generalisation of classic first-order path-based schemes to second-order energies while maintaining the same low complexity $O(N \log N)$. The results shown in Section 6.6 suggest that, compared to the above first-order approaches, it scores similar or better results and usually requires considerably less user-input. The automatic variant is primarily suitable for segmenting round or elliptic objects, that might possess large boundary gaps, even under strong noise. Anisotropy can be included in the algorithm in order to allow corners in the contour.

6.1 Introduction

Most energies used for edge based image segmentation are biased towards short curves. For instance, consider the prototypical energy functional for classic geodesic active contours [CKS97]:

$$E(\Gamma) = \int_{\Gamma} f(s)\, ds\,, \tag{6.1}$$

with the stopping term f. Since this function is positive and bounded on the compact image domain, the global minimum of the energy $\mathscr{E}$ is 0. Yet there is no useful global minimiser, because the energy converges to 0, if and only if a curve becomes infinitesimally short. As a consequence, local- rather than global minima are searched for, usually by gradient descent techniques, and the GAC requires initialisation close to the object of interest.

A few remedies for this problem have been suggested: the ratio energy introduced by Jermyn and Ishikawa [JI01] (see (6.3) below) is scale invariant and thus bias-free, at least in theory. In complex images, however, objects minimising ratio energies such as (6.3) are often very small, therefore the bias remains (cf. experiments in [SC07]) in practice.

Graph cut segmentation techniques (e.g. [BFL06, BK03]) have a similar bias towards short boundaries as the standard GAC. The minimised cut cost is the sum of the edge weights in a suitable graph and thus the discrete version of the first-order integral energy in (6.1). Meaningful globally optimal solutions can be enforced by adding hard constraints in the form of object- and background seeds. Yet results as in Figs. 5.15(a) and 5.16(a),(b) illustrate that the bias towards short curves remains intrinsic to the standard graph cut method.

Another approach to obtain contour energies without this bias is to replace length- by curvature regularity. Introduced into Computer Vision by Sha'ashua and Ullman in their pioneering work [SU88], curvature regularised curves were applied in several edge grouping algorithms [TW95, EZ96, MTW99, WKS05]. Only recently, Schoenemann and Cremers [SC07] proposed a generalisation of Jermyn and Ishikawa's [JI01] ratio energy including curvature. This image segmentation energy can be minimised globally and has a bias towards longer-, rather than shorter curves. The main drawback of this approach is its computational inefficiency. See Section 6.2 for more details and further related work.

This chapter is organised as follows: Section 6.2 reviews known segmentation approaches based on dynamic programming, as well as curvature-based techniques. Two new segmentation energies are introduced in Section 6.3. A proof is given that the energies possess

nontrivial global minima in suitable function spaces. Motivated by this theoretical outcome, algorithms are proposed in Section 6.4, that compute approximately globally minimal curves, optionally open or closed. These core algorithms are integrated into user-guided and fully automatic schemes for image segmentation in Section 6.5. Section 6.6 presents results on several images, including quantitative comparisons with the state-of-the-art techniques. The discussion and conclusions in Section 6.7 complete the chapter.

6.2 State of the Art

6.2.1 Optimal approaches (Shortest Paths/Dynamic Programming)

For an explanation of the basic idea of Dynamic Programming (DP) see Section 2.4.1.

Early applications of DP in image segmentation

Presumably, Montanari [Mon71] was the first to apply DP to image segmentation. He extracted curves of a fixed length from images, that are optimal with respect to the following *figure of merit* (FOM):

$$g(z_1,\ldots,z_{N_v}) = \sum_{i=1}^{N_v} I(z_i) - q \sum_{i=2}^{N_v-1} \big(d(z_{i+1},z_i) - d(z_i,z_{i-1}) \mod 8\big) \ . \tag{6.2}$$

Here, the z_i, $i \in \{1,\ldots,N_v\}$, are the points on a curve, and $d(z_{i+1},z_i)$ is a number from 0 to 8 that codes the direction of the segment (z_{i+1},z_i). Maximising g in (6.2) provides a curve passing through points of high image intensity while keeping the curvature low. In doing so, some further constraints were imposed on the admissible points.

Martelli [Mar76] pointed out that every DP problem can be stated as a problem of finding the shortest path in a graph. He applied the A^* search algorithm to find optimal curves, when the start and goal nodes are given. Although Martelli employed a shortest path algorithm, his algorithm is still a DP approach very similar to [Mon71].

Fischler et al. [FTW81] were the first to interpret the problem of finding optimal curves in images as a shortest path search in a graph whose vertices are the image pixels. They suggested to apply either the A^* or the F^* search algorithm. Thereby, the efficiency was significantly improved compared to the classic DP approach. Note, that Cohen and Kimmel [CK97] (see Section 3.1.4) later adapted Fischler et al.'s idea and enhanced it by using the fast marching method [Set96] instead of graph search. The metrication error, a grid bias intrinsic to graph-based algorithms and resulting in blocky contours (see e.g. [BK03]), can be avoided with this PDE-based approach.

Mortensen and Barrett [MB95] developed a *live-wire* algorithm that incorporates a Dijkstra-like optimisation scheme (see Section 2.4.2) into a sophisticated user interface for image segmentation.

Geiger et al.'s [GGCV95] method requires the user to provide a rough polygonal approximation of the desired contour. The algorithm determines search windows, in which the optimal contour is computed using DP. By contrast to the approaches in [AWJ90, WS92], Geiger et al.'s method is not iterative and is guaranteed to find the optimal contour within the search window. Yet the computational cost is high: approximately cubic in the side length of the search window with a large constant factor. The authors implemented a multiscale scheme and some heuristics to speed up the algorithm.

Jermyn and Ishikawa [JI01] were the first to compute globally optimal closed contours without initialisation by the user. They studied ratio energies of the form

$$E(\Gamma) = \frac{\int_\Gamma \langle \nabla I, \nu \rangle \, ds}{\int_\Gamma ds} \,. \tag{6.3}$$

Identifying the image with a 2D grid graph G=$\langle \mathcal{V}, \mathcal{E} \rangle$ (cf. Fig. 2.8(a)), the discrete version W of energy (6.3) is

$$W(C) = \frac{\sum_{e \in C} \lambda(e)}{\sum_{e \in C} \tau(e)} \,. \tag{6.4}$$

Here, C denotes the discrete curve comprising a set of edges, and $\lambda : \mathcal{E} \to \mathbb{Z}$ and $\tau : \mathcal{E} \to \mathbb{Z}^+$ are the respective edge weight functions. Minimisation of (6.4) is achieved by an algorithm by Lawler [Law66]. Lawler had observed earlier that this minimisation problem is equivalent to finding t^* and C_t^* such that C_t^* is the minimum weight cycle in a graph $\langle \mathcal{V}, \mathcal{E} \rangle$ with edge weights $w_{t*}(e) = \lambda(e) - t^* \tau(e)$, and $\sum_{e \in E} w_{t^*}(e) = 0$. This can be achieved by repeatedly decreasing t until the graph with edge weights w_t has no longer negative cycles. In each iteration the graph has to be checked for negative cycles by e.g. the Bellman-Ford-Moore algorithm [Bel58] yielding a computational complexity of $O(|\mathcal{D}|N^2)$, where N is the number of image pixels and $|\mathcal{D}|$ is the size of the used neighbourhood system (see Section 2.4.2).

While the approach proposed by Jermyn and Ishikawa is elegant and efficient, there are some practical restrictions too. First, the options for user interaction are apparently very limited. Second, in spite of the scale invariant energy (6.3) the method tends to pick small-, rather than large regions, as pointed out in [SC07].

Schoenemann and Cremers [SC07] proposed to improve Jermyn and Ishikawa's method

by including curvature in the energy. They considered the functional

$$E(\Gamma) = \frac{\int_\Gamma \langle \nabla I, \nu \rangle \, ds}{\int_\Gamma (|\kappa|^q + \lambda) \, ds} \,, \tag{6.5}$$

where κ is the curvature, ν the curve normal, λ a positive parameter and $q \in \{1, 2\}$. Schoenemann and Cremer's experiments support that taking into account the curvature removes the bias towards small curves. However, this is achieved at the expense of efficiency: a very large product graph is needed in which each vertex corresponds to a pixel in conjunction with one particular direction for incoming edges. This results in a graph with $|\mathscr{D}|N$ vertices and $|\mathscr{D}|^2 N$ edges, summing up to a complexity of $O(|\mathscr{D}|^3 N^2)$ for each iteration of Lawler's [Law66] minimum ratio algorithm. Large neighbourhood systems have to be used to obtain a realistic approximation of curvature – Schoenemann and Cremers use $|\mathscr{D}| = 32$ (cf. Fig. 2.8(b)) – making their algorithm too slow for most applications.

Recently, Windheuser et al. [WSC09] proposed an algorithm for boundary segmentation given imprecise user input. They employed the following second order energy:

$$E(\Gamma) = \alpha \sum_{i=1}^{K} d(\Gamma, z_i) + \int_\Gamma (f(s) + \lambda |\kappa|^2) \, ds \,, \tag{6.6}$$

where the z_i are the K nodes provided by the user, α and λ are coefficients, d is a distance function, and f is an edge indicator function such as (3.8). Minimisation of the energy (6.6) is achieved by a shortest path computation in a large graph: in order to incorporate the curvature, a product graph is defined as in [SC07]. Then, a copy of the product graph is added for each provided node, resulting in a four-dimensional layered graph. After adding virtual start and end vertices, optimal open curves can be computed by Dijkstra's algorithm (DA). Since the graph has $N|\mathscr{D}|K$ vertices and more than $N|\mathscr{D}|^2 K$ edges, the complexity of DA becomes about $O(N|\mathscr{D}|K \log(N|\mathscr{D}|K))$. Closed contours can be found, when one point of the contour is provided. Note, that automatic image segmentation is not possible with this model.

6.2.2 Applications of Elastica in Computer Vision

The problem of the elastica were first studied and partly solved by James Bernoulli in the early 1690s. Later, Euler [Eul44] was the first to give a comprehensive characterisation of the elastica, hence they are often attributed to Euler. Elastica are minimisers of the energy

$$E_{el}(\Gamma) = \int_\Gamma (\alpha \kappa^2 + \beta) \, ds \,, \tag{6.7}$$

where α and β are positive constants. The classic elastica are open curves connecting two fixed endpoints. In this setting, minimisers can be computed explicitly, however including elliptic functions (see e.g. [Mum94]). Basic properties of elastica are reviewed in Appendix 6.A.1.

In their pioneering work [SU88], Sha'ashua and Ullman introduced energies in the spirit of elastica to the Computer Vision community. They incorporated discrete curvature into a saliency measure. Given a curve Γ comprising the edges $p_i, p_{i+1}, \ldots, p_N$, they defined the saliency of Γ as

$$\phi(\Gamma) = \sum_{j=i}^{i+N} \sigma_j \rho_{ij} C_{ij} \,, \tag{6.8}$$

with

$$\sigma_j = \begin{cases} 1, & \text{if } p_j \text{ is edge} \\ 0, & \text{else} \end{cases} \tag{6.9}$$

and $\rho_{ij} = \rho^{g_{ij}}$ where in turn $\rho \in [0,1)$ is a constant and g_{ij} is the number of gap, i.e. non-edge, elements between p_i and p_j. Further

$$C_{ij} = e^{-\int_{p_i}^{p_j} \kappa^2(s)\, ds} \,. \tag{6.10}$$

To sum up, Sha'ashua and Ullman's saliency measure takes large values for long curves with little curvature and gaps. The saliency of each image pixel is then defined as the maximum saliency of all possible curves emanating from the pixel. This map is computed by a multistage optimisation algorithm using dynamic programming. Note, that an edge map or a line drawing is necessary to apply this algorithm. As pointed out by Alter and Basri [AB98], the computational complexity of the method is $O(|\mathscr{D}|^2 N^2)$, where $\mathscr{D}$ and N are defined as previously in this chapter.

Later, Mumford's work [Mum94] on elastica attracted considerable interest in the Computer Vision community. He pointed out that elastica can be used to naturally interpolate occluded edges. In fact, Mumford proved rigorously that elastica are maximum likelihood curves for the reconstruction of subjective contours in the spirit of Kanisza [Kan74].

Subsequently, elastica were applied to edge grouping in several works, see e.g. [TW95, EZ96, MTW99, WKS05]. All these works apply an edge detecting- prior to the grouping step, what makes them suitable for segmenting salient objects with strong edges. Similarly, elastica were applied to image inpainting, see e.g. [CKS02, AM03].

Recently, image segmentation algorithms including curvature terms were proposed in [SC07, SYMS08, ZC07, WSC09]. Both Schoenemann and Cremers [SC07], and Windheuser et al. [WSC09] used large product graphs to find optimal solutions (see the detailed review above). Sundaramoorthi et al. [SYMS08] applied level sets in conjunction with Sobolev gradient flows to implement second-order energies. Zhu and Chan [ZC07] also employed level sets to capture illusory contours of Kanizsa-type objects [Kan74].

A further application proposed by Shah [Sha02] describes contour smoothing while preserving corners.

6.3 Second-order Energies for Active Contours

Below, two new image segmentation energies are introduced. It will be proven in Section 6.3.2 that these energies, as opposed to classic first-order energies such as (6.1), possess nontrivial global minimisers. Unlike the second-order ratio energy (6.5), they can be minimised efficiently by a newly developed bidirectional search scheme for second-order energies (Section 6.4).

6.3.1 Statement of the Energies

Let $I : \Omega \to \mathbb{R}^+$ be a greyscale image on the rectangular domain Ω and $\Gamma : J \to \Omega$ be a closed contour. Further, let $f : \Omega \to \mathbb{R}^+$ denote an edge indicator function, taking small values close to desired image features and larger values elsewhere.

This chapter proposes to add curvature terms to (6.1) as follows:

$$E_1(\Gamma) = \int_\Gamma f(s)(c\kappa^2 + 1)\, ds \tag{6.11}$$

and

$$E_2(\Gamma) = \int_\Gamma \left(c\kappa^2 + f(s)\right)\, ds\,, \tag{6.12}$$

where κ is the curvature of Γ, and c is a positive constant.

Note, that the energies (6.11) and (6.12) differ from those discussed in [SYMS08]. Sundaramoorthi et al. introduced two different second-order energies: the first one is scale invariant, while the second one increases the arc length of the evolving curve. In contrast to their work, the energies defined here have an equilibrium of a fixed scale, depending on the parameter c. This property, that manifests itself in a priori bounds for the arc length of potential minimisers (see Proposition 13), can be useful in applications where the scale of the object of interest is roughly known in advance. Further, it is questionable, whether the second-order energies introduced in [SYMS08] allow exact or approximate global minimisation.

Note, that anisotropy can easily be added to (6.11) and (6.12) by replacing κ and f with $\kappa(.,\nu)$ and $f(.,\nu)$, respectively. Hence, the models become more flexible, and corners in certain directions can be allowed.

6.3.2 Properties of the Proposed Energies

There is a close relationship between the energies E_1 and E_2 proposed in Section 6.3.1 and both the classical GAC energy (6.1) and the elastica energy (6.7). On the one hand, energies

E_1 and E_2 can be interpreted as a second-order regularisations of the GAC functional (6.1). On the other hand, they can be understood as a weighted version of the elastica, attracted to certain image features.

The key difference between the two proposed energies is the following: while the curvature term is weighted with the feature detector function f in E_1, it has a constant weight c in E_2. Thus, the curvature regularisation prior is strong in E_2, making this energy suitable for robust segmentation of objects with comparatively simple shapes, such as approximately round or elliptic objects, even under strong noise. In energy E_1, the curvature regularisation is relaxed when the contour proceeds along an edge, therefore this energy can model a broader range of object shapes. It should be used when salient objects with strong edges are to be segmented. The different characteristics of the two energies are illustrated on sample images in Fig. 6.10 (Section 6.6).

Theorem 15 in Appendix 6.A.1 states that for a constant image, i.e. $f \equiv 1$, the global minima of E_1 and E_2 are circles of radius $\sqrt{c}$. Therefore, the global minima of E_1 and E_2 in an image with very weak edges ($1 \geq f \geq 1-\varepsilon$ with a small $\varepsilon > 0$) can be expected to be close to circles of radius $\sqrt{c}$.

Below, a priori estimates for the arc length of the minimisers of (6.11) and (6.12) are provided. The proof is an adapted version of a proof previously given in [Kru99].

Proposition 13. *Let $\mathscr{C} = \{\Gamma | \Gamma : J \to \Omega\}$ be the set of all smooth, closed curves in Ω and $\widetilde{\mathscr{C}} \subset \mathscr{C}$ an arbitrary subset. Further let $\Gamma_0 \in \widetilde{\mathscr{C}}$ be an arbitrary element of $\widetilde{\mathscr{C}}$ and $f_- > 0$ be the positive minimum of f in Ω. If Γ_- is a minimiser of E_1 over $\widetilde{\mathscr{C}}$, its arc length $\|\Gamma_-\|$ meets the following a priori bounds:*

$$\frac{4\pi^2 c f_-}{E_1(\Gamma_0)} \leq \|\Gamma_-\| \leq \frac{E_1(\Gamma_0)}{f_-} \,. \tag{6.13}$$

Accordingly, for a minimiser Γ_- of E_2 over $\widetilde{\mathscr{C}}$ it holds that

$$\frac{4\pi^2 c}{E_2(C_0)} \leq \|\Gamma_-\| \leq \frac{E_2(C_0)}{f_-} \,. \tag{6.14}$$

The proof is given in Appendix 6.A.2. From this proposition it follows that bounds for the arc length of a globally minimising curve over the set of all smooth closed curves can be derived solely from the given parameters. In particular, if the position of one point of the contour is set, the respective minimising curve has bounded arc length.

Note, that lower bounds as in (6.13) do not exist for the GAC energy (6.1) where infinitesimally short curves are the global minima. Yet Proposition 13 does not make any statement

about the existence of a global minimiser, more sophisticated techniques from functional analysis (see e.g. [Dac89]) are necessary to obtain the latter. The main result of this section (Theorem 14 below) proves the existence of global minimisers in a Sobolev space (see e.g. [Ada75, Kre99]). Sobolev spaces are often the appropriate function spaces for the minimisers of functionals and the solutions of PDEs. They are defined as the closures of the space C^∞ of smooth functions with respect to certain integral norms. As opposed to the classic spaces C^k of differentiable functions, Sobolev spaces are *complete* w.r.t. these norms, by definition. Since most energies of interest in science are integral energies, it is apparent that Sobolev spaces are a natural choice to search for a minimiser. The definition of the particular class of Sobolev spaces H_p^2 in conjunction with the following theorem is given in Appendix 6.A.3:

Theorem 14. *Let $\Omega \subset \mathbb{R}^2$ be a rectangular domain, $f : \Omega \to \mathbb{R}^+$ be a continuous function and $c > 0$ constant. Then the energy*

$$E_1(\Gamma) = \int_\Gamma f(s)\left(c\kappa^2 + 1\right)\, ds \tag{6.15}$$

has a global minimiser $\Gamma_{min} \in H_p^2(J,\Omega)$ with $J = [0,1]$ the unit interval. The minimiser has a strictly positive arc length, i.e. $\|\Gamma_{min}\| > 0$.

The proof and several remarks on the theorem are given in Appendix 6.A.3.

Note, that due to the general Sobolev inequalities (e.g. [Kre99, Ada75]) the space H_p^2 can be embedded into the Holder space $C^{1,\alpha}$, with $0 < \alpha < 1/2$ (see [Kre99] for the definition of Holder spaces). Consequently, the minimising curve Γ_{min} is in $C^{1,\alpha}(J,\Omega)$ and possesses a continuous first-order derivative.

6.4 Efficient Approximation of Global Minimisers

Functionals (6.11) and (6.12) can be locally minimised by deriving the PDE for the respective gradient descents. Computing the L^2 gradient flow yields fourth-order parabolic PDEs (cf. [Kru99]). Sundaramoorthi et al. derived H^1 (i.e. Sobolev-) gradient flows to reduce the order of the resulting PDE to two. This PDE-based approach is quite slow and delivers only local minima. A different, graph-based approach to minimise the energies E_1 and E_2 (see Section 6.3.1) is presented here. Subsequently, this section develops efficient algorithms with a complexity of $O(N \log N)$ to compute approximate global minimisers of the above energies, when appropriate constraints are provided (N is the number of image pixels). Later, in Section 6.5,

these core algorithms are incorporated into a user-guided and an automatic segmentation scheme. The goal is to combine the strengths of the frameworks presented in [CK97] and [SC07], resulting in convergence within a reasonable time frame.

The following scenarios are studied in this section:

1. Open contours with directional constraints in one endpoint;
2. Open contours with directional constraints in both endpoints;
3. Closed contours with direction constraints in one point.

Each of these steps is based on the respective previous steps.

Grid graphs

The graphs used for the proposed pseudo-elastica algorithms are directed 2D grid graphs $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ (see Section 2.4.2) with vertices $\mathcal{V}$ that are embedded into $\mathbb{R}^2$ in a regular grid-like manner. The edges are generated by topologically identical neighbourhood systems $\mathcal{D}$ for all vertices. In order to obtain realistic approximations of the curvature, a large 32-neighbourhood system is chosen (cf. Fig. 2.8(b)). Due to its grid graph structure, the vertices and edges of $\mathcal{G}$ can be identified with vectors in $\mathbb{R}^2$.

6.4.1 Open contours with directional constraints in one endpoint

This scenario is the easiest to solve. Yet some key concepts of the framework developed in this chapter can be illustrated in this step.

The proposed algorithm generalises well-known Dijkstra type algorithms [Dij59, Set96] to second-order energies. Incorporating curvature into dynamic programming is not new. Yet previous approaches are either iterative schemes computing local energy minima (e.g. [AWJ90, WS92]), or quite inefficient ([SU88, GGCV95, SC07]). The approach presented here differs from these works; it is based on Dijkstra's algorithm and is computationally efficient.

Minimal paths & Dijkstra's algorithm (DA)

Let $G = \langle \mathcal{V}, \mathcal{E} \rangle$ be a directed graph with non-negative edge weights and $v, w \in \mathcal{V}$ two vertices. Consider the problem of finding a minimal path from v to w. DA starts from vertex v with distance value zero and then updates all neighbouring vertices with a tentative distance value. Since all edge weights are non-negative, the smallest tentative distance value can be fixed and a new iteration of updating the neighbour vertices begins. Once the distance value for w is fixed, the shortest path can be tracked back. An important feature of DA is that for

every vertex with a fixed distance value its predecessor vertex in the shortest path is known. This property shall be used to estimate the curvature of the contour.

Dijkstra's algorithm is reviewed in more detail in Section 2.4.2.

Dynamically computing curvature

The estimation of the curvature is achieved similarly as in e.g. [SU88, SC07]: by computing the angle between two adjacent edges. As stated in Section 6.2, both these algorithms have at least quadratic complexity in the number of image pixels N. In order to obtain a more efficient algorithm, the scheme developed here incorporates curvature estimation into a greedy core algorithm, that turns out to have the considerably lower complexity $O(N \log N)$.

Figure 6.1(a) illustrates the main idea of approximating the curvature, under the assumption that the minimal path up to pixel p has already been computed. In order to update the neighbours of p, the edge weights need to be defined first. This is achieved dynamically: while length of the line segment and value of the edge indicator function are fixed for a particular edge, the curvature values are not. Next, the curvature is approximated analysing the respective angles in the contour. In Fig. 6.1(a), for instance, the weights of the edges (p,q) and (p,r) are computed with curvature values derived from the angles γ_1 and γ_2, respectively. Note, that these angles have absolute values $\leq \pi$ and are signed, e.g. γ_1 and γ_2 in Fig. 6.1(a) have different signs. Apparently, the curvature cannot be defined for a single edge, but rather for a pair of edges. Thus, the weight of an edge is computed from the curvature and a weighted average of f on the edge e and its predecessing edge e_p (see (6.17) below). For the energy E_1, the weight ω of the edge $e = (p,q)$ is computed as follows (with $e_p = (w,p)$):

$$\omega(e) = \widehat{f}(e_p, e)(c\kappa_{\gamma_1}^2 + 1)\ell(e) \tag{6.16}$$

$$\text{with} \qquad \widehat{f}(e_p, e) = \frac{\ell(e_p)}{\ell(e_p)+\ell(e)} f(e_p) + \frac{\ell(e)}{\ell(e_p)+\ell(e)} f(e) \tag{6.17}$$

with $\ell(.)$ the length of an edge. E_2 is treated analogously. For the starting edge, that obviously does not have a predecessing edge, assume $\kappa = 0$ and define

$$\omega(e) = f(e)\ell(e)\,. \tag{6.18}$$

The definition of the curvature term κ_{γ_1} is discussed next. A first guess is

$$\kappa_{\gamma_1} = C \cdot \gamma_1\,, \tag{6.19}$$

with C a suitable weight depending on the length of the adjacent edges (proposed in [SC07]). Yet considerations regarding the scaling behaviour of the curvature (detailed in Appendix

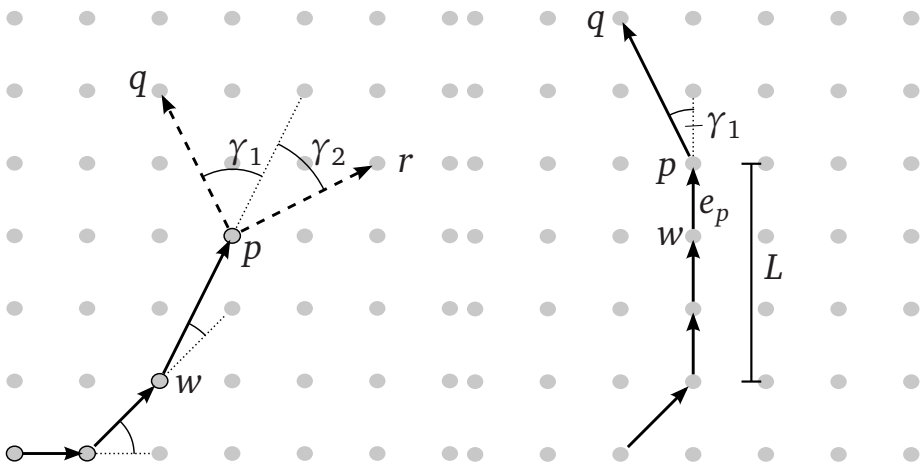


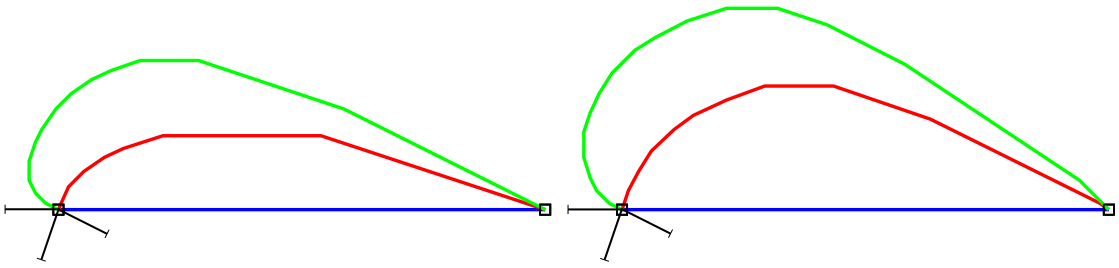
Figure 6.1: The idea of dynamic curvature estimation (a). A refined curvature approximation is achieved by tracking straight segments over multiple edges (b).

6.A.4) suggest that (6.19) has to be enhanced to suit the proposed framework. Therefore, the approximation of the curvature in (6.19) is refined as follows (see Fig. 6.1(b)): the algorithm tracks along the minimal path, when a straight line is extended over multiple edges, and sums up the total length L of the straight segment. If the absolute curvature angle $|\gamma_1|$ does not exceed a certain threshold $p_{\kappa,1}$ (roughly $2 \cdot \frac{2\pi}{|\mathscr{D}|}$), the estimated curvature in p is obtained by dividing the curvature angle γ_1 by L. Otherwise, γ_1 is divided by the length of the incoming edge $e_p = (w, p)$. This procedure can be summarised in the formula

$$\kappa_{\gamma_1} := \frac{C\gamma_1}{l} \tag{6.20}$$

$$\text{with} \quad l = \begin{cases} L & \text{if} |\gamma_1| \leq p_{\kappa,1} \\ \ell(e_p) & \text{if} |\gamma_1| > p_{\kappa,1} \end{cases}, \tag{6.21}$$

where C is a suitable constant (in the thesis: $C = 3\pi$). By this means, the curvature scales as desired for sufficiently smooth – in a discrete sense – contours differing only in their scale (cf. Fig. 6.16 in Appendix 6.A.4).



(a) (b)

Figure 6.2: Pseudo-elastica connecting two points with different starting directions and parameter $c = 10$ (a) and $c = 30$ (b).

Results for the classic elastica

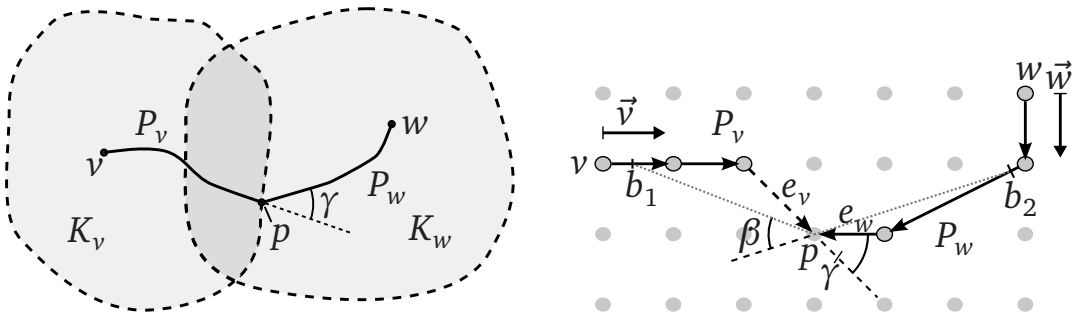
In general, the computed paths are not exact minimisers of the respective energies (6.11) and (6.12). Yet the examples in Fig. 6.2 underline that, qualitatively, the obtained contours approximate the expected solutions well in the absence of image data ($f \equiv 1$). The computed pseudo-elastica behave naturally when the parameters are changed (cf. [Mum94]): note the increase in rigidity from Fig. 6.2(a) to Fig. 6.2(b). Not surprisingly, straight path segments are favoured due to the penalisation of curvature.

6.4.2 Open contours with directional constraints in both endpoints

The procedure developed above is not directly applicable in the case of directional constraints in both endpoints. Yet the latter scenario is useful in applications, and moreover it is the basis of the algorithm for closed contours in Section 6.4.3. Below, a new algorithm is developed to establish approximate solutions using two parallel Dijkstra-like schemes (each as described in Section 6.4.1). One scheme emanates from the starting point, the other one from the endpoint. Note, that splitting a minimal path problem into two distinct distance computations is an intuitive approach. Arguably, this *bidirectional search* approach was first proposed by Pohl [Poh71]. More recently, similar ideas were pursued in the Computer Vision community (e.g. [KAB95, LY07]), see Appendix 6.A.5 for a review of the key idea. The novelty of the algorithm proposed here lies in the generalisation of this notion to second-order energies.

The bidirectional Dijkstra approach

The problem of computing pseudo-minimal paths with given directions in both endpoints (see Fig. 6.3) is tackled as follows: starting from the endpoints v and w, the distance functions



(a) (b)

Figure 6.3: The bidirectional Dijkstra scheme: (a) illustrates the key idea, and (b) details the notation.

d_v and d_w are computed simultaneously – each by a dynamic Dijkstra scheme as described in Sect. 6.4.1. Consider a 2D grid graph $\langle \mathcal{V}, \mathcal{E} \rangle$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$. The sets of vertices labeled 'known' or 'trial' by the respective Dijkstra schemes D_v and D_w after i iterations are denoted by $K_v(i)$ and $K_w(i)$ (cf. Fig. 6.3(a)). It is

$$K_{v|w}(i) = \{p \in \mathcal{V} : d_{v|w}(p) < \infty \text{ after } i \text{ iterations}\} . \tag{6.22}$$

With increasing i, the sets $K_v(i)$ and $K_w(i)$ propagate and, eventually, start to overlap (Fig. 6.3(a)). For a vertex p in the intersection the preliminary pseudo-minimal paths (as in Section 6.4.1) can be tracked back to the endpoints v and w, respectively. The following paragraph details how the second-order energies (6.11) and (6.12) of the concatenated path can be approximated by a score value.

By contrast to the procedure for first-order minimal paths, the smoothness of the intersection between the partial paths has to be taken into account. Assume, that the bidirectional scheme has progressed to a stage i_0, where $K_v(i_0)$ and $K_w(i_0)$ overlap, i.e. $K_v(i_0) \cap K_w(i_0) \neq \emptyset$. Further assume, without loss of generality, that D_v updates a pixel p which has already been visited, and labeled 'trial' or 'known', by D_w. Preliminary pseudo-minimal paths P_v and P_w can then be tracked back from p to v and w, respectively (see Fig. 6.3(a)). The curvature in p is approximated as follows (see Fig. 6.3(b)): first, the local curvature is computed from the angle γ, describing the change of direction of the adjacent edges e_v and e_w. If $|\gamma|$ is below a certain threshold $p_{\kappa,2}$ (e.g. $\pi/4$) the intersection is at least locally sufficiently smooth. In this case, the curvature is reestimated in a larger neighbourhood in order to impose smoothness on a larger scale. The simple scheme in [BM75] is used: given a neighbourhood size parameter m , the scheme localises the two points b_1, b_2 that have the distance m from p. Here,

distance means arc length distance on the path, rather than euclidean distance in $\mathbb{R}^2$. Then, the refined curvature is computed from the change of direction β in the m-neighbourhood (Fig. 6.3(b)). A score for the quality of the concatenated path is summed up from the partial distance values $d_v(p)$ and $d_w(p)$, and the intersection energy. The latter is computed using (6.16), and the curvature value discussed above (see Algorithm 4).

Algorithm 5 on page 118 presents pseudocode for the bidirectional Dijkstra search scheme. For the sake of clarity, single endpoints v and w are assumed, yet the generalisation to lists of endpoints, necessary e.g. for the user-guided scheme proposed in Section 6.5.2, is straightforward. The difference between the score- and the energy value in Algorithm 5 is explained in the following paragraph.

Algorithm 4: Computation of the intersection energy (here for energy E_1 (6.11)).

Input: Vertex $p \in \mathcal{V}$;predecessor functions $pred_v : \mathcal{V} \setminus v \to \mathcal{V}$ and $pred_w : \mathcal{V} \setminus w \to \mathcal{V}$

Output: Intersection energy E_I.

$u \leftarrow pred_v(p)$; $r \leftarrow pred_w(p)$ `// get predecessors`

$e_v \leftarrow (u, p)$;$e_w \leftarrow (r, p)$

$l_{av} = 0.5 \cdot (\ell(e_v) + \ell(e_w))$ `// average length needed for weighting`

compute angle γ between edges e_v and e_w `// see Fig. 6.3(b)`

if $|\gamma| \leq p_{\kappa,2}$ **then** `// intersection is locally smooth`

  compute b_1, b_2 with path distance m from p

  compute angle β between lines $\overline{b_1 p}$ and $\overline{p b_2}$

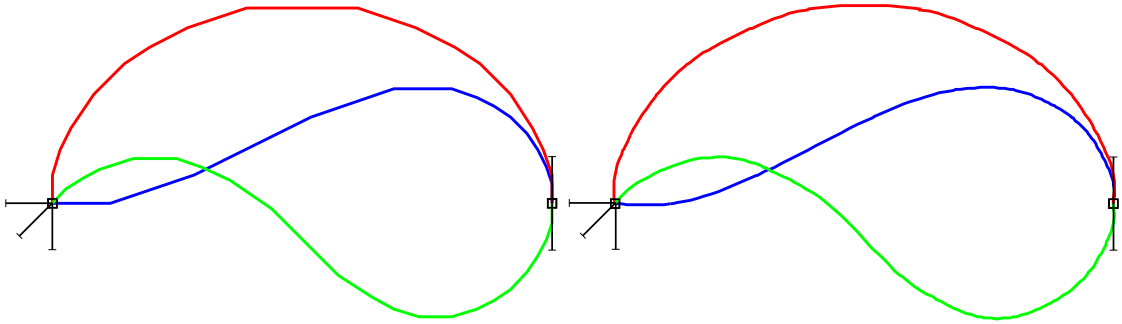
  $\kappa \leftarrow C\beta / l_{av}$ `// cf. (6.20)`

else $\kappa \leftarrow C\gamma / l_{av}$

$E_I \leftarrow \widehat{f}(e_v, e_w)(c\kappa^2 + 1)$ `// see (6.16)`

Results for the classic elastica

The results from the suggested bidirectional scheme for classic elastica ($f \equiv 1$) are shown in Fig. 6.4. Qualitative comparisons with the results in [Mum94] confirm that the curves computed with the novel approach approximate the exact elastica quite well.



(a) (b)

Figure 6.4: Open pseudo-elastica with different directional constraints, visualised without (a) and with (b) spline interpolation (parameter $c = 30$).

Algorithm 5: Pseudo-minimal paths using two simultaneous Dijkstra schemes.

Input: weightfunction ω; endpoints v, w; start/end-directions $\vec{v}$, $\vec{w}$; parameters p_E, p_t with $1 < p_E \leq 2$ and $1 < p_t$; hitherto minimal energy E_{min}.

Output: Pseudo-shortest path P.

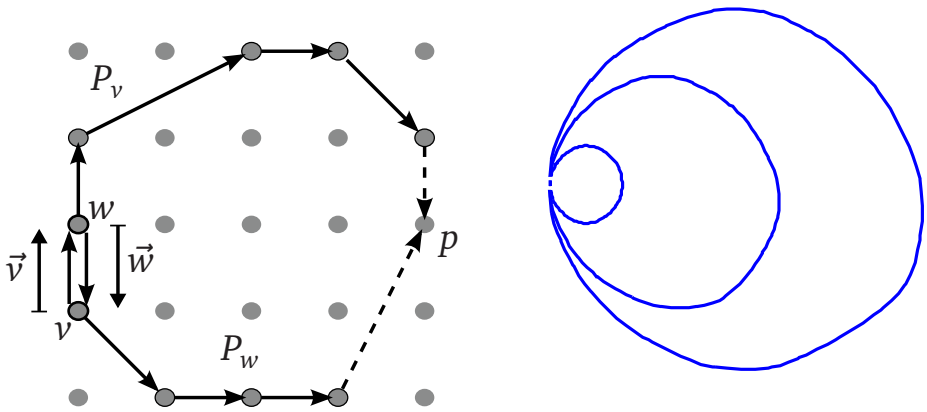
$E_{v|w} \leftarrow 0$; $S_{min} \leftarrow \infty$; $E_{min,old} \leftarrow E_{min}$;
foreach *vertex* p **do**
    $d_{v|w}(p) \leftarrow \infty$;
$d_v(v) \leftarrow 0$; $d_w(w) \leftarrow 0$;
insert v into Q_v and w into Q_w; `// initialise priority queues`
while $\Big($*(Q_v not empty and $E_v \leq E_{min}$) or (Q_w not empty and $E_w \leq E_{min}$)*$\Big)$
    and $\Big(E_v + E_w < p_E \cdot E_{min}\Big)$ **do**
  `// Dijkstra step in the first branch`
  if Q_v *not empty and* $E_v \leq E_{min}$ **then**
    pop $u = \arg\min_{q \in Q_v} d_v(q)$ from Q_v;
    $E_v \leftarrow d_v(u)$;
    foreach *admissible neighbour* p *of* u **do**
      compute weight $\omega((u,p))$; `// as per (6.16)`
      update $d_v(p)$, $pred_v(p)$ and Q_v if necessary;
      if $d_w(p) < \infty$ **then** `// Was` p `reached by other branch yet?`
        $r \leftarrow pred_w(p)$;
        compute intersection energy E_I; `// see Algo. 4`
        $S \leftarrow d_v(p) + d_w(p) + E_I$; `// compute score`
        if $S < S_{min} \wedge S < p_t \cdot E_{min}$ **then**
          update S_{min};
          $E_{min} \leftarrow S_{min}$;
          store $u_- \leftarrow u$, $p_- \leftarrow p$ and $r_- \leftarrow r$;
  `// Second branch analogously`
  ⋮
  construct P from u_-, p_-, r_- and $pred_{v|w}()$

Energy vs. score

Algorithm 5 distinguishes between the minimal *score* value S_{min} and the minimal *energy* value E_{min}. For better understanding of this distinction note the following (all line numbers below refer to Algorithm 5):

- Only the score value is of relevance for the computation of a single path – open or closed – since $E_{min} = \infty$ is passed to the algorithm and subsequently $E_{min} = S_{min}$ (see line 20).
- Yet the energy value has some importance when the minimal among several contours with different endpoints has to be computed. After the computation of one contour, its energy value is consolidated in a post-processing step (see Section 6.5.3). If the consolidated energy is smaller than the hitherto minimal energy value, it is stored in E_{min} and passed to the algorithm for the computation of the next contour. The condition in line 7 effects that the bidirectional scheme stops when the path is unlikely to become minimal any more (depending on parameter p_E).
- The values S_{min} and E_{min} are updated, if $S < S_{min}$, and S is smaller or slightly larger than E_{min} (depending on parameter p_t, see lines 18-20). Consequently, the E_{min} value, when passed to the algorithm before, may temporarily be replaced by a larger value. This procedure takes into account that the consolidated energy (see Section 6.5.3) of the candidate path might nevertheless be minimal, since it usually deviates slightly from the score value.

The choice of all the parameters, including p_E and p_t, is detailed in Appendix 6.A.6.



(a) (b)

Figure 6.5: Computing closed curves (a): the two branches start from v and w, respectively, and intersect in p. Closed pseudo-elastica for different c-values (b).

6.4.3 Closed contours with a given direction in one point

The algorithm proposed in Section 6.4.2 can compute closed contours when one point v of the contour and the respective starting direction $\vec{v}$ are given (Fig. 6.5(a)). In this case, one has to set $w = v + \vec{v}$ and $\vec{w} = -\vec{v}$, and apply Algorithm 5 as described above. Only two minor adaptions have to be made:

- The trivial points v and w are not allowed as possible intersection points p;
- The segment between v and w is traversed by both partial paths P_v and P_w, therefore the weight of this segment must be split. Instead of applying (6.18), the edge weights in the respective Dijkstra scheme are set to

$$\omega(e) = \frac{1}{2} f(e)\ell(e) \qquad \text{for } e \in \{(v,w),(w,v)\} \ . \tag{6.23}$$

Results for the classic elastica

Again, the algorithm is tested first in the simple, yet challenging case of the classic elastica, where the actual solutions are well-known. Theorem 15 (p. 140) states that the unique minimisers of the classic elastica energy (6.7) are circles of a certain radius. Figure 6.5(b) shows the pseudo-elastica computed for different coefficients c with the above approach. These results suggest the following two conclusions: first, the approach presented here is not suitable for computing quantitatively exact minimisers of the functionals (6.11) and (6.12), as the contours in Fig. 6.5(b) show slight deviations from the analytic solutions, being the perfect

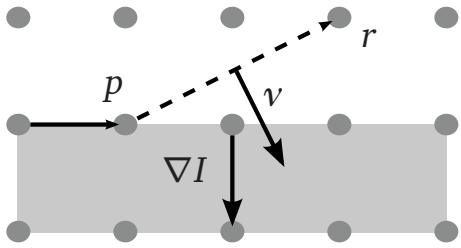


Figure 6.6: Aligning an edge to the object boundary.

circles. Yet second, the approximate solutions are qualitatively quite satisfactory, even in the absence of features to guide the contour.

These results for the classic elastica, i.e. $f \equiv 1$ in (6.11,6.12), underline that the proposed energies do not have the bias towards very short curves such as the classic GAC. Further, the results indicate that energy (6.12) is particularly suitable for the segmentation of approximately round objects in noisy images (cf. Fig. 6.13, p. 134).

6.5 Application to Image Segmentation

The above framework for efficient optimisation of second-order energies can be applied to both user-guided and automatic image segmentation schemes. This section discusses the implementation of these schemes in detail.

6.5.1 Edge detector functions

In order to take advantage of all the information at hand, a suitable edge detector function has to be defined. Consider the function introduced in [CCCD93, MSV95] and classically used for geodesic active contours:

$$f = \frac{1}{1 + |\nabla I_\sigma|^2} \,. \tag{6.24}$$

Here, I_σ is an image smoothed by convolution with a suitable kernel. Yet in the graph-based approach developed here it has to be ensured that an edge of the path is aligned accurately with the object boundary in the image. This is the case, when the unit normal ν of an edge is parallel to the image gradient ∇I (see Fig. 6.6), i.e. the inner product $\langle \nu, \nabla I \rangle$ takes large absolute values. Therefore, the following choice of f is reasonable:

$$f_1 = \frac{1}{1 + \lambda^2 \cdot |\langle \nu, \overline{\nabla I}_\sigma \rangle|^2} \,, \tag{6.25}$$

where the gradient $\overline{\nabla I}_\sigma := \frac{1}{2}(\nabla I_\sigma(p) + \nabla I_\sigma(r))$ is averaged along the edge (p, r), and the parameter λ controls the influence of the gradient term. In many images, the object of interest

is constantly brighter or darker than the background. In such scenarios, the function (6.25) is adapted to take full account of this additional information. Then, the appropriate detector functions are:

$$f_2^{\pm} = \frac{1}{1 + \lambda^2 \cdot \left(\max\left\{0, \pm\langle v, \nabla I_{\sigma,av}\rangle\right\}\right)^2} \,. \tag{6.26}$$

Depending on whether the object or the background is brighter, f_2^+ or f_2^- has to applied. In the following, these two complementary functions are subsumed under the notation f_2.

6.5.2 User guided segmentation

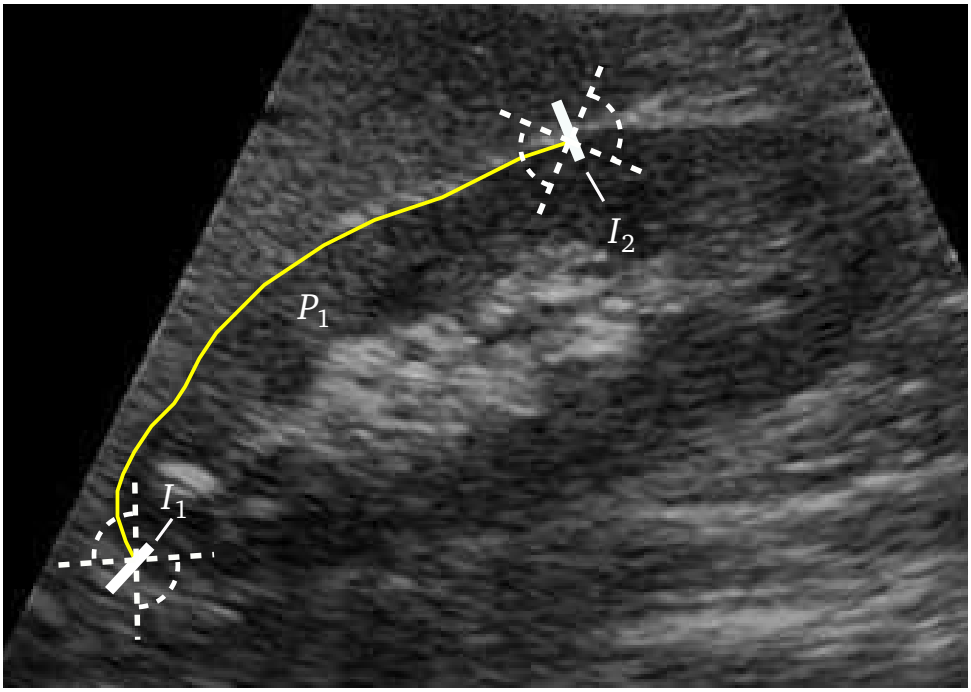
Open pseudo-elastica can be computed efficiently with the algorithm proposed in Section 6.4.2. This scheme can be applied to image segmentation, analogously to the application of shortest paths [Set96] to user-guided segmentation in [CK97]. The second-order approach developed in the preceding section provides two important advantages over first-order methods (e.g. [CK97]):

- The second-order term provides enhanced rigidity of the contour and the ability to interpolate parts of the object boundary in a natural and smooth way (cf. Fig. 6.4(b)). This is useful for the segmentation of very noisy imagery, e.g. ultrasound imagery;
- The possibility to incorporate admissible directions in the endpoints provides increased robustness and minimises user input. In most cases two streaks across the boundary of the desired object are sufficient.

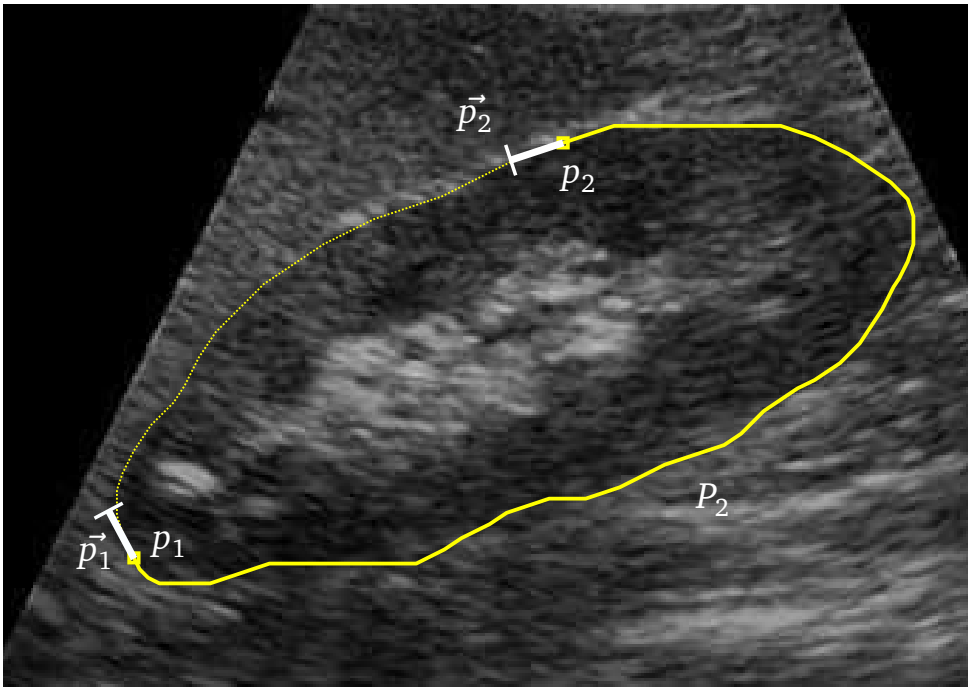
The proposed algorithm

Required input The user provides n ordered streaks $I_1, \ldots, I_n$ across the boundary of the object of interest. For simplicity, it is assumed that the streaks are given by lines. The robustness and efficiency of the algorithm is further improved, when the lines are assumed to run roughly perpendicular through the boundary (e.g. with a 45 degree tolerance on each side as shown in Fig. 6.7(a)). Note, that in many applications two streaks are sufficient, yet for very noisy or complex objects more input might be necessary.

Output A closed curve running through pixels $p_i \in I_i$. The sections P_i of the curve are pseudo-optimal in the sense of Section 6.4.2. As opposed to related first-order schemes (e.g. [CK97]), smooth transitions are ensured by the directional constraints in the intersection points (such as p_1 and p_2 in Fig. 6.7(b)).



(a)



(b)

Figure 6.7: Example for user-guided segmentation (ultrasound kidney image): two lines I_i across the object boundary are provided by the user. For the computation of P_1 only rough constraints are applied (a). The last section (P_2) is constrained to smoothly close P_1 (b).

Algorithmic procedure For each section, Algorithm 5 is applied to find a pseudo-minimal path satisfying the following constraints:

- For the first section, a contour connecting the input sets I_1 and I_2 is computed. The contour is constrained to pass roughly perpendicular to I_1 and I_2 (angles in Fig. 6.7(a)). In many applications, it is advisable to use the orientation sensitive detector function f_2 as given by (6.26). In this case, f_2 is tested with both possible orientations (i.e. f_2^- and f_2^+), and the path with the lower energy indicates the proper orientation;

- For the interior sections ($P_i, 1 < i < n$) the procedure is as follows: the pseudo-minimal path is constrained to start from the second-last vertex of the previous section (P_{i-1}) in direction of the last edge of P_{i-1}. The conditions in the endpoint are as described for the first section above;

- For the last section P_n, the starting direction is derived from P_{n-1}, while the ending direction is inferred from P_1 (Fig. 6.7(b)).

6.5.3 Automatic segmentation

The algorithm developed in Section 6.4.3 computes pseudo-optimal contours running through a given pixel. In order to obtain best possible solutions, all N pixels of an image have to be tested, and the contour with the minimal energy has to be chosen. Although the execution of the core algorithm is fast, this exhaustive search is not feasible in practice. Section 6.5.3 describes some rules and heuristics to accelerate this process considerably. However, the algorithm for closed pseudo-elastica presented in Section 6.4.3 is revisited first.

A posteriori energy consolidation for closed contours

The pseudo-elastica algorithms have been developed to achieve high efficiency, while delivering solutions close to the global minimisers of the respective energies. The exactness of the optimisation was sacrificed for the sake of practical usability.

Yet it is crucial for the proposed automatic segmentation scheme, that the quality of several pseudo-optimal closed contours, each emanating from a different pixel, can be compared by means of their energy values. The score value (S in Algorithm 5), used for the optimisation of each contour, is not a good candidate to compare different contours. It can even yield different values for the same curve depending on the intersection point. Therefore, an energy consolidation step is added to the algorithm, that is run after the pseudo-optimal contour is

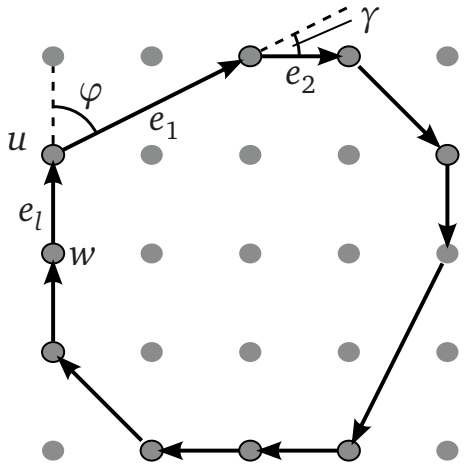
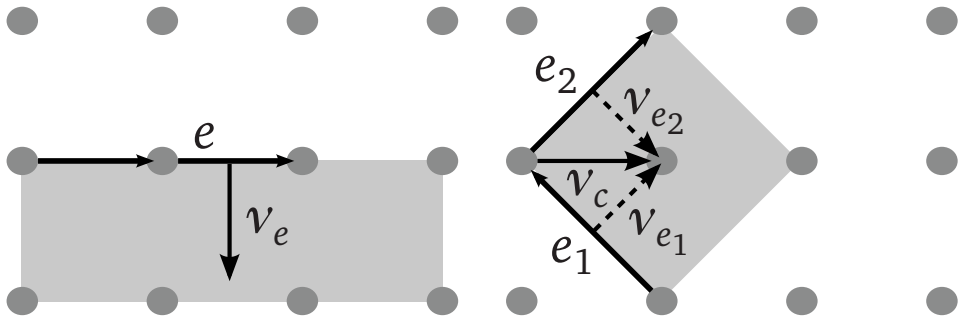


Figure 6.8: Consolidation of the contour energy.

computed. A consistent energy value is established in this step, according to the rules stated in Section 6.4.1. The consolidation is achieved as follows (see Fig. 6.8): assume, the contour was optimised starting from point w. Initially, the starting point is shifted to the next point u where the direction changes. Then the path is traversed once in each possible direction, calculating the energy according to the rules from Section 6.4.1. Each time, this procedure starts from the second edge (e_2 in Fig. 6.8), and the weight of the first edge e_1 is added after traversing the path. This strategy ensures a correct curvature estimation in u as per (6.20). Ultimately, the lower of the two energy values is chosen as the contour energy value. The latter does no longer depend on the starting point and the traversing direction.

Increasing the efficiency

Fortunately, the search for a global minimum can be sped up considerably. First, not all pixels are equally likely to belong to an optimal contour, so pixels with small values of f can be prioritised. There may be other criteria for ranking the pixels, depending on the particular application sought after. Second, the core algorithm does not have to be executed entirely for all pixels. While processing the candidate pixels from the ranked list, the energy of the current minimal contour is stored (in E_{min}, see Algorithm 5). The bidirectional Dijkstra scheme immediately aborts once the minimal energy is exceeded by a certain percentage – defined by the parameter p_E (see Appendix 6.A.6) –, as this path cannot become globally minimal any longer. Thus, the execution of the core algorithm becomes continuously faster during the optimisation. Furthermore, convergence criteria should be imposed on the process (see Section 6.6.2).



(a) (b)

Figure 6.9: Edge normal (a) and corner normal (b). The corner normal ν_c is defined as the normalised sum of adjacent edge normals (here: ν_{e_1} and ν_{e_2}).

6.5.4 Anisotropic energies

The anisotropic versions of the segmentation energies briefly stated in Section 6.3.1 can be readily incorporated in the optimisation framework. For instance, consider energy (6.12): the anisotropic variant of this energy is

$$E_{2,a}(\Gamma) = \int_\Gamma \Big(c(w_1(\nu_c)\cdot\kappa)^2 + w_2(\nu_e)\cdot f(s) \Big)\, ds\,. \tag{6.27}$$

Here, w_1 and w_2 are anisotropic weighting functions, i.e. they penalise certain orientations in the plane more than others. ν_c denotes a corner- respectively an angle-normal and ν_e denotes the usual edge normal (see Fig. 6.9). Further note, that the existence result in Theorem 14 continues to hold for these modified energies if w_1 and w_2 are continuous and strictly positive.

6.5.5 Computational Complexity

The graph in the proposed method has N vertices and about $N\cdot|\mathscr{D}|$ edges for an image with N pixels. As in [SC07], a neighbourhood system with size $|\mathscr{D}| = 32$ (cf. Fig. 2.8(b)) is chosen in order to obtain a sufficiently realistic approximation of the curvature. The heap structure for the Dijkstra scheme is implemented by a Fibonacci heap [FT87]. This results in a complexity of $O(N\cdot|\mathscr{D}| + N\log N)$ for one execution of the algorithm in Section 6.4.1, which asymptotically equals $O(N\log N)$. The algorithms in Sections 6.4.2 and 6.4.3 run two instances of the basic Dijkstra scheme simultaneously, and therefore have the same complexity.

The user-guided segmentation algorithm described above requires $(n+1)$ executions of the core algorithm from Section 6.4.2, when n streaks across the boundary are provided by the user. The complexity bound $O(N\log N)$ remains valid, yet with an increased constant.

For the automatic segmentation scheme, the worst-case complexity and the complexity, when a heuristic stop criterion is applied, have to be distinguished. In the worst case, the algorithm from Section 6.4.3 has to be run N times, once for each image pixel. This procedure results in a complexity of $O(N^2 \log N)$ and is too slow in practice. Yet if a stop criterion is applied, an approximation of the minimal energy can be obtained much faster. In the experiments with the automatic segmentation scheme, the algorithm usually stopped after $O(\sqrt{N})$ tested pixels (see Table 6.2).

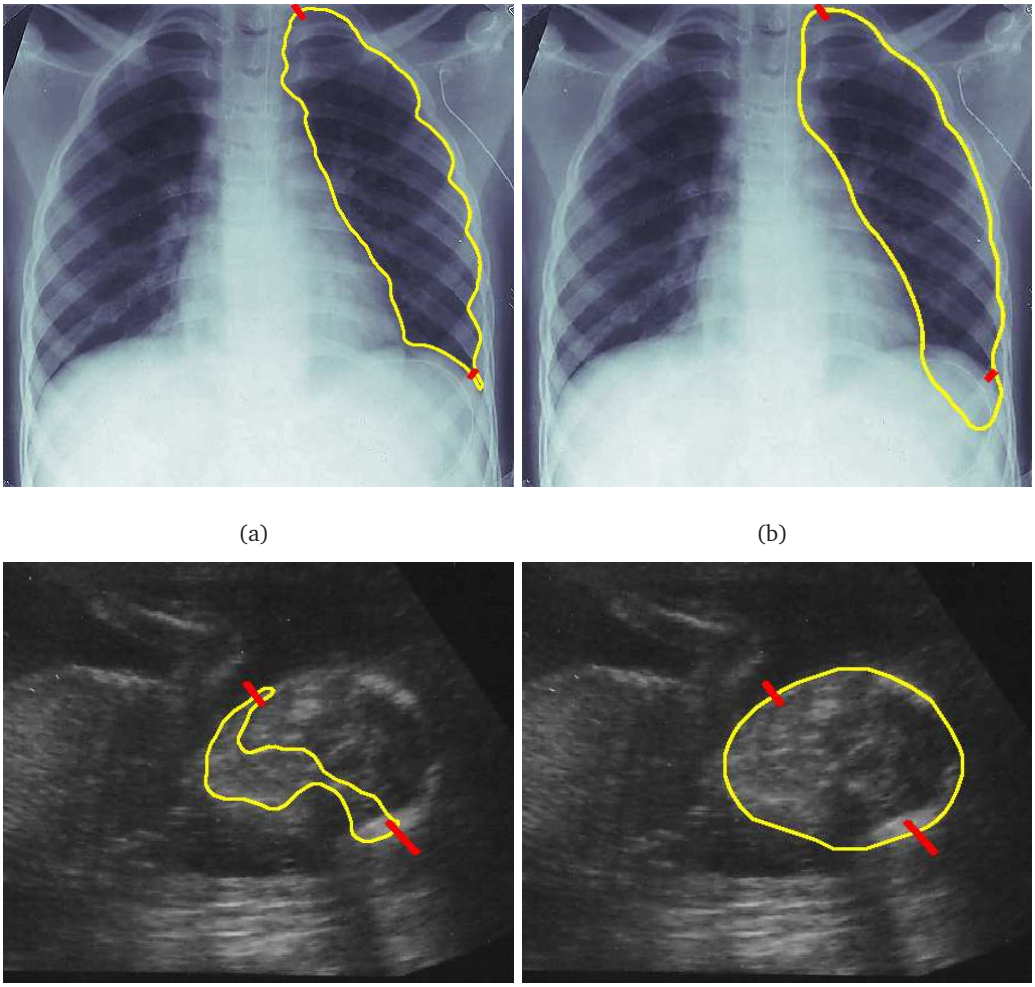
6.6 Experimental Results

This section presents results of the pseudo-elastica based algorithms for several images from different application areas, such as medical imaging and face analysis. Subsequently, the pseudo-elastica are compared to state-of-the-art segmentation methods with respect to accuracy and efficiency. All algorithms were coded in C++, and the experiments were run on an Intel Core 2 Duo 2.5 GHz machine.

Note, that all the parameters inside the pseudo-elastica core algorithms (see Section 6.4) were kept constant across all experiments presented in the thesis. The respective values are detailed in Appendix 6.A.7.

6.6.1 User guided segmentation

Comparison of E_1 and E_2 Goal of the first experiments is to emphasise the different characteristics of the energies E_1 and E_2, proposed in the equations (6.11) and (6.12) in Section 6.3. Figures 6.10(a),(b) illustrate that energy E_1 is suited better for segmenting salient objects. As the curvature term is weighted by the feature indicator function f, less regularity is imposed, and object details can be captured (see bottom right corner of the contour in Fig. 6.10(a)). The contour computed with energy E_2 (see 6.10(b)) is overregularised and fails to capture the contour of the lung in the bottom part. Figures 6.10(c) and (d) show an example image where strong regularity is desirable. Energy E_2 allows for the robust segmentation of round objects, even under strong noise and if large parts of the object boundary are missing.



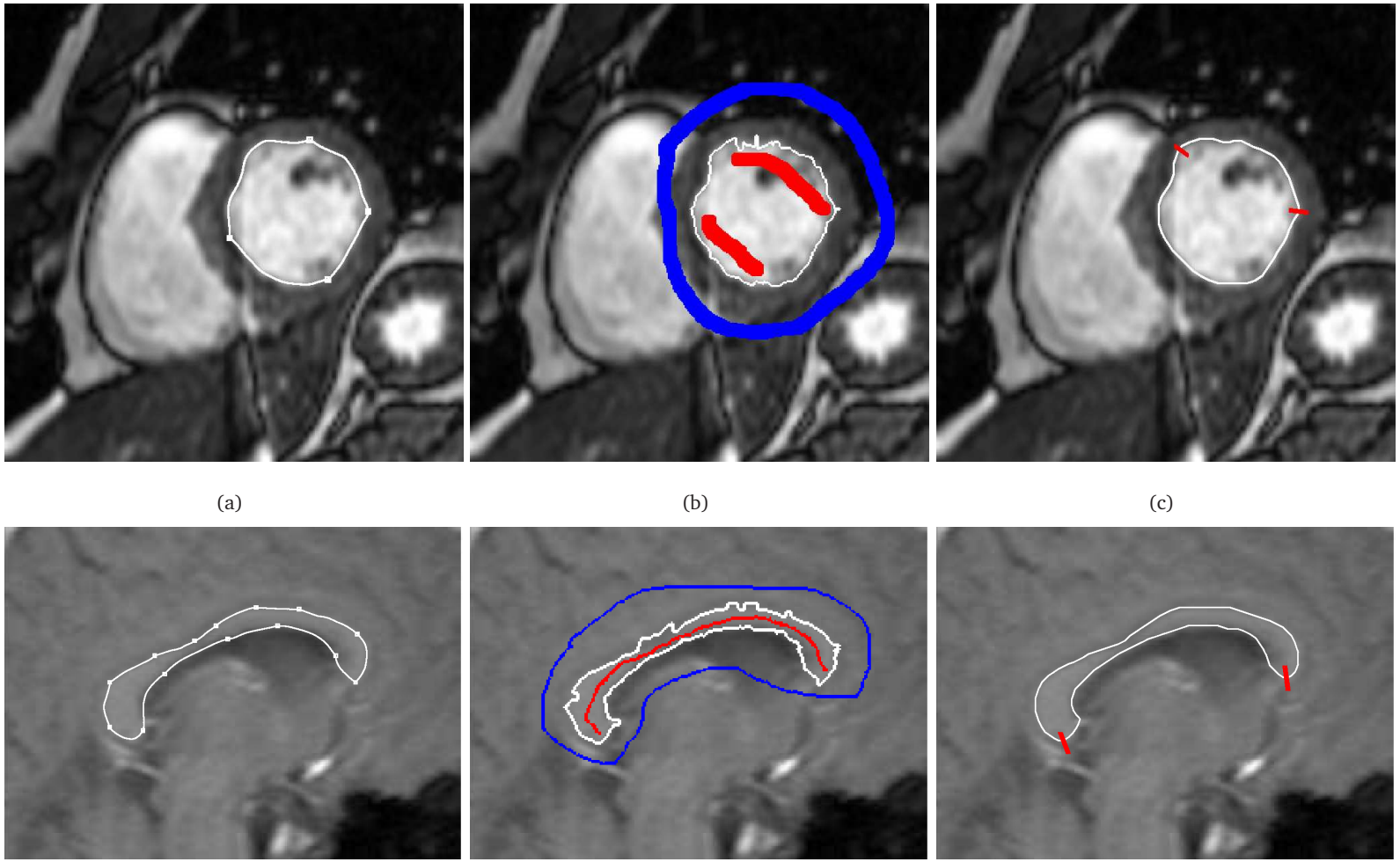
(a) (b)

(c) (d)

Figure 6.10: Comparison of the proposed energies E_1 (left) and E_2 (right). The red streaks are provided by the user to guide the segmentation (model parameters: $c, \lambda = 1$). While E_1 enables better segmentation of detailed objects (top), E_2 provides more robustness in the presence of strong noise (bottom).

CT Images Next, the algorithm is tested on medical images. The most popular methods for image segmentation require some kind of user input to enable accurate and reliable results for complex images. The pseudo-elastica method is compared with two of the top recognised schemes in the field of medical imaging: the piecewise globally optimal active contours [CK97] and the graph cut based active contours [BK03]. The parameters for the pseudo-elastica were set to $\lambda = c = 1$ for all four images shown in Figs. 6.11 and 6.12. By contrast, the parameters had to be separately set for each image for Cohen and Kimmel's method [CK97] and graph cut GACs [BK03] (see Table 6.1).

The results on two CT images are shown in Fig. 6.11, and a quantitative evaluation is given in Table 6.1. Ground truth contours were delineated manually by a medical expert. Both, visual impression and the statistical values in Table 6.1, suggest that the path-based segmentation approaches deliver better results on the two test images than the graph cut based one. The accuracy reached by the pseudo-elastica algorithm is comparable to that of Cohen/Kimmel [CK97]. However, the examples clearly show that considerably less user input is required by the pseudo-elastica method. For the corpus callosum image with its weak edges, for instance, more than 10 nodes had to be provided to Cohen and Kimmel's algorithm by the user (Fig. 6.11(d)). This is a tedious and time consuming task, that often requires several trials before an acceptable result is obtained. On the other hand, two lines across the object boundary are mostly sufficient to obtain a robust and accurate result with pseudo-elastica (Figs. 6.11(c),(f)).

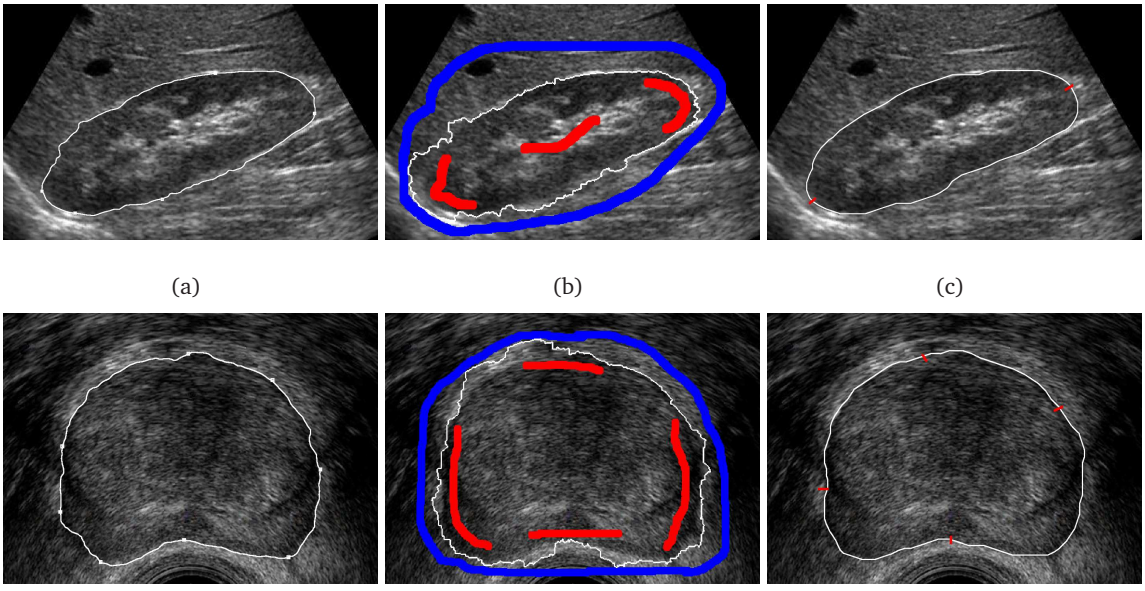


(a) (b) (c)

(d) (e) (f)

Figure 6.11: Comparison of the Cohen/Kimmel AC (left), graph cut GAC (middle) and pseudo-elastica (right) on CT images of a heart ventricle (top) and a corpus callosum (bottom). User input is visualised as squares (nodes for Cohen/Kimmel method [CK97]), blue/red streaks (seeds for graph cut) and red streaks (input lines for pseudo-elastica).

Ultrasound Images Further tests were made on ultrasound images. Such images usually contain a great amount of speckle noise and large gaps in the object boundaries which making them difficult to process. The increased rigidity of the pseudo-elastica, in conjunction with their ability to smoothly interpolate in the absence of guiding features (see Fig. 6.4), makes them a suitable candidate for this task. Accordingly, the pseudo-elastica method performed better than its competitors on the ultrasound test images. The graph cut method can hardly cope with the strong noise and gaps in the object boundaries. Cohen/Kimmel's method yields satisfactory results if sufficiently many nodes are provided by the user. Yet the evaluation in Table 6.1 indicates that, while the mean error values for Cohen/Kimmel's method and the pseudo-elastica are similar, the maximum error and the standard deviation are significantly lower for pseudo-elastica. The second-order term in the segmentation energy stabilises the contour in the presence of strong noise.



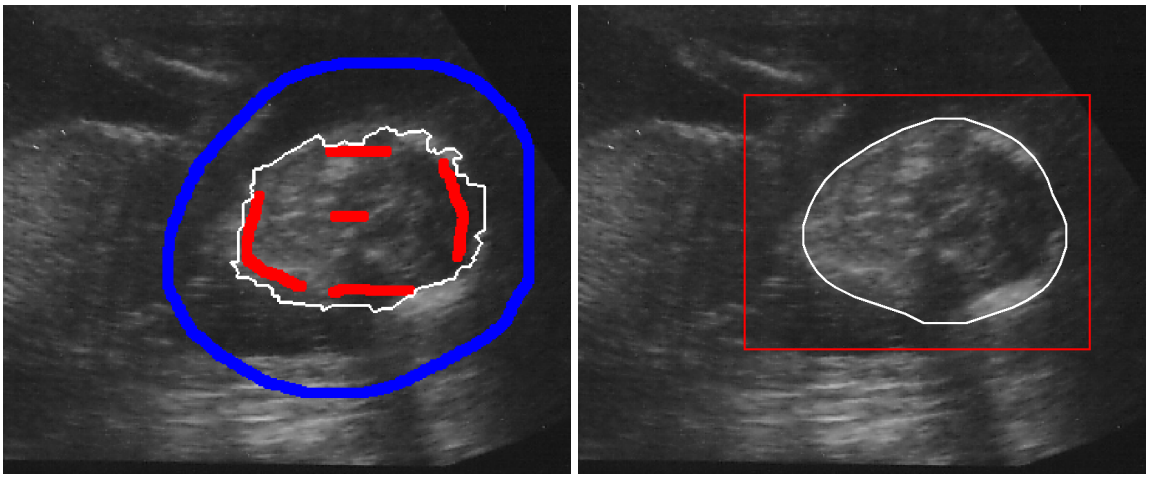
(a) (b) (c)

(d) (e) (f)

Figure 6.12: Comparison of the Cohen/Kimmel AC (left), graph cut GAC (middle) and pseudo-elastica (right) on ultrasound images of a kidney (top) and prostate (bottom). User input is visualised as squares (nodes for Cohen/Kimmel method [CK97]), blue/red streaks (seeds for graph cut) and red streaks (input lines for pseudo-elastica).

Image	Size (in pixels)	**Pseudo-Elastica** error in pixel				time	**Cohen/Kimmel [CK97]** error (in pixels)				time	**Graph cut [BK03]** error (in pixels)				time
		c	mean	max	std.	in s	w	mean	max	std.	in s	σ	mean	max	std.	in s
Heart	378×378	1.0	1.3	4.0	1.0	0.3	0.025	1.1	3.8	0.9	0.2	2.5	1.7	8.0	1.5	0.2
C. Callosum	276×200	1.0	1.1	3.3	0.7	0.4	0.25	1.2	4.3	1.0	0.2	1	1.4	7.1	1.4	0.1
Kidney	456×292	1.0	1.4	6.5	1.2	0.8	0.1	1.4	10.5	1.9	0.3	2.9	4.9	26.5	6.0	0.6
Prostate	539×389	1.0	2.3	6.9	1.7	1.5	0.1	2.5	15.3	3.0	0.6	5	8.2	28.3	7.5	0.7

Table 6.1: Accuracy evaluation for user-guided segmentation. Pseudo-elastica: energy E_1 was used for the two CT images, whereas energy E_2 was used for the two ultrasound images ($\lambda = 1$ for all four images). The parameters w for [CK97] and σ for [BK03] (see (3.14) and (3.19)) had to be chosen separately for each image in order to obtain satisfactory results.



(a) (b)

Figure 6.13: Ultrasound image of a fetal head. (a): graph cut result with user input. (b) automatically obtained result with pseudo-elastica (the search was restricted to the red window).

6.6.2 Automatic segmentation

Both the theoretical and the empirical evidence suggest that weighted pseudo-elastica are well suited for the robust segmentation of round objects even if the image is very noisy or if parts of the boundary are missing. By means of the scheme discussed in Section 6.5.3, a completely automatic process can be set up for the segmentation of such objects. As discussed in Section 6.5.5, the computational costs of the pseudo-elastica increase significantly in this scenario (cf. Tables 6.1 and 6.2). However, convergence can be reached in reasonable time if a suitable convergence criterion is applied. A simple criterion (cf. Section 6.5.3) was used in the tests presented here: when the minimal energy did not decrease over a fix number of tested pixels (a certain fraction of all pixels in the search window/image, depending on a previously defined parameter p_c), the search assumed convergence and halted. The value of the parameter was set to $p_c = 0.5$ for the experiments shown in Figs. 6.13 and 6.14, and to $p_c = 1.0$ for all eye segmentation experiments (see below and Section 7.4).

The results displayed in Fig. 6.13, Fig. 6.14, and Table 6.2 underline that the closed version of the pseudo-elastica yields good results, when round objects have to be segmented automatically and under strong regularisation constraints. Since these contraints are strongest in energy E_2 (6.12), this energy is chosen to segment the indistinct contour in Fig. 6.13. Energy E_1 is applied in the experiments shown in Fig. 6.14, in order to enable a higher segmentation accuracy.

Image	Size (in pixels)	**Pseudo-Elastica** error (in pixels) mean	max	std.	pixels tested	time (in s)	**Graph cut [BK03]** error (in pixels) mean	max	std.	time (in s)
Head	241×171	1.7	4.3	1.2	314	58	5.9	17.8	4.4	0.3
Cell 1	180×136	1.1	3.0	0.8	201	14.7	1.3	5.4	1.1	0.1
Cell 2	180×165	1.0	3.0	0.7	292	30.7	3.3	13.1	3.5	0.1

Table 6.2: Accuracy evaluation for automatic segmentation. Pseudo-elastica: energy E_2 was used for the head image ($c = 5, \lambda = 1$), whereas energy E_1 was used for the two cell images ($c = 15, \lambda = 0.3$). Graph cuts [BK03]: f as in (3.19) with $\sigma = 2.5$ for the head image, and $\sigma = 2$ and $\sigma = 3$, respectively, for the cell images.

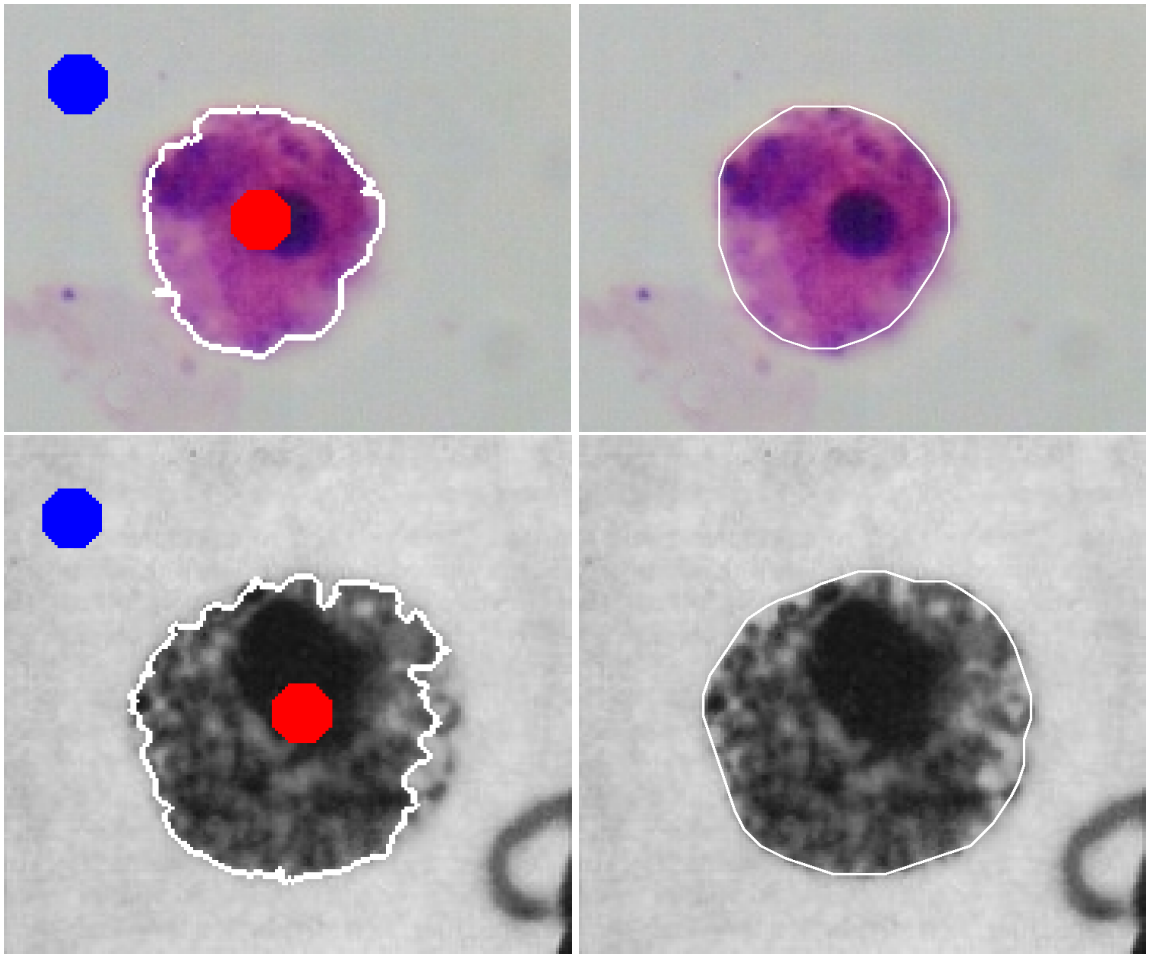


Figure 6.14: Automatic segmentation of cell nuclei. Left column: semi-automatic approach with graph cuts, right column: fully automatic approach with pseudo-elastica.

Anisotropic energies

As an application of the anisotropic energy (6.27) suggested in Section 6.5.4, eye contours can be segmented from 3D face data in an accurate and automatic manner. The results of this project are published in [KDG09]. Energy E_2, defined by (6.12), was used for the eye segmentation experiments, since its strong regularity prior is desirable in the presence of noise and gaps in the eye ridges (see Section 5.6).

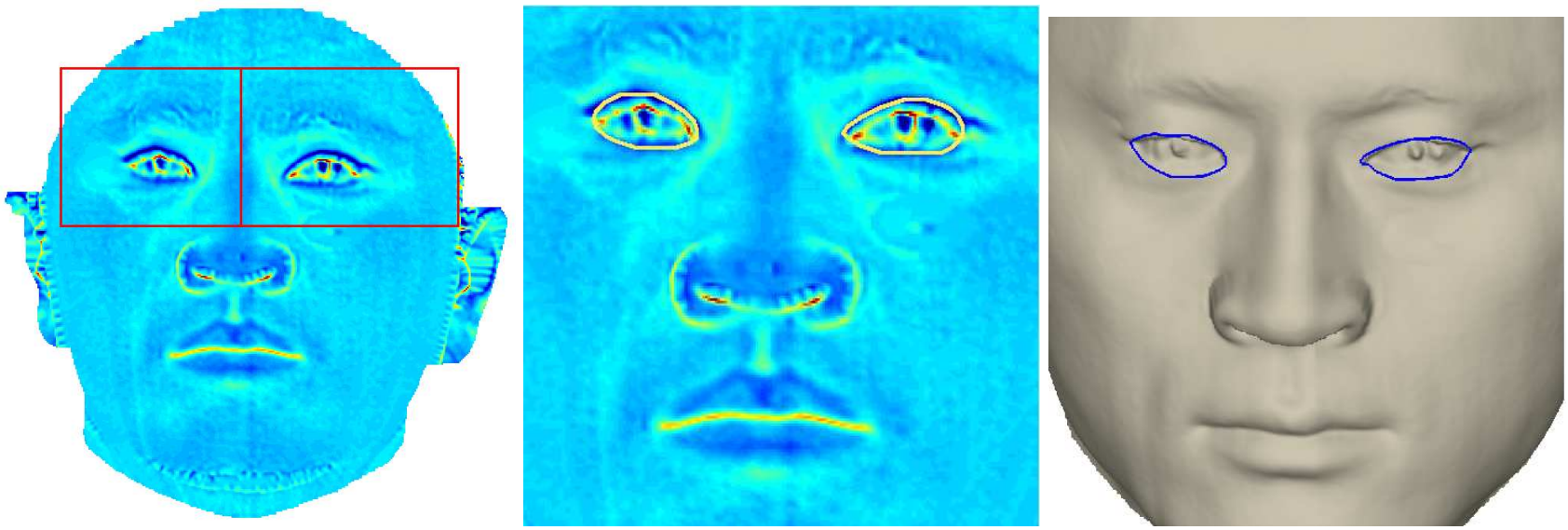
As proposed in Section 5.6, curvature characteristics of the face surface are incorporated into the active contour stopping term. The procedure is as follows: first, a low-level algorithm roughly determines two windows containing the left and the right eye, respectively (Fig. 6.15(a)). To define the detector function f in (6.27), the outliers (1% and 99% percentiles) of the mean curvature image H are cut off. Then the curvature is linearly rescaled. The detector function f for an edge e starting from pixel p is defined as

$$f(p,e) = \frac{1}{2}(\widetilde{H}(p) + \widetilde{H}(p+e)) \,, \tag{6.28}$$

with $\widetilde{H}$ the rescaled mean curvature. Note, that the expression $p+e$ is well-defined, since vertices and edges in a 2D grid graph can be identified with vectors in $\mathbb{R}^2$.

Eye contours resemble ellipses rather than circles and may possess corners. Assuming that the eyes are roughly aligned horizontally, the following anisotropic weighting function is defined to penalise vertical edges and corners: $w_1(\nu) = |\nu_2| \cdot 0.8 + 0.2$ and $w_2(\nu) = |\nu_1| \cdot 0.8 + 0.2$. Note, that $\nu = (\nu_1, \nu_2)$ means the edge normal in w_2 and the corner normal in w_1 (see Fig. 6.9). As a consequence, w_1 and w_2 favour corners and edges in horizontal direction, respectively. An analogous prior for other directions in the plane can be obtained by adapting w_1 and w_2 accordingly.

The experiments on face data emphasise that pseudo-elastica are quite robust with respect to the choice of the model parameters. As in [KDG09], $c = 5$ is chosen across all datasets for processing a database with 50 faces. The average processing time is about 36 seconds for both eyes. See Chapter 7 for several sample results and a quantitative evaluation.



(a) (b) (c)

Figure 6.15: Application of anisotropic pseudo-elastica to fully automatic eye contour segmentation from 3D range images. Mean curvature image with search windows (a), computed pseudo-optimal contours (b), visualisation on face surface (c).

6.7 Discussion and Conclusions

This chapter has developed a novel framework for 2D image segmentation using pseudo-elastica. Pseudo-elastica are open or closed curves that approximate global minimisers of certain second-order energies. Two new segmentation energies penalising curvature were introduced, that allow for an efficient and flexible optimisation in a graph-based setting. The central theoretical result of this chapter proves that non-trivial global minimisers exist for the proposed energies. A novel framework was developed to efficiently approximate these minimisers using a bidirectional Dijkstra-like search. The core algorithm has the complexity[1] $O(N \log N)$, which is significantly less than existing globally optimal segmentation methods [SU88, SC07, WSC09] incorporating curvature regularity.

The core algorithm was integrated into both user-guided and fully unsupervised image segmentation schemes. Results on various images, including noisy ultrasound images, suggest that the user-guided variant scores favourably against state-of-the-art algorithms [BK03, CK97]. Equally good or better results were obtained with significantly less user input. Due to typical processing times of a few seconds, the presented method constitutes a feasible alternative to first-order shortest path approaches such as [CK97]. The automatic scheme is primarily suited for the segmentation of roughly round, or elliptic objects under difficult conditions, i.e. strong noise and large gaps in the boundary. Anisotropy can be included to allow for corners in certain directions.

Main drawback of the pseudo-elastica compared to other methods such as [CK97, JI01, BK03, SC07] is that the computed contours are not the exact energy minimisers. Heuristics are applied in order to obtain an efficient scheme for practical use, and some parameters inside the algorithm have to be set. Yet the algorithm showed robustness with respect to the choice of the parameters, since one set of values, detailed in Appendix 6.A.6, could be used across all experiments presented in the thesis. The tests detailed in this chapter suggest that the pseudo-elastica, despite the use of heuristics, provide a satisfactory approximation of the exact elastica qualitatively, and that the quantitative accuracy of the method in image segmentation applications compares well with state-of-the-art methods [BK03, CK97].

Like other segmentation algorithms based on shortest path techniques (e.g. [JI01, SC07]), pseudo-elastica cannot detect multiply-connected regions. This distinguishes the path-based

[1] N denotes the number of image pixels.

methods from the implicit methods, such as level sets [OS88] and graph cuts [BVZ01]. Moreover, there is no obvious extension of the method to 3D volumetric images. Yet Ardon and Cohen [AC06] have applied shortest path techniques to 3D surface segmentation. Studying, whether pseudo-elastica can be embedded into a similar scheme, thus providing curvature regularity for surface segmentation, is an exciting avenue for future research.

Conclusions

1. *Classic shortest path based AC algorithms can be generalised to second-order energies, while the computational complexity is maintained. However, heuristics have to be applied at the expense of exactness;*

2. *A user-guided and an automatic segmentation scheme can be derived from the core pseudo-elastica algorithm;*

3. *In comparison to AC techniques based on classic shortest paths, the second-order model provides increased robustness in the presence of noise. In addition, significantly less input is required from the user;*

4. *Anisotropic energies can readily be incorporated into the proposed energies and optimisation scheme. Corners in the contour can be allowed, and expansion of the contour into certain directions can be favoured. Thus, the shape prior imposed by the second-order energies can be adapted to the respective application.*

6.A Appendix

6.A.1 Some Properties of Elastica

As stated in Section 6.2.2, elastica are minimisers of the second-order energy:

$$E_{el}(\Gamma) = \int_{\Gamma} (\alpha\kappa^2 + \beta)\, ds\,, \tag{6.29}$$

where α and β are positive constants. Classically, elastica have been studied as open curves connecting two fixed endpoints. In this setting, minimisers can be computed explicitly using elliptic functions [Mum94].

For closed curves the following theorem was proven by Wen [Wen93]:

Theorem 15 (Wen, 1993)**.** *The unique minimisers of (6.29) are the circles of radius* $r = \sqrt{\frac{\alpha}{\beta}}$.

In order to illustrate this theorem, consider the problem of computing the circle with a fixed centre that minimises functional E_{el}. Let C_r be the circle with radius r and centre x_0. Then

$$E_{el}(C_r) = \int_{C_r} \left(\frac{\alpha}{r^2} + \beta\right) ds = 2\pi \left(\frac{\alpha}{r} + \beta r\right)\,. \tag{6.30}$$

To minimise (6.30) the zeros of the derivative have to be computed. The derivative is

$$\partial_r E_{el}(C_r) = 2\pi \left(\frac{-\alpha}{r^2} + \beta\right)\,, \tag{6.31}$$

It is easily checked that $r_{min} = \sqrt{\frac{\alpha}{\beta}}$ is the only non-negative zero of $\partial_r E_{el}(C_r)$ and that the second derivative of $E_{el}(C_r)$ is positive in r_{min}. Thus, the energy E_{el} has a unique global minimum among the considered circles.

To understand that minimisers of E_{el} are indeed circles, consider the respective Euler-Lagrange equations (see [Mum94])

$$2\alpha\partial_{ss}\kappa + \alpha\kappa^3 - \beta\kappa = 0\,. \tag{6.32}$$

The gradient descent equation for E_{el} is readily derived:

$$\partial_t \Gamma = 2\alpha\partial_{ss}\kappa + \alpha\kappa^3 - \beta\kappa\,. \tag{6.33}$$

When evolving, the $\partial_{ss}\kappa$-term in (6.33) pushes the curve towards circular forms: where the curvature is maximal it becomes negative and pushes the curve inwards. The opposite happens in sections of minimal curvature.

6.A.2 Proof of Proposition 13

Proof. The proof is given for estimate (6.13); the estimate for E_2 can be obtained analogously. It follows from the definition of E_1 and the minimality of Γ_- that

$$\int_{\Gamma_-} f \, ds \leq E_1(\Gamma_-) \leq E_1(\Gamma_0) \, . \tag{6.34}$$

As $\int_{\Gamma_-} f \, ds \geq \|\Gamma_-\| \cdot f_-$, the right-hand inequality of (6.13) is readily obtained. An L^2-bound of the curvature is required for the left-hand side. From

$$\int_{\Gamma_-} f c\kappa^2 \, ds \leq E_1(\Gamma_-) \leq E_1(\Gamma_0) \tag{6.35}$$

one concludes that

$$\int_{\Gamma_-} \kappa^2 ds \leq \frac{E_1(\Gamma_0)}{c f_-} \, . \tag{6.36}$$

Holder's inequality yields the estimate

$$\left(\int_{\Gamma_-} |\kappa| \right)^2 \leq \|\Gamma_-\| \int_{\Gamma_-} \kappa^2 \, ds \overset{(6.36)}{\leq} \|\Gamma_-\| \frac{E_1(\Gamma_0)}{c f_-} \, . \tag{6.37}$$

The leftmost expression in (6.37) can be estimated by the Fenchel-Fary inequality for closed curves:

$$\int_{\Gamma} |\kappa| ds \geq 2\pi \, . \tag{6.38}$$

Plugging (6.38) in (6.37) yields the lower bound for $\|\Gamma_-\|$. □

6.A.3 Proof of Theorem 14

In order to parametrise closed curves by Sobolev functions, *periodic* Sobolev spaces need to be defined.

Definition 10. *Let $J = [0,1]$ be the unit interval, and $C_p^\infty(J) := \{u \in C^\infty(\mathbb{R}) : \forall x : u(x) = u(x+1)\}$ the space of smooth 1-periodic functions. Then the periodic Sobolev Space $H_p^m(J)$ is defined as the closure of $C_p^\infty(J)$ with respect to the H^m-norm*

$$\|u\|_{H^m} = \sum_{0 \leq k \leq m} \|u^{(k)}\|_{L^2} \, , \tag{6.39}$$

where $u^{(k)}$ is the k-th derivative of u, and the L^2 norm is defined as

$$\|u^{(k)}\|_{L^2} = \left(\int_J |u^{(k)}(r)|^2 \, dr \right)^{\frac{1}{2}} \, . \tag{6.40}$$

Note, that the periodic Sobolev spaces H_p^m inherit most of the properties of the standard Sobolev spaces: for instance, they are Hilbert spaces and thus reflexive (see e.g. [Kre99, Ada75]).

Theorem 14 *Let $\Omega \subset \mathbb{R}^2$ be a rectangular domain, $f : \Omega \to \mathbb{R}^+$ be a continuous function and $c > 0$ constant. Then the energy*

$$E_1(\Gamma) = \int_\Gamma f(s)\left(c\kappa^2 + 1\right) ds \tag{6.41}$$

has a global minimiser $\Gamma_{min} \in H_p^2(J,\Omega)$ with $J = [0,1]$ the unit interval. The minimiser has a strictly positive arc length, i.e. $\|\Gamma_{min}\| > 0$.

Remarks:

1. For the sake of simplicity, the functional stemming from the energy (6.11) is studied here. The proof works analogously for the energy (6.12);

2. A similar result for the classical elastica functional has been proven in [BdMP93]. Yet these authors study the functional incorporating boundaries of bounded sets instead of curves. In this general case the proof is much more intricate, and only weak regularity results are obtained for the minimiser;

3. The proof for certain open curves in the case of the classical elastica functional is indicated succinctly in [BM04];

4. To the best of the author's knowledge, the proof for closed curves given here is new and cannot be found elsewhere in the literature.

Proof. Following the classical procedure of the direct method of the calculus of variations (e.g. [Dac89]), the existence of a global minimiser is proven in two steps:

1. Show that minimising sequences of E_1 in $H_p^2(J,\Omega)$ contain a subsequence converging weakly towards some element Γ_{min} of $H_p^2(J,\Omega)$;

2. Show weak lower semicontinuity of E_1 in $H_p^2(J,\Omega)$.

As an intermediate step 1a), further properties of the limit element Γ_{min} are derived.

Proof of Step 1)

The strict positivity of the integrand of E_1 implies

$$L := \inf_{\Gamma \in H_p^2(J,\Omega)} E_1(\Gamma) \geq 0. \tag{6.42}$$

Now, choose a minimising sequence $\Gamma_k \in H_p^2(J,\Omega)$ with $E_1(\Gamma_k) \overset{k\to\infty}{\to} L$. It is well known from functional analysis (e.g. [Kre99]) that Sobolev spaces H^k are reflexive, and hence, closed and

bounded sets are weakly compact. The estimate

$$\|\Gamma_k\|_{H^2} \leq C \tag{6.43}$$

has to be established for some constant $C > 0$, in order to conclude the existence of a weakly convergent subsequence.

Without loss of generality, the Γ_k are assumed to be parametrised with constant velocity i.e. $|\partial_r \Gamma_k| = c_k$. Note, that there is an element Γ_0 in the minimising sequence so that

$$\forall k : E_1(\Gamma_k) \leq E_1(\Gamma_0)\,. \tag{6.44}$$

Therefore, the arc length $\|\Gamma_k\|$ can be uniformly bounded by an argument like in Proposition 13. This implies

$$\exists c_0, C_0 > 0 \; \forall k : c_0 \leq \|\Gamma_k\| = c_k \leq C_0. \tag{6.45}$$

The following bound for the first derivatives of the Γ_k is now readily derived:

$$\begin{aligned} \|\partial_r \Gamma_k\|_{L^2}^2 &= \int_J |\partial_r \Gamma_k|^2 \, dr \\ &= \int_J c_k^2 \, dr = c_k^2 \overset{(6.45)}{\leq} C_0^2\,. \end{aligned} \tag{6.46}$$

Note, that r is the variable in the parametrisation interval. Consider the second derivative: due to Proposition 2 in Chapter 2, arc length derivatives can be transformed to parameter derivatives by

$$\partial_{ss} = |\partial_r \Gamma_k|^{-2} \partial_{rr}\,, \tag{6.47}$$

as $|\partial_r \Gamma_k|$ is constant for each k.

Since the absolute curvature is the length of the curvature vector, one derives

$$E_1(\Gamma_k) = \int_{\Gamma_k} f(s) \left(c|\partial_{ss}\Gamma_k|^2 + 1 \right) ds \tag{6.48}$$

$$\overset{(6.47)}{=} \int_J f(\Gamma_k) \left(c \frac{|\partial_{rr}\Gamma_k|^2}{|\partial_r \Gamma_k|^2} + 1 \right) |\partial_r \Gamma_k| \, dr \tag{6.49}$$

$$\overset{(6.45)}{\geq} \frac{c f_-}{C_0^2} \|\partial_{rr}\Gamma_k\|_{L^2}^2\,, \tag{6.50}$$

where f_- again is the minimum of f on the domain Ω. The desired bound follows using (6.44):

$$\forall k : \|\partial_{rr}\Gamma_k\|_{L^2} \leq \sqrt{\frac{C_0^2}{c f_-} E_1(\Gamma_0)}\,. \tag{6.51}$$

It remains to show that the Γ_k are uniformly bounded in $L^2(J)$. As a compact subset of $\mathbb{R}^2$ the image domain Ω is contained in some ball $B_R(0)$, and it holds:

$$\forall k : \|\Gamma_k\|_{L^2}^2 = \int_J |\Gamma_k|^2 \, dr \leq R^2 \, . \tag{6.52}$$

Finally, combining (6.46), (6.51) and (6.52) yields

$$\forall k : \|\Gamma_k\|_{H^2} \leq R + C_0 + \sqrt{\frac{C_0^2}{cf_-} E_1(\Gamma_0)} \, . \tag{6.53}$$

Since $H_p^2(J)$ is reflexive, there exists a $\Gamma_{min} \in H_p^2(J)$ and a weakly convergent subsequence k_ν with $\Gamma_{k_\nu} \rightharpoonup \Gamma_{min}$ in $H_p^2(J)$. Further, as all $\Gamma_{k_\nu} \in H_p^2(J, \Omega)$, it can be concluded that $\Gamma_{min} \in H_p^2(J, \Omega)$ (with a similar argument as below for the arc length).

Intermediate Step

For Step 2, the following assertion on the arc length of Γ_{min} is required

$$c_0 \leq |\partial_r \Gamma_{min}| \leq C_0 \, . \tag{6.54}$$

Note, that (6.54) is a pointwise assertion. Since, so far, only weak convergence in a space with integral norm has been established for the minimal sequence, the step to classical function spaces has to be made. It follows from the general Sobolev inequalities [Kre99, Ada75] that $H_p^2(J)$ can be compactly embedded into the Holder space $C_p^{1,\alpha}(J)$ with $0 < \alpha < 1/2$. One concludes that $\Gamma_{k_\nu} \to \Gamma_{min}$ converges strongly in $C_p^{1,\alpha}(J)$. In particular, this implies $\|\partial_r \Gamma_k - \partial_r \Gamma_{min}\|_{C^0} \to 0$ and (6.54). As a consequence, the minimising curve has strictly positive arc length.

Proof of Step 2)

The weak lower semicontinuity of E is necessary in order to prove that Γ_{min} is a minimiser of E. As indicated in [BM04, p. 856], this can be concluded from a general theorem [Dac89, Theorem 3.4, p. 74]. While this theorem is formulated for integral functionals including derivatives up to the first order, it can be generalised to second-order derivatives straightforward. Note the following:

- In Theorem 14, the open domain Ω from the theorem in [Dac89] is the open unit interval. The boundary points 0 and 1 form a Lebesgue null set in $\mathbb{R}$, and thus, they can be removed from the integration domain without changing the integral.

- Substituting $u = \Gamma_k$, $v = \partial_r \Gamma_k$ and $w = \partial_{rr}\Gamma_k$, Eq. (6.49) can be reformulated as

$$E_1(\Gamma_k) = \int_J g(r,u,v,w)\, dr \qquad \text{with} \tag{6.55}$$

$$g(r,u,v,w) = (c\frac{|w|^2}{|v|^2} + 1) f(u)|v| \,. \tag{6.56}$$

  In order for g to be a Caratheodory function, the postulated continuity of f is necessary. Note further, that the domain of v has to be restricted to obtain continuity of g in v. Due to(6.45) and (6.54) the domain of v can be restricted to the set $\mathbb{R}^2 \setminus \overline{B}_{\tilde{c}}(0)$ with some $0 < \tilde{c} < c_0$.

- The main prerequisites in the theorem in [Dac89] are (i) non-negativity of g and (ii) convexity of g in w, i.e. the second derivative. It is easily checked that both of them hold for g in (6.56).

Now, [Dac89, Theorem 3.4] can be applied to the minimal sequence Γ_{k_v}, and the desired weak lower semicontinuity is obtained, i.e.

$$E(\Gamma_{min}) \leq \liminf_{k_v \to \infty} E(\Gamma_{k_v}) = L \,. \tag{6.57}$$

Hence, Γ_{min} is a global minimum of E_1. □

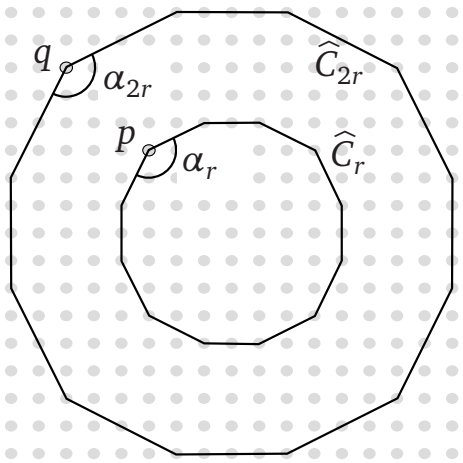


Figure 6.16: Representation of two circle-like contours in a 16-neighbourhood system.

6.A.4 Refining curvature

Large circle-like curves have a small curvature, the circle C_r with radius r being a simple example with $\kappa_{C_r} = \frac{1}{r} \overset{r\to\infty}{\to} 0$. This scaling property of the curvature is an essential prerequisite for the theoretical results in Section 6.3.2, since it causes the curvature L^2 term in (6.11) and (6.12) to decrease, when the curve is scaled up (cf. also (6.30)). Consequently, a lower bound for the arc length of minimisers can be derived (see Proposition 13), which in turn is required for the proof of Theorem 14.

In a graph-based framework, computing the curvature is problematic, as there is a positive lower bound for the angles δ_i between two non-identical directions (see Fig. 2.8(b), p. 29). Therefore, and due to the reasoning in the previous paragraph, the estimation of the curvature in the proposed framework should not be based solely upon the angle between neighbouring edges as in (6.19). Consider, for instance, Fig. 6.16: $\widehat{C}_r$ and $\widehat{C}_{2r}$ are discrete approximations of circles with the respective radii, yet the interior angles of both contours in p and q – denoted by α_r and α_{2r} – as well as the lengths of the adjacent edges are equal. Thus, the computation of the curvature as per (6.19) yields equal curvature values for $\widehat{C}_r$ and $\widehat{C}_{2r}$.

To achieve a more realistic scaling behaviour, the approximation of the curvature in (6.19) is refined as follows (see Fig. 6.1(b)): the algorithm tracks along the minimal path if a straight line is extended over multiple edges and sums up the total length L of this straight segment. If the absolute curvature angle $|\gamma_1|$ does not exceed a certain threshold $p_{\kappa,1}$ (roughly $2 \cdot \frac{2\pi}{|\mathscr{D}|}$), the estimated curvature in p is obtained by dividing the curvature angle γ_1 by L. Otherwise, γ_1 is divided by the length of the incoming edge $e_p = (w, p)$. This procedure can be summarised in

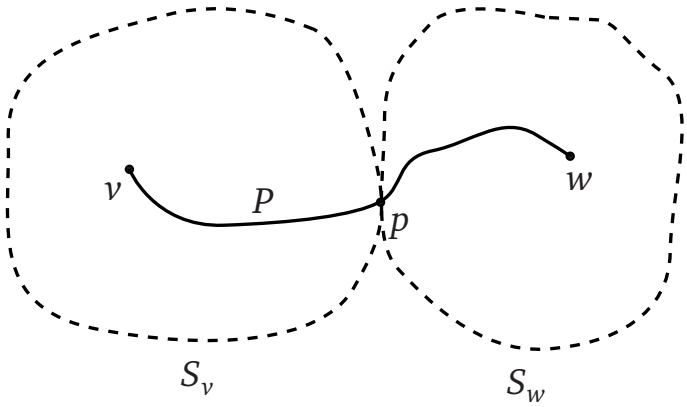


Figure 6.17: Construction of a minimal path P from v to w by simultaneously computing distance functions from v and w. p is the point where the Dijkstra/Fast Marching schemes first intersect. S_v and S_w are spheres of equal distance to v and w, respectively.

the formula

$$\kappa_{\gamma_1} := \frac{C\gamma_1}{l} \tag{6.58}$$

$$\text{with} \quad l = \begin{cases} L & \text{if} |\gamma_1| \leq p_{\kappa,1} \\ \ell(e_p) & \text{if} |\gamma_1| > p_{\kappa,1} \end{cases}, \tag{6.59}$$

where C is a suitable constant (in this thesis: $C = 3\pi$). By this means, the curvature scales as desired for sufficiently smooth contours differing only in their scale (as in Fig. 6.16).

6.A.5 On finding minimal paths

Splitting a minimal path problem into two distinct problems is an intuitive approach, which was arguably pursued first by Pohl [Poh71]. Consider the classical problem of connecting two points $v, w \in \Omega \subset \mathbb{R}^2$ by a weighted minimal path in Ω (Fig. 6.17). Let further d_v and d_w denote the respective weighted distance functions ($d_v(x) := \text{dist}(v, x)$, d_w analogously) and $d_{sum} := d_v + d_w$ the sum of the distances to source and destination. It is a well-known fact from differential geometry (see e.g. [KAB95]) that the set

$$P = \{x \in \Omega : d_{sum}(x) = \min_x d_{sum}\} \tag{6.60}$$

contains exactly those points lying on possible minimal paths from v to w. Given a point $p \in P$, one concludes that there is a minimal path running through p. An important observation is that the desired path from v to w can be obtained by concatenating the partial minimal paths from v to p and from p to w. This follows from a simple argument: assume, there is another

minimal path from v to w through p. Then, changing one branch of the path to the respective partial minimal path yields a shorter path, which is a contradiction.

In practice, when the discrete distance functions d_v, d_w are simultaneously computed using Dijkstra-type algorithms [Dij59] or [Set96], the sets of 'known' points continuously grow (cf. Algorithms 1 and 2). Their boundaries approximate the generalised spheres of constant weighted distance (in Fig. 6.17: S_v and S_w). Figure 6.17 illustrates that one particular point $p \in P$ can be found by computing the first point that is accepted in both branches: in this point the spheres S_v and S_w touch each other.

6.A.6 Remarks on the choice of parameters

The core algorithm presented in Section 6.4.2 includes heuristics in order to achieve acceptable efficiency. This section makes some remarks about the choice of the parameters inside the algorithm – not the model parameter c – and their respective robustness.

Two different kinds of parameters have to be distinguished: the score affecting- and the optimisation parameters.

Score affecting parameters

The following three parameters affect the score value that measures the quality of a concatenated path. As highlighted below, the algorithm shows robustness when varying the parameters in a certain range.

- Curvature parameter $p_{\kappa,1}$ (p. 113) – Choice across the thesis: $p_{\kappa,1} = 0.15\pi = 27°$. Good results can also be obtained for $0.11\pi \leq p_{\kappa,1} \leq 0.2\pi$. The parameter must not be chosen too large, since this allows unwanted corners in the contour, if one arm enclosing the angle is sufficiently long. If it is chosen too small, the curvature estimation discussed in Section 6.4.1 does not function any more, and the algorithm may deliver undesirable results.

- Curvature parameter $p_{\kappa,2}$ (p. 115) – Choice across the thesis: $p_{\kappa,2} = \pi/4 = 45°$. The parameter must not be chosen too large, as this allows corners in the contour where the two branches intersect. Good results can also be obtained for $\pi/8 \leq p_{\kappa,2} \leq \pi/3$.

- Neighbourhood size parameter m (p. 115) – Choice across the thesis: $m = 7$. Good results can also be obtained for a wide range of values from $m = 5$ through $m = 12$. In the tests on constant images (i.e. the classic elastica in Figs. 6.4 and 6.5) the algorithm

showed a certain sensitivity towards the choice of this parameter. On images in practice however, such as the examples shown in Section 6.6, one obtains virtually identical results for different choices of m in the above range.

Optimisation parameters

The two optimisation parameters do not affect the computation of the contour score, but the exhaustiveness of the search for the pseudo-minimal contour. In particular, it is not clear a priori how the energy optimisation parameter p_E impacts on the computational efficiency and the optimality of the results. The choice of p_E might be critical and, therefore, has to be based on empirical tests. See Appendix 6.A.7 for an empirical evaluation of different values of p_E.

- Energy optimisation parameter p_E (p. 118, line 8) – Choice across the thesis: $p_E = 1.5$. In order to understand this parameter consider Fig. 6.3(a) and Algorithm 5. E_{min} is an approximation of the total energy, or generalised distance, between v and w. Letting the two Dijkstra schemes run until they reach their respective goal vertex, one obtains $E_v \approx E_w \approx E_{min}$ and $E_v + E_w \approx 2E_{min}$. Thus, when setting $p_E = 2$, virtually every possible intersection of the two Dijkstra branches is checked. For first order shortest paths, as considered in Fig. 6.17, the optimal path is found as soon as the branches intersect, i.e. $p_E = 1 + \varepsilon$ with a small positive ε is sufficient. In the proposed algorithm, setting p_E to some value between 1 and 2 is a trade-off between optimality and efficiency. The tests in Appendix 6.A.7 showed that in practice choosing $p_E > 1.5$ does not deliver better results, while significantly slowing down the algorithm.

- Score tolerance parameter p_t (p. 118, line 19) – Choice across the thesis: $p_t = 1.3$. Choosing a larger value for p_t does not significantly change the results of the algorithm. If a minimal energy value E_{min} is passed to the algorithm, it has been consolidated before (see Section 6.5.3). Since there may be slight deviations between a score and a consolidated energy value, a path is accepted as a candidate minimal path, even if its score is slightly higher than E_{min}.

6.A.7 Empirical tests for parameter p_E

In order to determine a suitable value for parameter p_E (see Appendix 6.A.6), the automatic algorithm was run on several images with different choices of p_E in the range between 1.5 and 2.0. The computed minimal contour energy as well as the processing time were evaluated. In order to obtain sufficiently robust results, the algorithm was run three times for every configuration and the respective average values were computed.

Images with little and strong noise were considered separately, the results being shown in Figs. 6.18 and 6.19. Apparently, the processing time grows considerably with an increasing value of p_E, while the optimality of the algorithm is not improved.

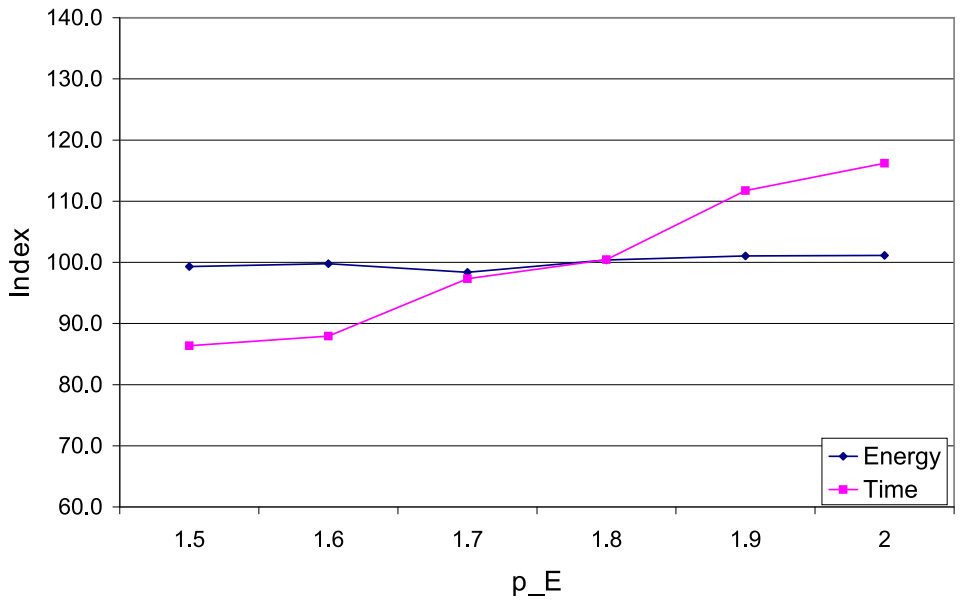


Figure 6.18: Evaluation of efficiency and optimality for different values of p_E. Indices for five sample images with little noise were computed and averaged.

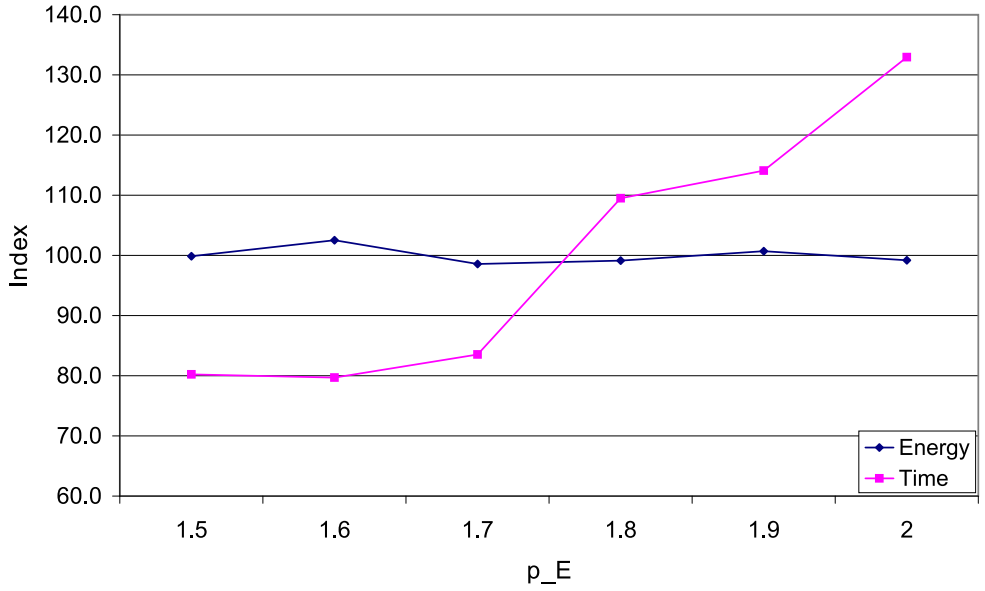


Figure 6.19: Evaluation of efficiency and optimality for different values of p_E. Indices for six sample images with strong noise were computed and averaged.

Chapter 7

Results for Face Feature Segmentation

7.1 Introduction

Chapters 4, 5 and 6 have introduced new techniques that can be applied to the problem of face feature segmentation. Aim of this chapter is to evaluate these techniques quantitatively.

More precisely, the following questions shall be studied:

- How do different techniques for curvature-based contour extraction compare?
- Does incorporating the surface geometry in the GAC evolution process improve the results?
- How do contours extracted by means of curvature characteristics compare to classic texture-based contours in terms of robustness and accuracy?
- Is curvature-based face feature segmentation an actual alternative to texture-based segmentation?

Due to its large range of applications, the lip contour segmentation problem was chosen for a detailed comparative study.

The rest of the chapter is organised as follows: Section 7.2 introduces the data base used for the experiments in this chapter. The comparison methodology is presented in Section 7.3. In Section 7.4, the specifications for the respective algorithms tested in the study are given. Section 7.5 elaborates on quantitative evaluations, as well as a number of sample results for lip and eye contour segmentation. Ultimately in Section 7.6, the outcomes of this chapter are discussed, and conclusions are drawn.

Figure 7.1: Dataset from the BJUT database. From left to right: original face surface as shown in the BJUT viewing software; after conversion to a range map; respective texture image.

7.2 Data Base

The experiments were carried out on 50 randomly chosen datasets of the BJUT 3D face database. This database was acquired with a CyberWare 3030RGB/PS scanner, refer to the Technical Report [bju05] for the technical specifications. The texture data is provided in 24 bit precision as an image of 489×478 points.

7.2.1 Preprocessing

The face data in the BJUT database is available in the form of triangulated surfaces with the corresponding texture information. In order to enable a more intuitive data handling, the surfaces were converted to range image representation using the *gridtrimesh* algorithm [Bri07] explained in Section 5.3. By a simple adaption of this algorithm, rectangular 2D texture images were computed from the texture coordinates. The dimensions of both, range and texture image, were set as follows: the size of the bigger dimension (usually the height) was fixed to 350 pixels, the other dimension was of variable size depending on the aspect ratio. Figure 7.1 illustrates that no essential information was lost during this process.

Conversion to implicit representation

Given a surface in the form of a range image, an implicit representation of the surface can be readily computed. After the conversion to range images the face surfaces were given as:

$$X(u_1,u_2) = \big(u_1,u_2,Z(u_1,u_2)\big)^\top \qquad (u_1,u_2) \in \Omega\,, \tag{7.1}$$

with $\Omega \subset \mathbb{R}^2$ a rectangular domain. With

$$z_- = \min_{(u_1,u_2)\in\Omega} Z(u_1,u_2) \quad \text{and} \quad z_+ = \max_{(u_1,u_2)\in\Omega} Z(u_1,u_2)\,, \tag{7.2}$$

a level set function ψ could then be defined on the volume $\Omega \times [z_-, z_+]$ by

$$\psi(x,y,z) = z - Z(x,y)\,. \tag{7.3}$$

The zero level set of ψ is the surface parametrised by X in (7.1). The function ψ was then transformed to a signed distance function by the fast marching method [Set96] (Section 2.3).

Low-level Preprocessing

Prior to the feature contour segmentation, a low-level preprocessing step computed rough bounding windows for the mouth and the eye region, one window for each eye . A simple algorithm from [KDG09], based only on curvature- and surface characteristics, was applied.

7.3 Comparison Methodology

Two different sets of ground truth contours were prepared for the eyes and lips, where one set was entirely based on surface information and one on texture data.

7.3.1 Curvature-based Ground Truth

Some 3D face databases (e.g. GavabDB [MS04]) do not contain texture data. Therefore, it is one objective of the experiments to compare several different techniques for lip contour segmentation based on curvature data. It was assumed that no texture data is available, and the ground truth contour has to be defined by curvature data only (see Fig. 7.2).

7.3.2 Texture-based Ground Truth

In order to compare curvature-based with texture-based methods, a second set of ground truth contours was manually delineated on the basis of the texture images (Fig. 7.3).

7.3.3 Error Statistics

Subsequently, the ground truth contours were used to obtain error statistics for the obtained feature contours. For each contour, the mean and maximum distance to the ground truth contour, as well as the standard deviation of the distance, was calculated. In the next step, the statistics of the error distribution over the test set were computed. In order to avoid a skew in the results, the mean error values were cut off at the 98% percentile before computing the average mean error (cf. Table 7.1).

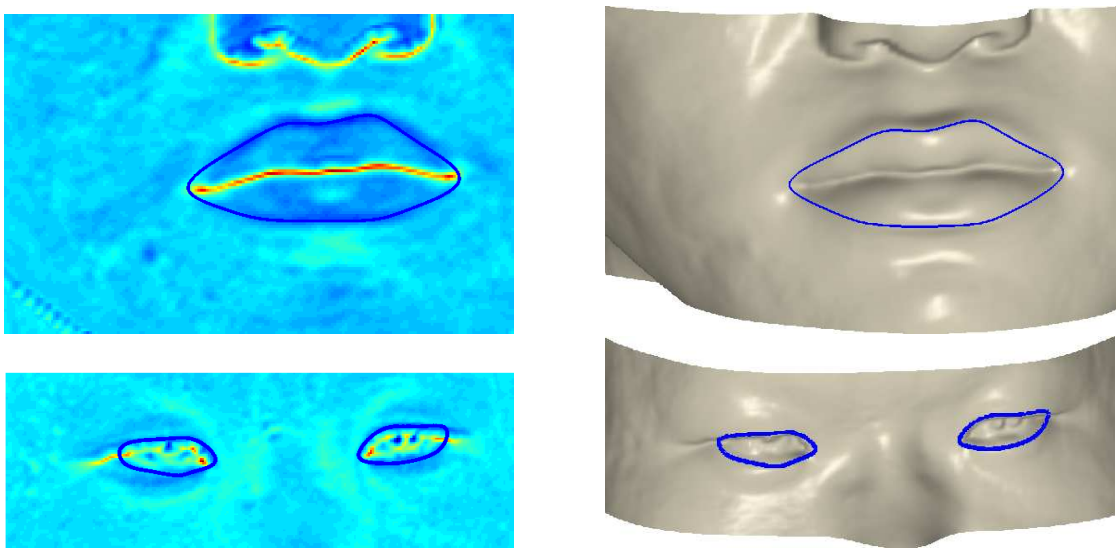


Figure 7.2: Manually drawn ground truth based on curvature information: on curvature image (left), visualised on face surface (right).

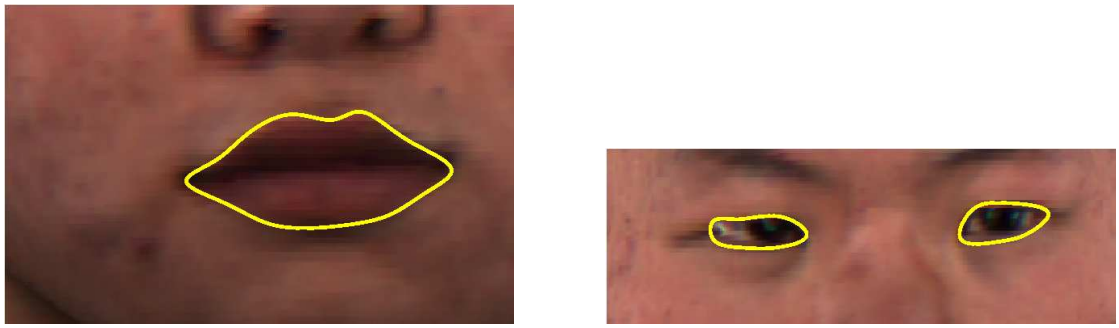


Figure 7.3: Manually drawn ground truth based on texture information.

7.4 Algorithm Specifications

7.4.1 Lip Contours

In the following, each tested algorithm is stated together with its specifications.

Curvature-based Contours

GAC-like Algorithms The following variants of the GAC were tested:

- Standard 2D GACs [CKS97] (Section 5.5.1), with the evolution equation:

$$\partial_t \Gamma = \left(f(\mu\kappa + c) - \langle \nabla f, \nu \rangle \right) \nu \; ; \tag{7.4}$$

- 3D GACs as introduced in Chapter 4, with the adapted evolution equation on surfaces:

$$\partial_t \Gamma = \Big(f(\mu\kappa_g + c) - \langle \nabla_M f, \hat{v} \rangle \Big) \hat{v} \; . \tag{7.5}$$

  Here, κ_g is the geodesic curvature, ∇_M is the surface gradient and $\hat{v}$ is the co-normal;

- Spira's surface GACs [SK07] (plus a balloon force) as per flow (7.5), yet implemented using the parametric surface representation.

Due to the apparent similarity of the three segmentation models, equal parameters could be set for all of them: empirical tests suggested the choice of $\mu = 0.2$ for the curvature term coefficient, and $c = 0.025$ for the balloon term coefficient.

The surface data was smoothed by a 5×5 Gaussian filter, before the mean curvature H was computed. Then the outliers of the curvature spectrum were cut at the 1% and 98% percentiles. The stopping term for the three GAC variants was computed as proposed in (5.11):

$$f = \frac{0.8}{\pi} \cdot \arccos\left(-\frac{\tilde{H} - \tilde{H}_{max}/2}{\tilde{H}_{max}/2} \right) + 0.2 \, , \tag{7.6}$$

where $\widetilde{H}$ was obtained by scaling H to the interval $[0, \widetilde{H}_{max}]$ (with $\widetilde{H}_{max} = 2.5$), refer to Section 5.5.1 for details.

Globally Optimal AC Algorithm (ACG) [CK97] This algorithm was implemented as described in Section 5.5.2 respectively [KDG09]. Cohen and Kimmel's method [CK97] for computing globally optimal ACs based on the following Eikonal equation was used (see Section 2.3):

$$|\nabla U| = P \; . \tag{7.7}$$

Here, P is a suitably chosen potential term, see Section 5.5.2 for details and the empirically chosen parameters. Again, one set of parameters was used for all datasets. Better results for this approach were obtained when the surface was smoothed by a 7×7 Gaussian filter, and the curvature spectrum was cut at the 1% and 99% percentiles.

Texture-based Contours

Classic ACs [KWT88] The classic active contours [KWT88] (cf. Section 3.1.1) were applied successfully to lip segmentation and tracking in the past, see e.g. [LDC+99, DEL02, ECC04]. Therefore, this technique was chosen to compute texture-based contours on the 50 test datasets.

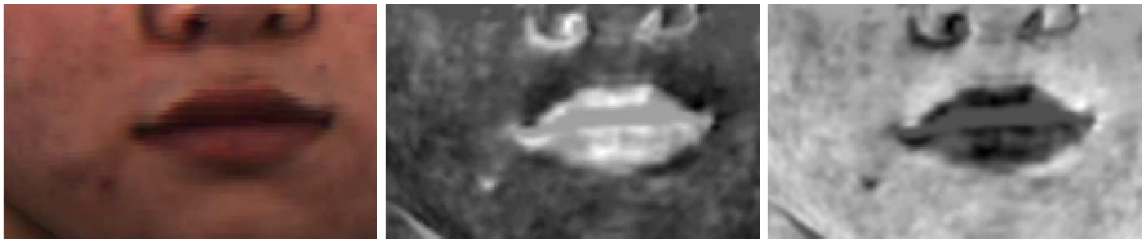


Figure 7.4: Mouth region, from left to right: original texture image; preprocessed pseudo hue image; logarithmic hue image.

ACs are parametrised curves evolving by the equation

$$\partial_t \Gamma = \alpha \partial_{rr} \Gamma - \beta \partial_{rrrr} \Gamma - \omega_P \nabla P(\Gamma)\,, \tag{7.8}$$

where α and β are the Tikhonov parameters, and ω_P is an additional weighting term for the potential term ∇P. For simplicity, a semi-automatic system was implemented: first, mouth corners were defined manually by the user. Then, two open snakes, one for each of the lip contours, were evolved until convergence.

The following preprocessing steps were carried out on the texture image:

- Cut at the 5% and 95% percentiles of each of the R, G, and B channels to discard outliers;
- Computation of the pseudo hue (e.g. [ECC04]) $h_p = R/(R+G)$ and logarithmic hue ([LDC+99]) $h_l = 256 \cdot G/R$;
- Smoothing by a 5×5 Gaussian filter;
- Normalisation to pixel values in $[1, 100]$.

The hue images h_p and h_l provide a better discrimination of the lips from the skin than, for instance, the respective greyscale transformed image. Figure 7.5 shows a sample result of the AC on a pseudo hue image. The classic AC scheme without a balloon force was used; the Tikhonov parameters were empirically set to $\alpha = \beta = \frac{1}{2}$ across all datasets. Furthermore, the potential term was weighted with $\omega_P = -\frac{1}{20}$, and the time step was set to $\Delta t = 0.15$.

Multimodal Approach

Adapted Globally Optimal AC Scheme (ACGA) A simple modification of the ACG algorithm was tested. The same manually set, texture-based mouth corners as for the classic ACs

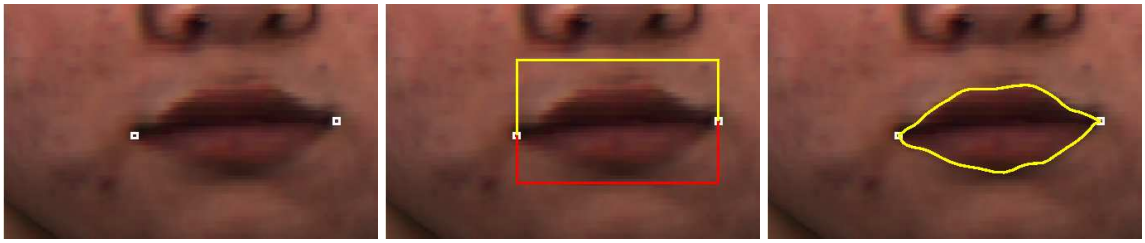


Figure 7.5: Example for snake segmentation of 2D lip contours, from left to right: manually fixed mouth corners; initial contours; contours after convergence.

(see above) were employed, instead of mouth corners derived from curvature data (as in ACG, see Section 5.5.2). Thus, a multimodal approach based on both, texture- and curvature data, was simulated. In the following, this scheme is named *adapted global AC (ACGA)*.

7.4.2 Eye Contours

The pseudo-optimal second-order AC framework developed in Chapter 6 was applied to 3D eye contour extraction (see Section 6.6.2). The obtained contours are approximate global minimisers of the anisotropic second-order energy

$$E_{2,a}(\Gamma) = \int_\Gamma \left(c \cdot (w_1(\nu_c) \cdot \kappa)^2 + w_2(\nu_e) \cdot f(s) \right) \, ds \,, \tag{7.9}$$

where κ is the curvature, w_1 and w_2 are the anisotropic weighting functions (see Section 6.5.4), and ν_c and ν_e are the corner normal and the edge normal, respectively (see Fig. 6.9). To model the elliptic shape of the eyes and the eye corners, the weighting functions were set to $w_1(\nu) = |\nu_2| \cdot 0.8 + 0.2$ and $w_2(\nu) = |\nu_1| \cdot 0.8 + 0.2$, where ν_1 and ν_2 are the coordinates of the 2D vector $\nu = (\nu_1, \nu_2)^\top$.

The model parameter was set to $c = 5$ across all datasets. The surface was smoothed by a 7×7 Gaussian filter, before H was computed using (5.6). Subsequently, H was linearly rescaled to the interval $[0.01, 2]$. Since this method is graph-based, the feature detector function f in (7.9) has to be defined for edges rather than for pixels. For an edge e emanating from a pixel p, f was set to $f(p,e) = \frac{1}{2}(\widetilde{H}(p) + \widetilde{H}(p+e))$, where $\widetilde{H}$ denotes the rescaled mean curvature, and $p + e$ is understood in the sense of vector arithmetics in $\mathbb{R}^2$. Then, the automatic scheme explained in Section 6.6.2 computed an approximate globally optimal minimiser of (7.9).

7.4.3 Implementation Details

All experiments were run on an Intel Core 2 Duo 2.5 GHz machine. The GAC schemes, i.e. the standard GAC [CKS97], Spira and Kimmel's GAC on surfaces [SK07] and the 3D GAC proposed in Chapter 4 of this thesis, were programmed in MATLAB. The standard 2D GAC and Spira's GAC were coded by the author as described in the respective papers. The classic AC was implemented using a free MATLAB toolbox [Li05]. Further, the fast marching algorithm at the core of both the ACG and the ACGA scheme was implemented using [Pey08]. While the latter comes as a MATLAB toolbox, the main routines are precompiled in C++.

It has to be emphasised that the main focus of the study is rather on accuracy evaluation than on the processing times. Being obtained by uncompiled MATLAB code, the processing times for the three different GAC schemes, for instance, are certainly neither optimal nor exactly comparable to these of the schemes based on fast marching. Note that for 2D standard GAC there are optimised algorithms, e.g. [PM04], that are at least one order of magnitude faster than the straightforward implementation used in this chapter.

7.5 Experimental Results

7.5.1 Lip Contours

Curvature-based Contours

The quantitative results in Table 7.1 and Fig. 7.7 show that the proposed 3D GAC performed slightly better than the other GAC versions. This suggests that incorporation of the surface geometry has a positive effect on the segmentation accuracy, yet the results do not improve drastically. Further, the experiments confirmed that it is problematic to include a balloon term into Spira's model for parametric surfaces [SK07].

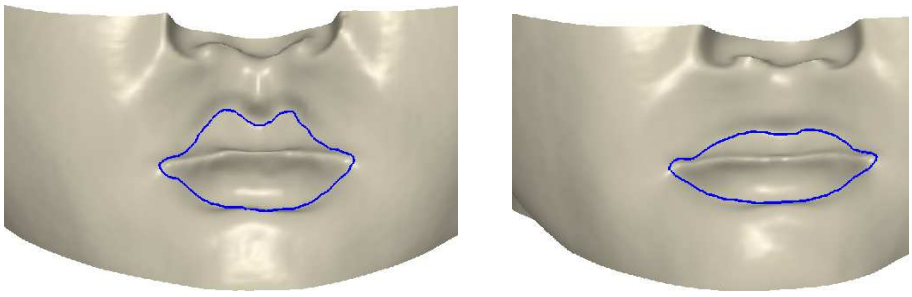


Figure 7.6: Examples for successful lip contour segmentation using 3D GAC.

For some datasets the contour was trapped in spurious local minima, suggesting the balloon coefficient was chosen too low. However, setting a higher balloon force caused the contour to pass the desired ridges. Overall, Spira's algorithm even performed worse than the standard 2D GAC on curvature images.

Method	Mean	Max.	Std.	Mean Error	Processing
	Error (in pixels)			median (in pixels)	time (s)
2D GAC	0.93 ± 0.36	3.9 ± 2.2	0.9 ± 0.5	0.8	150 ± 51
Spira's GAC	0.94 ± 0.39	4.0 ± 2.7	0.9 ± 0.5	0.8	226 ± 84
3D GAC	0.86 ± 0.29	3.7 ± 1.7	0.8 ± 0.4	0.8	220 ± 74
ACG	0.96 ± 0.41	3.6 ± 2.3	0.9 ± 0.8	0.8	0.08 ± 0.01

Table 7.1: Accuracy evaluation for lip contours compared to curvature-based ground truth.

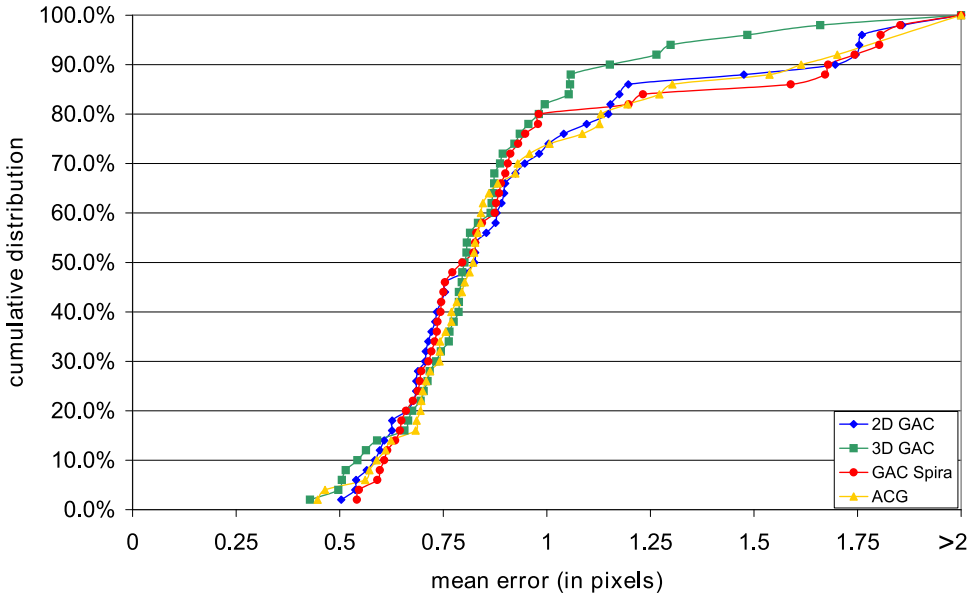


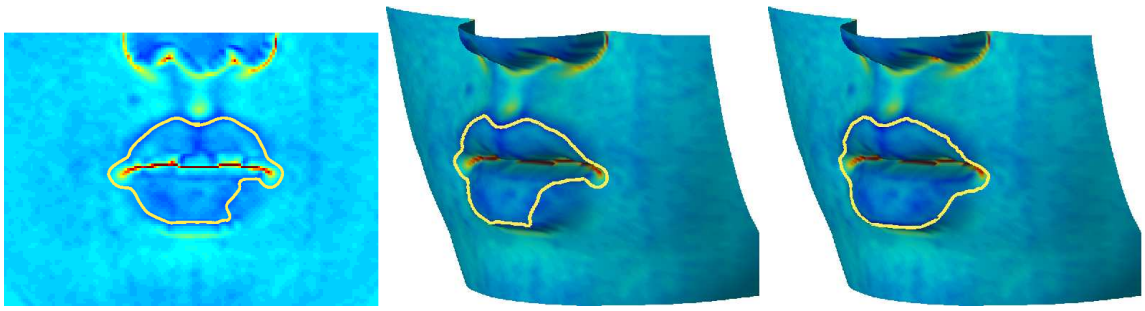
Figure 7.7: Evaluation of lip contour segmentation accuracy with respect to manually delineated ground truth based on curvature data.

Influence of the surface geometry Figure 7.8 shows an example where only the 3D GAC converged properly (Fig. 7.8(c)). Interestingly, the contours behaved differently just at a location where the surface becomes quite steep, and thus the projected, *flat* geometry differs significantly from the actual surface geometry (cf. Fig. 4.13, p. 70). Consequently, the balloon force in the 2D GAC was too high, causing the contour to pass the desired ridge (Fig. 7.8(a)). A similar effect occured for Spira's GAC (Fig. 7.8(b)), caused by the problematic scaling behaviour of the balloon term, that was discussed at length in Section 4.3.3.

As stated previously in the chapter the model parameters were optimised empirically for each of the GAC schemes individually. Yet it showed that identical parameters for the balloon and the curvature term ($\mu = 0.2$ and $c = 0.025$, see p. 157) delivered the best overall results for all three models.

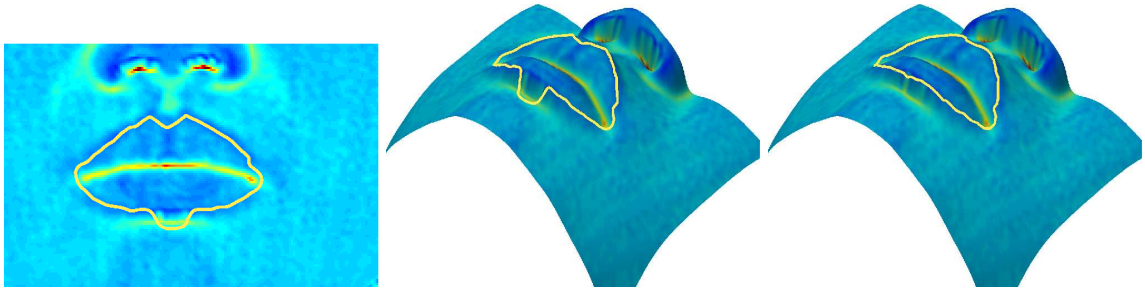
A further example, where incorporating the surface geometry improved the segmentation result, is shown in Fig. 7.9. Again, the problem for standard GAC and Spira's GAC occured at a spot with large surface slope, respectively on the verge of it: just under the lower lip contour. The standard GAC became trapped before a minor ridge of positive mean curvature (Fig. 7.9(a)), as did Spira's GAC (Fig. 7.9(b)). By contrast, the 3D GAC delivered the desired result (Fig. 7.9(c)).

As the 3D GAC usually yielded similar or better results than the 2D- and Spira's GAC, it stands representatively for the GAC type algorithms in the following comparative considerations.



(a) 2D GAC (b) Spira's GAC (c) 3D GAC

Figure 7.8: Results of different GAC approaches.



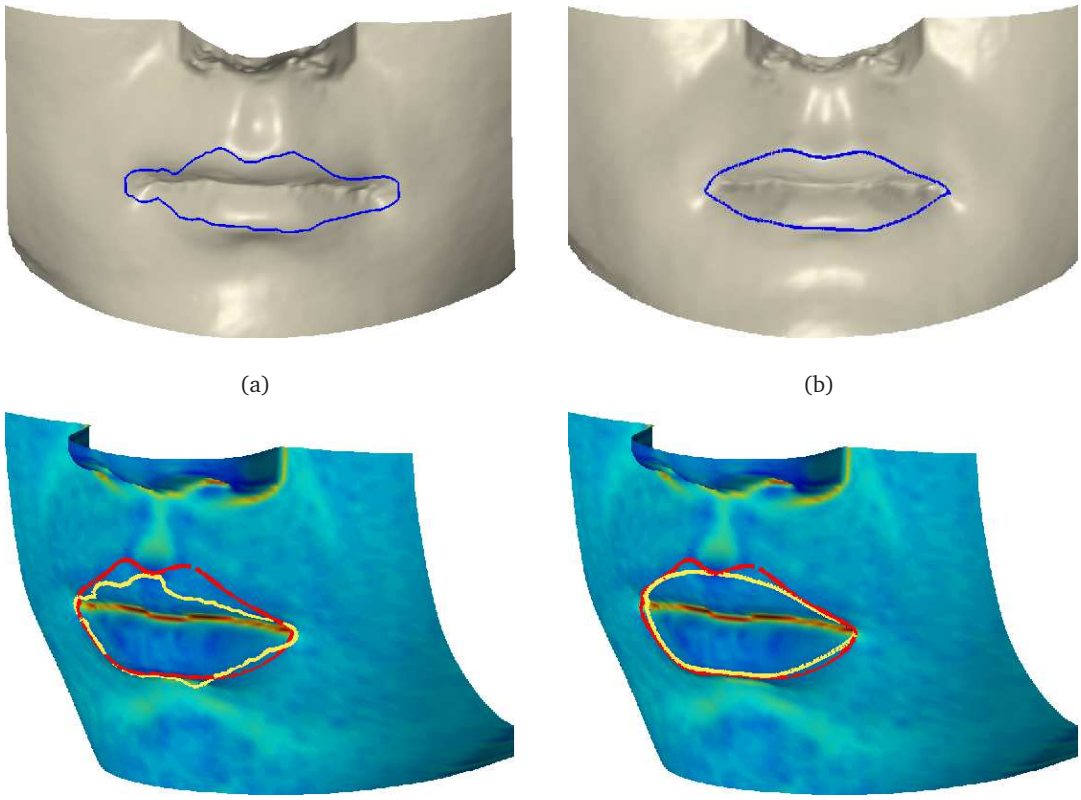
(a) 2D GAC (b) Spira's GAC (c) 3D GAC

Figure 7.9: Results of different GAC approaches.

Failure of curvature-based approaches Figure 7.10 displays examples where all GAC-like methods failed to produce a satisfactory result. One possible reason was low data quality, e.g. scanning artefacts, causing the evolution-based GACs to get trapped in spurious local minima (Fig. 7.10(a)). Not surprisingly, the ACG scheme performed better in such cases (Fig. 7.10(b)).

In Fig. 7.10(c), the 3D GAC passed the desired lip contour due to indistinct ridges. A better, but still quite unsatisfactory result (Fig. 7.10(d)) was achieved by the ACG algorithm, since the parabolic shape prior (Section 5.5.2) ensures a rough resemblance to lip contours even in the absence of external data.

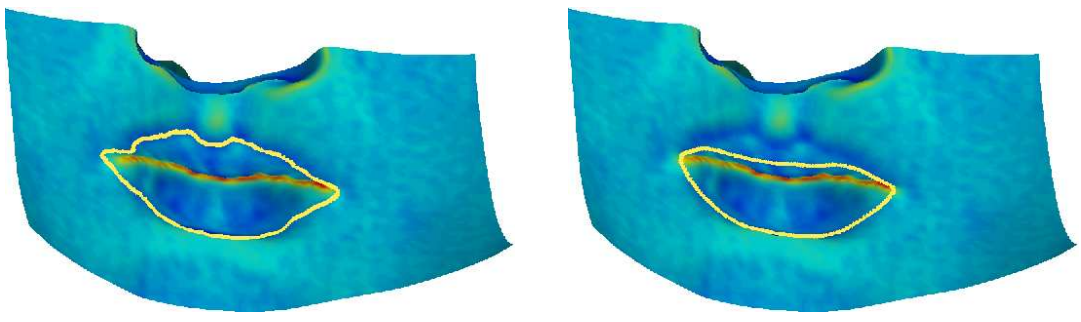
Figure 7.11 illustrates under what circumstances the evolution-based GACs performed better than ACG. If there were areas of distinctive negative mean curvature inside the actual lip ridges, the globally optimal ACs tended to pick a 'short cut' connection between the mouth corners, instead of following the lip contour (Fig. 7.10(b)). Here, the local minimum found by the 3D GAC (Fig. 7.10(a)) was more desirable than the global minimum.



(a) (b)

(c) (d)

Figure 7.10: Sample datasets where all GAC methods produced unsatisfactory results: contours obtained with the 3D GACs (left), respective contours obtained by the ACG scheme (right). The red contour in the bottom row represents the curvature-based ground truth contour.



(a) (b)

Figure 7.11: Sample dataset where the GAC methods (here: 3D GAC, left) succeeded, while the ACG scheme (right) failed.

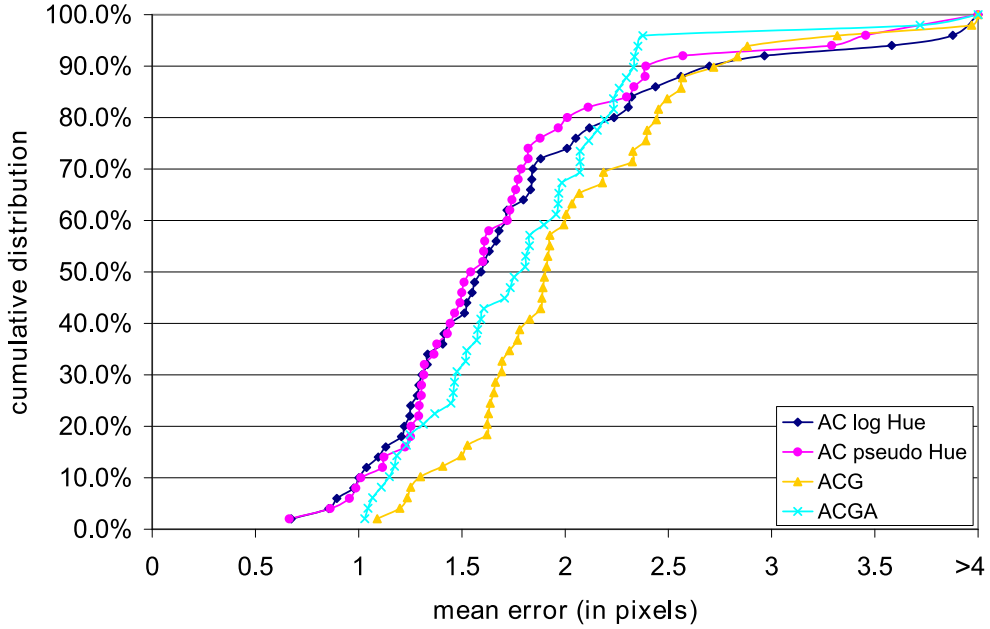
Texture-based Contours

The methods evaluated in the preceding section can be applied to 3D face features segmentation even if no texture data is available. Yet the reported results suggest that, due to its robustness and relative independence of pose and illumination, curvature-based contour segmentation might be an alternative to texture-based segmentation.

The contours computed from curvature characteristics, as well as the classic AC described in Section 7.4.1, were compared to the texture-based ground truth and error statistics were calculated (Table 7.2). The statistics suggest that overall the texture-based ACs were closest to the ground truth contours, with the results on the pseudo hue images being the best. This is not surprising, keeping in mind that the ground truth was defined on the texture image, and all other contours in the study were computed by curvature characteristics. Yet the ACGA scheme performs only slightly worse than the texture-based ACs. In fact, the more detailed evaluations in Fig. 7.12 show an interesting effect: the cumulative distribution curve of the pseudo hue AC exhibits a uniform, steep slope until about the 75% mark, then it becomes noticeably flatter, indicating quickly deteriorating accuracy results. The cumulative distribution curve of the ACGA algorithm looks similar, yet it flattens only at the

Method	Mean	Max.	Std.	Mean Error
	Error (in pixels)			median (in pixels)
2D GAC	2.2 ± 0.6	6.4 ± 2.3	1.6 ± 0.6	2.1
Spira's GAC	2.2 ± 0.5	6.5 ± 2.6	1.6 ± 0.6	2.1
3D GAC	2.0 ± 0.5	5.9 ± 1.5	1.5 ± 0.4	2.0
ACG	2.1 ± 0.6	6.0 ± 1.7	1.6 ± 0.5	1.9
ACGA	1.8 ± 0.6	5.3 ± 1.6	1.4 ± 0.5	1.8
classic AC (pseudo hue)	1.7 ± 0.7	5.7 ± 2.4	1.5 ± 0.8	1.6
classic AC (log. hue)	1.8 ± 0.8	5.8 ± 2.5	1.6 ± 0.9	1.6

Table 7.2: Accuracy evaluation for lip contour segmentation compared to texture-based ground truth.



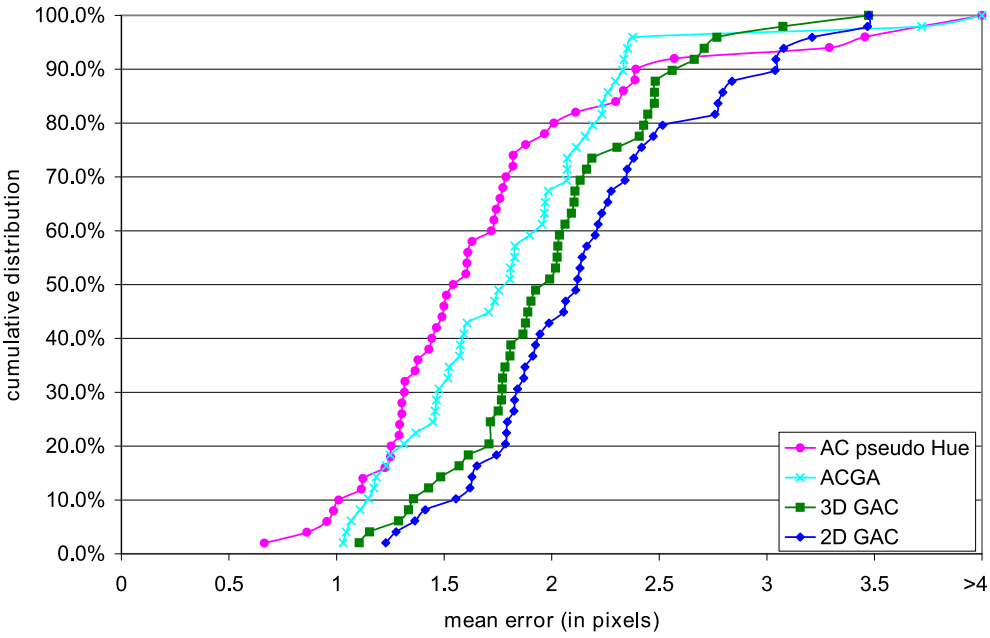


Figure 7.12: Evaluation of lip contour segmentation accuracy with respect to manually delineated ground truth based on texture data.

95% mark. The latter observation suggests that, at least on the test set at hand, the ACGA scheme provides more robust results than the texture-based classic ACs. Note, that this assertion remains valid, to a minor degree, also for the 3D GACs.

Accuracy distribution The above stated effect is illustrated in Fig. 7.13. Results with decreasing accuracy are compared for three approaches: the texture-based ACs and two curvature-based methods (3D GACs and ACGAs). Clearly, the curvature-based approaches capture the lip characteristics quite well even when they deliver a relatively poor accuracy (right column, 90% percentile). In particular, the ACGA approach yields acceptable results for up to 95% of all datasets (see Fig. 7.12). For the curvature-based methods, the deviation from the ground truth usually stems from a uniform shift rather than from local divergence (e.g. Fig. 7.13(i)). By contrast, poor results obtained by the texture-based approach lose the typical mouth shape (e.g. Fig. 7.13(c)).

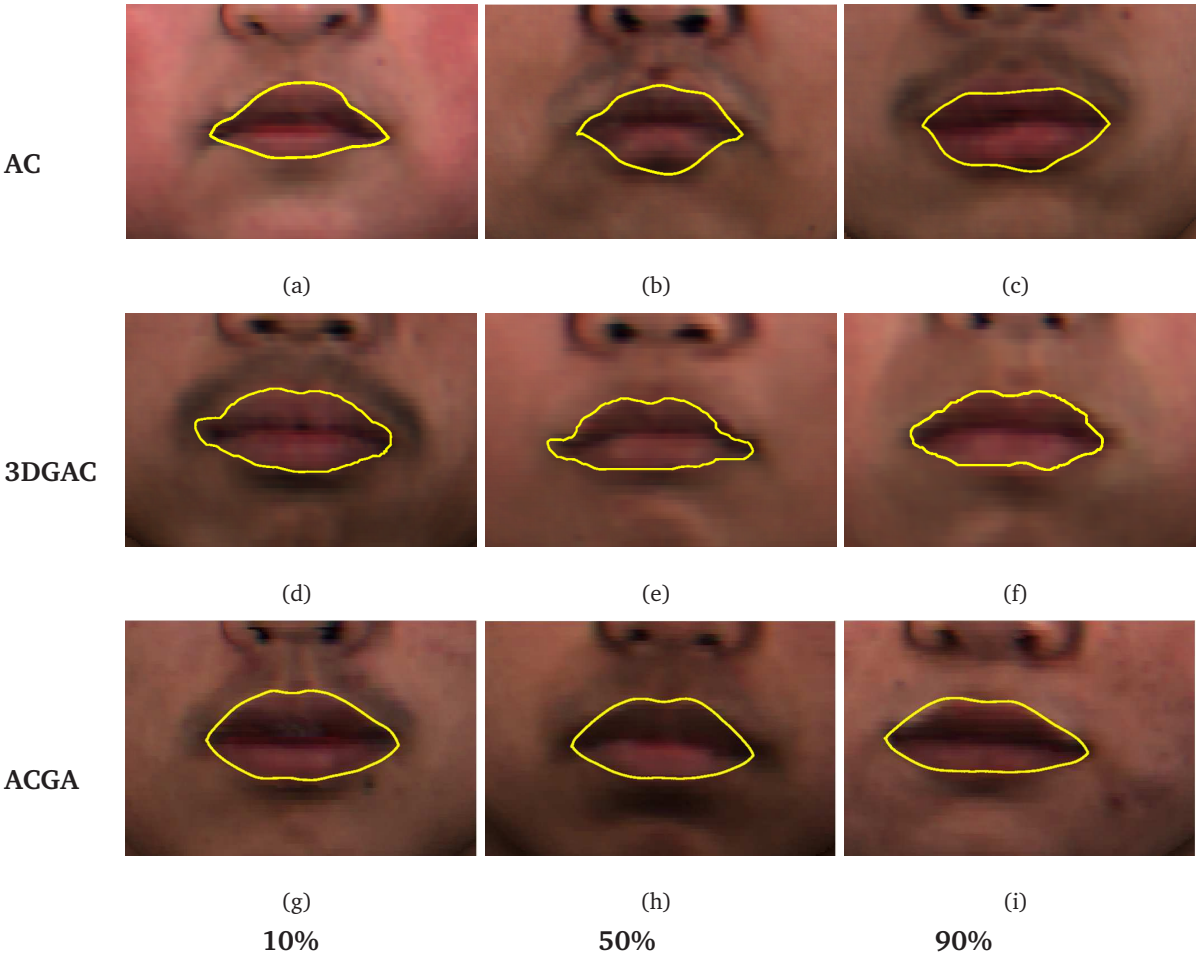


Figure 7.13: Sample results for three different approaches. The results in the first row (AC) were obtained with ACs on pseudo hue images; second row: 3D GAC on curvature data; third row: ACGA on curvature images. The three columns illustrate the deterioration of the results with decreasing accuracy: the first column represents the 10% percentile of the mean error distribution; second column: 50% percentile; third column: 90% percentile (cf. Fig. 7.12).

Benefit of the curvature-based approach It was analysed, to what extent the approaches presented in the thesis provide an actual benefit, compared to classic texture-based methods. First, the contours obtained with ACs on pseudo hue images were classified into the segments 'satisfactory' and 'unsatisfactory'. To this end, a suitable accuracy (i.e. mean error) threshold t_a was defined by visual assessment, resulting in a 60%/40% ratio (Table 7.3). Subsequently, the results obtained by curvature-based approaches were classified by means of the same threshold.

The cross tabulations of texture- vs. curvature-based ACs in Table 7.3 provides some intriguing insights. Both curvature-based schemes (ACGA and 3D GAC) yielded satisfactory results for at least 50% of the datasets with poor results by the ACs on texture. This corresponds to an additional 20% segment of the entire test set.

The top rows of the Figs. 7.14 and 7.15 display sample data sets where the respective curvature-based approaches yielded good results, while the classic ACs did not. The most frequent reasons for failure of the latter become apparent: weak edges at the upper lip, causing the contour to pass by, and dark, poorly illuminated regions under the lower lip, trapping the contour in spurious local minima. The opposite case is exemplified in the bottom rows of Figs. 7.14 and 7.15. Here, the texture-based ACs provided satisfactory accuracy, whereas the approaches using curvature data did not. Note, that the curvature-based methods converged properly, yet the obtained high curvature ridges differed from the texture-based lip contours.

		Classic ACs			
		Satisfactory	Unsatisfactory	Total	
ACGA	Satisfactory	22%	22%	44%	100%
	Unsatisfactory	38%	18%	56%	
3D GAC	Satisfactory	4%	20%	24%	100%
	Unsatisfactory	56%	20%	76%	
	Total	60%	40%	100%	

Table 7.3: Cross tabulations of classified accuracy results (texture-based ACs vs. curvature-based ACs).

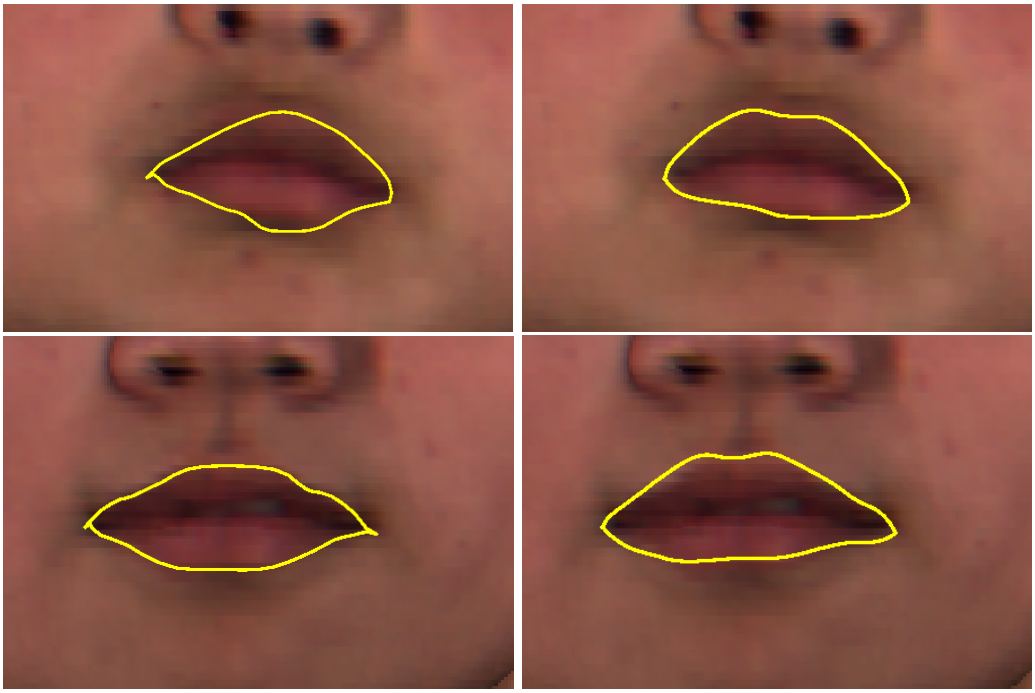


Figure 7.14: Comparison of texture- (AC, left) and curvature-based (ACGA, right) results.

Figure 7.15: Comparison of texture- (AC, left) and curvature-based (3D GAC, right) results.

7.5.2 Eye Contours

The pseudo-optimal second-order AC developed in Chapter 6 was applied to 3D eye contour segmentation (see Section 6.6.2). The results obtained on the 50 test datasets from the BJUT database were first compared to a ground truth, based only on shape/curvature data. This evaluation measures how good the obtained contours match the *anatomical* eye contours, which is of interest in the absence of texture data. Subsequently, the obtained curvature-based contours were compared to ground truth contours defined on the basis of texture data. Overall statistics of the accuracy evaluation are shown in Table 7.4, and cumulative distribution curves are visualised in Fig. 7.16. Not surprisingly, the contours, computed on curvature information, deliver a much more accurate approximation of the curvature-based than the texture-based ground truth contours. Note that both curves in Fig. 7.16 become noticeably flatter at approximately the 60% mark, indicating more quickly deteriorating accuracy results for the remaining 40% of the test datasets. The comparison with the results for lip contours shown in Figs. 7.7 and 7.12 underlines that the algorithms for the lips demonstrate a better robustness and more accurate results.

Ground truth based on	Mean	Max.	Std.	Mean Error median (pixels)	Processing time (s)
	Error (in pixels)				
Curvature	1.5 ± 0.8	6.9 ± 4.2	1.6 ± 1.2	1.2	36.1 ± 13.7
Texture	2.5 ± 0.8	8.7 ± 4.4	2.1 ± 1.2	2.2	36.1 ± 13.7

Table 7.4: Pseudo-optimal second-order ACs: accuracy evaluation for eye contour segmentation compared to curvature- and texture-based ground truth contours.

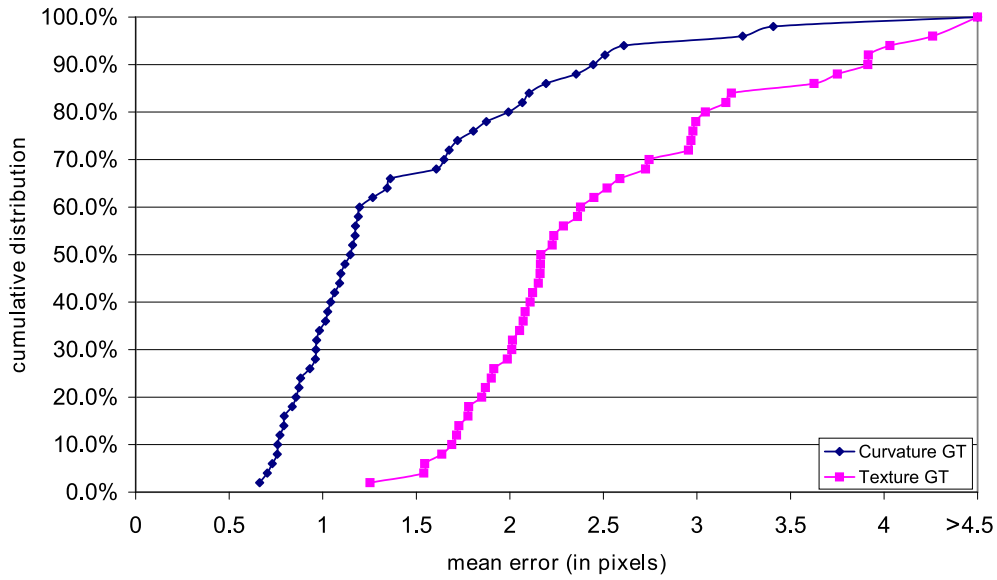
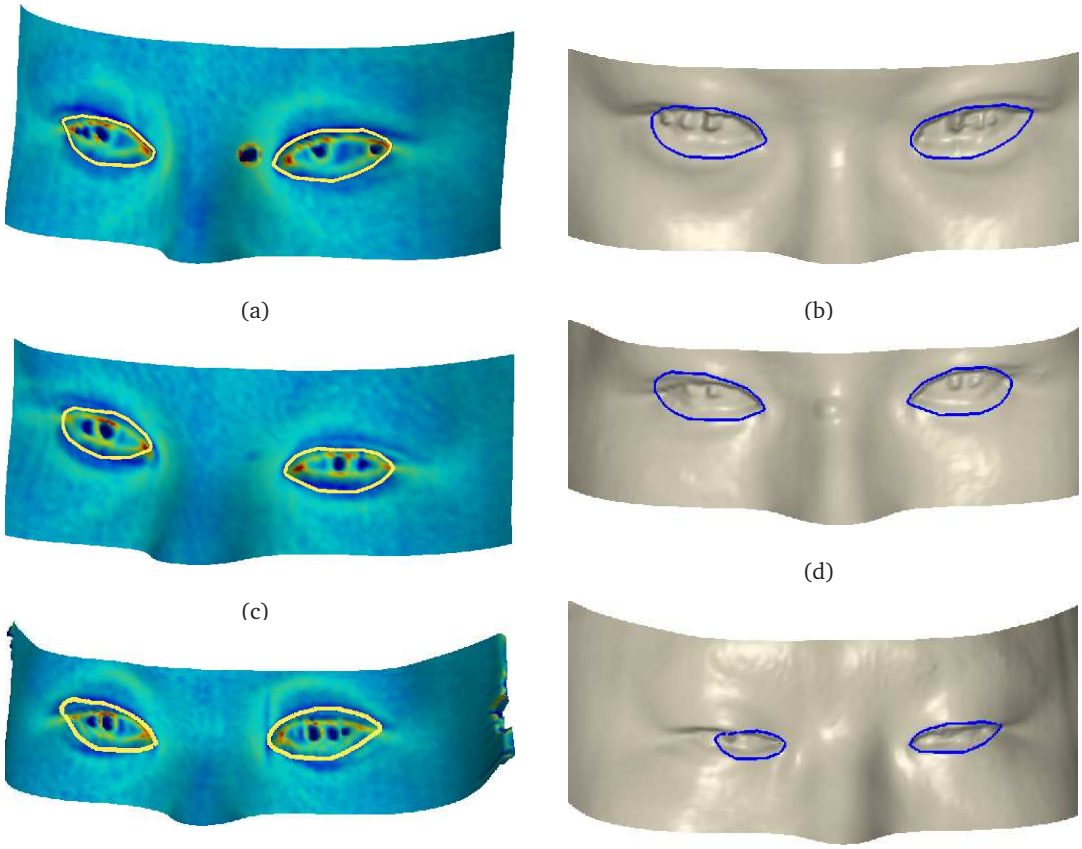


Figure 7.16: Pseudo-optimal second-order ACs: evaluation of eye segmentation accuracy with respect to texture- and curvature-based ground truth contours.

Comparison with curvature-based ground truth Figure 7.17 shows sample results for good, average, and poor eye segmentation results, obtained by the pseudo-optimal second-order AC. In several cases (see e.g. Fig. 7.17(e)) the contours provide useful information on the position of the eyes, even despite large deviations from the ground truth contours.

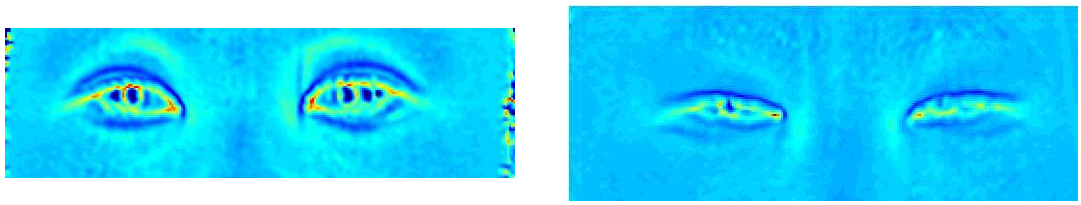
Figure 7.18 illuminates the most frequent reasons for poor accuracy results:

- The presence of further, possibly more distinctive, ridges above or below the desired ridges that model the outer eye contour (Fig. 7.18(a),7.17(e));
- Low data quality and scan artefacts (Figs. 7.18(b),7.17(f)).



(a) (b) (c) (d) (e) (f)

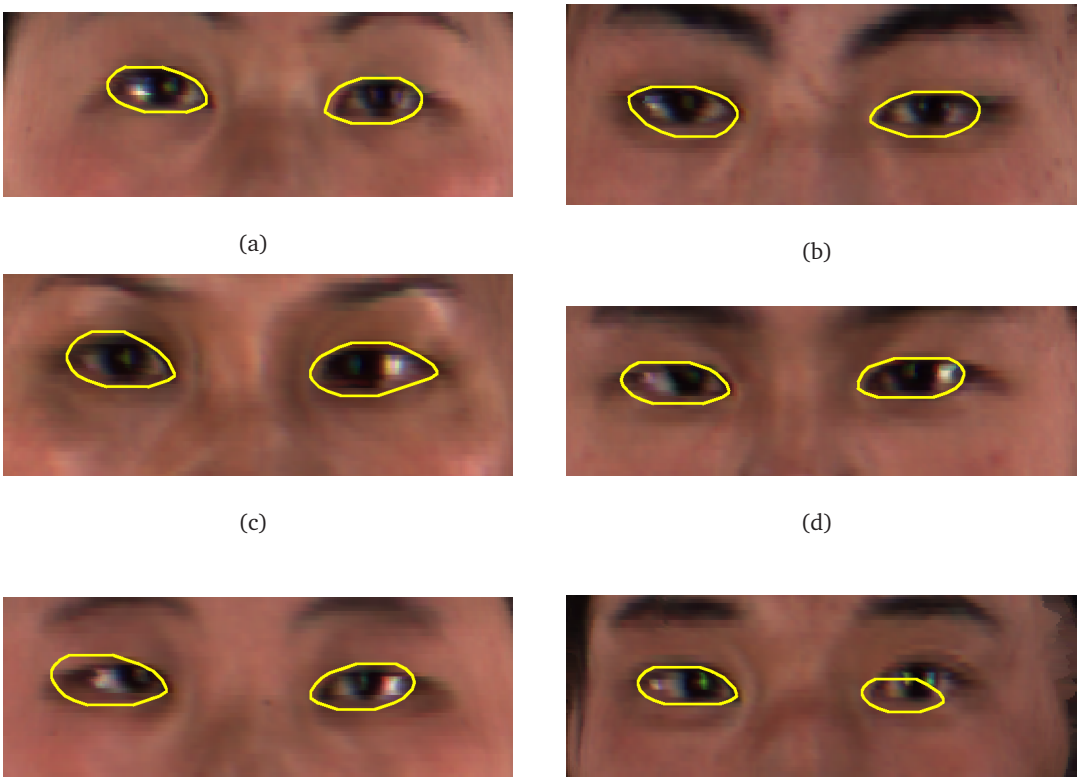
Figure 7.17: Sample results for eye contour segmentation using curvature information. Visualised on surfaces colorised with mean curvature information (left) and plain surfaces (right). Top row: good results (10% accuracy percentile); middle row: average results (50% accuracy percentile); bottom row: poor results (90% accuracy percentile).



(a) (b)

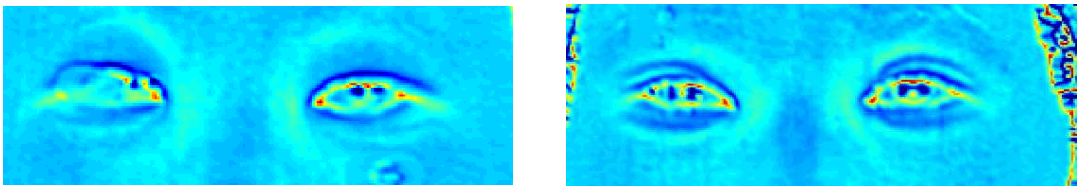
Figure 7.18: Mean curvature images exemplifying typical reasons for poor results. The images correspond to the results depicted in Figs. 7.17(e) and 7.17(f).

Comparison with texture-based ground truth Figure 7.19 indicates that for at least 50% of all datasets, eye contours computed on the basis of curvature information approximate the actual texture-based contours quite well. In Fig. 7.20, the curvature images behind the poor results in Figs. 7.19(e) and 7.19(f) are displayed. Again, the unsatisfactory results are caused by low scan quality (Fig. 7.20(a)) and artefacts (Fig. 7.20(b)).



(a) (b) (c) (d) (e) (f)

Figure 7.19: Sample results for eye contour segmentation using curvature information. Visualised on texture images. Top row: good results (10% accuracy percentile); middle row: average results (50% accuracy percentile); bottom row: poor results (90% accuracy percentile). The respective ground truth contours were defined by means of texture information.



(a) (b)

Figure 7.20: Mean curvature images corresponding to the results depicted in Figs. 7.19(e) and 7.19(f).

7.6 Discussion and Conclusions

This chapter has applied several techniques, including those developed in the thesis, to face feature contour segmentation using 3D data. The lip segmentation problem has been chosen for a comparative study of 3D, 2.5D and 2D approaches. Initially, different curvature-based approaches have been compared. Subsequently, the results based on curvature data have been contrasted with texture-based results. To the best of the author's knowledge such a study has not been published so far.

Some interesting new insights could be gained: Figure 7.7 suggests that segmentation results may be improved by incorporating the surface geometry into the evolution process. This had been conjectured before (see [SK07]), yet without any experimental evidence. Unfortunately, the gain in performance comes at the cost of increased processing times. The theoretical analysis in Section 4.3.3 showed that Spira and Kimmel's GAC model on parametric surfaces [SK07] becomes ill-posed once a balloon term is incorporated. This effect is confirmed empirically by Figure 7.7 and Table 7.1, which clearly indicate that Spira and Kimmel's [SK07] does not perform better than the standard 2D GAC on curvature images. Therefore, for the time being, the generalised GAC model proposed in Chapter 4 is the only option for GAC evolution using the surface geometry. Yet as the computations for the 3D GAC are executed in the 3D space, the computational load is considerable and the processing times may be prohibitive for some applications. By contrast, the 2.5D shortest path based ACG/ACGA schemes perform remarkably well while being very efficient (see Table 7.1). This approach with complexity $O(N \log N)$, may even be suitable for real-time applications[1].

The comparison with the texture-based approach shows that curvature-based feature contour segmentation is a feasible option, if texture data is of comparatively poor quality, or not available. In fact, this is often the case in current 3D acquisition systems: ideal lighting conditions for active range scanners are usually suboptimal for texture image generation [BR05]. Table 7.3 suggests that for about 20% of all faces in the test set, the curvature-based approaches provide satisfactory results while classic ACs on texture images fail.

Yet it has to be emphasised that the experiments reported in this chapter should be considered merely as a starting point for further investigations. The proposed algorithms ought to be tested on larger face databases of varying quality and ethnic groups, acquired by laser

[1] N denotes the number of image pixels.

scanners or alternative techniques.

At any rate, multimodal segmentation schemes are an exciting option for the future, combining the best of 'both worlds', 2D- and 3D data.

Conclusions

1. *Integration of the surface geometry into the GAC model can improve face feature segmentation results;*

2. *Curvature-based face feature segmentation techniques have the potential to provide more robust results than texture-based techniques;*

3. *Existing 2D systems can benefit from the usage of curvature information; there is a significant likelihood ($\geq$ 50% in the reported lip segmentation experiments) that a curvature-based technique may succeed where a texture-based method fails.*

Chapter 8

Conclusions and future directions

This chapter draws overall conclusions from the research presented in the thesis. Likewise, it is a reflection on the work done in the course of study. Note that chapter-specific conclusions can be found in the respective Chapters 4,5,6, and 7. Ultimately, potential future directions are discussed.

Summary

The focus of the thesis is the application of AC techniques for segmentation of surface data in 3D.

The question, whether results from the well-known 2D GAC can be improved by including 3D surface geometry into the segmentation scheme, served as a catalyst for Chapter 4. A thorough analysis of existing works revealed intrinsic drawbacks, affecting the segmentation results in an undesirable manner. Hence, a new generalised GAC model for curved surfaces was proposed. It was shown that the new approach inherits the advantageous properties of its 2D archetype, and it outperforms existing works in terms of accuracy.

In Chapter 5 the perspective was shifted to a particular application – the segmentation of feature contours from 3D face data. It was demonstrated that efficient 2D techniques can be employed for surface segmentation, when the data is given in the form of range images.

Motivated by the sample results for 3D eye contour segmentation in Chapter 5, a novel second-order AC framework for 2D images was introduced in Chapter 6. It enables the generalisation of known path-based first-order techniques to second-order energies. Experiments on medical images emphasise that in many cases the resulting AC algorithm yields similar, or even better outcomes while significantly less user input is required. The automatic variant

allows for segmenting round objects under strong noise, even if their boundary has large gaps.

Finally, Chapter 7 evaluated the algorithms from the Chapters 4,5, and 6 on a test set of 3D faces. Techniques based on face curvature characteristics were compared to a classic texture-based approach. The evaluation suggests that methods utilising 3D information can provide more robust results than merely texture-based schemes, particularly, if the depth data offers better resolution than the texture data. This, however, is the case in many acquisition systems with active range scanners.

Answers to Research Questions

This section refers to the research questions formulated previously (see Section 1.4).

- *Is there a natural generalisation of the geodesic active contour model to curved surfaces, and if yes, which is it?*

  Both theoretical and experimental results in Chapter 4 suggest that the new GAC scheme on implicit surfaces, rather than existing works on parametric surfaces exhibits the natural generalisation of the classic 2D GAC. In contrast to its competitors, it is fully implicit and allows a proper adoption of the approved balloon force concept.

- *Can face feature segmentation be improved by incorporating the face surface geometry into the evolution process?*

  The results in Section 7.5.1 indicate that incorporating the face surface geometry into the GAC model can improve the segmentation results, yet not drastically. The measured increase in accuracy is just under 10%.

- *Is it feasible to design a face feature contour segmentation system based only on surface shape data and curvature characteristics?*

  Yes, the techniques developed in Section 5.5 and the automatic second-order AC scheme proposed in Section 6.6.2 constitute a system for face feature segmentation employing only curvature data (see also [KDG09]).

- *How does curvature-based segmentation perform compared to well-known texture-based approaches?*

  The experiments for lip segmentation in Chapter 7 clearly show that curvature-based techniques can provide more robust results than texture-based approaches (cf. Figs.

7.12 and 7.13). Furthermore, the curvature-based techniques yielded satisfactory results for more than 50% of the datasets, where the purely texture-based approach failed. The latter outcome underlines that curvature-based techniques can provide a considerable additional benefit to classical 2D schemes.

- *Is there an efficient algorithm able to deliver at least approximately globally optimal second-order ACs?*

  Chapter 6 has developed an optimisation scheme for certain second-order energies, that inherits its complexity $O(N \log N)$ from first-order shortest path based techniques[1]. Hence, it is significantly more efficient than already known algorithms [SU88, SC07, WSC09].

Conclusions

The following conclusions constitute the essence of the research process.

1. 3D information enhances 2D segmentation results.

2. The pseudo-optimal second-order ACs offer a promising alternative to well-known first-order schemes.

3. The implicit, rather than the parametric surface representation allows a natural generalisation of the 2D GAC model.

4. Segmentation of 3D surface data does not necessarily require 3D techniques.

Future Directions

The following section highlights possible avenues for further investigation that have emerged in the course of study. They are relevant as they clearly indicate the future potential of the research done.

Face feature segmentation Several new techniques for face feature contour segmentation, with focus on 3D data, have been researched here. The approach for lip contour segmentation

[1] N is the number of image pixels.

presented in Section 5.5.2 could be extended to lip tracking in 3D image sequences, as the observed processing times suggest sufficient efficiency.

Curvature data can be employed to obtain a first guess of a feature contour, which is refined by means of texture data. In principle, integrating the proposed algorithms into multimodal schemes, with 2D- and 3D data complementing each other, opens up exciting new opportunities.

Pseudo-optimal second-order ACs So far, algorithms that provide globally optimal solutions with curvature regularity are restricted to 2D images [SU88, SC07, WSC09]. All these methods rely on shortest path schemes, since it is questionable whether graph cut based techniques can implement second-order energies. If the user provides suitable constraints, however, weighted shortest paths can be applied to surface segmentation in 3D volume data [AC06]. Integrating the open pseudo-elastica into Ardon and Cohen's approach could enable curvature regularity for surface segmentation in volume data. The promising results of the pseudo-elastica on 2D ultrasound images suggest that the resulting 3D segmentation scheme could be particularly suitable for segmenting 3D ultrasound volumes.

Thanks to its nature with one independent subproblem for each tested pixel, the automatic segmentation scheme could be significantly accelerated by an efficient implementation on a parallel architecture. Further it is worthwhile studying if this is also possible on a modern GPU processor. Both variants would allow the use of the algorithm for time sensitive applications such as object tracking.

GACs on surfaces The generalised GAC model introduced in Chapter 4 is based on curve evolution via the level set method [OS88, CBMO02]. While the latter has proven highly successful in the past, it also features some limitations. Only local energy minimisers are computed, and discrete approaches using graph cuts or shortest path algorithms are more efficient. A substantial improvement would be the generalisation of an efficient and globally optimal 2D GAC technique, as in [BK03, AT05], for triangulated surfaces. Yet both approaches make explicit use of the flat geometry of 2D images, and it is uncertain whether the key ideas translate to the surface scenario.

Due to its technical intricacy, the latter avenue for GACs on surfaces certainly exhibits the most challenging among the stated future directions.

List of Publications

[GKM+08] A. Gastelum, M. Krueger, J. Marquez, G. Gimel'farb, and P. Delmas. Automatic 3d lip shape segmentation and modelling. In *Proc. 23rd International Image and Vision Computing NZ Conference*, pages 1–6, Christchurch, New Zealand, Nov. 2008.

[KDG07] M. Krueger, P. Delmas, and G. Gimel'farb. Towards feature extraction on implicit surfaces using geodesic active contours. In *Proc. 22nd International Image and Vision Computing NZ Conference*, pages 294–299, Hamilton, New Zealand, 2007.

[KDG08a] M. Krueger, P. Delmas, and G. Gimel'farb. Active contour based segmentation of 3d surfaces. In *Proc. European Conference on Computer Vision*, pages 350–363, Marseille, France, 2008.

[KDG08b] M. Krueger, P. Delmas, and G. Gimel'farb. On 3d face feature segmentation using implicit surface active contours. In *Proc. 23rd International Image and Vision Computing NZ Conference*, pages 1–6, Christchurch, New Zealand, 2008.

[KDG09] M. Krueger, P. Delmas, and G. Gimel'farb. Automatic and efficient 3d face feature segmentation with active contours. In *Proc. 24th International Image and Vision Computing NZ Conference*, pages 165–170, Wellington, New Zealand, 2009.

Appendix A

Source Code for the 3D GAC

A.1 MATLAB Code

```
%*************************************************************
%           3DGAC_main.m
%           MATLAB Code for 3D GAC on implicit surfaces
%           Main loop
%*************************************************************

clear;
clc;

global N Nx Ny Nz X Y Z index_matrix H1_abs H2_abs H3_abs nb_gamma_val a_gp
 epsilon dt_adv dt1 epsi;
% *********** Set parameters ****************
N=120;
Nx=N;
Ny=N;
Nz=N;

iterations=1500;
thresh=0.8;             % for visualisation purposes
sigma=1.2;                  % scale parameter in Gaussian kernel
gkernelsize=5;          % size of Gaussian kernel
alpha=0.9;                  %CFL-constant

beta=1;         %weight for the curvature term  (in the thesis: mu)
delta=-1;       %weight for advection-term (-1)
gamma=-0.4; %weight balloon-force (in the thesis: c)

slope_th=1.2;       %threshold for average gradient slope
a_gp=3;         %Number of grid points for optimization of LF scheme
cnt_th=10;          %max. iterations before reinitialisation
dt_fix=1.0;         %time step

%narrowband parameters
nb_beta_val=2;
nb_gamma_val=4;
dist_th=1;        %threshold for curve distance to NB

epsi=0.1;
epsilon=0.001;      %value for gradient norm regularisation

x=1:Nx;
y=1:Ny;
z=1:Nz;
```

```
[X,Y,Z]=meshgrid(x,y,z);

G=fspecial('gaussian',gkernelsize,sigma);

phi=init_curve;
phi=reinitialize_fast_marching(phi);

H1_abs=ones(Ny,Nx,Nz);
H2_abs=H1_abs;
H3_abs=H1_abs;

dt1=get_dt_normal(0.9, H1_abs,H1_abs,H1_abs);
index_matrix=def_index_matrix;

psi=init_surface;
psi=reinitialize_fast_marching(psi);

%smooth surface by curvature flow
sprintf('Smoothing the surface...')
psi_sm=curvature_flow(psi,3,0.8);


%grad psi is calculated by centered diffs
[psi_x,psi_y,psi_z]=centered_first3D(psi_sm);
dt_adv = get_dt_advect_global(alpha, psi_x,psi_y,psi_z);


dist=phi;

image=init_image(bg);

% Standardise image
image=resurrect_global(image,psi,psi_x,psi_y,psi_z,10,0.9);
% smooth image by Gaussain convolution
img_smooth=smooth3(image,'gaussian',gkernelsize,sigma);

plot_curveonsurf(phi,psi,X,Y,Z,thresh,image);
%title('Initial curve');
camlight;


[img_x,img_y,img_z]=centered_first3D(img_smooth);

 %calculate standard stopping function with surface gradient
f=1./(1+(img_x.^2+img_y.^2+img_z.^2));

[f_x,f_y,f_z]=centered_first3D(f);
[f_x,f_y,f_z]=project_tangent(f_x,f_y,f_z,psi_x,psi_y,psi_z);

f_x=delta*f_x;
f_y=delta*f_y;
f_z=delta*f_z;

clear img_x img_y img_z img_smooth but; %remove obsolete vars

it=1;
reinit_flag=1;
reinit_cnt=0;

anz_iter_reinit=round((nb_gamma_val+1)/alpha)+1; %(4.71) in the thesis
d=10;           %subsequently, d is the curve distance from the narrowband
it=1;               %iteration counter

while(it <= iterations)
    if (reinit_flag==1)         %narrowband needs to be updated
       nb_gamma=define_narrowband_curve(phi,psi,nb_beta_val,nb_gamma_val);
       nb_delta=narrowband_hull(nb_gamma,nb_beta_val-dist_th);
       absphi=abs(phi);
```

```
        psi_nb=psi(nb_gamma);

        %check, if extended narrowband intersects the boundary
        if length([find(nb_delta<=2*Nx*Ny);find(nb_delta>Nx*Ny*Nz-2*Nx*Ny);...
            find(mod(nb_delta,Nx*Ny)<=2*Ny);find(mod(nb_delta,Nx*Ny)>Nx*Ny-2*Ny);...
            find(mod(nb_delta,Ny)<=2);find(mod(nb_delta,Ny)>=Ny-1)])>0
             error('Error! Surface is too close to boundary!');
        end
        sprintf('Narrowband updated...\n')
        %define cut function
        c=cut_function(phi(nb_gamma),psi(nb_gamma),nb_beta_val,nb_gamma_val);

        [psi_nb,psi_xp2,psi_xm2,psi_yp2,psi_ym2,psi_zp2,psi_zm2]=extend(psi,nb_gamma);
        reinit_flag=0;
    end
    if mod(it,50)==0
              plot_curveonsurf(phi,psi,X,Y,Z,thresh,image);
              drawnow;
    end

    delta_gac=evolve_gac(phi,psi,psi(nb_gamma),psi_xp2,psi_xm2,...
        psi_yp2,psi_ym2,psi_zp2,psi_zm2,beta,f(nb_gamma),f_x(nb_gamma),...
        f_y(nb_gamma),f_z(nb_gamma),gamma,psi_x(nb_gamma),psi_y(nb_gamma),...
        psi_z(nb_gamma),nb_beta_val,nb_gamma,c);


    phi(nb_gamma) = phi(nb_gamma) + dt_fix* delta_gac;
    phi_nb=phi(nb_gamma);

    %check if reinitialisation is necessary
    d=get_curve_distance(phi(nb_gamma),psi(nb_gamma),dist(nb_gamma),nb_beta_val);
    s=get_av_slope(phi,nb_gamma,psi_x(nb_gamma),psi_y(nb_gamma),psi_z(nb_gamma));
    sprintf('distance from beta-band: %0.5g.\n, average slope in NB: %0.5g.',d,s)
    if isempty(d)
        d=0;
    end
    if (reinit_cnt==cnt_th || abs(1-s)>=(slope_th-1) ||...
                        (d<=(dist_th) && ¬isempty(d) && d¬=0))
         phi=reinit(phi,psi_x(nb_delta),psi_y(nb_delta),psi_z(nb_delta),...
                                        anz_iter_reinit,0.9,nb_delta);
         phi=resurrect(phi,psi(nb_delta),psi_x(nb_delta),psi_y(nb_delta),...
                                psi_z(nb_delta),nb_gamma_val,0.9,nb_delta);
         dist=phi;
         retain_dist=0;
         sprintf('Phi reinitialised...\n')
         reinit_cnt=0;
         reinit_flag=1;
         absphi=abs(phi);
     else
         reinit_cnt=reinit_cnt+1;
     end
     disp(it);
     it = it+1;
end

%*********************************************************
%                   init_curve.m
%                   defines the initial curve
%*********************************************************

function phi=init_curve

global X Y Z Nx Ny Nz N;

%Simple curve on a sphere
phi=-(Y-70);
```

```
%*********************************************************
%                   init_surface.m
%                   defines the surface
%*********************************************************

function psi=init_surface

%initialisation of the implicit surface
%here the simple example of a sphere with radius R

global X Y Z Nx Ny Nz

R=40;
psi=sqrt((X-Nx/2).^2+(Y-Ny/2).^2+(Z-Nz/2).^2-R^2);

%*********************************************************
%           init_image.m
%           defines the image/texture on the surface
%*********************************************************

function image=init_image(bg)

global X Y Z Nx Ny Nz;

image=repmat(bg,[Ny Nx Nz]);

%sphere multiple objects
 image((X-Nx/2).^2+(Z-(Nz/2+30)).^2<=15^2)=100;
 image((Y-Ny/2).^2+(Z-(Nz/2+30)).^2<=15^2)=100;

%*********************************************************
%           get_dt_normal.m
%           computes the Euler time step for a normal flow
%*********************************************************

function dt = get_dt_normal(alpha, H1_abs, H2_abs, H3_abs)

if alpha <= 0 | alpha >= 1
    error('alpha needs to be between 0 and 1!');
end

%max reduces the dimension step by step...
maxs = max(max(max(H1_abs + H2_abs+ H3_abs)));
dt = alpha/(maxs+(maxs==0));

%*********************************************************
%           get_dt_advect_global.m
%           computes the Euler time step for an advection flow
%*********************************************************

function [dt] = get_dt_advect_global(alpha,u,v,w)
% Calculate the Euler time step.

if alpha <= 0 | alpha >= 1
    error('alpha needs to be between 0 and 1!');
end

maxs = max(max(max(abs(u)+ abs(v)+abs(w))));
dt = alpha/(maxs+(maxs==0));

%*********************************************************
%           calculate_advect.m
%           computes a local advection time step
%*********************************************************

function [delta] = calculate_advect(phi,u_nb,v_nb,w_nb,nb_flag)

[phi_x_minus,phi_x_plus,phi_y_minus,phi_y_plus,phi_z_minus,phi_z_plus]=...
```

```
                                        getWENO5_nb(phi,nb_flag);

% Upwinding
phi_x=(u_nb>0).*phi_x_minus+(u_nb<=0).*phi_x_plus;
phi_y=(v_nb>0).*phi_y_minus+(v_nb<=0).*phi_y_plus;
phi_z=(w_nb>0).*phi_z_minus+(w_nb<=0).*phi_z_plus;

% now compute advection term
delta= u_nb.*phi_x + v_nb.* phi_y + w_nb.*phi_z;

%*********************************************************
%           calculate_advect_global.m
%           computes a global advection time step
%*********************************************************

function [delta] = calculate_advect_global(phi,u,v,w)

phi_x=zeros(size(phi));
phi_y=zeros(size(phi));
phi_z=zeros(size(phi));

%compute fifth-order accurate WENO scheme
[phi_x_minus,phi_x_plus,phi_y_minus,phi_y_plus,phi_z_minus,phi_z_plus]=...
                                        getWENO5_std(phi);

%-------------------------------------------------------------------------
% Upwinding
phi_x=(u>0).*phi_x_minus+(u<=0).*phi_x_plus;
phi_y=(u>0).*phi_y_minus+(u<=0).*phi_y_plus;
phi_z=(u>0).*phi_z_minus+(u<=0).*phi_z_plus;

% now compute advection term
delta= u.*phi_x + v.* phi_y + w.*phi_z;

%*********************************************************
%           calculateH_dist.m
%           computes a local normal flow time step
%*********************************************************

function delta = calculateH_dist(phi,psi_x_nb,psi_y_nb,psi_z_nb,...
                                        Vn_nb,nb_flag)

global H1_abs H2_abs H3_abs a_gp epsilon dt1;

[phi_x_minus,phi_x_plus,phi_y_minus,phi_y_plus,phi_z_minus,phi_z_plus]=...
                                        getWENO5_nb(phi,nb_flag);

%compute Local-Lax-Friedrich scheme

%average derivative (= central difference)
phi_x=0.5*(phi_x_minus+phi_x_plus);
phi_y=0.5*(phi_y_minus+phi_y_plus);
phi_z=0.5*(phi_z_minus+phi_z_plus);

[phi_x_T,phi_y_T,phi_z_T]=project_tangent(phi_x,phi_y,phi_z,...
                                        psi_x_nb,psi_y_nb,psi_z_nb);
abs_grad=phi_x_T.^2+phi_y_T.^2+phi_z_T.^2;

%note, that all alpha-values from the LF-scheme are equal 1 (estimated)
alphax=1;
alphay=1;
alphaz=1;

delta=Vn_nb.*(sqrt(abs_grad)-1)...
      - 0.5*alphax.*(phi_x_plus-phi_x_minus)...
      - 0.5*alphay.*(phi_y_plus-phi_y_minus)...
      - 0.5*alphaz.*(phi_z_plus-phi_z_minus);
```

```
%*********************************************************
%           curvature_flow.m
%           computes the mean curvature flow of a surface
%*********************************************************

function psi=curvature_flow(psi,iterations,alpha);

global epsilon;

it=1;

while(it<=iterations)
    [psi_x,psi_y,psi_z,psi_xx,psi_yy,psi_zz,psi_xy,psi_xz,psi_yz] =...
                              centered_firstsecond3D(psi);

    %calculate mean curvature
    H=(psi_xx.*(psi_y.^2+psi_z.^2)+psi_yy.*(psi_x.^2+psi_z.^2)+...
        psi_zz.*(psi_x.^2+psi_y.^2)-2*(psi_x.*psi_y.*psi_xy+...
        psi_y.*psi_z.*psi_yz-psi_x.*psi_z.*psi_xz))./...
            (psi_x.^2+psi_y.^2+psi_z.^2+epsilon).^1.5;

    abs_grad=sqrt(psi_x.^2+psi_y.^2+psi_z.^2);

    delta_cf=H.*abs_grad;

    dt_cf=alpha/6;      %fixed time step
    psi = psi + dt_cf* delta_cf;

    it=it+1;
end

%*********************************************************
%           resurrect_global.m
%           computes a global advection flow to ressurrect phi on psi
%*********************************************************

function [phi]=resurrect_global(phi,psi,psi_x,psi_y,psi_z,iterations,alpha)

S_psi_0 = psi./sqrt(psi.^2 + 1);        %regularised signum-function

abs_grad_psi=sqrt(psi_x.^2+psi_y.^2+psi_z.^2+epsilon^2);

psi_x=S_psi_0.*psi_x./abs_grad_psi;    %integrate signum in gradient field
psi_y=S_psi_0.*psi_y./abs_grad_psi;
psi_z=S_psi_0.*psi_z./abs_grad_psi;

it=0;
while(it < iterations)
  [delta_advect] = calculate_advect_global(phi,psi_x,psi_y,psi_z);
    dt = get_dt_advect_global(alpha, psi_x,psi_y,psi_z);
    phi = phi - dt* delta_advect;
    it = it+1;
end

%*********************************************************
%           resurrect.m
%           computes a local advection flow to ressurrect phi on psi
%*********************************************************

function [phi]=resurrect(phi,psi_nb,psi_x_nb,psi_y_nb,psi_z_nb,...
                                iterations,alpha,nb_flag)

global epsilon dt_adv;

%regularised signum-function
S_psi_0_nb = psi_nb./sqrt(psi_nb.^2 + 1);

abs_grad_psi=sqrt(psi_x_nb.^2+psi_y_nb.^2+psi_z_nb.^2+epsilon^2);
```

```
%integrate signum in gradient field
psi_x_nb=S_psi_0_nb.*psi_x_nb./abs_grad_psi;
psi_y_nb=S_psi_0_nb.*psi_y_nb./abs_grad_psi;
psi_z_nb=S_psi_0_nb.*psi_z_nb./abs_grad_psi;

it=0;
while(it < iterations)
    [delta_advect] = calculate_advect(phi,psi_x_nb,psi_y_nb,psi_z_nb,nb_flag);
    phi(nb_flag) = phi(nb_flag) - dt_adv* delta_advect;
    it = it+1;
end


%*********************************************************
%           reinit.m
%           reinitialises phi in the narrowband
%*********************************************************

function [phi]=reinit(phi,psi_x_nb,psi_y_nb,psi_z_nb,...
                    iterations,alpha,nb_flag)

global dt1  H1_abs H2_abs H3_abs
S_phi_0 = phi./sqrt(phi.^2 + 1);        %regularised signum-function


it=0;
t=0;
while(it < iterations)
    delta_normal = calculateH_dist(phi,psi_x_nb,psi_y_nb,psi_z_nb,...
                         S_phi_0(nb_flag),nb_flag);
  dt=dt1;
    phi(nb_flag) = phi(nb_flag) - dt* delta_normal;
    it = it+1;
end

%generic value for phi outside the narrowband
but=ones(size(phi));
flag=but(nb_flag);
phi(¬flag)=8;


%*********************************************************
%           evolve_gac.m
%           computes a time step for the 3D GAC
%*********************************************************

function delta_gac=evolve_gac(phi,psi,psi_nb,psi_xp2,psi_xm2,...
            psi_yp2,psi_ym2,psi_zp2,psi_zm2,beta,b_nb,u_nb,v_nb,w_nb,gamma,...
            psi_x_nb,psi_y_nb,psi_z_nb,nb_beta,nb_flag,c)

global  H1_abs H2_abs H3_abs epsilon a_gp;

%compute values at neighbouring pixels
nb_xp2=shift_band(nb_flag ,0,2,0);
nb_xm2=shift_band(nb_flag ,0,-2,0);

nb_yp2=shift_band(nb_flag ,2,0,0);
nb_ym2=shift_band(nb_flag ,-2,0,0);

nb_zp2=shift_band(nb_flag ,0,0,2);
nb_zm2=shift_band(nb_flag ,0,0,-2);

phi_c=phi(nb_flag);
phi_xp2=phi(nb_xp2);
phi_xm2=phi(nb_xm2);

phi_yp2=phi(nb_yp2);
phi_ym2=phi(nb_ym2);

phi_zp2=phi(nb_zp2);
```

```
phi_zm2=phi(nb_zm2);

psi_c=psi_nb;
[phi_x_minus,phi_x_plus,phi_y_minus,phi_y_plus,phi_z_minus,phi_z_plus]=...
                                                    getWENO5_nb(phi,nb_flag);
phi_x=0.5*(phi_x_minus+phi_x_plus);
phi_y=0.5*(phi_y_minus+phi_y_plus);
phi_z=0.5*(phi_z_minus+phi_z_plus);

% hyperbolic terms
%determine dissipation terms (alpha values , s. osher/fedkiw)
%get dissipation values from external MEX function
[alphax,alphay,alphaz]=get_dissipation_LF(phi_x_minus,phi_x_plus,...
            phi_y_minus,phi_y_plus,phi_z_minus,phi_z_plus,psi_x_nb,psi_y_nb,...
            psi_z_nb,gamma,b_nb,u_nb,v_nb,w_nb,a_gp,epsilon);

[phi_x_T,phi_y_T,phi_z_T]=project_tangent(phi_x,phi_y,phi_z,...
                    psi_x_nb,psi_y_nb,psi_z_nb); %get tangential component

abs_grad=sqrt(phi_x_T.^2+phi_y_T.^2+phi_z_T.^2+epsilon^2);


delta_an=gamma*b_nb.*abs_grad +(u_nb.*phi_x + v_nb.* phi_y + w_nb.*phi_z)...
     -0.5*alphax.*(phi_x_plus-phi_x_minus)...
     -0.5*alphay.*(phi_y_plus-phi_y_minus)...
     -0.5*alphaz.*(phi_z_plus-phi_z_minus);

%-------------------------------
% now the curvature term
%-------------------------------
%curvature is discretised in two steps
%according to divergence-definition

%first calculate neighbour-gradients in x-direction...
%POINT i+1,j,k

nb_110=shift_band(nb_flag ,1,1,0);
nb_m110=shift_band(nb_flag ,-1,1,0);
nb_011=shift_band(nb_flag ,0,1,1);
nb_01m1=shift_band(nb_flag ,0,1,-1);

phi_x_p=(phi_xp2-phi_c)/2;
phi_y_p=(phi(nb_110)-phi(nb_m110))/2;
phi_z_p=(phi(nb_011)-phi(nb_01m1))/2;

psi_x_p=(psi_xp2-psi_c)/2;
psi_y_p=(psi(nb_110)-psi(nb_m110))/2;
psi_z_p=(psi(nb_011)-psi(nb_01m1))/2;

%POINT i-1,j,k

nb_1m10=shift_band(nb_flag ,1,-1,0);
nb_m1m10=shift_band(nb_flag ,-1,-1,0);
nb_0m11=shift_band(nb_flag ,0,-1,1);
nb_0m1m1=shift_band(nb_flag ,0,-1,-1);

phi_x_m=(phi_c-phi_xm2)/2;
phi_y_m=(phi(nb_1m10)-phi(nb_m1m10))/2;
phi_z_m=(phi(nb_0m11)-phi(nb_0m1m1))/2;

psi_x_m=(psi_c-psi_xm2)/2;
psi_y_m=(psi(nb_1m10)-psi(nb_m1m10))/2;
psi_z_m=(psi(nb_0m11)-psi(nb_0m1m1))/2;

%project gradients
[phi_x_T_m,phi_y_T_m,phi_z_T_m]=project_tangent(phi_x_m,phi_y_m,phi_z_m,...
                      psi_x_m,psi_y_m,psi_z_m); %get tangential component

[phi_x_T_p,phi_y_T_p,phi_z_T_p]=project_tangent(phi_x_p,phi_y_p,phi_z_p,...
```

```
                        psi_x_p,psi_y_p,psi_z_p); %get tangential component

abs_grad_plus=sqrt(phi_x_T_p.^2+phi_y_T_p.^2+phi_z_T_p.^2+epsilon^2);
abs_grad_minus=sqrt(phi_x_T_m.^2+phi_y_T_m.^2+phi_z_T_m.^2+epsilon^2);

grad_psi_plus=sqrt(psi_x_p.^2+psi_y_p.^2+psi_z_p.^2);
grad_psi_minus=sqrt(psi_x_m.^2+psi_y_m.^2+psi_z_m.^2);

grad_p=phi_x_T_p./abs_grad_plus.*grad_psi_plus;
grad_m=phi_x_T_m./abs_grad_minus.*grad_psi_minus;

kappag_x=(grad_p-grad_m)/2;

% calculate neighbour-gradients in y-direction...
%POINT i,j+1,k
nb_101=shift_band(nb_flag ,1,0,1);
nb_10m1=shift_band(nb_flag ,1,0,-1);

phi_x_p=(phi(nb_110)-phi(nb_1m10))/2;
phi_y_p=(phi_yp2-phi_c)/2;
phi_z_p=(phi(nb_101)-phi(nb_10m1))/2;

psi_x_p=(psi(nb_110)-psi(nb_1m10))/2;
psi_y_p=(psi_yp2-psi_c)/2;
psi_z_p=(psi(nb_101)-psi(nb_10m1))/2;

%POINT i,j-1,k
nb_m101=shift_band(nb_flag ,-1,0,1);
nb_m10m1=shift_band(nb_flag ,-1,0,-1);

phi_x_m=(phi(nb_m110)-phi(nb_m1m10))/2;
phi_y_m=(phi_c-phi_ym2)/2;
phi_z_m=(phi(nb_m101)-phi(nb_m10m1))/2;

psi_x_m=(psi(nb_m110)-psi(nb_m1m10))/2;
psi_y_m=(psi_c-psi_ym2)/2;
psi_z_m=(psi(nb_m101)-psi(nb_m10m1))/2;

%project gradients

[phi_x_T_m,phi_y_T_m,phi_z_T_m]=project_tangent(phi_x_m,phi_y_m,phi_z_m,...
                        psi_x_m,psi_y_m,psi_z_m); %get tangential component

[phi_x_T_p,phi_y_T_p,phi_z_T_p]=project_tangent(phi_x_p,phi_y_p,phi_z_p,...
                        psi_x_p,psi_y_p,psi_z_p); %get tangential component

abs_grad_plus=sqrt(phi_x_T_p.^2+phi_y_T_p.^2+phi_z_T_p.^2+epsilon^2);
abs_grad_minus=sqrt(phi_x_T_m.^2+phi_y_T_m.^2+phi_z_T_m.^2+epsilon^2);

grad_psi_plus=sqrt(psi_x_p.^2+psi_y_p.^2+psi_z_p.^2);
grad_psi_minus=sqrt(psi_x_m.^2+psi_y_m.^2+psi_z_m.^2);

grad_p=phi_y_T_p./abs_grad_plus.*grad_psi_plus;
grad_m=phi_y_T_m./abs_grad_minus.*grad_psi_minus;

kappag_y=(grad_p-grad_m)/2;


% calculate neighbour-gradients in z-direction...
%POINT i,j,k+1
phi_x_p=(phi(nb_011)-phi(nb_0m11))/2;
phi_y_p=(phi(nb_101)-phi(nb_m101))/2;
phi_z_p=(phi_zp2-phi_c)/2;

psi_x_p=(psi(nb_011)-psi(nb_0m11))/2;
psi_y_p=(psi(nb_101)-psi(nb_m101))/2;
psi_z_p=(psi_zp2-psi_c)/2;

%POINT i,j,k-1
```

```
phi_x_p=(phi(nb_01m1)-phi(nb_0m1m1))/2;
phi_y_p=(phi(nb_10m1)-phi(nb_m10m1))/2;
phi_z_p=(phi_c-phi_zm2)/2;

psi_x_p=(psi(nb_01m1)-psi(nb_0m1m1))/2;
psi_y_p=(psi(nb_10m1)-psi(nb_m10m1))/2;
psi_z_p=(psi_c-psi_zm2)/2;

%project gradients

[phi_x_T_m,phi_y_T_m,phi_z_T_m]=project_tangent(phi_x_m,phi_y_m,phi_z_m,...
                      psi_x_m,psi_y_m,psi_z_m); %get tangential component

[phi_x_T_p,phi_y_T_p,phi_z_T_p]=project_tangent(phi_x_p,phi_y_p,phi_z_p,...
                      psi_x_p,psi_y_p,psi_z_p); %get tangential component

abs_grad_plus=sqrt(phi_x_T_p.^2+phi_y_T_p.^2+phi_z_T_p.^2+epsilon^2);
abs_grad_minus=sqrt(phi_x_T_m.^2+phi_y_T_m.^2+phi_z_T_m.^2+epsilon^2);

grad_psi_plus=sqrt(psi_x_p.^2+psi_y_p.^2+psi_z_p.^2);
grad_psi_minus=sqrt(psi_x_m.^2+psi_y_m.^2+psi_z_m.^2);

grad_p=phi_z_T_p./abs_grad_plus.*grad_psi_plus;
grad_m=phi_z_T_m./abs_grad_minus.*grad_psi_minus;

kappag_z=(grad_p-grad_m)/2;
kappag= (kappag_x+kappag_y+kappag_z);      %... and compute divergence

abs_grad_psi=sqrt(psi_x_nb.^2+psi_y_nb.^2+psi_z_nb.^2+epsilon^2);

delta_k=beta*b_nb.*kappag .* abs_grad./abs_grad_psi;
delta_gac=c.*(delta_k-delta_an);

%***********************************************************
%           shift_band.m
%           computes index values of the shifted pixels/points
%***********************************************************

function sb=shift_band(nb_flag,y,x,z)
global Nx Ny Nz

sb=nb_flag+z*Nx*Ny+x*Ny+y;

%***********************************************************
%           project_tangent.m
%           projects vectors onto the tangent space
%***********************************************************

function [u,v,w]=project_tangent(u,v,w,psi_x,psi_y,psi_z)
%orthogonal projection on the tangent space as per Eq. (4.16), p.62

global epsilon;
abs_grad2=psi_x.^2+psi_y.^2+psi_z.^2+epsilon^2;
ip=(u.*psi_x+v.*psi_y+w.*psi_z)./abs_grad2;
u=u-ip.*psi_x;
v=v-ip.*psi_y;
w=w-ip.*psi_z;

%***********************************************************
%           getWENO5_nb.m
%           calculates 5th order accurate back- and forward derivatives
%            on a narrowband defined by nb_flag
%***********************************************************

function [phi_x_minus,phi_x_plus,phi_y_minus,phi_y_plus,...
           phi_z_minus,phi_z_plus]=getWENO5_nb(phi,nb_flag)

%define corresponding neighbours...
%...in order to calculate derivatives
```

```
nb_xp=shift_band(nb_flag,0,1,0);
nb_xp2=shift_band(nb_flag,0,2,0);
nb_xp3=shift_band(nb_flag,0,3,0);
nb_xm=shift_band(nb_flag,0,-1,0);
nb_xm2=shift_band(nb_flag,0,-2,0);
nb_xm3=shift_band(nb_flag,0,-3,0);

nb_yp=shift_band(nb_flag,1,0,0);
nb_yp2=shift_band(nb_flag,2,0,0);
nb_yp3=shift_band(nb_flag,3,0,0);
nb_ym=shift_band(nb_flag,-1,0,0);
nb_ym2=shift_band(nb_flag,-2,0,0);
nb_ym3=shift_band(nb_flag,-3,0,0);

nb_zp=shift_band(nb_flag,0,0,1);
nb_zp2=shift_band(nb_flag,0,0,2);
nb_zp3=shift_band(nb_flag,0,0,3);
nb_zm=shift_band(nb_flag,0,0,-1);
nb_zm2=shift_band(nb_flag,0,0,-2);
nb_zm3=shift_band(nb_flag,0,0,-3);

phi_c=phi(nb_flag);
phi_xp=phi(nb_xp);
phi_xp2=phi(nb_xp2);
phi_xp3=phi(nb_xp3);
phi_xm=phi(nb_xm);
phi_xm2=phi(nb_xm2);
phi_xm3=phi(nb_xm3);

phi_yp=phi(nb_yp);
phi_yp2=phi(nb_yp2);
phi_yp3=phi(nb_yp3);
phi_ym=phi(nb_ym);
phi_ym2=phi(nb_ym2);
phi_ym3=phi(nb_ym3);

phi_zp=phi(nb_zp);
phi_zp2=phi(nb_zp2);
phi_zp3=phi(nb_zp3);
phi_zm=phi(nb_zm);
phi_zm2=phi(nb_zm2);
phi_zm3=phi(nb_zm3);

%note: all following calculations are performed only on the narrowband....

%first backward derivative : for notation see osher/fedkiw pp.32
% k=i-1 f

D1_im12=phi_c-phi_xm;   %first divided diff and backward difference
D1_im32=phi_xm-phi_xm2;  % D1_im12 stands for D1_{i-1/2}
D1_im52=phi_xm2-phi_xm3;
D1_ip12=phi_xp-phi_c;
D1_ip32=phi_xp2-phi_xp;
D1_ip52=phi_xp3-phi_xp2;

v1=D1_im52;
v2=D1_im32;
v3=D1_im12;
v4=D1_ip12;
v5=D1_ip32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

epsilon = 1e-6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
```

```
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_x_minus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;


v1=D1_ip52;
v2=D1_ip32;
v3=D1_ip12;
v4=D1_im12;
v5=D1_im32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_x_plus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;

%*****************************************************
% y - derivative

%first backward derivative : for notation see osher/fedkiw pp.32
% k=i-1 f

D1_im12=phi_c-phi_ym;   %first divided diff and backward difference
D1_im32=phi_ym-phi_ym2;  % D1_im12 stands for D1_{i-1/2}
D1_im52=phi_ym2-phi_ym3;
D1_ip12=phi_yp-phi_c;
D1_ip32=phi_yp2-phi_yp;
D1_ip52=phi_yp3-phi_yp2;

v1=D1_im52;
v2=D1_im32;
v3=D1_im12;
v4=D1_ip12;
v5=D1_ip32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_y_minus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;
```

```
v1=D1_ip52;
v2=D1_ip32;
v3=D1_ip12;
v4=D1_im12;
v5=D1_im32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_y_plus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;


% z - derivative
D1_im12=phi_c-phi_zm;   %first divided diff and backward difference
D1_im32=phi_zm-phi_zm2;  % D1_im12 stands for D1_{i-1/2}
D1_im52=phi_zm2-phi_zm3;
D1_ip12=phi_zp-phi_c;
D1_ip32=phi_zp2-phi_zp;
D1_ip52=phi_zp3-phi_zp2;

v1=D1_im52;
v2=D1_im32;
v3=D1_im12;
v4=D1_ip12;
v5=D1_ip32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_z_minus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;

v1=D1_ip52;
v2=D1_ip32;
v3=D1_ip12;
v4=D1_im12;
v5=D1_im32;

data1=v1/3 - 7*v2/6 + 11*v3/6;
data2=-v2/6 + 5*v3/6 + v4/3;
data3= v3/3 + 5*v4/6 - v5/6;

S1=(13/12)*(v1-2*v2+v3).^2+(1/4)*(v1-4*v2+3*v3).^2;
S2=(13/12)*(v2-2*v3+v4).^2+(1/4)*(v2-v4).^2;
S3=(13/12)*(v3-2*v4+v5).^2+(1/4)*(3*v3-4*v4+v5).^2;

alpha1 = 0.1./((S1+epsilon).^2);
```

```
    alpha2 = 0.6./((S2+epsilon).^2);
    alpha3 = 0.3./((S3+epsilon).^2);
    a_total = alpha1+alpha2+alpha3;

phi_z_plus= (alpha1./a_total).*data1 + (alpha2./a_total).*data2 ...
        + (alpha3./a_total).*data3;
```

The function *getWENO5_std.m* was programmed analogously using suitable boundary conditions.

```
%*************************************************************
%           centered_first3D.m
%           approximate first-order derivatives by central differences
%*************************************************************
function [fx,fy,fz] = centered_first3D(f)

%central differences are computed by summing up the backward- and
%forward differences

[fxm,fxp,fym,fyp,fzm,fzp]=firstderivatives_3D(f);
fx=(fxp+fxm)/2;
fy=(fyp+fym)/2;
fz=(fzp+fzm)/2;


%*************************************************************
%           firstderivatives_3D.m
%           approximates first-order forward/backward differences
%*************************************************************

function [fxm,fxp,fym,fyp,fzm,fzp]=firstderivatives_3D(f)

f_xp1 = zeros(size(f));
f_xm1 = zeros(size(f));
f_yp1 = zeros(size(f));
f_ym1 = zeros(size(f));
f_zp1 = zeros(size(f));
f_zm1 = zeros(size(f));

f_xm1(:,2:end,:) = f(:,1:end-1,:);
f_xm1(:,1,:) = 2*f_xm1(:,2,:)-f_xm1(:,3,:);
f_xp1(:,1:end-1,:) = f(:,2:end,:);
f_xp1(:,end,:) = 2*f_xp1(:,end-1,:)-f_xp1(:,end-2,:);

fxm=f-f_xm1;
fxp=f_xp1-f;

clear f_xm1 f_xp1;

f_ym1(2:end,:,:) = f(1:end-1,:,:);
f_ym1(1,:,:) = 2*f_ym1(2,:,:)-f_ym1(3,:,:);
f_yp1(1:end-1,:,:) = f(2:end,:,:);
f_yp1(end,:,:) = 2*f_yp1(end-1,:,:)-f_yp1(end-2,:,:);

fym=f-f_ym1;
fyp=f_yp1-f;
clear f_ym1 f_yp1;

f_zm1(:,:,2:end) = f(:,:,1:end-1);
f_zm1(:,:,1) = 2*f_zm1(:,:,2)-f_zm1(:,:,3);
f_zp1(:,:,1:end-1) = f(:,:,2:end);
f_zp1(:,:,end) = 2*f_zp1(:,:,end-1)-f_zp1(:,:,end-2);


fzm=f-f_zm1;
fzp=f_zp1-f;
clear f_zm1 f_zp1;
```

```
%*********************************************************
%           define_narrowband_curve.m
%           delivers the index vector of the computed narrow band
%*********************************************************

function nb_flag=define_narrowband_curve(phi,psi,nb_beta_val,nb_gamma_val)

global index_matrix

nb_flag=index_matrix(sqrt(abs(phi).^2+abs(psi).^2)<=nb_gamma_val);
nb_flag=sort(nb_flag);

%*********************************************************
%           narrowband_hull.m
%           function starts form a narrowband flag field and adds
%           neighboring pixels
%*********************************************************

function nb_flag_hull=narrowband_hull(nb_flag,width)

global Nx Ny Nz

%first check, if surface respectively the narrow band is
% too close to the cuboid boundary
if length([find(nb_flag<=Nx*Ny);find(nb_flag>Nx*Ny*Nz-Nx*Ny);...
           find(mod(nb_flag,Nx*Ny)<=Ny);find(mod(nb_flag,Nx*Ny)>Nx*Ny-Ny);...
           find(mod(nb_flag,Ny)==1);find(mod(nb_flag,Ny)==0)])>0
    error('Error! Surface is too close to boundary!');
end

sz=size(nb_flag,1);
vec=zeros(sz*(2*width+1)^3,1);

cnt=0;
for i=-width:width
    for j=-width:width
        for k=-width:width
            nb=nb_flag+i*Nx*Ny+j*Ny+k;
            vec((cnt*sz+1):(cnt+1)*sz,1)=nb;
            cnt=cnt+1;
        end
    end
end

%duplicate entries in array but have to be deleted
vec=sort(vec);
%precompiled MEX-routine removes duplicates
nb_flag_hull=nodupkey_c(vec);

%*********************************************************
%           cut_function.m
%           computes cut function as in Peng et al.
%*********************************************************
function c=cut_function(phi_nb,psi_nb,nb_beta_val,nb_gamma_val)

dist=sqrt(abs(phi_nb).^2+abs(psi_nb).^2);

c=(dist<=nb_beta_val)+...
  (dist>nb_beta_val).*(dist-nb_gamma_val).^2.*...
  (2*dist+nb_gamma_val-3*nb_beta_val)./(nb_gamma_val-nb_beta_val).^3;

%*********************************************************
%           get_curve_distance.m
%           approximates the distance of the curve from the beta-band
%
%*********************************************************

function d=get_curve_distance(phi_nb,psi_nb,dist_nb,nb_beta_val)
```

```
global epsi;

curve=(abs(phi_nb)<=epsi & abs(psi_nb)<=epsi);
d=nb_beta_val-max(abs(dist_nb(curve)));

%*********************************************************
%           get_av_slope.m
%           compute average slope in the narrow band
%*********************************************************

function s=get_av_slope(phi,nb_flag,psi_x_nb,psi_y_nb,psi_z_nb)

%define corresponding neighbours...
%...in order to calculate derivatives
nb_xp=shift_band(nb_flag ,0,1,0);
nb_xm=shift_band(nb_flag ,0,-1,0);

nb_yp=shift_band(nb_flag ,1,0,0);
nb_ym=shift_band(nb_flag ,-1,0,0);

nb_zp=shift_band(nb_flag ,0,0,1);
nb_zm=shift_band(nb_flag ,0,0,-1);

phi_c=phi(nb_flag);
phi_xp=phi(nb_xp);
phi_xm=phi(nb_xm);

phi_yp=phi(nb_yp);
phi_ym=phi(nb_ym);

phi_zp=phi(nb_zp);
phi_zm=phi(nb_zm);

phi_x=(phi_xp-phi_xm)/2;
phi_y=(phi_yp-phi_ym)/2;
phi_z=(phi_zp-phi_zm)/2;

[phi_x_T,phi_y_T,phi_z_T]=project_tangent(phi_x,phi_y,phi_z,...
                                          psi_x_nb,psi_y_nb,psi_z_nb);

abs_grad=sqrt(phi_x_T.^2+phi_y_T.^2+phi_z_T.^2);
s=mean(abs_grad);

%*********************************************************
%           extend.m
%           compute value of function f on neighbouring pixels
%*********************************************************

function [f_nb,f_xp2,f_xm2,f_yp2,f_ym2,f_zp2,f_zm2]=extend(f,nb_flag)

%define corresponding neighbours...
nb_xp2=shift_band(nb_flag,0,2,0);
nb_xm2=shift_band(nb_flag,0,-2,0);

nb_yp2=shift_band(nb_flag,2,0,0);
nb_ym2=shift_band(nb_flag,-2,0,0);

nb_zp2=shift_band(nb_flag,0,0,2);
nb_zm2=shift_band(nb_flag,0,0,-2);

f_nb=f(nb_flag);
f_xp2=f(nb_xp2);
f_xm2=f(nb_xm2);

f_yp2=f(nb_yp2);
f_ym2=f(nb_ym2);

f_zp2=f(nb_zp2);
```

```
f_zm2=f(nb_zm2);

%*********************************************************
%           plot_curveonsurf.m
%           plot curve on an implicit surface
%*********************************************************

function plot_curveonsurf(phi,psi,X,Y,Z,delta,colors)
figure;
colormap(pink(256));

%define the intersection of Phi and psi
IS=abs(phi)+abs(psi);
log=(IS<delta);

colors(log)=0;  %black curve
%colors(log)=250;  %white curve

[f,v,c]=isosurface(psi,0,colors);
p=patch('Vertices',v,'Faces',f,'FaceVertexCData',c);
set(p,'FaceColor','flat','edgecolor','none');
set(p,'FaceLighting','gouraud',...
    'AmbientStrength',.3,'DiffuseStrength',.5,...
    'SpecularStrength',.2,'SpecularExponent',25);

axis off;axis image
shading interp;lighting gouraud
```

A.2 Precompiled C/C++ - Code

The following functions were coded and compiled as MEX-functions and then called from MATLAB.

```
/*****************************************************************/
/*                   nodupkey_c.c                                */
/*                removes duplicate entries from a vector        */
/*****************************************************************/
#include "mex.h"

#define max(a,b) ((a) > (b) ? (a) : (b))
void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
    double *v,*w;
    int m,n,i,n_new,*flag,cnt;
    double old;
    if(nlhs != 1 || nrhs!=1)
        mexErrMsgTxt("The number of input and or output arguments is wrong!");

    m=mxGetM(prhs[0]);
    n=mxGetN(prhs[0]);
    n=max(m,n);     /* number of elements of the input vector */
    n_new=n;

    v=mxGetPr(prhs[0]);

    flag=mxCalloc(n,sizeof(int));


    for(i=0;i<n;i++) flag[i]=0;

    /*find double entries */
    old=v[0];
    for(i=1;i<n;i++){
        if(v[i]==old){
            flag[i]=1;
```

```
            n_new--;
        }
        else old=v[i];
    }

    plhs[0]=mxCreateDoubleMatrix(n_new,1,mxREAL);
    w=mxGetPr(plhs[0]);
    cnt=0;
    for (i=0;i<n;i++)
        if (flag[i]==0)
            w[cnt++]=v[i];
}

/*************************************************************************/
/*     get_dissipation_LF.c                                              */
/*     computes dissipation coeffs as per global Lax-Friedrich scheme */
/*************************************************************************/
#include "mex.h"
#include "math.h"
#include "matrix.h"

#define min(a,b) ((a) < (b) ? (a) : (b))
#define max(a,b) ((a) > (b) ? (a) : (b))

void mexFunction(int nlhs, mxArray *plhs[],int nrhs,
                                 const mxArray *prhs[])
{
    double *alphax,*alphay,*alphaz;
    double *phi_x_minus,*phi_x_plus,*phi_y_minus,*phi_y_plus,
        *phi_z_minus,*phi_z_plus,*psi_x_nb,*psi_y_nb,*psi_z_nb,gamma,
        *b_nb,*u_nb,*v_nb,*w_nb;
    double *x,*y,*z,*proj_x,*proj_y,*proj_z;
    double ip,abs_grad,butler1,butler2,butler3,abs_grad_psi2,epsilon;
    int i,j,k,l,n,a_gp1,a_gp2,a_gp3,p;
    double H1abs,H2abs,H3abs;
    double phi_x_min,phi_x_max,phi_y_min,phi_y_max,phi_z_min,phi_z_max;

    if(nlhs != 3 || nrhs!=16)
        mexErrMsgTxt("The number of input and or output
                                arguments is wrong!");

    n=mxGetM(prhs[0]);
    a_gp1=(int)mxGetScalar(prhs[14]);
    a_gp2=a_gp1*a_gp1;
    a_gp3=a_gp2*a_gp1;

    epsilon=(double)mxGetScalar(prhs[15]);

    plhs[0]=mxCreateDoubleMatrix(n,1,mxREAL);
    plhs[1]=mxCreateDoubleMatrix(n,1,mxREAL);
    plhs[2]=mxCreateDoubleMatrix(n,1,mxREAL);

    alphax=mxGetPr(plhs[0]);
    alphay=mxGetPr(plhs[1]);
    alphaz=mxGetPr(plhs[2]);

    x=mxCalloc(a_gp3,sizeof(double));
    y=mxCalloc(a_gp3,sizeof(double));
    z=mxCalloc(a_gp3,sizeof(double));

    proj_x=mxCalloc(a_gp3,sizeof(double));
    proj_y=mxCalloc(a_gp3,sizeof(double));
    proj_z=mxCalloc(a_gp3,sizeof(double));


    phi_x_minus=mxGetPr(prhs[0]);
    phi_x_plus=mxGetPr(prhs[1]);
    phi_y_minus=mxGetPr(prhs[2]);
```

```
    phi_y_plus=mxGetPr(prhs[3]);
    phi_z_minus=mxGetPr(prhs[4]);
    phi_z_plus=mxGetPr(prhs[5]);
    psi_x_nb=mxGetPr(prhs[6]);
    psi_y_nb=mxGetPr(prhs[7]);
    psi_z_nb=mxGetPr(prhs[8]);

    gamma=mxGetScalar(prhs[9]);

    b_nb=mxGetPr(prhs[10]);
    u_nb=mxGetPr(prhs[11]);
    v_nb=mxGetPr(prhs[12]);
    w_nb=mxGetPr(prhs[13]);


    /* determine alpha x*/
    /* global min/max values for y and z   */
     for(p=0;p<n;p++){
        if(p==0){
            phi_x_min=min(phi_x_minus[p],phi_x_plus[p]);
            phi_x_max=max(phi_x_minus[p],phi_x_plus[p]);
            phi_y_min=min(phi_y_minus[p],phi_y_plus[p]);
            phi_y_max=max(phi_y_minus[p],phi_y_plus[p]);
            phi_z_min=min(phi_z_minus[p],phi_z_plus[p]);
            phi_z_max=max(phi_z_minus[p],phi_z_plus[p]);
        }
        else{
           phi_x_min= min(min(phi_x_minus[p],phi_x_plus[p]),phi_x_min);
           phi_x_max= max(max(phi_x_minus[p],phi_x_plus[p]),phi_x_max);
           phi_y_min= min(min(phi_y_minus[p],phi_y_plus[p]),phi_y_min);
           phi_y_max= max(max(phi_y_minus[p],phi_y_plus[p]),phi_y_max);
           phi_z_min= min(min(phi_z_minus[p],phi_z_plus[p]),phi_z_min);
           phi_z_max= max(max(phi_z_minus[p],phi_z_plus[p]),phi_z_max);
        }
    }

    l=0;
     /* prepare gridvalues for y/z for optimization   */
    for(k=0;k<a_gp1;k++)
        for(i=0;i<a_gp1;i++)
            for(j=0;j<a_gp1;j++){
                *(x+l)=phi_x_min+i*((phi_x_max-phi_x_min)/(a_gp1-1));
                *(y+l)=phi_y_min+j*((phi_y_max-phi_y_min)/(a_gp1-1));
                *(z+(l++))=phi_z_min+k*((phi_z_max-phi_z_min)/(a_gp1-1));
            }


    for(p=0;p<n;p++){

        l=0;
        butler1=0;
        butler2=0;
        butler3=0;
       /* maximize H1abs (absolut value of x-derivative of Hamiltonian
*/
        for(k=0;k<a_gp1;k++)
            for(i=0;i<a_gp1;i++)
                for(j=0;j<a_gp1;j++){
                    abs_grad_psi2=psi_x_nb[p]*psi_x_nb[p]+
                    psi_y_nb[p]*psi_y_nb[p]+psi_z_nb[p]*psi_z_nb[p]+
                    epsilon*epsilon;

                    ip=(*(x+l)*psi_x_nb[p]+ (*(y+l))*psi_y_nb[p]+
                                        (*(z+l))*psi_z_nb[p])/abs_grad_psi2;
                    *(proj_x+l)=*(x+l)-ip*psi_x_nb[p];
                    *(proj_y+l)=*(y+l)-ip*psi_y_nb[p];
                    *(proj_z+l)=*(z+l)-ip*psi_z_nb[p];
```

```
                    abs_grad=sqrt(pow(*(proj_x+l),2)+pow(*(proj_y+l),2)
                                                +pow(*(proj_z+l),2));

                    H1abs=fabs((*(proj_x+l)/abs_grad)*gamma*b_nb[p]+u_nb[p]);
                    H2abs=fabs((*(proj_y+l)/abs_grad)*gamma*b_nb[p]+v_nb[p]);
                    H3abs=fabs((*(proj_z+l)/abs_grad)*gamma*b_nb[p]+w_nb[p]);
                    butler1=max(butler1,H1abs);
                    butler2=max(butler2,H2abs);
                    butler3=max(butler3,H3abs);
                    l++;
                }
        alphax[p]=butler1;
        alphay[p]=butler2;
        alphaz[p]=butler3;
    }

    mxFree(x);
    mxFree(y);
    mxFree(z);

    mxFree(proj_x);
    mxFree(proj_y);
    mxFree(proj_z);
}
```

Note that the function *reinitialise_fast_marching* called in *3DGAC_main.m* was programmed using the Toolbox Fast Marching [Pey08]. Therefore the code is not given here.

References

[AB98] T. D. Alter and Ronen Basri. Extracting salient curves from images: An analysis of the saliency network. *International Journal of Computer Vision*, 27(1):51–69, 1998.

[AC06] R. Ardon and L. D. Cohen. Fast constrained surface extraction by minimal paths. *International Journal of Computer Vision*, 69(1):127–136, 2006.

[Ada75] R. Adams. *Sobolev Spaces*. Academic Press, Boston, 1975.

[AM03] L. Ambrosio and S. Masnou. A direct variational approach to a problem arising in image reconstruction. *Interfaces and Free Boundaries*, 5(1):63–81, 2003.

[AT05] Ben Appleton and Hugues Talbot. Globally optimal geodesic active contours. *Journal of Mathematical Imaging and Vision*, 23(1):67–86, 2005.

[AWJ90] A. A. Amini, T. E. Weymouth, and R. C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[BBTV07] I. Bogdanova, X. Bresson, J. Thiran, and P. Vandergheynst. Scale-space analysis and active contours for omnidirectional images. *IEEE Transactions on Image Processing*, 16(7):1888–1901, 2007.

[BCF06] Kevin W. Bowyer, Kyong Chang, and Patrick Flynn. A survey of approaches and challenges in 3d and multi-modal 3d + 2d face recognition. *Computer Vision Image Understanding*, 101(1):1–15, 2006.

[BD62] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, New Jersey, 1962.

[BdMP93] G. Bellettini, G. dal Maso, and M. Paolini. Semicontinuity and relaxation properties of a curvature depending functional in 2d. *Annali della Scuola Normale Superiore di Pisa, Classe di Scienze*, 20(4):247–297, 1993.

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[BFL06] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[bju05] The bjut-3d large-scale chinese face database. Technical report MISKL-TR-05-FMFR-001, Beijing University of Technology (BJUT), 2005.

[BK03] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proc. International Conference on Computer Vision*, pages 26–33, Nice, France, 2003.

[BK04] S. Bischoff and L. Kobbelt. Parameterization-free active contour models with topology control. *The Visual Computer: International Journal of Computer Graphics*, 20(4):217–228, 2004.

[BM75] J. R. Bennett and J. S. MacDonald. On the measurement of curvature in a quantized environment. *IEEE Transactions on Computers*, 24(8):803–820, 1975.

[BM04] G. Bellettini and L. Mugnai. Characterization and representation of the lower semicontinuous envelope of the elastica functional. *Annales de l'Institut Henri Poincare, Analyse Non Lineaire*, 6(21):839–880, 2004.

[BMC+03] M. Bertalmio, F. Memoli, L. T. Cheng, G. Sapiro, and S. Osher. Variational problems and partial differential equations on implicit surfaces: Bye bye triangulated surfaces? In S. Osher and N. Paragios, editors, *Geometric Level Set Methods in Imaging, Vision, and Graphics*, pages 381–398. Springer, New York, 2003.

[BR05] Chris Boehnen and Trina Russ. A fast multi-modal approach to facial feature detection. In *Proc. IEEE Workshops on Application of Computer Vision*, pages 135–142, Breckenridge, CO, USA, 2005.

[Bri07] Willie Brink. Gridtrimesh algorithm. MATLAB File exchange, 2007.

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[BWK05] Stephan Bischoff, Tobias Weyand, and Leif Kobbelt. Snakes on triangle meshes. In *Proc. Bildverarbeitung für die Medizin*, pages 208–212, Heidelberg, Germany, 2005.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH 2001*, pages 67–76, Los Angeles, USA, 2001.

[CBF05] Kyong I. Chang, Kevin W. Bowyer, and Patrick J. Flynn. An evaluation of multimodal 2d+3d face biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):619–624, 2005.

[CBMO02] Li-Tien Cheng, Paul Burchard, Barry Merriman, and Stanley Osher. Motion of curves constrained on surfaces using a level-set approach. *Journal of Computational Physics*, 175(2):604–644, 2002.

[CCCD93] Vincent Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1):1–31, 1993.

[CET01] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[CK97] Laurent D. Cohen and Ron Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24(1):57–78, 1997.

[CKS97] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[CKS02] Tony Chan, Sung Ha Kang, and Jianhong Shen. Euler's elastica and curvature based inpainting. *SIAM Journal of Applied Mathematics*, 63:564–592, 2002.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH '96*, pages 303–312, New Orleans, USA, 1996.

[CMM91] S. Chandran, T. Maejima, and S. Miyazaki. Global minima via dynamic programming: energy minimizing active contours. In *Proc. SPIE Vol. 1570 Geometric Methods in Computer Vision*, pages 391–402, San Diego, USA, 1991.

[Coh91] L.D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):211–218, 1991.

[CP98] S. Chandran and A. K. Potty. Energy minimization of contours using boundary conditions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):546–549, 1998.

[CRAC06] C. Conde, L. J. Rodriguez-Aragon, and E. Cabello. Automatic 3d face feature points extraction with spin images. In *Proc. International Conference on Image Analysis and Recognition*, pages 317–328, Povoa de Varzim, Portugal, 2006.

[CSJ05] Dirk Colbry, George Stockman, and Anil Jain. Detection of anchor points for 3d face verification. In *Proc. Computer Vision and Pattern Recognition (CVPR) - Workshops*, pages 118–125, San Diego, USA, 2005.

[CTCG95] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models—their training and application. *Computer Vision Image Understanding*, 61(1):38–59, 1995.

[CV01] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[Dac89] Bernard Dacorogna. *Direct Methods in the Calculus of Variations*. Springer, New York, 1989.

[dC76] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[DEL02] Patrice Delmas, Nicolas Eveno, and Marc Lievin. Towards robust lip tracking. In *Proc. International Conference on Pattern Recognition*, volume 2, pages 528–531, Quebec, Canada, 2002.

[DFPH09] A. Delaunoy, K. Fundana, E. Prados, and A. Heyden. Convex multi-region segmentation on manifolds. In *Proc. International Conference on Computer Vision*, page tbd, Kyoto, Japan, 2009.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[EB10] In Encyclopedia Britannica online, November 2010. Retrieved November 28, 2010, from Encyclopædia Britannica Online: http://www.britannica.com/EBchecked/topic/130691/computer-vision.

[ECC04] Nicolas Eveno, Alice Caplier, and Pierre-Yves Coulon. Accurate and quasi-automatic lip tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):706–715, 2004.

[EG87] C. L. Epstein and M. Gage. The curve shortening flow. In A. Chorin and A. Majda, editors, *Wave Motion: Theory, Modeling, and Computation*. Springer, New York, 1987.

[EG92] L. C. Evans and R. F. Gariepy. *Measure Theory and Fine Properties of Functions*. CRC Press, 1992.

[Eul44] L. Euler. *Methodus inveniendi lineas curvas maxima minimive proprietate gaudentes*. Lausanne, 1744.

[EZ96] J.H. Elder and S.W. Zucker. Computing contour closure. In *Proc. European Conference on Computer Vision*, pages 399–412, Cambridge, UK, 1996.

[FH05] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[FTW81] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. *Computer Graphics and Image Processing*, 15:201–223, 1981.

[GDM+08] A. Gastelum, P.J. Delmas, J. Marquez, et al. 3d lip shape sph based evolution using prior 2d dynamic lip features extraction and static 3d lip measurements.

In Alan Wee-Chung Liew and Shilin Wang, editors, *Visual Speech Recognition: Lip Segmentation and Mapping*, pages 216–241. IGI Global, Hershey, PA, USA, 2008.

[GGCV95] Davi Geiger, Alok Gupta, Luiz A. Costa, and John Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, 1995.

[GH86] M. Gage and R.S. Hamilton. The heat equation shrinking convex plane curves. *Journal of Differential Geometry*, 23:69–96, 1986.

[GKM+08] A. Gastelum, M. Krueger, J. Marquez, G. Gimel'farb, and P. Delmas. Automatic 3d lip shape segmentation and modelling. In *Proc. 23rd International Image and Vision Computing NZ Conference*, pages 1–6, Christchurch, New Zealand, Nov. 2008.

[GM75] M. Gurtin and A. Murdoch. A continuum theory of elastic material surfaces. *Archive for Rational Mechanics and Analysis*, 57(4):291–323, 1975.

[Gor92] G.G. Gordon. Face recognition based on depth and curvature features. In *Proc. Computer Vision and Pattern Recognition*, pages 808–810, Champaign, IL, USA, Jun 1992.

[Gra87] M. Grayson. The heat equation shrinks embedded plane curves to round points. *Journal of Differential Geometry*, 26:285–314, 1987.

[Gra97] A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Boca Raton, FL, 2nd edition, 1997.

[HA03] F. Hetroy and D. Attali. From a closed piecewise geodesic to a constriction on a closed polyhedral surface. In *Proc. Pacific Graphics*, pages 394–398, Canmore, Canada, 2003.

[HECC06] Z. Hammal, N. Eveno, A. Caplier, and Py. Coulon. Parametric models for facial features segmentation. *Signal Processing*, 86(2):399–413, 2006.

[HKIU07] K. Hara, R. Kurazume, K. Inoue, and K. Urahama. Segmentation of images on polar coordinate meshes. In *Proc. International Conference on Image Processing*, pages II: 245–248, San Antonio, USA, 2007.

[JI01] Ian H. Jermyn and Hiroshi Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1075–1088, 2001.

[JK04] Moonryul Jung and Haengkang Kim. Snaking across 3d meshes. In *Proc. Pacific Conference on Computer Graphics and Applications*, pages 87–93, Seoul, South Korea, 2004.

[Joh91] F. John. *Partial Differential Equations*. Springer, New York, 1991.

[JP99] Guang-Shan Jiang and Danping Peng. Weighted eno schemes for hamilton–jacobi equations. *SIAM Journal of Scientific Computing*, 21(6):2126–2143, 1999.

[JYS07] H. Jin, A. Yezzi, and S. Soatto. Mumford-shah on the move: Region-based segmentation on deforming manifolds with application to 3-d reconstruction of shape and appearance from multi-view images. *Journal of Mathematical Imaging and Vision*, 29(2-3):219–234, 2007.

[KAB95] Ron Kimmel, Arnon Amir, and Alfred M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):635–640, 1995.

[Kan74] G. Kanizsa. Contours without gradients or cognitive contours? *Italian Journal of Psychology*, 1:93–112, 1974.

[KCK07] A. Khan, William J. Christmas, and Josef Kittler. Lip contour segmentation using kernel methods and level sets. In *Proc. International Symposium on Visual Computing*, pages 86–95, Lake Tahoe, USA, 2007.

[KDG08a] M. Krueger, P. Delmas, and G. Gimel'farb. Active contour based segmentation of 3d surfaces. In *Proc. European Conference on Computer Vision*, pages 350–363, Marseille, France, 2008.

[KDG08b] M. Krueger, P. Delmas, and G. Gimel'farb. On 3d face feature segmentation using implicit surface active contours. In *Proc. 23rd International Image and Vision Computing NZ Conference*, pages 1–6, Christchurch, New Zealand, 2008.

[KDG09] M. Krueger, P. Delmas, and G. Gimel'farb. Automatic and efficient 3d face feature segmentation with active contours. In *Proc. 24th International Image and Vision Computing NZ Conference*, pages 165–170, Wellington, New Zealand, 2009.

[Kim97] Ron Kimmel. Intrinsic scale space for images on surfaces: The geodesic curvature flow. *Graphical models and image processing: GMIP*, 59(5):365–372, 1997.

[Kim08] Ron Kimmel. personal communication, 2008.

[KKO+96] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Conformal curvature flows: From phase transitions to active vision. *Archives of Rational Mechanics and Analysis*, 134:275–301, 1996.

[Kre99] Rainer Kress. *Linear Integral Equations*. Springer, New York, 1999.

[Kru99] Matthias Krueger. Existenztheorie fuer eine geometrische evolutionsgleichung vierter ordnung. Master's thesis, University of Bonn, Germany, September 1999.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[Law66] E.L. Lawler. Optimal cycles in doubly weighted linear graphs. In *Proc. International Symposium on the Theory of Graphs*, pages 209–2136, Rome, Italy, 1966.

[LCJ04] X. Lu, D. Colbry, and A. K. Jain. Three-dimensional model based face recognition. In *Proc. International Conference on Pattern Recognition*, pages 362–366, Cambridge, UK, 2004.

[LDC+99] M. Lievin, P. Delmas, P. Y. Coulon, F. Luthon, and V. Fristot. Automatic lip tracking: Bayesian segmentation and active contours in a cooperative scheme. In *Proc. IEEE International Conference on Multimedia Computing and Systems*, pages 9691–9696, Florence, Italy, 1999.

[Li05] Bing Li. Active model toolbox. MATLAB File exchange, 2005.

[LJ06] X. Lu and A. K. Jain. Automatic feature extraction for multiview 3d face recognition. In *Proc. International Conference on Automatic Face and Gesture Recognition*, pages 585–590, Southampton, UK, 2006.

[LL02] Y. Lee and S. Lee. Geometric snakes for triangular meshes. *Computer Graphics Forum*, 21(3):229–238, 2002.

[LLS+05] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometry and Design*, 22(5):444–465, 2005.

[LTB96] Juergen Luettin, Neil A. Thacker, and Steve W. Beet. Active Shape Models for Visual Speech Feature Extraction. In D. G. Storck and M. E. Hennecke, editors, *Speechreading by Humans and Machines*, volume 150 of *NATO ASI Series, Series F: Computer and Systems Sciences*, pages 383–390. Springer Verlag, 1996.

[LWC05] L. Lui, Y. Wang, and T.F. Chan. Solving pdes on manifolds with global conformal parametriazation. In *Proc. Variational, Geometric, and Level Set Methods in Computer Vision*, pages 307–319, Beijing, China, 2005.

[LY07] H. Li and A. Yezzi. Local or global minima: Flexible dual-front active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):1–14, 2007.

[Mar76] Alberto Martelli. An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, 19(2):73–83, 1976.

[Mau00] Sean Mauch. A fast algorithm for computing the closest point and distance transform. *Technical Report, California Institute of Technology*, 2000.

[MB95] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, pages 191–198, Los Angeles, USA, 1995.

[MBV97] M. Milroy, C. Bradley, and G. Vickers. Segmentation of a wrap-around model using an active contour. *Computer Aided Design*, 29(4):299–320, 1997.

[Mit08] Ian M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35:300–329, June 2008.

[Mon71] Ugo Montanari. On the optimal detection of curves in noisy pictures. *Communications of the ACM*, 14(5):335–345, 1971.

[MS04] A. Moreno and A. Sanchez. Gavabdb: A 3d face database. In *Proc. COST275 Workshop on Biometrics on the Internet*, pages 75–80, Vigo, Spain, 2004.

[MSV95] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[MT95] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *Proc. International Conference on Computer Vision*, page 840, Cambridge, UK, 1995. IEEE Computer Society.

[MTW99] S. Mahamud, K. Thornber, and L.R. Williams. Segmentation of salient closed contours from real images. In *Proc. International Conference on Computer Vision*, pages 891–897, Kerkyra, Greece, 1999.

[Mum94] D. Mumford. Elastica and computer vision. In C.L. Bajaj, editor, *Algebraic Geometry and its Applications*, pages 491–506. Springer, New York, 1994.

[NC09] P. Nair and A. Cavallaro. 3-d face detection, landmark localization, and registration using a point distribution model. *IEEE Transactions on Multimedia*, 11(4):611–623, 2009.

[OF03] S. Osher and R. Fedkiw. *Level sets methods and dynamic implicit surfaces*. Springer, New York, 2003.

[OS88] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[Pey08] Gabriel Peyre. Toolbox fast marching. MATLAB File exchange, 2008.

[PM04] G. Papandreou and P. Maragos. A fast multigrid implicit algorithm for the evolution of geodesic active contours. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, volume II, pages 689–694, Washington, DC, 2004.

[PMZ+99] D. Peng, B. Merriman, H.-K. Zhao, S. Osher, and M. Kang. A pde based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[Poh71] I. Pohl. Bi-directional search. *Machine Intelligence*, 6:127–140, 1971.

[pri] http://www.cs.princeton.edu/gfx/proj/sugcon/models/.

[RM08] Steven J. Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3):1943–1961, 2008.

[Ron94] Rémi Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, 1994.

[SB07] Abu Sayeed Md. Sohail and Prabir Bhattacharya. Automated lip contour detection using the level set segmentation method. In *Proc. International Conference on Image Analysis and Processing*, pages 425–430, Modena, Italy, 2007.

[SC07] T. Schoenemann and D. Cremers. Introducing curvature into globally optimal image segmentation: Minimum ratio cycles on product graphs. In *Proc. International Conference on Computer Vision*, pages 1–6, Rio de Janeiro, Brazil, October 2007.

[Set96] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. National Academy Of Sciences*, volume 93, pages 1591–1595, 1996.

[Sha02] Jayant Shah. Elastica with hinges. *Journal of Visual Communication and Image Representation*, 13(1-2):36–43, 2002.

[SK07] Alon Spira and Ron Kimmel. Geometric curve flows on parametric manifolds. *Journal of Computational Physics*, 223:235–249, 2007.

[SLTZ98] K. Siddiqi, Y.B. Lauziere, A. Tannenbaum, and S. Zucker. Area and length minimizing flows for shape segmentation. *IEEE Transactions on Image Processing*, 3(7):433–443, 1998.

[SO88] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.

[SO89] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes,ii. *Journal of Computational Physics*, 83(1):32–78, 1989.

[SPR06] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.

[SS01] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, NJ, USA, 2001.

[SU88] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proc. International Conference on Computer Vision*, pages 321–327, 1988.

[SYMS08] G. Sundaramoorthi, A. Yezzi, A. C. G. Mennucci, and G. Sapiro. New possibilities with sobolev active contours. *International Journal of Computer Vision*, 2008.

[TKC00] Ying-Li Tian, Takeo Kanade, and Jeffrey Cohn. Dual-state parametric eye tracking. In *Proc. International Conference on Automatic Face and Gesture Recognition (FG)*, pages 110 – 115, Grenoble, France, March 2000.

[TMR09] L. Tian, C. Macdonald, and S. Ruuth. Segmentation on surfaces with the closest point method. In *Proc. International Conference on Image Processing*, Cairo, Egypt, 2009.

[TMS04] F. Tsalakanidou, S. Malassiotis, and M. G. Strintzis. Integration of 2d and 3d images for enhanced face authentication. In *Proc. IEEE 6th International Conference on Automatic Face and Gesture Recognition*, pages 266–271, Seoul, Korea, 2004.

[TW95] K. K. Thornber and L. R. Williams. Analytic solution of stochastic completion fields. *Biological Cybernetics*, 75:141–151, 1995.

[WCH02] Y.J. Wang, C.S. Chua, and Y.K. Ho. Facial feature detection and face recognition from 2d and 3d images. *Pattern Recognition Letters*, 23(10):1191–1202, August 2002.

[WDG08] A. Woodward, P. Delmas, and G. Gimel'farb. A 3d video scanner for face performance capture. In *Proc. 23rd International Image and Vision Computing NZ Conference*, pages 1–6, Christchurch, New Zealand, Nov. 2008.

[WDZC05] C. Wu, J. Deng, W. Zhu, and F. Chen. Inpainting images on implicit surfaces. In *Proc. Pacific Graphics 2005*, pages 142–144, University of Macau, 2005.

[Wen93] Yingzhong Wen. L^2 flow of curve straightening in the plane. *Duke Mathematical Journal*, 70(3):683–698, 1993.

[WKS05] S. Wang, T. Kubota, and J. M. Siskind. Salient closed boundary extraction with ratio contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):546–561, 2005.

[WLN05] Kwok-Wai Wan, Kin-Man Lam, and Kit-Chong Ng. An accurate active shape model for facial feature extraction. *Pattern Recognition Letters*, 26(15):2409–2423, 2005.

[WS92] Donna J. Williams and Mubarak Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.

[WSC09] T. Windheuser, T. Schoenemann, and D. Cremers. Beyond connecting the dots: A polynomial-time algorithm for segmentation and boundary estimation with imprecise user input. In *Proc. International Conference on Computer Vision*, page tbd, Kyoto, Japan, 2009.

[YB07] Ping Yan and Kevin W. Bowyer. Biometric recognition using 3d ear shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1297–1308, 2007.

[YTW99] A. Yezzi, A. Tsai, and A. Willsky. A statistical approach to snakes for bimodal and trimodal imagery. In *Proc. International Conference on Computer Vision*, pages 898–903, Kerkyra, Greece, 1999.

[ZC07] Wei Zhu and Tony Chan. A variational model for capturing illusory contours using curvature. *Journal of Mathematical Imaging and Vision*, 27(1):29–40, 2007.

[ZMT05] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.