Inaugural-Dissertation

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich–Mathematischen Gesamtfakultät
der
Ruprecht–Karls–Universität
Heidelberg

vorgelegt von
Dipl.–Inf. Christian Gosch
aus Mainz

Tag der mündlichen Prüfung: 16. Juli 2009

Contour Methods for View Point Tracking

Gutachter:	**Prof. Dr. Christoph Schnörr**
Zweitgutachter:	**Prof. Dr. Anders Heyden**

Abstract

Shape of objects, in particular the shape of object outlines, has for a long time been a focus in the literature and is widely regarded as carrying important information for visual and cognitive tasks, such as object recognition and object tracking. This thesis is concerned with techniques related to shape information from 2D images. The main contribution is a purely 2D shape based method for following view point changes of an observer relative to an object given an image sequence. Several techniques involved in such a task are covered in some detail. In particular, segmentation methods yielding contours, and shape representations are treated. On the shape side, classical representations and methods are included to a smaller extent, and a more recent, more sophisticated manifold of shapes including computational technicalities is treated in more detail.

Variational segmentation methods based on the successful level set representation are used for segmenting and tracking curves in image sequences. While this field has grown rapidly and is still developing further, this work covers enough detail to describe the implementation used for experiments, as well as useful extensions to the basic methods. Finally, a method for tracking a view point relative to a moving object based only on 2D shape information is investigated and applied in experiments with some success. Future directions as well as limits of a purely outline based method are examined.

Zusammenfassung

Gestalt, insbesondere die Gestalt von Objektgrenzen, die Objekte in einem Bild vom Hintergrund trennen, ist seit langem Gegenstand der Forschung. Die Bedeutung von Gestaltinformation für Aufgaben wie Objekterkennung und Verfolgung wird im allgemeinen hoch eingeschätzt. Diese Arbeit beschäftigt sich mit Methoden, die im Zusammenhang mit Gestalt von 2D-Kurven verwendet werden. Der Hauptbeitrag ist eine rein 2D-gestaltbasierte Methode, um Blickpunktänderungen eines Beobachters relativ zu einem Objekt zu verfolgen, gegeben eine Sequenz von Bildern. Mehrere Methoden, die in diesem Zusammenhang wichtig sind, werden beleuchtet. Insbesondere sind dies Segmentierungsverfahren, die Konturen aus Bildern liefern, und Repräsentierungen für Gestaltinformation. Klassische Gestaltrepräsentierungen und Methoden sind in kleinerem Umfang enthalten. Speziell eine neuere Möglichkeit in Form einer Gestaltmannigfaltigkeit wird genauer behandelt, inklusive einiger wichtiger Details zur Implementierung und zum Rechnen mit Gestalt. Auf der Segmentierungsseite werden Methoden basierend auf den erfolgreichen Level-Set-Verfahren beschrieben. Diese werden benutzt, um Bilder zu segmentieren und Kurven über eine Sequenz von Bildern zu verfolgen. Dieses Gebiet hat in der Vergangenheit ein starkes Interesse von vielen Seiten auf sich gezogen und wird an vielen Stellen weiterentwickelt. Innerhalb der vorliegenden Arbeit werden alle nötigen Details von Level-Set-Verfahren beschrieben, die für die in Experimenten benutzte Implementierung wichtig sind, ebenso wie nützliche Erweiterungen zu den grundlegenden Verfahren. Schließlich wird eine Methode eingeführt, die die Verfolgung eines Blickpunktes relativ zu einem bewegten Objekt ermöglicht, ohne eine explizite interne 3D-Repräsentierung des Objektes zu verwenden. Einige Experimente zeigen, daß ein solches Verfahren funktionieren kann; Grenzen des Verfahrens und ein Ausblick auf mögliche Erweiterungen werden aufgezeigt.

Acknowledgements

First, I would like to thank Prof. Christoph Schnörr for his expert supervision and for fruitful discussions leading me into several highly interesting topics. I would also like to thank Prof. Anders Heyden and Ketut Fundana for insightful discussions and cooperation.
Christian Schellewald, Annette Stahl, Thomas Schüle, Stefan Weber, and Paul Ruhnau also for helpful discussions and hints. Christian Schellewald, Annette Stahl, Wolfgang Römer, and Stefan Weber for proof-reading and several helpful comments.
Very special thanks go to my father for his tremendous and ongoing support and encouragement. To my sister, for her encouragement and for being a wonderful person.
And finally, to Annette Stahl, Christian Schellewald, Stefan Weber, and Holger and Johanna Herbel for being there to help in very harsh weather.
Thank you.

For Wilhelm, Bettina, and Ursula.

Contents

Notation

I_k	Identity matrix in $\mathbb{R}^{k\times k}$
1_k or $\mathbf{1}_k$	Vector of ones in $\mathbb{R}^k$
$\mathcal{SO}_m$	Special orthogonal group (the group of rotations in $\mathbb{R}^m$)
$\mathrm{Exp}_x(v)$	Exponential map at point x in tangent direction v
$\mathrm{Log}_x(y)$	Inverse exponential map — this is at times also written as $\overrightarrow{xy}$
$T_x(M)$	Tangent space at $x \in M$
$\perp_x(M)$	Normal space at $x \in M$
$E[\cdot]$	Expectation
$\mathbb{S}^2$	The unit sphere in $\mathbb{R}^3$, $\mathbb{S}^2 = \{x \in \mathbb{R}^3 : \sqrt{x^\top x} = 1\}$

Chapter 1

Introduction

1.1 Motivation

One important cue that has been used frequently in the past for object detection, object classification, object tracking, pose estimation, and related tasks, is the *shape* of a *silhouette* or *contour* that separates an object of interest from the background within an image. Several interrelated questions arise when thinking about images and this notion of shape — one is *"How can we extract meaningful silhouettes from an image?"* and leads to the field of *image segmentation*. Another is *"How can we represent shape in a useful manner?"*. There has been much research going on in recent years revolving around possible representations and methods to utilise shape representations for computer vision tasks.
In a classical approach, contours are represented as a set of points on the contour curve. However, an important class of contours can more naturally be described as two-dimensional, simple, closed curves. This has been taken up in the past by a few researchers and spaces of curves have been introduced that allow for calculating distances between two curves and for calculating smooth interpolations.

Now, thinking about images of simple, three-dimensional, rigid objects, it is clear that most objects will look differently from different points of view, and produce different silhouettes. Also, from our everyday experience, we expect that when smoothly turning an object, the object's silhouette will also change more or less smoothly. Another question then could be whether we are able to infer a change of view point relative to the object, only by observing a change in the object contour. This task has been tackled before, utilising three-dimensional object models to internally represent objects; but can this also be done *not* knowing the 3D object model, but only knowing how the object outline looks like from a number of view points?
A related, more speculative question is *"How does the brain do it?"*, so how do humans store information about object outlines and how do humans use

it to deduce three-dimensional pose, or pose change, of an object when confronted with a new outline? Is a brain capable of actually constructing a 3D model and using that, or does it refer to previously seen contours and then infers assumptions about new ones from them? While it seems unlikely that humans use only contour information to infer knowledge about an object, it is certainly a part of the whole.
The importance of contours or *outline-shape* for human object *recognition* has also been assessed in psychological experiments, for example [55, 54, 82, 56], and interestingly, also the recognition performance in conjunction with changes in view point were apparently of some interest to Psychologists [54, 56].

In the scope of this thesis, it was tried to use changing contour information of images of an object moving and specifically rotating in three dimensions, in order to track the view position on a view sphere around the object. A 3D model was not used for internal representation, but a different representation relying on observed shape was employed. For contour extraction and evolution, some aspects of the successful level set based methods for image segmentation have been examined and compiled.

1.2 Related Work

Work related to the specific topics is cited in the right context in their respective chapters.

1.2.1 Object Representation

Edelman and Bülthoff [36], Poggio and Edelman [105], Ullman and Basri [134], Edelman and Weinshall [37] use the idea of interpolation using a finite number of 2D object representations, in the context of object recognition. Poggio and Edelman [105] and Edelman and Bülthoff [36] use radial basis functions to interpolate non-linearly between sets of features from several familiar views of an object, in order to recognise objects, not using 3D models directly. Edelman et al. [36] stress the similarity of object representation using 2D views to human performance.
Ullman and Basri [134] use linear combinations of edge maps of images of an object with known correspondences to model all 3D views of the object under the assumption of orthographic projection.

Some works [105, 36, 37] stress the presence of psychophysical evidence that object representation by a collection of 2D views may be related to the internal representation used in human cognition [35, 10, 11], and there appear to be several experiments suggesting that human brains might work with a finite set of known 2D representations of 3D objects rather than view-independent 3D representations. Edelman and Bülthoff [35] describe experiments where the same input stimuli were used for humans as well as for computational models using 2D-views based object representations. They show a limited range of views of previously unknown computer generated objects to humans for training, and then show different views to the same subjects. They claim that according to their experiments and experiments from others referenced in [35], humans have difficulties generalising views of objects rotated further than about 30 degrees from a known view. That is even though the objects are shown as wire frames. They point out that this behaviour is close to their computer models using interpolation of 2D views. Bülthoff and Edelman [10] also come to that conclusion. These works also divide the theories of object representation into 3D *object centred* and 2D *viewer centred*. Somewhat simplified decision criteria for recognition are stated in [10]. For 3D representations, that is $\|P\,T\,X_{3D} - X_{2D}\| < \theta$ with projection P, transformation T, 3D object X_{3D} and 2D measurement X_{2D}. For 2D representations, there using a linear combination, the decision criterion is $\| \sum_i \alpha_i\, X_i - X\| < \theta$, with known 2D views X_i, measurement X, and weights α_i. While this thesis does not deal with recognition, the latter is somewhat similar to the approach taken to follow a changing contour on a view sphere later in Chapter 5, and the former is closer to works based on 3D internal representations such as [108].

1.2.2 View Point Tracking

On the level of view point or rather pose tracking of rigid objects, there have been many approaches using 3D models as internal representation, such as [45, 71, 121, 8, 108, 114]. Stark and Fuchs [121] use a 3D object representation for pose estimation combined with an active contour model for 2D tracking in the image plane. They use corner points of the contour of projected 3D objects as landmarks, modelling contours with B-splines.
Earlier works include Gennery [45] or Koller et al. [71], both using 3D object models.

Kollnig and Nagel [72] track vehicles in traffic scenes using 3D models. They compute artificial “gradient” images from projections of these models and align those to a given gradient image. Alignment is done by minimising an energy for pose parameters specialised for the situation at hand (position relative to street plane, vehicle orientation, velocity, and angular velocity).

Brox et al. [8] use a 3D object model to combine the tasks of image segmentation and pose estimation. Even though their work does not primarily focus on tracking an object pose, they nevertheless present experiments using image sequences. For segmentation, they use a level set formulation of a statistical image energy with curve length regularisation, and an additional prior term incorporating information from a 3D pose estimation they propose in their paper. Given a fixed segmentation, they optimise for a 3D pose fitting the current segmentation well, and fixing that in turn optimise for a segmentation. These two steps are iterated to gain both pose and segmen-

tation.
Rosenhahn et al. [108] also propose using a 3D object model, taking up the ideas of [8]. They focus on tracking of pose in image sequences, also using a region based level set segmentation method with the same region terms and prior as [8].
Schmaltz et al. [114] also use 3D models as internal object representation, like in [8, 108]. They do not use a segmentation algorithm, but optimise a region based energy in the style of [8] directly using the projection of the 3D model, without an additional contour evolution.

A survey of 3D tracking methods can be found in [77]. Notice that 3D tracking as such involves all six degrees of freedom of 3D motion (observer's position on the view sphere, camera tilt, and translation), while the methods described later in this thesis concentrate on view sphere position.

1.2.3 Segmentation

In the light of the task of curve extraction, there are relations of this work to image segmentation. By image segmentation, one usually means the subdivision of an image into semantically or otherwise meaningful regions. While this task is done practically continuously by normally developed humans, it has so far proven to be very difficult to do by algorithms running on computers. Therefore, there has been a considerable amount of research in the past in this field.
The *active contours* type of segmentation methods have been very successful in the past, following the original work on *snakes* by Kass et al. [63]. Caselles et al. [14] used an implicit level set representation [97] for curves instead of an explicit representation, and Caselles et al. [15] introduced the *geodesic active contours*, also using level sets. There has been a large amount of work following this direction. While the active contour methods were so far using information about edges in an image, there have also been approaches using region information, most prominently Chan and Vese, and Chan, Sandberg and Vese [130, 131]. Following this, there has been a plethora of work relating to the same basic approach. There has, for instance, been much effort to extend the energy that is minimised in order to do image segmentation. Such extensions include terms incorporating external, prior knowledge about the segmented object or background, for example Cremers et al. [28, 26], Leventon et al. [78], Chen et al. [19], and Cremers [30].
For tracking a 2D contour along an image sequence, a simple region based method using level sets was proposed by Moelich and Chan [93]. Active contours and level set methods have also been used in 3D pose tracking as mentioned in the previous section, introducing prior knowledge from an internal object representation into the curve evolution.
Further, less directly related work includes Etyngier et al. [39], who use Laplacian eigenmaps [6] for embedding a set of training shapes into a low

dimensional euclidean space. They present a method for projecting previously unseen shapes to the submanifold represented by the training samples, in order to model a shape prior for image segmentation.

1.3 Contribution

While the works mentioned in the previous section, and many others, employ 3D models for internal representation of objects, within this thesis an internal representation is considered that uses only 2D shape information of rigid objects from known view points, but not a complete 3D model. This representation approach is also motivated by a suspicion that some Psychologists appear to have: that human brains might make use of contour data in a similar way [54, 56], as mentioned above. This work should therefore, from the perspective of pose, not be considered a competitor to established pose estimation methods (estimation is not even considered, only view point tracking).

In Chapter 2, a few important things from classical, statistical shape analysis are explained, which are partly used in the implementation used for experiments in this thesis. Chapter 3 then moves on to explain some general concepts and a specific space of shapes that had recently been introduced and that is later used as the space containing shape samples from rigid objects. Details are given about implementation issues. Working with outline shape, a way of extracting contours from images needs to be accounted for. There is certainly not one method that can be said to be "best"; noting that there are other ways to obtain contours, this thesis concentrates for this task on the level set segmentation framework. Chapter 4 therefore accumulates several aspects of the powerful level set method for representing segment boundaries. The chapter covers as much as possible that is useful for the task of contour extraction and has been used for experiments in this thesis. Additionally, leads are provided for the reader that can be followed to get into further detail and more current research. Included are also some issues regarding a finite difference implementation. There is still much active research going on in the field of segmentation using *active contours*, in which the level set method has been employed many times. Additionally, in Appendix B a quite recently introduced segmentation method is described that is closely related to a binary region based level set segmentation method covered in Chapter 4, and that under certain circumstances can find a global solution to the respective minimisation problem. This method could be used as a replacement for the conventional region-based segmentation in some situations. In Chapter 5, the newly introduced view point tracking methodology is explained, using the previously described methods and shape space. Utilising the shape description and space from Chapter 3, an object shape representation is developed that does without an explicit 3D model. This representation of a known object is subsequently used in the task of tracking the view point of an observer with respect to the object in an image sequence, based only on object outline. To represent and evolve a contour in an image, the level set segmentation methods described in Chapter 4 are put into action and combined with the object shape representation, in that the latter is included as “prior knowledge” in the contour evolution to keep the evolving contour close to a known shape. Finally, Chapter 6 concludes and the Appendix gives some details which are referenced from the main text.

Chapter 2

Shape and Classical Shape Distances

“ ’What was that?’
’Something red.’
’Where are we?’
’Somewhere green.’
’Shapes’, muttered Arthur, ’I need shapes!’ ” (Douglas Adams)

2.1 Defining Shape

When we want to talk about *shape*, the first natural step is to find a definition of shape that captures what we normally mean. The notion of shape has been used in various contexts. The spatial configuration of several features defining an object in an image has been called *shape*, for example Shokoufandeh et al. [3] use configurations of ellipses to capture the shape of objects in two-dimensional images. Crandall et al. [24] introduced simple graph structures to encode configurations of object parts. Others, like [58, 51], use *skeletons* to describe an object’s shape. There is an abundance of literature on using shape information in segmentation, detection, and recognition of objects. This chapter is concerned with shape information contained in object *contours* represented by landmark points, and particularly with some aspects of a classical approach of shape representation and shape distances.

One problem with shape is that in practice, there is not *the* representation of object shape. This chapter accounts for a fairly wide spread theory and introduces concepts common to different approaches to shape, in particular the notion of mean shape.
In this spirit, this chapter first introduces some important ideas from classical, landmark based statistical shape analysis, before Chapter 3 goes on to more recent approaches to modelling shape of two-dimensional curves.

A quite generic, intuitive definition[1] of shape can be made by defining when two shapes are equivalent:

Definition 2.1.1 *(Shape) Two spatial configurations are said to have equivalent* shape *if they are equivalent with respect to an "irrelevance transformation".*

Kendall [64, 57] gives a more specific definition for the case of Euclidean transformations, that appears appropriate in this context:

Definition 2.1.2 *(Shape)* Shape *is all the geometrical information that remains when location, isotropic scale, and rotational effects are filtered out from an object.*

An object here denotes a 2D contour. Notice that it may be arguable whether these definitions should actually be called definitions. However, they do give an intuitively clear idea of what we mean when we say "shape".
When speaking of contours, we usually mean two-dimensional regular curves, that means smooth, differentiable curves $c(t) : I \mapsto \mathbb{R}^2$ with $\dot{c}(t) \neq 0\,\forall t$, defined on an interval $I = [a, b], \mathbb{R} \ni a < b \in \mathbb{R}$. Many times, contours are additionally closed curves, so that $c(a) = c(b)$ and $\dot{c}(a) = \dot{c}(b)$.

In the framework of *statistical shape analysis* [57, 66], shapes are usually represented by *landmark points*. In case of a contour, these landmark points would typically be points situated on the contour, like in the active shape model from Cootes et al. [22]; however, they may in general just be point clouds not related to a curve at all. These landmark points are conveniently given as *configuration matrices* containing the coordinates of one point per row.

Definition 2.1.3 *(Configuration matrix) A matrix $X \in \mathbb{R}^{k \times m}$ containing k landmark points in m dimensions is called a* configuration matrix.

To remove variance with respect to location from a configuration matrix X', one simply translates all landmark points so that their combined centre of mass coincides with the origin. This can be written compactly with a *centring matrix*

$$C = I_k - \frac{1}{k} 1_k \, 1_k^\top$$

by setting the centred configuration matrix $X = C \cdot X'$. I_k is the identity matrix in k dimensions, 1_k is a k-vector of ones.

Definition 2.1.4 *(Pre-shape) Let $X' \in \mathbb{R}^{k \times m}$ be a configuration matrix and $C = I_k - 1/k \, 1_k \, 1_k^\top$ the centring matrix. Then the landmark points in the rows of the matrix*

$$X = \frac{C\, X'}{\|C\, X'\|}$$

[1]This definition was given by Jan Koenderink at the Workshop on Shape Perception in Human and Computer Vision (SPHCV), Marseille, 2008.

constitute what is called pre-shape. *The used norm is the Frobenius norm.*

A pre-shape is invariant with respect to translation and scale, but not to rotation. Note that in later chapters, the notion of shape and pre-shape will be altered a little.

2.2 Shape Matching and Distance

To be able to compare different shapes, we need a notion of distance between them. There are several possibilities, of which a few will be introduced in this section. Namely, the widely established full Procrustes distance and an affine invariant distance will be discussed.

2.2.1 Procrustes Distance

The Procrustes[2] distance is found by minimising over Euclidean transformations to find the minimal Euclidean distance between two shapes.
The *full Procrustes distance* [48, 57] is defined as

$$d_F(X_1, X_2) = \inf_{\Gamma \in \mathcal{SO}_m, \beta \in \mathbb{R}} ||Z_2 - \beta\, Z_1\, \Gamma||,$$

with Z_i the pre-shape of X_i, β the scaling parameter and Γ a rotation matrix. Notice that despite the fact that the pre-shape representation is invariant to isotropic scale, the scaling parameter is needed to allow for a *relative* scale between Z_1 and Z_2.
The task of finding the transformation parameters of one shape with respect to another is called *ordinary full Procrustes analysis.* Obviously, the full Procrustes distance is invariant to the euclidean transformations rotation, translation, and isotropic scale. This distance is given by

$$d_F(X_1, X_2) = \left(1 - \left(\sum_{i=1}^{m} \lambda_i\right)^2\right)^{\frac{1}{2}} \tag{2.1}$$

with the singular value decomposition $Z_2^\top Z_1 = V\, \Lambda\, U^\top$, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$; the optimal rotation and scale parameters $\hat{\Gamma}, \hat{\beta}$ are given by

$$\hat{\Gamma} = UV^\top \tag{2.2}$$

$$\hat{\beta} = \sum_{i=1}^{m} \lambda_i \tag{2.3}$$

[57, 65, 116, 117]. There are modified versions of the full Procrustes distance, specifically the *partial Procrustes distance* and *Procrustes distance*, which do not optimise for the scale parameter [57]. Figure 2.1 illustrates calculating

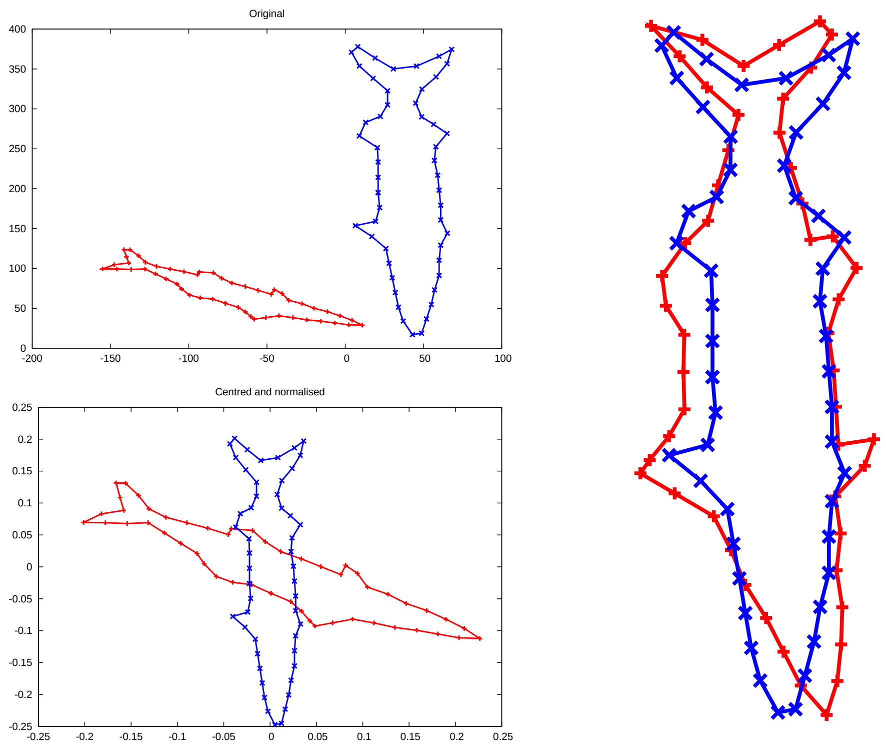


Figure 2.1: Illustration for Procrustes alignment and distance. Top left: Two input configuration matrices X_1, X_2. Bottom left: The centred and normalised matrices Z_1, Z_2. Right: The aligned curves $\hat{\beta}\, Z_1\, \hat{\Gamma}, Z_2$ using (2.2) and (2.3).

the full Procrustes distance. For the two-dimensional case, it turns out that writing the configurations in terms of complex-valued vectors is beneficial for calculating the full Procrustes distance.
Writing two *centred* configurations of k points as $x, y \in \mathbb{C}^k$, the problem at hand is then to minimise

$$\begin{aligned} d^2(x,y) &= \|y - \beta\, e^{i\theta}\, x\|^2 && (2.4) \\ &= y^\star y - \beta\, e^{-i\theta}\, x^\star y - \beta\, e^{i\theta}\, y^\star x + \beta^2\, x^\star x\,. && (2.5) \end{aligned}$$

[2]There is a story as to where the name *Procrustes* came from in [57] on page forty-two.

Deriving with respect to the turning angle θ and setting the derivative to zero gives

$$\begin{aligned} \frac{\partial d^2(x,y)}{\partial \theta} &= i\,\beta\, e^{-i\theta}\, x^\star y - i\,\beta\, e^{i\theta}\, y^\star x = 0 && (2.6)\\ \Rightarrow\qquad & e^{-i\theta}\, x^\star y = e^{i\theta}\, y^\star x && (2.7)\\ \Rightarrow\qquad & \frac{x^\star y}{y^\star x} = e^{2\,i\theta}\,. && (2.8) \end{aligned}$$

Setting $\gamma\, e^{i\phi} := x^\star y$ this becomes

$$\begin{aligned} \frac{e^{i\phi}}{e^{-i\phi}} = e^{2\,i\phi} &= e^{2\,i\theta} && (2.9)\\ \Rightarrow \theta &= \arg(x^\star y)\,. && (2.10) \end{aligned}$$

Next, deriving (2.4) with respect to the scale β and also setting to zero gives

$$\begin{aligned} \frac{\partial d^2(x,y)}{\partial \beta} &= -e^{-i\theta}\, x^\star y - e^{i\theta}\, y^\star x + 2\,\beta\, x^\star x = 0 && (2.11)\\ \Rightarrow\ 2\,\beta\, x^\star x &= e^{i\theta}\, y^\star x + e^{-i\theta}\, x^\star y && (2.12)\\ &= e^{i\theta}\,\gamma\, e^{-i\phi} + e^{-i\theta}\,\gamma\, e^{i\phi} && (2.13)\\ &= 2\,\gamma && (2.14) \end{aligned}$$

since we set $\phi = \theta$ from (2.10). So,

$$\beta = \frac{\gamma}{x^\star x} = \frac{|x^\star y|}{x^\star x}\,. \qquad (2.15)$$

The full Procrustes distance for normalised (notice x, y have not been required to be normalised above) configurations can then be calculated as

$$d(x,y) = \left(1 - \frac{y^\star x\, x^\star y}{x^\star x\, y^\star y}\right)^{\frac{1}{2}}\,. \qquad (2.16)$$

Equation (2.16) will be needed a little further down in Section 2.4.1.

2.2.2 Matching Under Affine Transformations

To see a different shape distance allowing for a wider class of transformations, assume for a moment that we want not only invariance to translation, rotation, and scale, but to the whole set of affine transformations.
Consider we want to find an affine transformation consisting of a linear transformation matrix $A \in \mathbb{R}^{n\times n}$ and translation vector $t \in \mathbb{R}^n$, so that the square error between $X_2 \in \mathbb{R}^{m\times n}$ and $X_1 \cdot A^\top - 1_m \cdot t^\top$ is minimal, where $X_2, X_1 \in \mathbb{R}^{m\times n}$ are configuration matrices like before, containing coordinates of one point per row. That means we have to minimise the trace

$$F(A,t) = tr(D \cdot D^\top) \qquad (2.17)$$

with

$$D = X_2 - (X_1 \cdot A^\top - 1_m \cdot t^\top) \,. \tag{2.18}$$

A solution to the minimisation of (2.17) can be found in Appendix C.1. Figure 2.2 shows an alignment using euclidean transformations found with Procrustes analysis as described above, and one using affine transformations for comparison. One of the contours has been sheared prior to alignment to show the difference more clearly.

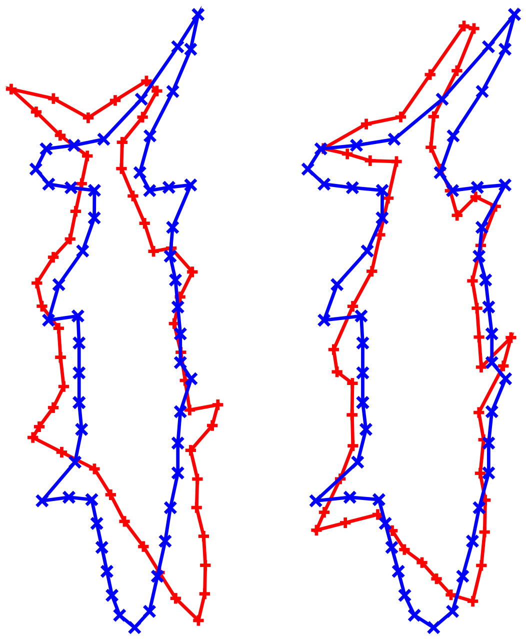


Figure 2.2: Alignments using euclidean transformations (left) and affine transformations. Left: Alignment using Equations (2.2) and (2.3). Right: Alignment using affine transformations minimising (2.17).

2.2.3 Affine Invariant Shape Distance

In order to calculate a *distance* between shapes that is invariant under affine transformations, we need to normalise the shapes appropriately. Notice that for Euclidean transformations, as for the Procrustes distance, the point configurations are normalised with respect to translation by translating each point configuration's origin to its centre of mass, and with respect to scale by dividing by the matrix norm of the centred point configuration. In the affine case, as for example mentioned by Werman and Weinshall [136], we need a different normalisation so that the distance is symmetric. In addition to the translation of the origin to the centre of mass, one applies a *moment normalisation* [136]

$$X' = X \cdot S_X \,. \tag{2.19}$$

$X \in \mathbb{R}^{m\times n}$ is the configuration matrix containing the centred landmark points, and $S_X \in \mathbb{R}^{n\times n}$ is chosen so that $X'^\top \cdot X' = I$, with $I \in \mathbb{R}^{n\times n}$ the identity matrix. That given, one calculates S_X as

$$\begin{aligned} & S_X^\top X^\top X S_X = I \\ \Leftrightarrow\quad & X^\top X = \left(S_X^{-1}\right)^\top \left(S_X^{-1}\right) \\ \Rightarrow\quad & S_X = \left(X^\top X\right)^{-\frac{1}{2}} \end{aligned}$$

and can use

$$\begin{aligned} d(X_2, X_1) &= \inf_{A,t} \left\| X_2 S_{X_2} - (X_1 A^\top - \mathbf{1}t^\top)\right\| \\ &= \inf_{A',t'} \left\| X_1 S_{X_1} - (X_2 A'^\top - \mathbf{1}t'^\top)\right\| = d(X_1, X_2) \end{aligned}$$

as distance between X_1, X_2.
As an aside, it is noted that Begelfor and Werman [5] present a very different representation of affine invariant shape by identifying point configurations with elements of a specific manifold, allowing for calculation of local means and covariances in an elegant manner. This is not followed here any further, as another shape space will be introduced and used later on.

2.3 Registration

So far, it was assumed that pairwise point correspondences between the two shapes are known. Now, suppose we are given two 2D curves and do not know the exact point correspondences except start and end point. Or worse, closed contours without even knowing corresponding starting points. Looking only at the problem of finding a suitable starting point in case of closed curves, the brute force method to find a starting point that minimises a given distance measure is to just try every possible starting point, which increases computation time significantly.
A few researchers have tackled these problems and provided algorithms for matching curves and also to alleviate the computational cost of matching closed curves. Sebastian et al. [118] presented a dynamic programming method for matching two curves, and also present a modification for matching closed curves that only increases the computational complexity from $O(n^2)$ to $O(n^2 \log n)$, instead of $O(n^3)$ for the brute force approach. This method was later also used in [91] and related work, modified to use another metric; this will in another context be described in Chapter 3.

Schmidt, Farin et al. [40, 115] present a related method for which they also give a worst case run time on the order $O(n^2 \log n)$, but claim that on average the run time was lower in their experiments and outperformed [118] for very large point counts.

2.4 Mean Shape

When we calculate shape distances, we will often also want to calculate means of sets of shapes, like for example for k-means type clustering algorithms [81, 7] or for interpolating between several shapes using weighted averages.
The following sections briefly discuss an approach using full Procrustes distance. Following that, the next chapter will introduce more recent approaches to shape representation and also to means on shape spaces.

2.4.1 Full Procrustes Mean

The mean pre-shape μ of a set of pre-shapes $\{Z_1, \dots, Z_n\}$ using full Procrustes distance is estimated using *full generalised Procrustes analysis* [49, 57]. This method augments ordinary full Procrustes analysis for more than two shapes. The aim of generalised Procrustes analysis is to find transformation parameters

$$\{\beta_i^\star, \Gamma_i^\star, t_i^\star\} := \arg \min_{\beta_i, \Gamma_i, t_i} \sum_{i=1}^{n} \left\| (\beta_i \, Z_i \, \Gamma_i + 1_k \, t_i^\top) - \mu \right\|^2 \tag{2.20}$$

with

$$\mu := \frac{1}{n} \sum_{j=1}^{n} (\beta_j \, Z_j \, \Gamma_j + 1_k \, t_j^\top) \,. \tag{2.21}$$

In [57], an iterative algorithm is given for solving (2.20) and finding the average μ for N dimensional data.
For the two dimensional case, which is more interesting in our context, the solution can be found in closed form by writing the point configurations as complex vectors.
Given centred and normalised configurations $\{z_1, \dots, z_n\}$, find an average $\hat{\mu}$ for which

$$\hat{\mu} = \arg \min_{\mu} \sum_{i=1}^{n} d^2(z_i, \mu) \,. \tag{2.22}$$

Using the formula for full Procrustes distance for complex configurations (2.16), the right hand sum is

$$\sum_{i=1}^{n} d^2(z_i, \mu) = \sum_{i=1}^{n} \left(1 - \frac{\mu^\star z_i z_i^\star \mu}{\mu^\star \mu} \right) = n - \mu^\star S \mu \, \frac{1}{\mu^\star \mu} \,, \tag{2.23}$$

so one looks for a normalised μ that maximises $\mu^\star S\mu$ with

$$S = \sum_{i=1}^{n} z_i z_i^\star \,, \tag{2.24}$$

$$\hat{\mu} = \arg \max_{\|\mu\|=1} \mu^\star S\mu \,, \tag{2.25}$$

which can be found by solving

$$\hat{\mu} = \arg \max_{\mu} F(\mu) \tag{2.26}$$

with

$$F(\mu) = \mu^\star S\mu - \lambda\,(\mu^\star\mu - 1) \tag{2.27}$$

by setting

$$\frac{\partial F}{\partial \mu} \;=\; S\,\mu - \lambda\,\mu = 0 \tag{2.28}$$

$$\Rightarrow S\,\mu \;=\; \lambda\,\mu\,, \tag{2.29}$$

so $\hat{\mu}$ is the eigenvector corresponding to the largest eigenvalue of S. Note that $\hat{\mu}$ is a pre-shape. Figure 2.3 shows a collection of similar, noisy curves and a mean $\hat{\mu}$.

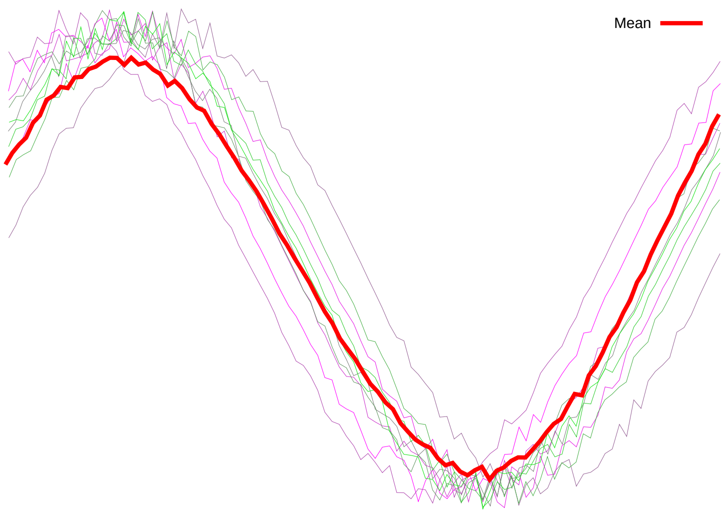


Figure 2.3: Full Procrustes mean of a collection of curves. The mean is shown in thick, red lines.

Full Procrustes Mean with Cyclic Permutations

If one is dealing with point configurations of closed contours and if the starting point correspondences of $\{z_1, \ldots, z_n\}$ are not known, calculating the mean becomes more computation-intensive, because the minimisation must be carried out for every possible choice of starting point for each of the point configurations.
Therefore, Cremers [30, p. 41] proposes an iterative method for finding an alignment, which alleviates the computational burden. No proof of convergence for that method is given, but nevertheless it seems to work fine in experiments.

Chapter 3

Spaces of Elastic Shape

"Finally, the shapes were becoming proper shapes, instead of vague and wobbling shapeless shapes." (Douglas Adams)

3.1 Motivation

So far, in classical, statistical shape analysis, the considered space of shapes is the space consisting of all elements

$$\{Z\,\Gamma\,:\,Z \in \mathbb{R}^{k\times m} \text{ is a pre-shape and } \Gamma \in \mathcal{SO}_m\}\,,$$

equipped with the inner product $\mathrm{tr}(X^\top Y)$ used to define the full Procrustes distance. While this is useful in many situations and has been used extensively in the past, other representations of shape and other metrics have been introduced and investigated more recently. The shape representation so far has also assumed the knowledge of *landmark points*. This is sensible in many applications where landmark points can either be found automatically or where human expert knowledge is available. When looking at shape in the confines of this thesis, 2D contours of objects are examined, which can much more naturally be described by *curves*. So from a more theoretical point of view, it makes sense to represent shape as *functions* describing smooth curves instead of as a finite number of landmark points. In this light, recent approaches from Klassen, Mio, Srivastava, Joshi, and others [38, 90, 4, 92, 91] have been taken into account. Object contours are represented as differentiable curves $\alpha\,:\,\mathbb{R} \mapsto \mathbb{R}^2$, $\dot{\alpha}(s) \neq 0$. A curve $\alpha(s)$ is normalised with respect to length, and described by a pair of functions: an *angle function* $\Theta(s)$ and a *log speed function* $\Phi(s)$. The two-dimensional plane is identified with the complex number plane. The angle function defines the angle of the curve tangent with the positive real axis of the complex number plane, while the log speed function defines the logarithm of the length of the tangent, so that

$$\alpha(s) = \alpha_0 + \int_0^s \underbrace{e^{\Phi(\tau)}\,e^{i\,\Theta(\tau)}}_{\dot{\alpha}}\,d\tau\,,$$

with α_0 an integration constant that is not considered any further. Figure 3.1 illustrates a closed curve with a few velocity vectors. Notice that this

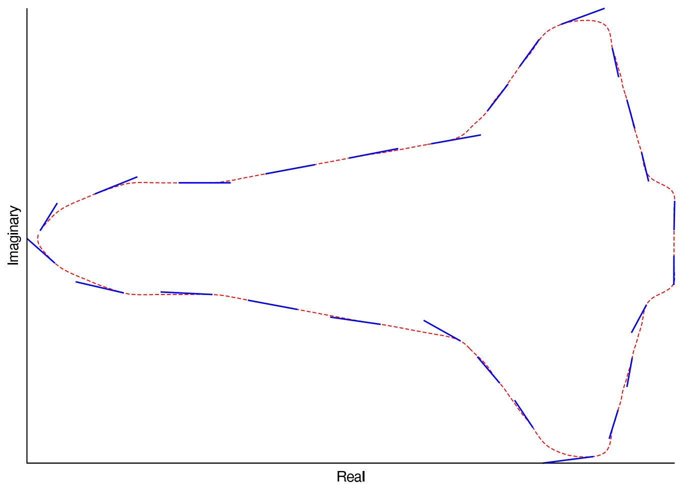


Figure 3.1: Shape representation. A closed curve α is shown in red and a few velocity vectors $\dot{\alpha}$ are indicated in blue.

representation works only for regular curves, since $e^{\Phi(\tau)} \neq 0 \quad \forall \tau$. This representation, together with normalisation with respect to curve length, is already invariant to scale and translation, and it describes curves as continuous functions. To add structure to the space of all curves described by such function pairs (Φ, Θ), one introduces a Riemannian metric, namely an incarnation of the *elastic shape metric* [91] introduced by Younes [143], as will be detailed later on.
In the following section this shape space is described, and a few details needed for an actual implementation and used computational methods are worked out.

3.2 Space of Elastic, Closed Pre-Shapes

This section introduces a space of shapes represented by continuous functions, which also satisfy Definition 2.1.2 with the additional requirement that shape must be *invariant with respect to re-parametrisation*. The name *pre-shape* has a different meaning here than in the classical Definition 2.1.4 in that it denotes shape without parametrisation invariance. This formulation was developed in [90, 91] with a predecessor in [38].

Shapes are represented by pairs of log speed and angle functions. Given a planar regular curve α, i.e. $\alpha : I \mapsto \mathbb{R}^2$ is smooth on the interval I, $\dot{\alpha} \neq 0$.

Let $I = [0,1]$ for convenience. The curve $\alpha(t)$ can then be written as

$$\alpha(t) = \alpha_0 + \int_0^t \dot{\alpha}(\tau)\, d\tau$$

with the velocity vector $\dot{\alpha}$ in complex coordinates

$$\dot{\alpha}(t) = e^{\Phi(t)} e^{i\Theta(t)}$$

and α_0 an integration constant.
Dropping α_0, which only accounts for translation, the *shape* of a curve will be represented by the pair of functions (Φ, Θ). It is easy to see from the above equation that Φ is the logarithm of the speed of a particular parametrisation of α, and Θ is the function giving the angle of the velocity vector with the real axis. Notice that this representation contains both closed and open curves. In order to restrict the pairs (Φ, Θ) to closed curves, one demands that the integral over the velocity vector function is zero:

$$\int_0^1 e^{\Phi(t)} e^{i\Theta(t)}\, dt = 0\,. \tag{3.1}$$

The shape of a curve should be invariant under Euclidean transformations of the curve, that is under rotation, uniform scaling, and translation. Additionally, it should be invariant under different parametrisations of the same curve.
The (Φ, Θ) representation is so far already invariant with respect to translation. The scale and rotational invariances are enforced by taking only those (Φ, Θ) into account which fulfil

$$\int_0^1 e^{\Phi(t)}\, dt = 1 \quad \text{and} \quad \int_0^1 \Theta(t) e^{\Phi(t)}\, dt = \pi\,, \tag{3.2}$$

that means all curves of length 1 and mean turning angle π. This choice is of course arbitrary, it just has to be fixed.
The authors of [91] call the space of pairs (Φ, Θ) fulfilling Equations (3.1) and (3.2) *closed pre-shape space.*

Definition 3.2.1 *Let $I = [0,1]$ and $\alpha : I \mapsto \mathbb{R}^2$ a regular curve. Let $\Phi : I \mapsto \mathbb{R}$ and $\Theta : I \mapsto \mathbb{R}$ so that $\dot{\alpha}(t) = e^{\Phi(t)} e^{i\Theta(t)}$.*
The space $\mathcal{H}$ is the space of all pairs of such functions (Φ, Θ).

Definition 3.2.2 *The* closed pre-shape space $\mathcal{C} \subset \mathcal{H}$ *is defined as the subspace of $\mathcal{H}$ for which Equations (3.1) and (3.2) hold:*

$$\mathcal{C} := \left\{ (\Phi, \Theta) \in \mathcal{H} : \begin{array}{ll} \int_0^1 e^{\Phi(t)} e^{i\Theta(t)}\, dt = 0 & \textit{(closure)} \\ \int_0^1 e^{\Phi(t)}\, dt = 1 & \textit{(scale)} \\ \int_0^1 \Theta(t) e^{\Phi(t)}\, dt = \pi & \textit{(rotation)} \end{array} \right\}.$$

The space $\mathcal{C}$ is called pre-shape and not shape space because its elements are still not invariant under the action of re-parametrisation. A re-parametrisation of a curve $\alpha(t)$ is expressed with an orientation preserving diffeomorphism[1] $\gamma : I \mapsto I$, so that the curve becomes $\beta(t) = \alpha(\gamma(t))$. The velocity then becomes

$$\dot{\alpha}\left(\gamma(t)\right) = \dot{\gamma}(t) e^{\Phi(\gamma(t))} e^{i\Theta(\gamma(t))} = e^{\Phi(\gamma(t)) + \log(\dot{\gamma}(t))} e^{i\Theta(\gamma(t))} \,. \tag{3.3}$$

In a pre-shape space, if (Φ_1, Θ_1) and (Φ_2, Θ_2) define two pre-shapes that differ only by parametrisation, then in a *shape space* they should both identify the same element. This means that the action of re-parametrisations needs to be removed from the closed pre-shape space $\mathcal{C}$ in order to yield a closed shape space.
A way to incorporate parametrisation is mentioned in [91]. In the calculation of the inverse exponential map, a diffeomorphism is included directly in the energy (3.29) to be minimised for that purpose. The calculation of the inverse exponential or Log map will be described further in Section 3.2.4.
Since incorporating parametrisation adds a significant computational cost [91], it is not done here, but instead the alternative solution is adopted, which is to re-parameterise one pre-shape with respect to the other using a matching algorithm prior to computing geodesics. This is explained in more detail in Section 3.2.5.
For practical computations, a discrete version of the closed pre-shape space will also be needed, where functions are replaced by N-vectors:

Definition 3.2.3 *Given $(\Phi, \Theta) \in \mathcal{H}$, let $\tilde{\Phi}, \tilde{\Theta} \in \mathbb{R}^N$ with*

$$t_1 = 0 < t_2 < \cdots < t_N = 1\,,$$

$$t_{i+1} - t_i = \Delta t = const \quad \forall\; i \in \{1, \ldots, N-1\}\,,$$

and with $\tilde{\Phi}_i = \Phi(t_i)$ and $\tilde{\Theta}_i = \Theta(t_i)$. The space of all $(\tilde{\Phi}, \tilde{\Theta})$ is called $\mathcal{H}_N$, the discrete pre-shape space. Analogously, $\mathcal{C}_N$ is the discrete closed pre-shape space.

Let us from now on write (Φ, Θ) instead of $(\tilde{\Phi}, \tilde{\Theta})$ to ease notation wherever ambiguities are unlikely.

[1]A diffeomorphism is a differentiable, bijective map f for which the inverse map f^{-1} is also differentiable.

3.2.1 Riemannian Metric

The inner product on $\mathcal{H}$ that has been introduced in [90],

$$\langle (h_1, f_1), (h_2, f_2) \rangle_{(\Phi,\Theta)} = a \int_0^1 h_1(t)\, h_2(t)\, e^{\Phi(t)}\, dt + b \int_0^1 f_1(t)\, f_2(t)\, e^{\Phi(t)}\, dt\,, \quad (3.4)$$

is applied to elements (h_i, f_i), $i \in \{1, 2\}$ of the tangent space $T_{(\Phi,\Theta)}(\mathcal{H})$. See the beginning of the next Section 3.2.2 for a quick recap.
The idea of an *elastic* shape metric has been established in [143] and has been further discussed by other researchers, predominantly Michor, Mumford [88, 89]. The metric induced by the inner product (3.4) introduces a certain steerability of stretching versus bending. a is called the *tension*, and b the *rigidity* coefficients. [90, 91]. By choice of these coefficients, the metric can be adjusted to put more weight on stretching or on bending when comparing two elements.
Consider in the following a discretised version of this inner product on $T_{(\Phi,\Theta)}(\mathcal{H}_N)$,

$$\langle (h_1, f_1), (h_2, f_2) \rangle_{(\Phi,\Theta)} = \frac{a}{N} \sum_{i=1}^{N} h_{1,i}\, h_{2,i}\, e^{\Phi_i} + \frac{b}{N} \sum_{i=1}^{N} f_{1,i}\, f_{2,i}\, e^{\Phi_i}\,. \quad (3.5)$$

3.2.2 Tangents

For a k-dimensional submanifold of $\mathbb{R}^n$ such as $\mathcal{C}_N$, the tangent space at $x \in \mathcal{C}_N$ is defined as $T_x(\mathcal{C}_N) := Df_u(T_u(\mathbb{R}^k))$ for a parametrisation $f : U \mapsto \mathcal{C}_N$, $f(u) = x$. The differential Df of a differentiable map f is a map $Df_x : T_x(\mathbb{R}^k) \mapsto T_{f(x)}(\mathbb{R}^n)$.
$T_x(\mathcal{C}_N)$ does not depend on the parametrisation f [73]. The tangent space $T_u(\mathbb{R}^k)$ is formally defined as an element of the *tangent bundle* $T(\mathbb{R}^k) := \mathbb{R}^k \times \mathbb{R}^k$ by $T_u(\mathbb{R}^k) := \{u\} \times \mathbb{R}^k$ [73]. The tangent bundle of $\mathcal{C}_N$ is again defined as $T(\mathcal{C}_N) := \bigcup_{x \in \mathcal{C}_N} T_x(\mathcal{C}_N)$, the union of all tangent spaces. For more detail, see e.g. [73].

The tangent space at any element of the linear space $\mathcal{H}$ is $\mathcal{H}$ itself. From this point on, we are working in the spaces of discretised function pairs and pre-shapes, $\mathcal{H}_N$ and $\mathcal{C}_N$. For practical, approximative calculations of geodesics, we will need to project an element from the surrounding space $\mathcal{H}_N$ to a tangent space $T_x(\mathcal{C}_N)$. This projection is described in detail in the following sub-section.

Projection of Elements from $\mathcal{H}_N$ to $T_x(\mathcal{C}_N)$

In order to project $(h, f) \in \mathcal{H}_N$ to $T_{(\Phi,\Theta)}(\mathcal{C}_N)$, define the map

$$G = (G^1, G^2, G^3, G^4) : \mathcal{H}_N \mapsto \mathbb{R}^4\,,$$

the continuous version $\mathcal{G}$ of which reads

$$\begin{aligned}
\mathcal{G}^1(\Phi,\Theta) &= \int_I e^{\Phi(t)}\,dt & (3.6)\\
\mathcal{G}^2(\Phi,\Theta) &= \int_I \Theta(t) e^{\Phi(t)}\,dt & (3.7)\\
\mathcal{G}^3(\Phi,\Theta) &= \int_I \cos(\Theta(t)) e^{\Phi(t)}\,dt & (3.8)\\
\mathcal{G}^4(\Phi,\Theta) &= \int_I \sin(\Theta(t)) e^{\Phi(t)}\,dt\,. & (3.9)
\end{aligned}$$

Replacing the integrals by sums, the discrete versions are

$$\begin{aligned}
G^1(\Phi,\Theta) &= \frac{1}{N}\sum_{i=1}^{N} e^{\Phi_i} & (3.10)\\
G^2(\Phi,\Theta) &= \frac{1}{N}\sum_{i=1}^{N} \Theta_i e^{\Phi_i} & (3.11)\\
G^3(\Phi,\Theta) &= \frac{1}{N}\sum_{i=1}^{N} \cos(\Theta_i) e^{\Phi_i} & (3.12)\\
G^4(\Phi,\Theta) &= \frac{1}{N}\sum_{i=1}^{N} \sin(\Theta_i) e^{\Phi_i}\,. & (3.13)
\end{aligned}$$

G^1 and G^2 express Equations (3.2) while the second two stem from (3.1) because $e^{i\,x} = \cos(x) + i\,\sin(x)$.
Using this map, the closed pre-shape space $\mathcal{C}_N$ can be defined as the level set $G^{-1}(1, \pi, 0, 0)$. The discrete maps (3.10)–(3.13) will differ when other approximations for the integrals are used.
The gradient of $\mathcal{G}^i(\Phi,\Theta)$ is defined by

$$\langle \text{grad}\, \mathcal{G}^i(\Phi,\Theta), (h,f)\rangle_{(\Phi,\Theta)} = D_{(h,f)}\mathcal{G}^i(\Phi,\Theta)\,,$$

with $D_{(h,f)}\mathcal{G}^i(\Phi,\Theta)$ denoting the directional derivative at (Φ,Θ) in the direction (h, f), see for example [73]. Calculating the directional derivatives for (3.6)–(3.9) (see also Appendix D.1),

$$\begin{aligned}
\langle \text{grad } \mathcal{G}^1(\Phi,\Theta),(h,f)\rangle_{(\Phi,\Theta)} &= \int_I e^{\phi(\tau)} h(\tau)\, d\tau \\
\langle \text{grad } \mathcal{G}^2(\Phi,\Theta),(h,f)\rangle_{(\Phi,\Theta)} &= \int_I e^{\phi(\tau)} \left[f(\tau) + h(\tau)\theta(\tau)\right]\, d\tau \\
\langle \text{grad } \mathcal{G}^3(\Phi,\Theta),(h,f)\rangle_{(\Phi,\Theta)} &= \int_I e^{\phi(\tau)} \left[\cos(\theta(\tau))h(\tau) - f(\tau)\sin(\theta(\tau))\right]\, d\tau \\
\langle \text{grad } \mathcal{G}^4(\Phi,\Theta),(h,f)\rangle_{(\Phi,\Theta)} &= \int_I e^{\phi(\tau)} \left[\cos(\theta(\tau))f(\tau) + h(\tau)\sin(\theta(\tau))\right]\, d\tau\,,
\end{aligned}$$

one then completes the left hand sides to

$$\text{grad } \mathcal{G}^1(\Phi(t),\Theta(t)) = \left(\frac{1}{a}, 0\right) \tag{3.14}$$

$$\text{grad } \mathcal{G}^2(\Phi(t),\Theta(t)) = \left(\frac{\Theta(t)}{a}, \frac{1}{b}\right) \tag{3.15}$$

$$\text{grad } \mathcal{G}^3(\Phi(t),\Theta(t)) = \left(\frac{\cos\Theta(t)}{a}, -\frac{\sin\Theta(t)}{b}\right) \tag{3.16}$$

$$\text{grad } \mathcal{G}^4(\Phi(t),\Theta(t)) = \left(\frac{\sin\Theta(t)}{a}, \frac{\cos\Theta(t)}{b}\right). \tag{3.17}$$

The gradients of $\mathcal{G}^i$ in equations (3.14)–(3.17) span the normal space of the space of continuous closed pre-shapes, $\mathcal{C}$, at (Φ,Θ).
Similarly, the gradient of the discrete map G in equations (3.10)–(3.13) is then

$$\text{grad } G_1(\Phi,\Theta) = \left(\frac{1}{a}, 0\right) \tag{3.18}$$

$$\text{grad } G_2(\Phi,\Theta) = \left(\frac{\Theta}{a}, \frac{1}{b}\right) \tag{3.19}$$

$$\text{grad } G_3(\Phi,\Theta) = \left(\frac{\cos\Theta}{a}, -\frac{\sin\Theta}{b}\right) \tag{3.20}$$

$$\text{grad } G_4(\Phi,\Theta) = \left(\frac{\sin\Theta}{a}, \frac{\cos\Theta}{b}\right). \tag{3.21}$$

The calculation can be found in Appendix D.1. Note that 1, 0, and the values of the trigonometric operators in the above equations (3.18)–(3.21) are vectors. Also mind that these gradients may vary when using other discretisations for the integrals than (3.10)–(3.13).

To project from $(h,f) \in \mathcal{H}_N$ to $T_{(\Phi,\Theta)}\mathcal{C}_N$, one first finds an orthonormal basis of the normal space $\perp_{(\Phi,\Theta)}(\mathcal{C}_N)$ spanned by the gradients[2] of G_1,G_2,G_3,G_4, and then simply subtracts the normal components from (h,f).

Algorithm 1 Projection from $(h, f) \in \mathcal{H}_N$ to $T_{(\Phi,\Theta)}(\mathcal{C}_N)$. The QR decomposition $A = QR$ gives an orthonormal matrix Q that spans the same space as A [47]. In fact, any orthogonalisation would do.

```
Require: $(h, f) \in \mathcal{H}_N$, $(\Phi, \Theta) \in \mathcal{C}_N$
1: procedure ProjectToTCn($(h, f), (\Phi, \Theta)$)
2:     Calculate the $QR$ decomposition of
           $\{\text{grad } G_1(\Phi,\Theta), \text{grad } G_2(\Phi,\Theta), \text{grad } G_3(\Phi,\Theta), \text{grad } G_4(\Phi,\Theta)\}$
       using the inner product (3.5). This yields an orthonormal basis
       $\{e_1, e_2, e_3, e_4\}$ of the normal space $\perp_{(\Phi,\Theta)}(\mathcal{C}_N)$.
3:     Calculate the projection by subtracting the normal component:
           $\Pi_{(\Phi,\Theta)}(h, f) = (h, f) - \sum_{i=1}^{4} \langle (h, f), e_i(\Phi,\Theta)\rangle_{(\Phi,\Theta)} e_i(\Phi,\Theta)\,.$
4:     return $\Pi_{(\Phi,\Theta)}(h, f)$
5: end procedure
```

This is detailed in Algorithm 1. For implementation, we use the *modified Gram-Schmidt* algorithm [47] for computing the QR decomposition for orthogonalisation, see Algorithm 2. The inner products used in the algorithm were replaced by the inner product (3.5).

3.2.3 Projecting from $\mathcal{H}_N$ to $\mathcal{C}_N$

Like for elements of the tangent spaces, elements of $\mathcal{C}_N$ may in the course of numerical computations leave the actual manifold $\mathcal{C}_N$. They will in that case be projected back to $\mathcal{C}_N$ by Algorithm 3, which is further described in this sub-section.

Recalling that for any $c \in \mathcal{C}_N$ it must hold that $G(c) = (1, \pi, 0, 0)$, a root of the residual vector $r(c) = (1, \pi, 0, 0) - G(c)$ is sought using Newton's method, restricted to changes in normal direction at c. To see how that works, consider

$$F(c) = (1, \pi, 0, 0) - G(c)$$

of which we seek a nearby root. The Jacobi matrix of F is then

$$F'(c) = J^F(c) = -J^G(c)$$

with the i-th row of $J^G(c)$ given by

$$J_i^G(c) = \text{grad } G_i(c) \quad i = 1, \ldots, 4\,.$$

[2]For the normal space it must hold that $\langle n, v\rangle = 0$ for each $n \in \perp_{(\Phi,\Theta)}(\mathcal{C}_N)$ and $v \in T_{(\Phi,\Theta)}(\mathcal{C}_N)$.

Algorithm 2 Upper algorithm: The modified Gram-Schmidt algorithm taken from [47]. The input is a matrix $A \in \mathbb{R}^{m\times n}$ which is orthogonalised to yield $A = QR$, with Q orthonormal and R upper triangular. The colon operator denotes the usual slicing operator for vectors and matrices. Lower algorithm: Rewritten Gram-Schmidt algorithm using elements of $\mathcal{H}_{m/2}$ and the inner product (3.5). Note that here, $A = ((\Phi_1, \Theta_1), \dots, (\Phi_n, \Theta_n))$ is a tuple where each element is an element of $\mathcal{H}_{\frac{m}{2}}$.

```
1: for $k = 1$ to $n$ do
2:     $R(k,k) = \|A(1:m,k)\|$
3:     $Q(1:m,k) = A(1:m,k)/R(k,k)$
4:     for $j = k+1$ to $n$ do
5:         $R(k,j) = Q(1:m,k)^\top A(1:m,j)$
6:         $A(1:m,j) = A(1:m,j) - Q(1:m,k)R(k,j)$
7:     end for
8: end for
```

```
1: for $k = 1$ to $n$ do                                ▷ $n = 4$ for $\mathcal{C}_N$
2:     $R(k,k) = \|A_k\|_{(\Phi,\Theta)}$
3:     $Q_k = A_k/R(k,k)$
4:     for $j = k+1$ to $n$ do
5:         $R(k,j) = \langle Q_k, A_j\rangle_{(\Phi,\Theta)}$
6:         $A_j = A_j - Q_k\, R(k,j)$
7:     end for
8: end for
```

The Newton method would then give

$$\langle J_i^G(c_k), \overrightarrow{c_k c_{k+1}}\rangle_{c_k} = F_i(c_k)\,.$$

Restricting to the normal space, use instead

$$J^G(c_k)\,(B(c_k)\,(x_{k+1} - x_k)) = F(c_k)\,,$$

where $\mathbb{R}^{2\,N\times 4} \ni B(c) = (\text{grad } G_1(c), \text{grad } G_2(c), \text{grad } G_3(c), \text{grad } G_4(c))$ contains a normal space basis in its columns. Seeing that the columns of $B(c)$ are just the rows in $J^G(c)$, set

$$J^\perp_{ij}(c) = \langle \text{grad } G_i(c), \text{grad } G_j(c)\rangle_c\,,$$

using the inner product on $\mathcal{H}_N$. Then solve

$$J^\perp(c_k)\,x = F(c_k)$$

for $x \in \mathbb{R}^4$ and seeing that

$$\sum_{j=1}^{4} x_j \operatorname{grad} G_j(c_k) = c_{k+1} - c_k \,, \tag{3.22}$$

$$\Rightarrow \quad c_{k+1} = c_k + \sum_{j=1}^{4} x_j \operatorname{grad} G_j(c_k) \,. \tag{3.23}$$

Algorithm 3 Projection from $\mathcal{H}_N$ to $\mathcal{C}_N$ using Newton's method restricted to the normal spaces.

```
Require: $c = (\Phi, \Theta) \in \mathcal{H}_N$ and $\varepsilon > 0$ fixed
 1: procedure ProjectToCn($c$)
 2:     $r \leftarrow \infty$
 3:     while $\|r\| > \varepsilon$ do
 4:         $r \leftarrow (1, \pi, 0, 0) - G(c)$
 5:         Compute Jacobi matrix $J^{\perp}_{ij} = \langle \operatorname{grad} G_i(c), \operatorname{grad} G_j(c) \rangle_c$
 6:         Solve $J^{\perp} \cdot x = r$ for $x$
 7:         $c \leftarrow c + \sum_i x_i \cdot \operatorname{grad} G_i(c)$
 8:     end while
 9:     return c
10: end procedure
```

3.2.4 Geodesics and Geodesic Distances

Given two points p_1, p_2 in a Riemannian manifold M equipped with the inner product $g_p(v, w) = \langle v, w \rangle_p$, a path $\gamma(t) \subset M$ with

$$t \in [t_1, t_2] \,, \quad \gamma(t_1) = p_1 \,, \quad \gamma(t_2) = p_2$$

is called a *geodesic* connecting p_1 and p_2 if it has zero tangential acceleration everywhere. Intuitively, this means that a point following γ moves with constant velocity with respect to the structure of the manifold.
For any point $p_1 \in M$, $v \in T_{p_1}(M)$ there exists an $\varepsilon > 0$ and exactly one geodesic $\gamma_{p_1,v}(t)$, $t \in (-\varepsilon, \varepsilon)$ with

$$\gamma_{p_1,v}(0) = p_1 \,, \quad \dot{\gamma}_{p_1,v}(0) = v \,.$$

Given p_1 and v,

$$\operatorname{Exp}_{p_1}(v) := \gamma_{p_1,v}(1)$$

is called the *exponential map* at p_1 in the tangential direction v. Similarly, we call the inverse exponential map $\operatorname{Log}_{p_1}(p_2) := \operatorname{Exp}^{-1}_{p_1}(p_2) = v$ the logarithmic map or log map, assuming that $\operatorname{Exp}_{p_1}(v) = p_2$. We thereby keep an analogy

to the usual log operator and follow the notation of Pennec et al. [140], Fletcher et al. [42], and possibly others. For the general concepts, refer to books such as [33, 34, 73].
The *geodesic distance* between two points $p_1, p_2 \in M$ is the length of a shortest geodesic connecting the two points. Note that only if the definition domain of all geodesics can be extended to $\mathbb{R}$, a geodesic distance can be computed for all point pairs. We assume that the manifolds we consider have this property called *geodesic completeness* (see Definition 5.3.3). This implies the absence of any boundary or singular points on M.

Computing the Exponential Map

Geodesics in $\mathcal{C}_N$ are computed as follows. Given an element $x \in \mathcal{C}_N$ and $v \in T_x(\mathcal{C}_N)$, the geodesic $\gamma_{x,v}(t)$ is calculated by numerically approximating an infinitesimal step in the direction of the geodesic by adding a small fraction of the tangent vector v to x and projecting the result back to $\mathcal{C}_N$ yielding x'; see Figure 3.2 for a simple illustration. The tangent v is parallel transported to the updated x'; this procedure is detailed on in the next sub-section. The resulting transported vector is then projected to $T_{x'}(\mathcal{C}_N)$ and scaled back to the original tangent vector's length to remove any artificially added acceleration — recall that a point moving along a geodesic is non-accelerated. The procedure is repeated until the resulting path has the requested length. The exponential map is then calculated by using the above approximation to calculate $\mathrm{Exp}_p(v) = \gamma_{x,v}(1)$. Algorithm 4 summarises this procedure.

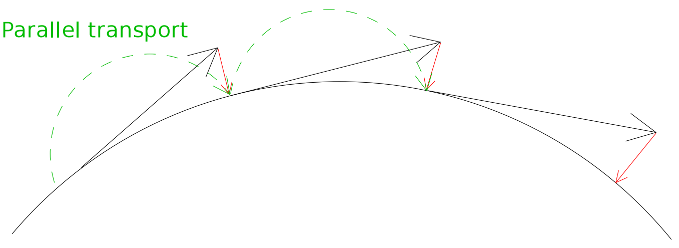


Figure 3.2: Illustration of the approximative computation of a geodesic path on $\mathcal{C}_N$. The steps are 1. take an infinitesimal step (black arrows), 2. project back to $\mathcal{C}_N$ (red arrows), 3. transport tangent in parallel to geodesic (green arcs).

Approximating Parallel Transport. From the requirement that a geodesic curve $\gamma(t) = (u^1(t), \dots, u^n(t))$ must have zero geodesic curvature, $\nabla_{\dot\gamma}\dot\gamma = 0$, follows the system of differential equations

$$\ddot{u}^k(t) + \sum_{i,j} \dot{u}^i(t)\, \dot{u}^j(t)\, \Gamma^k_{ij}(\gamma(t)) = 0 \quad \forall k\,, \tag{3.24}$$

Algorithm 4 Numerical approximation of a geodesic on $\mathcal{C}_N$. The used projection procedures are defined in Algorithms 3 and 1.

```
Require: $0 < \varepsilon \ll 1$
 1: procedure GeodesicApprox(x,v,t)
 2:     $x' \leftarrow x$
 3:     $T \leftarrow 0$
 4:     while $T < t$ do
 5:         $x' \leftarrow$ ProjectToCn($x' + \varepsilon\, v$)
 6:         $T \leftarrow T + \varepsilon$
 7:         Parallel transport $v$ along geodesic using Equation (3.28)
 8:         $v \leftarrow$ ProjectToTCn($v, x'$).
 9:     end while
10: end procedure
```

see for example [73, 34]. Γ^k_{ij} denote the Christoffel symbols. For each component of the discrete inner product (3.5), these must hold. So writing the inner product for one component $(\phi, \theta) = (\Phi_i, \Theta_i)$ as

$$\langle x, y \rangle_{\phi,\theta} = a\, e^{\phi}\, x_1\, y_1 + b\, e^{\phi}\, x_2\, y_2\,,$$

the local representation of the metric is

$$g = \begin{pmatrix} a\, e^{\phi} & 0 \\ 0 & b\, e^{\phi} \end{pmatrix}. \tag{3.25}$$

The Christoffel symbols can then be calculated with

$$\Gamma^k_{ij} = \frac{1}{2} \sum_{\ell} g^{k\ell} \left(\frac{\partial}{\partial u^j} g_{i\ell} + \frac{\partial}{\partial u^i} g_{j\ell} - \frac{\partial}{\partial u^\ell} g_{ij} \right), \tag{3.26}$$

where $u^1 = \phi$, $u^2 = \theta$ and $g^{k\ell}$ denotes $(g^{-1})_{k\ell}$. Doing this by hand or with a computer algebra program as shown in Appendix D.2 yields

$$\Gamma^1_{11} = \Gamma^2_{12} = \Gamma^2_{21} = \frac{1}{2}\,, \quad \Gamma^1_{22} = -\frac{b}{2\,a}\,, \tag{3.27}$$

the other four symbols are zero. Using these in the differential equations for the geodesic (3.24), one gets for the geodesic $\gamma(t) = (u^1(t), u^2(t))$

$$\ddot{u}^1 + \frac{1}{2} \left(\dot{u}^1\right)^2 - \frac{b}{2\,a} \left(\dot{u}^2\right)^2 = 0$$

and

$$\ddot{u}^2 + \dot{u}^1\, \dot{u}^2 = 0\,.$$

For the numerical approximation, this leads to an update of $(h, f) \in T_{(\Phi,\Theta)}(\mathcal{C}_N)$ in the direction of the geodesic $\gamma_{(\Phi,\Theta),(h,f)}$ of

$$(h, f)_{\text{updated}} = (h, f) - \varepsilon \left(\left(\frac{h^2}{2} - \frac{b\, f^2}{2\, a} \right), h\, f \right) \tag{3.28}$$

where the products in the right hand term are element-wise products.

Computing the Inverse Exponential Map

The logarithmic map $\mathrm{Log}_{p_0} : \mathcal{C}_N \mapsto T_{p_0}(\mathcal{C}_N)$ is the inverse of the exponential map at p_0. In order to find a geodesic connecting two given points $p_0, p_1 \in \mathcal{C}_N$, one can compute a tangent vector $\overrightarrow{p_0 p_1} = (h^\star, f^\star) = \mathrm{Log}_{p_0}(p_1)$. In [91], it is proposed to calculate $(h^\star, f^\star)$ by minimising a functional

$$E(h,f) = \|\mathrm{Exp}_{p_0}(h,f) - p_1\|^2 \tag{3.29}$$

for $(h,f) \in T_{p_0}(\mathcal{C}_N)$, as had already been done in [135, 38]. We use gradient descent:

$$(h,f)_{i+1} = (h,f)_i - \varepsilon \cdot \nabla E(h_i, f_i)\,.$$

In order to compute a gradient of E, define a linear map L projecting from $\mathcal{H}_N$ to the normal space at $p_0 = (\Phi_0, \Theta_0)$, using the fact that grad $G^i(p)$ spans the normal space at p:

$$L(h,f) := \begin{pmatrix} \langle (h,f), \mathrm{grad}\ G^1(p_0)\rangle_{p_0} \\ \langle (h,f), \mathrm{grad}\ G^2(p_0)\rangle_{p_0} \\ \langle (h,f), \mathrm{grad}\ G^3(p_0)\rangle_{p_0} \\ \langle (h,f), \mathrm{grad}\ G^4(p_0)\rangle_{p_0} \end{pmatrix}.$$

We can write $L(h,f)$ as a matrix $L \in \mathbb{R}^{4\times 2\cdot N}$ by setting

$$\begin{aligned} L_{i,j} &= \frac{1}{N} \cdot a \cdot e^{\Phi_{0,j}} \cdot \big(\mathrm{grad}\ G^i(p_0)\big)_j \\ L_{i,N+j} &= \frac{1}{N} \cdot b \cdot e^{\Phi_{0,j}} \cdot \big(\mathrm{grad}\ G^i(p_0)\big)_{N+j} \end{aligned}$$

for $i \in \{1,\ldots,4\}$ and $j \in \{1,\ldots,N\}$.
L maps to the normal space at p_0, so that $\ker(L)$, the null space of L, is the tangent space at p_0. We use the singular value decomposition to find an orthonormal basis of $\ker(L)$. Let $L = U \cdot S \cdot V^\top$ be the singular value decomposition of L. Assuming the singular values on the diagonal of S are sorted in descending order, take $B = (V_5, \ldots, V_{2\cdot N})$, the last $2\cdot N - 4$ columns of V, the singular values of which are zero. The columns of B are then an orthonormal basis of $\ker(L)$.
Now, $(h,f) \in T_{p_0}(\mathcal{C}_N)$ can be represented as $(h,f) = B \cdot x$ with $x \in \mathbb{R}^{2\cdot N-4}$. One then estimates the partial derivatives needed for the gradient of E as

$$\frac{\partial E}{\partial x_i} \approx \frac{E(h + \varepsilon\, h_i, f + \varepsilon\, f_i) - E(h,f)}{\varepsilon},$$

with (h_i, f_i) being just the basis vector contained in the i-th column of B.

3.2.5 Shape Matching

Given two shapes $p_0, p_1 \in \mathcal{C}_N$ and the methods from the previous sections, it is possible to calculate a geodesic between p_0 and p_1. If we want to use this to calculate a shape distance, it is required that the curves representing the shapes are parametrised so that the geodesic distance between p_0 and p_1 is minimal[3]. Since for computation time reasons we do not put optimisation for parametrisations into the Log map calculation [90] as mentioned in Section 3.2, we need to find suitable parametrisations by means of a matching procedure prior to calculating a geodesic.
In order to match two curves $\alpha_0(t), \alpha_1(t)$, one can use a dynamic programming strategy similar to the one used in [118] to find a homeomorphism $\tilde{\gamma}(t)$ that gives correspondences between the curves by associating $\alpha_0(t)$ with $\alpha_1(\tilde{\gamma}(t))$. The curves are represented by their respective pre-shapes $(\Phi_i, \Theta_i) \in \mathcal{C}$, $i \in \{0, 1\}$.
Before continuing with the energy to be minimised for matching, note that applying a re-parametrisation by a diffeomorphism $\gamma(t)$ to a curve $\alpha(t)$ gives for the velocity

$$\begin{aligned} \frac{d}{dt}\alpha(\gamma(t)) &= \dot{\gamma}(t)\,\dot{\alpha}(\gamma(t)) \\ &= \dot{\gamma}(t)\,e^{\Phi(\gamma(t))}\,e^{i\,\Theta(\gamma(t))} \\ &= e^{\Phi(\gamma(t))+\log(\dot{\gamma}(t))}\,e^{i\,\Theta(\gamma(t))}\,. \end{aligned} \tag{3.30}$$

Assuming two shapes (Φ_0, Θ_0) and (Φ_1, Θ_1), a suitable diffeomorphism $\gamma(t)$ is sought for by minimising the functional

$$E(\gamma) = \|(\Phi_0, \Theta_0) - (\Phi_1, \Theta_1) \circ \gamma(t)\|^2_{(\Phi_0,\Theta_0)}$$

$$= \Big\langle [(\Phi_0, \Theta_0) - (\Phi_1, \Theta_1) \circ \gamma(t)], [(\Phi_0, \Theta_0) - (\Phi_1, \Theta_1) \circ \gamma(t)] \Big\rangle_{(\Phi_0,\Theta_0)}$$

$$= \Big\langle \big(\Phi_0 - [\Phi_1(\gamma(t)) + \log(\dot{\gamma}(t))], \Theta_0(t) - \Theta_1(\gamma(t))\big), \\ \big(\Phi_0 - [\Phi_1(\gamma(t)) + \log(\dot{\gamma}(t))], \Theta_0(t) - \Theta_1(\gamma(t))\big) \Big\rangle_{(\Phi_0,\Theta_0)}$$

$$= a \int_I [\Phi_0(t) - (\Phi_1(\gamma(t)) + \log(\dot{\gamma}(t)))]^2 \, e^{\Phi_0(t)}\, dt + \\ b \int_I [\Theta_0(t) - \Theta_1(\gamma(t))]^2 \, e^{\Phi_0(t)}\, dt \tag{3.31}$$

[3]Recall that the elements of the shape space (as opposed to pre-shape space) should be invariant with respect to re-parametrisation, and that we use parametrised representations for the curves.

where the elastic metric from Equation (3.4) and the derivative from (3.30) are used.

In order to approximate a solution for (3.31), a dynamic programming scheme is used which will now be summarised. γ is approximated by a piecewise linear function $\tilde{\gamma}$ defined on an $N \times N$ grid, where N is the number of points sampled equidistantly on both curves using arc length. $\tilde{\gamma}$ starts on the grid at $(0, 0)$ and ends at $(N-1, N-1)$. In between, the slopes of the line segments that make up $\tilde{\gamma}$ are always strictly positive. $\tilde{\gamma}$ is only a homeomorphism (and not a diffeomorphism), since it is not differentiable at the finite number of points where one linear segment ends and the next begins. Theoretically, to obtain the optimum, one would have to compute the energy on segments from each grid point i, j to all other grid points “below” that point, i.e. for all $0 < k < i$, $0 < l < j$. To reduce the computational cost, the energy is computed only on a smaller, fixed neighbourhood $\mathcal{N}_{ij}$ and used to find a locally optimal line segment, as proposed by [118, 91], to reduce the complexity from $O(N^3)$ to $O(N^2)$. The neighbourhood that has been used in this work is illustrated along with the grid in Figure 3.3.

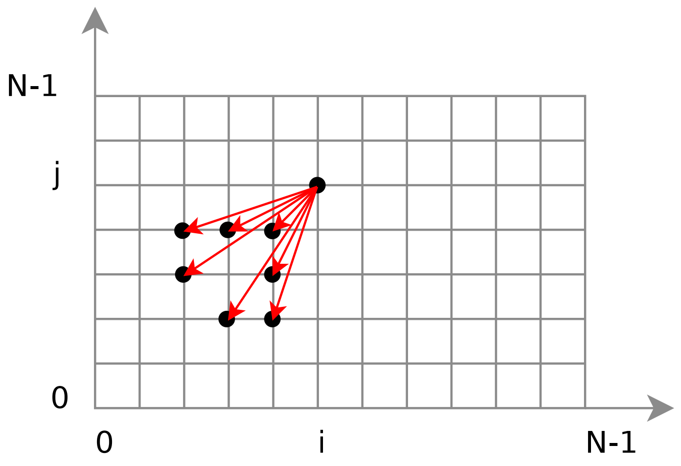


Figure 3.3: Illustration of a possible neighbourhood $\mathcal{N}_{ij}$ that can be used in the dynamic programming approach to matching. Points on (i, x) or (x, j) do not make sense, since they would imply that two points on the resulting curve coincide, which contradicts the requirement that the curve must be regular. Note that for closed curves, it makes sense to duplicate the start point and therefore add another column and row to the grid (not shown here).

Say that two grid points are labelled as (k, l) and (i, j), with $k, l, i, j \in \{0, \ldots, N-1\}$ and $k < i$, $l < j$. The energy (3.31) on a line segment joining

(k,l) and (i,j) is written as

$$E(k,l,i,j) = a\int_{I_{ki}} \left[\Phi_0(t) - (\Phi_1(\tilde\gamma_{klij}(t)) + \log(\dot{\tilde\gamma}_{klij}(t)))\right]^2 e^{\Phi_0(t)}\,dt + b\int_{I_{ki}} \left[\Theta_0(t) - \Theta_1(\tilde\gamma_{klij}(t))\right]^2 e^{\Phi_0(t)}\,dt\,. \quad (3.32)$$

I_{ki} is the sub-interval of I corresponding to the segment from k to i, and $\tilde\gamma_{klij}$ is the linear segment of $\tilde\gamma$ joining the grid points $(k,l),(i,j)$.
The minimum energy is then found by computing intermediate energies

$$H(0,0) = 0 \quad (3.33)$$
$$H(i,j) = E(\hat k,\hat l,i,j) + H(\hat k,\hat l) \quad (3.34)$$

with

$$(\hat k,\hat l) = \arg\min_{(k,l)\in\mathcal{N}_{ij}} \left(E(k,l,i,j) + H(k,l)\right).$$

$\mathcal{N}_{ij}$ is the aforementioned neighbourhood of grid points of point (i,j). $H(i,j)$ is computed from $(0,0)$ to $(N-1,N-1)$ and then the optimal path is found by back tracking.

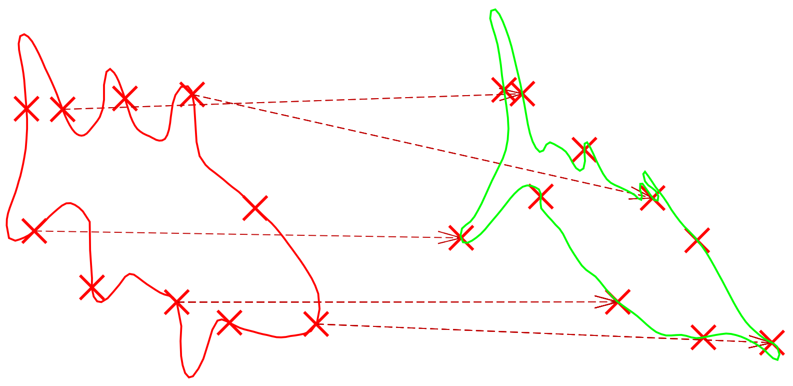


Figure 3.4: Example for curve matching. The indicated points indicate a few roughly corresponding points found by minimising Equation (3.31); only some correspondences are indicated by arrows in order not to obscure the image too much. Note that we have pre-determined a common start point by applying full Procrustes matching for all cyclic permutations. The matching homeomorphism and angle functions are depicted in Figure 3.5. These curves are taken from the Surrey fish data base from Mokhtarian et al. [94].

Notice that in order to find correspondences for closed curves, one would have to run the matching process for all possible start points on one curve. In

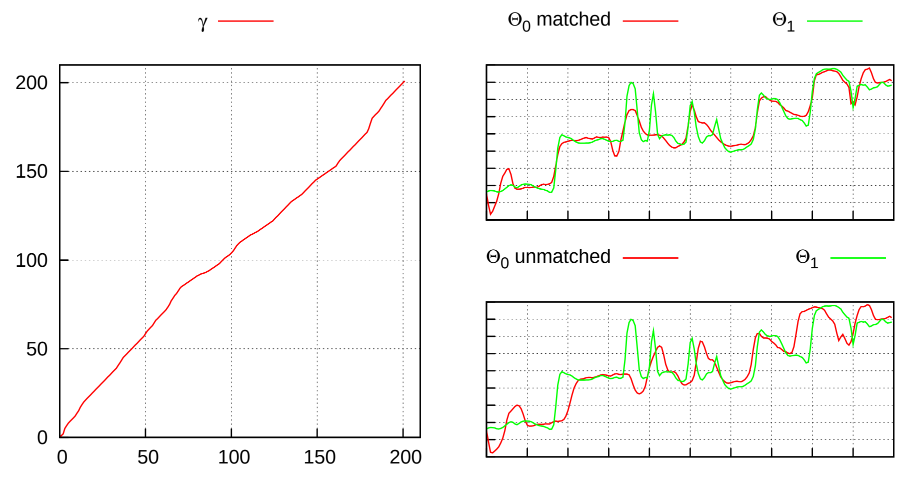


Figure 3.5: Homeomorphism $\tilde{\gamma}$ for the matching in Figure 3.4, and the angle functions Θ_0 and Θ_1 before and after matching. Notice how Θ_0 of the matched curve is changed to fit Θ_1. Here, $a/b = 1$.

the discrete case, this would mean that the complexity for matching increases from $O(n^2)$ for open curves to $O(n^3)$ for closed ones. One can alleviate the problem to $O(n^2 \log n)$ as described in [118], but the computation time is of course still increased.
Figures 3.4 and 3.5 depict an example for the result of this matching procedure similar to one of the examples in [91]. One can also resort to predetermining a start point by first applying full Procrustes matching for all cyclic permutations of one of the curves, instead of running the minimisation for all possible start points.

3.3 Alternative Shape Representation with a Single Function

Joshi et al. [60] propose an alternative representation of shape using only one function instead of a pair of functions. In contrast to the approach using function pairs from [91], the inner product defined on the tangent space at a point p is not depending on p, so computation is simplified.
While this model is not used in this thesis, it is still mentioned briefly to point out another direction of development.
Given a regular curve $\beta : [0, 2\pi] \mapsto \mathbb{R}^n$, the authors of [60] introduce the

shape function

$$q(s) = \frac{\dot{\beta}(s)}{\sqrt{\|\dot{\beta}(s)\|}} \,. \tag{3.35}$$

The used norm is the canonical Euclidean norm. Using this, $\|q(s)\|$ is the square root of the curve's speed, as can be seen by calculating

$$\begin{aligned} \|q(s)\| = \langle q(s), q(s)\rangle^{\frac{1}{2}} &= \left\langle \frac{\dot{\beta}(s)}{\sqrt{\|\dot{\beta}(s)\|}}, \frac{\dot{\beta}(s)}{\sqrt{\|\dot{\beta}(s)\|}} \right\rangle^{\frac{1}{2}} \\ &= \left(\frac{\langle \dot{\beta}(s), \dot{\beta}(s)\rangle}{\|\dot{\beta}(s)\|} \right)^{\frac{1}{2}} = \left(\frac{\|\dot{\beta}(s)\|^2}{\|\dot{\beta}(s)\|} \right)^{\frac{1}{2}} = \|\dot{\beta}(s)\|^{\frac{1}{2}} \,. \end{aligned}$$

Therefore, the curve $\beta(t)$ can be expressed by

$$\beta(t) = \int_0^t \|q(\tau)\| \, q(\tau) \, d\tau \,, \tag{3.36}$$

leaving out the constant accounting for translation only.
For open curves, it is noted in [60] that the space of elastic curves of length 1 using the above representation is the unit sphere:

$$\mathcal{B} \equiv \left\{ q \,:\, [0, 2\pi] \mapsto \mathbb{R}^n \;\middle|\; \int_0^{2\pi} \langle q(s), q(s)\rangle \, ds = 1 \right\} . \tag{3.37}$$

Using the geometry of the sphere, simple formulas are provided in [60] for calculating a geodesic given two points and the tangent vector between them, as well as parallel transport of tangent vectors along a geodesic.
For closed curves, the matter is once again more involved. Similar to [91], the inner product is defined in [60] as

$$\langle u, v\rangle := \int_0^{2\pi} \langle u(s), v(s)\rangle \, ds \,. \tag{3.38}$$

Note that by not distinguishing speed and angle functions, an adjustment of stretching and bending as before is no longer possible.
In the case of closed curves, similar to [91], a map $\mathcal{G} : \mathcal{Q} \mapsto \mathbb{R}^n$ is defined on the space of all elastic curves $\mathcal{Q} := \{q : [0, 2\pi] \mapsto \mathbb{R}^n\}$. $\mathcal{G}$ is, component-wise, defined as $\mathcal{G}_i = \int_0^{2\pi} q_i(s) \, \|q(s)\| \, ds$, $i = 1, \ldots, n$. This map is then used to implicitly define the space of all closed elastic curves as $\mathcal{A} := \mathcal{G}^{-1}(0)$. Together with $\mathcal{B}$, $\mathcal{A}$ is used to define the space of all elastic, closed curves of unit length, $\mathcal{C} := \mathcal{A} \cap \mathcal{B} \subset \mathcal{Q}$. The elements of $\mathcal{C}$ are therefore invariant to scale and translation, but not to rotation.
Instead of the method proposed before, a *path straightening method* [70] is suggested in [60] to compute a geodesic: An initial solution for the geodesic is

assumed and then refined in order to minimise an energy which measures the length of the considered curve on $\mathcal{C}$. This involves computation on the space of all curves on $\mathcal{C}$, for which numerical algorithms are given in [60]. They claim that the path straightening method is more stable than a shooting method, by keeping the start and end points fixed and optimising over all paths connecting the two points.
The lack of rotation invariance is tackled in [59], by introducing another minimisation into the calculation of geodesics, over rotations and re-parametrisations. This adds computational complexity, and [59] no longer explicitly claims to be more efficient than the previous model using function pairs. This was not verified in the scope of this thesis.

Chapter 4

Level Set Segmentation

4.1 Introduction

Topics discussed in this thesis revolve around *shape*. That is in this context the information contained in object *contours*, or *silhouettes*, which is invariant to a set of transformations.

A natural question is how these contours come to be in the first place, how to acquire them given that all we have are two dimensional grey value or colour images.
To this end, one can use a segmentation algorithm which segments an image into regions, and then use the boundaries of the regions. *Active contour* techniques have emerged in the past, starting with the *snakes* model from Kass et al. [63] in 1988. Active contour methods evolve one or more curves by minimising some energy functional, which usually consists of one energy that is *internal* to the curve, that means it depends only on the curve itself, and one that depends on image data and is called *external*. Other terms adding prior knowledge about the curve to be extracted have been proposed as well. While the original work of Kass used splines to describe and evolve the curve, later work [15] used the level set framework, which had been introduced originally for front propagation in computational physics [97].
When using parametric representations of curves, topological changes and reparameterisation usually have to be handled by more or less complicated and arbitrary extra algorithms. The level set framework, representing contours implicitly as level sets of a higher dimensional function, has the unarguable advantages that

1. changes in topology of an evolving contour are naturally handled, without any additions to the model.
2. it is parameter-free.
3. it allows for simple ways to approximate geometric properties of the embedded curve.

However, the flexibility does not come for free. There are numerical issues which must be taken into account and the computational work load is generally larger than for parametric models. A good speed up can be achieved by using multi resolution techniques for implementing level sets, and also by using so called narrow band methods, which restrict calculations to a narrow band around the level set of interest. Parallel computation can also speed up implementations.

In this chapter, several aspects of image segmentation and curve evolution using the level set framework are discussed. The level set method for active contours has attracted the interest of many researchers in recent years, so that many facets have been examined and countless additions to the basic method have been published. Some of these facets are described in this chapter, with the aim to compile the information needed to actually use the method in an implementation and to understand the implementation used in this thesis, in which many of the described techniques are present.

It is, in the scope of this thesis, not possible to mention and describe each and every aspect of the method, which is still a field of active research, nor the large variety of applications and modifications it has undergone and is still undergoing.

4.2 Level Sets

Let $\Omega \subset \mathbb{R}^2$, representing the image domain, and let $\Phi : \Omega \mapsto \mathbb{R}$ be a smooth function so that the *interface* C of dimension 1 is the zero level set of Φ,

$$C = \{x \,|\, \Phi(x) = 0\}\,. \tag{4.1}$$

The interface has here *codimension* 1 since it is one dimensional and embedded in a two dimensional function. We do in this thesis not look into interfaces of higher codimension.
In image segmentation tasks, the iso contours of this function Φ are used to represent curves. So Φ acts as an *embedding function* for the actual curve, and the curve is *implicitly* given by Φ. One of the main advantages of using level sets is their capability to deal naturally with changing topology without the need for complicated manoeuvres to split or merge curves [97, 32].
Evolution of the embedding function over time can be used to simulate a changing interface due to some external force or velocity field, and due to internal forces. While the external force drives the contour towards possible boundaries of objects using image or other data, the purpose of the internal force is usually to impose certain properties onto the curve, such as smoothness.

4.3 Tools and Notation

The descriptions and examples in this chapter are mostly for $\mathbb{R}^2$, but can generally be used for $\mathbb{R}^n$. $\Omega \subset \mathbb{R}^n$ denotes the image domain. $\Phi : \Omega \mapsto \mathbb{R}$ is the embedding function. We call

$$\Omega^- = \{x \,|\, \Phi(x) < 0\}$$

the *inside* of the area that is bounded by the contour C. Analogously,

$$\Omega^+ = \{x \,|\, \Phi(x) > 0\}$$

is called the *outside*. The contour itself, the zero level set of the embedding function Φ,

$$C = \{x \,|\, \Phi(x) = 0\}$$

is also known as the *interface* between Ω^- and Ω^+.

The normal N of a level set can be expressed as

$$N = \frac{\nabla\Phi}{|\nabla\Phi|}\,.$$

Furthermore, when expressing a functional in terms of an embedding function, the Heaviside function

$$H(\Phi) = \begin{cases} 0 & \Phi \leq 0 \\ 1 & \Phi > 0 \end{cases}$$

comes in handy to distinguish between two different phases. A volume integral over two different phases divided by the zero level set which one may call *inside* and *outside* can then be expressed as

$$\int_\Omega f(x)\,(1 - H(\Phi(x)))\,dx$$

and

$$\int_\Omega f(x)\,H(\Phi(x))\,dx$$

respectively.

For numerical implementations, a regularised version H_ε of the Heaviside-function is used, with some small parameter ε. Chan [17] delivers an argument on what happens if one leaves out the approximation to $H(\cdot)$ in the two-phase, piecewise constant segmentation model from [18] and what the consequences are. This can be found in Appendix B.
Possible regularised functions are

$$H_{\varepsilon,1}(\Phi) = \begin{cases} 0 & \Phi < -\varepsilon \\ \frac{1}{2} + \frac{\Phi}{2\varepsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\Phi}{\varepsilon}\right) & -\varepsilon \leq \Phi \leq \varepsilon \\ 1 & \varepsilon < \Phi \end{cases} \tag{4.2}$$

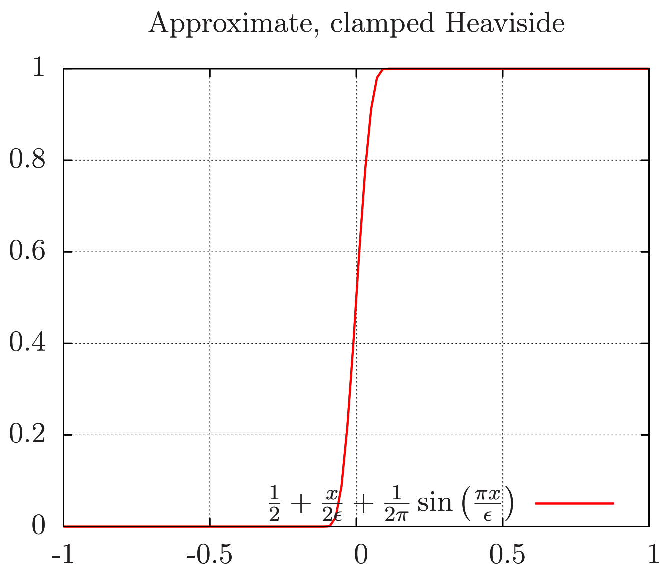
$$\delta_{\varepsilon,1}(\Phi) = H'_{\varepsilon,1}(\Phi) = \begin{cases} 0 & \Phi < -\varepsilon \\ \frac{1}{2\varepsilon} + \frac{1}{2\varepsilon}\cdot\cos\left(\frac{\pi\Phi}{\varepsilon}\right) & -\varepsilon \leq \Phi \leq \varepsilon \\ 0 & \varepsilon < \Phi \end{cases}, \tag{4.3}$$

or a non-clamped version

$$H_{\varepsilon,2}(\Phi) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{\Phi}{\varepsilon}\right)\right) \tag{4.4}$$

$$\delta_{\varepsilon,2}(\Phi) = \frac{1}{\pi\left(\frac{\Phi^2}{\varepsilon^2}+1\right)\varepsilon}, \tag{4.5}$$

see for example [130, 18]. Both regularised versions of $H(\Phi)$ are illustrated in Figure 4.1.



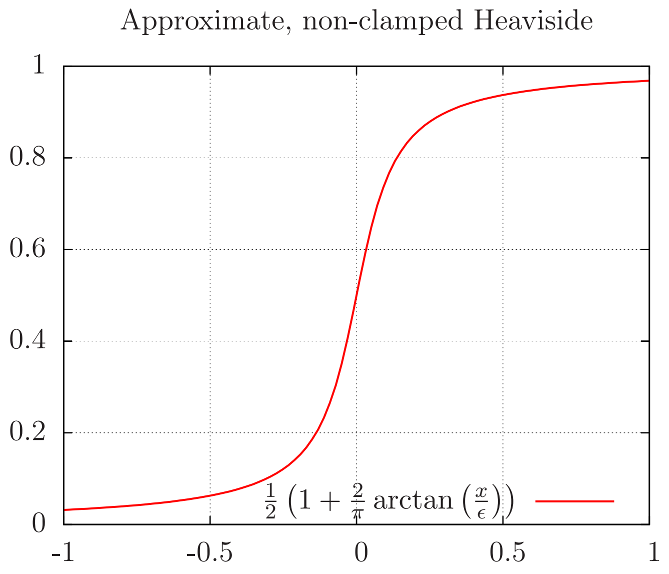


Figure 4.1: Upper: clamped, regularised Heaviside function. Lower: non-clamped, regularised Heaviside function. Here, we chose $\epsilon = 0.1$.

4.4 Signed Distance Functions

There are infinitely many implicit functions Φ to embed a zero level set. Theoretically, any embedding function would do. Minding that one needs to calculate numerically on Φ, specifically to approximate derivatives, Φ should certainly be smooth and should not exhibit extremely large or extremely small slope. So called signed distance functions are good candidates and are being used in the level set segmentation method; some desirable properties of signed distance functions will be mentioned a little further down.

Definition 4.4.1 (Distance function) *A function*

$$d : \Omega \mapsto \mathbb{R}^+ ,$$

$$d(x) = \min_{x_I \in C} (|x - x_I|) \tag{4.6}$$

with

$$|\nabla d| = 1 , \tag{4.7}$$

that maps to each x the distance to the closest point $\hat{x}_I$ on the boundary C is called a distance function.

Definition 4.4.2 (Signed distance function) *A function*

$$\Phi : \Omega \mapsto \mathbb{R} ,$$

$$|\Phi(x)| = d(x) \quad \forall x \in \Omega \tag{4.8}$$

$$\Phi(x) = \begin{cases} 0 & \forall x \in C \\ -d(x) & \forall x \in \Omega^- \\ d(x) & \forall x \in \Omega^+ \end{cases} \tag{4.9}$$

is called a signed distance function.

The choice of sign for the inside and outside regions of Ω is arbitrary, but in this text the set $\{x : \Phi(x) < 0\}$ is called the *inside* of a contour C.
Notice that the property $|\nabla\Phi| = 1$ is not true for points that are equidistant from two points on the zero level set C: this leads to a “kink” in Φ. Theoretically, this is a problem since at these points, the derivative of Φ is not defined; Φ is not smooth. If one is calculating with Φ defined only on a finite grid, this does not pose a real problem *in practice*: the derivative is indeed not 1 around these points, but it is smeared by the discretisation [120]. Therefore, one should keep in mind that the derivative near the kinks will be between -1 and 1, but the numerical results will be well behaved.
An illustration of a signed distance function embedding the contour of an elephant is shown in Figure 4.2.

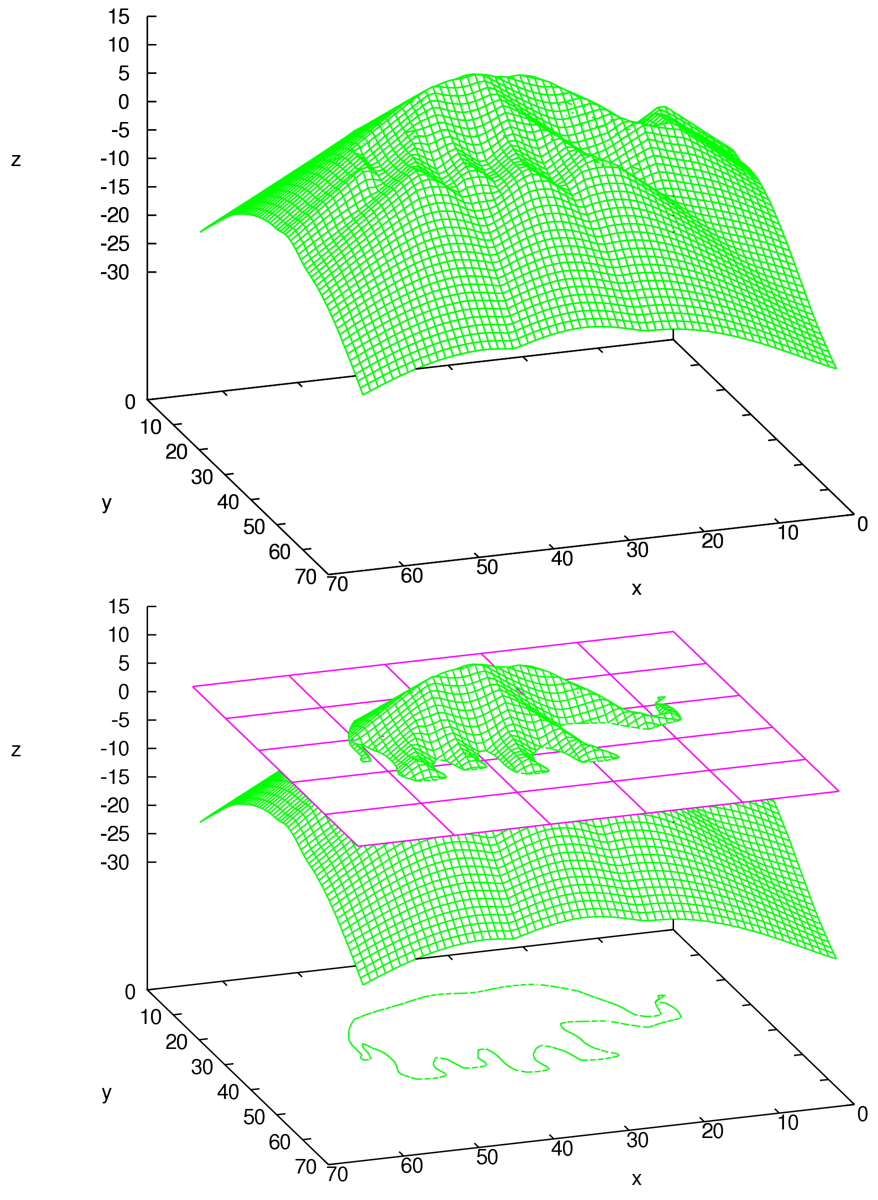


Figure 4.2: A signed distance function embedding the contour of an elephant at its zero level. The upper image shows only the embedding function. The lower image shows the same embedding function together with a plane drawn at the zero level $z = 0$ and the isocontour at $z = 0$ drawn at the bottom.

The use of signed distance functions as embedding functions also results in a few simplifications, which can be desirable for the reduced amount of computation needed. Specifically,

$$\begin{aligned} N &= \frac{\nabla\Phi}{|\nabla\Phi|} = \nabla\Phi \\ \kappa &= \nabla \cdot N \\ &= \nabla \cdot (\nabla\Phi) \\ &= \Delta\Phi \\ &= \Phi_{xx} + \Phi_{yy} + \Phi_{zz} \end{aligned} \tag{4.10}$$

where κ is the *curvature* of the interface or equivalently the *divergence* of the normal field N.

Summarising, signed distance functions are good candidates for embedding functions, since they exhibit the desirable properties that they are smooth except for certain points where their numerical implementations are expected to fail in a predictable, "graceful" manner, and that they can lead to a reduction of computational costs, depending on the algorithms which are applied.

4.5 Moving Interfaces

4.5.1 Explicit Formulation

We are interested in simulating the motion of a contour C. Let $V(x)$ be a vector field describing an external velocity. We can write the motion of the interface C under the influence of that velocity field explicitly as

$$\frac{dx}{dt} = V(x) \quad \forall x \in C\,. \tag{4.11}$$

This brings problems with distortions in the discretised case and with topological changes of the interface (like splitting and merging) [120]. The distortion problems stem from the fact that we calculate only on a finite set of points representing C and that these points can move very far apart during the evolution process, making the representation of C very inaccurate, or move very close together, which would introduce numerical problems calculating with point distances. Topological changes, splitting and merging of interfaces, is not handled automatically. As a result of these considerations, one will have to introduce specialised methods to deal with these situations, as has been done for example in [76, 129, 87, 32].
However, one should not forget that the explicit method also has two practically appealing advantages, namely that it is usually very fast, and that it is memory efficient.

4.5.2 Implicit Formulation

Consider the evolution of $\Phi(x)$ over time t through simple convection [120] under a vector field V, governed by the partial differential equation

$$\Phi_t + V \cdot \nabla\Phi = 0 \,. \tag{4.12}$$

Since $\nabla\Phi$ is parallel to the normal field of the level sets of Φ, and thereby normal to the embedded interface C, only the component V_n of V that is normal to C has an actual impact on the rate of change Φ_t.
Using the normal vector N, we obtain

$$V_n \cdot N \cdot \nabla\Phi = V_n \cdot \frac{\nabla\Phi}{|\nabla\Phi|} \cdot \nabla\Phi \,,$$

and (4.12) becomes

$$\Phi_t + V_n \cdot |\nabla\Phi| = 0 \,, \tag{4.13}$$

describing motion in normal direction. In this form, it is also referred to as *level set equation*.

4.6 Level Sets in Image Segmentation

Level set methods can be used for image segmentation using *active contours*, contours that evolve from an initial configuration following an energy minimisation. In the original work by Kass et al. [63], active contours were implemented using parametric spline functions. The advantage of using parametric functions to describe a contour is clearly their speed. The major drawback, however, is the lack of the ability to automatically cope with changes in curve topology, as has been explained previously.
This drawback is overcome when using an implicit function that embeds the contour as its zero level set [97] (or any other fixed level set). Since the evolution is carried out on the embedding function, topology changes can occur naturally and no additional heuristics need to be used.

4.7 Active Contours

In [63], Kass, Witkin, and Terzopoulos introduced a *parametric active contour model* which they christened *snakes*. A curve $s(t)$ defined by a spline is pushed towards sought image features by optimising an energy functional $E(s(t))$, defined on the snake, to a local minimum — this is why they call the model *active*.
The energy functional originally proposed by [63] is quite general and consists of a regularising term E_{int}, an image term E_{im} to drive the snake towards

desired image features, and some external energy E_{ext} which can represent user control or some other constraints:

$$E(s(t)) = \int_0^1 E_{\text{int}}(s(t)) + E_{\text{im}}(s(t)) + E_{\text{ext}}(s(t))\, dt \tag{4.14}$$

Here it is assumed that the snake $s(t)$ is parametrised by arc length in the interval $[0, 1]$.
The internal energy in the original work involves the first and second spatial derivatives of the snake, and therefore has a smoothing effect on the snake when the energy is minimised. As image terms E_{im}, [63] proposes the image intensity itself, pulling the snake towards a specific grey value, or an edge term depending on the image gradient, which pulls the snake towards intensity edges.

4.8 Geodesic Active Contours

In [14, 15], Caselles et al. derive a model of curve evolution that is related to the mean curvature flow[1] in a Riemannian space of curves $s(t)$ with a metric induced by the image I. They introduce an edge detector function $g : [0, \infty) \mapsto \mathbb{R}_+$ that must be strictly decreasing, and show that if one chooses

$$E(s(t)) = \alpha \int_0^1 |s'(t)|^2\, dt + \lambda \int_0^1 g(|\nabla I(s(t))|)^2\, dt$$

for the active contour model (4.14), its minimisation is equivalent to minimising

$$\int_0^1 g(|\nabla I(s(t))|)\, |s'(t)|\, dt\,, \tag{4.15}$$

which can be seen as the length of the curve $s(t)$ under the metric

$$\gamma_{ij} = g(|\nabla I(s(t))|)^2\, \delta_{ij}\,, \quad i, j \in \{1, 2\}\,.$$

Minimising (4.15) means to follow a curve shortening flow according to the metric γ_{ij}.
In [15], the authors provide an implicit formulation using the level set method to solve for a minimiser of (4.15).
This *geodesic active contour* model is not, like the original snake model from Kass et al., dependent on the parametrisation of the contour.

Note that with the geodesic active contour model, curves are always attracted to high absolute image gradients, which gives rise to the following problems: If the image is blurry, gradients may be quite small, while in areas with a large amount of noise, gradients will be high where there are no actual

[1]The mean curvature flow minimises the area of a surface patch (or in this case the length of a curve) [73].

object boundaries. Objects with fuzzy boundaries are hard to segment with this approach, since the boundaries would in such cases not lead to high image gradients; think for example of astronomical images of nebulae.

4.9 Region Based Active Contours

Instead of letting a curve evolve towards intensity edges, another strategy is to search for a curve that divides an image into homogeneous regions; this was formulated by Mumford and Shah [95]. An approximation using regions of constant grey value was formulated by Chan and Vese [131]. The advantage over edge based models is that objects with more diffuse boundaries can be better segmented from the background, if the image conforms to the model assumptions. Also, depending on certain user parameters, region based methods can be quite robust against noise.

4.9.1 The Mumford-Shah Functional

Mumford and Shah [95, 46] formulated the image segmentation problem as the minimisation of the energy functional (4.16). Let $u_0 : \Omega \mapsto \mathbb{R}$ be the original image on a domain Ω, and $C \subset \Omega$ a set of discontinuities (the region boundaries) dividing the image domain into a number of open, disjoint subsets with $\Omega_1 \cup \cdots \cup \Omega_n \cup C = \Omega$, then to segment u_0 into smooth regions, solve

$$\min_u F(u, C) = \int\limits_{\Omega \setminus C} (u - u_0)^2 \, dx + \alpha \cdot \int\limits_{\Omega \setminus C} |\nabla u|^2 \, dx + \beta \int\limits_C d\sigma \qquad (4.16)$$

for a piecewise smooth $u(x)$ with discontinuities along the boundaries C. The first term is a fidelity term that asserts that the minimiser u should not be too different from u_0, the second term penalises variations on the sought regions, and the third term penalises the length of the region boundaries. Mumford and Shah conjectured that there is a minimiser of (4.16) for which C is a union of differentiable curves [95, 46].

4.9.2 The Chan and Vese Model

Chan and Vese [131] have introduced a region based segmentation method using the level set framework, that is based on the idea of the Mumford-Shah functional. Assuming the piecewise constant case instead of a piecewise smooth approximation, and that exactly two regions are to be found, the energy to be minimised is

$$\begin{aligned} F(c_1, c_2, C) &= \mu \cdot \text{Length}(C) + \nu \cdot \text{Area}(\text{inside}(C)) \\ &\quad + \lambda_1 \cdot \int_{\text{outside}(C)} |u_0(x,y) - c_1|^2 \, dx \, dy \\ &\quad + \lambda_2 \cdot \int_{\text{inside}(C)} |u_0(x,y) - c_2|^2 \, dx \, dy \end{aligned} \tag{4.17}$$

with $\mu, \nu, \lambda_1, \lambda_2$ fixed parameters. μ weights the penalisation of the contour's length. The area term has the effect of a so called balloon term by "blowing up" (or shrinking) the contour C. The parameter ν is often set to zero, so that no area term is used. λ_1, λ_2 weight the influence of the image data and most often are fixed to 1.
The original input image is denoted with u_0. The values c_1, c_2 minimising (4.17) for fixed C are the average grey values outside and inside the curve C, respectively.

To write this with C expressed in terms of an embedding function Φ, one uses the Heaviside function $H(\Phi)$ to express the inside (or outside) A of a curve C as

$$A = \{x \,:\, H(\Phi(x)) < 0\}$$

while the curve itself can be represented as

$$C = \{x \,:\, \Phi(x) = 0\} \,.$$

In practice, an approximation H_ε of $H(\cdot)$ is used, such as (4.2). Then,

$$\begin{aligned} \text{Length}(C) &\approx \int_\Omega |\nabla H_\varepsilon(\Phi(x,y))| \, dx \, dy \\ &= \int_\Omega \delta_\varepsilon(\Phi(x,y)) |\nabla \Phi(x,y)| \, dx \, dy \end{aligned}$$

and

$$\text{Area}(\Phi \geq 0) \approx \int_\Omega H_\varepsilon(\Phi(x,y)) \, dx \, dy$$

with $\Phi \geq 0$ inside the contour C defined by the zero level set.
The energy is then written as

$$\begin{aligned} F(c_1, c_2, \Phi) =& \mu \cdot \int_\Omega \delta_\varepsilon(\Phi(x,y)) |\nabla \Phi(x,y)| \, dx \, dy \\ &+ \nu \cdot \int_\Omega H_\varepsilon(\Phi(x,y)) \, dx \, dy \\ &+ \lambda_1 \cdot \int_\Omega |u_0(x,y) - c_1|^2 \cdot H_\varepsilon(\Phi(x,y)) \, dx \, dy \\ &+ \lambda_2 \cdot \int_\Omega |u_0(x,y) - c_2|^2 \cdot (1 - H_\varepsilon(\Phi(x,y))) \, dx \, dy \,. \end{aligned} \tag{4.18}$$

In order to do gradient descent to minimise the above functional, one calculates the Euler-Lagrange equation and introduces an artificial time, which leads to the gradient flow

$$\begin{aligned}\frac{\partial \Phi}{\partial t} = \delta_\varepsilon(\Phi)\Big[&\mu \cdot \operatorname{div}\left(\frac{\nabla \Phi}{|\nabla \Phi|}\right) \\ &- \nu \\ &- \lambda_1 (u_0 - c_1)^2 \\ &+ \lambda_2 (u_0 - c_2)^2\Big] = 0 \qquad (4.19)\end{aligned}$$

with a given initial Φ_0 at time $t = 0$. When using this for gradient descent, c_1 and c_2 are taken to be fixed for one time step, and then are recalculated taking Φ fixed, minimising (4.18) for Φ and c_1, c_2 in turn.
For H_ε, the authors in [131] use (4.4):

$$H_\epsilon(x) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{x}{\epsilon}\right)\right)$$

for some $\epsilon > 0$. This function is greater than zero everywhere, so it is possible for zero level curves to "pop out of nowhere", so that even inner contours can occur during evolution. Interestingly, Chan comes back to this in [17] where it is shown that when using H_ε as above, the flow (4.19) is related to a very similar algorithm described in [17] which, for constant c_1, c_2, finds global minimisers; see also Appendix B. For an alternative approximation of the Heaviside function with limited support, see Section 4.3.

The influence of the length and area terms in the above equations is illustrated in Figures 4.3, 4.4, 4.5, with a given initial embedding function and all other terms set to zero. The shortening effect of mean curvature flow can be seen in Figure 4.3. In comparison, the impact of another useful regularising term from Delingette [32], called curvature diffusion regularisation, can be seen in Figure 4.11. It will be described in Section 4.9.6.

An example of an image segmented with the level set method using Chan and Vese's functional [131] is shown in Figure 4.6. Noise has been added to the image to show the influence of the regularising terms — Figure 4.7 shows the evolution of the zero level set for the same image without using any regularising terms, leaving only the data terms. While the segmentation using smoothness terms in Figure 4.6 is quite close to the truth, the unregularised version in Figure 4.7 of course results in a segmentation which includes the noise.

The region-based segmentation method can easily be augmented to vector valued input data. In [130], a method is proposed which boils down to using an average over all colour channels (or other modalities) of an input image,

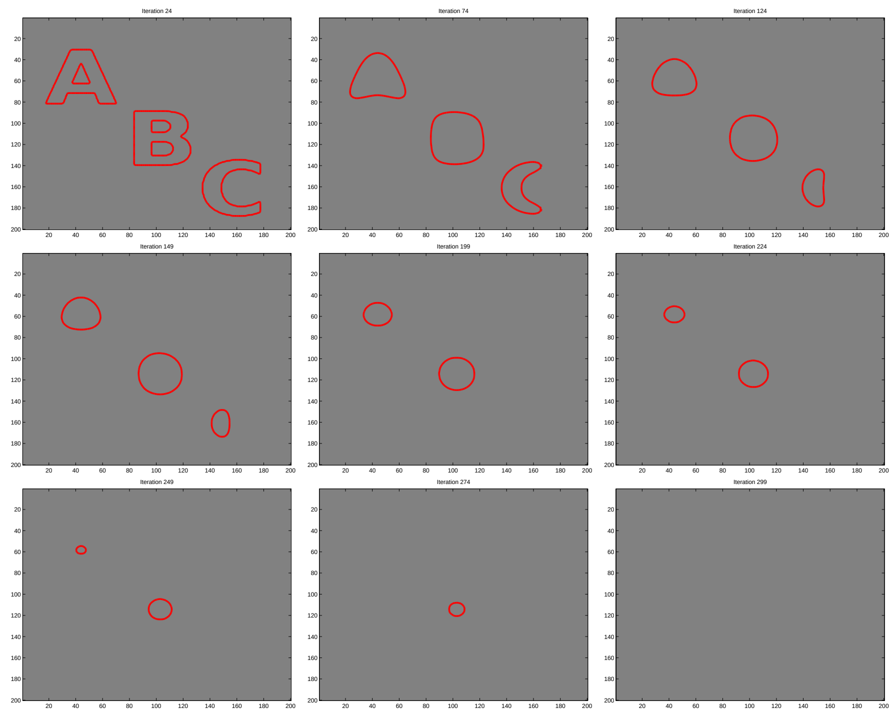


Figure 4.3: Influence of the length term when minimising functional (4.18) with $\mu > 0, \nu = \lambda_{\{1,2\}} = 0$. The process is initialised with the curves shown in the upper left image. Following the mean curvature flow, the curves get smoothed and shortened until they finally vanish.

so that Equation (4.18) becomes

$$\begin{aligned} F(c_1, c_2, \Phi) = \mu \cdot & \int_\Omega \delta(\Phi(x,y))|\nabla\Phi(x,y)|\,dx\,dy \\ & + \int_\Omega \frac{1}{N}\sum_{i=1}^{N} \lambda_{1,i} \cdot |u_{0,i}(x,y) - c_{1,i}|^2 \cdot H(\Phi(x,y))\,dx\,dy \\ & + \int_\Omega \frac{1}{N}\sum_{i=1}^{N} \lambda_{2,i} \cdot |u_{0,i}(x,y) - c_{2,i}|^2 \cdot (1 - H(\Phi(x,y)))\,dx\,dy\,. \end{aligned} \tag{4.20}$$

Here, $c_{1,i}, c_{2,i}$ are the average values of each colour channel outside and inside the zero level contour, respectively. Note $c_{\{1,2\}}$ are now N-vectors. $u_{0,i}$ denotes the i^{th} colour channel of the original image, and $\lambda_{\{1,2\},i}$ can be used to weight each channel independently. Figure 4.8 depicts a simple RGB image segmented with different values for the vector λ to select each colour channel, and for all of them together.

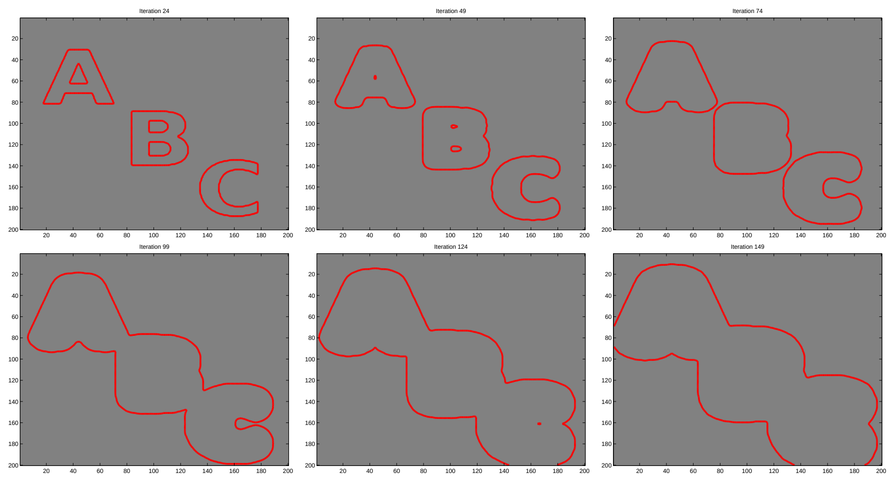


Figure 4.4: Influence of the area term when minimising functional (4.18): $\mu = 0, \nu > 0, \lambda_{\{1,2\}} = 0$. The curve gets “blown”, hence the area term is also called balloon term. The initial contour is shown in the upper left image. If the process was continued, the contours would eventually pass the image borders and vanish.

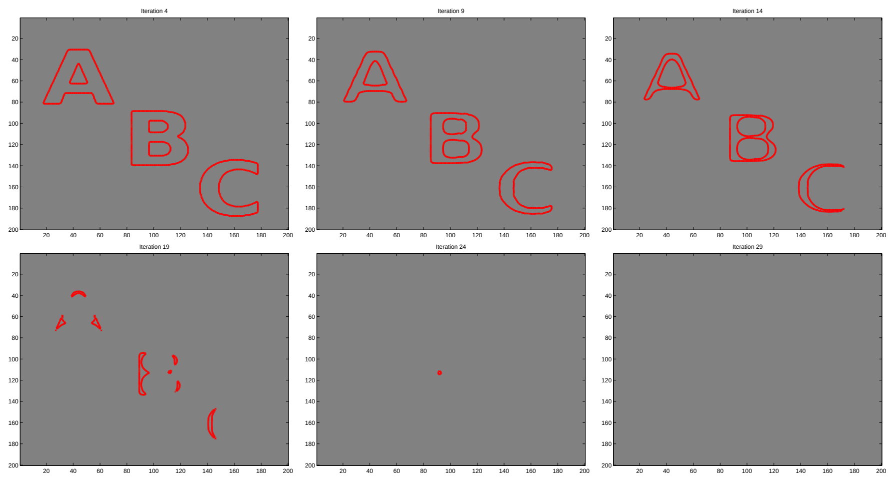


Figure 4.5: Influence of the area term when minimising Equation (4.18), here with a negative weight: $\mu = 0, \nu < 0, \lambda_{\{1,2\}} = 0$. The curve shrinks, without smoothing effect as opposed to the length term.

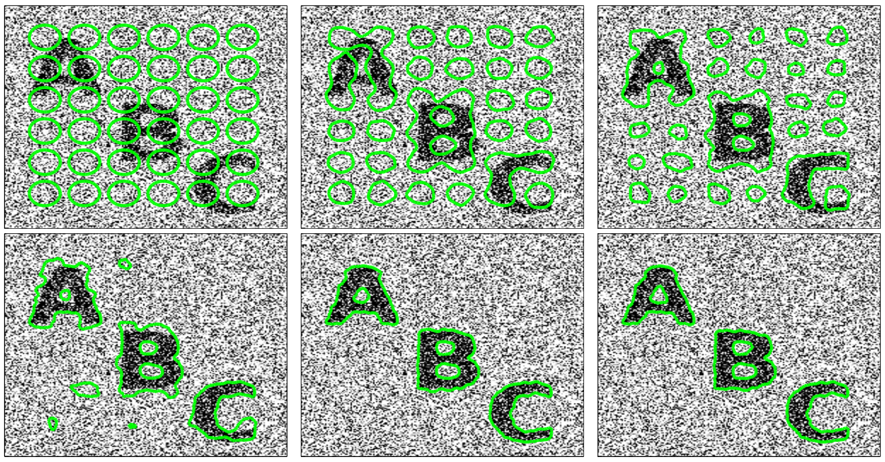


Figure 4.6: Binary segmentation using the level set model after Chan and Vese from Equation (4.18). Here, $\mu = 0.1, \lambda_{\{1,2\}} = 1.0$, and in addition the regulariser from Delingette, Equation (4.34) was used with a weight of 0.15.

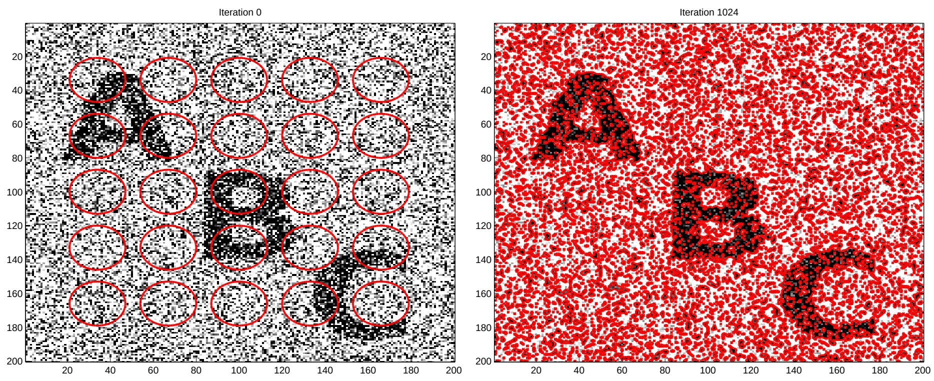


Figure 4.7: Binary segmentation using the level set model after Chan and Vese from Equation (4.18). To show the impact of regularisers, no regularising term was used in this example, only the data terms.

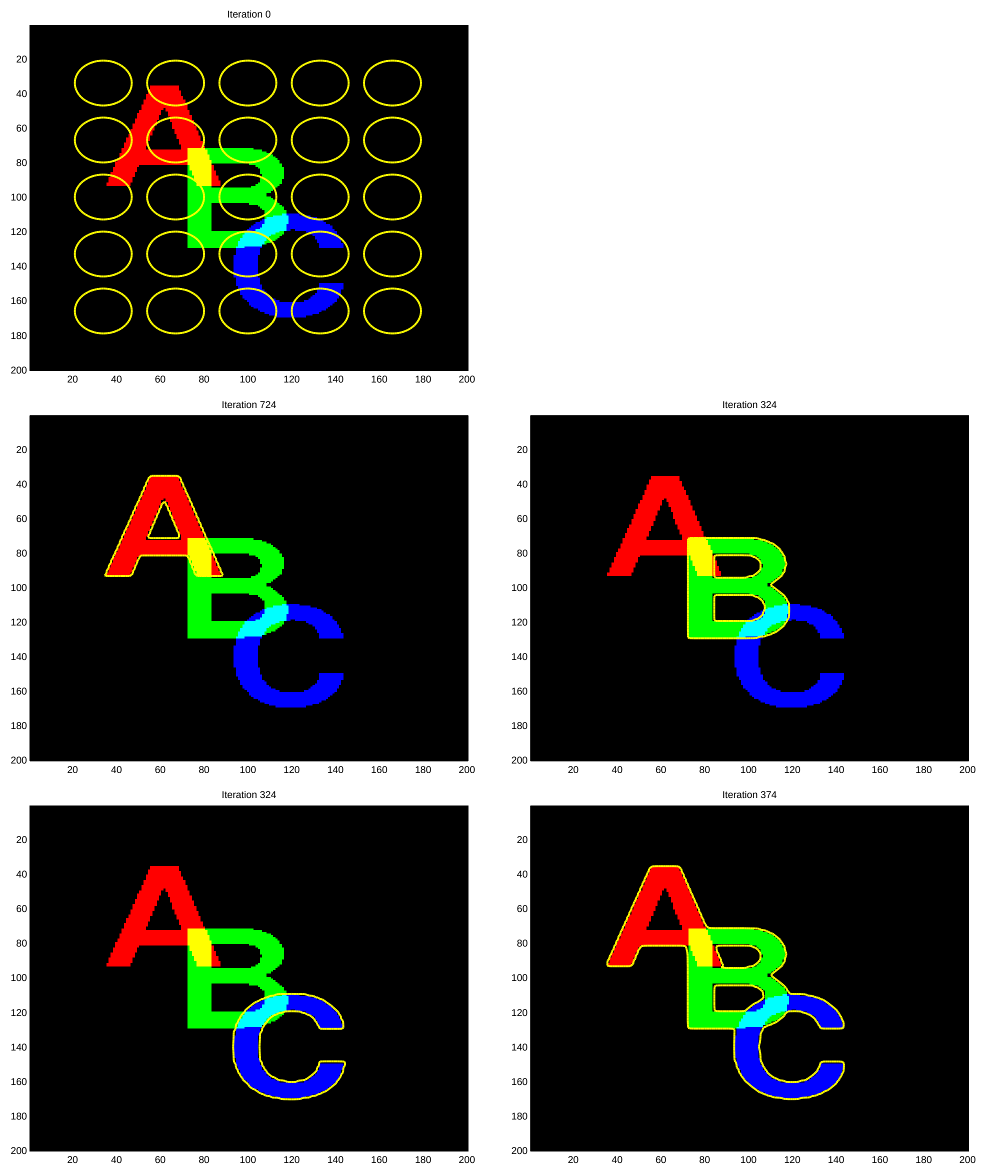


Figure 4.8: Example of segmentation using the vector valued Chan and Vese model from Equation (4.20). Top left: Initialisation. Middle: $\lambda_{\{1,2\}} = (1,0,0)^\top$ and $\lambda_{\{1,2\}} = (0,1,0)^\top$. Bottom: $\lambda_{\{1,2\}} = (0,0,1)^\top$ and $\lambda_{\{1,2\}} = (1,1,1)^\top$.

So far, the Chan and Vese model allows to segment data into two phases; for completeness, a multi-phase extension will be mentioned here in the following. In [83], the same authors introduce a modification that allows for 2^m segments using m embedding functions. This method guarantees that every pixel is element of exactly one segment of the image. In order to label 2^m different segments, m embedding functions $\Phi = (\Phi_1, \dots, \Phi_m)$ and m Heaviside functions $H(\Phi) = (H(\Phi_1), \dots, H(\Phi_m))$ are used. $H(\Phi)$ is a binary vector that is used for labelling segments, which leads to the number of 2^m distinguishable segments. Going back to the scalar version of the original Chan and Vese model, Equation (4.17), the problem is restated using a vector of average intensities $c = (c_1, \dots, c_{2^m})$ and characteristic functions

$$\chi_i,\ i = 1, \dots, 2^m$$

$$\chi_i(x) = \begin{cases} 1 & \text{if } x \text{ inside segment } i \\ 0 & \text{otherwise} \end{cases}$$

to

$$F(c, \Phi) = \sum_{i=1}^{2^m} \int_\Omega (u_0 - c_i)^2 \chi_i \, ds + \mu \sum_{i=1}^{2^m} \int_\Omega |\nabla \chi_i| \, ds \,. \tag{4.21}$$

The second, length term in (4.21) is approximated with the simpler

$$\mu \sum_{i=1}^{m} \int_\Omega |\nabla H(\Phi_i)| \, ds \,. \tag{4.22}$$

Notice that (4.22) is *not* the same as in the original equation, where the characteristic functions are combinations of the Heaviside functions of the embedding functions Φ. In (4.22), each Heaviside function is accounted for individually, which leads to boundaries being counted multiple times. The authors of [83] mention that their results were nevertheless still good on account of the dominance of the image term.

4.9.3 Statistics Based Data Terms and Texture Segmentation

The original, purely colour based data terms from Chan and Vese [131, 130] work well in settings with images that are close enough to being piecewise constant, and can also cope with noise in simple images. There is nothing that keeps one from using other features than colour or intensity in the models introduced so far, so it seems only logical to apply other features wherever appropriate, and so this has been done extensively in the past. For natural images and with the aim to enable segmentation with respect to texture as opposed to colour, other data terms have been shown to work quite well in many cases. Sandberg et al. [113] propose to use the multichannel Chan and Vese model for texture segmentation by using the output

of a set of filters as input image.
Rousson and Deriche [109] generalise the vector valued region based method [130] by describing the regions using probability distributions of grey or colour values. They rewrite the energy functional to minimise for an image $g(x)$ as

$$F = \sum_{i=1}^{N} \int_{\Omega_i} -\log p_i(g(x))\, dx + \text{Length}(\partial\Omega) \tag{4.23}$$

with probability densities $p_i(g)$ for each region Ω_i. Using Gaussians, they put

$$p_i(g) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(g-\mu_i)^\top \Sigma_i^{-1}(g-\mu_i)} \tag{4.24}$$

with means μ_i and covariances Σ_i. Noting that means and covariances can be calculated with

$$\mu_i = \frac{\int_{\Omega_i} g(x)\, dx}{\int_{\Omega_i} 1\, dx} \tag{4.25}$$

$$\Sigma_i = \frac{\int_{\Omega_i} (\mu_i - g(x))\,(\mu_i - g(x))^\top}{\int_{\Omega_i} 1\, dx} \tag{4.26}$$

and simplifying the log density functions a little yielding

$$e_i(x) := \log|\Sigma_i| + (g(x) - \mu_i)^\top \Sigma_i^{-1}(g(x) - \mu_i)\,, \tag{4.27}$$

they then end up with an update equation for the level set evolution of

$$\frac{\partial\Phi}{\partial t} = \delta_\varepsilon(\Phi)\left(\mu\,\text{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right) + e_2 - e_1\right)\,. \tag{4.28}$$

These equations can then be used for gradient descent to solve for Φ, updating μ_i, Σ_i after each time step similar to Chan and Vese's region based algorithm [130] described previously in this chapter.
A good review of the statistical point of view of region based level set segmentation can be found in Cremers et al. [27].

In [9], the authors propose to use not only colour information, but also structural information and motion information to drive the evolution process. In addition, probability distributions over the feature image are used instead of using only a mean. In this section, only the colour and structural components will be mentioned, for the motion component refer to [9]. The general idea is to do the energy minimisation not on the original input image, but on a field of feature vectors computed from the image, using the vector valued approach in Equation (4.20). These feature vectors can, of course, also contain the colour channels of the original image.

Given a feature image u with N channels, the authors of [9] propose to minimise for the data term

$$E(\Omega_i, p_{ij}) = -\sum_{j=1}^{N} \left(\int_{\Omega_1} \log p_{1j}(u_j(x))\, dx + \int_{\Omega_2} \log p_{2j}(u_j(x))\, dx \right), \quad (4.29)$$

with Ω_1, Ω_2 denoting the inner and outer image regions, respectively, and p_{ij} denoting probability density functions for all channels j and regions i.
Written in terms of an embedding function Φ, this reads

$$E(\Phi, p_{ij}) = -\sum_{i=1}^{2}\sum_{j=1}^{N} \left(\int_{\Omega} \log p_{ij}(u_j)\chi_i(\Phi)\, dx \right), \quad (4.30)$$

where $\chi_1(x) = H(x), \chi_2(x) = 1 - H(x)$. The update term for the gradient descent is then

$$\frac{\partial \Phi}{\partial t} = \sum_{j=1}^{N} \left(\log \frac{p_{1j}(u_j)}{p_{2j}(u_j)} H'(\Phi) \right) \quad (4.31)$$

where $H'(\cdot)$ denotes the spatial derivative of $H(\cdot)$ (see Appendix A.3 for the derivation of (4.31)).

This can readily be used to replace the data terms in Equation (4.19), resulting in

$$\frac{\partial \Phi}{\partial t} = \delta_\varepsilon(\Phi) \left[\mu \cdot \operatorname{div}\left(\frac{\nabla \Phi}{|\nabla \Phi|} \right) + \sum_{j=1}^{N} \left(\log \frac{p_{1j}(u_j)}{p_{2j}(u_j)} \right) \right]. \quad (4.32)$$

As possible distributions p_{ij}, [9] suggests Gaussians and non-parametric histograms. The distributions are updated, like the mean colour values for the standard Chan and Vese data terms, in each iteration of the level set evolution.
In addition to the already available colour channels from the original image data, the use of the *structure tensor* J is also proposed in [9] as additional features to capture the local structure of the image I:

$$J_\rho = K_\rho \star (\nabla I \nabla I^\top) = \begin{pmatrix} K_\rho \star I_x^2 & K_\rho \star I_x I_y \\ K_\rho \star I_x I_y & K_\rho \star I_y^2 \end{pmatrix}, \quad (4.33)$$

with K_ρ a Gaussian kernel with standard deviation $\sigma = \rho$. The operator $\star$ denotes convolution, so that J_ρ is smoothed. The elements $K_\rho \star I_x^2$, $K_\rho \star I_y^2$, and $K_\rho \star I_x I_y$ are then used as additional elements in the feature vector for each image pixel. Note that for multi-channel data like RGB images, the

values are simply added over all channels. Also note that in [9], not Gaussian smoothing, but non–linear diffusion is used to smoothen the structure tensor. This is done in order to reduce the blurring of edges which leads to inexact segmentations.
An example segmentation using Gaussian distributions of the colour channels and the structure tensor is shown in Figure 4.9.

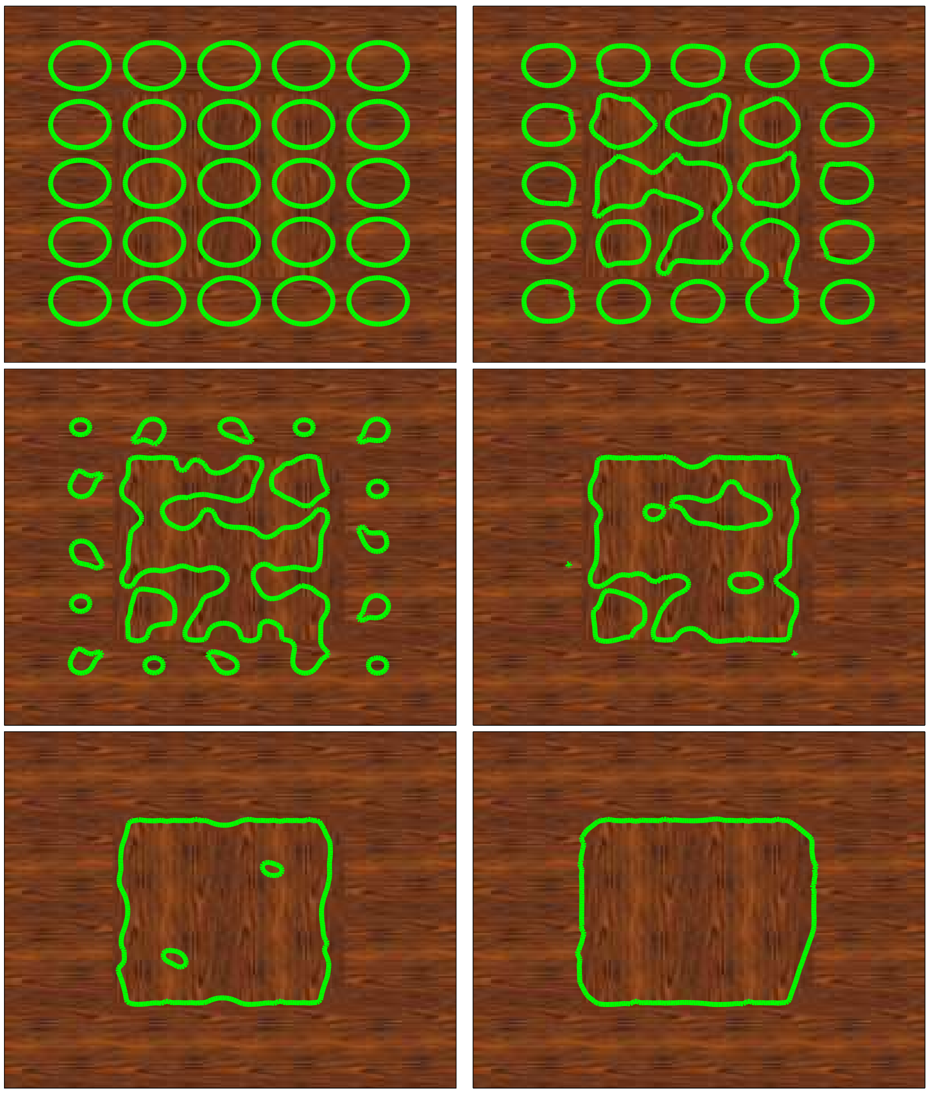


Figure 4.9: Example of using statistics based terms from [9] for level set segmentation. Here, a Gaussian of all three colour channels and the structure tensor was used in Equation (4.32). The weight for the curvature term was chosen to be $\mu = 0.0001 \cdot 255^2$.

4.9.4 Texture Segmentation with Wavelet Features

Besides the colour or grey value segmentation, segmentation using texture information has also been proposed in connection with the level set framework. Features capturing local texture information have already been introduced by using the non-linear structure tensor in Equation (4.33) for segmentation using statistics on a feature vector.
This is quite similar to using a set of wavelet features and one would expect the segmentation to be similar when using either the non-linear structure tensor or wavelet transform coefficients as image features. For dyadic wavelets, the transformed coefficients are

- A low pass filtered version of the original data
- A complementary high pass filtered version
- For 2-D images, the high pass coefficients consist of horizontally, vertically, and both horizontally and vertically high pass filtered versions of the original data

Consider discrete wavelet *frames* (see for example [12, 85, 41]), where the output of the complementary filters is not decimated like in usual wavelet transform implementations. Furthermore, consider Haar wavelets (see for example [12]). The output of this transform would then be very similar to the information conveyed by the non-linear structure tensor. Namely,

- Two high pass parts corresponding to the derivatives I_x, I_y in the structure tensor
- One high pass part corresponding to the diagonally directed derivative I_{xy} which takes the place of $I_x I_y$ in the structure tensor

Notice that the derivatives I_x, I_y are used in the structure tensor as squares, and that $I_x I_y$ is of course different from I_{xy}.
Nevertheless, from the wavelet decomposition one can get similar results, as is illustrated by comparing Figures 4.9 and 4.10, where the same image was segmented using colour and structure tensor on the one hand, and the coefficients of a discrete wavelet decomposition of the input image on the other hand.
Moreover, using more than one level of any hierarchical decomposition as image features also intrinsically introduces segmentation on multiple scales. Using wavelet frames, there is obviously no speed-up by using multiple scales, since the transform is not decimated.

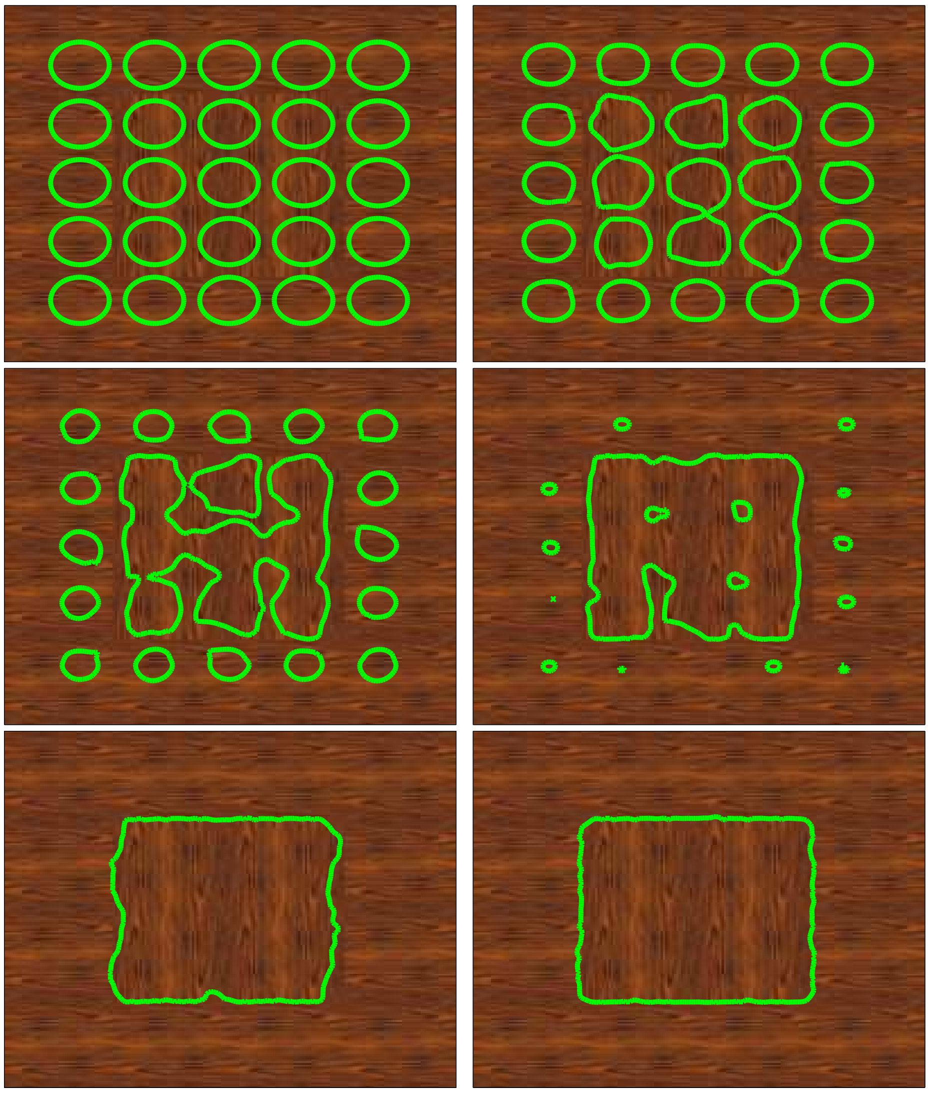


Figure 4.10: Example of using statistics based terms for level set segmentation. In this example, Gaussian distributions of all bands of a 2-stage discrete wavelet decomposition of the image were used in Equation (4.32). The weight for the curvature term was chosen to be $\mu = 0.0001 \cdot 255^2$. Comparing this to Figure 4.9, one can see that the results are similar, as expected.

4.9.5 Combination of Geodesic and Region Based Segmentation

Kimmel and Bruckstein [68, 69] integrate the geodesic active contour model (4.15) and the piecewise constant region terms from the region based active contour model (4.17) by using the former as regularisation term, replacing the length term in the original region based model. In fact, choosing the edge detector function g in (4.15) to be $g(x) = 1$ yields the original length term.
Sagiv, Sochen, and Zeevi [20] revive this idea for a texture segmentation approach, using the responses of a set of directed band pass filters as feature vectors for a model similar to the multi channel Chan and Vese model (4.20). As proposed previously in [68, 69], they replace the first, regularising term by a geodesic length term, using as g a function that detects discontinuities in texture rather than colour. Bresson et al. [138] also combine the geodesic boundary term and the region term, additionally introducing a shape prior.

4.9.6 Additional Regularising Terms

Curvature Diffusion Regularisation

In the context of parametric active contours, Delingette et al. [32] propose to push a curve towards a linear curvature profile by imposing a force

$$f_{\text{normal}}(s) = \left(\frac{1}{2 \cdot |s_0|} \int\limits_{s-s_0}^{s+s_0} \kappa(\sigma)\, d\sigma - \kappa(s) \right) \cdot \mathbf{n}(s) \tag{4.34}$$

which depends on the deviation from the mean curvature κ around a neighbourhood of size $2 \cdot s_0$ on the curve. Here, $\mathbf{n}(s)$ denotes the outward normal at point s on the curve.
The force (4.34) will drive a closed curve locally towards a smooth circle, without the effect of shortening the curve as does the length term in (4.19). Since using finite difference schemes to numerically calculate this force tends to be unstable, Delingette et al. propose a geometric approach using trigonometric functions on point triples involving the tangent of the local turning angle Φ of the curve.[2] This regularising force is meant to be used in the context of evolution of a parametric curve, for which Delingette's geometric implementation can readily be used. For an implicit method using level sets, Delingette proposes that to get decent estimates for each level set, one should extract parametric contour lines for each level set and apply (4.34) to the parametric contour. Working on the whole embedding function would mean to extract a parametric contour for each level set, in practice that means for

[2]There appears to be a typing error in the original paper where they explain the calculation of $L(r, \Phi, e)$ — the signs of parameter μ in the case switch is swapped.

a large number of level sets. This would be computationally expensive, but an approximation for a level set method has been proposed in the master thesis of Picinbono [104]. Equation (4.34) for two dimensions reads

$$F(u,v) = \left[\frac{1}{(2u_0+1)(2v_0+1)} \int_{u-u_0}^{u+u_0} \int_{v-v_0}^{v+v_0} \kappa(x,y)\,dx\,dy \right] - \kappa(u,v) \qquad (4.35)$$

where the neighbourhood is now $(2 \cdot u_0) \cdot (2 \cdot v_0)$. The mean curvature regularisation is approximated on the two dimensional embedding function by applying a linear filter to the curvature term $\nabla \cdot (\nabla\Phi/|\nabla\Phi|)$ in order to get a local mean of the curvature. In the implementation used in this thesis, the filter mask

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -24 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad (4.36)$$

was applied to the discretised curvature term. Trying Picinbono's approximation suggested that the results are quite acceptable, depending on the image and the chosen multiplicative factor for the force term (4.34). Figure 4.11 illustrates the impact of the approximation on an embedded curve: The contours are smoothed, but are not globally shrunk like when using mean curvature flow. Figure 4.12 illustrates the effect of Delingette's regulariser on a parametric curve. Here also, it can be seen that the curve is smoothed, but not shrunken very much. Local features tend to be preserved, depending on the width parameter s_0 in (4.34).

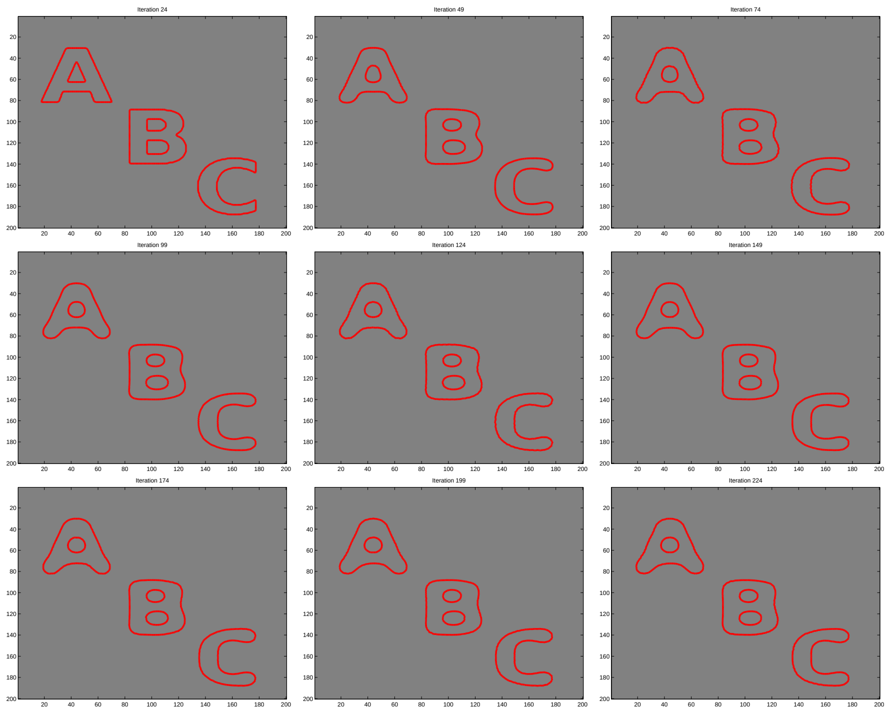


Figure 4.11: Influence of the level set approximation of Delingette's curvature diffusion regularisation (4.34) [32, 104]. The zero level set is smoothed, but is not shrinked as opposed to the mean curvature flow. See also Figure 4.3.

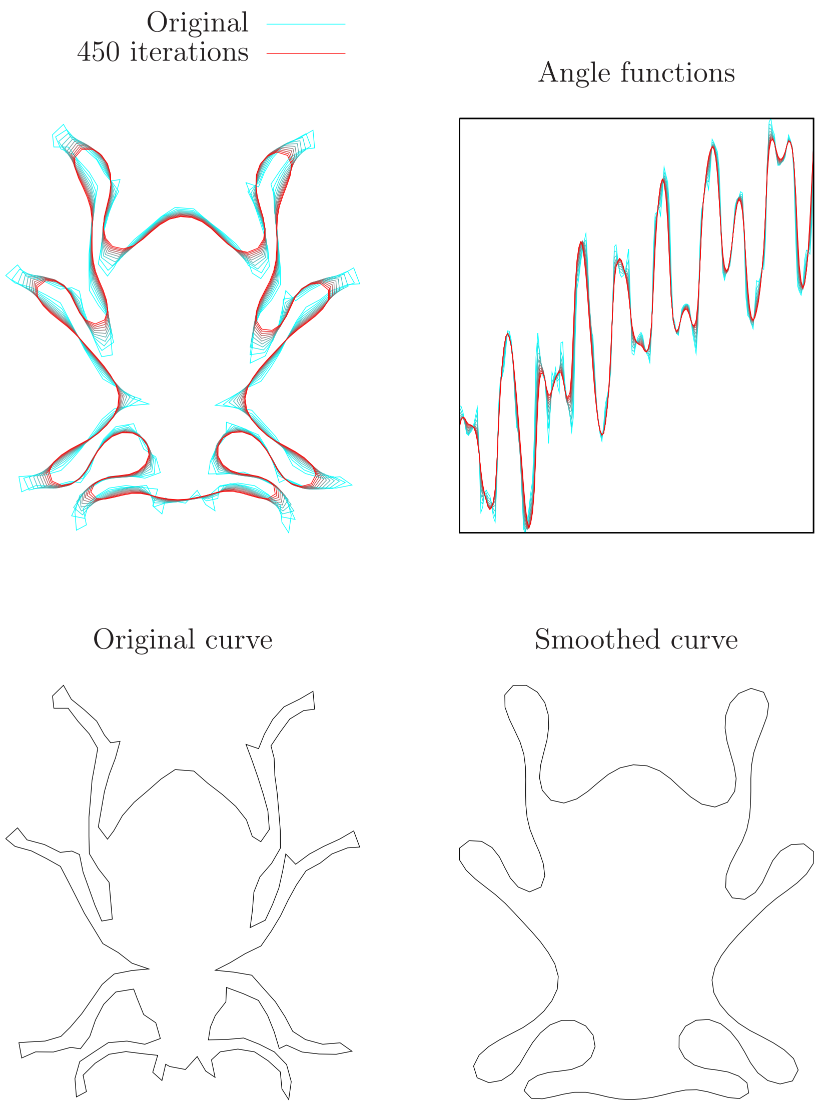


Figure 4.12: Illustration of the effect of Delingette's curvature diffusion regularisation force, applied to a parametric curve. This shown curve, discretised with 200 points, is smoothed for 450 iterations with a neighbourhood size of 5 points to each side. Top left: Original and smoothed curve, with intermediate steps shown every 50 iterations. Top right: The corresponding angle functions of the curves, showing the smoothing effect there as well. Bottom row: Original curve and the smoothed curve after 450 iterations.

Signed Distance Penalisation

Li et al. [21] suggest to add another regularisation term $P(\Phi)$ to a segmentation energy such as (4.17). $P(\Phi)$ is designed to penalise the deviation of the embedding function Φ from the signed distance property and is given as

$$P(\Phi) = \int_{\Omega} \frac{1}{2}(|\nabla\Phi| - 1)^2 \, dx \, dy \,, \tag{4.37}$$

which leads to an additional internal force

$$\left.\frac{\partial\phi}{\partial t}\right|_{Li} = \alpha \left(\Delta\Phi - \nabla \cdot \left(\frac{\nabla\Phi}{|\nabla\Phi|} \right) \right) . \tag{4.38}$$

Even though they claim adding this term made re-initialisation obsolete for their implementation of geodesic active contours, this term adds a very strong regularisation which smoothes the contour very much. See Figure 4.13 for an example of the Chan and Vese model with and without the term (4.38). It can be seen that the curve is smoothed very much, while the signed distance property is not maintained very well.
By adding the regularisation term (4.38), *the energy minimisation itself drives the embedding function towards the signed distance property.*
The signed distance property is merely a property that leads to more stable evolution and nothing that we particularly demand of the embedding function in order to minimise an energy that was originally crafted to yield image segmentation. It is therefore intuitively better to let the embedding function evolve so that the original energy functional is minimised and to re-initialise from time to time in order to get the embedding function into a shape which allows stable evolution, but *without changing the interface curve.* Note that in practice, when using the standard approach for re-initialisation from Equation (4.45) [120], there is a usually small displacement of the interface curve which depends on the used discretisation. However, the effect of this has only a small impact on the final result compared to adding a term like (4.38).

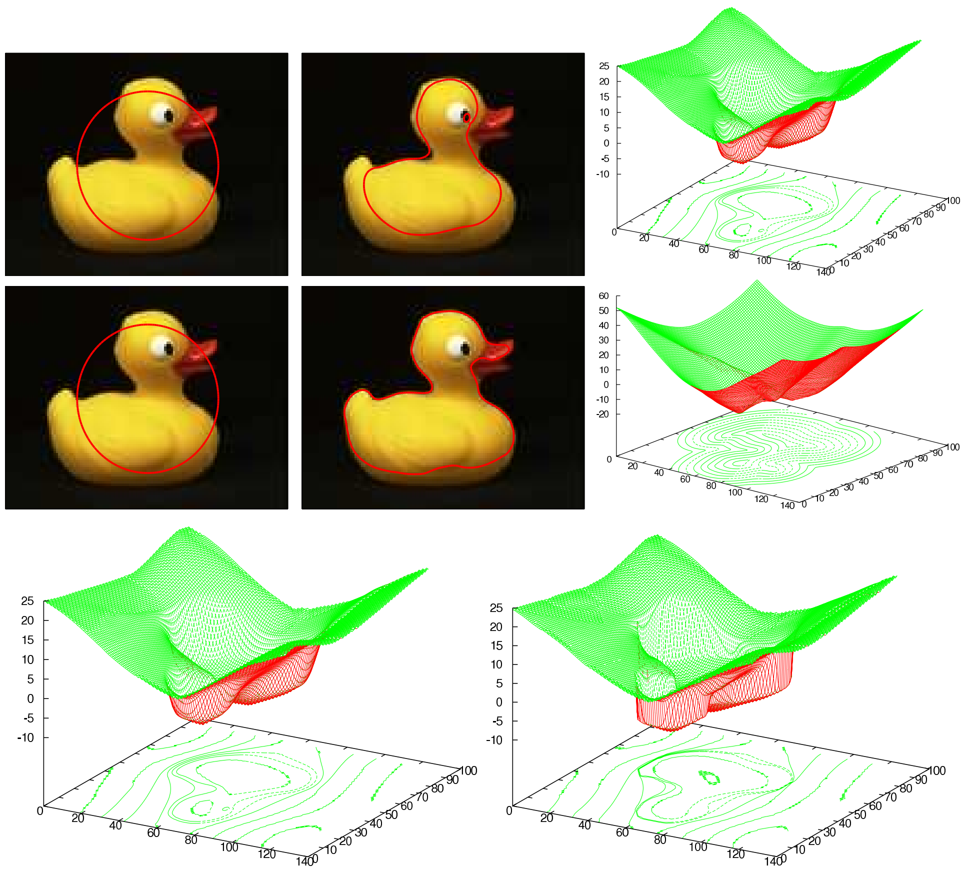


Figure 4.13: Influence of the term (4.38) on level set evolution. Top row: initialisation and segmentation result with the term (4.38) with weight $\alpha = 0.4 \cdot 255^2$. Middle row: initialisation and segmentation result without term (4.38), but with re-initialisation every 10 time steps. In both cases, $\mu = 0.1 \cdot 255^2$ (see Equation (4.19)). The rightmost image shows the resulting embedding function Φ with contour lines indicated at the bottom. Bottom row: resulting embedding function when using term (4.38) (left side) and when using neither this nor re-initialisation (right side). It can be seen in the top row that the signed distance property is hardly maintained, even though it seems better than when using neither regularisation nor re-initialisation, as shown in the bottom row. However, the unwanted smoothing effect on the contour can clearly be seen, and the result of just using re-initialisation looks much more pleasing.

4.9.7 Prior Knowledge

Using a level set algorithm for image segmentation in a setting with more realistic images, one faces a few problems. Those problems can be summarised under the keywords noise, clutter, and occlusion. *Noise* can depend on detector quality, the circumstances of image acquisition, or the modality of the data. For example, images taken at night will usually contain a certain amount of colour noise, or images taken in the rain will contain a bunch of information on rain drops which we may not be interested in; images taken with a high-quality astronomical camera might result in less noisy data than still images from an ultra sound device.
Clutter usually refers to non-smooth image areas which do not belong to any object, and which make the segmentation process considerably harder.
Occlusion of course means that parts of the object we would like to segment are not visible, but are for example occluded by other objects in the foreground.

In order to alleviate some of the problems arising with realistic images, *prior knowledge* about the objects to be segmented has been introduced by a number of researchers, for example in [78, 79, 19, 30, 98, 106, 138, 144, 61]. This knowledge, or shorthand *prior*, can in energy minimisation schemes be introduced as an extra term in the energy that is to be minimised.
Chen et al. have used a simple model of shape to steer a level set evolution in [19]. In the simplest case, the shape is the average curve $C^\star$ of a set of known curves, which may stem from a set of example images manually segmented by a human expert. Chen adds a distance term to the geodesic active contour model, which then reads

$$E(C, \mu, R, T) = \int_0^1 \left(g(|\nabla I|\, C(p)) + \underbrace{\frac{\lambda}{2}\, d^2(\mu\, R\, C(p) + T)}_{\text{prior term}} \right) |C'(p)|\, dp \tag{4.39}$$

where $d(\cdot)$ is the distance function of the argument from the prior curve $C^\star$. μ, R, and T are a scaling factor, rotation matrix, and translation vector. The energy functional is then minimised with respect to these parameters as well.

Leventon et al. [79] have incorporated a statistical prior knowledge term in the geodesic active contour model. They represent each of N given training curves as signed distance functions u_i. The variability of the training set is calculated with principal component analysis, which had been used previously to model shape variability [57]. Using the matrix

$$M = (\operatorname{vec}(u_1 - \mu), \ldots, \operatorname{vec}(u_N - \mu))\,,$$

with $\operatorname{vec}(\cdot)$ an operator that creates a column vector from a d-dimensional grid, and with

$$\mu = \frac{1}{N}\sum_{i=1}^{N} u_i\,,$$

the eigenvalue decomposition $U\,\Sigma\,U^\top = 1/N\,M\,M^\top$ of the covariance matrix gives the principal modes of variation as the eigenvectors, which are the columns of U. Using U, any signed distance function u, embedding a curve, is approximated using $\alpha = U_k^\top\,(u-\mu)$, the coefficients of the first k principal components of the model learned from the training samples. The vector α gives an approximation of u by $u \approx U_k\,\alpha + \mu$. Leventon then uses the vector α and eigenvalue matrix Σ to model the probability of a curve with a Gaussian distribution. α and the pose of the target curve are estimated in each evolution step and an additional force term is added to the update equation.
Notice that the dimension of $M\,M^\top$ is K^2, where K is the number of grid points used to represent each embedding function. So, implicitly representing the $(d-1)$-dimensional level sets, say a curve for $2D$ images or a surface for $3D$ images, results in a considerable inflation of the amount of data compared to a parametric representation.
An approach similar to Leventon [79] has been taken in [132].

Cremers et al. describe linear and non-linear statistical shape models for segmentation with a parametric snake model [28, 25, 26, 30]. The linear model comprises a principal component analysis of a set of properly aligned training curves. The model is incorporated in the energy minimisation scheme by adding a shape energy, for which it is suggested to use the Mahalanobis distance [84] of the evolving curve, given the model computed from the training set.

Bresson et al. [138] combine the shape term from Chen et al. [19] and Leventon et al. [79], yielding a term depending on the PCA coefficients of a shape and its euclidean transformation parameters with respect to the segmented image.

Riklin-Raviv et al. [106] extend the standard Chan and Vese model as introduced in Section 4.9.2 with a prior knowledge term. Their prior is exactly one fixed embedding function $\tilde{\Phi}$, which they transform similar to Chen et al. [19]: they additionally allow for perspective transformations of the prior embedding function $\tilde{\Phi}$, which is the *generalised cone* consisting of all rays passing through a fixed vertex and the points on the contour in the image plane. Thereby, this model allows for a certain amount of perspective distortion of the resulting prior contour, in addition to euclidean transformations. Their shape energy is formulated in terms of the area of non-overlapping regions of the current contour embedded in Φ, and the transformed prior contour embedded in $T(\tilde{\Phi})$, where $T(\cdot)$ denotes the transformation:

$$E_{\text{Riklin-Raviv}}(\Phi) = \int_\Omega \Big(H(\Phi) - H(T(\tilde{\Phi}))\Big)^2\,do\,.$$

The transformation parameters are optimised for in alternation with the embedding function Φ.

This energy also has the effect that the term does not depend on the size of the background area anymore [107], as in the method from Chen in (4.39). This question has also previously been tackled by Cremers and Soatto [29], see also references therein. For this reason, it may be a good idea to consider it for prior integration into a level set method also without perspective transformations as

$$E_{\text{Shape}} = \frac{1}{2}\int_{\Omega}\Big[H(\Phi(x)) - H(\tilde{\Phi}(s\,\Gamma\,x + T)\Big]^2\,dx\,. \tag{4.40}$$

Note that T now denotes translation. In order to find directions for gradient descent, one derives

$$\begin{aligned}
\frac{dE_{\text{Shape}}}{dT} &= -\int_{\Omega}\Big[H(\Phi(x)) - H(\tilde{\Phi}(s\,\Gamma\,x + T))\Big] \\
&\quad \cdot H'(\tilde{\Phi}(s\,\Gamma\,x + T)) \cdot \nabla\tilde{\Phi}(s\,\Gamma\,x + T)\,dx && (4.41) \\
\frac{dE_{\text{Shape}}}{ds} &= -\int_{\Omega}\Gamma\,x\,\Big[H(\Phi(x)) - H(\tilde{\Phi}(s\,\Gamma\,x + T))\Big] \\
&\quad \cdot H'(\tilde{\Phi}(s\,\Gamma\,x + T)) \cdot \nabla\tilde{\Phi}(s\,\Gamma\,x + T)\,dx && (4.42) \\
\frac{dE_{\text{Shape}}}{d\theta} &= -\int_{\Omega}\Big[H(\Phi(x)) - H(\tilde{\Phi}(s\,\Gamma\,x + T))\Big] \\
&\quad \cdot H'(\tilde{\Phi}(s\,\Gamma\,x + T)) \cdot s \cdot \frac{d\Gamma}{d\theta} \cdot \nabla\tilde{\Phi}(s\,\Gamma\,x + T)\,dx\,, && (4.43)
\end{aligned}$$

assuming that

$$\Gamma = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

The transformation parameters are optimised for in turn with the augmented general segmentation energy functional

$$E = \mu\,E_{\text{Smooth}} + E_{\text{Data}} + \alpha\,E_{\text{Shape}}\,. \tag{4.44}$$

The procedure is given in Algorithm 5.

To illustrate the process of segmentation with a prior template, Figure 4.14 shows a sequence of a segmentation using the colour based, two-phase segmentation of Chan and Vese [131], including the prior term (4.40). The same image was segmented without prior and the result is shown in Figure 4.15. Clearly, the gaps in the cross-shaped structure can be filled by using the simple template prior, while the normal segmentation procedure unsurprisingly results in several connected components.

Algorithm 5 Level set segmentation with prior (4.40).

```
Require: $\Phi$: embedding signed distance function, $\tilde{\Phi}$: embedding signed
    distance function of the prior contour, $s, \theta, T$: initial scale, rotation, and
    translation parameters
1: while $\Phi$ has not reached steady state do
2:     Update $\Phi$ using gradient descent of $E$ in (4.44), for one time step
3:     $s, \theta, T \leftarrow \arg\min_{s,\theta,T} E_{\text{Shape}}$    ▷ from Equation (4.40), using (4.41),
    (4.43), optionally (4.42)
4: end while
```

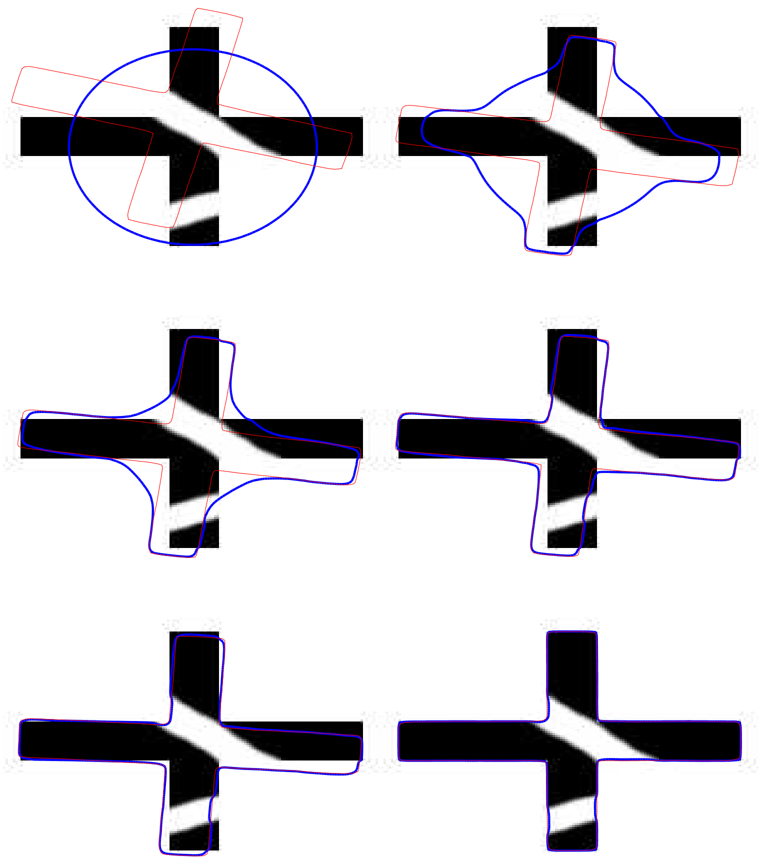


Figure 4.14: Level set segmentation with prior using (4.44), with $\mu = 0.2, \alpha = 0.8, \lambda_{1,2} = 1$. The data term is the colour based term after Chan and Vese [131]. The upper left image shows initialisation, the lower right shows the final result. The red, thin line outlines the prior template, the blue line outlines the zero level set. See also Figure 4.15.

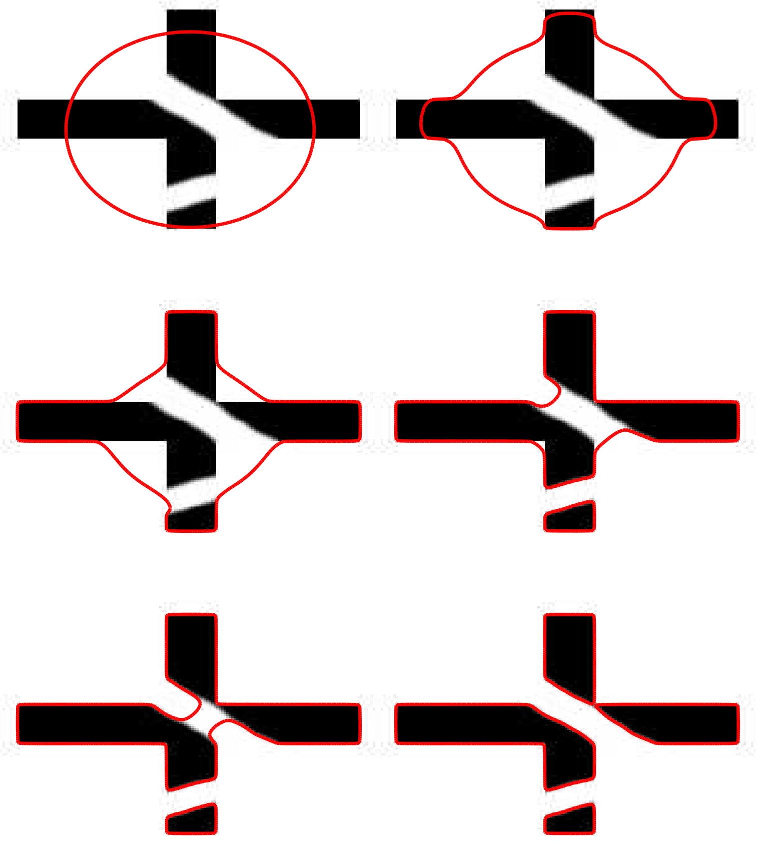


Figure 4.15: Level set segmentation without prior after (4.44), with $\mu = 0.2, \alpha = 0, \lambda_{1,2} = 1$. The data term is the colour based term after Chan and Vese [131]. The upper left image shows initialisation, the lower right shows the final result. The red line outlines the zero level set. See also Figure 4.14 for the same segmentation with a prior.

More recently, Rousson and Paragios [110] use a prior energy based on pixel-wise Gaussian distributions of aligned training samples given as signed distance functions $\{\tilde{\Phi}_1, \ldots, \tilde{\Phi}_n\}$. From these, an average $\tilde{\Phi}_m$ and standard deviation $\tilde{\sigma}_m$ are calculated. These are not signed distance functions any longer; this issue is resolved in [110] by applying signed distance re-initialisation and using the resulting Φ_m and "confidence map" σ_m in place of the mean and standard deviation for the Gaussian distributions.

4.10 Initialisation

The results of Chan and Vese's region based level set method for image segmentation depend on the initial embedding function Φ_0. The reason is that calculating on the zero level curve, one ends up in a local extremal of the energy functional, which is non-convex. Depending on how Φ is initialised, different local optima will be found. A few widely used possibilities for the initial zero level set are

1. One centred circle of varying radius.
2. Several small circles, spread evenly across the image.
3. One large box near the boundary of the image. This has mainly been used for geodesic active contours. For region based active contours, it does not make a lot of sense, since one of the two initial regions will be initially very small compared to the other.
4. An ellipse or circle overlapping the object of interest. This is then of course depending on the individual image.

One can also use the iso curve of the mean grey value of an image as initial zero level curve. This has proven to give good results on a few images, which can be understood when realising that the assumption we implicitly make about an image is that we can segment it into two more or less homogeneous regions. So at least for the image term in (4.19), it is at least not counter-intuitive that the mean grey value iso curve will in many cases give a good initialisation. Figure 4.16 shows results for a few different initialisations for one image.

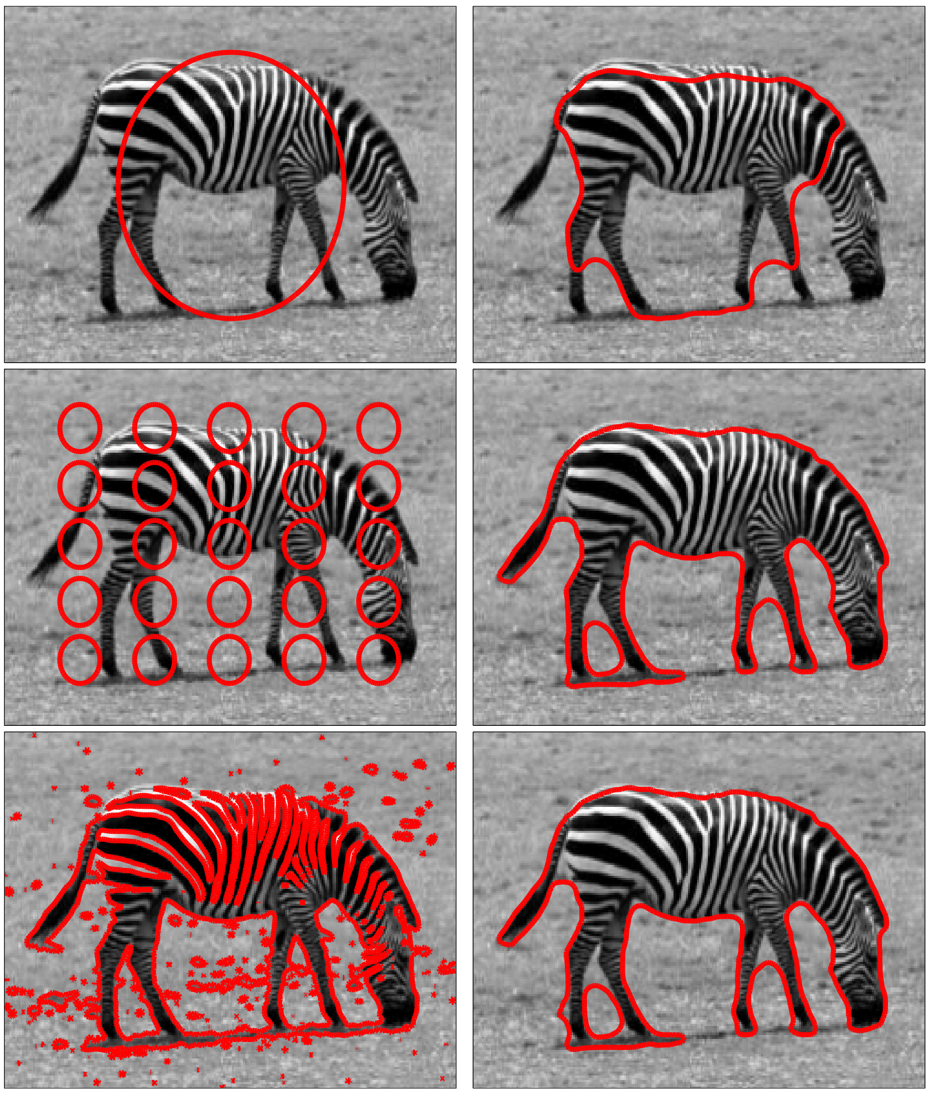


Figure 4.16: Impact of initialisation on level set segmentation. Left column: Initialisation. Right column: Respective segmentation results. One can see how different initialisations lead to different results. The region based stopping condition (4.50) was used. All parameters are the same for all three segmentations — only initialisations differ. Clearly, the segmentation ends up in a different local minimum for the first initialisation (one big circle) than for the other two. Using multiple small circles (middle) or mean grey value (bottom) as initialisation in this case apparently lead to the same minimum, or at least very similar minima. Notice that this is not necessarily so. However, a grid of small circles seems to be a good choice in many experiments. The data term used here is the Gaussian grey value distribution from [109] described in Section 4.9.3.

4.11 Re-Initialisation (a.k.a. Re-Distancing)

The signed distance function has nice properties with respect to the behaviour of the level set evolution. Since $|\nabla\Phi| = 1$, the normal of the embedding function Φ becomes

$$N = \operatorname{div}\frac{\nabla\Phi}{|\nabla\Phi|} = \Delta\Phi,$$

where Δ is the Laplace operator, and working with normals becomes much simpler. Also, there are no areas where $|\nabla\Phi|$ is extraordinarily large or small, which can cause numerical problems. However, even if the embedding function is initialised to be a signed distance function, this property is in general not automatically maintained by the evolution process. This means the signed distance function will generally develop into something different during evolution. This is why many researchers decide to use a re-initialisation procedure to regain the signed distance property during evolution. The standard approach introduced in [128] is to solve the *re-initialisation equation*

$$\Phi_t + S\left(\Phi_0\right) \cdot \left(|\nabla\Phi| - 1\right) = 0 \tag{4.45}$$

to steady state. $S(\cdot)$ denotes the sign of its argument, Φ_0 is the initial embedding function. In [128, 120] it is mentioned that using a smeared version of the sign function S adds numerical stability, as they found out by experimentation; Sussman et al. [128] suggest using

$$S(\Phi_0) = \frac{\Phi}{\sqrt{\Phi_0^2 + (\Delta x)^2}}.$$

For functions Φ which are initially far from a signed distance function, [99] introduces

$$S(\Phi) = \frac{\Phi}{\sqrt{\Phi^2 + |\nabla\Phi|^2(\Delta x)^2}}\,. \tag{4.46}$$

Equation (4.46) then has to be updated in each step when solving (4.45). It does in fact work better in practice and can even be used to initialise a signed distance function from an initially binary function Φ_0. An illustration is shown in Figure 4.17.

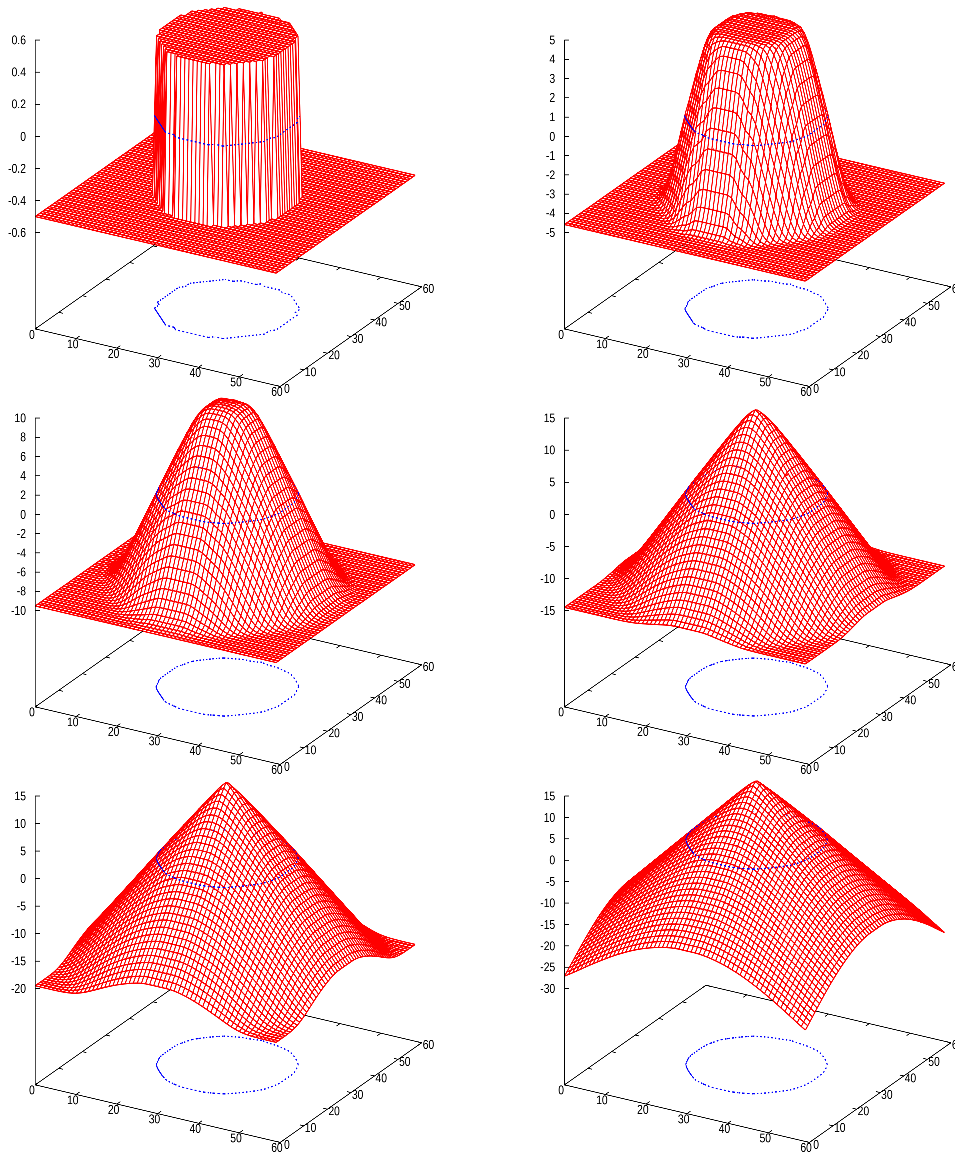


Figure 4.17: Illustration of re-initialisation by solving Equation (4.45) with the sign approximation (4.46). The initial Φ_0, shown at the top left, is a binary function with values $\{-0.5, 0.5\}$. The following graphs show the re-initialisation at intermediate steps and at the lower right the final result can be seen. Clearly, the indicated zero level set changes slightly during the re-initialisation. Compare also with Figure 4.18. A direct comparison of the resulting zero level sets can be seen in Figure 4.19.

Ideally, the interface $\{x|\Phi(x) = 0\}$ will not be changed by re-initialisation. In numerical implementations, however, the interface will be moved, as indicated in Figures 4.17 and 4.19. To minimise this effect, Sussman et al. [127] introduce a local area preserving constraint, demanding that the area inside (and outside) the boundary does not change. This is done by adding a term on the right hand side of Equation (4.45):

$$\Phi_t + S(\Phi_0) \cdot (|\nabla\Phi| - 1) = \lambda \cdot H'(\Phi) \cdot |\nabla\Phi| \,. \tag{4.47}$$

The right hand side will only be acting on grid points close to the interface. Taking that the area of a grid cell (i, j) is $A_{i,j} = \int_{\Omega_{i,j}} H(\Phi)\, dx$, demanding the area change to be zero amounts to

$$\int_{\Omega_{i,j}} H'(\Phi) \cdot \Phi_t \, dx = 0 \,. \tag{4.48}$$

Inserting Φ_t from Equation (4.47) into (4.48) leads to λ:

$$\int_{\Omega_{i,j}} H'(\Phi) \cdot \left[-S(\Phi_0)\,(|\nabla\Phi| - 1) + \lambda\, H'(\Phi)\, |\nabla\Phi| \right] \, dx = 0$$
$$\Rightarrow \quad \lambda_{i,j} = \frac{\int_{\Omega_{i,j}} H'(\Phi)\, S(\Phi_0)\, (|\nabla\Phi| - 1)\, dx}{\int_{\Omega_{i,j}} H'^2(\Phi)\, |\nabla\Phi|\, dx} \,.$$

One then calculates λ and uses it in (4.47). This retains the zero level set more accurately in comparison to (4.46) at the cost of a higher computational burden. An illustration can be found in Figure 4.18. It can be seen that the zero level set changes a lot less than in Figure 4.17. A direct comparison between the zero level sets resulting from re-initialisation and the initial zero level set can be seen in Figure 4.19.

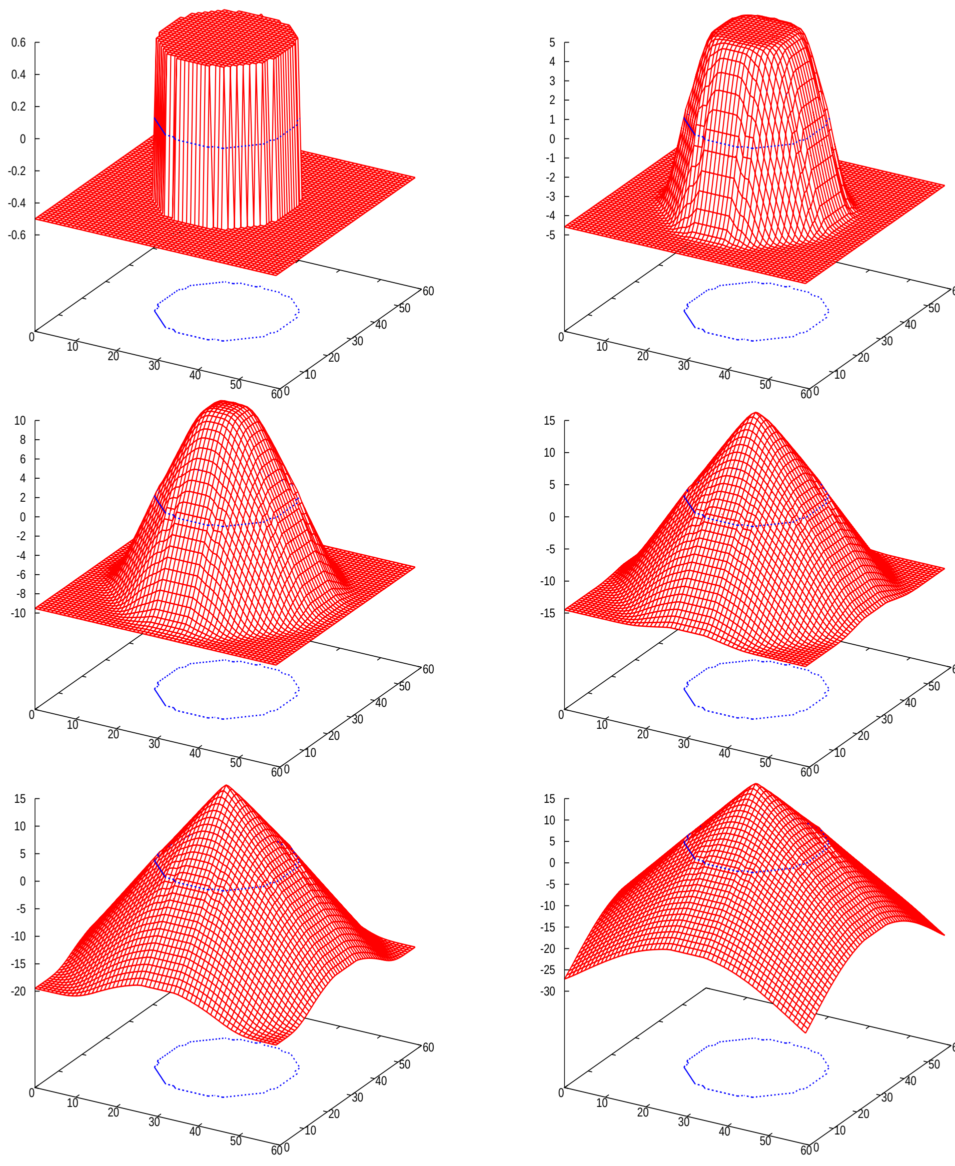


Figure 4.18: Illustration of re-initialisation by solving Equation (4.47) with the sign approximation (4.46). The initial Φ_0, shown at the top left, is a binary function with values $\{-0.5, 0.5\}$. The following graphs show the re-initialisation at intermediate steps and at the lower right the final result can be seen. Observe here that the indicated zero level set is changed less than in Figure 4.17. A direct comparison of the resulting zero level sets can be seen in Figure 4.19.

Osher and Fedkiw [120] note that this approach significantly improves upon spatial discretisations using low order polynomial approximations, but that more accurate discretisations may not need this extension.

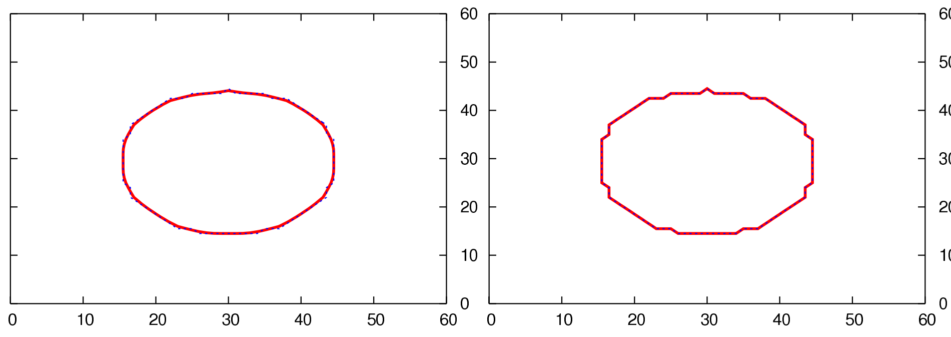


Figure 4.19: Comparing the resulting and initial zero level sets of re-initialisation using Equation (4.45) with (4.46) (left hand side) and Equation (4.47) (right hand side). The initial zero level set is indicated in blue, dashed lines, the resulting re-initialised zero level set in a red, thicker line. Clearly, on the left side the result is moved more from the initialisation than on the right side.

4.11.1 Drawbacks of Re-Initialisation

The practice of re-initialising the embedding function is beneficial to the stability of the level set method, but is also somewhat random. It is unclear if and when the re-initialisation should take place. After 10 iterations? 100 iterations? When a fraction of the gradient of Φ is "sufficiently far" away from one? The choice of if and when to re-initialise seems arbitrary. Often, authors state that they re-initialise after each time step of the actual gradient descent. This is crucial when the implementation numerically relies on the signed distance property.

There are other approaches to force Φ to be a signed distance function, for example one by Li [21] where a term for penalising deviation from the signed distance property was added to the energy functional that is being minimised by the level set evolution. Restating from Section 4.9.6, this term has a strong regularising effect that can smoothen the zero level curve more than may be wanted by the user; in fact, one can even leave out the mean curvature term from the standard Chan and Vese model if using Li's regulariser. Since the aim of the energy minimisation is to yield a segmentation and not to push Φ to signed distance, adding a term as in [21] is not a good choice.

Re-initialisation, having very little impact on the zero level curve itself, therefore appears to be by far superior.

Another more recent approach that tries to free the level set method from the need for re-initialisations was provided by Gelas et al. in [44]. They model the embedding function Φ and the zero level set with compactly supported radial basis functions to get a continuous representation.

4.12 Stopping Condition

Minimising an energy, we would like to stop the evolution process when a steady state is reached, that means when the energy is not further minimised by the gradient descent. In terms of a software implementation, one can continue the evolution until the energy E_t no longer decreases. In practice, when the numbers involved are on the order of magnitude of machine precision, one can use the condition

$$E_t > E_{t-1} \,. \tag{4.49}$$

Another possibility is to examine the step $\Delta\Phi$ and stop whenever

$$\max\{\Delta\Phi\} < \varepsilon$$

for some small, fixed ε.
A third way to detect stationarity is to examine the change of the inside (or outside) region of the zero level set within n time steps,

$$D = \int_\Omega \left[H(\Phi_{t-n}) - H(\Phi_t)\right]^2 \, dx \,,$$

and stopping if

$$D < \varepsilon \,. \tag{4.50}$$

Note that using the actual energy value as stopping condition only works when we are capable of calculating the energy. For terms like the *curvature diffusion regularisation* in Equation (4.35), where only a force is given, it is not clear how to calculate the energy. Therefore, it is generally a good idea to use the region based stopping condition (4.50).

4.13 Numerical Implementations

This section describes the numerical methods that were used to create the segmentation examples shown in this thesis. Given a partial differential equation such as (4.19), one seeks a numerical solution. The implementation used in this work uses finite differences and an *explicit scheme*, which simplifies the task of augmenting the basic method with new experimental terms. Such terms can for example be additional priors, or different regularising terms. Fully implicit schemes require considerably more effort, but on the other hand they are unconditionally stable and allow for large time steps. Such schemes may therefore be of interest in cases where the involved system of equations can be expected to remain the same, and where these systems can be compiled *and* solved efficiently.

4.13.1 Finite Differences

When calculating with data defined on a discrete grid, one has to approximate derivatives numerically. Let us assume that the grid is always regular. A widely used method that is relatively easy to use, are finite differences. The derivative of a function $f(x)$ can be approximated in several ways. Consider the forward (4.51), backward (4.52), and central (4.53) differences

$$\begin{aligned} \Delta_{+x} f(x) &\coloneqq f(x+\Delta x) - f(x) \qquad (4.51)\\ \Delta_{-x} f(x) &\coloneqq f(x) - f(x-\Delta x) \qquad (4.52)\\ \delta_x f(x) &\coloneqq f(x+\frac{1}{2}\Delta x) - f(x-\frac{1}{2}\Delta x) \qquad (4.53) \end{aligned}$$

with the grid spacing Δx. These can be used as approximation of derivatives by multiplying by a factor $1/\Delta x$ — but care must be taken in many cases as to which approximation is appropriate and which is not. A prominent case is the upwind scheme which will be mentioned further down in this section in the context of the discretisation of the signed distance re-initialisation.
A second order central difference can be obtained by applying the central difference operator twice:

$$\delta_x^2 f(x) \coloneqq f(x+\Delta x) - 2\, f(x) + f(x-\Delta x)\,. \qquad (4.54)$$

4.13.2 Temporal Discretisation

Discretising the left hand term of

$$\frac{\partial \Phi}{\partial t} + f(\Phi) = 0 \qquad (4.55)$$

can be done in the simplest case with a forward difference as

$$\frac{\Phi_{n+1} - \Phi_n}{\Delta t} + f(\Phi) = 0\,. \qquad (4.56)$$

This is also called *forward Euler* scheme and leads to a truncation error of first order, $O(\Delta t)$. More accurate schemes for the temporal discretisation can be used, although [120] notes that in most cases, forward Euler discretisation suffices. If necessary, higher accuracy can be achieved by applying a higher order Runge-Kutta scheme (see e.g. [137]): At time step n, calculate Φ_{n+1} with the forward Euler method. Using this, go one step further to get Φ_{n+2} in the same manner and calculate a weighted average

$$\Phi_{n+1} = \frac{1}{2}\left(\Phi_n + \Phi_{n+2}\right).$$

This scheme would be second order accurate.
A third order scheme starts like the second order scheme, but in the averaging step one calculates

$$\Phi_{n+\frac{1}{2}} = \frac{3}{4}\,\Phi_n + \frac{1}{4}\,\Phi_{n+2}\,,$$

then uses this intermediate solution to calculate another forward Euler step to yield $\Phi_{n+\frac{3}{2}}$, and finally averages again to get

$$\Phi_{n+1} = \frac{1}{3}\,\Phi_n + \frac{2}{3}\,\Phi_{n+\frac{3}{2}}\,.$$

However, now having mentioned higher order approximations for the time discretisation, all results depicted in this thesis were obtained using the simple forward Euler scheme.

4.13.3 Spatial Discretisation

Curvature Term

The term

$$\operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right) \tag{4.57}$$

in the basic evolution equation (4.19) for the Chan and Vese model can for a signed distance function with $|\nabla\Phi| = 1$ be simplified to

$$\nabla\cdot\nabla\Phi = \Delta\Phi = \sum_{i=1}^{d} \frac{\partial^2\Phi}{\partial x_i^2}\,.$$

Here, d is the dimensionality of the domain of Φ; in the case of 2D images, $d = 2$.
Then, leaving all other terms out of the evolution equation, one ends up with the *heat equation*[3]

$$\frac{\partial\Phi(x,t)}{\partial t} = \sum_{i=1}^{d} \frac{\partial^2\Phi(x,t)}{\partial x_i^2}\,.$$

For the sake of simplicity, let us look only at one dimension. One can then write the discretisation

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} = \frac{\delta_x^2\Phi^n}{(\Delta x)^2}\,, \tag{4.58}$$

assuming a regular grid with grid spacing Δx.

[3]This equation is used to model the distribution of heat over time. The same equation is also called *diffusion equation*, since it is also used to model diffusion processes.

Not relying on the signed distance property, one can also choose to use a direct discretisation of (4.57) by using a forward difference for $\nabla\Phi/|\nabla\Phi|$ and a backward difference for the second derivatives (or vice versa),

$$\operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right) \approx \sum_{i=1}^{d} D_{-x_i}\left(\frac{D_{+x_i}\Phi}{\sqrt{\sum_{j=1}^{d}(D_{+x_j})^2+\varepsilon}}\right), \tag{4.59}$$

where we set

$$D_{\pm x_i} f(x) := \frac{\Delta_{\pm x_i} f(x)}{\Delta x_i}\,.$$

The term ε is a small constant to guarantee numerical stability at grid points where the forward differences are near zero, that is near machine precision in a computer implementation.

Re-initialisation

The differential equation (4.45) to solve for re-initialisation is

$$\Phi_t + S(\Phi_0)\cdot(|\nabla\Phi| - 1) = 0 \tag{4.60}$$

and describes motion in normal direction similar to the *level-set equation* (4.13). When discretising (4.60), some care must be taken concerning the term $S(\Phi_0)\cdot|\nabla\Phi|$. We must use an upwind scheme (see Appendix A.2), writing

$$S(\Phi_0)\cdot|\nabla\Phi| \tag{4.61}$$

as

$$\underbrace{\left(S(\Phi_0)\frac{\nabla\Phi}{|\nabla\Phi|}\right)}_{=:a}\nabla\Phi\,. \tag{4.62}$$

But now, the sign of term a depends on the sign of $S\,\nabla\Phi$. This is not a problem as long as the signs of the forward and backward differences $\Delta_+\Phi$ and $\Delta_-\Phi$ are equal. If they are not, the upwind scheme will break down since the choice of forward or backward difference alters the sign of a. This is the case in "V"-shaped situations as indicated in Figure 4.20. At the cusp, the forward and backward differences yield different signs. These situations are handled explicitly by Godunov's scheme [120, 111].

Let us look at only one dimension in the following. The choice for a finite difference is made for each dimension independently.

If $S\,\Delta_{-x}\Phi \le 0$ and $S\,\Delta_{+x}\Phi \ge 0$, the motion is expansive, which means information flows out towards the left side and towards the right side. Godunov's scheme sets the derivative $\Phi_x = 0$ in this case. If $S\,\Delta_{-x}\Phi \ge 0$ and $S\,\Delta_{+x}\Phi \le 0$, information is flowing in from both directions. In this case, Godunov's scheme chooses the direction from which information flows in faster, that means it chooses $\Phi_x = \Delta_{-x}\Phi$ if $|S\,\Delta_{-x}\Phi| > |S\,\Delta_{+x}\Phi|$, and

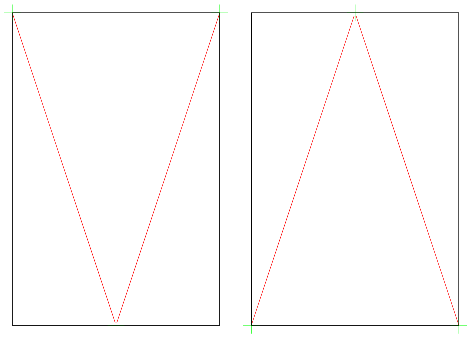


Figure 4.20: Critical situations handled by Godunov's scheme.

$\Phi_x = \Delta_{+x}\Phi$ otherwise.
In case $S\,\Delta_{-x}\Phi$ and $S\,\Delta_{+x}\Phi$ both have the same sign, Godunov's scheme gives the same results as the simple upwind scheme.

4.13.4 Implicit Methods

Instead of assuming only values from the past in calculating finite differences, one can also choose to use values from the *future*. This results in *implicit finite difference schemes* which are always stable and convergent, but at a higher computational cost per time step compared to explicit schemes. Also, the implementation effort is usually higher for implicit schemes. Generally, the approximation of a PDE in our level set setting is then

$$\frac{\Phi_{i+1} - \Phi_i}{\Delta t} = -\nabla E(\Phi_{i+1}) \tag{4.63}$$

and we are seeking the values for Φ_{i+1} at time step t_{i+1}. Transforming (4.63), one gets the system

$$\Phi_{i+1} + \nabla E(\Phi_{i+1})\,\Delta t = \Phi_i$$

which must then be solved in each time step. For a simple example, consider the heat equation

$$\frac{\partial \Phi}{\partial t} - \nabla^2 \Phi = 0\,.$$

Using second order central differences for $\nabla^2\Phi$, the discretisation then reads

$$\frac{\Phi_{i+1}(x) - \Phi_i(x)}{\Delta t} = \frac{\Phi_{i+1}(x+1) - 2\,\Phi_{i+1}(x) + \Phi_{i+1}(x-1)}{h^2}\,.$$

For the sake of clarity, the above equation is written only for one space dimension with grid spacing h. Then, solving for $\Phi_i(x)$,

$$\Phi_{i+1}(x)\,\frac{h^2}{\Delta t} - \left(\Phi_{i+1}(x+1) - 2\,\Phi_{i+1}(x) + \Phi_{i+1}(x-1)\right) = \Phi_i(x)\,\frac{h^2}{\Delta t} \tag{4.64}$$

$$\Rightarrow (1 + 2\,r)\,\Phi_{i+1}(x) - r\,\Phi_{i+1}(x+1) - r\,\Phi_{i+1}(x-1) = \Phi_i(x) \qquad (4.65)$$

with $r \coloneqq \Delta t/h^2$. This is a tri-diagonal linear equation system which is solved in each step.

While implicit methods are stable, they also generally require more computational effort per time step and more implementation effort than explicit methods. On the other hand, the time step is not restricted as in explicit methods. Also, modification of an existing implementation is much simplified by using an explicit implementation. For these reasons, the experimentation code that was used in the scope of this thesis is using the explicit method. New terms can in this way simply be added and no equation systems need to be assembled.

4.14 Further Extensions and Perspectives

In this section, some directions of recent research that have so far not been mentioned are pointed out. While none of this was actually used in the experiments within this thesis, it should be understood as leads to other interesting topics connected to level set segmentation. It should be pointed out again at this point, that the literature about curve evolution and level set based segmentation is vast and constantly evolving, and not everything can be mentioned here.

4.14.1 Topology Preservation

The idea of preservation of topology during level set evolution has been presented by a few authors, as mentioned further below. This sub-section does not claim to be a concise overview of methods.

Most recently, Le Guyader and Vese [75] proposed an extension of the geodesic active contour model to preserve the topology of an evolving contour. While the freedom to change topology can be one of the big strengths of the implicit level set formulation of image segmentation, this may not in all cases be wanted. If, for example, objects of known topology are to be extracted from images, then this property of the level set method is clearly a drawback. However, one would still like to benefit from the independence of parametrisation that comes with the implicit formulation.

With this motivation, [75] introduces a topology preserving method based on geodesic active contours. This idea of topology preservation was not new at that time, and [75] has been preceded by works from Han et al., Alexandrov et al., Sundaramoorthi et al., and Rochery et al. [86, 2, 52, 53, 126, 125].

The advantage of [75] lies in its simplicity and seamless integration into the implicit framework. Topology preserving extensions to the basic segmentation method can for instance be of interest in the application of tracking, where a known object is to be tracked that is not supposed to change its

topology during tracking. Also, when segmenting certain objects of which the topology is known, separating a curve or merging two curves may be unwanted during curve evolution. The authors of [75] mention some examples, like the segmentation of brain tissue from magnetic resonance images, or the segmentation of multiple cells from microscopic images, where single cells may be very close to each other, so that conventional level set segmentation will surely result in one curve, even if initialised with more than one. So clearly, for these use cases and specifically for semi–automatic segmentation where the user initialises with a few separate curves, topology preservation can be a very nifty feature.
The topology constraint is incorporated in [75] by adding an energy to the segmentation energy $F(\Phi)$, so that the minimisation task is now

$$\min_{\Phi} F(\Phi) + \mu\, E(\Phi)\,. \tag{4.66}$$

The idea is to forge E so that on any two distinct points x, y on the zero level set $C := \{z|\Phi(z) = 0\}$ which are close to each other and about to split or merge C, an additional, repelling potential is added. Assume that Φ is a signed distance function, so that $|\nabla\Phi(x)| = 1$, and that $\Phi(x) < 0$ in the inside of C and $\Phi(x) > 0$ in the outside. In that case, $\nabla\Phi(x)$ is the unit outward normal vector of any level set at point x. In case a split or merge is imminent close to two points $x, y \in C$, the directions of the normals at x, y will be about opposite:

$$\langle \nabla\Phi(x), \nabla\Phi(y)\rangle \approx -1\,.$$

Le Guyader et al. [75] propose the energy

$$E(\Phi) = -\int_\Omega\int_\Omega \Bigg[\exp\left(-\frac{\|x-y\|_2^2}{d^2}\right) \cdot \langle \nabla\Phi(x), \nabla\Phi(y)\rangle \cdot \underbrace{H(\Phi(x)+\ell)\, H(\ell-\Phi(x))}_{h(\Phi(x))} \cdot \underbrace{H(\Phi(y)+\ell)\, H(\ell-\Phi(y))}_{h(\Phi(y))} \Bigg]\, dx\, dy\,. \tag{4.67}$$

The exponential term weights the rest of the energy depending on the distance of points x, y. The terms $h(\Phi(x)), h(\Phi(y))$ effectively restrict the calculation to a band of width ℓ around the zero level set. Finally, when the inner product $\langle \nabla\Phi(x), \nabla\Phi(y)\rangle$ of the two normals at x, y is negative, the energy gets large, while if it is positive, the energy is lower.
Calculation of the first variation of (4.67) is presented in [75] along with numerical methods to compute the gradient descent. Notice that the term (4.67) introduces a global dependence between data, and that the evolution is computationally intensive at first sight. The update for one time step would be of order $O(n^2)$, when n is the number of data points — the evolution

equation contains an integral over the whole image domain. To soften the impact of this drawback, [75] proposes to only calculate the integral locally on a square around a point x, the size of which depends on ℓ. This does make sense since the integral also contains the windowing functions $h(\cdot)$.

4.14.2 Metric Issues

Sundaramoorthi et al. [124] point out that it had previously been observed [88, 141] that all previous work on active contour evolution had implicitly assumed a specific metric when calculating gradient flows; the gradient depends on the used inner product by

$$D_h f = \langle h, \text{grad } f \rangle \,,$$

$D_h f$ denotes the directional derivative of f in the direction h. After defining their shape space M which consists of closed, regular curves, they define a set of inner products on the tangent space $T_c(M)$, $c \in M$. Let $h, k \in T_c(M)$, L the length of the curve c, then the considered inner products from [124] are

$$\langle h, k \rangle_{H^0} := \frac{1}{L} \int_c h(s)\, k(s)\, ds \tag{4.68}$$

$$\langle h, k \rangle_{H^n} := \langle h, k \rangle_{H^0} + \lambda\, L^{2\,n} \left\langle h^{(n)}, k^{(n)} \right\rangle_{H^0} . \tag{4.69}$$

Here, $h^{(n)} = \frac{d^n h(s)}{ds^n}$ denotes the n-th derivative of h. The usually implied inner product (4.68) is called the H^0 inner product. The authors of [124] stress some undesirable features of flows based on H^0 gradients, such as non-smoothness of gradient flows, sensitivity to noise, locality of deformations. Because of this, they consider using higher order inner products (4.69) for active contours. Looking at (4.69), it is stressed in [124] that when increasing λ so that ultimately $\lambda \to \infty$, *translations* are preferred in gradient flows in a natural way. So by choosing λ, one can steer the preference of translations over other deformations during contour evolution — notice this is in contrast to H^0 gradient flows, which as mentioned allow only for local deformations and do not include a global regularity term.

Chapter 5

View Point Tracking Using Shape Sub-Manifolds

5.1 Notation

The lower case ϕ, θ denote spherical coordinates and should not be confused with (Φ, Θ), which denotes an element of the pre-shape space $\mathcal{C}$.

5.2 Introduction

Chapter 3 introduced a space of shapes, or rather pre-shapes with a matching procedure, that will now be put into action. Recall from the introductory Chapter 1 that we would like to investigate tracking of a view point relative to an object, using samples of shapes from different view points only. The considered setting is as follows. Given that we know shapes $c_i \in \mathcal{C}_N$ of one specific object, taken from a number of positions on a view sphere around the object. These may have been extracted from photographs taken from around the object, possibly in a controlled environment, or stem from a model of the object in question. Now, associate a view position $p_i \in \mathbb{S}^2$ with each shape c_i sampled in this way. Given a sequence of images in which the object is moving in 3D, we would then like to approximately track the position of the view point on $\mathbb{S}^2$ relative to the object, based only on the given sampled pairs of shape and position (c_i, p_i).
For extraction and tracking of a silhouette in a sequence of images, the level set method for segmentation can be readily applied. An extension to the basic region based level set method will be used, introducing a prior term stemming from the object model. The level set method is detailed on in Chapter 4.
As it will be needed in what follows, a concept of means on Riemannian manifolds is discussed in the following section.

5.3 Statistics on Shape Manifolds

When working with a shape manifold, we will have to calculate empirical means. Pennec [101, 139] has detailed on means, and also covariances, and a normal law on manifolds — this allows one to even use statistical methods locally on manifold data, such as principal component analysis.
This section describes Karcher means [62] and a gradient descent algorithm to calculate a Karcher mean in practice. Covariance and Gaussian distributions are not needed in what follows, but they may be of interest in further work. For details, the reader is referred to [139]. Note also that these techniques are of course not restricted to shape manifolds, but can be applied to any Riemannian manifold.

5.3.1 Karcher Means

Given a Riemannian manifold M, with a geodesic distance $d(x, y)$ defined for $x, y \in M$.

Definition 5.3.1 *(Variance) Let P be a probability on M.*

$$\sigma^2(y) = E[d^2(y,x)] = \int_M d^2(y,x)\, dP(x)$$

is called variance *with respect to y.*

Definition 5.3.2 *(Fréchet expectation) The set of points*

$$E[X] = \arg\min_{y \in M} E[d^2(y,x)]$$

are called Fréchet expectation*. X is the support of $P(x)$ on M.*

In the following, the notion of geodesic completeness will be used, and is therefore defined here.

Definition 5.3.3 *(Geodesic completeness) A Riemannian manifold M is called* geodesically complete*, if every geodesic γ can be extended to $\mathbb{R}$, i.e. $\gamma : \mathbb{R} \mapsto M$.*

We will later also need the notion of *cut locus* and *geodesic ball* [33]:

Definition 5.3.4 *(Cut locus) Given a geodesically complete Riemannian manifold M, $x \in M$, and $v \in T_x(M)$. If a finite $\mathbb{R} \ni t_0 \geq 0$ exists for which the geodesic $\mathrm{Exp}_x(t\, v)$ is still a geodesic of minimal length for $t \in [0, t_0]$, but not for $t > t_0$, then $\mathrm{Exp}_x(t_0\, v)$ is called* cut point *for x. The set of all cut points for all $v \in T_x(M)$ is called the* cut locus *$C(x)$.*

For instance, the cut locus of a point on the sphere consists of the respective antipodal point.

Definition 5.3.5 *(Geodesic ball) Let $B(y,r) = \{x \in M | d(x,y) < r\}$ so that $B(y,r)$ does not contain the cut locus of the centre y, so that for every point in B there exists a unique, minimal geodesic to y. Then $B(y,r)$ is called* geodesic ball. *$B(y,r)$ is called* regular *if $2 \cdot r \cdot \sqrt{\kappa} < \pi$, for κ the maximal Riemannian curvature in $B(y,r)$.*

Note that the Fréchet expectations are global minima — *Karcher means*, in contrast, are local minima (and thereby are a superset of Fréchet expectations). The original notion from Karcher [62], which is used synonymously with Karcher mean, is *Riemannian centre of mass*. Under certain conditions, the Riemannian centre of mass always exists and is even unique. With p a probability density function for elements of a Riemannian manifold M, these conditions, enumerated in [139], are:

Property 5.3.6

1. *If the support of p is completely inside a regular geodesic ball $B(y,r)$, there exists exactly one Riemannian centre of mass within B.*
2. *If the support of p is completely inside a regular geodesic ball $B(y,r)$ and if $B(y, 2 \cdot r)$ is also geodesic and regular, then $\sigma_x^2(z)$ is convex and has exactly one critical point in $B(y,r)$, which is the Riemannian centre of mass.*

The two were established by Kendall [67] and Karcher [62], respectively. Property 5.3.6 implies that if the neighbourhood on M of which a Karcher mean is calculated is *local enough*, then there is a unique Karcher mean. In practice, this means given a finite number of points on M, a unique empirical mean

$$\tilde{\mu} = \arg\min_{\mu} \frac{1}{N} \sum_{i=1}^{N} d^2(\mu, x_i)$$

exists if the points are close enough in the sense stated above, i.e. if $x_i \in B(\tilde{\mu}, r)$ as above and if $B(\tilde{\mu}, 2 \cdot r)$ is geodesic and regular.

5.3.2 Computing the Mean with Gradient Descent

Assuming a probability $P(x)$ defined on $x \in M$, we seek

$$\mu = \arg\min_{y} \sigma^2(y) = \arg\min_{y} \int_M d^2(y,x)\, dP(x)\,.$$

Notice that the cut locus $C(y)$ of y must be "out of the scope" of $P(x)$, meaning that $P(x) = 0$ whenever x is in the cut locus of y, to ensure differentiability of $\sigma^2(y)$. Then, the gradient of $\sigma^2(y)$ is

$$(\text{grad } \sigma^2)(y) = -2 \int_M \text{Log}_y(x)\, dP(x) = -2 \cdot E[\text{Log}_y(x)]\,. \tag{5.1}$$

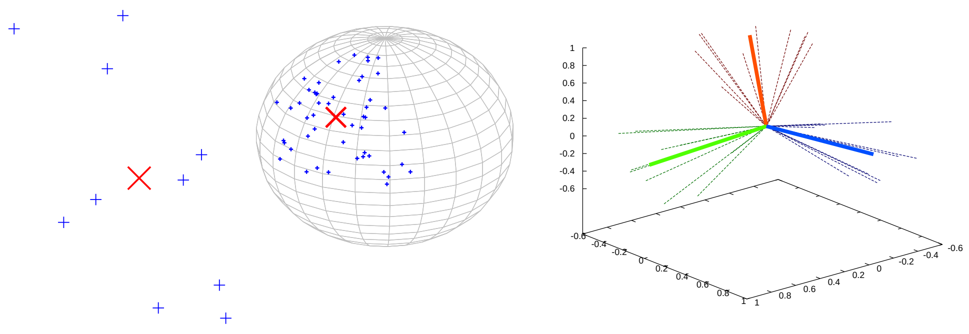


Figure 5.1: Illustrating Karcher means. Left: Karcher mean (red cross) of a set of points in the canonical 2D vector space. Middle: Karcher mean (red cross) of a set of points on the unit sphere $\mathbb{S}^2$. Right: Karcher mean in the space of rotations in 3D. The rotations are illustrated here as tripods representing the rotated standard basis $\{e_1, e_2, e_3\}$ in red, green, and blue, respectively. The input data points are shown as dashed, thin lines, the mean as solid, thicker lines. All means were calculated using exactly the same algorithm using the gradient descent (5.3).

$E[\cdot]$ denotes expectation, and $T_y(M) \ni \mathrm{Log}_y(x) = \mathrm{Exp}_y^{-1}(v)$ if $\mathrm{Exp}_y(v) = x$. Pennec notes in [139] that the variance

$$\sigma^2(y) = \int_{M \setminus C(y)} d^2(y, x)\, dP(x) \tag{5.2}$$

is in general defined only on M excluding the cut locus $C(y)$ which depends on the variable y, and that therefore, it is not immediately clear that just differentiating under the integral is valid. In the same paper, Pennec proves that (5.1) is indeed the gradient of (5.2). Using this result, one can define a gradient descent algorithm to find a *local* Riemannian centre of mass. For an empirical mean, given $x_i \in M$, $i \in \{1, \ldots, N\}$, this leads to the update rule

$$y_{i+1} = \mathrm{Exp}_{y_i}\left(\frac{1}{N}\sum_{j=1}^{N} \mathrm{Log}_{y_i} x_j\right) . \tag{5.3}$$

The exponential map is taking the combination of tangent vectors from $T_{y_i}(M)$ back to M to yield the update $y_{i+1} \in M$. The choice of initial value y_0 is of some importance, since the minimum constituting the mean is local; one can for example set $y_0 \leftarrow x_i$ for some $i \in \{1, \ldots, N\}$.
Notice that Karcher means are very general, and not restricted to specific manifolds, like shape manifolds. In the case of a vector space with the standard inner product, the Karcher mean is just the usual mean. Figure 5.1

shows means for points in $\mathbb{R}^2$ with the canonical inner product, for the unit sphere, and for the group of rotations $\mathcal{SO}_3$. For all three, the exact same implementation of Karcher means was used, only with different exponential and logarithmic maps, and inner products. Definitions for a covariance on Riemannian manifolds are also available; they are not used within this work, so the reader is referred for example to [139].

5.4 Object Appearance

5.4.1 Sampling a Pre-Shape Sub-Manifold

Let us assume to be given a set of contours of a fixed object collected from different viewpoints (see Figure 5.2), and regard these contours as samples of a sub-manifold $\mathcal{C}_{S,N}$ of $\mathcal{C}_N$, that we assume parameterised by the view-sphere $\mathbb{S}^2$.

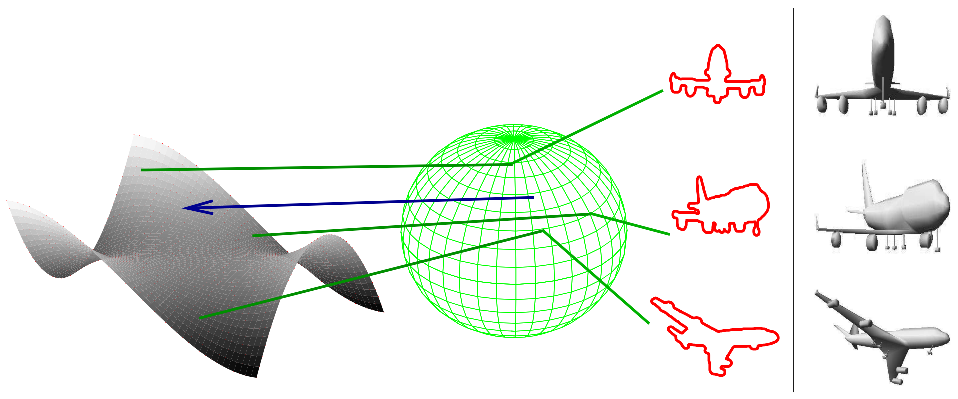


Figure 5.2: Illustration of a view sphere. Right hand: Indicated are three sampled contours of an aeroplane seen through a camera from points on the view sphere. The object is located at the centre of the sphere. Left hand: Illustration of the shape sub-manifold. The green lines between sphere and manifold indicate corresponding points, the blue arrow indicates a point that is interpolated using, in this case, three points which are neighbours on the sphere. This specific object was taken from the Princeton 3D shape benchmark [103].

Specifically, we take sample points from a unit sphere and approximate contours using these samples. It should be noted that these approximations can be expected to deviate from the true points on the sub-manifold $\mathcal{C}_{S,N}$. This deviation will likely depend on the local properties of the pre-shape manifold and the density of the sample points. Furthermore, we conjecture it may be helpful to have the view sphere sampling depend on the local properties of

$\mathcal{C}_N$ to lower the deviation from $\mathcal{C}_{S,N}$. This line of thought is discussed a little further in Section 6.1.

5.4.2 Interpolation with Weighted Karcher Means

Recalling that we do not really know the sub-manifold containing all shapes generated by all possible views of an object, but are given only a finite number of shape samples $C = \{c_1, \ldots, c_n\}$ at known associated view sphere positions, we need to be able to approximate shapes corresponding to an arbitrary view point $p \in \mathbb{S}^2$. This is done by *locally* interpolating using the shape samples C. Locally means that one finds a small neighbourhood M of samples from C so that the corresponding view coordinates are close to p. Using this neighbourhood, a weighted Karcher mean

$$\mu = \arg\min_{m \in \mathcal{C}} \sum_{i=1}^{|M|} a_i \cdot d(m, c_i)^2 \tag{5.4}$$

can be used as an approximation of the shape in $\mathcal{C}_{S,N}$ corresponding to the view sphere position p. The weights a_i are of course depending on the point p.
One may argue that this would not make a lot of sense in general since, given contours of different views of the same 3D object,

1. corresponding points on the contour may be invisible due to self-occlusion, and

2. the path found on $\mathcal{C}_N$ may not at all correspond to the change that the contour undergoes when a camera moves around the object on a view sphere.

Given that the view sphere is sampled densely enough, our hope is that effects like these are sufficiently small, and specifically considering point 2 it can be said from experiments that the interpolated changes are indeed close to the real change in object silhouette, depending on how much change is going on at the respective position on the view sphere.
Figures 5.6 and 5.7 give an idea of the interpolation quality at a few chosen locations for one object. The locations were deliberately chosen to show useful and less useful interpolations.

The quality of interpolation also depends on the complexity of the object in question — a simple cube will exhibit similar contours within a wider range of viewing angle than a more complex object, say an office chair, the shape of which may change more dramatically in the same range of angle. This leads to the conclusion that more complex objects need to be sampled more densely.
After these considerations, the interpolation will now be described. Please keep in mind that here, (ϕ, θ) in *lower case* greek letters denote a point on the

sphere $\mathbb{S}^2$, not an element of the space of vector pairs $\mathcal{H}_N$ which is denoted in *upper case* greek letters (Φ, Θ).

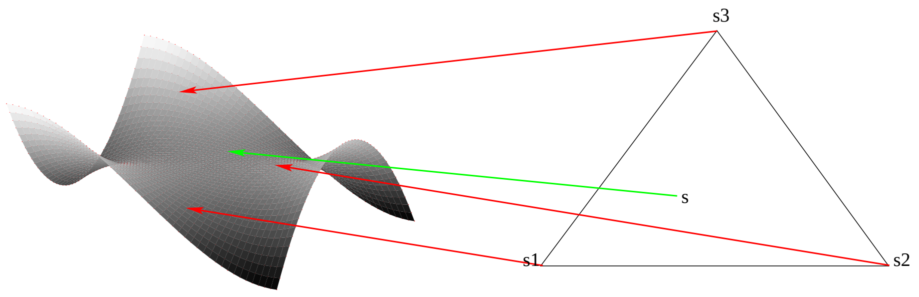


Figure 5.3: Triangle of shapes or sphere points, respectively. Using the shapes at $\{s_1, s_2, s_3\}$, a shape at s is to be interpolated. On the left hand, the (shape) manifold is illustrated.

Let $c_i = (\Phi_i, \Theta_i) \in \mathcal{C}_N$, $i = \{1, 2, 3\}$ be three shapes corresponding to three points $s_i = (\phi_i, \theta_i) \in \mathbb{S}^2$ on the unit sphere which are the closest known points to a point $s = (\phi, \theta) \in \mathbb{S}^2$ as illustrated in Figure 5.3. In order to interpolate, we calculate a Riemannian centre of mass (cf. Section 5.3) by minimising the variance using gradient descent with the update rule

$$p_{k+1} = \mathrm{Exp}_{p_k}\left(\sum_{i=1}^{3} a_i \cdot \mathrm{Log}_{p_k}(c_i)\right) , \quad \sum_{i=1}^{3} a_i = 1 . \tag{5.5}$$

Let

$$p_0 \coloneqq c_1 .$$

Then, compute the convex sum, using the barycentric coordinates a of (ϕ, θ) in the triangle on $\mathbb{S}^2$, on the tangent space and go back to $\mathcal{C}_N$ using the exponential map. Finally, update p_0 with this point and repeat until convergence. The steps are summarised in Algorithm 6. Illustrations can be found in Figures 5.4 and 5.5. The three shapes shown in the triangle corners were chosen randomly from shape databases, and others within the triangle are interpolated as described before.
This is of course not restricted to 3 neighbours, but can be done with any number of neighbours. The neighbourhood should not be too large in order to match only shapes against each other which can be matched in a meaningful way — otherwise, one would have to expect interpolations which do not resemble an actual view of the object.

5.4.3 Approximation by Kernel Regression Estimator

Beside the use of barycentric coordinates in a triangle as described above, a further possibility for obtaining weights for weighted Karcher means is

Algorithm 6 Geodesic interpolation in a triangle of points sampled from a view sphere. MATCH is the matching procedure described in Section 3.2.5, returning the matched curve.

```
Require: $c_i \in \mathcal{C}_N,\ s_i, s \in \mathbb{S}^2\,,\quad i \in \{1,2,3\}$
 1: procedure KARCHERINTERPOLATION($\{c_1,c_2,c_3\}, \{s_1,s_2,s_3\}, s$)
 2:     Find start points and orientation of $\{c_2,c_3\}$ w.r.t. $c_1$    ▷ or w.r.t. another fixed curve
 3:     $\{e,e_1,e_2,e_3\} \leftarrow$ The 3D coordinates of $\{s,s_1,s_2,s_3\}$ on the unit sphere
 4:     $\{a_1,a_2,a_3\} \leftarrow$ The barycentric coordinates of $e$ w.r.t. $\{e_1,e_2,e_3\}$ ($e$ is for this purpose projected onto the plane spanned by $\{e2-e1, e3-e1\}$)
 5:     $j = \arg\max_i a_i$
 6:     $p_0 \leftarrow c_j$    ▷ Or an initialisation from a previous result
 7:     repeat
 8:         $\{c'_1,c'_2,c'_3\} \leftarrow$ MATCH($p_0, \{c_1,c_2,c_3\}$)
 9:         $\{t_1,t_2,t_3\} \leftarrow \mathrm{Log}_{p_0}(\{c'_1,c'_2,c'_3\})$
10:         $t \leftarrow \sum_{i=1}^{3} a_i \cdot t_i$
11:         $p_0 \leftarrow \mathrm{Exp}_{p_0}(t)$
12:     until Convergence
13: end procedure
```

to use a kernel regression estimation [7] motivated method, which has been proposed by Davis et al. in [31] in order to do regression on general manifold-valued data and on brain MRT[1] images in particular. They provide an estimator using the Karcher expectation with kernel weights. Where the usual (set of) weighted Karcher expectation(s) on a Riemannian manifold M with geodesic distance $d(\cdot,\cdot)$ is

$$\mu = \arg\min_{m\in M} \sum_{i=1}^{n} a_i\, d(m,p_i)^2\,,$$

it now becomes

$$\widehat{m}_h(t) = \arg\min_{m\in M} \left(\frac{\sum_{i=1}^{n} \left[K_h(t-t_i)\cdot d(m,p_i)^2\right]}{\sum_{i=1}^{n} K_h(t-t_i)} \right)\,. \tag{5.6}$$

The kernel functions $K_h(t)$ with width parameter h are in [31] taken to be Gaussians, where the parameter space is a one dimensional time interval.
In our case, the manifold M is the closed pre-shape space $\mathcal{C}_N$, and the parameter space is the view sphere $\mathbb{S}^2$. So, $m, p_i \in \mathcal{C}_N$ and $t, t_i \in \mathbb{S}^2\,,\ i \in \{1,\ldots,n\}$. While one can in theory use all samples to calculate the weights in (5.6), it is of course not advisable to do so and instead to constrain the neighbourhood of relevant samples by choosing the kernel width sensibly, since

[1]Magnetic resonance tomography.

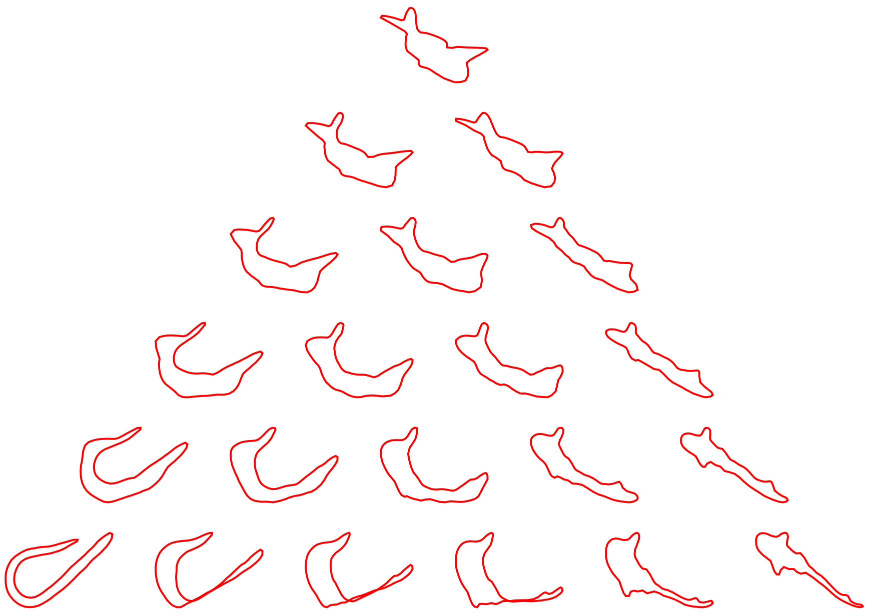


Figure 5.4: Experiment illustrating the interpolation in the triangle. The three corners show the three given shapes $\{c_1, c_2, c_3\}$, the other contours are interpolated using Algorithm 6. The triangle is sampled in barycentric coordinates at points $p_i = (0.2, 0.2, 0.2) \cdot (k, l, m)$, $\mathbb{N} \ni k, l, m \geq 0$, $\sum_j p_{i,j} = 1$, with "$\cdot$" denoting point-wise product. These contour curves were taken from the popular Surrey fish data base [94].

1. the distance $d(m, p_i)$ could generally *decrease* with increasing distance on the sphere, since an object may look similar from different angles.
2. as has already been mentioned before, the expectation (5.6) may not be meaningful otherwise, because we would be unable to match two contours so that naturally corresponding point pairs are associated — because naturally corresponding point pairs would likely not be visible if two contours are sampled from points too far apart on $\mathbb{S}^2$.

A number of neighbouring sampled shapes can be selected for consideration prior to calculating $\widehat{m}_h(t)$ in (5.6).
While this was implemented and tried, it is not further used in experiments, since a noteworthy difference in results compared to using three neighbouring shapes with barycentric coordinates could not be seen.

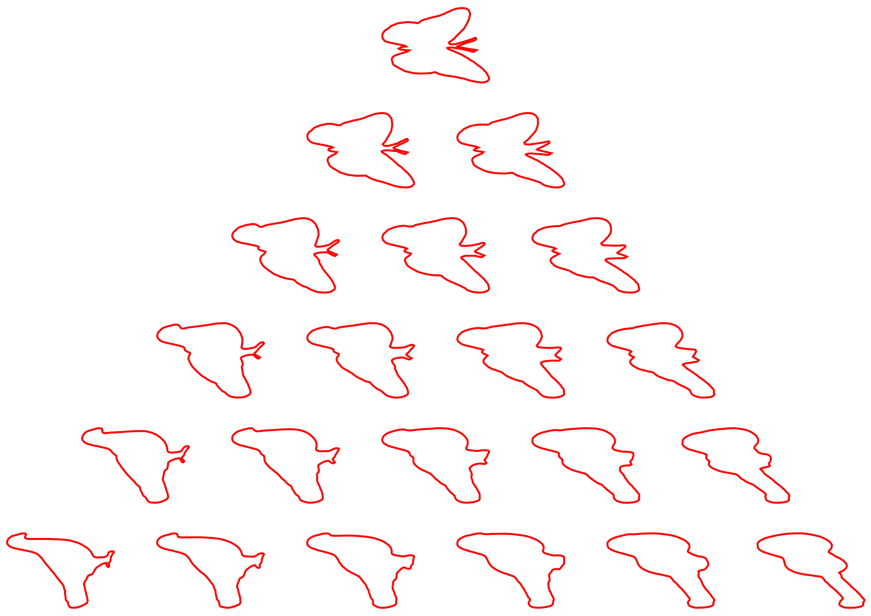


Figure 5.5: Another interpolation experiment, with different curves. See also Figure 5.4. These curves are randomly chosen from the MPEG-7-CE1 shape data base.

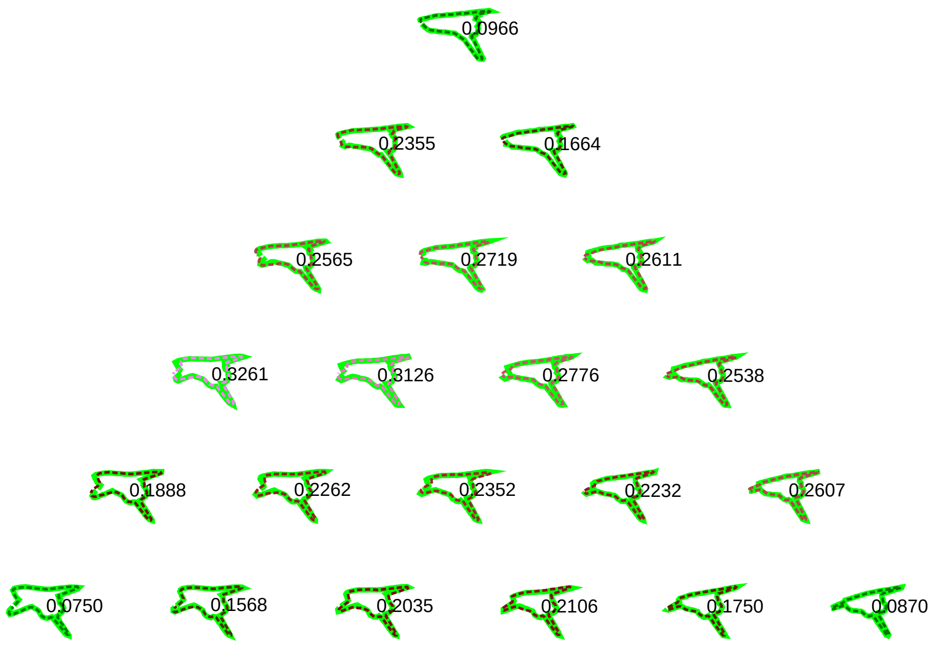


Figure 5.6: Area on the view sphere where there is acceptable agreement between sampled and interpolated shapes.The green shapes are sampled views of a 3D object model, the other shapes are interpolated at the corresponding spherical positions using the three shapes at the corners and barycentric weights. The colour coding indicates where the geodesic distances are lowest (green) and highest (violet). The numbers are the respective geodesic distances.

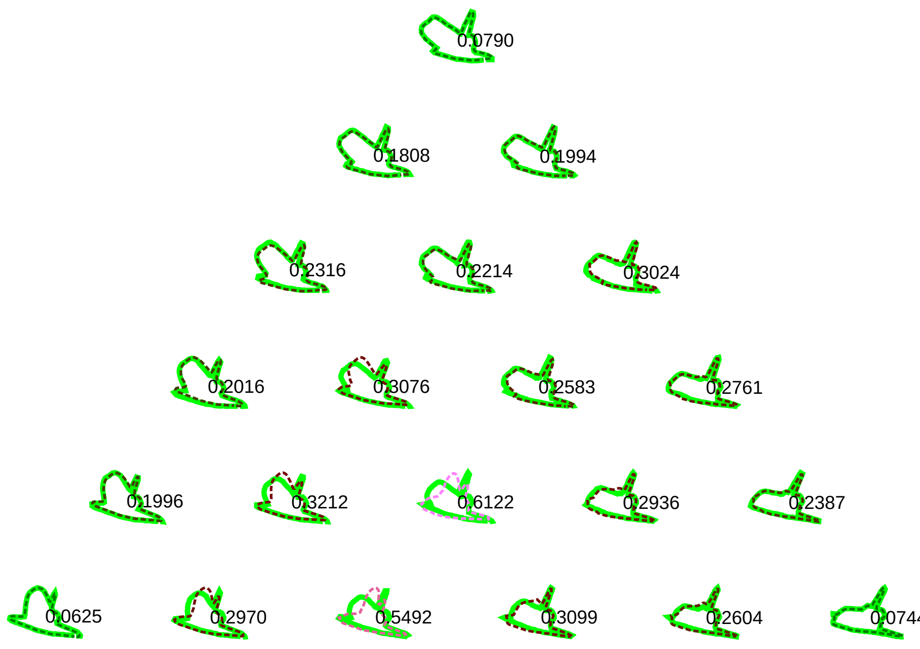


Figure 5.7: Area on the view sphere where there is much change and partially bad agreement between sampled and interpolated shapes.The green shapes are sampled views of a 3D object model, the other shapes are interpolated at the corresponding spherical positions using the three shapes at the corners and barycentric weights. The colour coding indicates where the geodesic distances are lowest (green) and highest (violet). The numbers are the respective geodesic distances.

5.4.4 Comment on Submanifold

At this point, let us take a look at the notion of submanifold, which we have been using loosely (in the form "sub-manifold"). We can think of objects which due to symmetry lead to the same shape at two different view points. This means that the subset of shapes defined by the shapes for all possible view points will have self-intersections, so it can not generally be said to be a submanifold in a mathematical sense (see e.g. [33]).

5.5 Mechanical Motion Model

Section 5.6 will introduce a method for obtaining pairs of view sphere points and shapes, $(p, c) \in \mathbb{S}^2 \times \mathcal{C}_N$, given a starting point $(p, c)_0$ and a sequence of updated shapes $q_1, \ldots, q_n \in \mathcal{C}_N$ from which points $p_1, \ldots, p_n \in \mathbb{S}^2$ are inferred. These points are used as potentially noisy measurements for a motion model. This model can be utilised to get a smooth continuous motion from the measurements p_i and to locally predict p into the near future. Such predictions can then be used to influence the contour extraction in the next frame of an image sequence, for instance. Using a mechanical model is a simple solution, which nonetheless appears to work fairly well in experiments. Of course, that depends on the choice of user parameters like friction coefficient and inertia. So as a lookout, one may want to look into Kalman filters or more advanced methods of *model predictive control*, also known as *receding horizon control*, see for example [13]. Simpler trackers like linear quadratic trackers, see for example [80], are out of question because they rely on the availability of data for the complete tracked path prior to computation.
Model predictive control methods, if applicable, would then have to be adapted to the geometry of the sphere.
This section now proceeds with introducing the applied mechanical model.

5.5.1 Mechanics on $T(\mathcal{SO}_3)$ or $T(\mathbb{S}^2)$ with Stokes Friction

We propose to use a motion model that is motivated by a physical model, considering motion under the influence of a potential field, with friction. The following generic treatment is in terms of time t, a position $s(t)$, and forces are generally denoted by F. The potential field will be termed V. Notice that the potential field V does not necessarily have to be something physical; it was only motivated by physics to have a meaning like some sort of attraction.
Stokes friction is a popular approximation to a friction force for motion in viscous liquids [50]:

$$F_R = -\beta \cdot \dot{s}(t)\,.$$

Assuming a potential field of the form

$$V = m \cdot g \cdot (s(t) - P)^2 \,,$$

with m a constant akin to inertia and P a "centre of attraction", the force acting on a mass point in the field will be the negative gradient of the potential. The factor g is a weight for the potential, which one can perhaps think of as being related to the acceleration of gravity when imagining a gravitational field. Thus

$$F_V = -\nabla V = -2 \cdot m \cdot g \cdot (s(t) - P) \,.$$

In order to get a law of motion, we apply Newton's second axiom

$$F = m \cdot \ddot{s}(t) \,,$$

which, when adding the frictional force and the force induced by the potential field then yields

$$F = F_V + F_R = \underbrace{-2 \cdot m \cdot g \cdot (s(t) - P)}_{-\nabla V} - \underbrace{\beta \cdot \dot{s}(t)}_{\text{Friction}} = m \cdot \ddot{s}(t) \,. \tag{5.7}$$

This is an inhomogeneous differential equation of second order, for which boundary conditions

$$\begin{aligned} s(0) &= 0 & (5.8) \\ \dot{s}(0) &= v_0 & (5.9) \end{aligned}$$

are assumed. A solution for (5.7)–(5.9) can be found in Appendix E.3.
Simulating this kind of motion in one and two dimensions yields trajectories as illustrated in Figures 5.8 and 5.9 for several values of the friction coefficient β. For these illustrations, it was taken that $m = 1$. In the one dimensional case, initial velocity $v_0 = 0$ was used, for the two-dimensional case it was set to $v_0 = (-50, -25)^\top$. The centre of the potential field was $P = 10$ and $P = (10, 10)^\top$, respectively.

To describe motion on a sphere or in the space of rotations, we can apply these motion equations locally on the tangent spaces of the unit sphere, or equivalently on the tangent spaces of the group $\mathcal{SO}_3$ of rotations, and take the resulting path back to the unit sphere or the group of rotations by means of the respective exponential map. Whether one chooses to use $\mathcal{SO}_3$ or $\mathbb{S}^2$ to describe the motion is somewhat arbitrary. While calculating on $T(\mathcal{SO}_3)$ is slightly more complicated, it naturally delivers a rotation as position, while on the sphere, the natural result is a point on the sphere. So the choice depends on if one wants to describe the relative camera pose by rotation or by a sphere point.
In the following, the calculations for $\mathcal{SO}_3$ are detailed. For the sphere $\mathbb{S}^2$,

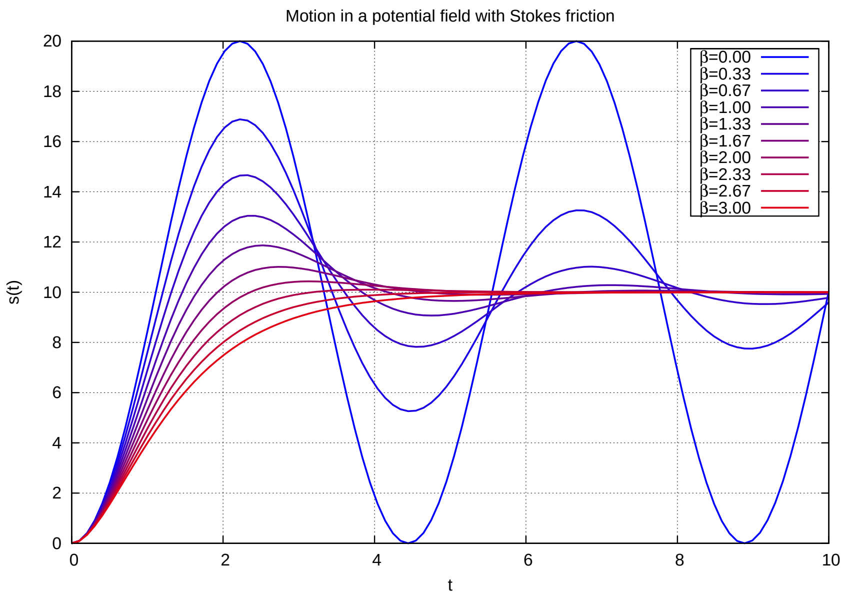


Figure 5.8: 1D motion simulated with the equations and values described in Section 5.5.1. See Figure 5.9 for a 2D example.

the calculations are a little simpler and can be done using the formulas given in Appendix E.2. Details on exponential and inverse exponential maps on $\mathcal{SO}_3$ can be found in Appendix E.1.

Very generally, the motion in the potential field can be calculated in the tangent spaces of a smooth manifold M as follows. Let $\sigma_0 \in M$ be a start position, $B(\sigma_0, r) \ni P \in M$ a centre of gravitation (p in Equation (5.7)). $B(\cdot,\cdot)$ is a geodesic ball as in Definition 5.3.5. One then uses the tangent space representation of the centre of attraction P, calculates the motion in the linear tangent space, and goes back to M by means of the exponential map:

$$
\begin{aligned}
p &:= \operatorname{Log}_{\sigma_0}(P) && (5.10)\\
\sigma(t) &:= \operatorname{Exp}_{\sigma_0}(s(t)) && (5.11)\\
\dot{\sigma}(t) &:= \dot{s}(t)\,. && (5.12)
\end{aligned}
$$

One should carefully note that $\dot{\sigma}(t)$ is in $T_{\sigma_0}(M)$, so that it needs to be parallel transported if it is to be used at, say, $\sigma(t)$.

Using the space of rotations, the motion model can be used to model the position of a point on the sphere, and also an equivalent rotation, like this:

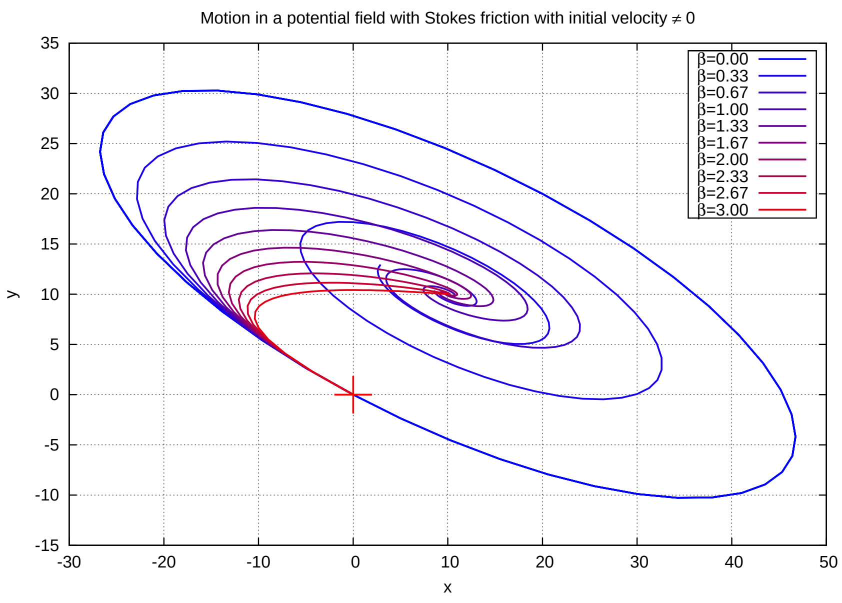


Figure 5.9: 2D motion simulated with the equations and values described in Section 5.5.1. The red cross marks the start point of motion at $(0, 0)$. See Figure 5.8 for a 1D example.

Let

$$\begin{aligned} v_0 &\leftarrow (0,0,0)^\top \\ s_0 &\leftarrow (0,0,0)^\top \\ R_0 &\leftarrow I \\ t_0 &\leftarrow 0\,, \end{aligned}$$

and set $p_0 \in \mathbb{R}^3$ to the fixed start position. Say at time $t = t_k$, $k \in \mathbb{N} \geq 1$, a new measurement, represented by a centre of attraction p_k, comes to our knowledge. Then

$$R(t) = \mathrm{Exp}_I(s(t)) \cdot R_0 \tag{5.13}$$

is used to calculate a rotation $R(t)$ using the origin R_0 and a "relative" rotation given by $s(t)$.

$$T_{R_0}(\mathcal{SO}_3) \ni v(t) = \dot{s}(t) \tag{5.14}$$

is the velocity along the path $s(t)$ on $T_{R_0}(\mathcal{SO}_3)$.

$$v_0 \leftarrow v(t_k - t_{k-1}) \tag{5.15}$$

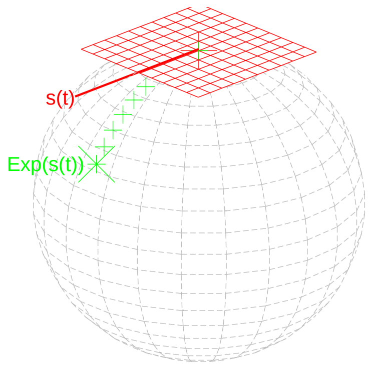


Figure 5.10: Illustrating motion on $\mathcal{SO}_3$, see Equation (5.13). The motion $s(t)$ is applied in the tangent space at identity, the result is pulled back to the space of rotations by the exponential map. To apply the motion not at identity, but at R_0, the result is then applied from the left to R_0.

then becomes the new initial velocity, used in the equations given in appendix E.3, and

$$R_0 \leftarrow R(t_k - t_{k-1}) \tag{5.16}$$

becomes the new origin. Then a rotation axis a can be computed by

$$a \leftarrow \frac{(R_0 \cdot p_0) \times p_k}{\|(R_0 \cdot p_0) \times p_k\|} \tag{5.17}$$

and the angle of rotation θ from the initial point $R_0\, p_0$ on the sphere to the new p_k is set to

$$\theta \leftarrow \arccos(p_k^\top \cdot R_0 \cdot p_0). \tag{5.18}$$

So the "centre of attraction" $P \in T_{R_0}(\mathcal{SO}_3)$ is

$$P \leftarrow a \cdot \theta \tag{5.19}$$

and the path on $\mathbb{S}^2$ can, until the next point p_{k+1} comes up, be calculated with

$$p(t) := R(t - t_k) \cdot p_0\,. \tag{5.20}$$

The functions $s(t)$ and $\dot{s}(t)$ are given by the equations (E.7)–(E.12), whichever apply. Note that it should always hold that the absolute angle of rotation applied by $R(t)$ from the current R_0 does not exceed π so that one never reaches the cut locus, where the resulting rotation would not be unique anymore — that means $R(t)$ must remain in a geodesic ball around R_0. Pictorially speaking, $R(t)$ being on the cut locus would yield a reflection in $(0, 0, 0)^\top$.
Figure 5.11 provides an illustration of what is happening on $\mathcal{SO}_3$: shown are two tripods representing orthonormal coordinate systems. One is rotated to

the other following the force generated by the potential field V. The first plot was generated without an initial velocity v_0, the other with an initial velocity v_0. The tripod drawn in thinner lines is rotated into the tripod with thicker lines, that means the latter is P in the equations above, while the former is rotated into identity and therefore represents the starting point, which we set to 0 in all equations. The curved, dashed lines illustrate the path that the end points of the tripod took along its journey towards P under the influence of V and a friction-like force.

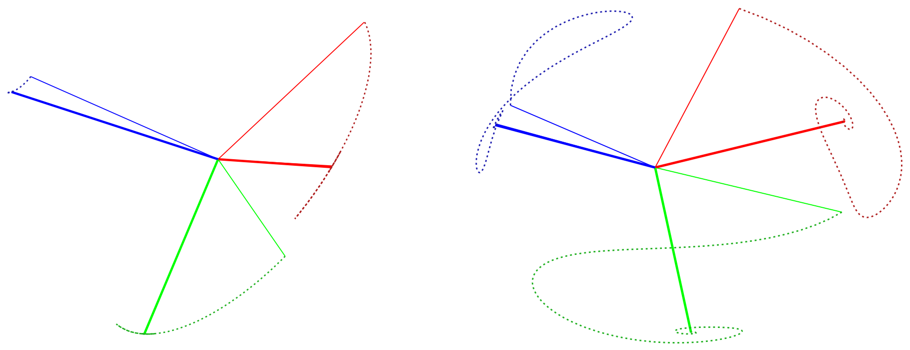


Figure 5.11: Application of the motion model to rotations, as described in Section 5.5.1. Left: $v_0 = 0$. Right: $v_0 \neq 0$. The start and target rotations are represented by the rotated tripod of standard basis vectors $\{e_1, e_2, e_3\}$ drawn in red, green, and blue, respectively. The tripod drawn in thin lines is the start rotation, the one in thick lines the target rotation. The dashed lines indicate the paths taken by the end points of the basis vectors during motion under a force field with friction.

5.5.2 Predictions

Given past measurements $p_i \in \mathbb{S}^2$, we would like to predict $s(t)$ locally. Assume to be given a new measurement P_k at time t_k, and the motion model to be at point $s(t_k)$. We then follow the trajectory governed by (5.7) until the distance $d(s(t_k), P_k)$ has been travelled, say at time t'_k, so that $d(s(t_k), s(t'_k)) = d(s(t_k), P_k)$, and then continue for an additional fixed time period $\Delta t = t'_k - t_k$ to obtain the prediction

$$p_{pred} \coloneqq s(t'_k + \Delta t)\,. \tag{5.21}$$

As illustrated in Figure 5.12, this simple "mechanical" model can result in rather sensible paths. The model can result in bad paths if one chooses bad values for the parameters m, g, β; an example for badly chosen parameters can be seen in Figure 5.13. However, it is not hard to find sensible values experimentally.

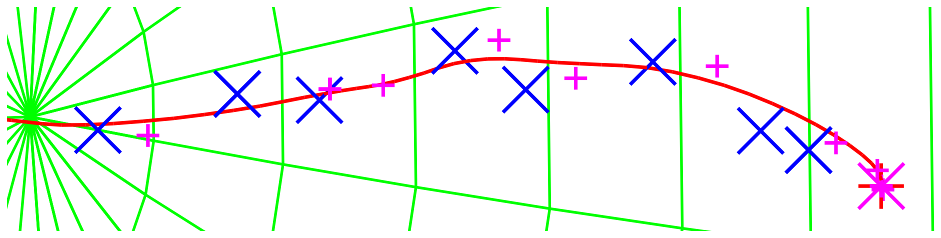


Figure 5.12: Representing and tracking shape changes as motion on the view sphere. Blue: measurements p_k. Red: path $s(t)$ of the mass point. Magenta: predicted points. The start point of the trajectory is at the far left end. The green grid lines indicate the underlying sphere. The parameters are here $m = 0.2, g = 0.5, \beta = 0.1$. The same illustration with badly chosen motion parameters can be seen in Figure 5.13.

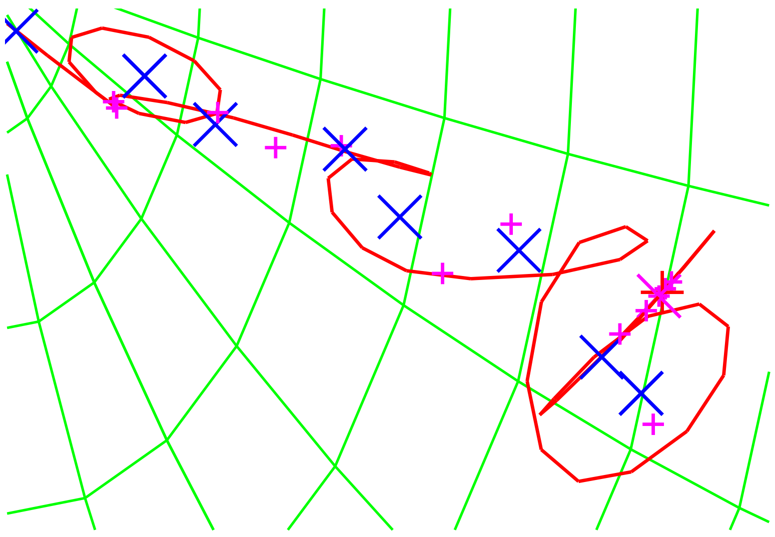


Figure 5.13: Simple motion model with badly chosen parameters $m = 2, g = 5, \beta = 0.1$.

5.6 Change in View Point

In the remainder of this chapter, most of the techniques and concepts discussed so far are integrated into a method for tracking a view point based on object outline. This section in particular deals with the task of keeping track of the view point relative to an object, given an initial view point position and a sequence of shapes capturing changes in view point.
The object is assumed to undergo some smooth rigid transformation in 3D space. For instance, imagine an aeroplane flying, turning, and thereby changing its silhouette, or a car driving along a road and maybe changing direction at a crossroads.
The contour extraction from the images is here achieved by using either the level set method or the closely related global method [17] sketched in Appendix B. In the following, a newly developed tracking mechanism for the spherical view position is introduced.

5.6.1 Problem Statement

Assume that we know initially a point $c_k \in \mathcal{C}$ and the corresponding position $t_k \in \mathbb{S}^2$. Now, suppose a new shape $q \in \mathcal{C}$ is to be considered, typically delivered by an image segmentation algorithm that is used to track an object over a number of frames. Figure 5.14 illustrates the problem at hand when we want to update the view position t_k: We wish to determine a point $c_{k+1} \in \mathcal{C}$, corresponding to $t_{k+1} \in \mathbb{S}^2$, on the sub-manifold modelled by the samples $p_i \in \mathcal{C}$ from the view sphere at spherical coordinates $t_i \in \mathbb{S}^2$, which is as close as possible to q. That is, we would like to minimise the geodesic distance $d(m, q) = \|\mathrm{Log}_m(q)\|_m$ by minimising

$$F(m, q) = \|\mathrm{Log}_m(q)\|_m^2 \,, \tag{5.22}$$

where m results from solving (5.4) (or (5.6)),

$$m(t) = \arg\min_{\tilde{m} \in \mathcal{C}} \left(\sum_{i=1}^{|M|} a_i(t) \cdot d(\tilde{m}, p_i)^2 \right) \tag{5.23}$$

with both the neighbourhood M and the weights a_i depending on the spherical position t.

5.6.2 Solution

We then solve at frame $k + 1$

$$t_{k+1}^\star = \arg\min_t F(m(t), q) \tag{5.24}$$

using non-linear conjugate gradient descent on the view sphere, as follows: Choose $b_{\ell,1}, b_{\ell,2} \in \mathbb{R}^3$ to be orthonormal basis vectors of the tangent space

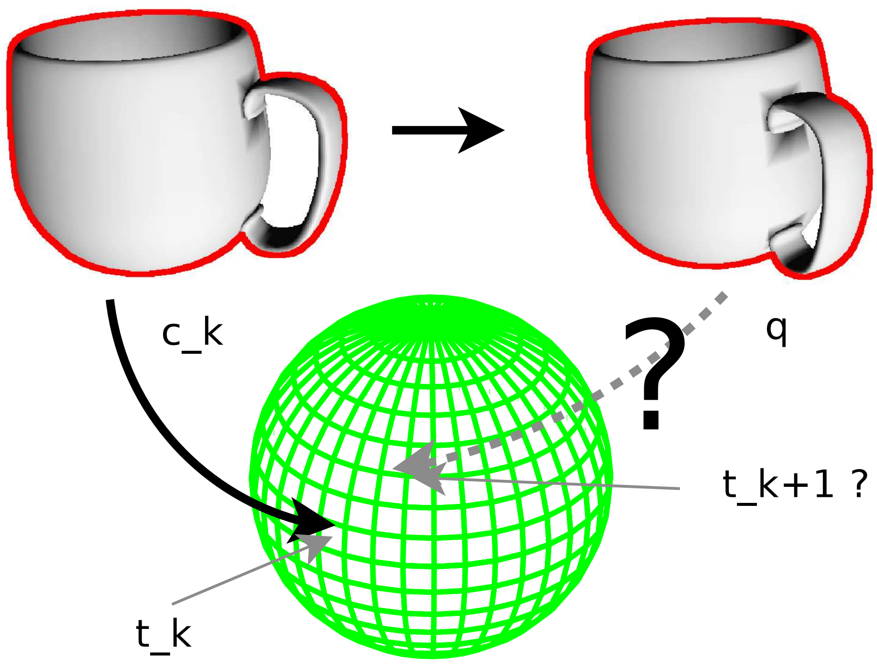


Figure 5.14: Keeping track of the spherical position: Shape c_k and position t_k are known, as well as a new shape q. What is the (approximate) position t_{k+1} on the view sphere corresponding to q?

$T_{t^\ell}(\mathbb{S}^2)$, and a small constant $\Delta > 0$. Notice that in the following equations, Exp and Log denote the exponential and inverse exponential maps on the sphere $\mathbb{S}^2$, not on the pre-shape space $\mathcal{C}$.

$$\text{trans} : T(\mathbb{S}^2) \times \mathbb{S}^2 \times \mathbb{S}^2 \mapsto T(\mathbb{S}^2), \quad v_2 = \text{trans}(v_1, t_1, t_2) \tag{5.25}$$

is a function that takes a tangent vector $v_1 \in T_{t_1}(\mathcal{C}_N)$ and translates it along a geodesic from t_1 to t_2, resulting in $v_2 \in T_{t_2}(\mathcal{C}_N)$. Formulas for calculating exponential, inverse exponential, and parallel transport on a sphere are given in Appendix E.2. Then, let

$$t^0 = t_k^\star\,, \quad \beta_{-1} = 0\,, \quad \tilde{d}_{-1} = 0\,,$$

and

$$\begin{aligned}
v_\ell &= \sum_{i=1}^{2} b_{\ell,i} \cdot \frac{F(m(\mathrm{Exp}_{t^\ell}(\Delta \cdot b_{\ell,i})), q) - F(m(t^\ell), q)}{\Delta} && (5.26)\\
d_\ell &= -v_\ell + \beta_{\ell-1}\tilde{d}_{\ell-1} && (5.27)\\
t^{\ell+1} &= \mathrm{Exp}_{t^\ell}(\alpha \cdot d_\ell) && (5.28)\\
\tilde{d}_\ell &= \mathrm{trans}(d_\ell, t^\ell, t^{\ell+1}) && (5.29)\\
\tilde{v}_\ell &= \mathrm{trans}(v_\ell, t^\ell, t^{\ell+1}) && (5.30)\\
\beta_\ell &= \frac{[v_{\ell+1} - \tilde{v}_\ell]^\top v_{\ell+1}}{v_\ell^\top v_\ell}\,. && (5.31)
\end{aligned}$$

Here, v_ℓ takes the role of the gradient direction, in the tangent space of $\mathbb{S}^2$ at the current point t^ℓ. From the gradient v_ℓ and the previous search direction $\tilde{d}_{\ell-1}$, the search direction d_ℓ is computed. The factor $\beta_{\ell-1}$ is calculated using the *Polak-Ribière* variant of the conjugate gradient method in Equation (5.31). The latter is according to [96] more robust than the alternative *Fletcher-Reeves* variant.
The rest of the above equations are needed to adapt to the geometry of the sphere. Specifically, we have to translate tangent vectors to the next iterate $t^{\ell+1}$, so that all involved tangents are elements of the same tangent space and can be combined (equations (5.29) and (5.30)). Since calculations take place in the tangent spaces, we also need to go back to the sphere using the exponential map in Equation (5.28).
In order to find a step length $\mathbb{R} \ni \alpha > 0$ for use in Equation (5.28), we use a standard line search procedure with the Armijo or *sufficient decrease* condition

$$F(m(\mathrm{Exp}_{t^\ell}(\alpha \cdot d_\ell)), q) \leq F(m(t^\ell), q) + c \cdot \alpha \cdot (v_\ell^\top d_\ell)\,, \quad 0 < c < 1 \qquad (5.32)$$

and choose the step length α according to Algorithm 7.

Algorithm 7 Back-tracking line search: step size selection for gradient descent using the Armijo condition.

```
procedure ArmijoStepLength
    Choose a fixed $\alpha_0$
    $\alpha \leftarrow \alpha_0$
    Choose $0 < \tau < 1$
    while $\alpha$ does not meet Armijo condition do
        $\alpha \leftarrow \alpha \cdot \tau$
    end while
    return $\alpha$
end procedure
```

Figure 5.15 shows the result of an experiment using the method proposed in this section. The green contours (left column) show shapes which were obtained from the same 3D object model that was used to create the samples modelling the view sphere. The red contours (right column) show the results from applying the view point tracking method from this section to each new frame. A frame is, in this experiment, only the respective contour, so curve extraction is so far not needed. Depicted on the right hand side of Figure 5.15 are the points on the view sphere corresponding to the input (green, left curve) and the result from the view point tracking (red, right curve). Both the resulting shapes and the curve on the view sphere suggest that the method works reasonably well here.

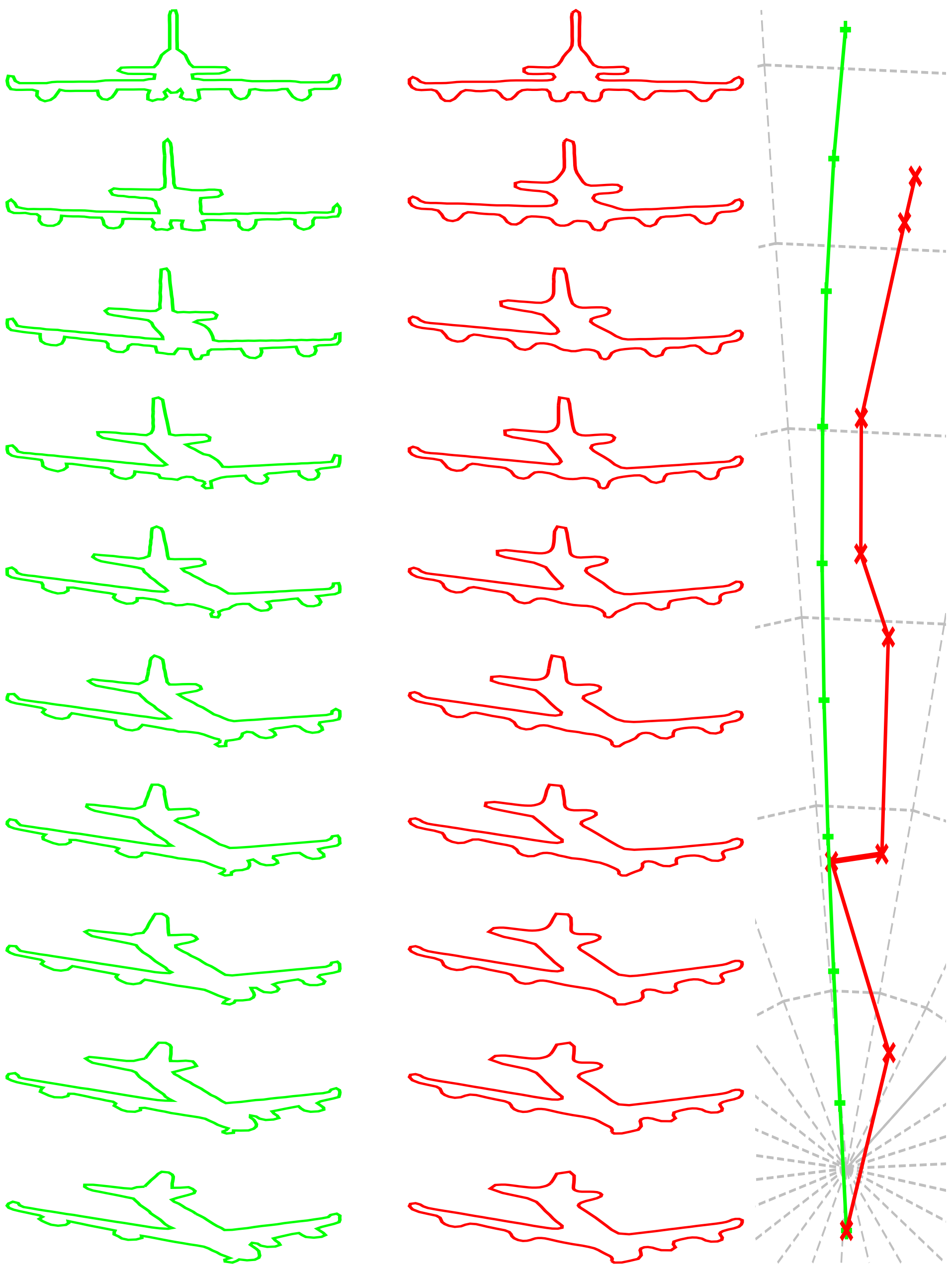


Figure 5.15: Estimating shapes on a sample path from the view sphere. Green (left): Input shapes sampled from the view sphere. Red (right): Estimated shapes using the methods from Section 5.6. On the left are the contours which correspond to the paths on the sphere shown on the right. The grid lines on the right indicate the sphere.

5.6.3 Convergence

Finally, note that the above application of conjugate gradient has been described in [1] (see also references therein). The treatment is quite general and covers any method utilising a sequence of gradient related[2] tangent vectors to minimise a function $F : M \supseteq U \mapsto \mathbb{R}$ on a manifold M. The general line search algorithm assumes a Riemannian manifold M, a *retraction* (see Definition 5.6.2 below) on M, and a *continuously differentiable* scalar field F on M. The latter is the issue in our context — it is not guaranteed that F from (5.22) is continuously differentiable everywhere. One can expect this to depend on the object in question, by the following argument: Imagine for example the object depicted in Figure 6.1 in the next chapter. Due to self-occlusion, there are several situations where the outline changes abruptly, so F can not be expected to be continuous at these views. This may also motivate the search for representations allowing for inner contours to be retained.

Keeping this in mind, for functions to be minimised which are continuously differentiable, the following theorem on convergence is stated in [1] (theorem 4.3.1):

Theorem 5.6.1 *If $\{x_k\}$ is an infinite sequence generated by Algorithm 8, stated below, then every accumulation point of $\{x_k\}$ is a critical point of the cost function F.*

For more details, see [1]. For completeness, the outline of the general gradient-based search method is given in Algorithm 8.
A retraction is defined in [1] as

Definition 5.6.2 (Retraction) *Given a manifold M. A smooth mapping $R : T(M) \mapsto M$ with R_x the restriction of R to $T_x(M)$, for which hold*

1. *$R_x(0_x) = x$ with 0_x denoting the zero element (origin) of $T_x(M)$*
2. *Identifying $T_{0_x}(T_x(M)) \simeq T_x(M)$, R_x satisfies*

$$DR_x(0_x) = id_{T_x(M)}$$

   with $id_{T_x(M)}$ denoting the identity mapping on $T_x(M)$

is called retraction *on M.*

[2]A sequence $\{x_k, \eta_k\}$ of points x_k on a manifold M and corresponding tangent vectors $\eta_k \in T_{x_k}(M)$ is called *gradient related* if for any subsequence $\{x_k\}_{k \in I}$ converging to a non-critical point of a function F on M, it holds that $\limsup_{k \to \infty,\, k \in I} \langle \text{grad } f(x_k), \eta_k \rangle < 0$ [1].

Algorithm 8 General gradient based search method from [1].

```
Require: Riemannian manifold $M$, continuously differentiable scalar field
    $F$ on $M$, retraction $R$ on $M$, initialisation $x_0 \in M$, $0 < c < 1$
1: function GeneralLineSearch($x_0$)
2:     $k \leftarrow 0$
3:     while Stopping condition not met do
4:         Choose some $\eta_k \in T_{x_k}(M)$ so that the sequence $\{\eta_i\}$ is gradient
    related
5:         Let $x_{k+1}$ the next iterate satisfying

               $F(x_k) - F(x_{k+1}) \geq c\,[F(x_k) - F(R_{x_k}(\alpha_k\,\eta_k))]$

    for $\alpha_k$ obtained using line search with the Armijo condition
6:     end while
7:     return The sequence $\{x_i\}$
8: end function
```

5.7 Integrating with Segmentation

To put the object tracking to work, we first consider integrating the estimated shape from the methods described in this chapter into a level set segmentation algorithm, details of which can be found in Chapter 4. That method has been used for tracking deforming contours in image sequences in [93]; the very simple idea is to use the outcome of one frame as initialisation for segmenting the next.

The variant used for our experiments uses the vector-valued image energy from Chan and Vese from Equation (4.20) with the additional regularisation term (4.35). To constrain the level-set evolution to remain close to the object shape during tracking, a term accounting for a shape prior is added which was introduced in Chapter 4 in Equation (4.40). This prior is almost identical to the one proposed by Riklin-Raviv et al. [106], but allowing only for euclidean and not for perspective transformations.

5.7.1 Combination with Level Set Image Segmentation

Given a sequence of images g_i, $i = 1, \dots, n$ and an initial embedding function Φ_1 and spherical position $(\phi_1, \theta_1) \in \mathbb{S}^2$, we would like to apply the methods for tracking on the view sphere from the previous sections for subsequent frames. That means, we would like to find the movement of a hypothetical camera on the view sphere, or equivalently the pose change of the object in subsequent frames.

Given the initialisation, the task is then to combine the *sphere tracking* methods with ways to keep track of the object while it moves *in the image plane*. Our approach then consists of the steps summarised in Algorithm 9.

Algorithm 9 View point tracking on the view sphere using an implicit curve representation for segmentation. The current or predicted shape can be used as prior knowledge to help steering the segmentation algorithm.

```
1: Let $\Omega$ be the image domain. Initialise manually
       $$\Phi : \Omega \mapsto \mathbb{R},\ |\nabla\Phi| = 1 \text{ and } (\phi_1, \theta_1) \in \mathbb{S}^2$$
2: for $i = 2$ to $n$ do
3:     Extract the curve $C_{i-1} \leftarrow \{x : \Phi(x) = 0\}$
4:     Set $P_i$ to the curve corresponding to $(\phi_{i-1}, \theta_{i-1}) \in \mathbb{S}^2$ or to a predicted
       curve using a motion model on $\mathbb{S}^2$
5:     Align $P_i$ to $C_{i-1}$ w.r.t. scale, rotation, and translation
6:     Keep track of the view sphere position: Set $(\phi_i, \theta_i)$ to the minimiser
       of Equation (5.22),
       $$(\phi_i, \theta_i) \leftarrow \arg\min_m \|Log_m(q)\|_m^2$$
       with $q \in \mathcal{C}_N$ corresponding to the contour $C_{i-1}$.
7:     Optionally set prior $\tilde{\Phi}_i$ to the signed distance function for which
       $$P_i = \{x : \tilde{\Phi}_i(x) = 0\}$$
8:     Evolve $\Phi$, optionally with prior $\tilde{\Phi}_i$
9: end for
```

5.7.2 Problems and Limitations

Problems can occur in various situations. If the object is changing position in the image plane from frame i to $i+1$ very rapidly, so that there is no or very little overlap with the initial embedding function Φ_{i+1}, the level set evolution will usually fail to segment the object, as illustrated in Figure 5.16.

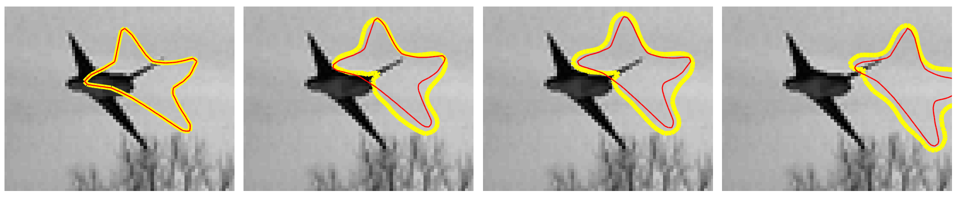


Figure 5.16: Example for the failure of the level set segmentation to capture the tracked object. Yellow/red: zero level set and prior, respectively. Shown on the far left is the initialisation from the previous frame in the sequence. The other images depict several stages of the level set evolution. It can be seen that the movement of the object with respect to the image plane is too large, so that the subsequent level set evolution does not segment the object. In the *image plane*, no motion model was used.

This problem could be alleviated for example by solving

$$\tilde{T} = \arg\min_T \int_\Omega (g(x) - c_I)^2 \, (1 - H(\Phi(x+T))) \, dx \,, \tag{5.33}$$

before the actual evolution, with respect to a translation T and setting

$$\Phi_{i+1} \leftarrow \Phi(x + \tilde{T})$$

as initialisation for frame $i+1$. Equation (5.33) results in a translation that, keeping the *inside* component c_I fixed, maximises the overlap with respect to grey value difference (or, for that matter, any other data modality). Alternatively, we let the evolution run while keeping the mean colour values c_1, c_2 in Equation (4.19) fixed. As long as there is some overlap between initial Φ and object, and the object is not occluded in object colour (so that the zero level set cannot "leak out" of the object), this solves the problem for cases as depicted in Figure 5.16. Notice that this problem might also be addressed by the recently proposed *Sobolev active contours* [124, 142], where the underlying metric leads to preference of translation over local deformation in a natural way when evolving a contour; see also Section 4.14.2 for references. That is in essence also what is done when solving (5.33) before commencing the normal level set evolution: first translate, then deform. Moelich and Chan [93] also use an intermediate step that enlarges the initial contour in the current frame if the evolving contour collapses, and repeat that enlargement step followed by curve evolution until a steady state is reached without the zero level set vanishing. In our experiments, however, this situation did not occur.

Another limitation is given by the segmentation approach that is applied. In the case of level set evolution after Chan and Vese with a weighted prior term, problems usually occur when the object is occluded in object colour, or more generally, when the occlusion yields very similar features which are used for segmentation, be it colour, texture statistics, or anything else. The same naturally holds for background areas. In such cases, the data term of the level set segmentation will drive the contour in one direction, while the prior term has to stand against that force. Raising the weight of the prior term only moves the problem somewhere else: Then, the prior will be too strong to allow for contour changes necessary for the view sphere tracking — this means that view sphere tracking will get stuck.

5.7.3 Possible Priors

Points on the view sphere predicted by the motion model can be used to provide a prior when segmenting subsequent frames of an image sequence. This can be done in several ways — the most obvious is to take the shape at $p_{pred} \in \mathbb{S}^2$ from Equation (5.21) as a template. It is also possible to use the shape at the current view point to compute a prior. The used segmentation method should be able to account for silhouette changes by itself, given that the prior information does not get weighted too strongly. This proved to be working in experiments with the level set method and the above mentioned prior. To incorporate the prior into the segmentation method, it would also be appealing to impose a vector field defined on a contour C that drives C along a geodesic in shape space towards the prior; this appears to be a sensible choice and has been proposed amongst others in [61]. Parametric active contour methods seem to be naturally suited for this sort of modification, since they work directly on points lying on the contour. For the implicit level set method, applying a vector field that is defined only on the level set defining the interface is a little more involved. Imposing a flow along a geodesic in the implicit framework for other distance measures has been proposed, for example, in [119]. The prior we use is a single shape either predicted by the motion model on the view sphere, or optionally by considering the result from the previous frame. The shape is interpolated using a weighted Karcher mean and converted to a binary image. This binary image is then used as a prior for segmentation, using the shape energy term (4.40).

5.8 Experiments and Evaluation

Figures 5.17, 5.18 and 5.19 show the results of the following experiment: for a given sequence $\{I_1, \ldots, I_n\}$ of images depicting a moving object, the

contour c_1 and view sphere position t_1 for the first image were initialised manually. Then, using the methods from this chapter, for each subsequent image I_{i+1} the contour c_{i+1} and the respective view sphere point t_{i+1} were updated. The contour c_i from the previous image was used for initialisation and as a weak prior for the segmentation of image I_{i+1}. The segmentation result from I_{i+1} was then used to calculate t_{i+1}, starting at t_i, using the method described in Section 5.6.
In Figure 5.19, an occluding object was added in a different scene, which could be successfully handled by using c_i as prior template for the level set segmentation algorithm.

In these figures, only a few snapshots of the whole sequences can be seen. The complete sequences consist of 100 frames for the experiments in Figures 5.17 and 5.18, and 50 frames for the one in Figure 5.19.
These experiments show that the sphere tracking mechanism is capable of keeping track of the view sphere position fairly well in these sequences, given a sufficient number of samples on the view sphere for interpolating the shape sub-manifold corresponding to the object.

The sequences from Figures 5.17, 5.18, and 5.19 are artificial in the sense that in front of a background image, a 3D computer graphics model of an object was moved.
Figure 5.20 shows images from a real recorded sequence. The shape samples to model the view sphere sub-manifold were still taken using a 3D model. The image sequence is showing a real scene. The tracking algorithm was able to track a sensible view position for a large part of that sequence. When the object contours become too ambiguous, however, view sphere tracking will fail.

Another case of failure can be seen in Figure 5.21. Level set evolution was in that case not used, but the effect is independent of the used segmentation method. It can be seen that the contour of the object looks very similar at some point regardless of whether the view point moved forward or backward a specified path. By this symmetry, the view sphere tracking mechanism gets confused and possibly turns around, instead of following the correct path. These sorts of problems may be alleviated by using better motion models, but they will still be imminent. The information purely from outline shape is not sufficient to resolve these ambiguities.

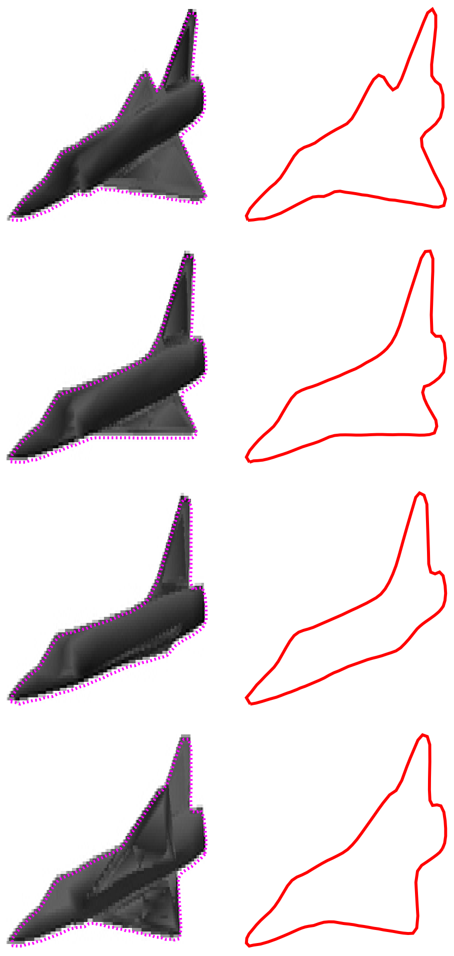


Figure 5.17: Tracking the view sphere position using only the segmented contours from a sequence of images. A few images from the sequences are shown on the left hand sides, the corresponding interpolated contours from the shape space $\mathcal{C}$ on the right. The initial position $t_0 \in \mathbb{S}^2$ and shape s_0 were given manually. Then for each image, the result from the previous one was used as initialisation. A region based level set segmentation was used, with a curvature regularisation term after [32]. *(Figure is continued)*

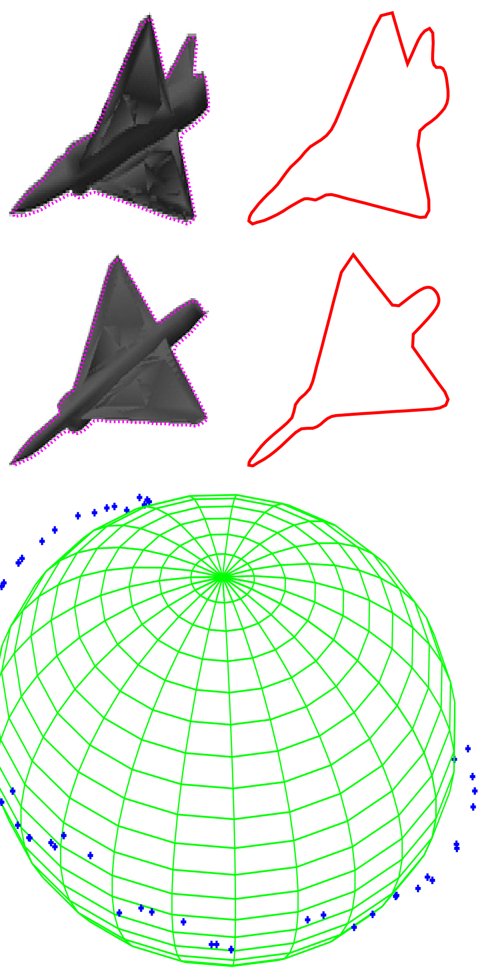


Figure 5.17: Bottom: shown are measurements obtained on the view sphere, for the complete sequence.

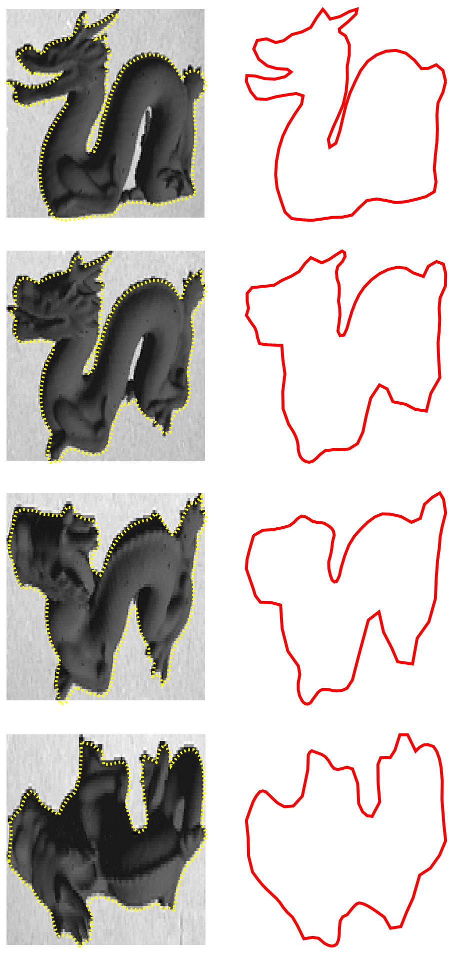


Figure 5.18: Tracking the view sphere position using only the segmented contours from a sequence of images. A few images from the sequences are shown on the left hand sides, the corresponding interpolated contours from the shape space $\mathcal{C}$ on the right. The initial position $t_0 \in \mathbb{S}^2$ and shape s_0 were given manually. Then for each image, the result from the previous one was used as initialisation. A region based level set segmentation was used, with a curvature regularisation term after [32]. *(Figure is continued)*

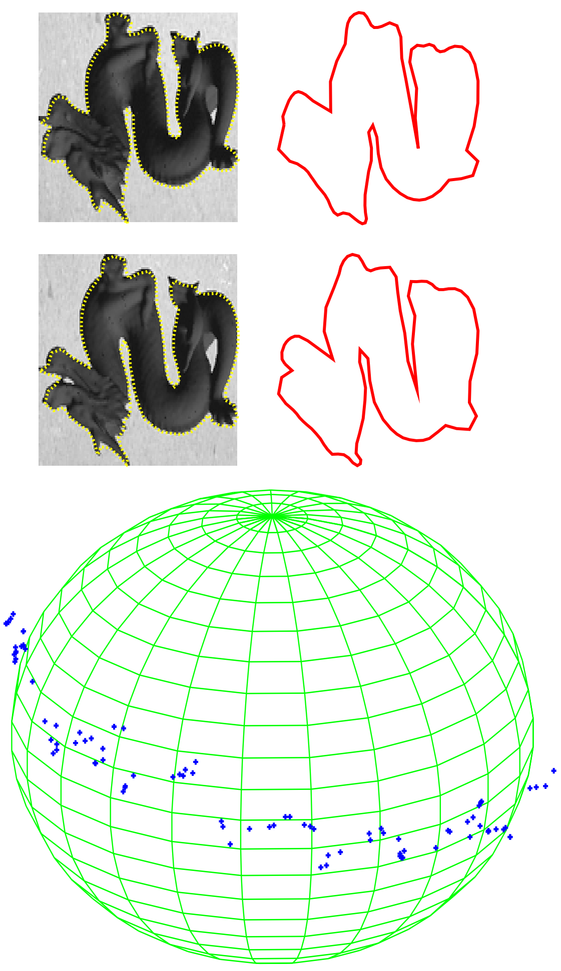
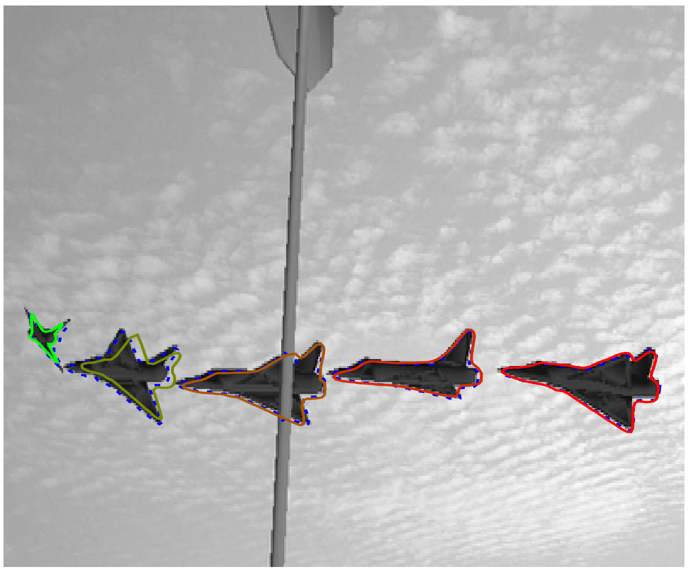


Figure 5.18: Bottom: shown are measurements obtained on the view sphere, for the complete sequence.



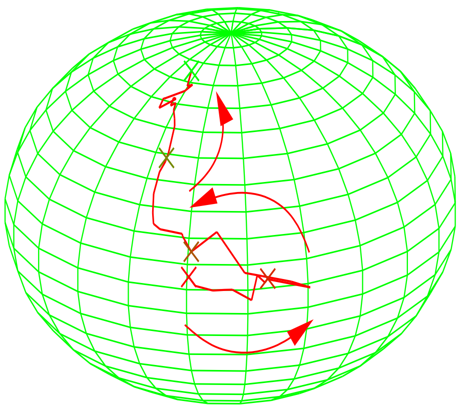


Figure 5.19: Sphere tracking experiment with occlusion. The upper image shows an illustration of the image sequence. The lower image shows the tracked view sphere path (the arrows indicate the direction of motion). The colour coding shows the corresponding contours and view sphere positions. Using the resulting shape from each previous frame to create a prior for the segmentation algorithm enables the sphere tracking to keep going for this sequence, where an occluding object moves in front of the object. *(Figure is continued)*

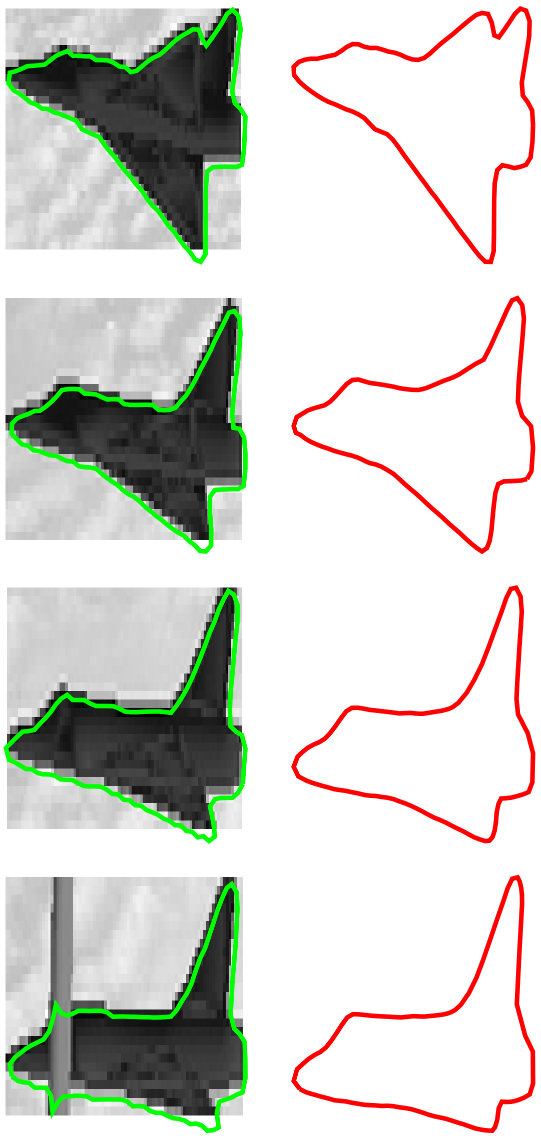


Figure 5.19: Each row shows the area of interest from every other frame with the superimposed segmentation result, followed by the contour representing the shape tracked on the view sphere. *(Figure is continued)*

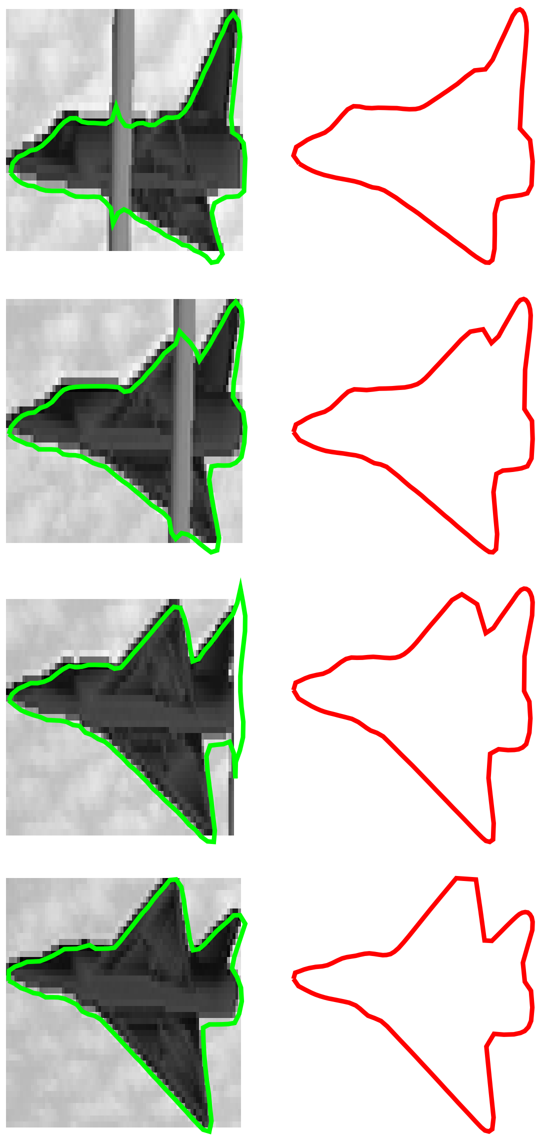


Figure 5.19: Each row shows the area of interest from every other frame with the superimposed segmentation result, followed by the contour representing the shape tracked on the view sphere.

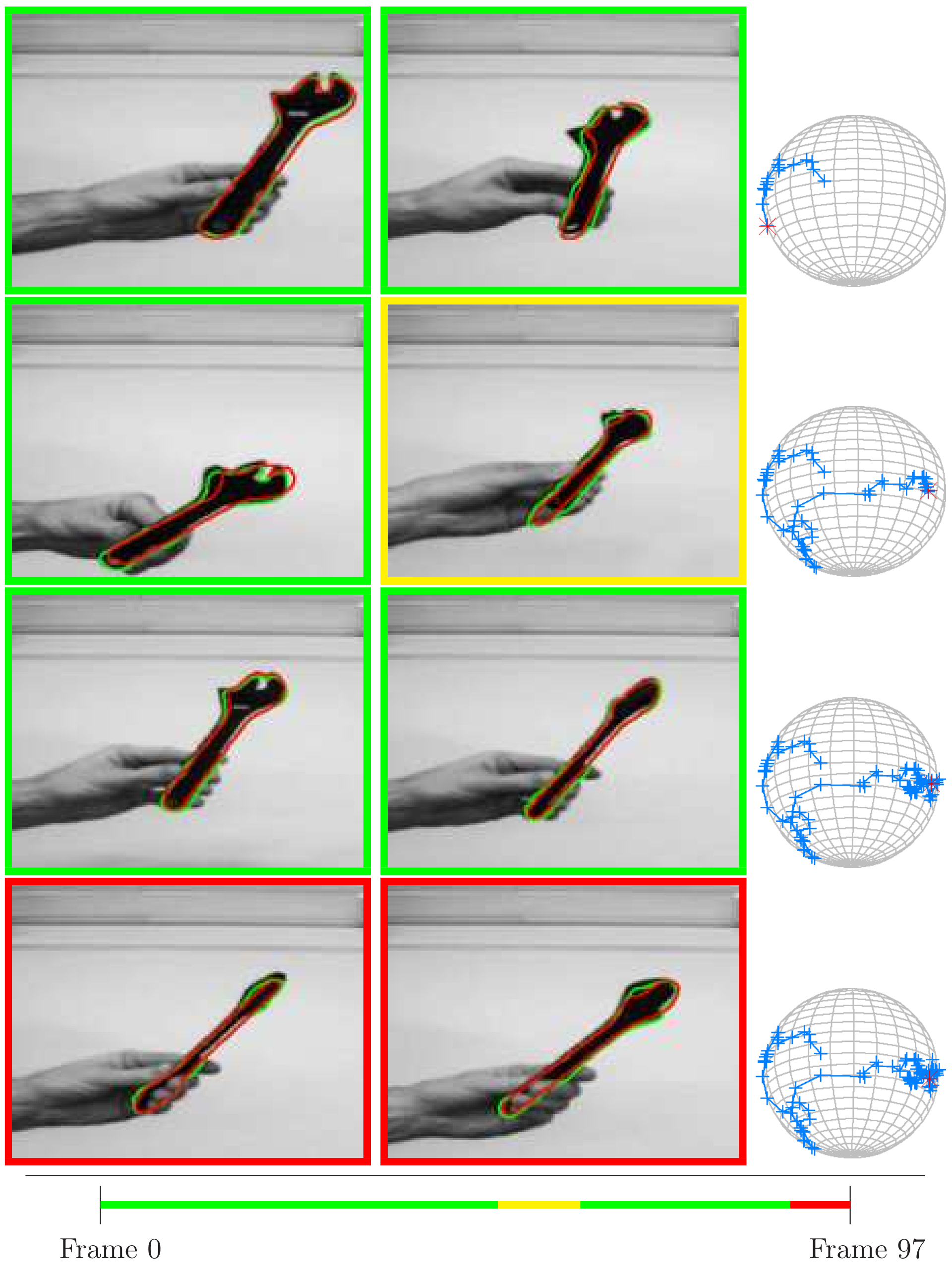


Figure 5.20: Sphere tracking with a real recorded sequence totalling 97 frames. Roughly every 20th is shown, the last three are closer. Indicated in each frame are the segmentation result (green) and aligned interpolated shape (red). Difficult situations where the view tracking goes wrong are indicated in red, yellow are situations which are just ok. The time line on the bottom indicates the situation for the whole 97 frames. The spheres on the right indicate the inferred view positions along the sequence.

5.9 Integrating with Template Matching

Until now, the introduced sphere tracking method uses the level set segmentation method to extract contours, and steers the segmentation by a predicted or previously obtained contour from the view sphere sub-manifold. Even though the results indicate that this can work quite well, some reasons for trying something different may come to mind:

1. The segmentation method inherently adds freedom for the curve to evolve to any shape, even those which are not in the modelled sub-manifold. While one can argue that this may in the future enable methods to detect loss of object, that means leaving the sub-manifold, detection is currently not the focus of this method.

2. The level set segmentation method adds more parameters that need to be tuned, specifically the weights for the various energy terms.

This section points out one rather obvious alternative, despite the fact that in this work, we keep using level set segmentation for its greater flexibility towards possible extensions.
As a replacement for the curve evolution in the proposed sphere tracking method, we tried a template matching approach. This is somewhat similar to Schmaltz et al. [114] who match the projection of a 3D model directly to image data.
For a simple, closed, regular curve c, let $A_c : \mathbb{R}^2 \mapsto \{0,1\}$ be an indicator function mapping each position on the 2D image to 0 or 1, so that

$$A_c(x) = \begin{cases} 0 & \text{if } x \text{ is outside the curve } c \\ 1 & \text{if } x \text{ is inside the curve } c\,. \end{cases}$$

Next, consider an energy

$$\begin{aligned} E(A_c, s, \beta, T, c_1, c_2) := &\int_\Omega A_c(s\,\Gamma\,x + T)\,\frac{1}{N}\sum_{i=1}^{N} \lambda_{1,i}\,(g_i - c_{1,i})^2\,dx \\ &+ \int_\Omega [1 - A_c(s\,\Gamma\,x + T)]\,\frac{1}{N}\sum_{i=1}^{N} \lambda_{2,i}\,(g_i - c_{2,i})^2\,dx\,, \end{aligned} \quad (5.34)$$

for an N-channel image $g : \mathbb{R}^2 \supset \Omega \mapsto \mathbb{R}^N$, $g_i \geq 0$, scale $s \in \mathbb{R}^+$, 2D rotation matrix $\Gamma(\beta) \in \mathcal{SO}_2$ with angle β, translation $T \in \mathbb{R}^2$, and $c_1, c_2 \in \mathbb{R}^N$. λ_1, λ_2 are vectors of weights with $\lambda_{j,i} \geq 0$, $j = 1, 2$, $i = 1 \ldots N$. This is minimised for the transformation s, β, T and c_1, c_2 in alternation, similar to the minimisation in e.g. [131]:

1. Minimise E for s, β, T with fixed c_1, c_2

2. Minimise for c_1, c_2; the optimal values are just the the mean inner and outer grey values

3. Repeat until convergence

Notice that to be differentiable, A_c can be approximated by, for example,

$$A_\varepsilon = H_{\varepsilon,1}\left(A_c - \frac{1}{2}\right)$$

with $H_{\varepsilon,1}$ from (4.2).
Then we replace the energy F in (5.22) by

$$F(c, g) \coloneqq \min_{s,\beta,T,c_1,c_2} E(A_c, s, \beta, T, c_1, c_2)\,.$$

$F(c, g)$ maps to each pair of curve c and image g an energy. This is used to replace F in Equation (5.26), where we use $F(\mathrm{curve}(m(t)), g)$ with a map

$$\mathrm{curve}\,:\,\mathcal{C}_N \mapsto \mathbb{R}^{N\times 2}$$

that takes each shape m to a corresponding discrete representation of a curve $c = \mathrm{curve}(m)$ that is aligned to the image structure, for example using the resulting curve from the previous frame or the given initial curve.
Figure 5.21 illustrates with an experiment that this template matching based sphere tracking also works in principal. In the same figure, another problem can be seen that is not directly connected with the method used for contour evolution or extraction: if the object exhibits silhouettes which are very similar in distinct directions of motion along the view sphere, the view point tracking can possibly follow the wrong direction — there is no way it can differentiate good from evil. In the depicted example, the actual path on the view sphere goes just one time around the sphere, while sphere tracking turns around at a critical position where there is a symmetry in shape change.

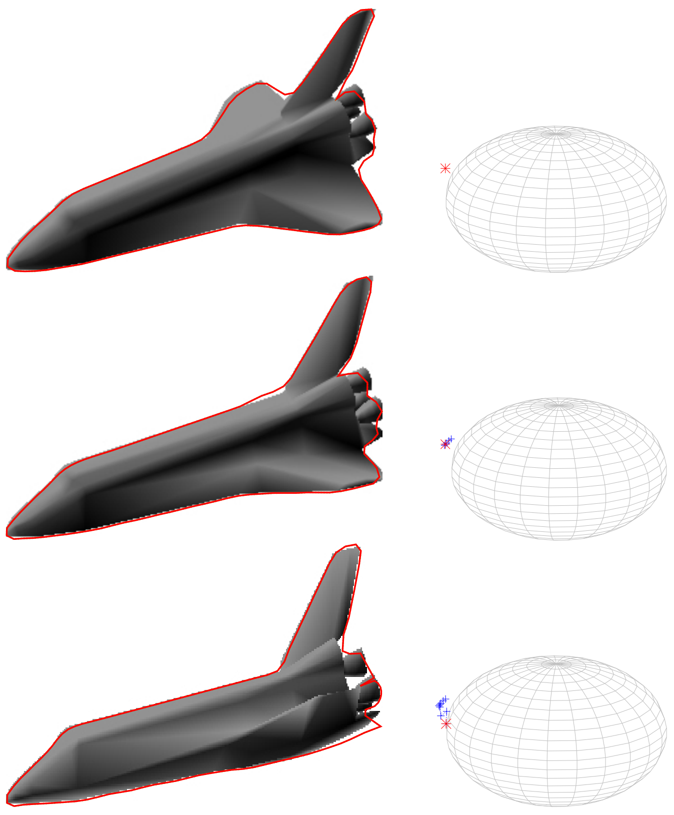


Figure 5.21: Sequence showing sphere tracking using the template matching method described in Section 5.9. Shown are the object image and the interpolated contour at the respective tracked sphere position, for every few frames. It is evident that the basic mechanism works here. What can also be seen on the following pages is that symmetries in the object contour lead to problems. *(Figure is continued)*

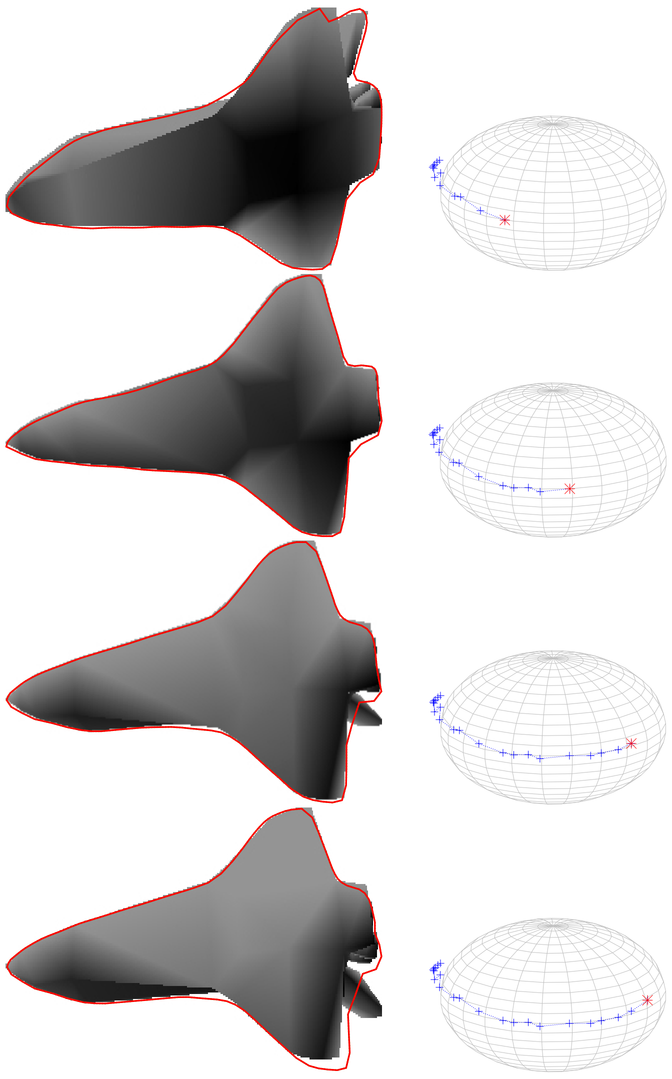


Figure 5.21: *(Figure is continued)*

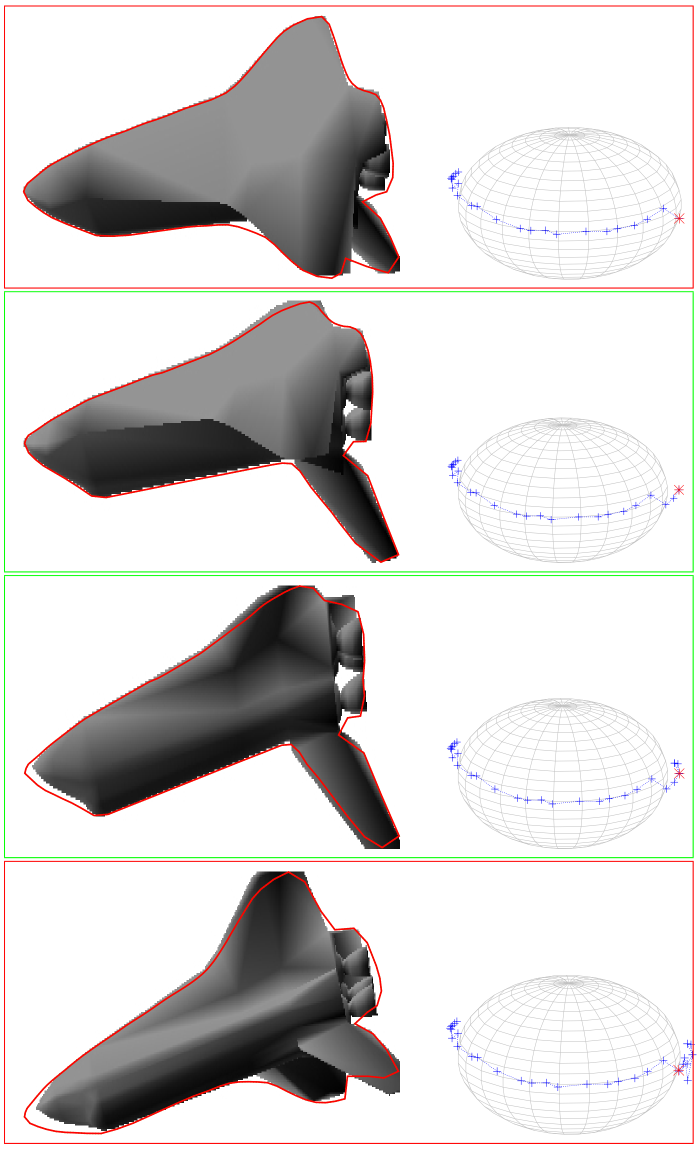


Figure 5.21: Looking at the boxed frames, it can be seen that the silhouettes are very similar in opposite directions of rotation. Therefore, sphere tracking can follow the wrong direction: instead of going on around the sphere, it turns around at about half the way (the rotation is once around the whole sphere). In the following frames, the tracked position is therefore wrong and tracking breaks down.*(Figure is continued)*

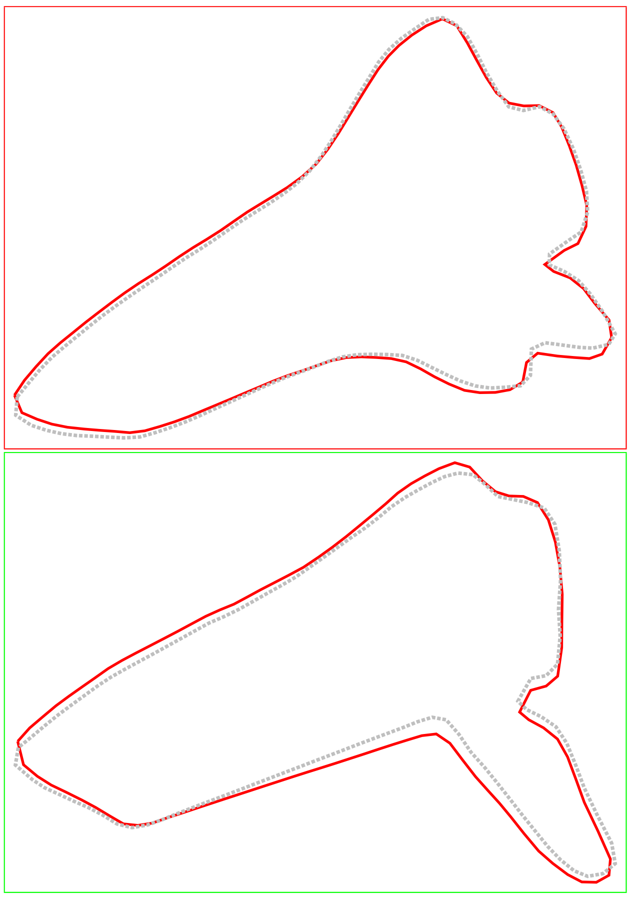


Figure 5.21: The shapes from the previous page are shown here, aligned and in different colours for better visibility. Red framed, upper box: the two shapes from the upper and lower frame on the previous page. They are so similar that a difference is barely discernible. The same holds for the lower, green framed box: it shows the two shapes from the middle two frames from the previous page.

Chapter 6

Conclusions

6.1 Discussion and Possible Future Work

In this work, we considered contours with one connected component. One possible future line of thinking could be to investigate possibilities for models with multiple connected components, that means multiple curve segments. If such models can be found and applied, they could possibly help in situations like the one depicted in Figure 6.1. In the figure you can see the outer contour of a rotating office chair in the upper row, while the bottom row shows also inner contours of the same rotation sequence. The outer contour changes quite abruptly at times, which is problematic if we are relying on smooth changes. The sequence showing also inner contours looks overall smoother and also contains more information about the object.

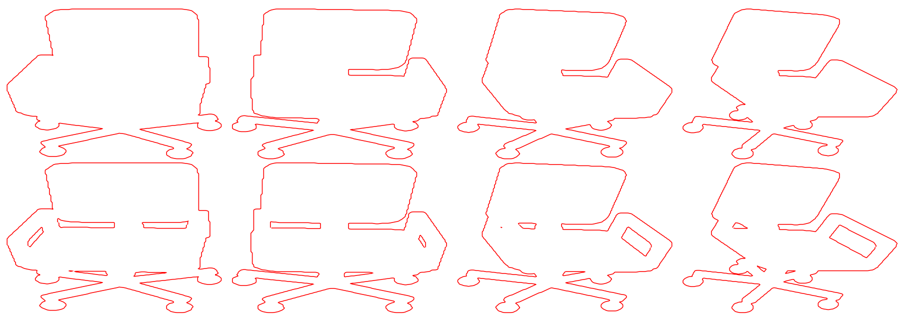


Figure 6.1: Contours of a rotating office chair. The upper row shows just the outer contour of the chair, while the bottom row shows also inner contours. It can be seen that the outer contour here exhibits abrupt changes, while outer and inner contours together change more smoothly.

Another point that may be added are different flows for the evolution of curves, such as flows along geodesics in $\mathcal{C}$. Although, whether that could improve the actual sphere tracking is not obvious.

Looking at modelling motion of a point on a sphere, model predictive control methods [13] may be investigated, which would then have to be applied to spheres.
On the computational side, taking exponential and logarithmic maps on $\mathcal{C}_N$ are the most time consuming parts of the proposed sphere tracking method. Are there faster and/or more exact alternatives?
The initialisation of the sphere tracking mechanism is done manually, so the starting conditions are known. One can think of automatic or semi-automatic initialisation schemes if such a method should be considered for use; for example, starting at several sampled sphere points close to an initial segmentation result and using these for sphere tracking concurrently, subsequently dropping all but one following some quality criterion.

Interpolation Quality and Adaptive Sampling

It is obvious that the quality of the interpolation at a point $s \in \mathbb{S}^2$ depends on the neighbouring shapes. *Quality* here means *how sensible is the interpolation* compared with a *real* outline taken at s. Clearly, it must be possible to automatically match the neighbouring shapes to one another in a way that matches naturally corresponding points. Therefore, the local variation in shape among the samples taken from around the view sphere must not be too large.
The local shape variation on $\mathbb{S}^2$ will be depending on the object at hand. Imagine the example of an aeroplane, and imagine it first seen from the top: when varying the camera position, one would still expect to get similar silhouettes which can be matched well. Now, imagine the same aeroplane seen from one side. In that case the silhouettes can change quite rapidly, for example due to the wings being visible after a small change of the view point.
So in the first case, one can sample less densely, while in the latter case one would need to sample more densely to get shapes which can indeed be matched in a way that makes sense. Just sampling very densely everywhere is not an option if an additional aim is to keep the number of sampled shapes low. Consequently, the sampling should be done in an adaptive manner, steered by a measure of local variability in shape, whenever possible.
The alternative to a denser sampling would be hand-labelled landmarks for each neighbouring pair of contours, which is a lot of tedious work we would like to avoid. However, this may be the only way out in cases where shape matchings can be ambiguous. For an illustration of the problem, refer first to Figures 6.2 and 6.3. Both show a triangle of shapes taken from the view sphere, together with a few additional contours obtained using images of the object, and their interpolated counterparts. The latter figure shows a triangle spanning a wider area of angle than the former. An example for a blatantly failed matching can be seen in Figure 6.4.

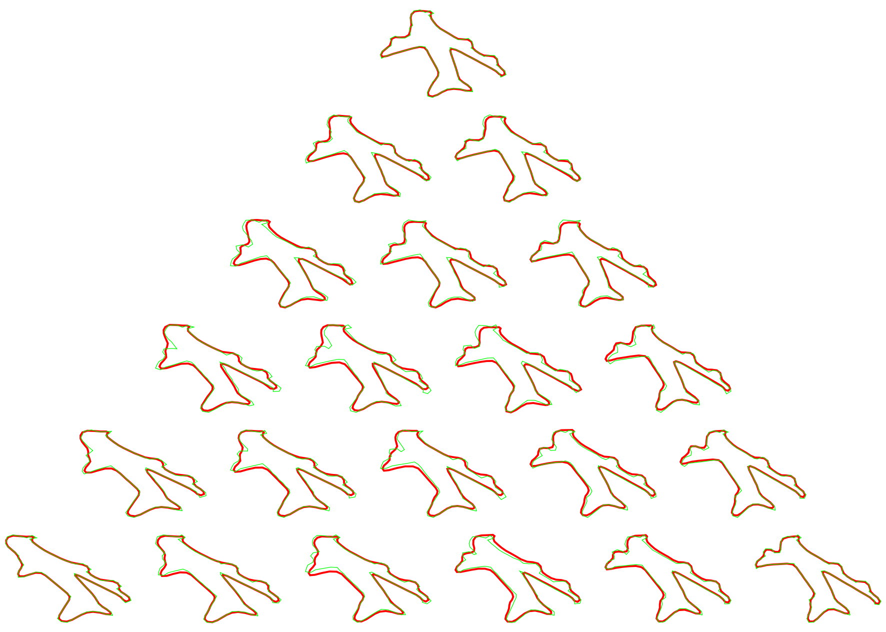


Figure 6.2: A first illustration of the difference between samples actually taken from the view sphere (green) and contours interpolated geodesically as described in Section 5.4.2 (red). Each pair is for visual purposes aligned using Procrustes distance minimisation. The differences in detail in the curve pairs, notably in the corners of the triangle, stem from the fact that the sampled (green) curves are the original curves from an image, while the interpolated (red) curves are reconstituted from (Φ, Θ) pairs.

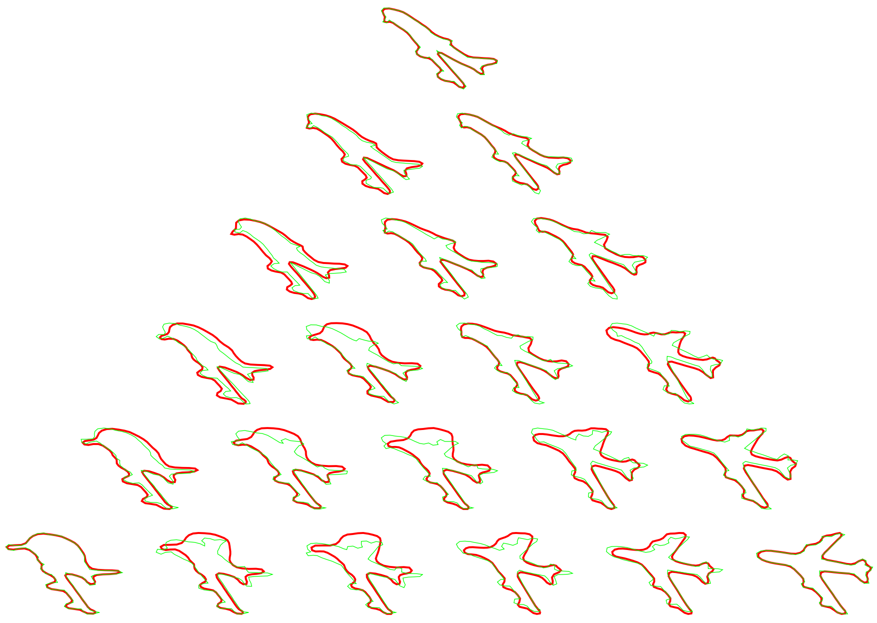


Figure 6.3: The same illustration as in Figure 6.2, but here the triangle spans a wider angle on the view sphere at different positions. The deviation of the interpolation from the sampled curves clearly gets larger.

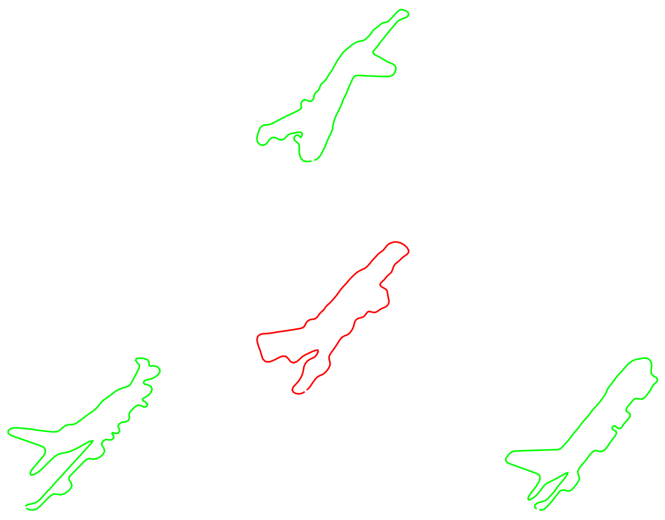


Figure 6.4: Illustration of bad matching while approximating a contour on the view sphere. The three green contours are samples from $\mathcal{C}_{S,N}$, the red contour is an approximation using equal weights for the green contours. It can be seen that the upper contour was not matched sensibly to the others. This example shows contours of an aeroplane sampled from the vertex points of a triangulated sphere with 162 roughly equidistant vertices.

Figures 5.6 and 5.7 show triangles from the discretisation of the view sphere, with corresponding sampled and interpolated shapes from the example object m1249. The geodesic distances between interpolation and sample are also given in numbers and colour coded in the colour of the interpolated shape. Figure 5.6 shows an area on the view sphere where the interpolation results are quite acceptable. Figure 5.7 shows an area where the interpolation is partly quite bad. A map of the geodesic distances between sampled and interpolated shapes for all view coordinates can be seen in Figure 6.5. One can see there that there are a few peaks pointing out areas where the interpolation gets bad. A possible future extension would be to use such information to adjust the sampling density on the view sphere.

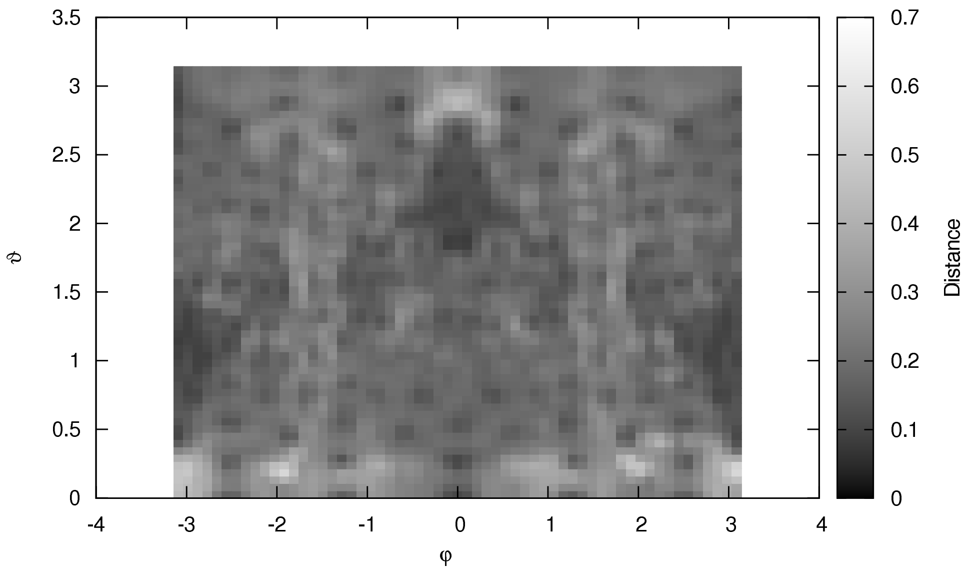


Figure 6.5: Geodesic distance between interpolated and sampled shapes for all view sphere positions, here given in spherical coordinates. Given is an approximation of the sphere by a mesh of triangles. In each triangle, distances between interpolated and sampled shapes were calculated at positions $p_i = (0.2, 0.2, 0.2) \cdot (k, l, m)$, $\mathbb{N} \ni k, l, m \geq 0$, $\sum_j p_{i,j} = 1$, such as illustrated in Figures 5.6 and 5.7. While this plot itself is interpolated from scattered data and therefore not exact, the important thing to notice is that there are regions where interpolations are quite close to sampled shapes, and some regions exhibit peaks where the interpolations are bad. See also Figures 5.6 and 5.7 for depictions of sampled versus interpolated shapes in selected triangles from the view sphere of this object (m1249).

Motion Model

As has already been mentioned in the beginning of this section, replacing the simple mechanical motion model by a model predictive control approach is a possibility for future extension.

Integration of Shape Term

A drawback of the simple integration of shape into the segmentation framework described in Section 5.7 is that this type of prior integration does not honour the structure imposed by the inner product (3.4) on the space $\mathcal{C}_N$. A different approach is to calculate an external vector field that reflects the amount of stretching and bending that needs to be applied to a curve C so that it approaches another curve C_p along a geodesic in $\mathcal{C}_N$.

Assume the embedding function $\Phi(x)$ and $C = \{x : \Phi(x) = 0\}$, and also a given prior curve P. Since we work with implicit *and* explicit representations, we unfortunately have to extract C from $\Phi(x)$ in order to calculate a geodesic, noting that this kind of prior integration would be more natural to do in parametric active contour models [61].
The way the proposed approach would work with level sets is then

1. Extract C from $\Phi(x)$
2. Let $p, c \in \mathcal{C}_N$ represent $P, C \in \mathbb{R}^{M\times 2}$ in closed pre-shape space
3. Match p to c
4. Calculate $V_c \coloneqq \mathrm{curve}(\mathrm{Exp}_c(\mathrm{Log}_c(p)\cdot\Delta)) - \mathrm{curve}(c)$, a velocity field defined on C which evolves C towards the prior along a geodesic; Δ is a small constant; $\mathrm{curve}(\cdot)$ takes an element from $\mathcal{C}_N$ and creates a configuration matrix
5. Transform V_c back to the original curve C using the optimal euclidean transformation that takes $\mathrm{curve}(c)$ to C (e.g. from full Procrustes distance minimisation) and call the result V_C
6. Initialise grid cells of $\Phi(x)$ adjacent to the zero level set using V_C
7. Extrapolate the velocity field
8. Evolve with this external field using (4.12)

6.2 Summary

Thinking about different properties of objects contained in images, and using these properties for machine vision tasks, object shape is certainly a strong one: While colour and texture can change due to influences like object instance or lighting conditions, the object outline may remain relatively stable considering similar views. This makes shape an attractive cue for object recognition, view tracking and pose estimation, and possibly other goals of machine vision. However, the problem of how to reliably *obtain* object contours from image data looms constantly over all applications using shape information. This leads to *image segmentation*, a problem which has proven to be notoriously difficult, and to which there has not been a general solution. Instead, there are many approaches to image segmentation that mostly deal with a very specific range of images, making certain assumptions about objects to be segmented. Some of the approaches which employ curve evolution techniques using the level set method have been reviewed in Chapter 4.
Summarising, the level set, region based methods have some attractive properties. For one, embedding curves in a higher dimensional function allows

for parameter-free representation, at the cost of higher computational complexity. The region based methods are naturally more resistant to noise than edge based methods, and more diffuse object boundaries are not so much a hindrance in segmentation, since boundaries are not detected using gradient information like for example in the geodesic active contour formulation. Another good property is the relative ease of integration of additional terms into the segmentation energy, such as the priors mentioned in Chapter 4. Finally, a drawback of the level set method compared to a parametrised representation is computation speed. This can be alleviated a little by computing only on a narrow band around the zero level set, but that makes contours "popping out of nowhere" impossible.

Working with contour shape requires a way of representation, and in our case also a way of calculating weighted averages. A glimpse of classical shape analysis was given in Chapter 2, and a more recent approach using an elastic shape metric in Chapter 3. Using these techniques, a method for tracking a view point relative to an object was introduced in Chapter 5. This technique uses only 2D shape information as input and also only 2D shape for internal object representation, with additional information about view position. 3D models were only used in order to automate and simplify experiments, not for object representation. While using 2D shape is potentially more memory efficient than using 3D information, especially in conjunction with an optimised sampling strategy when obtaining model shapes, it is also potentially closer to a representation of object shape in human brains as mentioned in the introductory Chapter 1. Whether leering at biological systems is always a good idea or not is of course open for discussion. My own point here is that since brains appear to do some things very efficiently, it is worth trying.

Change in contour shape does appear to convey considerable information about changing view points, even if considered isolated like in this thesis. But obviously, experiments also show that additional information is needed to conceive models incorporating shape as principal source of information for view point tracking that could be truly robust in practice. Directly using recently described shape manifolds allows us to work with object models that result in reasonable shape deformation, without using 3D models as internal object representation. This does, however, depend on the complexity of local shape changes of the object in question.

Appendix A

Level Sets

A.1 Method of Characteristics

The method of characteristics is a method for solving partial differential equations (PDE) of the form

$$\frac{\partial u(x,t)}{\partial t} + a(x,t)\,\frac{\partial u(x,t)}{\partial x} = b(x,t) \tag{A.1}$$

$$u(x,0) = u_0(x) \tag{A.2}$$

with $x \in \mathbb{R}$, $t > 0$. It is used to explain upwind differencing in the next section and is therefore summarised here.
Consider the differential equation $\underline{\text{in } x(t)}$

$$\frac{dx(t)}{dt} = a(x(t),t) \tag{A.3}$$

$$x(0) = x_0\,. \tag{A.4}$$

$(x(t),t)$ is a curve starting in the point $(x = x_0, t = 0)$ and is called a *characteristic*. Exploring the change of u along this curve, one gets

$$\frac{du(x(t),t)}{dt} = \frac{\partial u(x,t)}{\partial t} + \frac{\partial u(x,t)}{\partial x}\,\frac{dx(t)}{dt} = \frac{\partial u(x,t)}{\partial t} + a(x,t)\,\frac{\partial u(x,t)}{\partial x} = b(x(t),t)\,.$$

This means the solution of u of the original PDE is constant along $(x(t),t)$ for homogeneous problems, and given by integrating b over time for non-homogeneous problems, and therefore

$$u(x(t),t) = u(x_0,0) = u_0(x_0) + \int_0^t b(x(\tau),\tau)\,d\tau\,. \tag{A.5}$$

To solve the original PDE, one finds the characteristics given by the solutions to (A.3) – (A.4) and uses those to calculate solutions for the PDE along these characteristics with (A.5) [133].

A.2 Upwind Differencing

Consider a partial differential equation

$$\frac{\partial u(x,t)}{\partial t} + a(x,t)\,\frac{\partial u(x,t)}{\partial x} = 0 \tag{A.6}$$

with initial conditions

$$u(x,0) = u_0(x) \tag{A.7}$$

and a backward difference scheme, in one dimension to simplify illustration. Any grid point x depends on one neighbouring grid point, and this results in a triangular domain of dependence when calculating at point P shown in the time-space diagram in Figure A.1. In this figure, the lines PQ, PQ_2, PQ_3 are

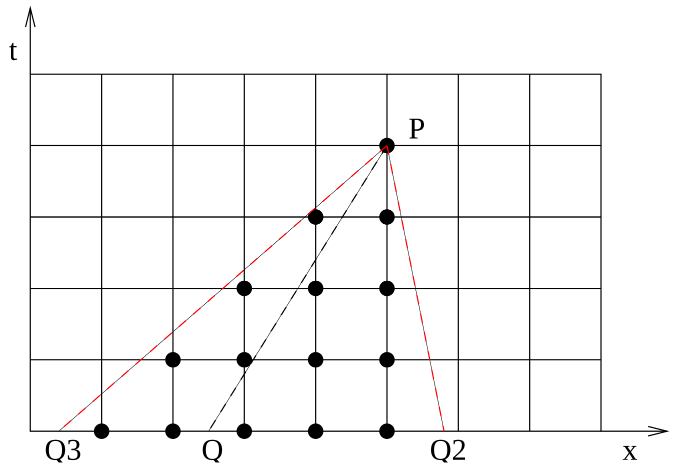


Figure A.1: Example for a domain of dependence for a one dimensional finite difference scheme using backward finite differences. $PQ, PQ2, PQ3$ illustrate lines of characteristics, where the solution to the partial differential equation is constant. PQ lies within the numerical domain of dependence, while the other two red lines are out of the domain of dependence. For the latter two characteristics, the scheme would therefore be unusable.

characteristics, along which information is transported over time by a PDE. These can be seen as defining a domain of dependence of the PDE. Now if this domain of dependence lies outside the numerical domain of dependence, the numerical scheme cannot capture the transported information and so will yield wrong results. This is the statement made by the CFL condition (after Courant, Friedrichs, Lewy [23]) for the convergence of finite difference schemes. Figure A.1 also illustrates that if $a < 0$ in Equation (A.6) (line PQ_2 in Figure A.1), the backward difference scheme is unable to capture any characteristic. Therefore, if $a < 0$, forward differences must be used.
What all of this means is that the *physical* flow of information governed

by the partial differential equation in question may not be faster than the *numerical* flow of information. Or equivalently, the *numerical* wave must travel at least as fast as the *physical* wave. In addition, if $a < 0$ we must use forward differences, and if $a > 0$ we must use backward differences. Schemes working after the latter principle are called *upwind*, since they always use information from the direction from which the information is transported[1] in order to approximate derivatives.

A.3 Derivation of Statistical Update Term

The term (4.31) in Section 4.9.3 can be derived as follows. Given

$$E(\Phi) = -\int_\Omega \sum_{j=1}^{N} \log p_{1,j}(u_j)\, H(\Phi)\, dx - \int_\Omega \sum_{j=1}^{N} \log p_{2,j}(u_j)\, (1 - H(\Phi))\, dx\,.$$

$$\begin{aligned}\left.\frac{d}{d\nu}\right|_{\nu=0} & E(\Phi + \nu\,\phi) = \\ & \left.\frac{d}{d\nu}\right|_{\nu=0} \Bigg[-\int_\Omega \sum_{j=1}^{N} \log p_{1,j}(u_j)\, H(\Phi + \nu\,\phi)\, dx \\ & \qquad - \int_\Omega \sum_{j=1}^{N} \log p_{2,j}(u_j)\, (1 - H(\Phi + \nu\,\phi))\, dx \Bigg] \end{aligned} \tag{A.8}$$

$$= -\int_\Omega \sum_{j=1}^{N} \log p_{1,j}(u_j)\, \phi\, H'(\Phi)\, dx + \int_\Omega \sum_{j=1}^{N} \log p_{2,j}(u_j)\, \phi\, H'(\Phi)\, dx \tag{A.9}$$

and setting this equal zero gives for any arbitrary ϕ

$$\begin{aligned} & \sum_{j=1}^{N} \left(-\log p_{1,j}(u_j) + \log p_{2,j}(u_j)\right)\, H'(\Phi) \\ = \; & \sum_{j=1}^{N} \log \frac{p_{2,j}(u_j)}{p_{1,j}(u_j)}\, H'(\Phi) = 0\,. \end{aligned} \tag{A.10}$$

Taking the negative of this gives the gradient descent (4.31).

[1]One can think of the direction "from which the wind blows".

Appendix B

Global Segmentation

B.1 Introduction

In Chapter 4, a few aspects of level set based segmentation methods were introduced. For the most part, the basis was the piecewise constant approximation by Chan and Vese [131] of the Mumford-Shah [95] model for segmentation. One of the properties of the piecewise constant, region based segmentation method is that the respective energy has local minima. While this is no problem in practice if a given initialisation is close enough to the desired solution and no narrow-band methods are being used, the result certainly *depends* on the initial values. Chan et al. [17] introduce ways to manipulate the original method from [131] for two-phase segmentation so that the minimisation problem is convex and the result can be transformed into a global optimum of the original problem *for fixed constant grey levels* in the two regions. That work builds largely on previous work from Chan and Esedoḡlu [16] and on work from Strang [123, 122] on maximal flows through continuous domains.
Note that [17] and [16] also treat image denoising with the widely spread and quite successful Rudin-Osher-Fatemi (ROF) model [112] and a modification of the same. This short appendix can be seen as an addition to Chapter 4 and deals with the image segmentation part, predominantly because it gives some insight into why the Chan and Vese model works well in practice, and since it can provide an alternative to the standard level set segmentation used for sphere tracking, in appropriate situations.

B.2 Globally Optimal Piecewise Constant Segmentation

With an image $f : D \mapsto \mathbb{R}$ and $D \subset \mathbb{R}^n$ denoting the image domain, and $\Sigma \subset D$ a subset of D, write the piecewise constant segmentation functional as

$$E(\Sigma, c_1, c_2) := \operatorname{Per}(\Sigma; D) + \lambda \int_\Sigma (c_1 - f(x))^2\, dx + \lambda \int_{D\setminus\Sigma} (c_2 - f(x))^2\, dx \quad \text{(B.1)}$$

(cf. Section 4.9.2). $\operatorname{Per}(\Sigma; D)$ denotes the perimeter of $\Sigma \subset D$.
The segmentation task is to solve

$$\min_{c_1, c_2 \in \mathbb{R};\, \Sigma \subset D} E(\Sigma, c_1, c_2)\,. \quad \text{(B.2)}$$

The set Σ is the set of points segmented as foreground, while $D \setminus \Sigma$ denotes background. For any fixed Σ, the optimal c_1, c_2 are given by the average over foreground and background, respectively:

$$c_1 = \frac{1}{|\Sigma|} \int_\Sigma f(x)\, dx\,,\; c_2 = \frac{1}{|D \setminus \Sigma|} \int_{D\setminus\Sigma} f(x)\, dx\,. \quad \text{(B.3)}$$

The minimisation of (B.1) is then usually done in two alternating steps: firstly, assume Σ constant and calculate c_1, c_2, and secondly assume c_1, c_2 constant and minimise for Σ. Since (B.1) is non-convex because the set of admissible Σ is non-convex, the minimisation is generally difficult and will get stuck in local minima.
Chan et al. [17] note that if $c_1, c_2 \in \{0, 1\}$ and if $f(x)$ is a binary function (image), then the above segmentation problem reduces to the ROF denoising problem [112] for binary functions.
Chan and Vese [131] model the boundary of Σ with the zero level set of an embedding function Φ, and the segmentation energy then reads

$$\begin{aligned} E_{CV}(\Phi, c_1, c_2) = &\int_D |\nabla H_\varepsilon(\Phi(x))|\, dx \\ &+ \lambda \int_D H_\varepsilon(\Phi(x))\,(c_1 - f(x))^2 + [1 - H_\varepsilon(\Phi(x))]\;(c_2 - f(x))^2\, dx \end{aligned} \quad \text{(B.4)}$$

with H_ε an approximation of the Heaviside function. The first variation of (B.4) leads to the gradient descent

$$\frac{d\Phi}{dt} = H'_\varepsilon(\Phi) \left[\operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|} \right) - \lambda \left((c_1 - f(x))^2 - (c_2 - f(x))^2 \right) \right]. \quad \text{(B.5)}$$

Now, Chan et al. observe in [17] that due to the noncompactly supported approximation of the Heaviside function that was used in [131], for the stationary solutions of (B.5) it must hold that

$$\operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|} \right) - \lambda \left((c_1 - f(x))^2 - (c_2 - f(x))^2 \right) \equiv 0 \quad \text{(B.6)}$$

since $H'_\varepsilon(\Phi) \neq 0$ everywhere, so (B.5) and

$$\frac{d\Phi}{dt} = \operatorname{div}\left(\frac{\nabla\Phi}{|\nabla\Phi|} \right) - \lambda \left((c_1 - f(x))^2 - (c_2 - f(x))^2 \right) \quad \text{(B.7)}$$

have the same stationary solutions.
They also provide an energy for which (B.7) is the gradient descent, which is

$$\int_D |\nabla\Phi(x)|\, dx + \lambda \int_D \left((c_1 - f(x))^2 - (c_2 - f(x))^2\right) \Phi(x)\, dx\,. \tag{B.8}$$

Since this is linear in Φ, (B.7) does really not have a stationary state, but approaches $\pm\infty$ depending on the sign of Φ, while the zero level set remains unaltered. This effect can be seen in experiments, also when one uses the H_ε-term. In that case, the effect is much slower depending on ε, but still there.
To get rid of this, the embedding function Φ is restricted so that

$$0 \leq \Phi(x) \leq 1 \quad \forall x\,.$$

In [17] a proof is provided for the following theorem (Theorem 2 in the paper) by transforming the stated energy into one that differs from (B.1) only by an additive constant that is independent of u. Note we follow the notation of [17] and write u instead of Φ now.

Theorem B.2.1 *Fix some $c_1, c_2 \in \mathbb{R}$. A global minimiser for (B.1) can be found by solving*

$$\min_{0\leq u\leq 1} E(u) \tag{B.9}$$

$$E(u) = \int_D |\nabla u(x)|\, dx + \lambda \int_D \left[(c_1 - f(x))^2 - (c_2 - f(x))^2\right] u(x)\, dx \tag{B.10}$$

followed by setting

$$\Sigma = \{x\, :\, u(x) \geq \mu\}$$

for almost any choice of $\mu \in [0, 1]$.

To solve (B.9), it is further proved in [17] that

$$\min_{0\leq u\leq 1} \int_D |\nabla u(x)|\, dx + \lambda \int_D s(x)\, u(x)\, dx$$

has the same solutions as

$$\min_u \int_D |\nabla u(x)|\, dx + \int_D \alpha\, \nu(u(x)) + \lambda\, s(x)\, u(x)\, dx \tag{B.11}$$

for

$$\nu(z) := \max\left\{0, 2\left|z - \frac{1}{2}\right| - 1\right\}$$

and $\alpha > \lambda/2\, \|s(x)\|_{L^\infty}$.
Using this, a gradient descent is then given by

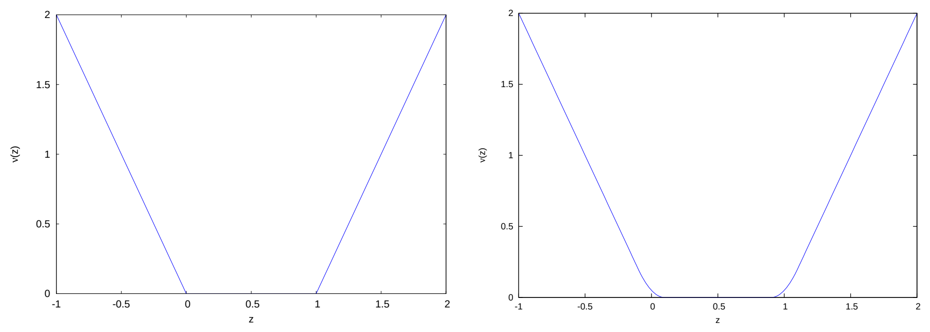


Figure B.1: Left: function $\nu(z) := \max\{0, 2\,|z - \frac{1}{2}| - 1\}$. Right: regularised version. The intervals $[-\varepsilon, \varepsilon]$ and $[1-\varepsilon, 1+\varepsilon]$ have here been regularised by using quadratic functions. In this case, $\varepsilon = 0.1$.

$$\frac{du}{dt} = \operatorname{div}\left(\frac{\nabla u(x)}{|\nabla u(x)|}\right) - \lambda\, s(x) - \alpha \nu'(u(x))\,.$$

It is worth noting that the said transformation of the original problem into a convex optimisation problem can be done because of the type of approximation chosen for the Heaviside function, H_ε. Since it has noncompact support and $H'_\varepsilon(\Phi(x)) \neq 0\ \forall x$, calculations can be extended to the whole domain D, and that is also why the original algorithm from Chan and Vese [131] is often successful in finding good segmentations. Notice that this will not be the case when using a narrow band implementation, which computes only on a narrow band around the zero level set of the embedding function — and thereby effectively disables finding contours which might otherwise be found.

Also, as has been noted in the introduction, the optimisation of the segmentation functional is done in two steps, if c_1, c_2 are unknown; one is to find globally optimal grey values c_1, c_2, the other is to find a globally optimal set Σ. It is not clear whether the two steps together also result in a global optimum of the segmentation functional for all three parameters c_1, c_2, Σ. Experiments suggest that usually, a good and thereby possibly global optimum is found — however, there is so far no proof that this is the case.

B.3 Introducing a Prior Template

A template term representing prior knowledge about the shape of a region to be segmented can be introduced similar to the level set framework. Given a template in the form of a characteristic function χ_p,

$$\chi_p(x) = \begin{cases} 1 & \text{if } x \in \text{ object region} \\ 0 & \text{if } x \in \text{ background region,} \end{cases} \tag{B.12}$$

consider an additional energy term

$$E'_p(u) = \int_\Omega (u(x) - \chi_p(x))^2 \, dx \,. \tag{B.13}$$

Setting the first variation to zero yields an update

$$\frac{\partial u}{\partial t} = -2\,(u - \chi_p)\,. \tag{B.14}$$

Inspired by this, choose

$$E_p(u) = \int_\Omega (\hat{u}(x) - \chi_p(x))\, u(x)\, dx \tag{B.15}$$

and using (B.10) minimise

$$E'(u, c_1, c_2) = E(u, c_1, c_2) + \gamma\, E_p(u). \tag{B.16}$$

The term $\hat{u}$ is a fixed version of u that gets updated after a solution to $\min_u E'$ is found. Also allowing for the transformations scale $0 < s \in \mathbb{R}$, translation $T \in \mathbb{R}^2$, and rotation $\Gamma \in \mathcal{SO}_2$ of the prior template gives

$$E_p(u, s, T, \Gamma) = \int_\Omega (\hat{u}(x) - \chi_p(s\,\Gamma\, x + T))\, u(x)\, dx \,. \tag{B.17}$$

While this works in principle [43], the use in situations where a local optimum is actually sought may be limited. Thinking about the view tracking application introduced in Chapter 5, often a local minimum will be wanted, for example in situations where background clutter is visible in similar colours or grey value as the tracked object. In such situations, methods more likely to find local minima may actually be preferable.

Appendix C

Shape

C.1 Affine Shape Matching

Expanding (2.18) into (2.17), the problem reads

$$F(A,t) = tr\Big[\Big(X_2 - X_1 \cdot A^\top + 1_m \cdot t^\top \Big) \cdot \Big(X_2 - X_1 \cdot A^\top + 1_m \cdot t^\top \Big)^\top \Big] \tag{C.1}$$

$$\{A^\star, t^\star\} = \arg\min_{A,t} F(A,t)\,. \tag{C.2}$$

We first derive with respect to A [102] to get

$$\frac{d}{dA}F(A,t) = -2 \cdot X_2^\top X_1 + 2 \cdot \Big(A X_1^\top X_1 \Big) - 2 \cdot t 1_m^\top X_1 \tag{C.3}$$

$$= 2 \cdot A X_1^\top X_1 + \text{const}\,. \tag{C.4}$$

Setting this to zero yields

$$2 \cdot A X_1^\top X_1 + \text{const} = 0 \tag{C.5}$$

$$\Leftrightarrow \quad 2 \cdot A X_1^\top X_1 = -\text{const}$$

$$\Leftrightarrow \quad A = \Big[2 \cdot X_2^\top X_1 + 2 \cdot (t 1_m^\top X_1) \Big] \cdot \Big(X_1^\top X_1 \Big)^{-1} \cdot \frac{1}{2}\,. \tag{C.6}$$

Deriving (C.1) with respect to t [102] gives

$$\frac{d}{dt}F(A,t) = 2 \cdot X_2^\top 1_m - 2 \cdot A X_1^\top 1_m + 2 \cdot t 1_m^\top 1_m\,. \tag{C.7}$$

Setting this again to zero, t becomes

$$t = \boxed{\Big(A X_1^\top 1_m - X_2^\top 1_m \Big) \cdot \frac{1}{1_m^\top 1_m}}\,, \tag{C.8}$$

where of course $1/(1_m^\top 1_m) = 1/m$.
Inserting (C.8) into (C.5) gives

$$AX_1^\top X_1 - X_2^\top X_1 - \frac{1}{m}\left[AX_1^\top 1_m - X_2^\top 1_m\right] \cdot 1_m^\top X_1 = 0 \tag{C.9}$$

$$\Leftrightarrow \quad A \cdot \left[X_1^\top X_1 - \frac{1}{m}X_1^\top 1_m 1_m^\top X_1\right] - X_2^\top X_1 + \frac{1}{m}X_2^\top 1_m 1_m^\top X_1 = 0 \tag{C.10}$$

$$\Leftrightarrow \quad A = \boxed{\left[X_2^\top X_1 - \frac{1}{m}X_2^\top 1_m 1_m^\top X_1\right] \cdot \left[X_1^\top X_1 - \frac{1}{m}X_1^\top 1_m 1_m^\top X_1\right]^{-1}}. \tag{C.11}$$

So A and t from (C.11) and (C.8) define the transformation that minimises (2.17).

Appendix D

Elastic Shape Space

D.1 Gradient of Map G

The gradient of the map G and in the continuous case $\mathcal{G}$, introduced in Section 3.2.2, is calculated as follows. Assuming the discrete equations (3.10)–(3.13), the gradients are defined via

$$\langle \text{grad } G^i, (h,f) \rangle_{(\Phi,\Theta)} = D_{(h,f)} G^i|_{(\Phi,\Theta)}$$

with $D_X Y$ denoting the directional derivative of a vector field Y in direction X. With

$$D_X Y|_p = \lim_{t \to 0} \frac{Y(p + t\,X) - Y(p)}{t}\,,$$

to calculate the directional derivative of G^1 with respect to (h,f),

$$D_{(h,f)} G^1|_{(\Phi,\Theta)} = \lim_{t \to 0} \frac{1}{t} \left(\frac{1}{N} \sum_{i=1}^{N} e^{\Phi_i + t\,h_i} - \frac{1}{N} \sum_{i=1}^{N} e^{\Phi_i} \right)\,,$$

use L'Hospital's rule to get

$$D_{(h,f)} G^1|_{(\Phi,\Theta)} = \lim_{t \to 0} \frac{1}{N} \sum_{i=1}^{N} h_i\, e^{\Phi_i + t\,h_i} = \frac{1}{N} \sum_{i=1}^{N} h_i\, e^{\Phi_i}\,. \tag{D.1}$$

Using the same definition of the directional derivative and also L'Hospital's rule for the other three components of G yields

$$D_{(h,f)} G^2|_{(\Phi,\Theta)} = \frac{1}{N} \sum_{i=1}^{N} f_i\, e^{\Phi_i} + \frac{1}{N} \sum_{i=1}^{N} \Theta_i\, h_i\, e^{\Phi_i} \tag{D.2}$$

$$D_{(h,f)} G^3|_{(\Phi,\Theta)} = \frac{1}{N} \sum_{i=1}^{N} -f_i\, \sin(\Theta_i)\, e^{\Phi_i} + \frac{1}{N} \sum_{i=1}^{N} h_i\, \cos(\Theta_i)\, e^{\Phi_i} \tag{D.3}$$

$$D_{(h,f)} G^4|_{(\Phi,\Theta)} = \frac{1}{N} \sum_{i=1}^{N} f_i\, \cos(\Theta_i)\, e^{\Phi_i} + \frac{1}{N} \sum_{i=1}^{N} \sin(\Theta_i)\, h_i\, e^{\Phi_i}\,. \tag{D.4}$$

Now, setting $(v, w) := \operatorname{grad} G^i(\Phi, \Theta)$ for each i and looking at

$$\langle (v, w), (h, f) \rangle_{(\Phi,\Theta)} = \frac{1}{N}\, a \sum_{i=1}^{N} v_i\, h_i\, e^{\Phi_i} + \frac{1}{N}\, b \sum_{i=1}^{N} w_i\, f_i\, e^{\Phi_i} = D_{(h,f)} G^i|_{(\Phi,\Theta)} \,,$$

insert Equations (D.1) – (D.4) in turn and complete the left hand sides for (v, w) to get Equations (3.18)–(3.21). The continuous case (3.6)–(3.9) can be solved in exactly the same way to yield (3.14)–(3.17).

D.2 Christoffel Symbols for Calculating Geodesics

The Christoffel symbols (3.26) can be calculated with this simple Maxima[1] program:

```
christoffel(g, u) :=
    block( [ginv: invert(g), len: length(u)],
           chris: array(chris, len, len, len),
           for i: 1 thru len do
              for j: 1 thru len do
                for k: 1 thru len do
                 (chris[i,j,k]:
                    1/2 * sum(ginv[k,l] *
                        (diff(g[i,l],u[j]) + diff(g[j,l],u[i])
                          - diff(g[i,j],u[l])), l, 1, len),
                  print ("i,j,k=", i, j, k, ": ", chris[i,j,k]))
         );
```

Entering the local representation of the metric (3.25) in Maxima,

```
g: matrix([a*exp(phi),0],[0,b*exp(phi)]);
```

defines

$$g = \begin{pmatrix} a\, e^{\phi} & 0 \\ 0 & b\, e^{\phi} \end{pmatrix} ,$$

and then

```
christoffel (g,['phi,'theta]);
```

[1] `http://maxima.sourceforge.net`. This program may be written differently, and using Maxima's itensor package.

gives

$$\begin{aligned}
i,j,k &= 111 : \frac{1}{2}\\
i,j,k &= 112 : 0\\
i,j,k &= 121 : 0\\
i,j,k &= 122 : \frac{1}{2}\\
i,j,k &= 211 : 0\\
i,j,k &= 212 : \frac{1}{2}\\
i,j,k &= 221 : -\frac{b}{2\,a}\\
i,j,k &= 222 : 0
\end{aligned}$$

from which we can then read the Christoffel symbols (3.27).

D.3 Reconstituting Curves from Elements in $\mathcal{H}_N$

Since the curve $\alpha(t) : \mathbb{R} \mapsto \mathbb{R}^2$ is

$$\alpha(t) = \alpha_0 + \int_0^t e^{\Phi(\tau)} e^{i\ \Theta(\tau)}\, d\tau\,,$$

set the discrete points $\alpha_1, \ldots, \alpha_M$ in complex coordinates in the simplest case to

$$\alpha_k \coloneqq \text{const} + \sum_{j=1}^{k} e^{\Phi_j} e^{i\ \Theta_j}$$

which, written as 2D vectors, is

$$\alpha_k \coloneqq \text{const} + \sum_{j=1}^{k} e^{\Phi_j} \cdot (\cos(\Theta_j), \sin(\Theta_j))^\top\,.$$

In practice, one may want to use a better approximation to the integral, such as the SIMPSON rule.

D.4 Calculating (Φ, Θ) from a Curve

In the following, we describe how to compute a pair $(\Phi, \Theta) \in \mathcal{C}_N$ from a curve $c : \mathbb{R} \mapsto \mathbb{R}^2$ approximated by a given polygon of points that lie on the curve. We approximate the real continuous curve with piecewise cubic polynomes. This yields, as an approximation to the velocity vector at the curve point c_i,

$$\dot{c}_i \coloneqq \frac{c_{i+1} - c_{i-1}}{2}\,.$$

The local turning angle α_i is calculated as the angle enclosed by the velocity vectors at points c_i, c_{i-1}:

$$\alpha_i \coloneqq \angle(\dot{c}_i, \dot{c}_{i-1})\,.$$

Algorithm 10 makes the computation of turning angle and speed explicit. Notice that the computation is at point p_1, not p_0.

Algorithm 10 Computing the local turning angle and speed approximation at point p_1 on a curve. sign_z is a function returning -1 if the sign of the z component of the argument is negative, or 1 otherwise.

```
Require: $p_{-1}, p_0, p_1, p_2 \in \mathbb{R}^2$ consecutive points on a regular curve
  procedure TurningAngle($p_{-1}, p_0, p_1, p_2$)
    $h_1 \leftarrow \frac{p_1 - p_{-1}}{2}$
    $h_2 \leftarrow \frac{p_2 - p_0}{2}$
    $speed \leftarrow |h_2|$
    $h_1 \leftarrow \frac{h_1}{|h_1|}$
    $h_2 \leftarrow \frac{h_2}{|h_2|}$
    $\alpha \leftarrow \arccos(\langle h_1, h_2 \rangle) \cdot \text{sign}_\text{z}\left(\begin{pmatrix} h_1 \\ 0 \end{pmatrix} \times \begin{pmatrix} h_2 \\ 0 \end{pmatrix}\right)$
    return $\alpha, speed$
  end procedure
```

We assume that it always holds $-\pi < \alpha_i < \pi$. This is in fact sensible, since $|\alpha_i| = \pi$ would mean that the curve directly turns around at a point, which again means that the curve speed must be zero at this point, which would contradict the assumption that c is always a *regular* curve. In order to compute an approximate angle function Θ, we sum the local turning angles, yielding

$$\Theta_i \coloneqq \sum_{k=0}^{i} \alpha_k\,,$$

and similarly, we use the velocity approximations to calculate the log speed function Φ as

$$\Phi_i \coloneqq \log(|\dot{c}_i|)\,.$$

D.4.1 Normalising Φ, Θ

Taking into account that Φ, Θ must fulfil $G(\Phi, \Theta) = (1, \pi, 0, 0)$, (Φ, Θ) must be normalised with respect to these conditions before we can use them in further calculations. In order to do so, we use a combination of a normalisation and the procedure to project (Φ, Θ) to $\mathcal{C}_N$. Explicitly, given $(\Phi, \Theta) \in \mathcal{H}_N$, we first find r_1, r_2 so that

$$\int_0^1 e^{\Phi(t) + r_1}\, dt = 1$$

and

$$\int_0^1 (\Theta(t) + r_2)e^{\Phi(t)+r_1}\,dt = \pi$$

in order to fulfil the first two conditions $G_1(\Phi, \Theta) = 1$ and $G_2(\Phi, \Theta) = \pi$. Solving the above equations for r_1 and r_2, respectively, yields

$$e^{r_1} = \frac{1}{\int_0^1 e^{\Phi(t)}\,dt}$$

and

$$r_2 = \frac{\pi - \int_0^1 \Theta(t)e^{\Phi(t)+r_1}\,dt}{\int_0^1 e^{\Phi(t)+r_1}\,dt}\,.$$

Notice that with $(\tilde{\Phi}, \tilde{\Theta}) = (\Phi+r_1, \Theta+r_2)$ we have normalised for only two of four conditions — the closure condition is not necessarily fulfilled. Assuming we are close enough to the shape we actually want to achieve, we then use the projection procedure ProjectToCn from Algorithm 3 and set

$$(\Phi, \Theta) \leftarrow \textsc{ProjectToCn}(\tilde{\Phi}, \tilde{\Theta})\,.$$

Appendix E

View Point Tracking

E.1 Rotations

Rotation matrices are characterised by

$$R\,R^\top = R^\top\,R = I, \quad \det(R) = 1\,,$$

so that if one differentiates the above formula, one finds that

$$\frac{dR}{dt}\,R^\top \text{ and } R^\top\,\frac{dR}{dt}$$

are skew symmetric matrices, so that

$$\frac{dR}{dt} = S_{w_r}\,R = R\,S_{w_l}\,.$$

Let $w_r = (x, y, z)^\top \in \mathbb{R}^3$,

$$S_{w_r} := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}\,.$$

If $R = I$, then $\dot{R}_I = S_{w_r}$. The space of skew symmetric matrices which contains S_{w_r} is isomorphic to $\mathbb{R}^3$ [100].
Say that

$$n = \frac{w_r}{|w_r|}$$

and

$$w_r = \theta\,n\,.$$

Then a rotation R around the axis n and for the angle θ is given by the *Rodrigues formula*

$$R = I + \sin(\theta)\,S_n + (1-\cos(\theta))\,S_n^2 = \cos(\theta)\,I + \sin(\theta)\,S_n + (1-\cos(\theta))\,n\,n^\top\,.$$

θ and S_n can be derived from R with

$$\theta = \arccos\left(\frac{\mathrm{trace}(R) - 1}{2}\right)$$

and

$$S_n = \frac{R - R^\top}{2\,\sin(\theta)}\,.$$

The latter formula exhibits problems around $\theta = \pi$ and $\theta = 0$, which can be tackled with using a Taylor expansion in the latter case; in the former case one needs to apply some more care, see for example [100].

E.1.1 Exponential and Logarithmic Maps

We assume that we look only at the tangent space at identity, $T_{Id}(\mathcal{SO}_3)$. We can do this since we can move any point in $R \in \mathcal{SO}_3$ to identity simply by applying $R^\top$, and move back by applying R.
Say $(\theta, n) \in T_{Id}(\mathcal{SO}_3)$, then the exponential map maps (θ, n) to the rotation around axis n with angle θ. So, $\mathrm{Exp}(\theta, n)$ is given by the Rodrigues formula. This happens to be the same as the matrix exponential $\exp(\theta\, S_n)$ [100]. The logarithmic map can then be calculated by applying the formulas for S_n and θ from above.

E.1.2 Metric

The metric on $\mathcal{SO}_3$ corresponds to the angle travelled from one rotation R_1 to another, R_2:

$$d(R_1, R_2) = \arccos\left(\frac{\mathrm{trace}(R_1^\top\, R_2) - 1}{2}\right)\,.$$

E.2 Unit Sphere

This section notes how to calculate exponential and inverse exponential maps on a unit sphere.
Let $x_1, x_2 \in \mathbb{S}^{n-1} \subset \mathbb{R}^n$, $v \in T_{x_1}(\mathbb{S}^{n-1})$, $\sqrt{v^\top v} = 1$, $\ell \in \mathbb{R}^+$. A geodesic is given by

$$f(x_1, v, t) = \cos(t\,\ell)\,x_1 + \sin(t\,\ell)\,v\,. \tag{E.1}$$

ℓ is the angular distance to the point x_2 that is reached at $t = 1$, $x_2 = f(x_1, v, 1) = \mathrm{Exp}_{x_1}(v\,\ell)$.
Conversely, given $x_1, x_2 \in \mathbb{S}^{n-1}$, one calculates the distance along a great circle as $d(x_1, x_2) = \ell = \arccos(x_1^\top x_2)$. Then,

$$x_2 = \cos(t\,\arccos(x_1^\top x_2))\,x_1 + \sin(t\,\arccos(x_1^\top x_2))\,v\Big|_{t=1} \tag{E.2}$$

$$\Rightarrow v = \frac{x_2 - (x_1^\top x_2)\,x_1}{\sin(\arccos(x_1^\top x_2))} \quad \forall x_1 \neq x_2\,. \tag{E.3}$$

This delivers the exponential and inverse exponential as

$$\operatorname{Exp}_{x_1}(\tilde{v}) = \begin{cases} f(x_1, \frac{\tilde{v}}{|\tilde{v}|}, |\tilde{v}|) & \text{if } \tilde{v} \neq 0 \\ x_1 & \text{else} \end{cases} \tag{E.4}$$

and

$$\operatorname{Log}_{x_1}(x_2) = \begin{cases} \frac{x_2 - (x_1^\top x_2)\, x_1}{\sin(\arccos(x_1^\top x_2))} \arccos(x_1^\top x_2) & \text{if } x_1 \neq x_2 \\ 0 & \text{else.} \end{cases} \tag{E.5}$$

A parallel transport of a tangent vector v from point x_1 to point x_2 is given by

$$v_2 = v - \frac{2\,(x_1 + x_2)\,\langle v, x_2\rangle}{\langle x_1 + x_2, x_1 + x_2\rangle}\,. \tag{E.6}$$

E.3 Motion Model

A solution for the differential equation (5.7)–(5.9) can be obtained with a computer algebra system[1]. Distinguishing three cases, it reads

$$s\,(t) = \frac{1}{m}\, e^{-\frac{b\,t}{2\,m}} \left(\frac{\sin\left(\frac{\sqrt{8\,g\,m^2 - b^2}\,t}{2\,m}\right) (2\,m\,(m\,v_0 - b\,P) + b\,m\,P)}{\sqrt{8\,g\,m^2 - b^2}} - m\,\cos\left(\frac{\sqrt{8\,g\,m^2 - b^2}\,t}{2\,m}\right) P \right) + P \tag{E.7}$$

if $8\,g\,m^2 > \beta^2$,

$$s\,(t) = \frac{1}{m}\, e^{-\frac{b\,t}{2\,m}} \left(\frac{\sinh\left(\frac{\sqrt{b^2 - 8\,g\,m^2}\,t}{2\,m}\right) (2\,m\,(m\,v_0 - b\,P) + b\,m\,P)}{\sqrt{b^2 - 8\,g\,m^2}} - m\,\cosh\left(\frac{\sqrt{b^2 - 8\,g\,m^2}\,t}{2\,m}\right) P \right) + P \tag{E.8}$$

if $\beta^2 > 8\,g\,m^2$, and

$$s\,(t) = e^{-\frac{b\,t}{2\,m}} \left(\frac{t\,(2\,m\,(m\,v_0 - b\,P) + b\,m\,P)}{2\,m^2} - P \right) + P \tag{E.9}$$

if $8\,g\,m^2 = \beta^2$.

[1] This solution was computed with Maxima, which is Free Software and can be acquired at http://maxima.sourceforge.net.

We will also need to calculate the velocity $\dot{s}(t)$, which is then

$$\begin{aligned}\dot{s}(t) = \frac{1}{\sqrt{8\,g\,m^2-b^2}}\,e^{-\frac{b\,t}{2\,m}}\left(4\,g\,m\,\sin\left(\frac{\sqrt{8\,g\,m^2-b^2}\,t}{2\,m}\right)P-\right.\\ b\,\sin\left(\frac{\sqrt{8\,g\,m^2-b^2}\,t}{2\,m}\right)v_0+\\ \left.\sqrt{8\,g\,m^2-b^2}\,\cos\left(\frac{\sqrt{8\,g\,m^2-b^2}\,t}{2\,m}\right)v_0\right),\end{aligned} \tag{E.10}$$

if $8\,g\,m^2 > \beta^2$,

$$\begin{aligned}\dot{s}(t) = \frac{1}{\sqrt{b^2-8\,g\,m^2}}\,e^{-\frac{b\,t}{2\,m}}\left(4\,g\,m\,\sinh\left(\frac{\sqrt{b^2-8\,g\,m^2}\,t}{2\,m}\right)P-\right.\\ b\,\sinh\left(\frac{\sqrt{b^2-8\,g\,m^2}\,t}{2\,m}\right)v_0+\\ \left.\sqrt{b^2-8\,g\,m^2}\,\cosh\left(\frac{\sqrt{b^2-8\,g\,m^2}\,t}{2\,m}\right)v_0\right),\end{aligned} \tag{E.11}$$

if $\beta^2 > 8\,g\,m^2$, and

$$\dot{s}(t) = \frac{e^{-\frac{b\,t}{2\,m}}\,\left(b^2\,t\,P+\left(4\,m^2-2\,b\,m\,t\right)\,v_0\right)}{4\,m^2} \tag{E.12}$$

if $8\,g\,m^2 = \beta^2$.

Appendix F

Used Objects

F.1 Object Models and Shapes

3D computer graphics models were used to generate images from object. The models were taken from the Princeton 3D shape benchmark data base [103] and are shown in Figure F.1.
Some shapes from the MPEG-7 core experiment 1 shape data base were used, see e.g. [74]. The fish shapes in Chapters 2 and 5 are from the Surrey fish data base [94].

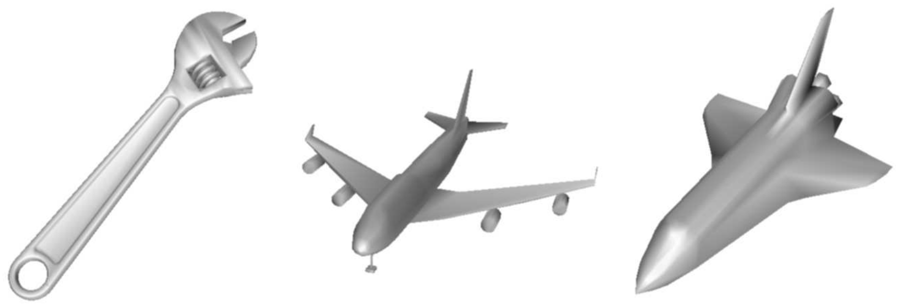


Figure F.1: Objects m1105, m1154, m1249 from left to right. From the Princeton 3D shape benchmark data base [103].

The spherical positions used to sample object images were pre-calculated using a 3D modelling program. A sphere was approximated by a refined icosahedron yielding 162 vertices.

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre.
Optimization Algorithms on Matrix Manifolds.
Princeton University Press, Princeton, NJ, January 2008.
126, 127

[2] O. Alexandrov and F. Santosa.
A topology-preserving level set method for shape optimization, May 2004.
Comment: 10 pages, 4 figures.
97

[3] Ali Shokoufandeh, Lars Bretzner, Diego Macrini, M. Fatih Demirci, Clas Jonsson, and Sven Dickinson.
The representation and matching of categorical shape.
Computer Vision and Image Understanding, 103:139–154, 2006.
19

[4] Anuj Srivastava, Shantanu H. Joshi, Washington Mio, and Xiuwen Liu.
Statistical Shape Analysis: Clustering, Learning, and Testing.
PAMI, 27(4):590–602, Apr. 2005.
29

[5] E. Begelfor and M. Werman.
Affine invariance revisited.
In *CVPR*, pages 2087–2094. IEEE Computer Society, 2006.
25

[6] M. Belkin and P. Niyogi.
Laplacian eigenmaps for dimensionality reduction and data representation.
Neural Computation, 15(6):1373–1396, 2003.
16

[7] C. M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.
26, 108

[8] T. Brox, B. Rosenhahn, and J. Weickert.

Three-dimensional shape knowledge for joint image segmentation and pose estimation, 2005.
15, 16

[9] T. Brox, M. Rousson, R. Deriche, and J. Weickert.
Unsupervised segmentation incorporating colour, texture, and motion.
In *INRIA*, 2003.
67, 68, 69, 70

[10] H. H. Bülthoff and S. Edelman.
Psychophysical support for a 2-D view interpolation theory of object recognition.
Proceedings of the National Academy of Science, 89:60–64, 1992.
14, 15

[11] H. H. Bülthoff, S. Edelman, and M. Tarr.
How are three-dimensional objects represented in the brain?
Cerebral Cortex, 5:247–260, 1995.
14

[12] C. S. Burrus, R. A. Gopinath, and H. Guo.
Introduction to wavelets and wavelet transforms: a primer.
Prentice Hall, 1998.
With additional material and programs by Jan E. Odegard and Ivan W. Selesnick.
71

[13] E. F. Camacho and C. Bordons.
Model predictive control.
Springer, London ; Berlin ; Heidelberg [u.a.], 2004.
113, 148

[14] V. Caselles, F. Catté, T. Coll, and F. Dibos.
A geometric model for active contours in image processing.
Numerische Mathematik, 66(1):1–31, Dec. 1993.
16, 58

[15] V. Caselles, R. Kimmel, and G. Sapiro.
Geodesic active contours.
In *ICCV*, pages 694–699, 1995.
16, 49, 58

[16] T. F. Chan and S. Esedoḡlu.
Aspects of total variation regularized L^1 function approximation.
SIAM Journal on Applied Mathematics, 65(5):1817–1837, 2005.
161

[17] T. F. Chan, S. Esedoglu, and M. Nikolova.
Algorithms for finding global minimizers of image segmentation and denoising models.

SIAM Journal of Applied Mathematics, 66(5):1632–1648, 2006.
51, 61, 121, 161, 162, 163

[18] T. F. Chan and J. Shen.
Morphologically invariant PDE inpaintings, May 2001.
51, 52

[19] Y. Chen, H. D. Tagare, S. Thiruvenkadam, F. Huang, D. Wilson, K. S. Gopinath, R. W. Briggs, and E. A. Geiser.
Using prior shapes in geometric active contours in a variational framework.
International Journal of Computer Vision, 50(3):315–328, Dec. 2002.
16, 79, 80

[20] Chen Sagiv, Nir A. Sochen, and Yehoshua Y. Zeevi.
Integrated active contours for texture segmentation.
IEEE TRANSACTIONS ON IMAGE PROCESSING, 15(6):1633–1646, June 2006.
73

[21] Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D. Fox.
Level Set Evolution Without Re-Initialization: A New Variational Formulation.
CVPR, 2005.
Level-Sets.
77, 91

[22] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham.
Active shape models: Their training and application.
Computer Vision and Image Understanding, 61(1):38–59, Jan. 1995.
20

[23] R. Courant, K. Friedrichs, and H. Lewy.
Über die partiellen Differenzengleichungen der mathematischen Physik.
Mathematische Annalen, 100(1):32–74, Dec. 1928.
158

[24] D. Crandall, P. Felzenszwalb, and D. Huttenlocher.
Spatial priors for part-based recognition using statistical models.
Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 1:10–17, 2005.
19

[25] D. Cremers, T. Kohlberger, and C. Schnörr.
Nonlinear shape statistics via kernel spaces.
In B. Radig and S. Florczyk, editors, *Pattern Recognition (Proc. DAGM)*, volume 2191 of *LNCS*, pages 269–276, Munich, Germany, Sept. 2001. Springer.

80

[26] D. Cremers, T. Kohlberger, and C. Schnörr.
Shape statistics in kernel space for variational image segmentation.
Pattern Recognition, 36(9):1929–1943, September 2003.
16, 80

[27] D. Cremers, M. Rousson, and R. Deriche.
A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape.
International Journal of Computer Vision, 72(2):195–215, Apr. 2007.
67

[28] D. Cremers, C. Schnörr, J. Weickert, and C. Schellewald.
Diffusion Snakes using statistical shape knowledge.
In C. Sommer and Y. Zeevi, editors, *Algebraic Frames for the Perception-Action Cycle*, volume 1888 of *LNCS*, pages 164–174, Kiel, Germany, Sept. 2000. Springer.
16, 80

[29] D. Cremers and S. Soatto.
A pseudo-distance for shape priors in level set segmentation.
In N. Paragios, editor, *IEEE 2nd Int. Workshop on Variational, Geometric and Level Set Methods*, pages 169–176, Nice, 2003.
81

[30] Daniel Cremers.
Statistical Shape Knowledge in Variational Image Segmentation.
PhD thesis, University of Mannheim, 2002.
shape.
16, 28, 79, 80

[31] B. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi.
Population shape regression from random design data.
In *International Conference on Computer Vision*, 2007.
108

[32] H. Delingette and J. Montagnat.
Shape and topology constraints on parametric active contours.
Computer Vision and Image Understanding, 83(2):140–171, Aug. 2001.
50, 56, 61, 73, 75, 132, 134

[33] M. P. do Carmo.
Riemannian Geometry.
Birkhäuser, 1992.
39, 102, 113

[34] M. P. do Carmo.
Differentialgeometrie von Kurven und Flächen.
Vieweg, Braunschweig ; Wiesbaden, 1998.

39, 40

[35] S. Edelman and H. Bülthoff.
Viewpoint-specific representations in three-dimensional object recognition.
Artificial Intelligence Memo, AIM-1239, August 1990.
14, 15

[36] S. Edelman and H. Bülthoff.
Modeling human visual object recognition.
Neural Networks, 1992. IJCNN., International Joint Conference on, 4:37–42 vol.4, Jun 1992.
14

[37] S. Edelman and D. Weinshall.
A self-organizing multiple-view representation of 3d objects.
Biological Cybernetics, 64(3):209–219, Jan. 1991.
14

[38] Eric Klassen, Anuj Srivastava, Washington Mio, and Shantanu H. Joshi.
Analysis of Planar Shapes Using Geodesic Paths on Shape Spaces.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(3), 2004.
29, 30, 41

[39] P. Etyngier, F. Segonne, and R. Keriven.
Shape priors using manifold learning techniques.
In *International Conference on Computer Vision*, 2007.
16

[40] D. Farin and P. H. N. de With.
Shortest circular paths on planar graphs.
In *27th Symposium on Information Theory in the Benelux*, 2006.
26

[41] M. F. A. Fauzi and P. H. Lewis.
A fully unsupervised texture segmentation algorithm.
In *British Machine Vision Conference*, pages xx–yy, 2003.
71

[42] P. T. Fletcher, C. Lu, S. M. Pizer, and S. C. Joshi.
Principal geodesic analysis for the study of nonlinear statistics of shape.
IEEE Trans. Med. Imaging, 23(8):995–1005, 2004.
39

[43] K. Fundana, A. Heyden, C. Gosch, and C. Schnörr.
Continuous graph cuts for prior-based object segmentation.
In *ICPR*, 2008.
165

[44] A. Gelas, O. Bernard, D. Friboulet, and R. Prost.
Compactly supported radial basis functions based collocation method for level-set evolution in image segmentation.
Image Processing, IEEE Transactions on, 16(7):1873–1887, 2007.
92

[45] D. Gennery.
Visual tracking of known three-dimensional objects.
Int. J. Computer Vision, 7(3), 1992.
15

[46] Gilles Aubert and Pierre Kornprobst.
Mathematical Problems in Image Processing, volume 147 of *Applied Mathematical Sciences*.
Springer, 2002.
59

[47] G. H. Golub and C. F. V. Loan.
Matrix computations.
Johns Hopkins Univ. Pr., Baltimore, Md. [u.a.], 3. ed. edition, 1996.
36, 37

[48] C. R. Goodall.
Procrustes methods in the statistical analysis of shape (with discussion).
J. R. Statist. Soc. Ser. B, 53:285–339, 1991.
21

[49] J. Gower.
Generalized procrustes analysis.
Psychometrika, 40(1):33–51, March 1975.
26

[50] W. Greiner.
Mechanik, volume 1.
Deutsch, 1993.
113

[51] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson.
Skeleton based shape matching and retrieval.
In *Shape Modelling and Applications Conference*, 2003.
19

[52] X. Han, C. Xu, and J. L. Prince.
A topology preserving level set method for geometric deformable models.
IEEE Trans. Pattern Anal. Mach. Intell, 25(6):755–768, 2003.
97

[53] X. Han, C. Xu, D. Tosun, and J. L. Prince.

Cortical surface reconstruction using a topology preserving geometric deformable model.
In *Workshop on Mathematical Methods in Biomedical Image Analysis*, pages xx–yy, 2001.
97

[54] W. Hayward, M. Tarr, and A. Corderoy.
Recognizing silhouettes and shaded images across depth rotation.
Perception, 28:1197–1215, 1999.
14, 17

[55] W. G. Hayward.
Effects of outline shape in object recognition.
Journal of Experimental Psychology, 1998.
14

[56] W. G. Hayward, A. C.-N. Wong, and B. Spehar.
When are viewpoint costs greater for silhouettes than shaded images.
Psychonomic Bulletin and Review, 12:321–327, 2005.
14, 17

[57] Ian L. Dryden and Kanti V. Mardia.
Statistical Shape Analysis.
Wiley, Mar. 1999.
Book.
20, 21, 22, 26, 79

[58] S. Joshi, S. Pizer, P. T. Fletcher, P. Yushkevich, A. Thall, and J. S. Marron.
Multiscale deformable model segmentation and statistical shape analysis using medial descriptions.
IEEE Trans. Medical Imaging, 21(5):538–550, May 2002.
19

[59] S. Joshi, A. Srivastava, E. Klassen, and I. Jermyn.
Removing shape-preserving transformations in square-root elastic (sre) framework for shape analysis of curves.
In *Workshop on Energy Minimization Methods in CVPR (EMM-CVPR)*, 2007.
47

[60] S. H. Joshi, E. Klassen, A. Srivastava, and I. Jermyn.
An efficient representation for computing geodesics between n-dimensional elastic shapes.
In *CVPR*, 2007.
45, 46, 47

[61] S. H. Joshi and A. Srivastava.

Intrinsic bayesian active contours for extraction of object boundaries in images.
In *Asian Conference on Computer Vision*, 2006.
79, 130, 154

[62] H. Karcher.
Riemannian center of mass and mollifier smoothing.
Communications on Pure and Applied Mathematics, 30(5):509–541, 1977.
102, 103

[63] M. Kass, A. Witkin, and D. Terzopoulos.
Snakes: Active contour models.
Int'l J. Comp. Vision, 1(4):321–331, 1988.
16, 49, 57, 58

[64] D. G. Kendall.
The diffusion of shape.
Advances in Applied Probability, 9:428–430, 1977.
20

[65] D. G. Kendall.
Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces.
Bull. London Math. Soc., 16(2):81–121, 1984.
21

[66] D. G. Kendall.
Shape and shape theory.
Wiley, Chichester ; Weinheim [u.a.], 1999.
20

[67] W. S. Kendall.
Probability, Convexity, and Harmonic Maps with Small Image I: Uniqueness and Fine Existence.
Proc. London Math. Soc., s3-61(2):371–406, 1990.
103

[68] R. Kimmel.
Fast edge integration, April 2003.
73

[69] R. Kimmel and A. Bruckstein.
Regularized laplacian zero crossings as optimal edge integrators.
International Journal of Computer Vision, 53(3):225–243, July 2003.
73

[70] E. Klassen and A. Srivastava.
Geodesics between 3d closed curves using path-straightening.
In *ECCV (1)*, pages 95–106, 2006.

46

[71] D. Koller, K. Daniilidis, and H. Nagel.
Model-based object tracking in monocular image sequences of road traffic scenes.
IJCV, 10(3):257–281, June 1993.
15

[72] H. Kollnig and H.-H. Nagel.
3d pose estimation by directly matching polyhedral models to gray value gradients.
International Journal of Computer Vision, 23(3):283–302, June 1997.
15

[73] W. Kühnel.
Differentialgeometrie : Kurven, Flächen, Mannigfaltigkeiten.
Vieweg, Braunschweig ; Wiesbaden, 3. edition, 2005.
33, 34, 39, 40, 58

[74] L. Latecki, R. Lakämper, and U. Eckhardt.
Shape descriptors for non-rigid shapes with a single closed contour.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-00)*, pages 424–429, Los Alamitos, June 13–15 2000. IEEE.
179

[75] C. Le Guyader and L. A. Vese.
Self-repelling snakes for topology-preserving segmentation models.
IEEE Transactions on Image Processing, 17(5):767–779, 2008.
97, 98, 99

[76] F. Leitner and P. Cinquin.
Dynamic segmentation: Detecting complex topology 3d objects.
In *Engineering in Medicine and Biology Society*, volume 13, 1991.
56

[77] V. Lepetit and P. Fua.
Monocular model-based 3D tracking of rigid objects: A survey.
Foundations and Trends in Computer Graphics and Vision, 1(1), 2005.
16

[78] M. Leventon, O. Faugeraus, and W. Grimson.
Level set based segmentation with intensity and curvature priors.
In *Workshop on Mathematical Methods in Biomedical Image Analysis Proceedings*, pages 4–11, June 2000.
16, 79

[79] M. Leventon, W. Grimson, and O. Faugeras.
Statistical shape influence in geodesic active contours.
pages 316–323.

79, 80

[80] F. L. Lewis.
Optimal Control.
John Wiley & Sons, 1986.
113

[81] S. Lloyd.
Least squares quantization in pcm.
Information Theory, IEEE Transactions on, 28(2):129–137, 1982.
26

[82] T. Lloyd-Jones and L. Luckhurst.
Outline shape is a mediator of object recognition that is particularly important for living things.
Memory & Cognition, 30:489–498(10), 1 June 2002.
14

[83] Luminita A. Vese and Tony F. Chan.
A multiphase level set framework for image segmentation using the mumford and shah model.
International Journal of Computer Vision, 50(3):271–293, 2002.
66

[84] P. C. Mahalanobis.
On the generalized distance in statistics.
Natl. Inst. Science, 12:49–55, 1936.
80

[85] S. Mallat.
A Wavelet Tour of Signal Processing.
AP Professional, London, 1997.
71

[86] Marie Rochery, Ian Jermyn, and Josiane Zerubia.
Higher order active contours and their application to the detection of line networks in satellite imagery.
VLSM Workshop, 2003.
97

[87] T. McInerney and D. Terzopoulos.
Topologically adaptable snakes.
In *International Conference on Computer Vision*, pages 840–845, 1995.
56

[88] P. W. Michor and D. Mumford.
Riemannian geometries on spaces of plane curves, 2003.
Comment: amslatex, 45 pagex, 8 figures, typos corrected.
33, 99

[89] P. W. Michor, D. Mumford, J. Shah, and L. Younes.
A Metric on Shape Space with Explicit Geodesics.
ArXiv e-prints, 706, June 2007.
33

[90] W. Mio and A. Srivastava.
Elastic-string models for representation and analysis of planar shapes.
In *CVPR (2)*, pages 10–15, 2004.
29, 30, 33, 42

[91] W. Mio, A. Srivastava, and S. Joshi.
On shape of plane elastic curves.
International Journal of Computer Vision, 73(3):307–324, 2007.
25, 29, 30, 31, 32, 33, 41, 43, 45, 46

[92] W. Mio, A. Srivastava, and X. Liu.
Contour inferences for image understanding.
International Journal of Computer Vision, 69(1):137–144, Aug. 2006.
29

[93] M. Moelich and T. Chan.
Tracking objects with the chan-vese algorithm.
Technical report, June 11 2004.
16, 127, 129

[94] F. Mokhtarian, S. Abbasi, and J. Kittler.
Efficient and robust retrieval by shape content through curvature scale space, October 1996.
44, 109, 179

[95] D. Mumford and J. Shah.
Optimal approximation by piecewise smooth functions and associated variational problems.
Communications on Pure Applied Mathematics, 42:577–685, 1989.
59, 161

[96] N. U. Optimization Technology Center.
The neos guide, http://www-fp.mcs.anl.gov/otc/guide/.
http://www-fp.mcs.anl.gov/OTC/Guide/.
123

[97] S. Osher and J. Sethian.
Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations.
Journal of Computational Physics, 79:12–49, 1988.
16, 49, 50, 57

[98] N. Paragios, M. Rousson, and V. Ramesh.
Non-rigid registration using distance functions.
89(2–3):142–165, Feb./Mar. 2003.

79

[99] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang.
A PDE-based fast local level set method.
Technical report, University of California at Los Angeles, 1999.
87

[100] X. Pennec.
L'incertitude dans les problèmes de reconnaissance et de recalage – Applications en imagerie médicale et biologie moléculaire.
Thèse de sciences (phd thesis), Ecole Polytechnique, Palaiseau (France), December 1996.
175, 176

[101] X. Pennec.
Probabilities and statistics on riemannian manifolds: Basic tools for geometric measurements.
NSIP, 1999.
102

[102] K. B. Petersen and M. S. Pedersen.
The matrix cookbook, 2005.
Version 20051003.
167

[103] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser.
The princeton shape benchmark.
In *Shape Modeling International*, 2004.
105, 179

[104] G. Picinbono.
Modèle géométrique de contour déformable implicite pour la segmentation et la simulation.
Master's thesis, université de Nice-Sophia Antipolis, 1997.
74, 75

[105] T. Poggio and S. Edelman.
A network that learns to recognize three-dimensional objects.
Nature, 343(6255):263–266, Jan. 1990.
14

[106] T. Riklin-Raviv, N. Kiryati, and N. A. Sochen.
Unlevel-sets: Geometry and prior-based segmentation.
In T. Pajdla and J. Matas, editors, *ECCV (4)*, volume 3024 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2004.
79, 80, 127

[107] T. Riklin-Raviv, N. Sochen, and N. Kiryati.
Shape-based mutual segmentation.

International Journal of Computer Vision, 79(3):231–245, Sept. 2008.
81

[108] B. Rosenhahn, T. Brox, and J. Weickert.
Three-dimensional shape knowledge for joint image segmentation and pose tracking.
Int. J. Comput. Vision, 73(3):243–262, 2007.
15, 16

[109] M. Rousson and R. Deriche.
A variational framework for active and adaptative segmentation of vector valued images.
In *In Proc. IEEE Workshop on Motion and Video Computing*, pages 56–62, 2002.
67, 86

[110] M. Rousson and N. Paragios.
Prior knowledge, level set representations and visual grouping.
International Journal of Computer Vision, 76:231, 2008.
85

[111] E. Rouy and A. Tourin.
A viscosity solutions approach to shape-from-shading.
SIAM J. Numer. Anal., 29(3):867–884, 1992.
95

[112] L. I. Rudin, S. Osher, and E. Fatemi.
Nonlinear total variation based noise removal algorithms.
Phys. D, 60(1-4):259–268, 1992.
161, 162

[113] B. Sandberg, T. Chan, and L. A. Vese.
A level-set and gabor-based active contour algorithm for segmenting textured images.
CAM report 02-39, July 2002.
66

[114] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer.
Region-based pose tracking.
Technical Report 196, Saarland University, 2007.
15, 16, 140

[115] F. R. Schmidt, D. Farin, and D. Cremers.
Fast matching of planar shapes in sub-cubic runtime.
In *International Conference on Computer Vision*, 2007.
26

[116] P. Schönemann.
A generalized solution of the orthogonal procrustes problem.

Psychometrika, 31(1):1–10, Mar. 1966.
21

[117] P. Schönemann and R. Carroll.
Fitting one matrix to another under choice of a central dilation and a rigid motion.
Psychometrika, 35(2):245–255, June 1970.
21

[118] T. Sebastian, P. Klein, and B. Kimia.
On aligning curves.
PAMI, 25(1):116–125, 2003.
25, 26, 42, 43, 45

[119] J. E. Solem.
Geodesic curves for analysis of continuous implicit shapes.
In *International Conference on Pattern Recognition*, pages 43–46, 2006.
130

[120] Stanley Osher and Ronald Fedkiw.
Level Set Methods and Dynamic Implicit Surfaces.
Number 153 in Applied Mathematical Sciences. Springer, 2003.
54, 56, 57, 77, 87, 91, 93, 95

[121] K. Stark and S. Fuchs.
A method for tracking the pose of known 3-D objects based on an active contour model.
In *International Conference on Pattern Recognition*, pages I: 905–909, 1996.
15

[122] G. Strang.
l^1 and l^∞ approximation of vector fields in the plane.
In H. Fujita, P. Lax, and G. Strang, editors, *Nonlinear Partial Differential Equations in Applied Science*, Lecture Notes in Num. Appl. Anal., pages 273–288. 1982.
161

[123] G. Strang.
Maximal flow through a domain.
Mathematical Programming, 26(2):123–143, June 1983.
161

[124] G. Sundaramoorthi, A. J. Yezzi, and A. Mennucci.
Sobolev active contours.
In N. Paragios, O. D. Faugeras, T. Chan, and C. Schnörr, editors, *VLSM*, volume 3752 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2005.
99, 129

[125] G. Sundaramoorthi and A. J. Yezzi, Jr.
More-than-topology-preserving flows for active contours and polygons.
In *International Conference on Computer Vision*, pages II: 1276–1283, 2005.
97

[126] G. Sundaramoorthi and A. J. Yezzi, Jr.
Global regularizing flows with topology preservation for active contours and polygons.
IEEE Trans. Image Processing, 16(3):803–812, Mar. 2007.
97

[127] M. Sussman and E. Fatemi.
An efficient, interface preserving level set re-distancing algorithm and its application to interfacial incompressible fluid flow.
89

[128] M. Sussman, P. Smereka, and S. Osher.
A level set approach for computing solutions to incompressible two-phase flow.
Journal of Computational Physics, 114:146–159, 1994.
87

[129] R. Szeliski, D. Tonnesen, and D. Terzopoulos.
Modeling surfaces of arbitrary topology with dynamic particles.
In *Transactions of the IEEE conference on Computer Vision and Pattern Recognition*. New York City, NY, June 1993.
56

[130] T.F. Chan, B.Y. Sandberg, and L.A. Vese.
Active contours without edges for vector-valued images.
Journal of Visual Communication and Image Representation, 11(2):130–141, 2000.
16, 52, 61, 66, 67

[131] Tony F. Chan and Luminita A. Vese.
Active Contours Without Edges.
IEEE Transactions on image processing, 10(2), Feb. 2001.
16, 59, 61, 66, 81, 83, 84, 140, 161, 162, 164

[132] A. Tsai, A. J. Yezzi, W. M. Wells, III, C. Tempany, D. Tucker, A. Fan, W. E. L. Grimson, and A. S. Willsky.
A shape-based approach to the segmentation of medical imagery using level sets.
IEEE Trans. Medical Imaging, 22(2):137–154, Feb. 2003.
80

[133] A. Tveito and R. Winther.

Introduction to Partial Differential Equations, A Computational Approach.
Springer, 2000.
157

[134] S. Ullman and R. Basri.
Recognition by linear combinations of models.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 13(10):992–1006, Oct 1991.
14

[135] Washington Mio, Anuj Srivastava, and Xiuwen Liu.
Learning and Bayesian Shape Extraction for Object Recognition.
ECCV, 2004.
41

[136] M. Werman and D. Weinshall.
Similarity and affine invariant distances between 2D point sets.
IEEE Trans. Pattern Anal. Mach. Intell, 17(8):810–814, 1995.
24, 25

[137] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling.
Numerical Recipes In C.
Cambridge University Press, 2002.
93

[138] Xavier Bresson, Pierre Vandergheynst, and Jean-Philippe Thiran.
A variational model for object segmentation using boundary information and shape prior driven by the mumford-shah functional.
International Journal of Computer Vision, 68(2):145, 2006.
73, 79, 80

[139] Xavier Pennec.
Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements.
Journal of Mathematical Imaging and Vision, 2006.
102, 103, 104, 105

[140] Xavier Pennec, Pierre Fillard, and Nicholas Ayache.
A riemannian framework for tensor computing.
Technical Report 5255, INRIA, 2004.
39

[141] A. J. Yezzi and A. Mennucci.
Conformal metrics and true "gradient flows" for curves.
In *ICCV*, pages 913–919. IEEE Computer Society, 2005.
99

[142] A. J. Yezzi and S. Soatto.

Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images.
International Journal of Computer Vision, 53(2):153–167, July 2003.
129

[143] L. Younes.
Computable elastic distances between shapes.
SIAM Journal on Applied Mathematics, 58(2):565–586, 1998.
30, 33

[144] Yunmei Chen, Feng Huang, Hemant D. Tagare, and Murali Rao.
A coupled minimization problem for medical image segmentation with priors.
IJCV, 71(3):259–272, 2007.
79