# Efficient Methods for Continuous and Discrete Shape Analysis

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Dipl.-Math. Frank R. Schmidt
aus Bonn

Bonn, 2010

Angefertigt mit Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. D. Cremers

2. Gutachter: Prof. Dr. M. Clausen

Tag der Promotion: 12. November 2010

# Summary

When interpreting an image of a given object, humans are able to abstract from the presented color information in order to really *see* the presented object. This abstraction is also known as *shape*. The concept of shape is not defined exactly in Computer Vision and in this work, we use three different forms of these definitions in order to acquire and analyze shapes. This work is devoted to improve the efficiency of methods that solve important applications of *shape analysis*.

The most important problem in order to analyze shapes is the problem of *shape acquisition*. To simplify this very challenging problem, numerous researchers have incorporated prior knowledge into the acquisition of shapes. We will present the first approach to acquire shapes given a certain shape knowledge that computes always the global minimum of the involved functional which incorporates a Mumford-Shah like functional with a certain class of shape priors including *statistic shape prior* and *dynamical shape prior*.

In order to analyze shapes, it is not only important to acquire shapes, but also to classify shapes. In this work, we follow the concept of defining a distance function that measures the dissimilarity of two given shapes. There are two different ways of obtaining such a distance function that we address in this work. Firstly, we model the set of all shapes as a metric space induced by the shortest path on

an orbifold. The shortest path will provide us with a *shape morphing*, i.e., a continuous transformation from one shape into another. Secondly, we address the problem of *shape matching* that finds corresponding points on two shapes with respect to a preselected *feature*.

Our main contribution for the problem of *shape morphing* lies in the immense acceleration of the morphing computation. Instead of solving *partial* resp. *ordinary differential equations*, we are able to solve this problem via a gradient descent approach that subsequently shortens the length of a path on the given manifold. During our run-time test, we observed a run-time acceleration of up to a factor of 1000.

*Shape matching* is a classical discrete problem. If each shape is discretized by $N$ shape points, most Computer Vision methods needed a cubic run-time. We will provide two approaches how to reduce this worst-case complexity to $O(N^2 \log(N))$. One approach exploits the planarity of the involved graph in order to efficiently compute $N$ shortest path in a graph of $O(N^2)$ vertices. The other approach computes a minimal cut in a planar graph in $O(N \log(N))$. In order to make this approach applicable to shape matching, we improved the run-time of a recently developed graph cut approach by an empirical factor of 2–4.

# Acknowledgments

First of all, I would like to express my gratitude to Prof. D. Cremers for supervising my dissertation and for introducing me into the field of Computer Vision. Both the wide range of research topics discussed in his group and the amount of time that Prof. D. Cremers spend with me personally, discussing different approaches and techniques provided a stimulating and productive environment for my thesis. Secondly, I want to thank Prof. M. Clausen for serving as an external referee for my thesis. Furthermore, I want to thank the committee members of the disputation, namely Prof. D. Cremers, Prof. M. Clausen, Prof. R. Klein and Prof. M. Rumpf.

I want to thank a number of people who contributed to this work. In particular there is E. Töppe and F. Barthel who collaborated with me during their diploma thesis, the graph cut implementation for shape matching and parts of the presented shape acquisition method evolved from joint work. I want to thank all the other members of our group for providing a unique atmosphere, namely T. Brox, T. Pock, T. Schoenemann, K. Kolev and U. Schlickewei. In particular, I want to thank B. Goldlücke, C. Nieuwenhuis, F. Steinbrücker, M. Klodt, E. Töppe and M. Oswald for proofreading the manuscripts and for providing helpful comments.

Then there is the group of M. Clausen with whom I conducted very fruitful discussions. Especially I like to thank F. Kurth, M. Müller,

R. Bardeli and T. Röder.

A special thank goes to members of the *Hausdorff Center for Mathematics* in Bonn. In particular, I thank S. Hainz, J. Müller, B. Berkels and B. Wirth for the discussions on several mathematical topics.

I want to thank a number of researchers for hospitality and stimulating discussions during various visits at other institutes. First of all this is the vision group at the University of Western Ontario who hosted me at two stays of one week and one month. In particular I want to thank Y. Boykov, O. Veksler and A. Delong for the great time and the intense discussion that we conducted. Then, there is the Technical University at Eindhoven who hosted me for a stay of two weeks. I enjoyed the collaboration with D. Farin very much and I profited immensely from his experience in the fields of real-time programming.

A great thanks goes to the organizers of the *SIAM Conference of Imaging Science 2008* for letting me present the convex shape acquisition method. I also like to thank IPAM for hosting me at their workshop on *Graph Cuts and Related Discrete or Continuous Optimization Problems*. I also would like to thank the image group at the DIKU in Copenhagen for their organization of the summer school 2008.

Finally, I would like to thank my parents, my brother and my sister for their invaluable support of my research.

# Contents

# Chapter 1

# Introduction

## 1.1 Shapes

Describing, measuring and comparing the shape of objects is very popular in a variety of different disciplines. Even one of the first documented form of human communication, namely the cave paintings during the stone age, reduced the described objects to a monochromatic shape[1]. Since then, humans strove to improve the visualization of the objects that they encountered. But even in modern society, the shape representation in form of a paper-cutting is still present. It reflects the fact that every person is able to recognize a given object even if color information is completely ignored. In fact, it has been believed that all relevant features of a given object are encoded by its shape. If we want to teach a computer not only to register but to truly see and interpret the world, Shape Analysis is an important task. In this work, we will discuss different aspects of Shape Analysis and present methods that to our knowledge are among the fastest in

[1] The cave painting image and the paper-cutting image of Figure 1.1 are taken from wikipedia.org and have been released to the public domain.



**Figure 1.1: Shapes and human perception.** Humans have always used shapes to describe their environment. *Left:* Cave drawing of the Stone Age. *Middle:* Paper-cutting from the silhouette of *Johann Wolfgang von Goethe* from the late 18th century. *Right:* Shape of the MPEG7-database.

these different application areas.

Before analyzing different shapes, we have to acquire a shape from a given image. This means, we have to distinguish a displayed object from its surroundings. In other words, we like to assign to every point $x$ of the image domain $\Omega \subset \mathbb{R}^2$ a binary value $\ell(x)$ telling whether the point $x$ is part of the object ($\ell(x) = 1$) or whether it is part of the background ($\ell(x) = 0$)). This binary labeling problem is also known as image segmentation because we segment an image into its semantical components. At the end of this segmentation process, we obtain the shape of a given object. Hence, every shape can be equivalently represented by either a mapping $\ell : \Omega \to \{0, 1\}$ or by a subset $S$ of the image domain $\Omega$.

In this work, we assume that the set $S \subset \Omega$ that represents a shape fulfills certain regularity conditions. Therefore, we assume that a shape can be represented by an open, connected subset $S$ of a compact image domain $\Omega$ such that the boundary $\partial S$ is a smooth one-dimensional manifold. Every such set $S$ can therefore be represented as a collection of closed contours $\{c_0, \ldots, c_k\}$. Furthermore, we assume that $c_0$ represents the outer contour of the subset $S$. Most

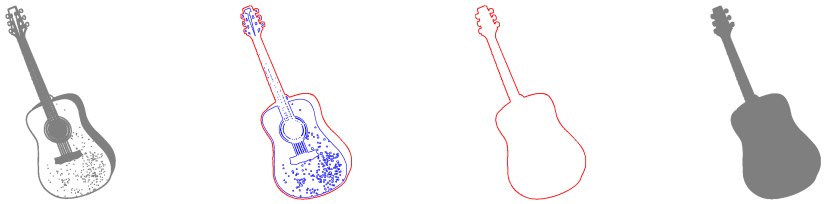


**Figure 1.2: Shapes representation.** A general shape (*left*) can be represented as multiple contours (*second from the left*). Out of simplicity, we will often only use the outer contour which is the most descriptive one (*second from the right*). Therefore, in fact a simplified shape is considered (*right*).

of the approaches, that we will present in this work, are focused on this outer contour. The other contours $c_1, \ldots, c_k$ have either a small contribution to the methods or are merely ignored. This is due to the observation that for most objects, the outer contour is the most descriptive one (cf. Figure 1.2).

Note that even the discussed representation is not unique in the sense that the shape of an object will not change if we move, rotate or scale the given object. Depending on the given application, we will therefore introduce a group operation on the set of shape representations. A *shape* will then be described as the equivalence class of a shape representation with respect to the predefined group operation.

One goal of this work is it to describe shapes in a way that the set of all different shapes forms a space equipped with a distance function. This means we have to define a function that measures the similarity of two given shapes. A value close to zero would imply that the two given shapes are very similar to one another. On the other hand, large distances should imply that the two given shapes are easy to distinguish and therefore, they are very dissimilar to one another. The concept of measuring the similarity of two different shapes ap-

pears in every chapter of this work and influences the meaning of shape similarity.

## 1.2 Continuous versus Discrete Methods

In this work, we will only address problems that can be formulated by means of optimizing a given cost functional. This formulation can be given as a continuous functional or a discrete function. As a consequence, contours can either be represented in a continuous or a discrete sense. In fact, there are certain problems that can be handled more efficiently if we choose a continuous representation. But there are also problems that benefit from a discrete representation.

Continuous representations of contours can either be explicit or implicit. The implicit representation of a contour consists of a differentiable mapping[2]

$$\varphi : \Omega \to \mathbb{R}$$

that assigns a real value to every point of the image domain $\Omega$. The contour is then defined by all points $x \in \Omega$ to which $\varphi$ assigns the value *zero*:

$$C = \{x \in \Omega | \varphi(x) = 0\}$$

Especially for image segmentation, this representation comes in very handy. But for other problems, explicit representations are much easier to handle. The explicit representation of a continuous contour is given as a differentiable mapping

$$c : \mathbb{S}^1 \to \Omega$$

that assigns a point of the image domain $\Omega$ to every point of the parameterizing circle $\mathbb{S}^1$. Therefore the contour $C \subset \Omega$ is given as the image of the mapping $c$.

[2] Usually, we assume differentiability in a weak sense.

Discrete representations are generally given as a set of landmarks $\{C_0, \ldots, C_{n-1}\}$. In this work, we focus on a representation where not only the landmarks but also an ordering of the landmarks is given. Therefore, a discrete, explicit contour is given as an $n$-tuple

$$C = (C_0, \ldots, C_{n-1}) \in \Omega^n$$

Implicit representations can also be used for discrete contours. In fact, every method that is implemented on a computer can only handle a finite number of parameters to represent a contour. The difference between continuous and discrete methods lies in fact in the moment of discretization:

**Continuous methods** In a continuous framework, the given problem is analyzed in a continuous context and thus, also the proposed method uses continuous operations like differential or integral operators. Only during the implementation a discretization is used.

**Discrete methods** In a discrete framework, we assume that we have to deal with data that is given in a discrete manner. This discrete data is then handled in a discrete sense. Instead of differential operators, graph theoretical approaches are normally used.

The advantages of minimizing a continuous cost functional are therefore:

- The developed method is formulated in a very general framework. Therefore, the results that are obtained for finer discretization converge towards the solution of the continuous framework. Therefore, discretization artefacts will be reduced by increasing the discretization size.

- Since the cost function that we like to minimize is given as a continuous, differentiable function, properties such as convexity can easily be studied. As a consequence, we can rule out local minima for convex functions even if we only minimize these functions via a gradient descent approach.

On the other hand, discrete methods have the following advantages:

- The runtime complexity can be given in the size of the input data. Therefore, we can estimate the maximum amount of time that the method consumes before providing a result.
- We always know that the minimum of a cost function over a finite set is taken by at least one element of this finite set. Therefore, the method will always terminate. Besides, we do not have to handle any form of numerical instabilities.

Overall, both discrete and continuous methods have their benefits. In this work, we always use the appropriate approach to solve a given problem as efficiently as possible.

## 1.3 Related Work

The study of shapes goes back several centuries. Even before the existence of digital images or even computers, there have been studies about shapes. Galilei [37] studied how the shape of an animal's bone reflects the size of the animal. His observation was that the bones do not only scale with the size of the animal, but that the *shape* of the bones changes. The similarity of shapes was studied in the early 20th century by Thompson [91]. He studied non-rigid body transformations to transform for example the shape of one fish (*Diodon*) into another (*Orthagoriscus*). This technique has been refined by Bookstein and Kendall [49, 7, 8]. For a more detailed

review, we refer to [32]. Recently, shapes have not merely been modeled as a set of points, but as a continuous closed line in the real plane. Younes, Mumford and Faugeras *et al.* worked on region based shape warping to measure the similarity of shapes [101, 20, 64, 86]. Besides a region based representation, one can also represent a shape as a closed contour. The concept of *Shape Morphing* for this kind of shapes has been studied by Younes, Mumford and Michor [66, 67, 102] as well as Klassen *et al.* [51, 50].

The computationally less expensive problem of *Shape Matching* has a long history in Computer Science and was originally influenced by the string matching method of Wagner and Fischer [97]. In addition to finding an optimal matching between two given strings, they also studied the induced distance function. This concept has first been applied to shape matching by McConnell *et al.* [65] and has since then become the core element of most shape matching methods [3, 38, 5, 100].

Besides the problem of shape morphing and shape matching, a central problem of Computer Vision has been the one of *shape acquisition*. Here, the problem of obtaining an object's shape from a given image is to be solved. This problem is either known as *image segmentation* or *shape denoising*. A brief review of classical segmentation approaches is presented in the beginning of Chapter 2 with a particular focus on those methods that can be formulated as an energy minimization process. The principle of introducing prior shape knowledge into Image Segmentation has been studied by several researchers [103, 42, 23, 58]. The focus of our work is set to the work by Cremers [24, 25]. The idea of formulating Image Segmentation and Shape Denoising as a convex functional that can be globally optimized was pioneered by Chan *et al.* [18].

The problem of shape classification with respect to a given distance function has been studied for a long time [46, 15, 34]. In this work, we will only focus on the problem of unsupervised learning. For more

sophisticated learning techniques, we like to refer to classical SVM methods as they are for example presented in [84].

## 1.4 Contribution

Certain parts of this work have been presented on different occasions [79, 82, 83, 80, 27, 81]. The main contribution can be split into four different components which is reflected by the structure of this work. In Chapter 2 and 3, we study continuous methods of *shape acquisition* and *shape morphing*. Chapter 4 is focused on discrete methods of *shape matching* and *shape clustering*. Chapter 5 concludes this work.

### Shape Acquisition

In Chapter 2, we address the problem of shape prior driven image segmentation. Our main contribution is it to formulate this problem as a minimization problem which can be solved globally. Hence, we can guarantee that our method finds the global optimum of a very challenging Computer Vision problem. One key contribution is the introduction of a new shape model, namely the *stochastic shape model*. It is a relaxed shape model where we assign to every point of the image domain the probability whether this point is an element of the shape. As a consequence of this relaxed shape definition, we can combine the advantages about the shape denoising method by Chan *et al.* [18] and the dynamical shape prior by Cremers [24]. By combining convex optimization and Lipschitz optimization techniques, we are therefore able to compute the global optimum of a *non-convex functional* very efficiently.

## Shape Morphing

In Chapter 3, we present the problem of shape morphing. To this end, every shape is represented as a mapping of the parameterizing circle into the real plane, i.e., $c : \mathbb{S}^1 \to \mathbb{R}^2$. The set of all shapes can then be modeled as an orbifold of infinite dimension. Any *morphing*, namely the continuous transformation from one shape into another can be represented as a path within this orbifold. By computing a geodesic between two different shapes, the length of such a geodesic defines a distance between the given shapes. Computing this distance efficiently is the goal of this chapter. After revising the classical shooting technique [51, 102], we propose a more efficient method to compute such a geodesic. Whereas the shooting method relies on solving an ordinary differential equation, our method addresses the original problem by applying the gradient descent method to the given energy functional that has to be minimized. As a consequence, our method is numerically much more stable than the shooting method and by construction, it provides an equidistant discretization of the geodesic in question. To make our method more accessible to a broader audience, we will also provide a brief introduction into the relevant differential-geometric concepts, such as tangential spaces and geodesics.

## Shape Matching

In Chapter 4, we propose two computationally inexpensive methods to compute the optimal matching, namely the best correspondence function between two different shapes. If each of the two shapes is discretized by $N$ shape points, both approaches exhibit a worst-case complexity of only $O(N^2 \log N)$. This is an important improvement with respect to the cubic complexity, that the classical *Dynamic Time Warping* approach [97, 65] (DTW) possesses. One approach reformulates the problem as a graph cut problem. Since the under-

lying graph is planar, we will then present an efficient graph cut method that makes use of this structure. Our method is based on a recent work by Borradaile and Klein [10] which is empirically slower than our method by a factor of about 2. Another method that we present exploits the graph structure of the Dynamic Time Warping approach. As a consequence, it improves the runtime significantly. To our knowledge, it is the fastest DTW based method for shape matching. To substantiate this, we also provide an extensive runtime comparison to other popular shape matching methods. In the last section of this chapter, we make use of the distance functions computed in Chapter 3 and 4 to merge different shapes into a class of similar shapes. We show that the shape matching approaches can measure the similarity of different shapes quite well.

### Conclusion

In Chapter 5, we present a conclusion of this work. Additionally, we discuss challenging open questions and give an outlook on future work.

# Chapter 2

# Shape Acquisition

In this chapter, we address the problem of acquiring shapes from images. An image is a mapping $I$ that assigns a color of a color-space $C$ to every point of a rectangular subset $\Omega$ of the real plane $\mathbb{R}^2$:

$$I : \Omega \to C \tag{2.1}$$

For gray-scaled images, $C$ can be identified with a convex subset of $\mathbb{R}$. But in general, $C$ is a convex subset of $\mathbb{R}^3$ which reflects the chosen color model. The focus of this work is not on different color models. Instead, every presented method can be applied on the one-dimensional gray-scale color space as well as on a three-dimensional color space like RGB, HSV or YUV.

Acquiring a shape results in assigning to every point $x$ of the rectangular image domain $\Omega$ a binary label $\ell(x)$ that indicates whether this point corresponds to a part of the observed object:

$$\ell : \Omega \to \{0,1\} \qquad (2.2)$$
$$x \mapsto \begin{cases} 1 & , x \text{ is part of the observed object} \\ 0 & , \text{otherwise.} \end{cases}$$

The central goal of this chapter is to acquire shapes with respect to prior shape knowledge. This means that we know what the object that we seek for will look like. As a consequence, a given set of shapes influences the search for the new shape that we are looking for in the given image. This means, that the *input data* consists not only of the given image but also of certain shapes.

To reach this shape acquisition goal, we will provide a brief introduction to different shape acquisition techniques in Section 2.1. Here, we will focus on *variational approaches*, i.e., on approaches that try to minimize a given *cost functional*. In Section 2.2, we will introduce the concept of *statistic shapes* which results in a convex shape space. Hence, vector space based dimension reduction or stochastic modeling can be applied to any collection of statistic shapes. This leads to a *knowledge driven shape acquisition* that we will present in Section 2.3. It is an extension of the work of Cremers [24] which results in a *non-convex* functional. In Section 2.4, we will show how this functional can be minimized efficiently. Especially, we will show that we can always guarantee to find the global optimum.

## 2.1 Classical Shape Acquisition Methods

The core element of every variational method is the definition of an energy functional that measures the performance of a possible output given the specific input data. For the problem of shape acquisition, we must therefore define such an evaluation functional. The easiest way of evaluating a given shape $\ell$ of the form (2.2), is to measure

the deviation of the color information $I(x)$ from a preselected color model. This can for example be done by assigning unique colors to foreground and background. So, if we assign to the foreground the color $\mu$ and to the background the color $\nu$, the following energy functional measures the performance of a given shape $\ell$:

$$E_{\text{color}}(\ell) = \int_{\Omega_1} (I(x) - \mu)^2 \, \mathrm{d}x + \int_{\Omega_0} (I(x) - \nu)^2 \, \mathrm{d}x \tag{2.3}$$

where

$$\Omega_i = \{x \in \Omega | \ell(x) = i\}.$$

Now, the goal is to find the global minimizer of the energy functional $E_{\text{color}}$. Since the minimizer of this functional is very sensitive to noise (see also Figure 2.1), researchers have been experimenting with different energies and it turned out, that the contour that separates foreground from background is a good descriptor of the involved shape and thus should contribute to the energy functional. Kass *et al.* [48] ignored in their *active contour* approach the region integral (2.3). Instead they considered the image information along the separating contour and penalized additionally the first and second derivative of the contour in order to regularize the segmentation. Another approach combined the first derivative of the contour with the image information along the contour. This *geodesic active contour* approach of Caselles *et al.* [16] addressed the image segmentation problem as minimizing the following energy functional:

$$E_{\text{GAC}}(C) = \int_C g(s) |\mathrm{d}C(s)| \tag{2.4}$$

In this functional, $g(s)$ is a positive function that depends inversely proportional on the image's gradient at the point $s \in C \subset \Omega$. One of the major drawbacks of this functional is the fact that the empty set is the minimizer of this functional. Therefore, other constraints were

introduced to guarantee that a possible global minimum provides a meaningful shape. For discrete approaches in this area, we would like to refer to the concept of *intelligent scissors* [70] or to that of *corridor scissors* [35].

Independent of this purely edge based approaches, Mumford and Shah [71] proposed in 1989 an approach that was based on (2.3). Their approach extended the functional (2.3) to a more general class of color models and incorporated an additional term that penalizes the length of the separating contour. Since we want to focus on the simple color model which assigns a unique color to foreground and background, we will here only consider the so called *piecewise-constant Mumford-Shah functional*:

$$E_{\mathrm{MS}}(\ell) = \int_{\Omega_1} (I(x) - \mu)^2 \, \mathrm{d}x + \int_{\Omega_0} (I(x) - \nu)^2 \, \mathrm{d}x + \gamma \cdot \mathrm{length}(C) \tag{2.5}$$

where $\mathrm{length}(C)$ describes the sum of the lengths of all $\Omega_0$-$\Omega_1$-separating contours. Note that $\Omega_0$, $\Omega_1$ and $C$ are uniquely defined by the labeling mapping $\ell$. Only $\gamma$ is a parameter that can be chosen by the user. It regulates the length of the shape's boundary contours. Since this model is only dependent on one parameter, this model is easy to adapt to given input data. But it took about one decade to find a way to solve the induced minimization task

$$\ell = \operatorname*{argmin}_{\ell:\Omega\to\{0,1\}} E_{\mathrm{MS}}(\ell, \mu, \nu) \tag{2.6}$$

in a continuous sense.

Previously, the discretized version of (2.5) was already known in the 80s and Geman and Geman [40] proposed in 1984 a method to solve this minimization problem via an approach that was based on simulated annealing. In 1989, Greig *et al.* [41] reformulated this problem as a minimum cut problem. Therefore, (2.6) could be computed in

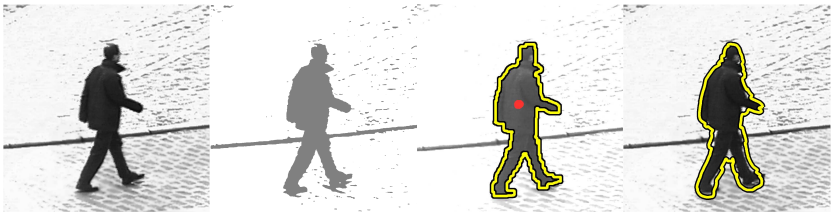


**Figure 2.1: Segmentation.** To retrieve a shape from a given image (1st frame), we can apply a color model which results in a noisy shape (2nd frame). The *geodesic active contour* model (2.4) can find an accurate shape, if we preselect some pixels that are part of the shape (red circle in the 3rd frame). Without any user interaction, the *Mumford-Shah functional* (2.5) can also find a correct segmentation (4th frame).

polynomial runtime. In 2001, Boykov and Kolmogorov presented an efficient algorithm [12] to compute the minimum cut and this method has since then become a very powerful tool in Computer Vision.

At the same time, Chan and Vese proposed one of the first continuous approaches [19] to solve the continuous functional (2.5) via a curve evoluting process. The region separating curve $C$ was modeled via a *signed distance function* $\varphi$:

$$\begin{aligned} \varphi : \Omega &\to \mathbb{R} \\ x &\mapsto \begin{cases} -\operatorname{dist}(x, C) & , \text{ if } x \in \Omega_0 \\ +\operatorname{dist}(x, C) & , \text{ if } x \in \Omega_1 \end{cases} \end{aligned} \tag{2.7}$$

Together with the Heaviside-function $H$:

$$\begin{aligned} H : \Omega &\to \mathbb{R} \\ x &\mapsto \begin{cases} 0 & , \text{ if } x \leq 0 \\ 1 & , \text{ if } x > 0 \end{cases} \end{aligned} \tag{2.8}$$

the functional (2.5) can be rewritten as

$$\begin{aligned} E_{\mathrm{CV}}(\varphi) = \int_{\Omega} & (I(x)-\mu)^2 \cdot H \circ \varphi(x) + \\ & (I(x)-\nu)^2 \cdot (1 - H \circ \varphi(x)) + \\ & \gamma \cdot \|\nabla (H \circ \varphi(x))\| \mathrm{d}x \end{aligned} \tag{2.9}$$

Therefore, the problem of image segmentation can be formulated as minimizing $E_{\mathrm{CV}}$ over the set of all signed distance functions. Using gradient descent approaches, a minimum of this functional can be found. But since the signed distance functions do not form a convex set, the method can potentially get stuck in a local minimum.

This problem was circumvented by another reformulation of this problem. In the seminal work of Chan, Esedoḡlu and Nikolova [18], the expression $H \circ \varphi$ has been replaced by the function $u : \Omega \to \{0, 1\}$. Additionally, the codomain of $u$ was extended to the convex set $[0, 1]$. Therefore the Chan-Vese functional (2.9) became

$$\begin{aligned} E_{\mathrm{TV}}(u) = \int_{\Omega} & (I(x)-\mu)^2 \cdot u(x) + \\ & (I(x)-\nu)^2 \cdot (1 - u(x)) + \\ & \gamma \cdot \|\nabla u(x)\| \mathrm{d}x \end{aligned} \tag{2.10}$$

Since the set of all functions $u$ form a convex set and the functional $E_{\mathrm{TV}}$ is convex, a gradient descent approach leads directly to a global optimum. It may of course be possible that such an optimum $u^*$ assigns to some points of the image domain numbers which are neither 0 nor 1. But if we threshold $u^*$ with respect to the value 0.5, we receive a function $\bar{u} : \Omega \to \{0, 1\}$ that represents a shape:

$$\begin{aligned} \bar{u} : \Omega \to & \{0, 1\} \\ x \mapsto & \begin{cases} 0 & , \text{ if } u^*(x) \leq 0.5 \\ 1 & , \text{ if } u^*(x) > 0.5 \end{cases} \end{aligned}$$

The important contribution of [18] was the so called *thresholding theorem* that states that $\bar{u}$ is also a global optimum of $E_{\mathrm{TV}}$. Hence, a continuous method to find the global optimum of (2.5) was found. In combination with a primal-dual scheme [17], the developed method was competitive to the graph cut methods in the mean of runtime by using an efficient GPU implementation [95]. Also it was shown, that the result of this continuous approach is more accurate than the graph cut approaches since it inhibits metrication errors [52].

In fact, the thresholding theorem does not depend on the fact that the image related data terms are quadratic. Moreover, any color model for the background and the foreground of an image can be used to find the global optimum of the image segmentation problem

$$E_{\mathrm{TV}}(u) = \int_{\Omega} f_I(x) \cdot u(x) + g_I(x) \cdot (1 - u(x)) + h_I(x) \cdot \|\nabla u(x)\| \mathrm{d}x \tag{2.11}$$

Please note that for $f_I \equiv g_I \equiv 0$, this functional becomes the geodesic active contour functional. Hence (2.11) covers any hybrid model that involves regional terms and edge terms. The shape acquisition method that we use in this chapter is an extension of the functional (2.11). Instead of penalizing the contour's length, we will use an energy functional that penalizes the deviation of a shape from a preselected shape model. To this end, we will introduce in the next section a shape model that forms a convex subset in a Hilbert space.

## 2.2 Stochastic Shapes

In the following, we introduce the concept of *stochastic shapes*. The notion of shape fundamentally differs from classical definitions of shape like $\ell$ in (2.2). We replace the hard decision of *a point is part of the shape* by a relaxed probability associated with each point. This probability should not be understood as a probability in the

sense of probabilistic measure theory. It should rather be seen as an application of the fuzzy set theory introduced by Zadeh [104]. The key contribution of the stochastic shape is to show that this fuzzy *relaxation* in the definition of shape gives rise to a number of advantages in the context of shape modeling and shape inference. Most prominently it enables us to acquire shapes from images in a globally optimal manner under the consideration of prior shape knowledge. So let us start by formally defining the stochastic shape:

**Definition 1** (Stochastic Shape)**.** An $L^2$-function

$$q : \Omega \to [0; 1] \tag{2.12}$$

which assigns to any point $x \in \Omega$ a probability $q(x)$ that $x$ is part of the shape (cf. Figure 2.2, left side) is called a *stochastic shape*. The space of all stochastic shapes will be denoted by $\mathcal{Q}$.

In contrast to explicit representations, the above implicit representation does not depend on a specific choice of parameterization. In contrast to alternative implicit representations of shape such as the signed distance function of (2.7) or alternative representations [33], the value of $q$ has a clear probabilistic interpretation. Also, there are no redundancies encoded in $q$. In contrast, a signed distance function $\varphi$ must almost everywhere fulfill the Eikonal equation $\|\nabla\varphi\| = 1$. As a consequence, the set of all signed distance functions is not convex. The set $\mathcal{Q}$ on the other hand forms a convex subset of a *Hilbert space*:

**Proposition 1.** *The stochastic shape space $\mathcal{Q}$ forms a convex subset of the* Hilbert Space $L^2(\Omega, \mathbb{R})$*. This Hilbert Space is equipped with the scalar product*

$$\langle p\,,\, q\rangle := \int_\Omega p(x) \cdot q(x) \mathrm{d}x \tag{2.13}$$

*The induced norm will be denoted*

$$\|q\| := \int_\Omega q(x)^2 \mathrm{d}x^{\frac{1}{2}} \tag{2.14}$$

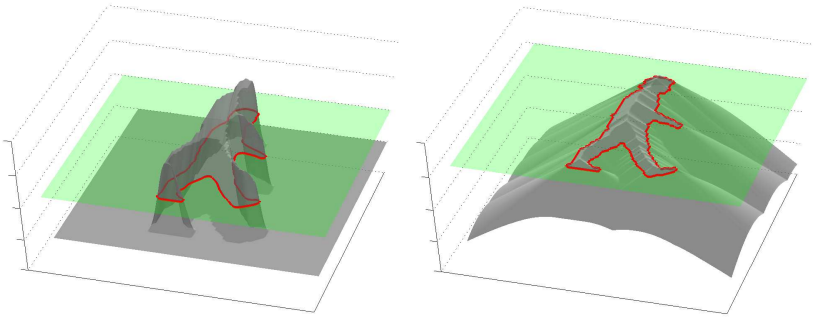


**Figure 2.2: A relaxed notion of shape.** In this chapter, we introduce a novel definition of *shape* as a function $q : \Omega \to [0, 1]$ specifying the probability that a pixel $x \in \mathbb{R}^2$ is part of the shape (left). In contrast to the commonly used signed distance representation (right), the resulting image segmentation with statistic shape priors corresponds to the minimization of convex functionals over convex domains.

*Proof.* Since $\mathcal{Q}$ is the set of all $L^2$ functions that map $\Omega$ to $[0, 1]$, it is obviously a subset of $L^2(\Omega, \mathbb{R})$. For the proof of convexity, assume $p, q \in \mathcal{Q}$. Now, we consider the convex combination $q_\gamma := \gamma p + (1-\gamma)q$ of these two shapes by choosing a specific $\gamma \in [0; 1]$. Then, we have:

$$\begin{aligned} q_\gamma(x) &= \gamma p(x) + (1-\gamma)q(x) & q_\gamma(x) &= \gamma p(x) + (1-\gamma)q(x) \\ &\geq \gamma \cdot 0 + (1-\gamma) \cdot 0 = 0 & &\leq \gamma \cdot 1 + (1-\gamma) \cdot 1 = 1 \end{aligned}$$

□

Note that while $\mathcal{Q}$ is not a Hilbert space like $L^2(\Omega, \mathbb{R}^2)$, $\mathcal{Q}$ is nonetheless a metric space. This means that the distance function

$$d(q_1, q_2) := \|q_1 - q_2\| = \int_\Omega (q_1(x) - q_2(x))^2 \mathrm{d}x^{\frac{1}{2}}$$

is a metric:

**Definition 2** (Metric)**.** Given a space $X$, a function $d : X \times X \to \mathbb{R}_0^+$ is a metric if it fulfills the following properties:

$$\begin{aligned} \forall x, y \in X: \quad & d(x,y) = 0 \Leftrightarrow x = y && \textit{(Positive Definiteness)} \\ \forall x, y \in X: \quad & d(x,y) = d(y,x) && \textit{(Symmetry)} \\ \forall x, y, z \in X: \quad & d(x,z) \leq d(x,y) + d(y,z) && \textit{(Triangle Inequality)} \end{aligned}$$

The convexity of $\mathcal{Q}$ shown in Proposition 1 has an important technical consequence for statistic shape modeling and shape inference. Especially, we can apply classical dimension reduction or model fitting methods to shapes.

But also semantically, the convexity gives rise to some important properties. The convexity of the shape space $\mathcal{Q}$ implies that any convex combination of a set

$$\chi = \{q_1, \ldots, q_N\} \subset \mathcal{Q}$$

of training shapes will correspond to a valid shape. In particular, the *mean*

$$\mu = \frac{1}{N} \sum_{i=1}^{N} q_i(x)$$

is a function which assigns to each point $x \in \Omega$ the average of all probabilities. Similarly, statistic notions such as *covariance matrices* and *eigenmodes* can be easily defined.

**Definition 3** (Eigenmodes)**.** The entries of the covariance matrix $\Sigma = (\sigma_{i,j})_{i,j=1,\ldots,N}$ are defined via the scalar product (2.13):

$$\sigma_{i,j} := \langle q_i - \mu \,,\, q_j - \mu \rangle = \int_\Omega (q_i(x) - \mu(x))(q_j(x) - \mu(x)) \mathrm{d}x$$

Let $v_i \in \mathbb{R}^N$ be the eigenvectors of $\Sigma$ with respect to the eigenvalue $\lambda_i \in \mathbb{R}$ such that $\lambda_1 \geq \ldots \geq \lambda_N$. Then, the following functions $\psi_1, \ldots, \psi_N : \Omega \to \mathbb{R}$

$$\psi_i(x) = \sum_{j=1}^{N} (v_i)_j \cdot (q_j(x) - \mu(x)) \tag{2.15}$$

are the eigenmodes of $\chi$.

Note that the eigenmodes are not necessarily stochastic shapes. They just define in what way most of the input shapes in $\chi$ deviate from their mean $\mu$. In particular, the eigenmodes help to reduce the infinite dimensionality of $\mathcal{Q}$ to a convex subset of finite dimension that still encodes the relevant information of $\chi$.

**Definition 4** (Finite subspace of $\mathcal{Q}$)**.** The subspace of stochastic shapes spanned by the first $n \leq N$ eigenmodes $\{\psi_1, \ldots, \psi_n\}$ of the set $\chi$ is

$$\chi_n := \left\{ q_\alpha = \mu + \sum_{i=1}^{n} \alpha_i \psi_i \,\middle|\, q_\alpha(x) \in [0,1] \right\} \tag{2.16}$$

which is a subset of the finite dimensional, affine space

$$\widehat{\chi}_n := \left\{ q_\alpha = \mu + \sum_{i=1}^{n} \alpha_i \psi_i \,\middle|\, \alpha \in \mathbb{R}^n \right\} \tag{2.17}$$

We will now show that not only $\mathcal{Q}$ but also the set of all parameters $\alpha$ in (2.16) is convex. This is a very important property for the image acquisition method that we will present in the next section.

**Lemma 1.** *The set $\chi_n$ in* (2.16) *is convex.*

*Proof.* For any $n \leq N$, the set $\chi_n$ is the intersection of the affine space $\widehat{\chi}_n$ with the convex space $\mathcal{Q}$. Since both sets are convex, their intersection is also convex which proves this lemma. □

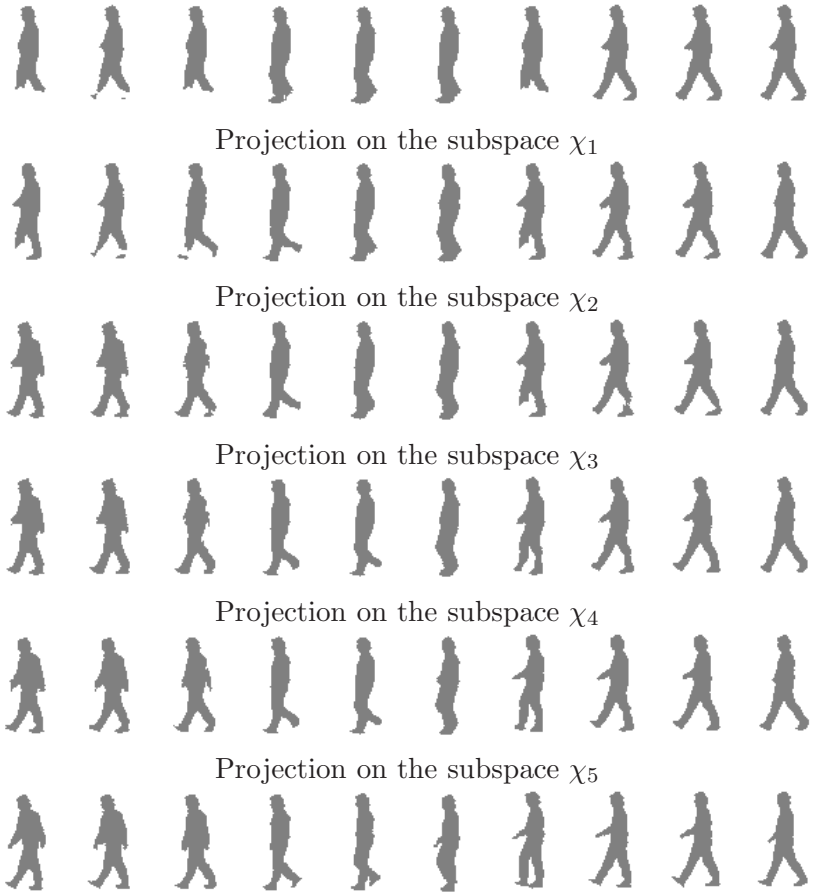



**Figure 2.3: Dimension Reduction.** A walking sequence (last row) is projected onto the low-dimensional spaces $\chi_1, \ldots, \chi_5$ ($1^{\text{st}}$ to $5^{\text{th}}$ row). Even at a one-dimensional projection the main information about walking is recovered.

The set of training shapes $\chi$ can thus be approximated by nested low-dimensional spaces:

$$\{\mu\} = \chi_0 \subset \chi_1 \subset \cdots \subset \chi_n \subset \cdots \subset \chi_N \subset \mathcal{Q}$$

The elements of a space $\chi_n$ are compactly represented by vectors $\alpha \in \mathbb{R}^n$ of *eigencoefficients*, modeling the shape

$$q_\alpha = \mu + \Psi \cdot \alpha := \mu + \sum_{i=1}^{n} \psi_i(x) \cdot \alpha_i. \tag{2.18}$$

We will now show that the set of eigencoefficients that describe stochastic shapes form also a convex set:

**Lemma 2.** *The set* $\mathcal{A}_n := \{\alpha \in \mathbb{R}^n | q_\alpha \in \mathcal{Q}\}$ *of all feasible* $\alpha$ *is convex.*

*Proof.* Let $\alpha_1, \alpha_2$ be two elements of $\mathcal{A}_n$ and $\gamma \in [0; 1]$. Then we have to prove that the shape representing vector $\alpha := \gamma\alpha_1 + (1-\gamma)\alpha_2$ is also feasible, i.e., $\alpha \in \mathcal{A}_n$:

$$\begin{aligned} q_\alpha = \mu + \Psi\alpha =& \mu + \gamma\Psi\alpha_1 + (1-\gamma)\Psi\alpha_2 \\ =& \gamma\left(\mu + \Psi\alpha_1\right) + (1-\gamma)\left(\mu + \Psi\alpha_2\right) \\ =& \gamma q_{\alpha_1} + (1-\gamma) q_{\alpha_2} \in \mathcal{Q}. \end{aligned}$$

□

So overall, we have defined the stochastic shape space $\mathcal{Q}$ and we could not only show that this shape space is convex. In addition we also introduced to every subset $\chi \subset \mathcal{Q}$, a meaningful notion of eigenmodes which define a convex, low-dimensional subspace $\chi_n$. Finally, we showed that the coordinate system $\mathcal{A}_n$ of this subspace is convex and we have a natural, affine mapping between $\mathcal{A}_n$ and $\chi_n$, namely $\alpha \mapsto \mu + \Psi \cdot \alpha$.

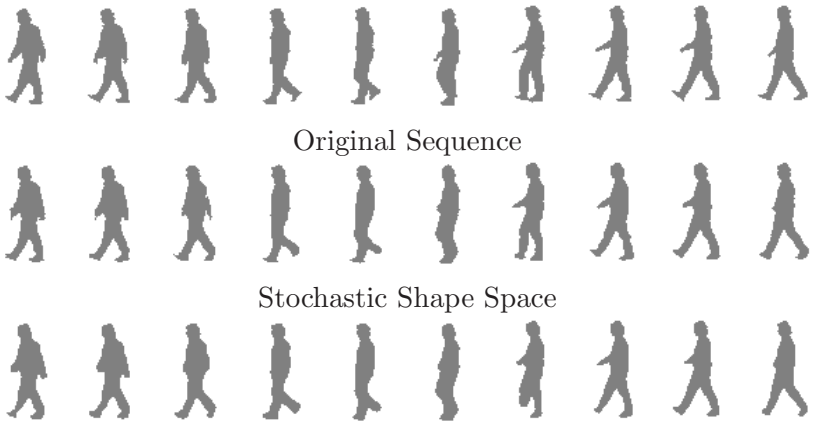



**Figure 2.4: Dimension Reductions.** If we perform a dimension reduction with respect to the stochastic shape model ($2^{\text{nd}}$ row), we receive better results than for the dimension reduction with respect to the signed distance functions ($3^{\text{rd}}$ row).

In Figure 2.3, a walking sequence $\chi \subset \mathcal{Q}$ and its orthogonal projection onto $\chi_1, \ldots, \chi_5$ is shown. One can see that even the five-dimensional space $\chi_5$ provides a good approximation of the relevant information encoded in $\chi$. In Figure 2.4, the projection results for the stochastic shape model is compared to a projection on a shape space based on the signed distance functions [24]. Especially in the third and seventh frame, we can see that the stochastic shape subspace encodes the details of $\chi$ more accurately than the shape model based on signed distance functions. In the next section, we will use this representation to incorporate *stochastic shape prior* into shape acquisition.

## 2.3 Knowledge-driven Shape Acquisition

The general shape acquisition functional (2.11) can retrieve a shape from an image under the assumption that the shown object can easily be distinguished by the color information that the image provides. Unfortunately, that is not always true. Due to clutter or occlusions, there are certain small regions in the image to which the pre-learned color model of (2.11) does not fit any more. On the other side, humans are able to detect a person even when it is raining or snowing. We believe that this is possible because humans are well aware of a person's shape or that they know how people normally walk. Hence, we are able to filter out disturbing information and thus, we can focus on the visual task that we like to solve. In Computer Vision this situation has been modeled by introducing shape prior into the segmentation [58, 94, 78, 24, 25].

In this section, we will propose a method to retrieve a stochastic shape from an image applying three different concepts of shape priors to the proposed stochastic shape model of Section 2.2. The concepts that we apply to the acquisition of stochastic shapes are the concepts of *static uniform shape priors* [94], *static Gaussian shape priors* [78] and *dynamical shape prior* [24]. To this end, we assume that a training set $\chi = \{q_1, \ldots, q_N\} \subset \Omega$ of known shapes is given. This training set will be used to define different shape priors. All these shape priors have in common that we only consider the first $n < N$ eigenmodes $\psi_1, \ldots, \psi_n$ of this training set as defined in (2.15). As in Section 2.2, we denote the convex, finite dimensional subset of $\Omega$ spanned by the $n$ eigenmodes as $\chi_n \subset \widetilde{\chi}_n$ and the space of all possible eigencoefficients is denoted as $\mathcal{A}_n$. Also, we use the notation $q_\alpha$ to denote the shape of $\chi_n$ that is been encoded by $\alpha \in \mathcal{A}_n$ via (2.18).

To retrieve a shape from an image using the shape prior, we like to minimize a cost function that penalizes the deviation from the color model like in (2.11). At the same time a deviation from the shape

prior should also be penalized. Therefore, we like to minimize a cost function of the following very general form:

$$E(\alpha) = E_{\mathrm{TV}}(q_\alpha) + \gamma E_{\mathrm{Shape}}(\alpha). \tag{2.19}$$

Here, $E_{\mathrm{Shape}}$ is a cost function that penalizes the deviation of a given shape $q_\alpha$ from the expected shape and $\gamma$ is a parameter that weights the importance of the chosen shape prior. For the cost functional $E_{\mathrm{TV}}$ we have to model the color distribution for the foreground and background via functions $f$ and $g$ respectively. Meaningful choices of $f$ and $g$ for shape acquisition are given by:

$$f = -\log p_{ob}(I) \qquad\qquad g = -\log p_{bg}(I) \tag{2.20}$$

where $p_{ob}$ and $p_{bg}$ represent the color histograms (probabilities) of object and background [77]. Since the shape prior acts as regularizer, we do not need the edge term $h$ of (2.11) any more and we set $h \equiv 0$.

For the shape energy $E_{\mathrm{Shape}}$ in (2.19), we consider one of the following three statistic shape priors:

1. **Static uniform shape priors** The distribution of training shapes is assumed to be uniform within the eigenmode space $\mathcal{A}_n$. Such a model was introduced for level set functions in [94] and it corresponds to setting $E_{\mathrm{Shape}} = 0$ in (2.19). One may argue that this leads to a function that does not differ from (2.11) and will therefore lead to the same solution. This is not true since the set of feasible solutions is reduced. Note that we optimize over $\alpha \in \mathcal{A}_n$. Therefore, we consider only shapes that can be represented via the first $n$ eigenmodes of $\chi$. Hence, this is in fact a shape prior even if we do not alter the corresponding energy function.

2. **Static Gaussian shape priors:** The distribution of training shapes is assumed to be Gaussian. By using the covariance

matrix $\Sigma$ computed in Section 2.2, we receive a Mahalanobis type energy of the following form:

$$E_{\text{Shape}}(\alpha) = \left\langle \alpha \,,\, \Sigma^{-1}\alpha \right\rangle \tag{2.21}$$

A related model was proposed for level set functions in [78]. It has an important advantages over the uniform shape prior, because it penalizes the deviation of the first few eigenmodes more than the deviation from the remaining eigenmodes. As a consequence the relevance of the first few eigenmodes is emphasized more adequately by this model than the uniform shape model. Since the cut off after the $n$th eigenmode leads to a virtual infinite penalization of the eigenmodes $\psi_{n+1}, \ldots, \psi_N$, the Gaussian shape prior is more sensitive to the choice of the parameter $n$. It should therefore be chosen carefully.

3. **Dynamical shape priors:** The evolution of shape vectors $\alpha$ is modeled by a linear dynamical system. This sophisticated shape prior can only be used if we have to deal with a whole collection of images. Therefore it is well suited for the problem of object tracking in videos. Since the stochastic shapes form a convex set, we can train a Markov chain model of the form

$$\beta_t = \sum_{i=1}^{k} A_i \beta_{t-i} + \eta, \tag{2.22}$$

where $\eta$ describes the involved Gaussian noise and $A_1, \ldots, A_k$ model the linear dependency of the current eigencoefficient $\beta_t$ with respect to previous eigencoefficients $\beta_{t-1}, \ldots \beta_{t-k}$. $k$ is also known as the size of the Markov chain model. Such a model can be learned by assuming that the elements of the shape prior $\chi$ are given in a ordered manner such that the sequence $q_1, \ldots, q_N$ describes a natural motion that can be modeled via the chain model (2.22). Then, every shape $q_i \in \chi$

can be modeled by an eigencoefficient vector $\beta_i \in \mathcal{A}_n$. These vectors can give rise to the autoregressive model parameters $k, A_1, \ldots, A_k$ via classical methods such as the one presented in [72]. This model now gives rise to the following shape energy at a certain time $t$:

$$E_{\text{Shape}}(\alpha) = \left\langle \alpha - v_t \,,\, \Sigma^{-1}(\alpha - v_t) \right\rangle, \tag{2.23}$$

where $v_t = \sum_{i=1}^{k} A_i \alpha_{t-i}^*$ is the prediction by the Markov chain based on shape estimates $\alpha_{t-1}^*, \ldots, \alpha_{t-k}^*$ obtained for the last $k$ images. A related model for level set functions was introduced in [24]. Since level set functions do not form a convex set, the resulting shape priors $v_t$ have to be projected back onto the space of feasible level set functions. For the statistic shape priors this is not necessary any more.

In order to simplify the models of (2.22) as much as possible, we assumed that the given shapes $q_i$ are transformed in a way that the pairwise distance of consecutive shapes is minimized. In other words, we compute the following minimum, prior to the model learning

$$\begin{aligned} \theta_i &= \operatorname*{argmin}_{\theta \in \mathrm{SE}(2)} \left\| q_i \circ \theta^{-1} - q_{i-1} \right\| \\ &= \operatorname*{argmin}_{\theta \in \mathrm{SE}(2)} \int_\Omega \left( q_i(\theta^{-1} x) - q_{i-1}(x) \right)^2 \mathrm{d}x \end{aligned}$$

and every shape $q_i$ is replaced by $q_i \left( \theta_1 \circ \cdots \circ \theta_i(x) \right)$. As a consequence, the learned shapes are free from possible rigid body transformations. Hence the autoregressive model represents only the continuous deformation of shapes. This makes model learning much easier and our model is thus independent from translation or rotation which may be influenced by camera movements.

For the shape acquisition on the other hand, the transformations have to be put back into play. Now, we have to consider any possible

transformation of $\theta \in \mathrm{SE}(2)$ in order to find the exact shape in the current image. To this end, we define an energy function $E(\alpha, \theta)$ that depends on shape parameters $\alpha \in \mathcal{A}_n$ as well as the transformation parameter $\theta \in \mathrm{SE}(2)$ modeling rigid body motions:

$$E(\alpha, \theta) = E_{\mathrm{TV}}(q_\alpha(\theta x)) + \gamma E_{\mathrm{Shape}}(\alpha) \tag{2.24}$$

with $E_{\mathrm{TV}}$ defined according to (2.11) and with $E_{\mathrm{Shape}}$ defined either as 0 or defined according to (2.21) or (2.23).

Even though the resulting function $E$ is not a convex function, we will show in the next section how we can guarantee to find the global optimum by combining a convex optimization technique with the concept of Lipschitz optimization.

## 2.4 Global Optimization

In this section, we will show how to globally optimize the non-convex function (2.24).

In order to do so, we use a separation of variables:

$$\min_{\alpha,\theta} E(\alpha, \theta) = \min_{\theta} \left( \min_{\alpha} E(\alpha, \theta) \right)$$

Thus, we have to apply two optimization methods. The first one computes the global optimum of $E(\alpha, \theta)$ with respect to $\alpha$. This global optimum will then be dependent on $\theta$. As a consequence, we have to compute the global optimum with respect to $\theta$ of a function $E^*$ that is induced by this first optimization process:

$$\begin{aligned} E^* : \mathrm{SE}(2) &\to \mathbb{R} \\ \theta &\mapsto \min_{\alpha \in \mathcal{A}_n} E(\alpha, \theta) \end{aligned} \tag{2.25}$$

In the next section, we will show how to compute $E^*$ by applying a convex optimization technique. In the subsequent section, $E^*$ is optimized via a Lipschitz optimization approach.

### 2.4.1 Convex Optimization for Deformations

In Section 2.4.2, we will show that $E^*$ can be optimized with a Lipschitz optimization technique. This approach only depends on evaluating $E^*$ at some discrete points. Nonetheless, it will up to a preselected error $\varepsilon$ always compute the global optimum of the continuous function. This optimization approach depends highly on the accurateness of computing $E^*(\theta)$. Thus, we have to show first that $E^*$ can be computed very efficiently. In fact, we show that computing $E^*(\theta)$ results in optimizing a convex function over a convex domain:

**Proposition 2.** *To evaluate $E^*$ at a given rigid body transformation $\theta \in \mathrm{SE}(2)$ results in optimizing a convex function over a convex domain.*

*Proof.* Due to Lemma 2 we know that the optimization domain $\mathcal{A}_n$ of feasible $\alpha$ values forms a convex set. Therefore, we need to prove that the optimization involved in evaluating the function $E^*$ is convex in $\alpha \in \mathcal{A}_n$ for any choice of data term and any of the three shape priors discussed in Section 2.3. The zero function that describes the uniform shape prior is obviously convex. For the Gaussian shape prior (2.21) as well as the dynamic shape prior (2.23), the introduced functions $E_{\mathrm{Shape}}$ are both quadratic in $\alpha$ with the positive definite covariance matrix $\Sigma^{-1}$ as their Hessian. Hence, all considered shape prior functions $E_{\mathrm{Shape}}$ are convex. The function $E_{\mathrm{TV}}$ that evaluates the goodness of the shape with respect to the image information, is also convex in $q_\alpha$ [18]. Since $q_\alpha$ is affine in $\alpha$ (cf. (2.18)), the function $\alpha \mapsto E_{\mathrm{TV}}(\alpha, \theta)$ is a composition of a convex function with an affine

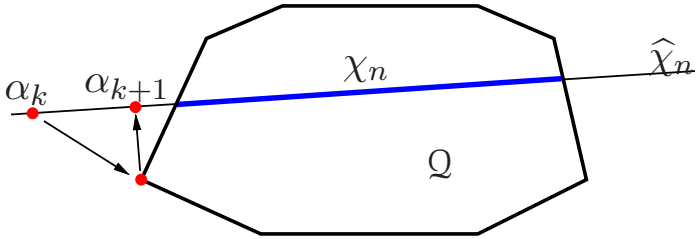



**Figure 2.5: Iterated Projections.** The intersection of the convex shape space $\mathcal{Q}$ with the finite dimensional space $\widehat{\chi}_n$ results in the convex set $\chi_n$. Starting with $q_{\alpha_k} \notin \chi_n$, a solution in $\chi_n$ is obtained by iterated projections.

mapping and thus convex in $\alpha$. Since $E$ is the sum of two convex functions, it is itself convex. □

Before we go on describing our Lipschitz optimization scheme, we give some remarks on how to compute $E^*$ efficiently. Since the energy $E$ is convex in $\alpha$, a gradient descent approach would always lead to the global optimum $\alpha^*$. In particular, if $\alpha^*$ is within the domain $\mathcal{A}_n$ of feasible $\alpha$, we have thus found a way to compute $E^*(\theta)$. But problems may occur if $\alpha^*$ is outside of our convex domain. In this case, we have to re-project the computed $\alpha$ at any time that we leave the convex domain [74, 75]. Since our convex domain is quite complicated, this projection can become computationally expensive. To cope with this problem, we propose an iteration scheme that approximates this projection. So, we will define a sequence $(\alpha_k)_{k\in\mathbb{N}}$ of eigencoefficients that converge towards an $\alpha' \in \mathcal{A}_n$ (cf. Figure 2.5). The presented method works as following:

1. Let $\alpha_1 := \alpha$.
2. For all $k = 1, 2, \ldots$, do the following

(a) At first, we project the given shape function $q_{\alpha_k}$ onto $\mathcal{Q}$ which amounts to setting all values to 1 or 0 which are above or below these levels respectively. This is in fact the orthogonal projection of $q_{\alpha_k}$ onto $\mathcal{Q}$ with respect to the $L^2$-norm of (2.14). Denote this projected shape as $s_k$.

(b) By projecting $s_k$ onto $\widehat{\chi}_n$, we receive a shape of the form $\mu + \Psi \cdot \alpha_{k+1}$. This defines the next $\alpha$ value. Since, the eigenmodes describe an orthonormal system, the components of $\alpha_{k+1}$ can be easily computed via

$$(\alpha_{k+1})_i = \int_\Omega (s_k(x) - \mu(x))\psi_i(x)\mathrm{d}x$$

Note that this sequence will not necessarily converge towards the right projection. But it will converge towards a point of $\mathcal{A}_n$ as we will show in the following proposition:

**Proposition 3.** *Let $\alpha \in \mathbb{R}^n$ an arbitrary vector and $(\alpha_k)_{k\in\mathbb{N}}$ be the sequence defined as above. Then*

$$\lim_{k\to\infty} \operatorname{dist}(q_{\alpha_k}, \chi_n) = 0$$

*Proof.* Let us assume that there exists an $\varepsilon > 0$ such that $\operatorname{dist}(q_{\alpha_k}, \chi_n) \geq \varepsilon$ holds for every $k \in \mathbb{N}$. We will show that this will lead to a contradiction and thus proving the proposition. For now, we choose an arbitrary element $q \in \chi_n$ that we will use as a reference shape. Note that every iteration $\alpha_k \mapsto \alpha_{k+1}$ consists of two orthogonal projections onto convex sets, namely onto $\mathcal{Q}$ in the first step and onto $\widehat{\chi}_n$ in the second step. The property of convexity of these two sets results in

$$\begin{aligned} \|s_k - q\|^2 &\leq \|q_{\alpha_k} - q\|^2 - \operatorname{dist}(q_{\alpha_k}, \mathcal{Q})^2 \\ \left\|q_{\alpha_{k+1}} - q\right\|^2 &\leq \|s_k - q\|^2 - \operatorname{dist}(s_k, \widehat{\chi}_n)^2 \end{aligned}$$

and as a consequence

$$\left\|q_{\alpha_{k+1}} - q\right\|^2 \leq \|q_{\alpha_k} - q\|^2 - \left(\operatorname{dist}(q_{\alpha_k}, \Omega)^2 + \operatorname{dist}(s_k, \widehat{\chi}_n)^2\right)$$

Since the distances $\operatorname{dist}(q_{\alpha_k}, \Omega)$ and $\operatorname{dist}(s_k, \widehat{\chi}_n)$ are always realized by elements which are at least by $\varepsilon$ away from $\mathcal{A}_n$, the whole expression can be simplified as

$$\left\|q_{\alpha_{k+1}} - a\right\|^2 \leq \|q_{\alpha_k} - a\|^2 - \operatorname{dist}(\widetilde{\Omega}, \widetilde{\chi}_n)^2 \tag{2.26}$$

with

$$\begin{aligned}\widetilde{\Omega} :=& \{q \in \Omega | \varepsilon \leq \operatorname{dist}(q, \chi_n) \leq \operatorname{dist}(q_{\alpha_1}, \chi_n)\} \\ \widetilde{\chi}_n :=& \{q \in \chi_n | \varepsilon \leq \operatorname{dist}(q, \chi_n) \leq \operatorname{dist}(q_{\alpha_1}, \chi_n)\}\end{aligned}$$

Since $\widetilde{\chi}_n$ and $\widetilde{\Omega}$ are disjoint compact sets, their pairwise distance is positive and according to (2.26), the distance between $\alpha_k$ and $a$ will drop below any possible threshold after a finite amount of time. This contradicts the assumption that $\operatorname{dist}(q_{\alpha_k}, \chi_n)$ will always be above $\varepsilon$. Thus, the proposition is proven. □

Note that the orthogonal projection of a gradient update always results in an element that has still a smaller energy value than the starting element. Therefore, the method of iterated projections leads to a shape in $\chi_n$ that has a smaller energy than the starting shape. Hence, combining this approximative projection scheme with a gradient descent approach will eventually lead to the global minimum. In our experiments, we noted that after five iterated projections, the process of iterated projections normally halts. Therefore, the time consumption is considerably reduced via the presented approximative projection scheme. In Figure 2.6, an example for the iterated projection is shown. If we look at the thresholded shape in the second row of that figure, we can observe how the iterated projection

**Figure 2.6: Iterated Projections for Stochastic Shapes.** The method of iterated projections ($1^{\text{st}}$ row) pushes the probabilities at every pixel towards the interval $[0; 1]$ of feasible values. For the shape thresholded at 0.5 ($2^{\text{nd}}$ row), this results in restoring important parts of the shape.

results in repairing the threshold shape. The right leg for example is repaired by this projection. This is an important property of the shape prior driven shape acquisition. The observed data is repaired with respect to the pre-learned shape prior.

### 2.4.2 Lipschitz Optimization for Transformations

As we have seen in the last section, we are able to compute $E^*(\theta)$ of (2.25) for any transformation $\theta \in \mathbb{S}^1$. Now, we have to minimize this function. Note that we cannot use the same technique as in the last section because $E^*$ is highly non-convex. But we will show that $E^*$ is Lipschitz continuous and as a consequence, we can find an approximation of the global optimum by an adapted exhaustive search. Even though an exhaustive search is a classical discrete optimization scheme, the main contribution is to provide a good approximation of the global minimum with respect to original *continuous* function. So in the end, we are able to solve the problem of sampling a continuous domain in finite time. This is done by sampling it only at finite positions and at the same time we can guarantee that we have

a good approximation of the global minimum with respect to the continuous domain.

To this end, we assume that every shape of $\chi$ lies within a finite radius of $\rho$ near the origin, and thus all shapes are bounded from above by the function $q_{\text{supp}}$:

$$\forall \alpha \in \mathcal{A}_n \colon q_\alpha(x) \leq q_{\text{supp}}(x) = \begin{cases} 1, & \text{if } \|x\| \leq \rho \\ 0, & \text{else.} \end{cases} \tag{2.27}$$

As a consequence, the support of $q_{\text{supp}}$ lies inside of the ball $B_\rho(0)$ of radius $\rho$ centered in 0. We will now show that under mild regularity assumptions, the function $E^*$ can be globally optimized on SE(2) using the idea of Lipschitz continuity [43]. This technique was used in Computer Vision before to solve the problem of point cloud registration [59].

To find the global optimum of $E^*$, we iteratively subdivide the $\theta$-domain SE(2) into multiple smaller domains – see Figure 2.7. For every sub-domain $D \subset \text{SE}(2)$, we calculate the energy at one chosen sample $\theta_0$ which provides an *upper bound* for the global minimum. Provided that the gradient of $E^*(\theta)$ is bounded, a *lower bound* for each sub-domain $D$ can be determined. By performing a branch-and-bound method, we subdivide the sub-domains with the most promising lower bounds. In doing so, we iteratively find tighter lower bounds and terminate once sufficient accuracy is obtained.

To determine the lower bound for a sub-domain, we assume that the functional does not oscillate too rapidly. In other words, we need to assume that the following Lipschitz condition holds:

**Definition 5** (Lipschitz)**.** The function $F : \text{SE}(2) \to \mathbb{R}$ is called *Lipschitz continuous* if there exists a uniform $L \in \mathbb{R}$ such that for all $\theta_1, \theta_2 \in \text{SE}(2)$ the following inequality is fulfilled:

$$|F(\theta_1) - F(\theta_2)| \leq L \operatorname{dist}(\theta_1, \theta_2)$$

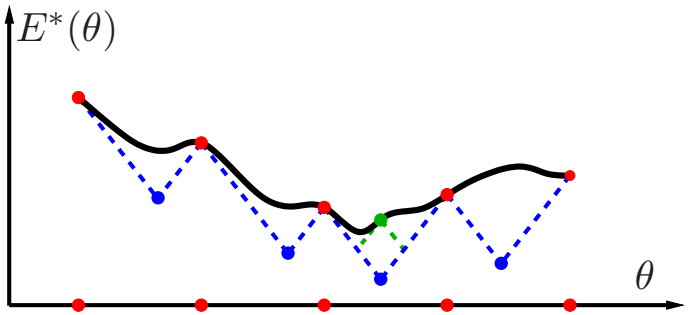



**Figure 2.7: Lipschitz approach.** If $E^*$ (solid line) is Lipschitz continuous (cf. Definition 5), then one can globally minimize it in a continuous sense by iteratively finding lower bounds.

For differentiable functions $F$ this definition is equivalent to the property that the derivative $\frac{\mathrm{d}F}{\mathrm{d}\theta}$ is bounded by $L$.

In order to prove that $E^*$ is Lipschitz continuous, we proceed as follows: First, we will estimate a Lipschitz constant $L$ for the function $E(\alpha, \theta)$ at fixed value $\alpha$. Afterwards, we will show in Proposition 4 that the same constant $L$ is also a Lipschitz constant for $E^*$ itself.

**Lemma 3.** *If all functions of the training set $\chi$ are Lipschitz continuous with constant $L_\chi$, then $E(\alpha, \theta)$ is Lipschitz continuous with respect to $\theta$. The Lipschitz constant is independent of $\alpha$.*

*Proof.* Any transformation $\theta \in \mathrm{SE}(2) = \mathbb{R}^2 \rtimes \mathrm{SO}(2)$ can be written as $\theta x = R(x+t)$, with a rotation $R \in \mathrm{SO}(2)$ and a translation $t \in \mathbb{R}^2$. In order to compute $\frac{\mathrm{d}E}{\mathrm{d}\theta}$, we have to compute $\frac{\mathrm{d}E}{\mathrm{d}t}$ and $\frac{\mathrm{d}E}{\mathrm{d}R}$. $\left\| \frac{\mathrm{d}E}{\mathrm{d}\theta} \right\|^2$ can

then be computed via $\left\|\frac{\mathrm{d}E}{\mathrm{d}t}\right\|^2 + \left\|\frac{\mathrm{d}E}{\mathrm{d}R}\right\|^2$:

$$\begin{aligned}\left\|\frac{\mathrm{d}E}{\mathrm{d}t}\right\| &= \left\|\int_{B_\rho(0)} \left(\nabla g(x-t) - \nabla f(x-t)\right) q_\alpha(Rx)\mathrm{d}x\right\| \\ &\leq \int_{B_\rho(0)} \|(\nabla g(x-t) - \nabla f(x-t))\| q_{\mathrm{supp}}(x)\mathrm{d}x \\ &\leq \left[\int_\Omega \|\nabla f(x) - \nabla g(x)\|^2 \mathrm{d}x\right]^{1/2} \sqrt{\pi}\rho\end{aligned}$$

Since SO(2) is a Lie group, $\frac{\mathrm{d}E}{\mathrm{d}R}$ is computed by projecting the resulting matrix on the Lie algebra $\mathfrak{so}(2)$ of skew-symmetric matrices which can be represented by one single real-value:

$$\begin{aligned}\left\|\frac{\mathrm{d}E}{\mathrm{d}R}\right\| &= \left\|\int_{B_\rho(0)} (f-g)(x-t)\frac{\mathrm{d}}{\mathrm{d}R} q_\alpha(Rx)\mathrm{d}x\right\| \\ &= \left\|\int_{B_\rho(0)} (f-g)(x-t)\left[\nabla q_\alpha(Rx)x^T\right]_{\mathfrak{so}(2)}\right\|\end{aligned}$$

where $\left[\left(\begin{smallmatrix} a_{11} & a_{12} \\ a_{21} & a_{22}\end{smallmatrix}\right)\right]_{\mathfrak{so}(2)} = a_{12} - a_{21}$ represents the orthogonal projection of $A$ onto the Lie algebra $\mathfrak{so}(2)$

$$\begin{aligned}&= \left\|\int_{B_\rho(0)} (f-g)(x-t)\det\left(x, \nabla q_\alpha(Rx)\right)\mathrm{d}x\right\| \\ &\leq L_\chi \rho \int_\Omega |(f-g)(x)|dx\end{aligned}$$

with $q_{\mathrm{supp}}$ defined in (2.27). Since $\left\|\frac{\mathrm{d}E}{\mathrm{d}\theta}\right\|^2 = \left\|\frac{\mathrm{d}E_i}{\mathrm{d}R}\right\|^2 + \left\|\frac{\mathrm{d}E_i}{\mathrm{d}t}\right\|^2$, we have found a uniform upper bound for $\nabla E$ that does not depend on $(\theta, \alpha) \in \mathrm{SE}(2) \times \mathcal{A}_n$. Thus, $E$ is Lipschitz continuous in $\theta$ and the Lipschitz constant does not depend on $\alpha$. □

**Proposition 4.** *Under the above regularity assumptions on the training shapes, the segmentation* $\operatorname{argmin}_{(\alpha,\theta)\in\mathcal{A}_n\times \mathrm{SE}(2)} E(\alpha,\theta)$ *can be determined in a globally optimal manner.*

*Proof.* It suffices to prove that $E^*(\theta)$ is Lipschitz continuous. Let $L$ be the Lipschitz constant of $E(\alpha,\theta)$ and $\theta_1$, $\theta_2$ be two different transformations of SE(2). Then, there are two elements $\alpha_1, \alpha_2 \in \mathcal{A}_n$ fulfilling $E^*(\theta_1) = E(\alpha_1,\theta_1)$ and $E^*(\theta_2) = E(\alpha_2,\theta_2)$ resp., i.e.:

$$E(\alpha_1,\theta_1) \le E(\alpha_2,\theta_1) \qquad \wedge \qquad E(\alpha_2,\theta_2) \le E(\alpha_1,\theta_2.)$$

Using the first inequality, we obtain

$$\begin{aligned} E^*(\theta_2) - E^*(\theta_1) &= E(\alpha_2,\theta_2) - E(\alpha_1,\theta_1) \\ &\ge E(\alpha_2,\theta_2) - E(\alpha_2,\theta_1) \ge -L\,\mathrm{dist}(\theta_1,\theta_2), \end{aligned}$$

while the second one gives:

$$\begin{aligned} E^*(\theta_2) - E^*(\theta_1) &= E(\alpha_2,\theta_2) - E(\alpha_1,\theta_1) \\ &\le E(\alpha_1,\theta_2) - E(\alpha_1,\theta_1) \le L\,\mathrm{dist}(\theta_1,\theta_2). \end{aligned}$$

Thus, $E^*$ is Lipschitz continuous with the Lipschitz constant $L$. □

Overall, we showed how to acquire shapes with prior shape knowledge by globally optimizing a non-convex energy functional. In the next section, we will apply this method to acquire shapes from a walking person.

## 2.5 Tracking Walking People

In Section 2.4 we introduced an algorithm to acquire a shape from an image under the condition that we have previously learned the structure of the shape that we expect. This algorithm is based on

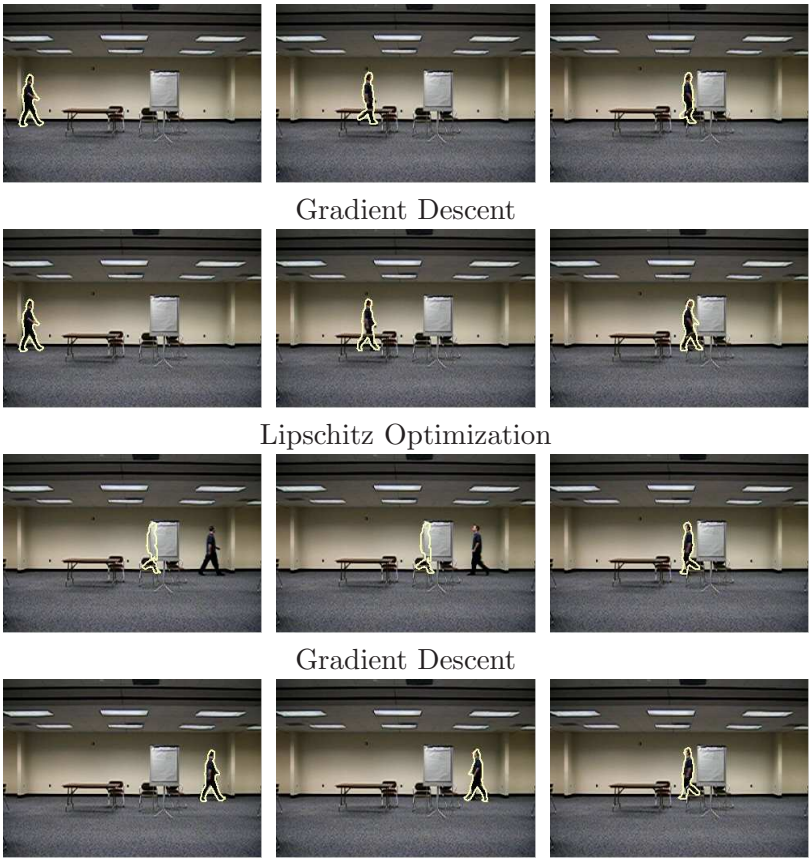



**Figure 2.8: Local versus global optimality.** Image segmentation with a dynamical shape prior, implemented by **gradient descent** (1st and 3rd row) and by **Lipschitz optimization** (2nd and 4th row). While gradient descent can handle partial occlusion by the table, it fails to handle total occlusion. The proposed Lipschitz optimization, on the other hand, guarantees the globally optimal solution and therefore reliably tracks the person upon reappearing from behind the white board.

convex minimization of deformation parameters interlaced with Lipschitz optimization of transformation variables.

To clarify the effect of the Lipschitz approach, we will show a comparison of the algorithm run without and with the Lipschitz optimization for a sequence showing a person walking in a cluttered scene. While more accurate results may be obtained with a user-specified stick-figure model, one should keep in mind that the proposed method does not require any user interaction in the model building. It can directly be applied to arbitrary stochastic shapes including purely binary shapes.

To this end, we construct a dynamical shape prior by hand-segmenting a different sequence (showing a different person walking at a different pace). By box-filtering these binary functions, we receive probabilistic shape functions that are Lipschitz continuous according to definitions 1 and 5. In order to reduce the dimensionality of the input data, we use $\mathcal{A}_6$ as the parameter space (cf. Figure 2.3).

As image energy, we use the approach (2.20) where $f(x)$ and $g(x)$ are the negative log probability for the observed intensity given that the pixel $x$ is part of the foreground or the background respectively. Neglecting the edge indicator term $h$ in (2.11), we receive the following energy functional:

$$E(\alpha, \theta) := \int_\Omega \log\left(\frac{p_{bg}(I)}{p_{ob}(I)}\right) q_\alpha(\theta x)\mathrm{d}x + \gamma\|\alpha - v_t\|^2_{\Sigma^{-1}}, \qquad (2.28)$$

where $\|\alpha - v_t\|^2_{\Sigma^{-1}} := (\alpha - v_t)^\top \Sigma^{-1}(\alpha - v_t)$ describes the energy of the dynamic shape prior – see equation (2.23).

During our experiments, we compare a pure gradient descent on $\alpha$ and $\theta$ with the proposed Lipschitz approach. Figure 2.8, 1st and 3rd row, shows that the pure gradient descent works well in the presence of partial occlusions such as the table. Yet, it fails to cope with larger occlusions where the local optimization gets stuck in a


**Figure 2.9: Detection of inconsistent solutions.** The plot on the left shows the image energy $E_{\mathrm{TV}}$ as a function of the frame number. Red crosses indicate the four frames shown on the right. Incorrect segmentation results due to a total occlusion of the object of interest can be automatically identified and suppressed (2nd and 3rd frame).

local minimum with respect to $\theta$. In addition, the gradient descent approach obviously requires an appropriate initialization. Both of these drawbacks are resolved by the proposed global optimization based on the combination of convexity and Lipschitz optimization – see Figure 2.8, $2^{\mathrm{nd}}$ and $4^{\mathrm{th}}$ row.

In Figure 2.9, we show that our algorithm also provides a reliable criterion to determine whether a computed result is consistent with data or not: Reliable segmentations correspond to low (negative) energy, while unreliable ones (full occlusion) correspond to high (positive) energy. As a consequence, we can detect the full occlusion of the white board. For the affected frames, the shape minimizer (2.28) is not meaningful in the sense that the appropriate shape cannot be found in the image. Hence, we can omit the minimizer of the shape functional. Therefore, we only acquire meaningful shapes with respect to (2.28). This demonstrates that a person can be reliably tracked through clutter and occlusions without the need to reinitialization.

## 2.6 Limitation of $L^2$-distances

In this chapter, we showed how a region-based shape model can be a powerful tool to acquire shapes from images. Nonetheless, the involved region-based $L^2$-metric is a rather primitive metric. If two shapes $\ell_1, \ell_2 \in \mathfrak{Q}$ are binary, i.e., if they are of the form (2.2), the following holds:

$$\|\ell_1 - \ell_2\|^2 = \text{area}\left(\{x \in \Omega | \ell_1(x) = 1\} \triangle \{x \in \Omega | \ell_2(x) = 1\}\right) \quad (2.29)$$

This means that the $L^2$-metric measures the symmetric difference of the two given shapes. Therefore, the $L^2$-distance has two important drawbacks:

**Sensitivity to Local Transformations:** Local transformations appear when the shape in question represents an object that consists of different parts which can be moved independently. In the example of a walking person, the legs and arms can be moved independently. For the person, these are only small changes. With respect to the $L^2$-distance, this may have a large effect if the extremities of the person exist in non overlapping areas of the image domain.

**Sensitivity to Local Deformations:** To recognize similar objects, we cannot always assume that the shape looks every time exactly the same. Even if the same person is present in the images, he/she may have changed clothes and as a consequence he/she appears thicker in winter or thinner in summer. But to have a general system that recognizes whether a given shape is the shape of *a person* and not only of *a specific person at a specific time*, this system should be robust with respect to local deformations.

For the presented shape acquisition, these drawbacks could be circumvented by applying a Markov chain model. Also, the image data

dominated the shape acquisition process. Note that the shape prior was only used to *repair* the shape acquisition. The most important data came from the images. This means that the shape prior was just used to improve the performance of a purely data driven shape acquisition method. In the next chapters, we will compare different shapes. Therefore, we have to find a method that compares shapes at a totally absence of any additional data. We are still interested in a space equipped with a distance function. But due to the mentioned drawbacks of the $L^2$-metric, we will use different spaces. In Chapter 3, a metric is defined by the shortest path length within an infinite dimensional manifold that defines a shape space. This shape space is more sophisticated than the stochastic shape space $\mathcal{Q}$. In Chapter 4, a pseudo-metric is defined by studying the group of shape diffeomorphisms. Such a diffeomorphism is also known as a matching. We will show that the pseudo-metric induced by the matching function is a very helpful tool to classify different shapes.

# Chapter 3

# Shape Morphing

In the previous chapter, the similarity between two shapes was computed as a region-based $L^2$-distance (cf. (2.14)). In this chapter, we want to advocate the philosophy that a shape is best described by its outer boundary, which can be represented by a contour in the plane. We will see that this concept of shape is much more robust with respect to local deformations. As a result, we do not have to model the most likely deformation to introduce a meaningful measure of similarity as it was done in the last chapter in the mean of the Markov chain model (cf. (2.22) and (2.23)). On the other hand we ignore possible holes inside of the shape (see also Figure 1.2).

A lot of researchers have put a considerable amount of effort into the understanding of shapes. Measuring the dissimilarity between two given shapes can be done by defining and examining metric spaces which model shapes (cf. [32, 26, 28, 6]). In this chapter we follow the idea that a contour is a smooth loop in the plane. Thus, we like to consider all mappings $c : \mathbb{S}^1 \to \mathbb{C}$ that assign a point in the plane $\mathbb{C}$ to every point $s$ of the circle $\mathbb{S}^1 = \{s \in \mathbb{C} || s| = 1\}$. Every closed loop can be described as such a mapping $c$. In this chapter, we will make use of the algebraic structure of $\mathbb{C}$. Therefore, the

plane is modeled as $\mathbb{C}$ instead of $\mathbb{R}^2$ as it was done in the previous chapter. In order to consider shapes which are independent with respect to translation or rotation in $\mathbb{C}$, we have to merge different representations of a shape. This results in a shape space $\mathcal{S}$ whose *elements* are described as sets of contours. Each of these sets is a class of equivalent contours that describe a *shape*. Hence, we have to define an equivalence relation which will be done via a group that operates on the set of contours. In this chapter, we will present different models of such shape spaces which have all in common that they do not form a convex set or even an affine vector space. But even for these spaces, we can define a metric by considering the paths $m : [0;1] \to \mathcal{S}$ that connect two different shapes $C_0 = m(0)$ and $C_1 = m(1)$. The distance $\operatorname{dist}(C_0, C_1)$ between $C_0$ and $C_1$ will then be defined as the shortest path length of such a path $m$ that connects $C_0$ and $C_1$. We call such a path $m$ a morphing because it describes how one shape $C_0$ is continuously transformed into $C_1$. In this chapter, we will present a variational method that computes such a morphing very efficiently.

Our work is built on several prior works. Michor and Mumford introduced in [66] a shape space $\mathcal{S}$ that is path-connected. Therefore, it makes sense to look for the shortest path that connects two different shapes. Since $\mathcal{S}$ is defined as an orbifold[1], every path can be measured and the shortest length of a shape-connecting path can be computed. But unfortunately, a quite canonical approach results in the trivial shape distance function $\operatorname{dist}(\cdot,\cdot) \equiv 0$. In order to cope with this result, Michor and Mumford introduced another approach that depends locally on the shapes' curvature. Since the computation of the shortest path results in solving a challenging *partial differential equation*, we will follow another approach.

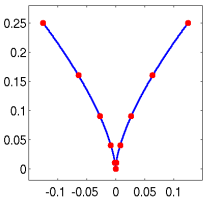
This approach was advocated by Klassen *et al.* [51]. Instead of con-

[1] An orbifold can be understood as a manifold modulo a group operation. For a detailed introduction see [92]

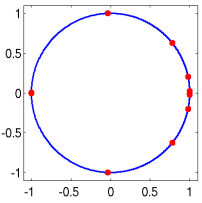
sidering all representations of a shape, they only considered those curves $c : \mathbb{S}^1 \to \mathbb{C}$ which are parameterized by arc-length. As a consequence, they received a shape space $\mathcal{S}_1$ on which shortest paths are much easier to compute than on $\mathcal{S}$, since it results in solving multiple ordinary differential equations. As a consequence, the calculation of a morphing path could in many cases be done within seconds using the so-called *shooting method*. This method uses a searching beam from the initial shape. That beam will be changed until it hits the target shape. During this transition from the initial shape to the target shape, the beam is deformed according to the underlying metric just as a light beam is bent by gravity in the theory of general relativity. Computationally, this method is still quite expensive but faster than the method proposed by Michor and Mumford.

The focus of this work is it to show how these shortest paths can be computed even more efficiently by minimizing the involved energy functional. This method is a gradient descent approach which converges quite rapidly in comparison to the shooting method. This chapter is organized as follows. In Section 3.1, we will present the two shape spaces $\mathcal{S}$ and $\mathcal{S}_1$. In Chapter 3.2, we will address the differential geometrical concepts of *submanifolds*, *tangent spaces* and *geodesics*. In order to make these concepts accessible to a broader audience, we will omit the concepts of *local charts* and *Riemannian metrics*. Instead, we concentrate our efforts on submanifolds that are isometrically embedded in Euclidean vector spaces.

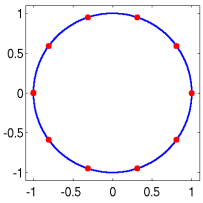
In Chapter 3.3, we will present our concept of computing a geodesic efficiently by consecutively shortening a path until it becomes a geodesic. We finalize this chapter with a comparison of our method to the shooting method. It turns out that our variational path-shortening method will be faster by a factor of up to 1000 depending on the used shape resolution.




$t \mapsto t^3 + i \cdot t^2$




$t \mapsto \exp(i \cdot t^2)$




$t \mapsto \exp(i \cdot t)$

**Figure 3.1: Immersion.** Smooth mappings $c : [0;1] \to \mathbb{C}$ into the plane $\mathbb{C}$ do not necessarily define a smooth contour. If $c$ is an immersion (3rd image), $c$ describes always a smooth contour.

## 3.1 Shape Spaces as Orbifolds

In this section we will present a shape model that differs from the model of stochastic shapes in the last chapter in the sense that we only consider shapes that consist of one connected, smooth boundary. This boundary can therefore be described via a contour $c : \mathbb{S}^1 \to \mathbb{C}$. To guarantee that $c$ describes a *smooth* contour, $c$ has to be an *immersion* (cf. Figure 3.1):

**Definition 6.** A mapping $c : \mathbb{S}^1 \to \mathbb{C}$ is called an *immersion* if it fulfills the following properties:

1. $c$ is a *smooth* mapping, i.e. for every $k \in \mathbb{N}$ the derivative $c^{(k)} : \mathbb{S}^1 \to \mathbb{C}$ exists.

2. $c'$ is *invertible*, i.e. the inequality $c'(s) \neq 0$ holds for every $s \in \mathbb{S}^1$.

The set of all immersions from $\mathbb{S}^1$ into $\mathbb{C}$ is denoted as $\mathrm{Imm}(\mathbb{S}^1, \mathbb{C})$. To every immersion, we define the *immersion index*. This is the winding number of the derivative $c'$ with respect to $0 \in \mathbb{C}$ (cf. Figure 3.2):

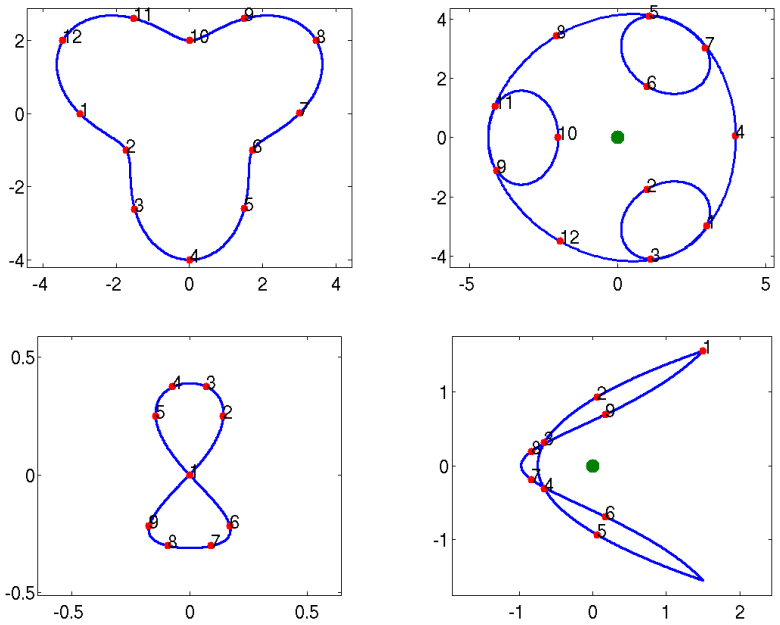



**Figure 3.2: Immersion Index.** Two contours (1st column) and their derivatives (2nd column) are shown. The contour of the 1st row has an immersion index of 1 while the contour of the 2nd row has an immersion index of 0. Hence, the contours cannot be transformed into one another.

$$\mathrm{ind}(c) = \frac{1}{2\pi} \int_{c'} \frac{x\mathrm{d}y - y\mathrm{d}x}{x^2 + y^2} \in \mathbb{Z} \tag{3.1}$$

The set of all immersion of a specific index $k \in \mathbb{Z}$ is denoted as $\mathrm{Imm}^k(\mathbb{S}^1, \mathbb{C})$. The set of all immersions $c \in \mathrm{Imm}^k(\mathbb{S}^1, \mathbb{C})$ that are parameterized by arc-length is denoted as $\mathrm{Imm}_1^k(\mathbb{S}^1, \mathbb{C})$. These are immersions for which $\|c'\| \equiv 1$ holds.

It follows from Algebraic Topology [44] that contours can not be transformed into one another if they differ by their immersion index. Also, if $c \in \mathrm{Imm}(\mathbb{S}^1, \mathbb{C})$ describes the boundary of a binary shape, the immersion index is either $-1$ or $+1$ depending on whether the boundary is swept in the clockwise or the counter clockwise sense respectively. Therefore in order to compute a morphing, we are in the following only interested in immersions of index 1. If a shape is now given in a binary form, the representing contour $c$ can be easily derived by starting at an arbitrary point of the boundary and following the boundary in a counter clockwise sense. Even though the orientation of the curve is fixed, the curve is still quite ambiguous in the sense that there is no natural way to define *the starting point* of the described sweeping process. Moreover, if we reparameterize the circle $\mathbb{S}^1$ via *any* differentiable mapping $\varphi : \mathbb{S}^1 \to \mathbb{S}^1$, we receive another curve $c \circ \varphi$ that describes the same contour as $c$. Since, we are only interested in reparameterizations that lead to another *immersion*, $\varphi$ itself has to be a diffeomorphism:

**Definition 7.** A mapping $c : \mathbb{S}^1 \to \mathbb{S}^1$ is a diffeomorphism if it fulfills the following properties:

1. $c$ is a bijective mapping.
2. $c$ and $c^{-1}$ are $C^\infty$-mappings.

The set of all diffeomorphisms of the circle is denoted as $\mathrm{Diff}(\mathbb{S}^1)$. Any of these diffeomorphisms describes a loop on $\mathbb{S}^1$ that passes $\mathbb{S}^1$ either in a counter clockwise sense or in a clockwise sense. The set of counter clockwise passing diffeomorphisms is denoted as $\mathrm{Diff}^+(\mathbb{S}^1)$ and the set of clockwise passing diffeomorphisms is denoted as $\mathrm{Diff}^-(\mathbb{S}^1)$.

Together with the composition, the sets $\mathrm{Diff}(\mathbb{S}^1)$ and $\mathrm{Diff}^+(\mathbb{S}^1)$ form groups, but from these both groups, only $\mathrm{Diff}^+(\mathbb{S}^1)$ acts on $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ as reparameterization group. This is because a diffeomorphism $\varphi \in \mathrm{Diff}^-(\mathbb{S}^1)$ maps any immersion $c \in \mathrm{Imm}^k(\mathbb{S}^1, \mathbb{C})$ onto an immersion of $\mathrm{Imm}^{-k}(\mathbb{S}^1, \mathbb{C})$. According to these observations, we can get rid of possible ambiguities of the presented shape representation by considering the quotient of $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ and $\mathrm{Diff}^+(\mathbb{S}^1)$:

**Definition 8** ([66])**.** The space of *immersion based shapes* is defined as the orbifold

$$\mathcal{S} := \mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C}) / \mathrm{Diff}^+(\mathbb{S}^1) \tag{3.2}$$

Within the space $\mathrm{Imm}^1_1(\mathbb{S}^1, \mathbb{C})$, there exists the following set

$$S = \{s \mapsto \tau \cdot s | \forall \tau \in \mathbb{S}^1\} \tag{3.3}$$

which contains multiple representations of the circle each of these only differs by its starting point. Furthermore, $S$ itself is a circle embedded into $\mathrm{Imm}^1_1(\mathbb{S}^1, \mathbb{C})$ and parameterized via $\tau$.

In [66] it was shown that $S$ is a *smooth, strong deformation retract* of the space $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$. Such a retract is a subset that contains most of the relevant topological properties. The mathematical definition is the following:

**Definition 9.** A subset $A \subset X$ of a topological space $X$ is called a *smooth strong deformation retract*, if there exists a smooth function $r : [0; 1] \times X \to X$ such that

- $r(0, x) = x$ for all $x \in X$.
- $r(1, x) \in A$ for all $x \in X$.
- $r(t, a) = a$ for all $a \in A$ and $t \in [0; 1]$.

A *smooth, strong deformation retraction* $r$ moves therefore any element $x$ of the superset $X$ smoothly on an element in the subset $A$. A lot of interesting topological properties can be transported from $A$ towards $X$. But first let us recapitulate what was shown by Michor and Mumford for the presented sets of immersions[2]:

**Theorem 1** ([66])**.** *The following two declarations hold:*

- $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ *is a* smooth strong deformation retract *of* $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$.
- $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ *contains the circle S of* (3.3) *as a* smooth strong deformation retract.

As a consequence, we can show that $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ and $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ are path-connected spaces. This means that to arbitrary elements of the respective sets, we can find a path within this set that connects the two elements:

**Corollary 1.** $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ *and* $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ *are path-connected spaces.*

*Proof.* Since the circle S of (3.3) is obviously path-connected, it suffices to show the following: *If A is a smooth, strong, path-connected retract of a topological space X, X itself is path-connected.* If we can show this, $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ is path-connected because $\mathbb{S}^1$ is path-connected and as a consequence, $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ is path-connected because $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ was already path-connected.

[2]Out of simplicity, we restrict ourselves to the properties of the index-1-immersions

Hence, let us assume that $r : [0;1] \times X \to X$ is a *smooth strong deformation retraction* from $X$ onto the path-connected space $A \subset X$. Further, let $x, y \in X$ be arbitrary but fixed elements of the superset $X$. Since $A$ is a path-connected space, there exists a path $p : [0;1] \to A$ that connects $a_x := r(1,x) \in A$ with $a_y := r(1,y) \in A$. Now, we define the following path $q$ in $X$:

$$\begin{aligned} q : [0;1] \to & X \\ t \mapsto & \begin{cases} r(3t,x) & \text{, if } t \in \left[0;\frac{1}{3}\right] \\ p(3t-1) & \text{, if } t \in \left[\frac{1}{3};\frac{2}{3}\right] \\ r(3-3t,y) & \text{, if } t \in \left[\frac{2}{3};1\right] \end{cases} \end{aligned}$$

Since $q(0) = r(0,x) = x$ and $q(1) = r(0,y) = y$, $q$ starts in $x$ and ends in $y$. In order to show that this is really a continuous path from $x$ to $y$, we have just to show that the path is well defined at the points $t = \frac{1}{3}$ and $t = \frac{2}{3}$:

$$\begin{aligned} r\left(3\frac{1}{3}, x\right) = r(1,x) = & a_x = p(0) = p\left(3\frac{1}{3} - 1\right) \\ p\left(3\frac{2}{3} - 1, x\right) = p(1) = & a_y = r(1,y) = r\left(3 - 3\frac{2}{3}, x\right) \end{aligned}$$

□

Note that any path $m : [0;1] \to \mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ defines a path $\widetilde{m} : [0;1] \to \mathcal{S}$ on the quotient space $\mathcal{S}$ by assigning to every $t \in [0;1]$ the equivalence class of $m(t)$, namely

$$\widetilde{m}(t) = [m(t)],$$

where $[m(t)]$ denotes the equivalence class in $\mathcal{S}$ that itself contains the immersion $m(t)$. Hence, the shape space $\mathcal{S}$ is like $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ a path-connected space and it makes sense to define the similarity

between two shapes as the length of the shortest path between two shapes. In order to compute the similarity between two shapes as efficient as possible, we want to focus on pure $L^2$-distances. Since it was shown in [66] that a canonical $L^2$-distance for $\mathrm{Imm}^1(\mathbb{S}^1, \mathbb{C})$ leads to a trivial metric on $\mathcal{S}$, we will now turn our attention on a shape space that is based on contours modeled via $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$, i.e., the contours have to be parameterized by arc-length. This shape concept was advocated by Klassen and his coworkers [51]:

1. Since this shape concept is based on contours which are parameterized by arc-length, $\mathrm{Diff}^+(\mathbb{S}^1)$ is not a group operation any more. This is because an arbitrary reparameterization $\varphi \in \mathrm{Diff}^+(\mathbb{S}^1)$ may change the speed of a contour. In fact, if $c$ is parameterized by arc-length, the following holds for $c \circ \varphi$:

$$\begin{aligned}\left\| \frac{\mathrm{d}}{\mathrm{d}s} c \circ \varphi(s) \right\| =& \left\| c'(\varphi(s)) \cdot \varphi'(s) \right\| \\ =& |\varphi'(s)| \cdot \underbrace{\left\| c'(\varphi(s)) \right\|}_{=1} = \varphi'(s)\end{aligned}$$

   We used the property that for any $\varphi \in \mathrm{Diff}^+(\mathbb{S}^1)$, $\varphi' > 0$ holds. To guarantee that $c \circ \varphi$ is also parameterized by arc-length, we allow therefore only those reparameterizations $\varphi$ for which $\varphi' \equiv 1$ holds. Therefore, a reparameterization is only allowed to change the *starting point* of the contour. As a result, the reparameterization group acts now as $\mathbb{S}^1$. To $c \in \mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ and $\tau \in \mathbb{S}^1 \subset \mathbb{C}$, the reparameterized contour $c_\tau$ is then defined as:

$$\begin{aligned} c_\tau : \mathbb{S}^1 \to& \mathbb{C} \\ \theta \mapsto& c(\theta \cdot \tau) \end{aligned} \tag{3.4}$$

2. Secondly, the modeled shapes should be invariant with respect to rigid body transformations. In the real plane $\mathbb{R}^2$ such as

transformation consists of a rotation $R = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \in \mathrm{SO}(2)$ and a translation $t \in \mathbb{R}^2$. Since we model the plane as $\mathbb{C}$, $R$ becomes a multiplication with the complex number $\cos(\alpha) + i \cdot \sin(\alpha) \in \mathbb{S}^1$ and $t$ is an element of $\mathbb{C}$. If a transformation $(R, t) \in \mathrm{SE}(2) = \mathbb{C} \rtimes \mathbb{S}^1$ is now given, the contour $s \mapsto c(s)$ and the contour $s \mapsto R \cdot (c(s) + t)$ should describe the same shape. The shape model that we intend to use will also eliminate these ambiguities.

Note that we have now two group operations that we like to eliminate. The reparameterization group that acts as $\mathbb{S}^1$ from the right hand side on the contour set $\mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$ and the rigid body transformation group that acts as $\mathrm{SE}(2)$ from the left hand side on the contour set. Overall, we are therefore interested in the following two spaces:

**Definition 10.** The space of *arc-length based preshapes* is defined as

$$\mathcal{C}_1 = \mathrm{SE}(2) \setminus \mathrm{Imm}_1^1(\mathbb{S}^1, \mathbb{C})$$

and the space of parameterization free *arc-length based shapes* is defined as

$$\mathcal{S}_1 = \mathcal{C}_1 / \mathbb{S}^1$$

Since every shape is parameterized by arc-length, every shape has a length of exactly $2\pi$. As a result, the presented shape model is also invariant with respect to rescaling. This takes into account that the size of a photographed object reflects the distance of the camera to the object. Hence, there is no relation between the observed size and the real size of the object. Also, Galilei [37] showed that objects that are *in reality* smaller exhibit another corporal structure. Our shape

concept would therefore still be able to differentiate this change of size.

It was shown in [51] that the preshape space $\mathcal{C}_1$ can be understood as a submanifold of the vector space $L^2([0;2\pi],\mathbb{R})$. Here, we want to repeat this construction briefly (cf. Figure 3.3):

1. Since $c \in \mathrm{Imm}_1^1(\mathbb{S}^1,\mathbb{C})$ is parameterized by arc-length, its derivative is a $C^\infty$-function $c' : \mathbb{S}^1 \to \mathbb{S}^1$ which can be encoded by a function $\theta : [0;2\pi] \to \mathbb{R}$ such that the following lifting equality holds [44]: $c'\left(e^{i\cdot x}\right) = e^{i\cdot\theta(x)}$. If we consider now $\theta$ instead of $c$, we obtain a representation that is invariant under translations in the plane $\mathbb{C}$.

2. To describe $\mathcal{C}_1$ we have now to get rid of possible rotations in the plane. If we apply a rotation $R = \cos(\alpha) + i\sin(\alpha)$ on a curve $c$ with its shape representative $\theta$, we receive the following equalities:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}s}\left[R\cdot c(s)\right] =& e^{i\cdot\alpha}\cdot c'(s) = e^{i\cdot\alpha}\cdot e^{i\theta(x)} \qquad \text{, with } s = e^{ix}\\ =& e^{i(\alpha+\theta(x))}\end{aligned}$$

   Hence, a rotation on $c$ acts now as addition on $\theta$. We can now fix a possible rotation by demanding for the following equality to hold:

$$\int_0^{2\pi} \theta(x)\mathrm{d}x = 2\pi^2 \tag{3.5}$$

   To choose $2\pi^2$ as constant is quite arbitrary and we could choose any other constant. But it was chosen in order to make the function $x \mapsto x$ feasible. This function is the $\theta$-representation of the shape "Circle".

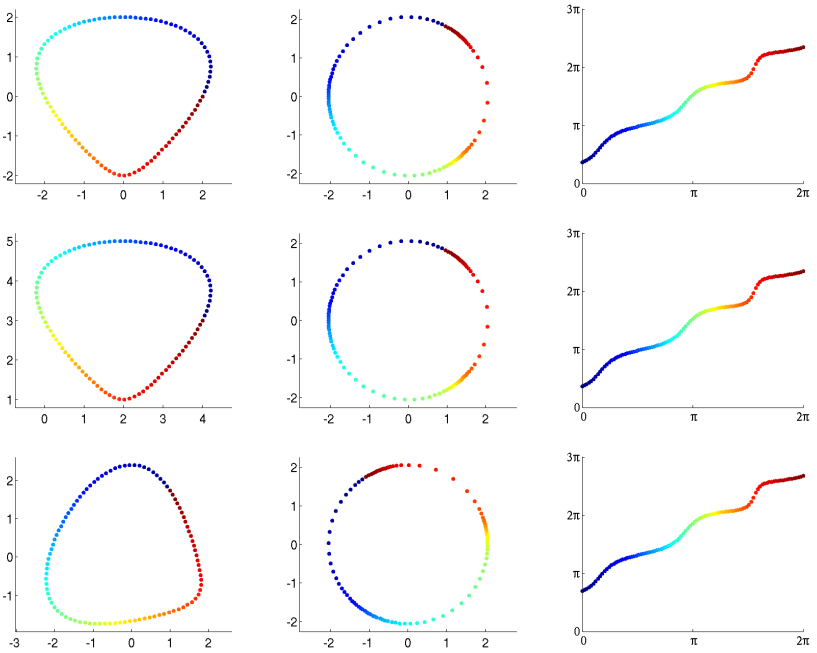



**Figure 3.3: Shape Representation.** Contours (1st column), its first derivatives (2nd column) and the $\theta$-representation (3rd column) are depicted. Translations of a contour (2nd row) are filtered out by the $\theta$-representation while a rotation (3rd row) results in a translation of the $\theta$-representation.

3. Also note that the immersion index of the immersion $c$ is also reflected by $\theta$ in the following sense:

$$\theta(2\pi) - \theta(0) = 2\pi \cdot \mathrm{ind}(c) = 2\pi$$

Therefore, the $\theta$-functions that describe elements of $\mathcal{C}_1$ are of the following affine $L^2$-space:

$$L := \big\{\theta : [0; 2\pi] \to \mathbb{R} \big| \theta(0) = \theta(2\pi) + 2\pi \wedge \\ \theta^{(k)}(0) = \theta^{(k)}(2\pi) \text{ for all } k > 0\big\}$$

4. In order to make sure that $\theta$ really represents the derivative of a contour, we have to *close the loop*. This means that we allow only those $\theta$ functions that describe a closed loop. Therefore, the following must hold:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \int_{\mathbb{S}^1} c'(s)\mathrm{d}s = \int_0^{2\pi} e^{i\theta(x)}\mathrm{d}x = \begin{pmatrix} \int_0^{2\pi} \cos(\theta(x)\mathrm{d}x \\ \int_0^{2\pi} \sin(\theta(x)\mathrm{d}x \end{pmatrix}$$

Concluding all these observations, $\mathcal{C}_1$ can be modeled as follows.

**Theorem 2** ([51])**.** *If we define the function* $\Psi : L \to \mathbb{R}^3$ *via*

$$\begin{aligned} \Psi : L \to &\mathbb{R}^3 \\ \theta \mapsto &\begin{pmatrix} \int_0^{2\pi} \theta(x)\mathrm{d}x \\ \int_0^{2\pi} \cos(\theta(x))\mathrm{d}x \\ \int_0^{2\pi} \sin(\theta(x))\mathrm{d}x \end{pmatrix}, \end{aligned}$$

$\mathcal{C}_1$ *can be modeled as the submanifold* $\Psi^{-1}(2\pi^2, 0, 0) \subset L \subset L^2([0; 2\pi], \mathbb{R})$.

As a consequence, the shape space $\mathcal{S}_1$ is like $\mathcal{S}$ an orbifold. In the following section, we will address the problem of finding a shortest path on submanifolds like $\mathcal{C}_1$. This will result in the very prominent

concept of the so-called shooting method. In the subsequent section, we will present an alternative method that is much faster than the shooting method. In fact, this path-shortening method is faster by a factor of up to 1000 depending on the shapes' resolution. Additionally, we will show how the concept of shortest paths in $\mathcal{C}_1$ can be extended to find shortest paths in $\mathcal{S}_1$.

## 3.2 Geodesics on Submanifolds

In this section, we will present the idea of geodesics and the prominent shooting method to compute such a geodesic. In the following, we will restrict ourselves to submanifolds $M$ that can be described as a subset of an Euclidean vector space $\mathbb{E}$ like the submanifold $\mathcal{C}_1$ that is a subset of the Euclidean vector space $L^2([0;2\pi],\mathbb{R})$. Since such a vector space is equipped with a scalar product $\langle\cdot\,,\,\cdot\rangle$ , we can compute the length of any smooth path $m : [0;1] \to \mathbb{E}$ via:

$$\text{length}(m) = \int_0^1 \left\langle m'(t)\,,\, m'(t)\right\rangle^{\frac{1}{2}} \mathrm{d}t \tag{3.6}$$

In order to find the shortest path between two points $x, y \in \mathbb{E}$, often the minimum of the following energy functional is considered

$$E(m) = \int_0^1 \left\langle m'(t)\,,\, m'(t)\right\rangle \mathrm{d}t \tag{3.7}$$

This is because every minimizer of $E(\cdot)$ is also a minimizer of length$(\cdot)$. Moreover, the global minimum of $E(\cdot)$ is a shortest path between $x$ and $y$ which is parameterized uniformally, i.e., $\left\|\frac{\mathrm{d}}{\mathrm{d}t}m^*(t)\right\|$ is constant:

**Theorem 3.** *If $m_0$ and $m^*$ are global minimizers of* length$(\cdot)$ *and $E(\cdot)$ resp., then*

1. $\left\|\frac{\mathrm{d}}{\mathrm{d}t}m^*(t)\right\| \equiv \text{length}(m^*)$

*2.* $\text{length}(m^*) = \text{length}(m_0)$

*Proof.* First, we observe the following inequality which is derived from the Cauchy-Schwarz inequality applied to an $L^2$-space:

$$\begin{aligned}\text{length}(m) &= \int_0^1 \left\|m'(t)\right\| \cdot 1 \mathrm{d}t \\ &\leq \left[\int_0^1 \left\|m'(t)\right\|^2\right]^{\frac{1}{2}} \cdot \left[\int_0^1 1^2\right]^{\frac{1}{2}} = E(m)^{\frac{1}{2}}\end{aligned}$$

This inequality becomes an equality if and only if the two functions $\|m'(t)\|$ and 1 are linearly dependent, i.e., if $m$ is parameterized uniformally.

1. If $m^*$ differs from its uniformally parameterized instance $\widetilde{m}$, we receive:

   $$E(\widetilde{m})^{\frac{1}{2}} = \text{length}(\widetilde{m}) = \text{length}(m^*) < E(m^*)^{\frac{1}{2}}$$

   This is a contradiction to the minimality of $m^*$. Therefore, $m^*$ must already be uniformally parameterized, i.e., $\left\|\frac{\mathrm{d}}{\mathrm{d}t} m^*(t)\right\| \equiv: a$ such that:

   $$\text{length}(m^*) = \int_0^1 \left\|\frac{\mathrm{d}}{\mathrm{d}t} m^*(t)\right\| \mathrm{d}t = \int_0^1 a \mathrm{d}t = a$$

2. Denoting the uniformally parameterized instance of $m_0$ as $\widetilde{m}_0$, we receive

   $$\begin{aligned}\text{length}(m_0) &= \text{length}(\widetilde{m}_0) = E(\widetilde{m}_0)^{\frac{1}{2}} \\ &\geq E(m^*)^{\frac{1}{2}} = \text{length}(m^*) \geq \text{length}(m_0)\end{aligned}$$

   Since the first and the last expression describe the same value, every inequality has to be an equality and $\text{length}(m^*) = \text{length}(m_0)$. □

This theorem shows that in order to find the shortest path between two elements of the Euclidean space $\mathbb{E}$, it suffices to minimize the functional $E(\cdot)$ instead of the functional length$(\cdot)$. Additionally by optimizing $E(\cdot)$, we obtain a path that is uniformally parameterized. Especially from an implementation point of view this parameterization has an important advantage. Since a path is normally stored in a discretized version, it comes in very handy that we always obtain a path that is parameterized uniformally. As a consequence, we are sure that no range of this path is oversampled.

Now we want to extend this concept of shortest paths on submanifolds. These are *smooth* subsets $M$ of an Euclidean space $\mathbb{E}$, e.g., a sphere or a cylinder in $\mathbb{R}^3$. To any point $x \in M$ of such a submanifold $M$, we obtain a tangent space, denoted as $T_xM$ such that for any smooth path $m : [0;1] \to M$, the following holds:

$$m'(t) \in T_{p(t)}M \qquad \text{, for all } t \in (0;1)$$

Hence, all possible *directions* of a path starting in $x$ are stored in $T_xM$. We assume that the tangent spaces at any base point $x$ are of the same dimension $k$ and we call $k$ the dimension of the submanifold $M$. A closed non-intersecting loop in $\mathbb{R}^3$ is therefore a one-dimensional submanifold and a sphere is a two-dimensional submanifold. For a more formal introduction to submanifolds and manifolds, we like to refer to [31, 21].

Since every tangent space $T_xM$ is a subset of $\mathbb{E}$, the scalar product of $\mathbb{E}$ can also be applied to vectors in $T_xM$. As a consequence, the functionals length$(\cdot)$ and $E(\cdot)$ are also defined for paths on a submanifold. This leads to the following metric on submanifolds

**Definition 11.** Let $M \subset \mathbb{E}$ be a path-connected submanifold of an Euclidean space $\mathbb{E}$. To two different points $x, y \in M$ of this submanifold, we assign the distance:

$$\operatorname{dist}_M(x,y) := \min_{\substack{m \text{ smooth path,} \\ m(0)=x,\, m(1)=y}} \operatorname{length}(m) \tag{3.8}$$

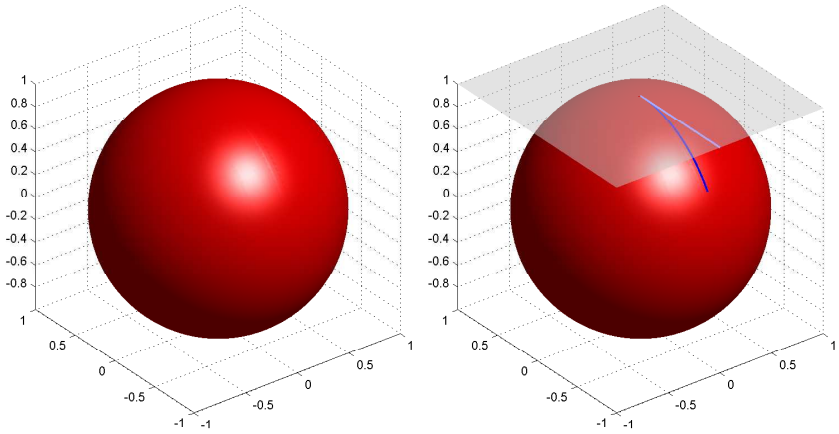



**Figure 3.4: Tangent Space and Exponential Mapping.** To any point $p$ of a manifold $M$ ($1^{\text{st}}$ image), a tangent space $T_pM$ can be defined. The exponential mapping maps every straight line in $T_pM$ passing through $0 \in T_pM$ onto a geodesic on $M$ passing through $p$ ($2^{\text{nd}}$ image).

It can be shown that $\text{dist}_M(\cdot,\cdot)$ is a metric on $M$ and the goal of this chapter is to compute this distance. The observations of Theorem 3 lead us to the following definition of a geodesic:

**Definition 12.** Given a submanifold $M \subset \mathbb{E}$ of an Euclidean space $\mathbb{E}$, a path $m : [0;1] \to M$ is called a geodesic if it fulfills the *Euler-Lagrange equation* with respect to the functional $E(\cdot)$.

If $M$ itself is a vector space of dimension $k$, the Euler-Lagrange equation of the functional $E$ is quite simple and results in

$$0 \equiv m''$$

Here, $m'$ and $m''$ are $k$-dimensional vector fields along $m$. But in the case of a submanifold, only $m'$ is a $k$-dimensional vector field,

i.e., $m'(t) \in T_{m(t)}M$ for every $t \in (0;1)$. For $m''$, this property does not hold any longer. But we can split $m''$ into a tangential ($k$-dimensional) vector field $m''^{\tan}$ and a normal vector field $m''^{\mathrm{nor}}$. Using the concept of Lagrangian multipliers, the *Euler-Lagrange equation* for submanifolds becomes:

$$0 \equiv m''^{\tan}$$

In order to compute a geodesic, an ordinary differential equation can be solved. In fact, given a starting point $x \in M$ and a starting direction $v \in T_xM$, the following differential equation

$$m(0) = x \qquad m'(0) = v \qquad m''^{\tan}(t) \equiv 0 \tag{3.9}$$

is an *initial value problem*. This problem fulfills the *Picard-Lindelöf* conditions and has a path $m_{x,v}\colon \mathbb{R} \to M$ as unique solution. This leads to the definition of the so-called exponential mapping

$$\begin{aligned} \exp_x : T_xM &\to M \\ v &\mapsto m_{x,v}(1) \end{aligned}$$

While the shooting method used in [51] makes use of the exponential mapping, the variational method that we will propose in Section 3.3 directly relies on the definition of a geodesic and is computed by minimizing the energy functional $E(\cdot)$. Our approach has several advantages over the previously used shooting method:

- First of all, we do not rely on computing the ordinary differential equation (3.9). As a consequence, our approach turns out to be much faster than the commonly used *shooting method*. Depending on the resolution of the shortest path, i.e., the amount of intermediate shapes that we compute, we receive an runtime acceleration of a factor of up to 1000.

- Since we do not have to approximate the solution of the ordinary differential equation, the proposed *path shortening* method is numerically much more stable. If we approximate (3.9) linearly like in [51], small errors may be accumulated, so that at the end we do not receive a path which is parameterized in a purely uniform manner. But for the path shortening method, the uniform parameterization is according to Theorem 3 a byproduct of the optimization process. In fact, errors are always dampened in every iteration step.

- Another important property of the method that we will propose in Section 3.3 is that it is symmetric in the sense that the shortest path from $x$ to $y$ is always the same as the shortest path from $y$ to $x$. From a theoretical point of view this may apply to every method that tries to compute a geodesic between two points. But the numerical stability of the shooting method relies on the curvature of the manifold $M$ at the starting point $x$. If this curvature differs at $x$ and $y$ the shortest path from $x$ may differ from the shortest path starting at $y$ if we use the shooting method. This restriction does not hold if one would use the path-shortening method of the following section.

## 3.3 Path-Shortening Method

In this section, we will present our method of computing a geodesic between two given points $x$ and $y$ on a submanifold $M$ of an Euclidean space $\mathbb{E}$. First, we will start with the case that $\mathbb{E}$ is an arbitrary finite dimensional space. After explaining our method for this rather general case, we will address the problem of finding a geodesic in the submanifold $\mathcal{C}_1$ of preshapes. Finally, we will explain how a shortest path on the quotient space $\mathcal{S}_1$ can be derived from geodesics on $\mathcal{C}_1$. The central idea of this path-shortening method is to start with an arbitrary $x$-$y$-connecting path and subsequently

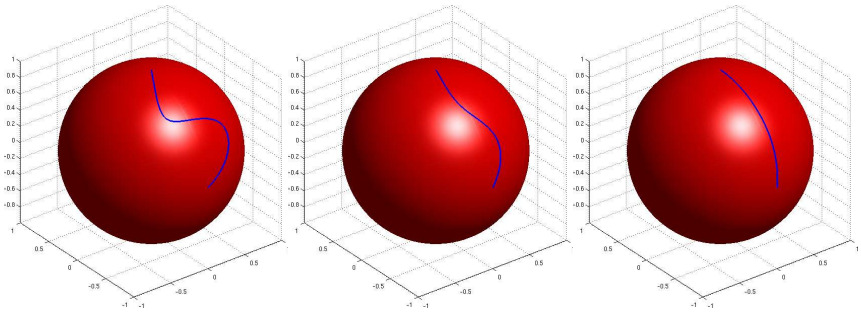


**Figure 3.5: Path-Shortening Method for a Sphere.** Starting with an arbitrary path on a manifold, the proposed method shortens the path until it becomes a geodesic.

shortening this path (cf. Figure 3.5). This will be done by applying a gradient descent method with respect to the functional (3.7).

### 3.3.1 Geodesics in a Finite Dimensional Space

If $M$ is a submanifold of a finite dimensional vector space, every point $x \in M$ can be encoded by finitely many coordinates. Therefore from now on, we assume that $M$ is a submanifold of $\mathbb{R}^N$ such that every point $x \in M$ has a representation $x = (x_0, \ldots, x_{N-1})$. In order to compute a geodesic, we have to encode a path between two points $x, y \in M$ in a discretized manner. Therefore, we assume that a path $m : [0; 1] \to M$ is encoded as $m^\Delta$ via

$$m^\Delta = \left( m\left(0\right) \ldots m\left(\frac{i}{n+1}\right) \ldots m\left(1\right) \right) \in \mathbb{R}^{N \times (n+2)} \tag{3.10}$$

Such a discrete representation encodes the two boundary of the path and $n$ intermediate points. For increasing $n$, we receive a finer discretization of the path $m$. Our path-shortening method will start

with an arbitrary discretized path $m^\Delta$ and during the process the path is altered in a way that the involved energy function $E(\cdot)$ is reduced by applying a gradient descent approach. As soon as the method terminates, we obtain a local minimizer of $E(\cdot)$ that describes a geodesic between $x$ and $y$. So instead of trusting in the numerical stability of an exponential mapping computation, we minimize $E(\cdot)$ directly.

Let us assume that $m^\Delta = (x, m_1, \ldots, m_n, y)$ is such a discretized path. Since the Fréchet derivative of $E(\cdot)$ is $-m''^{\tan}$, we have to alter $m^\Delta$ in the direction of the discretized $m''^{\tan}$. If we combine a forward and a backward difference scheme, we receive update vectors for $m_1, \ldots, m_n$. For $x$ and $y$, we get no update vectors. But this is not necessary, since we want to solve a *boundary condition problem* and the points $x$ to $y$ have to be fixed. After all, we want to compute the shortest path between $x$ and $y$. For the update vector in question, we receive

$$\delta m_i = \left( \frac{m_{i-1} + m_{i+1}}{2} - m_i \right)^{\tan} \tag{3.11}$$

To compute $\delta m$, we have to eliminate the normal component of the expression inside of the brackets in (3.11). The observation that we use to compute this efficiently is (cf. Figure 3.6)

$$\pi_M(m_i + d) = m_i \qquad \Leftrightarrow \qquad d^{\tan} = 0$$

Here, $\pi_M : \mathbb{R}^N \to M$ describes an orthogonal projection onto the submanifold $M$ from the surrounding space $\mathbb{R}^N$ and $d \in \mathbb{R}^N$ is a possible update direction of $m_i$. This observation is only true if $d$ is small enough. Otherwise it is possible that $m_i + d$ will be projected onto another point on $M$. For now, we assume that the update directions will be small enough such that this observation still holds.

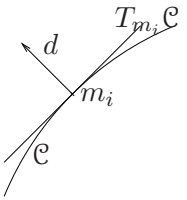



Figure 3.6: A small deformation $d$ from a given preshape $m_i$ is orthogonal to the tangent space $T_{m_i}\mathcal{C}$ at this given preshape, iff the projection of the deformed preshape $m_i + d$ onto the preshape manifold $\mathcal{C}$ is equal to $m_i$.

By stressing this observation even further, we obtain

$$\pi_M(m_i + d) \approx \pi_M(m_i + d^{\tan}) \tag{3.12}$$

This is the key to the path shortening method. Combining equations (3.11) and (3.12), we get the following update step

$$\begin{aligned} m_i + \delta m_i =& \pi_M \left( m_i + \left( \frac{m_{i-1} + m_{i+1}}{2} - m_i \right)^{\tan} \right) \\ \approx& \pi_M \left( \frac{m_{i-1} + m_{i+1}}{2} \right) \end{aligned}$$

This results in the proposed path shortening method:

Note that this algorithm uses for most of the time the linear structure of the surrounding space $\mathbb{R}^N$. The only additional function we have to compute is the projection $\pi_M$ onto the submanifold $M$. Please note that it is not clear whether the initial path between $x$ and $y$ that is computed in Line 2 is parameterized uniformally. But our variational approach will take care of an online gauge fix and as soon as the method terminates, we receive a path that has this important property. In the next section we will show how this method can be used to compute a geodesic on the submanifold $\mathcal{C}_1$ of preshapes.

**Algorithm 1** PATH SHORTENING METHOD

**Input:** Two points $x$, $y$ of the manifold $M \subset \mathbb{R}^N$ and the amount $n \in \mathbb{N}$ of intermediate points.

**Output:** Geodesic $m = (m_0 \cdots m_{n+1}) \in \mathbb{R}^{N \times (n+2)}$ with $m_0 = x$ and $m_{n+1} = y$.

1: **for all** $i = 0, \ldots, n+1$ **do**
2: $\quad m_i = \pi_M \left( x + (y - x) \cdot \frac{i}{n+1} \right)$
3: **end for**
4: **repeat**
5: $\quad$ **for all** $i = 1, \ldots, n$ **do**
6: $\quad\quad M_i = \pi_M \left( \frac{m_{i+1} + m_{i-1}}{2} \right)$
7: $\quad$ **end for**
8: $\quad \delta = \sum_{i=1}^{n} \|M_i - m_i\|^2$
9: $\quad$ **for all** $i = 1, \ldots, n$ **do**
10: $\quad\quad m_i = M_i$
11: $\quad$ **end for**
12: **until** $\delta$ is small enough

### 3.3.2 Geodesics on the Preshape Space

In the previous section, we addressed the problem of finding a geodesic on a submanifold $M$ that is embedded in the finite dimensional linear space $\mathbb{E}$. Now, we want to expand this concept to the more general submanifold $\mathcal{C}_1$ of preshapes.

The first thing, we have to provide is the projection mapping $\pi_{\mathcal{C}_1}$. Here, we use the approximative projection scheme presented in [51]:

**Algorithm 2** Approximative Projection on the Preshape Space

**Input:** $f \in L^2([0; 2\pi])$

**Output:** $\theta \in \mathcal{C}_1$ which is *close* to $f$.

1: $\theta = f$

2: **while** $\theta \notin \mathcal{C}_1$ **do**

3: $$r = \begin{pmatrix} \int_0^{2\pi} \theta(x)\mathrm{d}x - 2\pi^2 \\ \int_0^{2\pi} \cos(\theta(x))\mathrm{d}x \\ \int_0^{2\pi} \sin(\theta(x))\mathrm{d}x \end{pmatrix}$$

4: $$J = \begin{pmatrix} 1 & \frac{1}{2\pi}\int_0^{2\pi} \sin(\theta)\mathrm{d}x & \frac{1}{2\pi}\int_0^{2\pi} \cos(\theta)\mathrm{d}x \\ -\int_0^{2\pi} \sin(\theta)\mathrm{d}x & -\int_0^{2\pi} \sin(\theta)^2\mathrm{d}x & -\frac{1}{2}\int_0^{2\pi} \sin(2\theta)\mathrm{d}x \\ \int_0^{2\pi} \cos(\theta)\mathrm{d}x & \frac{1}{2}\int_0^{2\pi} \sin(2\theta)\mathrm{d}x & \int_0^{2\pi} \cos(\theta)^2\mathrm{d}x \end{pmatrix}$$

5: $$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = J^{-1}r$$

6: $\theta(x) := \theta(x) - u_1 - u_2\cos(\theta(x)) - u_3\sin(\theta(x))$.

7: **end while**

In Line 3, the deviation of $\theta$ from $\mathcal{C}_1$ is stored in $r$. If $r = 0$, $\theta$ must be an element of the preshape space $\mathcal{C}_1$ because this is how $\mathcal{C}_1$ was modeled in Theorem 2. In every iteration step, $\theta$ is pushed towards $\mathcal{C}_1$. This is done by computing the Jacobian $J$ of $\Psi$ in Line 4 and updating $\theta$ in Line 6 accordingly. In practice, the method will

be terminated as soon as $\theta$ is close enough to $\mathcal{C}_1$. So instead of the condition $\theta \notin \mathcal{C}_1$, we will check for the condition $\|r\| > \varepsilon$ for a pre-selected $\varepsilon > 0$. In our experiments, we choose $\varepsilon = 10^{-6}$.

Another problem that we have to solve is that an element of the linear space $L \subset L^2([0; 2\pi])$ cannot be encoded with finitely many parameters. Therefore, we also have to discretize the preshapes. To any preshape $\theta \in \mathcal{C}_1$, we choose an equidistant discretization $\theta^\Delta$:

$$\theta^\Delta = \left(\theta\left(\frac{0}{N}\right) \dots \theta\left(\frac{2\pi \cdot i}{N}\right) \dots \theta\left(\frac{2\pi \cdot (N-1)}{N}\right)\right) \in \mathbb{R}^N \quad (3.13)$$

The submanifold $\mathcal{C}_1$ intersected with this finite dimensional space $\mathbb{R}^N$ will be denoted as $\mathcal{C}_1^\Delta$. With the presented discretization scheme, the problem of finding a shortest path between two preshapes $\theta_1, \theta_2 \in \mathcal{C}_1$ can be reduced to the situation discussed in the last section by looking for a shortest path between $\theta_1^\Delta$ and $\theta_2^\Delta$ of $\mathcal{C}_1^\Delta$. Hence, Algorithm 1 can be used to find a geodesic between $\theta_1^\Delta$ and $\theta_2^\Delta$. In the next section, we will address the problem of finding a shortest path between *shapes* instead of just finding shortest paths between *preshapes*. Therefore, we have to study some properties of the quotient $\mathcal{S}_1 = \mathcal{C}_1/\mathbb{S}^1$.

### 3.3.3 Geodesics on the Shape Space

Knowing how to compute geodesics on $\mathcal{C}_1$, we have the major tool to compute geodesics on the shape space $\mathcal{S}_1$. Every element of this space consists of a whole equivalence class of different preshapes. The shape class of a preshape $\theta$ will be denoted as $[\theta]$ and it contains every preshape that can be created by the group operation of (3.4). Since this group operation was defined for contours, we have to reformulate

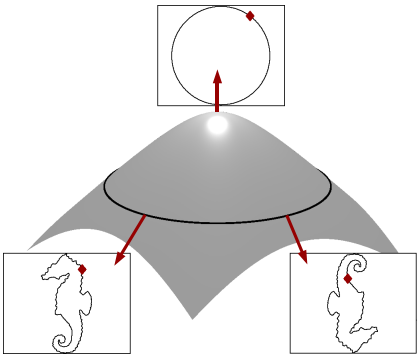


**Figure 3.7: Preshape Orbits.** Since any shape can be parameterized with differing starting points, it corresponds to a family of preshapes which form a closed curve on the manifold of preshapes. Symmetries of a given shape will be reflected by multiple coverings of this curve. In the case of a circle, this curve of preshapes will collapse to a single point.

it for $\theta$-functions. As a result, we receive

$$[\theta] = \{\theta_\alpha | \alpha \in [0; 2\pi]\} \tag{3.14}$$

$$\theta_\alpha : [0; 2\pi] \to \mathbb{R} \tag{3.15}$$

$$x \mapsto \begin{cases} \theta(x+\alpha) - \alpha & , \text{if } x + \alpha \leq 2\pi \\ \theta(x+\alpha-2\pi) - (\alpha - 2\pi) & , \text{if } x + \alpha > 2\pi \end{cases}$$

The reparameterization $\theta_\alpha$ is now defined for any $\alpha \in [0; 2\pi]$. But since $\theta_0 = \theta_{2\pi}$, we can identify 0 with $2\pi$ and thus, we still have a $\mathbb{S}^1$-like group operation. This is because the interval $[0, 2\pi]$ together with the *addition modulo* $2\pi$ describes the same group as $\mathbb{S}^1 \subset \mathbb{C}$ together with the *multiplication of complex numbers*. Therefore, we will call the group operation (3.15) still an $\mathbb{S}^1$-operation. One important property of this operation is the fact that it is a length-preserving group operation on $\mathcal{C}_1$. This means that if we apply a

specific reparameterization encoded by $\alpha \in [0; 2\pi]$ on a whole path $m : [0; 1] \to \mathcal{C}_1$, we receive a path of the same length:

**Lemma 4** ([51])**.** *Given a smooth path* $m : [0; 1] \to \mathcal{C}_1$ *and an* $\alpha \in [0; 2\pi]$*, another path* $m_\alpha : [0; 1] \to \mathcal{C}_1$ *of same length can be defined via* $m_\alpha(t) := m(t)_\alpha$*.*

Because of this lemma, the shortest path between two shapes $[\theta^1] \in \mathcal{S}_1$ and $[\theta^2] \in \mathcal{S}_1$ can be computed by looking for the shortest path between the two *orbits* $[\theta^1]$ and $[\theta^2]$ that form loops in the preshape space $\mathcal{C}_1$. So the problem of finding a shortest path in the quotient space $\mathcal{S}_1$ could be transformed into a shortest path problem on the submanifold $\mathcal{C}_1$. But this problem can be simplified even further. Let us assume that the shortest path $m$ between $[\theta^1]$ and $[\theta^2]$ starts at $\theta^1_\alpha$ and ends at $\theta^2_\beta$. Then according to Lemma 4, $m_{-\alpha}$ is a path of the same length. But because of

$$m_{-\alpha}(0) = m(0)_{-\alpha} = \theta^1_{\alpha-\alpha} = \theta^1 \qquad m_{-\alpha}(1) = m(1)_{-\alpha} = \theta^2_{\beta-\alpha}$$

we also find a shortest path from $[\theta^1]$ to $[\theta^2]$ that starts directly at the preshape $\theta^1$. Hence, the problem of finding a geodesic between the shapes $[\theta^1]$ and $[\theta^2]$ can be solved by finding a geodesic between the preshape $\theta^1 \in \mathcal{C}_1$ and the preshape orbit $[\theta^2] \subset \mathcal{C}_1$.

Since our discretization scheme of Algorithm 1 minimizes the pairwise quadratic distances of subsequent intermediate preshapes, we have to extend this concept to the shape orbits. In order to do so, we have to solve the problem of finding the $\alpha \in [0; 2\pi]$ such that the $L^2$-distance between two discretized preshapes $\theta^\Delta$ and $\eta^\Delta_\alpha$ is minimized. The solution of this subproblem will then be used to extend Line 6 of Algorithm 1 accordingly. Because of the discretization that we are using, we are looking for the optimal cyclic permutation of the discretized preshape $m^\Delta_i$ with respect to $m^\Delta_{i-1}$. This can be calculated via Discrete Fourier Transform which can be compute very efficiently [14]. The function to calculate $\alpha \in \left[\frac{0}{N}2\pi, \ldots, \frac{N-1}{N}2\pi\right]$

**Algorithm 3** GEODESICS ON THE SHAPE SPACE

**Input:** Two preshapes $\theta, \eta \in \mathcal{C}_1^\Delta \subset \mathbb{R}^N$ and the amount $n \in \mathbb{N}$ of intermediate preshapes.

**Output:** Geodesic $m = (m_0 \cdots m_{n+1}) \in \mathbb{R}^{N\times(n+2)}$ with $m_0 = \theta$ and $m_{n+1} \in [\eta]$.

$m_0 = \theta$
**for all** $i = 1, \ldots, n+1$ **do**
  $m_i = \pi_{\mathcal{C}_1^\Delta}\left(\theta + (\eta - \theta) \cdot \frac{i}{n+1}\right)$
  $\alpha = \mathrm{DFT}(m_{i-1}, m_i)$
  $m_i = (m_i)_\alpha$
**end for**
**repeat**
  **for all** $i = 1, \ldots, n$ **do**
    $M_i = \pi_{\mathcal{C}_1^\Delta}\left(\frac{m_{i+1}+m_{i-1}}{2}\right)$
    $\alpha = \mathrm{DFT}(m_{i-1}, M_i)$
    $M_i = (M_i)_\alpha$
  **end for**
  $\delta = \sum_{i=1}^{n} \|M_i - m_i\|^2$
  **for all** $i = 1, \ldots, n+1$ **do**
    $m_i = M_i$
  **end for**
**until** $\delta$ is small enough

given the preshapes $\theta^\Delta$ and $\eta^\Delta$ will be denoted as $\mathrm{DFT}(\theta^\Delta, \eta^\Delta)$. As a result, we receive the Algorithm 3 to find geodesics on the shape space $\mathcal{S}_1$.

Note that this approach is not a purely continuous approach. It is of course true that the gradient descent step is a classical continuous technique. Nonetheless, the computation of the *Discrete Fourier Transform* depends highly on the discretization size. Hence, the presented method is a semi-continuous approach.

## 3.4 Comparing Different Approaches

In the last section, we proposed a variational method to compute a geodesic on the shape space $\mathcal{S}_1$. In contrast to the shooting method, we do not need to solve multiple ordinary differential equations in order to receive such a geodesic. Therefore, it seems reasonable to assume that the proposed method is much faster than the shooting method. To substantiate this, we present several tests on shapes that are publicly available. First of all, we use the SQUID database [68] that has been provided by the University of Surrey. Additionally, we use the more recent MPEG7 shape database[3]. The samples that we use from these databases are chosen to illustrate the functionality of the proposed method.

We start our evaluation by showing that the distance function we used describes the similarity of different shapes better than the region based $L^2$-distance that we used in Chapter 2. Afterwards, we will compare the shooting method with the proposed variational path-shortening method. We show that in contrast to the shooting method, our method provides symmetric results and we will conclude this section with a run-time comparison that shows an improvement of a factor of 1000 depending on the shapes' resolution.

### 3.4.1 Contour Based vs. Region Based Approach

First, we will show how the presented method of *shape morphing* exhibits a shape metric that is more accurate than the region based $L^2$-distance used in Chapter 2. To this end, we consider the three different hands in Figure 3.8. The difference between the first and the second hand is from a perception point of view negligible. In

[3]The shape database *MPEG7 CE Shape-1 Part B* is online available and can be downloaded at `http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz`

**Figure 3.8: Three hands.** In the following, we will compute the pairwise distances of these three hands. These are samples of the MPEG7 shape database.

fact, the hands only differ in a slight bending of the thumb. But these two hands differ somewhat more from the third hand. This is because for the third hand, the distance between the thumb and the index finger and the distance between the ring finger and the pinky is much larger than the respective distances for the first two images. Hence, we expect the shape distance between the third hand and one of the first two hands be twice as big as the shape distance between the first two hands.

In Figure 3.9, we see the result of the region based $L^2$-distance. This distances is measured as the area of the symmetric difference of two *shape images*. To compare this distance to the morphing distance, we minimized the region based distance with respect to *rotation* and *translation* (cf. (2.29)). As you can see, this distance is very sensitive to local deformations like the bending of the thumb or the stretching of thumb and pinky.

Computing the geodesic on the shape space $\mathcal{S}_1$ via the *path shortening method* results in the three morphings of Figure 3.10. For the first two shapes (first row in Figure 3.10), this morphing looks very

**Figure 3.9: Region based distance.** The region based distance is looking for a rotation and translation such that the area of the symmetric difference is minimized. This symmetric difference consists of the following two components. The first one (green) represents the region of the first shape that is not a part of the second shape. The second component (red) is the region of the second shape that is not part of the first shape

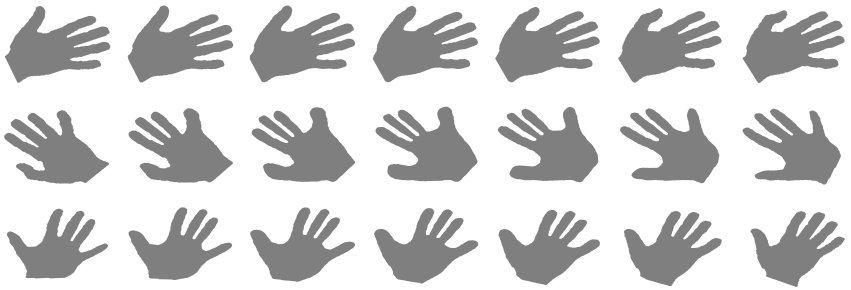


**Figure 3.10: Morphing paths between three hands.** Between the three hand shapes, there exist three different morphing paths. The morphing between the first and the second shape (first row) results in a slight bending of the thumb. For the second and the third morphing paths, a thickening of the thumb and the pinky is necessary in order to minimize the geodesic length of the chosen preshape manifold.

natural. But if we consider the morphing between the third shape and either of the first two shapes, this morphing looks less natural. The thumb and the pinky for example thickens during the morphing. Hence, the morphing can cope with some restrictions of the region based $L^2$-distance, but it is still not perfect. On the other hand, we are only interested in the distance and whether this distance reflects the object similarity that humans would perceive. The three pairwise distances define a triangle in the shape space that can be isometrically embedded into $\mathbb{R}^2$. Doing this, results in two different triangles, one for the region based distance and one for the morphing distance.

As you can see in Figure 3.11, the morphing distance results in a triangle in which the first two shapes are closer to one another than the third shape. Hence, the morphing based distance results in a more descriptive triangle than the region based distance. As a result, we assume that the metric induced by the minimal length of geodesic based morphing is more descriptive than the mere region based $L^2$-distance.

### 3.4.2 Path-Shortening Method vs. Shooting Method

#### Symmetric Behavior

Now, we like to address the difference between the computation of the geodesic with respect to the *shooting method* and with respect to the proposed *path shortening method*. In Figure 3.12 the results of both approaches are shown for the exemplarily morphing from a seahorse into a starfish. We choose this specific example because the distance between these two very dissimilar shapes is large. Hence, the geodesic reflects the structure of the preshape manifold $\mathcal{C}_1$ more than the small distance for the hand shapes used in Section 3.4.1.

The first two rows of Figure 3.12 show the result of the *shooting*

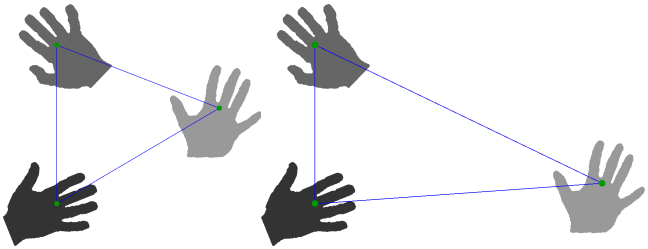


**Figure 3.11: Induced Metrics of Different Shape Spaces.** The pairwise distances of the three hand shapes is represented as a triangle in the real plane. The left triangle represents the metric structure of the region based stochastic shape space. The right triangle is shown with respect to the the shape space $\mathcal{S}_1$. The metric structure of $\mathcal{S}_1$ detects a higher dissimilarity between the first two shapes and the third shape.

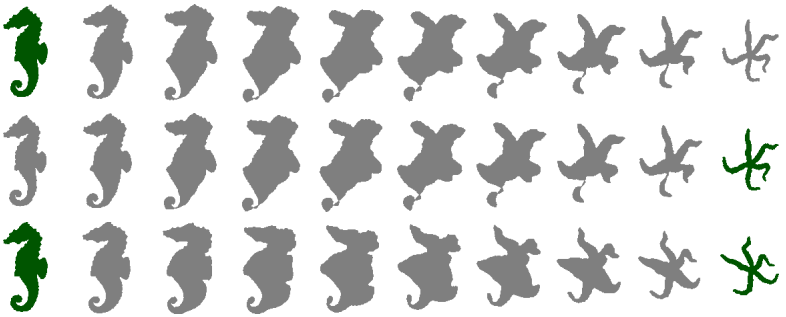


**Figure 3.12: The Computed Morphing Paths.** The morphing of a seahorse towards a star fish is calculated via two different methods. The green shape symbolizes the initial value of the used methods - these shapes were not altered. The first two rows show the results of the *shooting method* starting at the *seahorse* (1st row) and the *starfish* (2nd row). The *variational path-shortening method* (3rd row) fixes both the *seahorse* and the *starfish*.

*method* whereas the third row shows the geodesic computed with respect to the variational *path shortening method*. The first row shows shapes on the morphing path that starts at the seahorse shape and for the second row the computation started at the starfish shape. All these morphings are valid geodesics but the calculated alignments, i.e., the chosen star fish preshapes are different. This leads to a self-intersection in the first cases, whereas in the third case, the tail of the seahorse *unrolls* in an expected natural manner. This is due to the different alignments of the target shape. It is easy to see that the variational method moves the tip of the tail towards the tip of one of the five star fish extremities. Moreover, the first two geodesics provide a path length of 5.5447 and 5.5431, whereas the third geodesic has the length 4.6371. Therefore, the shooting method does not only get stuck in a local minimum but also it does not provide for symmetric results. In this example, we used a discretization of $N = 1000$ for the preshapes. Therefore, there exist 1000 different alignments for the target preshape. Calculating the geodesic distance between the preshapes with respect to all of these 1000 alignments, we could confirm that the distance computed by the *variational path-shortening method* represents the global minimum with respect to preshape alignment. Thus, the calculation of realignments in Line 10 of Algorithm 3 serves the purpose of finding the minimal distance between two given shapes. Overall, the proposed variational path-shortening methods should be favored over the commonly used shooting method in order to obtain meaningful symmetric distances.

**Run-time Behavior**

In the last test, we showed the superiority of the *path-shortening method* over the shooting method with respect to its exactness. Now, we will show that the path-shortening method is also much faster than the shooting method.

Figure 3.13 shows the computation times of the morphings presented in Figure 3.12. It varies from the computation time in [51] because we use highly resoluted preshapes and the methods stop only if they can provide a very accurate result. On the horizontal axis the discretization resolution of the geodesic is noted. This is the amount $n$ of intermediate shapes that both methods are using. First of all, we see that the computation time is not symmetric for the shooting method. Moreover, the computation time varies by 20 to 30 percent. This is due to the fact that the shooting method depends highly on the curvature at the starting shape. Hence, not only the result of the shooting method but also the computation time is asymmetric.

The variational method on the other hand is symmetric and thus, the runtime does not depend on the starting shape. In addition, the calculation time is less than 100 milliseconds in the highly resoluted case. If we use the same resolution as in [51], the variational method takes only a few milliseconds. To conclude our observations, the variational path-shortening method does not only reflect the morphing induced shape distance in the sense that it provides for symmetric results. It is also faster and thus computing distances as a length of geodesic becomes attractive for the Computer Vision field of Shape Analysis.

## 3.5 Limitations of Morphings

We saw that shape dissimilarity measured as the geodesic length between two shapes on the orbifold $\mathcal{S}_1$ reflects the perceptional dissimilarity of humans better than a mere region based $L^2$-distance as in Chapter 2. Nonetheless, it is has its limitations. First of all, we only computed *a geodesic* as defined in Definition 12. Such a geodesic fulfills the Euler-Lagrange equation of the energy functional (3.7). Thus, it may just be a *local minimum*. Whether it is *a global minimum* is not clear at all. As a consequence, we cannot be sure if the

proposed method really computes the distance as defined in (3.8).

Another problem that already appeared is due to the chosen representation of preshapes. In Figure 3.10, we saw already that the morphing may not be natural in the way that morphing one hand into another results in thickening and afterwards shrinking of certain fingers, e.g., thumb and pinky. This is in fact the result of the chosen shape representation. The problem of this representation is depicted in Figure 3.14. In this figure, we show how different points on the first shape are moved in the plane during their morphing. As a result, we receive a set of points of either shapes that are set in correspondence with respect to the morphing. This means that for every $t \in [0; 2\pi]$ the following two points are set in correspondence:

$$x_t = c_0(t) = e^{i\theta_0(t)} \qquad \text{and} \qquad y_t = c_1(t) = e^{i\theta_1(t)}$$

where $c_i$ is a representing curve of a shape and $\theta_i : [0; 2\pi] \to \mathbb{R}$ is the respective element of the preshape space $\mathcal{C}_1$. Since the curves are represented by arc-length, for arbitrary $s, t \in [0; 2\pi]$ the following holds:

$$\operatorname{dist}(c_0(s), c_0(t)) = |s - t| = \operatorname{dist}(c_1(s), c_1(t))$$

where $\operatorname{dist}(\cdot, \cdot)$ denotes the distance *on the specific contour*. Because of this distance preserving property, we obtain a sort of contour-based rigidity. If we choose one specific correspondence between points of different shapes, all correspondences of one contour with respect to the other is already defined. For the example of the two hands on the left hand side in Figure 3.14, this works quite well. But for the toy example on the right hand side of this figure, the restriction of this rigidity comes to light.

While for the region based $L^2$-distance of Chapter 2 a rigidity of the whole shape emerged (cf. Figure 3.9), we are here confronted with only a parameterization based rigidity. In the next chapter, we will address this problem. Instead of computing a whole morphing, we will only be interested in finding correspondences between points of different shapes. This approach is also known as *shape matching* since we are looking for points that are similar to one another, i.e., points that match.

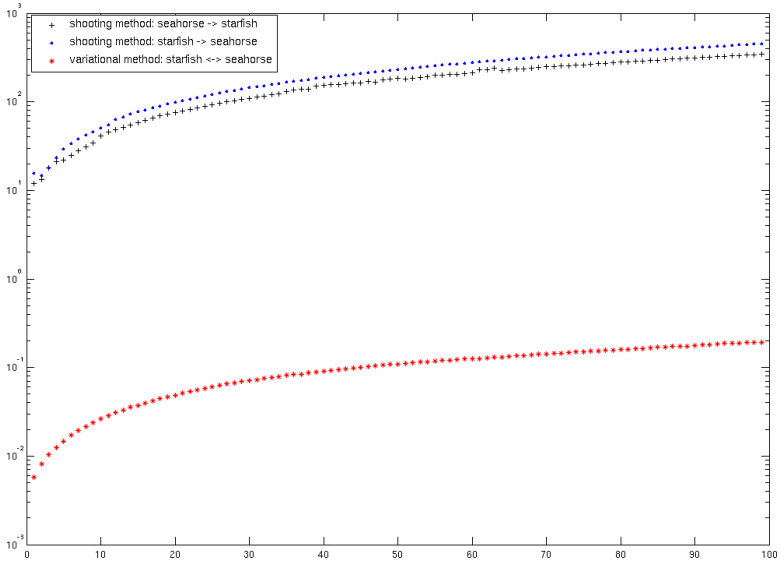



**Figure 3.13: Run-time for Different Morphing Methods.** The computation time in seconds to calculate geodesics is plotted against the discretization size of a morphing. The variational method has two advantages with respect to its computation time. **Symmetry:** The geodesic calculation does not depend on the starting shape, whereas the run-time for the shooting method varies by ca. 25%. **Run-time:** The variational method is faster by a factor of 1000.

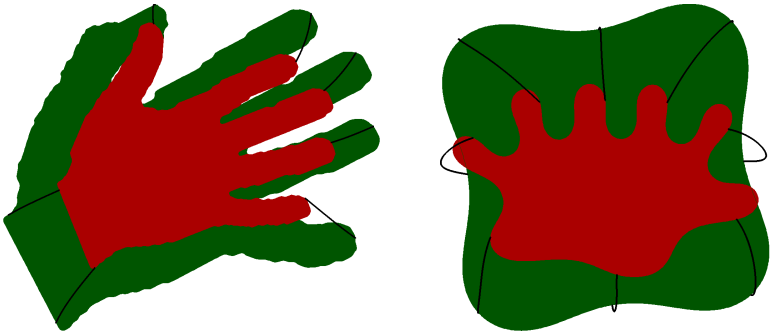


**Figure 3.14: Point Correspondences for Shape Morphing.** A morphing transforms one shape into another. Thus, every point $x$ of the first shape is moved via this morphing onto a point $y$ of the second shape. These paths are shown for some chosen points. For the two hands, these *correspondences* $x \leftrightarrow y$ reflect a certain *local similarity*. For the shapes on the right hand side, this similarity does not exist any more.

# Chapter 4

# Shape Matching

In the previous chapters, we introduced two different methods to measure the dissimilarity of shapes. This resulted in defining a metric $d(\cdot, \cdot)$ that assigns a non-negative distance to an arbitrary pair of shapes. In this chapter, we will make use of another distance function. It differs from the previous distance function in the sense that it is not a metric any more. Instead it is a relaxed version of a metric in the sense that the triangle inequality of Definition 2 does not hold any longer. Such a concept is known as a *pseudo-metric* and has already been studied in the form of region-based $L^2$-distances. Especially for level set approaches, pseudo-metrics have been successfully applied [28].

In order to define this new distance function, we consider the problem of shape matching which is very popular in Computer Vision in order to classify a selection of different shapes. Like in the preceding chapters, we still represent a shape as a mapping that is defined over the circle $\mathbb{S}^1$. The main advantage over the concepts described above is that the invariance with respect to translation, rotation or scaling will be directly incorporated into the description of shapes. Hence, we do not have to consider orbits with respect to some group

operations. In practice, this means that we use certain features that describe the *local appearance* of every contour point. Instead of representing a shape as a sequence of planar points, it is now modeled as a sequence of feature values. A very prominent feature is the *curvature*. This feature has certain advantages over other features. First of all, it is easy to compute and secondly it only depends on a small neighborhood of a given curve point. Thus, areas of the shape that are further away from a specific point do not disturb the computation of the curvature at this point.

This chapter is organized as follows. In Section 4.1, we address the problem of shape matching with respect to an arbitrary preselected feature. There we recapitulate the development of shape matching and explain our approach which is an important extension of the work of Sebastian *et al.* [85] in the sense that our approach is invariant with respect to re-parameterization. A closely related approach has been first introduced by Tagare [90], but the model of Tagare is difficult to handle since instead of a matching function, Tagare used a matching relation that in general can neither be represented as a matching function from the first shape onto the second shape nor vice versa.

In Section 4.2, we present a collection of different features. Especially, we present a method of computing the curvature via an integration process. As a result, the presented method to compute a curvature is numerically much more stable than classical approaches which involve the computation of the second derivative. In Section 4.4, we present the classical method of *Dynamical Time Warping* (DTW) and the drawbacks of this specific method which results for example in cubic runtime with respect to the input size. In order to reduce the runtime, we present two different methods in the succeeding sections.

In Section 4.4.1 we present a very rapid DTW approach which to our knowledge is the fastest DTW based method and in Section 4.5

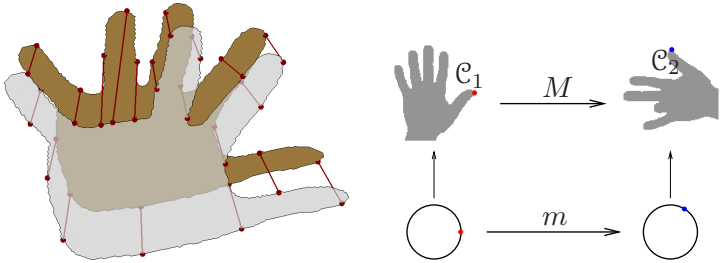



**Figure 4.1: Shape Matching.** *Left hand side:* Matching two shapes amounts to computing a correspondence between pairs of points on both shapes. *Right hand side:* Instead of looking for a mapping $M : \mathcal{C}_1 \to \mathcal{C}_2$, a matching $m : \mathbb{S}^1 \to \mathbb{S}^1$ is defined on the parameterization domains.

the problem is cast a *graph cut problem*. In Section 4.6, we provide two different tests on publicly available databases.

## 4.1 Pseudo-Metrical Shape Spaces

The problem of finding a match between two different sequences of data has a rather long history in Computer Science. A very classic approach is the finding of a *substring* in a given *text string*. But besides these exact matchings, the looking for *approximate matchings* has also become an import task. One of the first works addressing this problem was [97]. Here a metric on the set of *string characters* was assigned to model errors that may occur during the input of the search string. To find this elastic match, dynamic programming techniques were applied. This technique has a long tradition in the fields of string alignment, speech recognition, stereopsis and handwriting recognition [4, 55]. In [22], this concept was first applied to the matching of shapes. To this end, we assume that a shape is

represented as a closed loop in a preselected feature space $\mathcal{F}$ which is equipped with a metric $d(\cdot,\cdot)$. $\mathcal{F}$ can be any metric space but in practice, $\mathcal{F}$ is some Euclidean space like $\mathbb{R}$ or $\mathbb{R}^f$. In order to compute the similarity of different shapes efficiently, we assume that the metric $d$ of $\mathcal{F}$ can be computed very rapidly. Further on, we assume that transforming an arbitrary curve $c : \mathbb{S}^1 \to \mathbb{C}$ into the *feature loop* $f : \mathbb{S}^1 \to \mathcal{F}$ can be done easily. Examples of such *features* will be presented in Section 4.2. For now, we want to define the general concept of *feature based shapes* that differ from the shape concept of the previous sections in the sense that we do not need any group operation in order to abstract from the contour:

**Definition 13.** A metrical space $(\mathcal{F}, d)$ equipped with a mapping $F : \mathrm{Imm}(\mathbb{S}^1, \mathbb{C}) \to \mathrm{Map}(\mathbb{S}^1, \mathcal{F})$ is called a *feature space* and the mapping $F$ is called its *feature transformation*. Further on, we call $F$ invariant with respect to the group SE(2) of rigid body transformations if for any curve $c \in \mathrm{Imm}(\mathbb{S}^1, \mathbb{C})$, rotation $R \in \mathbb{C}$ and translation $T \in \mathbb{C}$, the following holds

$$F[c(\cdot)] \equiv F[R \cdot c(\cdot) + T] \tag{4.1}$$

From now on, we assume that we deal with a preselected feature space $(\mathcal{F}, d, F)$ such that $F$ is invariant with respect to the group SE(2) of rigid body transformations. Besides rigid body motions, other *shape transformations* are possible. In this chapter, we want to detect local stretching or contraction. Hence, we are looking for a direct correspondence mapping which maps the points of one shape to the correspondent points of the other shape. Since the points of a shape form an arbitrary subset of the plane $\mathbb{C}$, it is easier to find the correspondence directly on the parameterization domain $\mathbb{S}^1$ (cf. Figure 4.1).

To avoid self-occlusions during the matching process, a *matching* can be modeled via an orientation-preserving diffeomorphism $m : \mathbb{S}^1 \to \mathbb{S}^1$ that maps points of the first parameterization domain to the

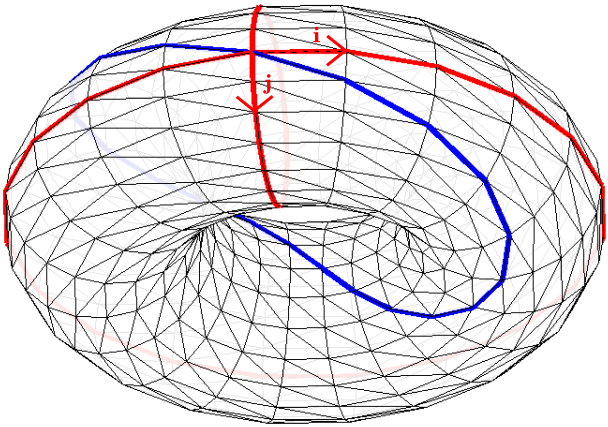



**Figure 4.2: Matching Loop.** Matching points on either of two shapes is equivalent to a cyclic path on a torus. If two curves $c_0$ and $c_1$ are both parameterized over a circle. The torus represents all possible correspondences. The graph $\Gamma(m)$ of a matching $m : \mathbb{S}^1 \to \mathbb{S}^1$ describes a loop (blue) that covers both parameterization spaces (horizontal and vertical red loops) exactly once. In this figure the *matching loop* of the matching $m(s) := s$ is sketched.

corresponding points of the second parameterization domain. On the space of these matchings, we will define a functional $E : \mathrm{Diff}^+(\mathbb{S}^1) \to \mathbb{R}_0^+$ that measures the quality of a matching. The goal of a matching algorithm is to find the minimum of $E$ which will mainly measure the $L^2$-distance of the feature differences. This results in the following functional:

$$\widetilde{E}^{\mathcal{F}}_{f_0,f_1}(m) = \int_{\mathbb{S}^1} d(f_0(s), f_1 \circ m(s))^2 \mathrm{d}s \tag{4.2}$$

This functional was used by Cohen *et al.* [22]. But it has the dis-

advantage that it is not symmetric in the sense that the equation $\widetilde{E}^{\mathcal{F}}_{f_1,f_2}(m) = \widetilde{E}^{\mathcal{F}}_{f_2,f_1}(m^{-1})$ does not hold in general. To overcome this, Sebastian *et al.* [85] and Tagare [90] propose different approaches. Here we follow the concept of Tagare who reformulated the functional (4.2) as a curve integral on a torus. This is possible because the graph $\Gamma(m)$ of a matching mapping $m : \mathbb{S}^1 \to \mathbb{S}^1$ can be represented as a closed loop on the product space $\mathbb{S}^1 \times \mathbb{S}^1$ which describes a torus (cf. Figure 4.2).

To reformulate (4.2) as a line integral has the important advantage that the resulting energy functional is not only symmetric, but also independent of the parameterization of $\Gamma(m)$. Hence, we use the following energy functional:

$$E^{\mathcal{F}}_{f_0,f_1}(m) = \int_{\mathbb{S}^1} d(f_0(s), f_1 \circ m(s))^2 \sqrt{1 + m'(s)^2}\mathrm{d}s \tag{4.3}$$

In this functional, the *data term* $d(f_0, f_1)^2$ is therefore integrated along the *matching loop* $s \mapsto (s, m(s))$. Since we use a line integral instead of a pure $L^2$-distance like in (4.2), the *smoothness term* $\sqrt{1 + m'^2}$ is directly coupled to the data term. As a result, we do not need any additional parameter in (4.3). Using this energy functional, a distance function for shapes is induced as follows:

$$D_{\mathcal{F}}(c_0, c_1) = \min_{m \in \mathrm{Diff}(\mathbb{S}^1)} \sqrt{E^{\mathcal{F}}_{F[c_0],F[c_1]}(m)} \tag{4.4}$$

The fact that (4.3) is parameterization invariant with respect to $\Gamma(m)$ is very important from an implementation point of view, because we do not have to take care that $m$ or $\Gamma(m)$ is been parameterized in a certain sense. Different parameterizations will lead to the same energy functional. The freedom in parameterization reduces the amount of restrictions for designing an efficient algorithm. But

$X$ $Y$ $Z$

**Figure 4.3: Semi-metric.** Here, we show an example of three shapes such that their pairwise distance does not fulfill the triangle inequality. If we denote the shape of the dog by $X$ and the shapes of the two faces as $Y$ resp. $Z$, we obtain the following distances: $D_{\mathcal{F}}(X, Y) = 453$, $D_{\mathcal{F}}(Y, Z) = 79$ and $D_{\mathcal{F}}(X, Z) = 547$. Obviously, the triangle inequality $D_{\mathcal{F}}(X, Z) \leq D_{\mathcal{F}}(X, Y) + D_{\mathcal{F}}(Y, Z)$ does not hold. As feature space $\mathcal{F}$, we used the *Inner Shape Context* presented in Section 4.2.2.

this freedom has to be paid off by loosing a property that all the shape distances presented above had. This is the triangle inequality which does not hold any more. An example is provided in Figure 4.3.

After presenting the general concept of shape matching, we want to present two different shape features in the following section. In Section 4.3, we will then address the problem of solving this problem. In that section, we will propose purely discrete approaches. These methods involve the computation of either a shortest path or the minimal cut in a graph.

## 4.2 Shape Features

In the last section, we introduced the functional (4.3), which measures the quality of a matching $m$. This functional highly depends on

the preselected feature space $(\mathcal{F}, d, F)$. There exist numerous shape features which capture the local shape by means of differential or integral invariants [63], the most commonly considered descriptor being curvature [69]. In this work, we are not focused on the introduction of new invariants, but rather on the question of how to efficiently compute a matching given any local shape descriptor.

For the sake of completeness, this section presents certain commonly used features like the curvature and the *Inner Shape Context* (ISC). Curvature is a good feature for shapes but it is not very popular because normally curvature is computed via a second derivative. This computation is very sensitive to small noise. In Section 4.2.1 we will present a novel method to compute the curvature at a certain point that results in an integration process instead of two differentiation processes. Hence, it is much more robust than the classical curvature computation. For the computation of the ISC, we follow the ideas of [60] which we will briefly present in Section 4.2.2.

### 4.2.1 Integral Curvature Computation

The shape matching as introduced above relies on local features that are invariant with respect to translation and rotation. In practice, these features need to be computed in a robust manner. To this end, Manay *et al.* [63] introduced different features that were all obtained by integration processes and since these features were invariant under rigid body motions, they were called *integral invariants*. One of these integral invariants approximated the curvature by calculating the intersection of the shape's interior and a circle of fixed radius $r$ (cf. Figure 4.4). In contrast to [63], we perform a Taylor approximation of the invariant which is *exact* up to the first order. As a consequence, we obtain a method to compute the curvature robustly by performing an integration instead of a differentiation scheme.

If we now consider the closed curve $c : \mathbb{S}^1 \to \mathbb{C}$ with its curvature

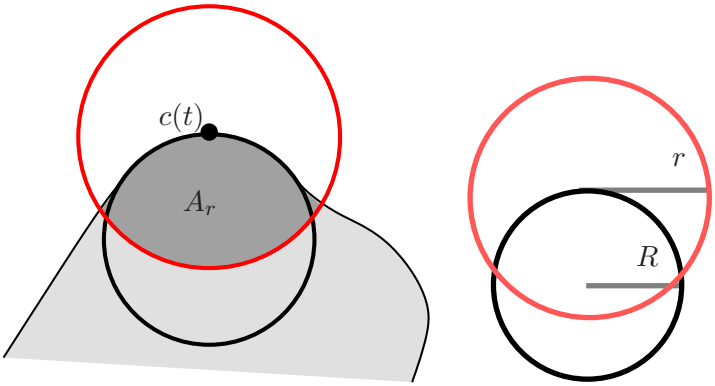



**Figure 4.4: Curvature calculation.** The curvature at any point along the curve can be estimated from the intersection $A_r$ of a ball with radius $r$ centered at the curve point with the interior of the shape. $R = \frac{1}{\kappa}$ is the radius of the osculating circle (cf. (4.5)).

function $\kappa : \mathbb{S}^1 \to \mathbb{R}$. Near the point $c(t)$, the curve $c$ can be described via its osculating circle of radius $R(t) := \frac{1}{\kappa(t)}$. In fact, the Greek used this concept to measure the curvature of a curve. The osculating circle is an second order approximation of $c$. Let us now consider the set

$$A_r(t) = \{x \in \text{int}(c) | \|x - c(t)\| \leq r\}$$

that consists of all points inside the curve $c$ that are also closer to the curve point $c(t)$ than a preselected radius $r$. Then, the area of this set $A_r(t)$ can be approximated via

$$\begin{aligned}
\operatorname{area}(A_r) \approx& \int_{-a}^{a} \sqrt{R^2-\tau^2} - \left[R - \sqrt{r^2-\tau^2}\right] \mathrm{d}\tau \\
=& \frac{\pi}{2}(R^2+r^2) - R^2 \cos^{-1}\left(\frac{a}{R}\right) - r^2 \cos^{-1}\left(\frac{a}{r}\right) \\
& - 2Ra + a \underbrace{\left(\sqrt{R^2-a^2} + \sqrt{r^2-a^2}\right)}_{=R} \\
=& R^2\left(\frac{\pi}{2} - \cos^{-1}\left(\frac{a}{R}\right)\right) + r^2\left(\frac{\pi}{2} - \cos^{-1}\left(\frac{a}{r}\right)\right) - Ra \\
=& R^2 \sin^{-1}\left(\frac{a}{R}\right) + r^2 \sin^{-1}\left(\frac{a}{r}\right) - Ra
\end{aligned}$$

whereas $a = \sqrt{r^2 - \left(\frac{r^2}{2R}\right)^2}$. Introducing $\varphi := \sin^{-1}\left(\frac{r}{2R}\right)$, we receive

$$\frac{\operatorname{area}(A_r)}{r^2} \approx \left(\frac{R}{r}\right)^2 \sin^{-1}\left(\frac{r}{R}\cos(\varphi)\right) + \frac{\pi}{2} - \varphi - \frac{R}{r}\cos(\varphi).$$

Because of $\frac{r}{R} = 2\sin(\varphi)$, we finally get the simplification

$$\frac{\operatorname{area}(A_r)}{r^2} \approx \frac{1}{2}\left(\frac{\varphi}{\sin(\varphi)^2} - \frac{\cos(\varphi)}{\sin(\varphi)}\right) + \frac{\pi}{2} - \varphi$$

The linear Taylor approximation of the right hand side leads to the expression $\frac{\pi}{2} - \frac{2}{3}\varphi$. Therefore, the curvature $\kappa$ can be approximated via

$$\kappa \approx \frac{2}{r}\sin\left(\frac{3\pi}{4} - \frac{3\operatorname{area}(A_r)}{2r^2}\right) \tag{4.5}$$

Note that the quadratic approximation error can be reduced by decreasing the radius $r$. Moreover, $\kappa = \lim_{r\to 0} \frac{2}{r}\sin\left(\frac{3\pi}{4} - \frac{3\operatorname{area}(A_r)}{2r^2}\right)$. In our implementation, we used the right hand side of (4.5) to calculate the curvature function of a given curve. In Figure 4.5, the

**Figure 4.5: Gaussian noise.**The matching between an original hand and a hand added with Gaussian noise is visualized. From left to right the standard deviation is $\sigma = 0, 0.5, 1, 3, 4$. At $\sigma = 4$ a matching starts to collapse (cf. point 4).

robustness of these feature with respect to high Gaussian noise is shown. For this test, we started with one contour and added in the direction of the curve's normal Gaussian noise of a preselected standard deviation $\sigma$. We observed that even at the presence of rather high Gaussian noise, the matching is quite accurate. The matching starts to collapse for $\sigma = 4$. But at this point it is difficult to recognize the represented shape even for a human.

### 4.2.2 Inner Shape Context

The robustness of the *integral feature* lies in the fact that we consider a larger neighborhood of a given contour point. A similar concept was used by Belongie *et al.* when they introduced their *Shape Context Feature* in [5]. It involves the pairwise distances between all contour points: For every point of a given contour, a histogram is computed that reflects the shortest paths to all other points of that contour. Hence to every pair of points, the distance and the angle with respect to the contour's tangent is computed and stored in a preselected histogram. This idea is sketched in the left hand side of Figure 4.6.

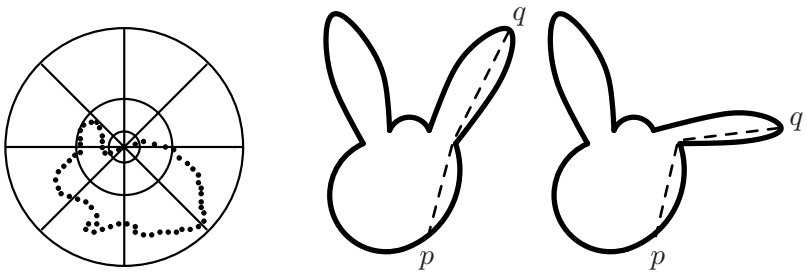



**Figure 4.6: Shape Context.** To compute the *Shape Context*, distances from one point to all the other points of the contour are stored in a histogram (left hand side). The *Interior Shape Context* considers only the shortest paths inside the shape. Hence, it is robust to articulations like the position of the bunny's ear.

This feature is very robust but it has also certain disadvantages. One of them is the fact that articulated shapes exhibit very different features. As a consequence, those shapes are difficult to match. But since articulated shapes are very natural, Ling and Jacobs proposed in [60] an extension of the original shape context. Instead of the Euclidean distance that was considered in the original shape context, they only considered paths that are inside of the given shape (cf. right hand side of Figure 4.6). By considering only these *interior paths*, they came up with another shape context – the *interior shape context*. Since this feature turned out to handle the matching of different shapes very well, we will use this feature in the following. In fact, the methods that we will present in Section 4.3 are independent of the preselected feature and the focus of the next section is to present efficient methods for shape matching for every possible feature space $(\mathcal{F}, d, F)$.

## 4.3 Discrete Approaches

In the last sections, we addressed the problem of matching two shapes with respect to a preselected feature space $(\mathcal{F}, d, F)$. The focus of this section is to minimize the energy functional $E$ of (4.3). From now on, we are only interested in purely discrete approaches. Traditionally the problem of shape matching has been cast into a shortest path problem through a two-dimensional planar graph, the edge weights of which incorporate the distance of the local features [65, 39, 3, 2, 38, 57, 93, 85, 73].

This rather general approach is known as *Dynamic Time Warping* (DTW) and will be presented in Section 4.4. Another approach that we first presented in [82] results in finding the minimal cut in a planar graph. This approach will be presented in Section 4.5. Both approaches, the DTW approach and the graph cut approach suffers from a rather high runtime. Thus, it is very costly to compare highly resoluted shapes. To overcome this drawback, we use the fact that the underlying graphs of both approaches are planar. This leads to different methods of reducing the involved runtime significantly by exploiting this important property of planarity. These methods are presented in Sections 4.4.1 and 4.5.1. If we want to compare shapes that are discretized by $N$ feature points, the resulting methods compute the global minimum of $E$ in almost quadratic runtime, i.e. in $O(N^2 \log(N))$. In Sections 4.4.2 and 4.5.2 we provide a runtime comparison of all the presented methods.

## 4.4 Dynamic Time Warping

Let us assume that two curves $c_0, c_1 : \mathbb{S}^1 \to \mathbb{C}$ and their feature loops $f_0, f_1 : \mathbb{S}^1 \to \mathcal{F}$ with respect to a feature space $(\mathcal{F}, d, F)$ are given. The goal now is to find a matching $m : \mathbb{S}^1 \to \mathbb{S}^1$ that minimizes its path length (4.3) on a torus. To simplify this problem,

let us also assume that we already know an initial matching, namely $m(x_0) = x_1$. This means, we know that $f_0(x_0)$ should be matched to $f_1(x_1)$. If the matching torus of Figure 4.2 is now cut open along the two red coordinate loops, we obtain a flat area $[0; 2\pi] \times [0; 2\pi]$ and the closed matching loop becomes a shortest path from $(0, 0)$ to $(2\pi, 2\pi)$. Since the problem of finding a shortest path through a graph has been intensively studied, it makes sense to reformulate the problem into a graph theoretical problem. Therefore, we assume that each of the two feature loops is given as a discretized sequence of $N$ feature points. This means that the parameterizing circle $\mathbb{S}^1$ is discretized equidistantly in $x_0, \ldots, x_{N-1}$. For Simplicity, we expand this notation such that $x_i$ is defined for any $i \in \mathbb{Z}$ via $x_i := x_{(i \mod N)}$.

In order to find the matching $m$, we have to find a shortest path within a graph of $O(N^2)$ vertices (cf. Figure 4.7) that we will now construct. The vertices $v_{i;j} \in V$ represent a possible match between $f_0(x_i)$ and $f_1(x_j)$ and the integrand of (4.3) at this point is $(f_0(x_i) - f_1(x_j))^2$. Therefore, the weight $w$ of any edge $(v_{i;j}, v_{k;l})$ carries the value of the path integral along this edge:

$$w(v_{i;j}, v_{k;l}) = \frac{(f_0(x_i) - f_1(x_j))^2 + (f_0(x_k) - f_1(x_l))^2}{2} \left\| \begin{pmatrix} k-i \\ l-j \end{pmatrix} \right\|$$

In order to allow only paths from $(0, 0)$ to $(2\pi, 2\pi)$ that represent a diffeomorphism, only those edges are allowed that sample both coordinate axis monotonically. As a consequence, we allow only horizontal edges to the right, vertical edges towards the bottom and diagonal edges that combine a horizontal and a vertical step.

Hence at each point-match, there exist only three different ways to proceed on the two curves. These three ways are represented by the following three outgoing edges:

1. One can proceed only on the first contour, which leads to edges of the form $(v_{i;j}, v_{i+1;j})$,

2. one can proceed only on the second contour, represented by edges $(v_{i;j}, v_{i;j+1})$ or

3. one can proceed one step on both contours. This is represented by edges of the form $(v_{i;j}, v_{i+1;j+1})$.

The problem of finding a shortest path from $(0,0)$ to $(2\pi, 2\pi)$ in the continuous plane can therefore be cast as finding the shortest path from $v_{0;0}$ to $v_{N;N}$ in the defined graph $G = (V, E, w)$. This can be computed very efficiently in $O(N^2)$ time steps [97].

This efficient computation relies extremely on the fact that we *know* an initial correspondence. This is in general not true. $D_{\mathcal{F}}(c_0, c_1)$ can therefore be calculated by finding an initial correspondence $(a, 0)$ and afterwards looking for a path of minimal weighted length from $(a, 0)$ to $(a + N, N)$. Since there is no natural way in finding such an initial point-match, computing the shortest circular path on the torus could be carried out with a brute-force method, where first an arbitrary initial correspondence is chosen, which is used as a starting point for computing a best matching afterwards. After repeating this for all possible initial correspondences, the global optimal match is computed at the cost of $O(N^3)$ computation steps [65, 38]. This can be achieved by expanding the involved graph to a $(N+1)\times(2N)$-grid (cf. right hand side of Figure 4.7).

In this section, we will show that the shape matching problem can in fact be solved in $O(N^2 \log(N))$. We developed two different methods that have this property. One method is an extension of [61] that applies a binary search to the DTW approach. Another approach is based on a reformulation of the shortest path problem as a graph cut problem. This will be the focus of Section 4.5.

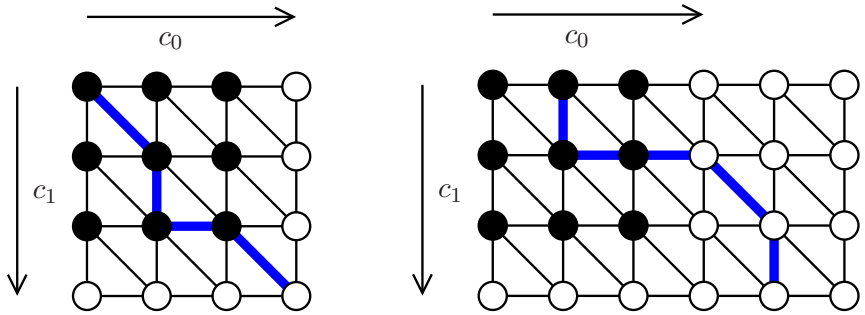



**Figure 4.7: Shape Matching Graph:** *Left hand side:* Matching $c_0$ onto $c_1$ results in finding a shortest path in a regular graph. To this end, the node information have to be carried forward to the right and to the bottom (blank nodes). *Right hand side:* To find the initial match, the graph has to be expanded to the right.

### 4.4.1 Efficient Shortest Cyclic Paths on a Torus

As we have seen in Section 4.4, the minimal matching of planar shapes can be cast as a problem of finding the shortest path through a graph. This graph is spanned by the two shapes, where the nodes of the graph encode the local similarity of respective points on each contour. While this problem can be solved using Dynamic Time Warping, the complete search over the initial correspondence leads to a cubic runtime in the number $N$ of sample points.

We propose an algorithm to determine this shortest cycle which has provably sub-cubic runtime. In Section 4.4.2, we will see that this method is faster than all other methods presented in this section. To the best of our knowledge, there is no algorithm that is faster than the one we are presenting here.

When searching for the shortest circular path, we can restrict the

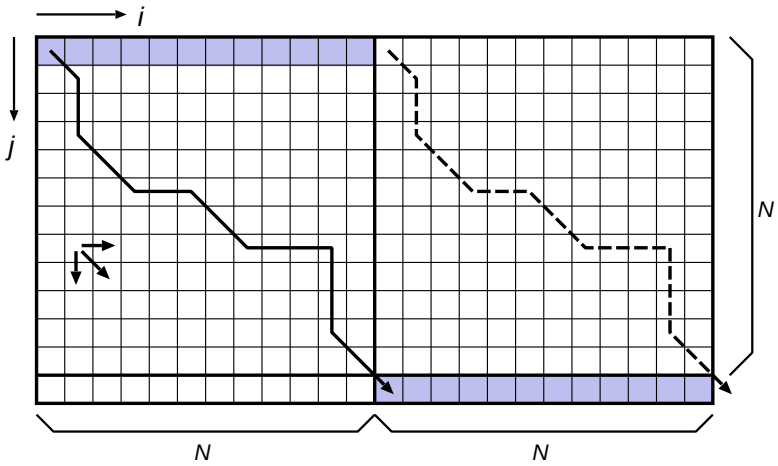



**Figure 4.8: Search area.** The path search is carried out on a graph of twice the size of the input matrix. A viable solution has to start in the top-left blue area and end at a matching node in the bottom-right blue area. The three allowed directions of the edges are indicated with the three arrows on the left side.

search w.l.o.g. such that the start-node of the path is of the form $v_{i;0}$, $i = 0, \ldots, N-1$. The corresponding end-node is then $v_{i+N;N}$ (see Figure 4.8).

In the description of the algorithm for computing the shortest circular path, we will apply the following theorem:

**Theorem 4.** *Let $G = (V, E, w)$ with $V = \{v_i\}_i$ be a graph and let $p_1 = v_{i_1} \ldots v_{i_n}$ and $p_2 = v_{j_1} \ldots v_{j_m}$ be paths of minimal length. Then we can state that if $p_1$ and $p_2$ have two nodes $v_p$ and $v_q$ in common, there is a path $p_2' = v_{k_1} \ldots v_{k_l}$ with the same weighted length as $p_2$, which has a common sub-path $v_p \rightsquigarrow v_q$ with path $p_1$.*

This theorem is based on the *principle of minimality* for shortest-

path methods, namely that any sub-path of a minimal path is itself a minimal path. A direct consequence of this theorem is that for any two minimum-length paths $p_1, p_2$, there exist two paths $p_1', p_2'$ with the same start and end nodes, which cross at most once. This property allows us to reduce the search area by constraining it on each side with a previously computed minimum-cost path.

The algorithm for computing the minimum cost path $v_{i;0} \rightsquigarrow v_{i+N;N}$ with unknown $i \in \{0, \ldots, N-1\}$ proceeds as follows.

- **Step 1:** The shortest path $p_l$ from $v_{0;0}$ to $v_{N;N}$ is computed with a standard DTW algorithm. Furthermore, we define the path $p_r$ as a copy of $p_l$, shifted by $N$ elements in the $i$-direction, i.e., $v_{N;0} \rightsquigarrow v_{2N;N}$. The paths $p_l$ and $p_r$ are depicted in Figure 4.8 as the bold path and the dashed path, respectively. Note that these two paths constitute boundary paths which reduce the search area from $2N^2$ to at most $N^2 + 2N$ nodes.

- **Step 2:** This step makes use of previously defined left and right bounding paths $p_l = v_{l;0} \rightsquigarrow v_{l+N;N}$ and $p_r = v_{r;0} \rightsquigarrow v_{r+N;N}$. In the first iteration, the paths are taken from the result of Step 1, so that $l = 0$ and $r = N$. In later iterations, other bounding paths will be used.

  We now compute a shortest-path tree, starting from the middle node $v_c = v_{(l+r)/2;0}$ at the top side of the graph (Figure 4.9). In one run of the DTW algorithm, we obtain all shortest paths to the nodes $v_{k;N}$ with $k \in l+N, \ldots, r+N$ at the bottom side. Considering Theorem 4, we can limit the DTW computation to the area between the two bounding paths $p_l, p_r$.

- **Step 3:** If we consider the shortest paths between $v_c$ and the bottom side of the graph, we see that the path $v_c \rightsquigarrow v_{l+N,N}$ obviously has a common sub-path with $p_l$, since both paths end at the same node. As we consider destination nodes $v_{k;N}$ with $k > l + N$, there is generally a largest $k'$ with $l + N \leq$

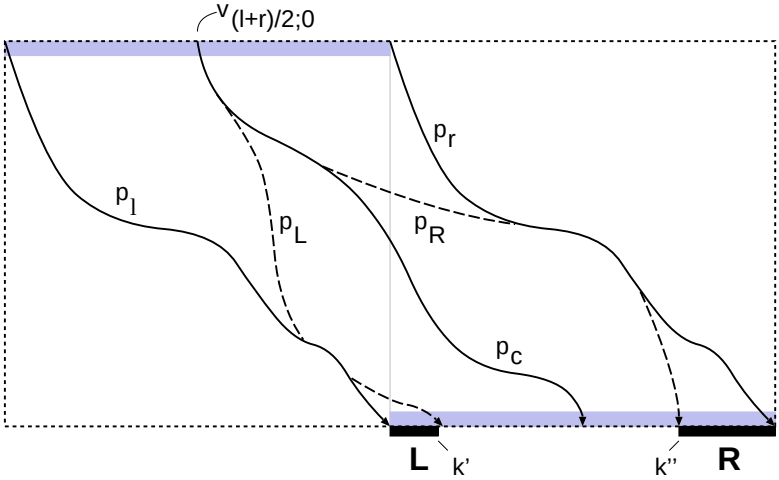



**Figure 4.9: Step 2 and 3.** A shortest-path tree rooted at $v_c = v_{(l+r)/2;0}$ is computed. The right-most node $v_{k',N}$ for which $v_c \rightsquigarrow v_{k',N}$ still has a sub-path with $v_l$ is determined. The shortest path to this node is denoted as $p_L$. Similarly, $p_R$ is determined as the left-most path that still has a sub-path with $p_r$.

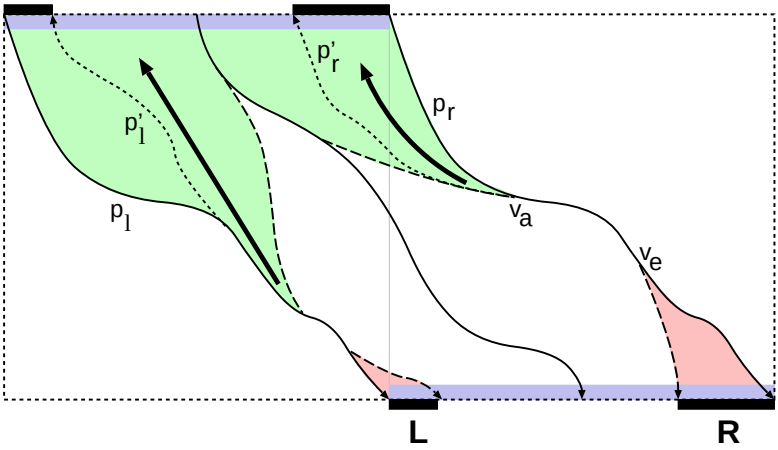



**Figure 4.10: Step 4.** Given $p_l$ and $p_r$, we know that all shortest circular paths ending in range $R$ include the sub-path $v_a \rightsquigarrow v_e$. Hence, build a shortest-path tree, rooted at $v_a$, in the indicated direction on the shaded area. By combining these paths with the sub-path $v_a \rightsquigarrow v_e$ and the shortest-path tree below $v_e$, all circular paths through the range $R$ can be obtained. The path $p'_r$ is the shortest circular path $v_{k'-N;0} \rightsquigarrow v_{k';N}$, which will be used as a bounding path in the following recursion step.

$k' \leq (l+r)/2 + N$ so that the shortest path $v_c \rightsquigarrow v_{k';N}$ still has a common sub-path with $p_l$. Let us denote the path from $v_c$ to $v_{k';N}$ as $p_L$, like it is depicted in Figure 4.9.

Similarly, we can find a smallest $k''$ with $(l+r)/2 + N \leq k'' \leq r + N$ so that the shortest path $v_c \rightsquigarrow v_{k'';N}$ still has a common sub-path with $p_r$. This defines the shortest path $p_R = v_c, \ldots, v_{k'',N}$ (cf. Figure 4.9).

- **Step 4:** According to Theorem 4, any circular path $v_{i;0} \rightsquigarrow v_{i+N;N}$ with $i \in R = \{l+N, \ldots, k''\}$ includes the common

sub-path of $p_r$ and $p_R$. By denoting the first node of this sub-path as $v_a$ and the last node as $v_e$, the common sub-path of $p_r$ and $p_R$ is $v_a \rightsquigarrow v_e$. If the shortest path from $v_{i;0}$ to $v_{i+N;N}$ passes through the area bounded by $p_R$ and $p_r$, we have therefore found some *initial point-matches* of the shape matching problem, namely any vertex on the path $v_a \rightsquigarrow v_e$. Hence, this subproblem can be solved in one single DTW step. Since we have already computed a part of this DTW step, it suffices to use DTW to compute a shortest-path tree, rooted at $v_a$, extending to the top-left and bounded by $p_R$ and $p_r$ (cf. Figure 4.10). The shortest-path tree rooted at $v_e$, extending to the bottom-right and bounded by $p_R$ and $p_r$ was already computed as part of the shortest-path tree from $v_c$ in Step 2.

We now have the shortest-path tree rooted at $v_a$ to all nodes $v_{i;0}$, $i + N \in R$ and the shortest-path tree rooted at $v_e$ to all nodes $v_{i;N}$, $i \in R$. By considering the sum of the cumulative costs for pairs of nodes $v_{i;0}, v_{i+N;N}$, we can identify the shortest circular path for the node range $R$. A similar process can be carried out to find the shortest circular path for the node range $L = \{k'; \ldots, l + N\}$ (cf. Figure 4.10).

Finally, we use the two shortest-path trees like above to extract the shortest-circular paths $p'_r = v_{k'-N,0} \rightsquigarrow v_{k',N}$ and $p'_l = v_{k''-N,0} \rightsquigarrow v_{k'',N}$. These two paths will constitute new bounding paths for later processing iterations.

- **Step 5:** The previous step has computed shortest paths for the subgraph bounded by $p_l$ and $p'_l$ and the subgraph bounded by $p'_r$ and $p_r$. What remains, is the subgraph bounded by $p'_l$ and $p'_r$. Since we have already computed the shortest path $v_{c;0} \rightsquigarrow v_{c+N;N}$ denoted as $p_c$, we can now divide the remaining subgraph along $p_c$ into two subgraphs, bounded by the paths $p'_l$ to $p_c$ and $p_c$ to $p'_r$. Both of these subgraphs can be processed recursively by restarting the processing at Step 2 for each of

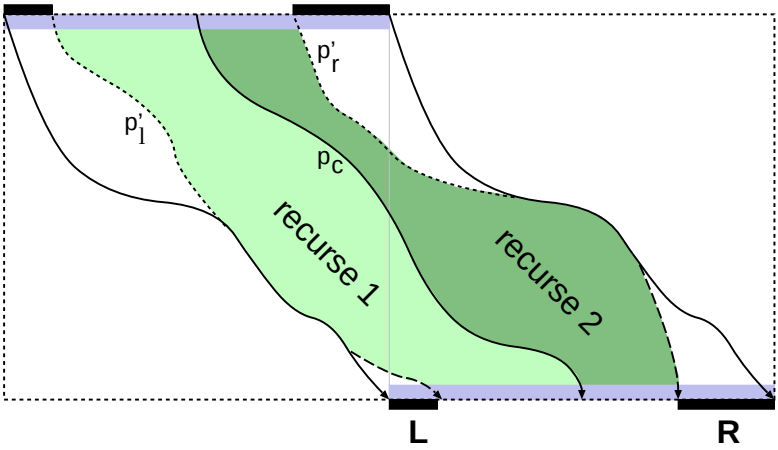



**Figure 4.11: Step 5.** All circular paths through the ranges $L$ and $R$ are already computed. Only the shortest circular paths in the range in between remain unknown. This range is processed recursively, first processing the graph between $p'_l$ and $p_c$, and then between $p_c$ and $p'_r$.

them. In the recursion, the new $p_l := p'_l$ and $p_r := p_c$ for the left subgraph, and $p_l := p_c$ and $p_r = p'_r$ for the right subgraph (Figure 4.11).

Each processing of Step 4 gives up to two candidate shortest circular-paths. Once the whole range of start nodes is processed, the path with the minimum-cost path is selected as the global solution.

The question that we like to address now, is how efficient is the proposed method to state-of-the-art methods. First of all, we will prove that the worst case complexity of our method is $O(N^2 \log(N))$ where $N$ is the number of sample points on each shape. Afterwards, a direct runtime comparison will be provided.

The proposed method defines a planar graph $G = (V, E)$ on a rectangular grid $V = \{v_{i;j} \mid 0 \leq i < 2N; 0 \leq j < N+1\}$. We are now looking for a shortest path from $v_{i;0}$ to $v_{i+N;N}$ where $i$ varies over the set $\{0, \ldots, N-1\}$. Since our method works recursively, we are especially interested in any connected subgraph $(V', E')$ of $G$ that includes a certain subset of the two boundaries $B_1 := \{v_{0;0}, \ldots, v_{N-1;0}\}$ at the top and $B_2 := \{v_{N;N}, \ldots, v_{2N-1;N}\}$ at the bottom of the original graph $G$.

**Lemma 5.** *Let $G' = (V', E')$ be a connected subgraph of $G$ containing exactly $b > 1$ corresponding boundary elements, i.e.*

$$v_{i;0} \in (V' \cap B_1) \Leftrightarrow v_{i+N;N} \in (V' \cap B_2),$$

*Then the algorithm finds the shortest path connecting an element $v_{i;0} \in B_1$ to its corresponding element $v_{i+N;N} \in B_2$. Moreover, for the number of calculation steps $T(n, b)$ the following holds:*

$$T(n, b) \leq 2(\log(b-1) + 1) \cdot (n + N(b-1)) \tag{4.6}$$

*where $n := |V'|$ is the number of vertices in $G'$.*

*Proof.* Since $G'$ is a subgraph of $G$, any shortest path given a start vertex and a target vertex can be calculated within $n$ calculation steps using Dynamic Time Warping (DTW). An upper bound for the runtime $T(n, b)$ is clearly $b \cdot n$ since every DTW run used by our method computes at least one shortest path from an initial node in $B_1$ towards the corresponding vertex in $B_2$. These properties will be used during the proof. We will prove this lemma by complete induction over the boundary length $b$.

**Initialization:** For $b = 2, 3, 4$, the expression on the right hand side of (4.6) is always greater than $b \cdot n$. Since $b \cdot n$ is an upper bound of the method, (4.6) holds for these cases.

**Induction step:** Assuming, we have proven the upper bound for all boundary lengths $b'$ fulfilling $4 \leq b' < b$. Now, we like to prove this upper bound for $b$ itself. First, the algorithm calculates a shortest path from the central point of the boundary which takes $n$ calculation steps. By doing so, the graph $G'$ is split into a left and a right subgraph with $n_L$ resp. $n_R$ vertices whereas $n_L + n_R \leq n + 2N$. If Step 3 and Step 4 has to be applied, an additional DTW run has to be computed. Overall, we obtain the following

$$\begin{aligned}
T(n,b) \leq & T\left(n_L, \frac{b+1}{2}\right) + T\left(n_R, \frac{b+1}{2}\right) + 2n \\
\leq & 2(\log(b-1)) \cdot (n_L + n_R + N(b-1)) + 2n \\
\leq & 2(\log(b-1)) \cdot (n + N(b+1)) + 2n \\
\leq & 2(\log(b-1)+1)(n + N(b+1)) - 2(b+1) \\
\leq & 2(\log(b-1)+1)(n + N(b-1)) + \\
& 2(\underbrace{2\log(b-1) - (b-1)}_{\leq 0}) \\
\leq & 2(\log(b-1)+1)(n + N(b-1))
\end{aligned}$$

□

With this lemma, the worst case runtime of the proposed matching algorithm can be characterized as follows.

**Theorem 5.** *The proposed shape matching algorithm has a worst-case complexity of* $O(N^2 \log(N))$.

*Proof.* Since $G = (V, E)$ is a subgraph of itself with $b = N$ boundary elements and $n \leq N^2 + 2N$ vertices, Lemma 5 guarantees that the shortest path can be calculated in less than

$$(4N^2 + 2N) \cdot (\log(N-1) + 1)$$

time steps. □

### 4.4.2 Experimental Comparisons

The above bound of $O(N^2 \log(N))$ on the computation time was proven based on a worst case analysis. In practice, the computation time is well below this worst case bound: As discussed in Step 4 of the algorithm, the algorithm cuts away parts of the graph, for which it can immediately compute the optimal solution. For most shape comparisons, such cases arise instantly, such that the recursion terminates after very few iterations.

Figure 4.12 shows a quantitative benchmark test of the proposed method and three state-of-the-art shape matching algorithms that we like to present briefly:

**Dynamic Time Warping** For every possible initial match $v_{i;0}$, we look for the shortest path from the point $v_{i;0}$ to the point $v_{i+N;0}$ within the graph $G = (V, E)$. To find a match, we always need $O(N^3)$ calculation steps.

**Branch and Bound** This method [1] starts with the initial matching set $S := \{v_{0;0}, \ldots, v_{0;N-1}\}$. Then the method looks for the shortest path within $G$ passing through $S$. If the shortest path is a valid matching, i.e. a cycle in the graph $G = (V, E)$, the method stops. Otherwise, $S$ will be subdivided. The worst case is still $O(N^3)$, but under certain conditions, an *average case* of $O(N^2 \log(N))$ is possible [1].

**Cyclic String Approach** In [62], Maes presented a shape matching approach that essentially uses Step 1, 2 and 5 of our approach. This approach provides a worst case complexity of $O(N^2 \log(N))$.

For two shapes and an increasing discretization level which ranges between 50 and 1000 points per shape, we compared the runtime of the described methods. While all algorithms compute the same

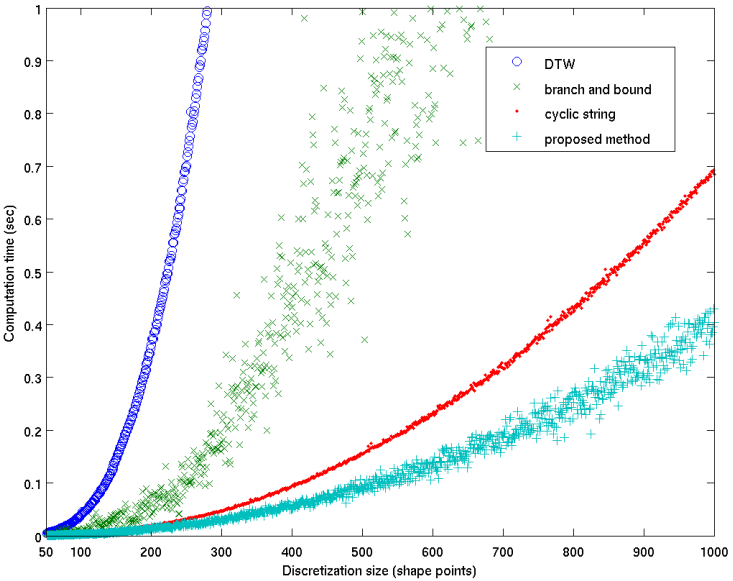



**Figure 4.12: Experimental runtime comparison.** In contrast to DTW, Branch-and-Bound and the cyclic-string approach [62], the proposed method exhibits consistently lower runtimes, in particular for larger problem sizes.

matching, the proposed method exhibits consistently lower runtimes, in particular for larger discretization. Moreover, it offers a more predictable performance, in the sense that the computation times exhibit a smaller spread than those of Branch and Bound.

## 4.5 Shape Matching as Graph Cut Problem

While a shortest path through the shape matching graph can be computed efficiently using *Dynamic Time Warping* (DTW), one of the key drawbacks of this approach is that Dynamic Time Warping requires a corresponding point pair for initialization. The most current methods therefore apply DTW for all possible initial correspondences, and then select the minimum of all computed shortest paths as the distance between the two shapes. We presented one of the most efficient formulation in Section 4.4.1. But in this method, too, the search for an initial point-match is done independently of the search for the complete match. In the following, we want to overcome this restriction.

Our goal is to develop a method that does not pursue a decoupling of the search for the initial match from the search of the complete match. To this end, we show that shape matching can be cast as a problem of finding a minimal cut through a network $G = (V, E, c, s, t)$. Such a network consists of a set of vertices $V$ that are connected via oriented edges $E \subset V \times V$ whereas the source $s \in V$ provides only outgoing edges and the sink $t \in V$ only incoming edges. Every edge $e = (u, v)$ is equipped with a positive capacity $c(e) \in \mathbb{R}^+$ and we call $C \subset E$ an $st$-cut if there is no path from the source $s \in V$ to the sink $t \in V$ in the reduced network $G_C = (V, (E \setminus C), c, s, t)$. The problem of finding a *minimal cut* $C$ can then be formulated as:

A lot of classical Computer Vision problems have recently been ad-

**Problem 1** MINIMAL CUT
**Input:** Graph network $G = (V, E, c, s, t)$
**Output:** $st$-Cut $C \subset E$ which minimizes $\sum_{e \in C} c(e)$

dressed by graph cut approaches, because they allow to efficiently solve the underlying labeling or correspondence problems in a globally optimal manner. In particular, researchers have employed graph cuts for stereo reconstruction with convex neighborhood potentials [13], for image segmentation [11, 41, 76], for image and video synthesis [56] or for multi-view reconstruction [88, 96].

We will now show how the shape matching graph $G$ of Section 4.4 can be transformed into a network $Z^*$ on which we can apply a graph cut method in order to solve the shape matching problem. The graph $G = (V, E, w)$ of the DTW approach comprises a planar grid structure and in order to find an optimal match, we look for shortest paths from $v_{i;0}$ to $v_{i+N;N}$ for all possible $i = 0, \ldots, N-1$. If we now identify any start-vertex $v_{i;0}$ of the graph with its target-vertex $v_{i+N;N}$, the graph $G$ becomes a cylinder and the formerly shortest path describes a shortest cycle on this cylindrical graph $Z = (V_Z, E_Z, w_Z)$. Note that every such shortest path separates the two outer boundaries of the graph $Z$ (green edges in Figure 4.13).

It is known that also cylindrical graphs like $Z$ can be embedded into the plane. Thus, the graph that we obtain after identifying the corresponding vertices can still be embedded into the plane $\mathbb{C}$. This means that the edges cut the plane open into different faces that we pool in the set $F$. An import property of planar graphs is that two different edges may *not cross* each other. Therefore, every edge $e \in E_Z$ has a well-defined left face $f_l(e) \in F$ and a well-defined right face $f_r(e) \in F$. To the cylindrical graph $Z$, we can therefore define the dual graph $Z^* := (F, E_Z^*, w_Z^*)$ by introducing dual edges $e^* := (f_l(e), f_r(e))$ which connect the left with the right face of an edge. To every dual edge $e^*$ we canonically assign the

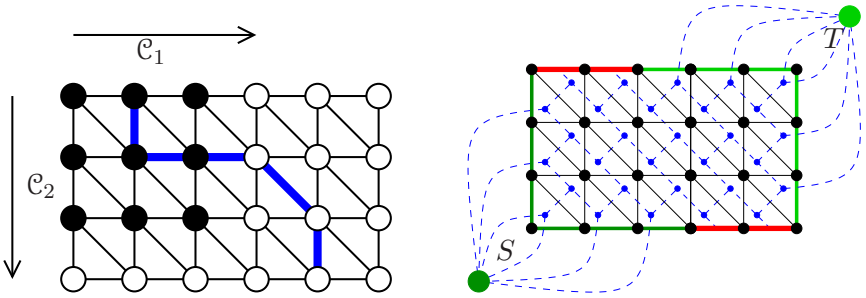



**Figure 4.13: Dual Graph.** A matching path in $G$ (left hand side) describes a cycle on the cylinder $Z$ (right hand side). $Z$ is constructed by identifying the vertices along the red edges. The green edges form the boundaries of $Z$. Cycles in $Z$ are equivalent to $S-T$-cuts in the dual cylinder $Z^*$ defined by the dashed edges.

weight $w_Z^*(e^*) := w_Z(e)$. Additionally, the two faces that are formed by the outer boundaries of the cylinder $Z$ will be denoted as source $S$ and sink $T$ (cf. Figure 4.13).

Interestingly, Whitney showed in [99] that for any planar graph $Z$, there is a one-to-one relationship between cycles on $Z$ and cuts in the dual graph $Z^*$. Therefore, the value of a minimal edge cut will be $D_{\mathcal{F}}(f_0, f_1)^2$ for given feature loops $f_0$ and $f_1$. Mathematically, this can be summarized in the following theorem.

**Theorem 6.** *Let $f_0$ and $f_1$ be two feature loops with respect to a feature space $(\mathcal{F}, d, F)$. Then, the following equation holds*

$$D_{\mathcal{F}}(f_0, f_1)^2 = \min_{\substack{C^* \subset E^* \\ C^* \text{ st-cut of } Z^*}} \sum_{e^* \in C^*} w_Z(e^*) \tag{4.7}$$

To solve this graph cut problem, we can use the method provided by Boykov and Kolmogorov [13] which works very fast for shape

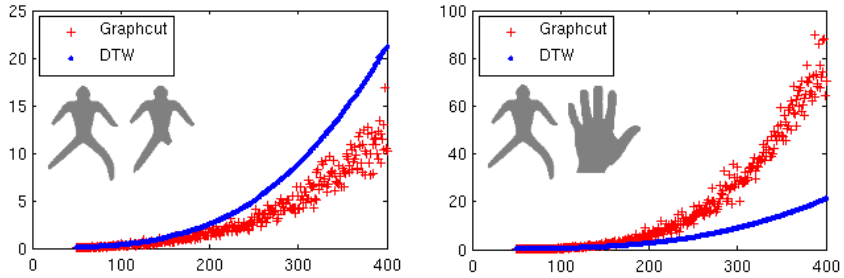



**Figure 4.14: Runtimes of shape matching.** Here, the runtime of the proposes graph cut method and the commonly known shortest path method using DTW is plotted against the sampling rate of both given shapes. We can see that there are cases where the graph cut method outperforms the DTW method and vice versa.

acquisition problems. In fact, a linear time could be empirically observed. Unfortunately, this algorithm is not as quick as expected for the dual shape matching cylinder $Z^*$. In Figure 4.14, we see that for similar shapes the proposed method outperforms the classical DTW method. On the other hand, for different shapes the opposite is the case. Therefore, it looks like the graph cut method handles similar shapes quite easier than dissimilar shapes. This is because for similar shapes $f_0, f_1$, the distance $D_{\mathcal{F}}(f_0, f_1)$ and thus the maximum flow is quite small. In other words, the maximum flow is close to the initial flow which is zero. Therefore, the amount of augmented paths that has to be examined by the graph cut algorithm is rather small and the proposed method works very rapidly in comparison to the classical DTW approach.

Overall the observed results are encouraging in the sense that the coupling of the initial point-match search with the complete match-search can be cast as a graph cut problem. Nonetheless, the used graph cut method does not perform very well on this specific graph. Hence, we have to develop another graph cut method that is bet-

ter designed for the shape matching graph $Z^*$. In Section 4.5.1 we will show that by exploiting the planarity of the involved graph, we are able to provide a very fast graph cut-based method for shape matching. For example, this method needs only $O(N^2 \log(N))$ computation steps to compute a shape matching if both involved shapes are discretized by $N$ points.

### 4.5.1 Efficient Planar Graph Cuts

In Section 4.5, we saw that the problem of shape matching can be cast as finding the minimal cut in a graph. In order to do this efficiently, we want to present a fast graph cut method for planar graphs. The method that we are about to present is based on the graph theoretical work of Borradaile and Klein [10, 9] and leads to an efficient method that we will then apply to shape matching. In Section 4.5.2, we will provide a runtime test of this method applied to the shape matching graph $Z^*$ of Section 4.5. For now, we assume that we have to deal with a general planar graph network $G = (V, E, F, f_l, f_r, c, s, t)$ where

$$
\begin{aligned}
V &\text{ denotes the set of vertices,}\\
E \subset V \times V &\text{ denotes the set of oriented edges,}\\
F &\text{ denotes the set of faces,}\\
f_l : E \to F &\text{ assigns to every edge its } \textit{left face},\\
f_r : E \to F &\text{ assigns to every edge its } \textit{right face},\\
c : E \to \mathbb{R}_0^+ &\text{ assigns to every edge a non-negative capacity,}\\
s \in V &\text{ denotes the } \textit{source} \text{ of the network and}\\
t \in V &\text{ denotes the } \textit{sink} \text{ of the network.}
\end{aligned}
$$

To date the graph cut algorithm of Boykov and Kolmogorov is con-

sidered to be the fastest existing algorithm for Computer Vision applications [13]. Nonetheless, there is no known polynomial upper bound for the runtime of the algorithm of Boykov and Kolmogorov. A lot of effort has been put into improving the runtime of maximum flow computation, by means of flow recycling [53], capacity scaling [47] or multi-scaling [29]. While these strategies often lead to reduced computation times, none of them reduces the worst case complexity of the methods that they were built on. The runtime tests of Section 4.5 (cf. Figure 4.14) demonstrate the lack of a strong upper bound on the graph cut computation time.

Nonetheless there are methods to compute the minimal cut in provable polynomial runtime. In the following we like to briefly present the development in this area. A major milestone to solve the general graph cut problem was the Min-Cut-Max-Flow theorem of Ford and Fulkerson [36] which stated that the MINIMAL CUT problem is equivalent to solving the MAXIMAL FLOW problem. A flow $f : E \to \mathbb{R}_0^+$ assigns to every edge a non-negative value that is bounded from above by the capacity function $c$. Additionally, the amount of *incoming flow* to an edge should be identical to the amount of *outgoing flow*:

$$\forall v \in V \setminus \{s,t\} : \sum_{e=(u,v)\in E} f(e) = \sum_{e=(v,u)\in E} f(e)$$

The problem of finding the maximal flow can then be cast as:

**Problem 2** MAXIMAL FLOW
**Input:** Graph network $G = (V, E, c, s, t)$
**Output:** Flow $f \leq c$ maximizing $\sum_{(v,t)\in E} f(v,t)$

The main idea to solve the MAXIMAL FLOW problem is to start with a flow $f : E \to \mathbb{R}_0^+$ that assigns 0 to every edge $e \in E$ and then to augment the flow along paths from source to sink on which none of the involved edges are saturated, i.e., $f(e) < c(e)$. This *augmenting path strategy* solves the problem and if we are only using

shortest paths, the problem can be solved in polynomial time [30]. Nonetheless, this runtime is in general too high.

On the other hand, Weihe showed [98] that an almost linear runtime[1] is an upper bound for planar networks. Unfortunately, the proposed method requires a rather complicated preprocessing step and hence is not ideally suited for practical implementations. To overcome this drawback, in [9] a new method was proposed that uses a simpler preprocessing step. The core idea of this method is to always augment the *leftmost* of all paths from source to sink. To this end, we have to store all leftmost paths towards the sink $t$ in a suitable data structure, namely in a tree spanning all vertices of the graph. One way to compute such a tree is the *right-first search*. This is a depth-first search of the graph where we always consider the edge that is situated *as right as possible*. This simple method produces certain difficulties if there are clockwise cycles in the graph. Hence, it makes sense to eliminate these clockwise cycles prior to the computation of the right-first search tree:

**Algorithm 4** Planar Maximal Flow [9]

**Input:** Planar Graph network $G = (V, E, F, f_l, f_r, c, s, t)$
**Output:** Maximal Flow $f : E \to \mathbb{R}^+$

1: Remove from G all clockwise cycles
2: Initialize the flow f with 0
3: **while** there is a non-saturated path from $s$ to $t$ **do**
4: saturate the leftmost path from $s$ to $t$
5: **end while**
6: return f

It has been shown that the elimination of clockwise cycles can be done quite elegantly by computing a shortest path tree through the dual graph [9]. Therefore, Step 1 of Algorithm 4 is a preprocessing

[1] By *almost*, we are referring to the $\widetilde{O}$-notation, i.e., $n \log(n) = \widetilde{O}(n)$.

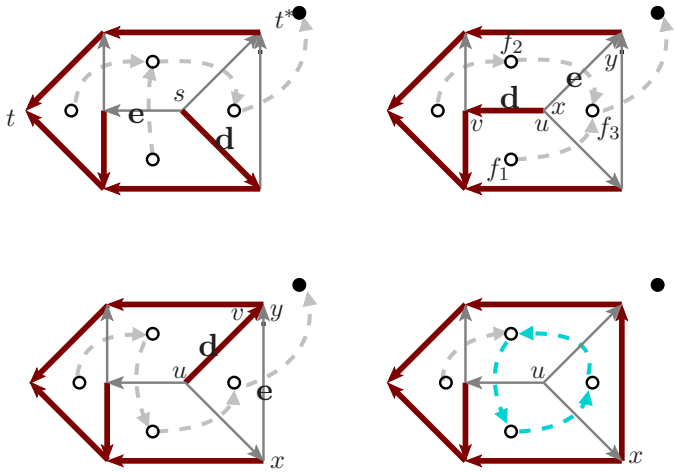



**Figure 4.15: Planar max flow method.** Bold and dashed edges indicate the spanning trees $T$ and $T^*$ resp. At every step, an edge $d$ of the tree T is substituted by an edge $e$. When the method terminates (bottom right), neither $T$ nor $T^*$ are trees anymore. Moreover, in $T^*$ a circle emerges which describes the minimal cut.

step which can be computed in $O(N \log N)$. A challenging implementation task is in fact Step 4. Like any augmenting path method, it (cf. Algorithm 5) maintains a spanning tree $T$ of all vertices $V$ to keep track of the augmenting path from source to sink efficiently. Additionally, the method also handles a spanning tree $T^*$ of all faces $F$ to support the updating scheme of $T$. Since the graph is planar, all edges which are not in $T$ form the tree $T^*$ (cf. Figure 4.15). Lines 8, 9 and 13 of Algorithm 5 take care of this invariant.

**Algorithm 5** Implementation of Step 4 [9]

1: Let $T$ be the right-first search tree backward from $t$.
2: Let $T^*$ be the spanning tree of $F$ consisting of all edges of $E - T$.
3: **repeat**
4: Augment path from $s$ to $t$, update the flow $f$ and let $d = (u, v)$ the closest edge to $t$ which is saturated.
5: Let $(f_1, f_2)$ the dual edge $d^*$ of $d$.
6: Let $e^* = (f_2, f_3)$ be the parent edge of $f_2$ in $T^*$.
7: Let $e = (x, y)$ be the primal edge with respect to $e^*$.
8: $T^* := T^* + \{(f_2, f_1)\} - \{(f_2, f_3)\}$.
9: $T := T - \{d\} + \{e\}$.
10: **if** $f_1$ is a descendent of $f_2$ within $T^*$ **then**
11: return f
12: **end if**
13: Reverse in $T$ the edges along the path from $x$ to $u$.
14: **until false**

It has been shown [9] that the **repeat**-loop is repeated at most $O(N)$ times and that the usage of Dynamic Tree [89] for $T$ and Euler Tour Tree [45] for $T^*$ results in an $O(\log N)$-runtime for the Lines 4-13. Hence, the method computes the maximal flow in $O(N \log N)$.

Nonetheless, the test for the termination condition (Line 10) is a bottleneck of Algorithm 5. The theoretical contribution of our work is to get rid of the Euler Tour Tree and instead maintain $T^*$ by an

array which stores the parent of each face. Originally, the Euler Tour Tree was used in order to test Line 10 in $O(\log N)$. However, modification and parent access (Line 6 and 8) of $T^*$ then take the same amount of time. We therefore propose an equivalent test on $T$ instead of $T^*$. We will present this alternative test and prove its equivalence in Theorem 7. Instead of Lines 10-12, we perform the test of Algorithm 6.

**Algorithm 6** Alternative Termination Condition for $T$

```
10: if there is no path from x to u in T then
11:     return f
12: end if
```

Surprisingly, the new test takes no additional time, since the path from $x$ to $u$ has to be identified in Line 13 anyway. Furthermore, the $T^*$-related Lines 6 and 8 can now be done in $O(1)$ instead of $O(\log N)$. This runtime reduction makes this method attractive for shape matching as we will see in Section 4.5.2. The following theorem proves the correctness and efficiency of the proposed method:

**Theorem 7.** *The proposed method solves the Maximum Flow problem in $O(N \log N)$.*

*Proof.* Our approach substitutes Lines 10–12 of Algorithm 5 with Algorithm 6. As long as $f_1$ is not a descendant of $f_2$, both approaches do not differ from one another. This is due to the fact that Line 13 of Algorithm 5 implies the existence of a path from $x$ to $u$ in $T$. Therefore, let us now assume that $f_1$ is in fact a descendant of $f_2$. Then, there exists a path from $f_1$ to $f_2$ in the dual tree $T^*$. In Line 8 the dual edge $(f_2, f_1)$ is inserted into $T^*$ and this data structure possesses now a counter clockwise circle which encloses the vertex $u$ (cf. Figure 4.15). Since $e^*$ was an edge that left $f_2$, the two vertices $x$ and $y$ are now outside of the just constructed circle. Therefore,

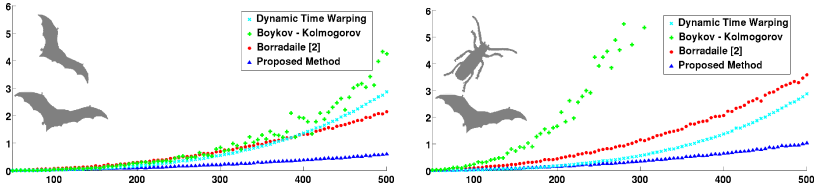



**Figure 4.16: Runtimes of Shape Matching.**The runtime on the $y$-axis is measured in seconds and depends on the amount of shape points of each shape. The proposed method outperforms the existing method of Boykov and Kolmogorov and is also faster than Dynamic Programming. The speed-up factor increases with an increasing shape resolution.

Line 10 of Algorithm 6 cannot find a path from $x$ to $u$ and the proposed method terminates returning the same flow as Algorithm 5.

Finally, we will prove that the extension (Algorithm 6) of the Algorithm 5 does not increase the runtime. The Dynamic Tree [89] handles Line 13 as follows. Starting from $x$, it attaches Dynamic Paths until either $u$ or the root $t$ of $T$ is found. If the sink $t$ is found, we know that there is no path from $x$ to $u$. Otherwise, a path from $x$ to $u$ was explicitly found and can be processed in Line 13. In both cases, the test of Line 10 in Algorithm 6 is natural extension of Line 13 and does not consume any additional computation time. Hence, the proposed method is always faster than that presented in [9]. □

### 4.5.2 Efficient Shape Matching via Planar Graph Cuts

As we have seen in Section 4.5, the shape matching problem can be formulated as finding the minimum cut through the planar graph network $Z^*$. Because of the planarity of $Z^*$, we can apply the presented graph cut method of Section 4.5.1. The size of the graph

is $O(N^2)$ and the proposed method runs therefore in $O(N^2 \log N)$. In Figure 4.16, the runtime for two different examples are given. As we can see, the presented implementation outperforms the other methods and provides a speed-up factor of 2–4 with respect to the the original work of Borradaile. Interestingly, the graph cut approaches are still faster at the presence of similar shapes. This fact was already observed in Section 4.5 for the method of Boykov and Kolmogorov. Nonetheless the method is slower than the efficient DTW-based method presented in Section 4.4.1. The advantage of this method lies in its formulation, namely that the shape matching problem can be cast as a graph cut problem and can be solved efficiently without iterating over possible initial matches. Hopefully, in the future there will be more efficient methods for graph cuts such that a graph cut based shape matching is competitive with DTW-based methods.

## 4.6 Shape Clustering

In this section, we will study whether the distance function (4.4) emulates human notion for object similarity. In order to demonstrate this, we conduct a clustering test on a given database of contours $c_1, \ldots, c_n$. To this end, we first select a feature space $(\mathcal{F}, d, F)$ and transform the given contours in feature loops $f_1, \ldots, f_n$ with $f_i = F(c_i)$. Subsequently, we compute the matrix $D$ of pairwise distances

$$D_{ij} = D_{\mathcal{F}}(f_i, f_j), \quad \forall\, i, j = 1, \ldots, n.$$

Afterwards, we test whether shapes of small pointwise distances look alike. In order to do this, we used two different annotated databases, i.e. databases that also provide information which shapes should be similar to one another. In Section 4.6.1, we used the curvature feature of Section 4.2.1 and applied it to a subset of the LEMS database [87]. In Section 4.6.2, we applied the Inner Shape Con-

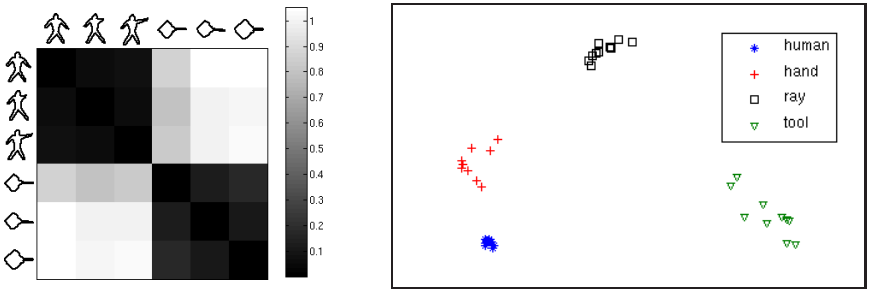



**Figure 4.17: Clustering.** On the left hand side, the pairwise dissimilarity of six given shapes according to $\text{dist}(\cdot,\cdot)$ are color-coded. On the right hand side, 40 shapes are projected into the Euclidean plane based on their pairwise distance. In general, this projection will not preserve pairwise distances since $\text{dist}(\cdot,\cdot)$ is not a metric. But even this approximation indicates that the distance function incorporates the human notion of shape similarity.

text [60] to the MPEG7 shape database[2]. For both databases we obtained promising results.

### 4.6.1 LEMS Database

The LEMS [87] database that we want to use here consists of 99 different contours. To cluster this database is considered to be rather simple. We use this database just to show that the proposed curvature feature of Section 4.2.1 provides good matching results. From this database, we selected 40 contours which describe the shape classes *hand*, *human*, *ray* and *tool*.

To visualize the computed result, we would like to plot the distance

[2]The shape database *MPEG7 CE Shape-1 Part B* is online available and can be downloaded at `http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz`

result in the real plane. Since the shape space is not a metric, there is no natural projection from the shape space into the real plane. Hence, we need an $\mathbb{R}^2$ approximation of the shape space spanned by the feature loops $f_i$. To this end, we applied the method of multi-dimensional scaling [54] that optimally preserves the pairwise distances, i.e. we computed a set of points

$$\left\{x_1, \ldots, x_n \in \mathbb{R}^2, \text{ such that } |x_i - x_j|^2 \approx D_{ij}^2 \; \forall i, j\right\}.$$

Figure 4.17 shows these 2D-points with their cluster membership color-coded. The clear separation of four clusters associated with the four shape classes indicates that the computed pairwise distances reproduce the human notion of shape similarity for this data base. Obviously, it suffices to use the provided distance as a measure of dissimilarity only. The shape clustering provided by the distance function coincides exactly with the human classification of shapes. It is in fact possible to separate human shapes and shapes of tools, rays and hands from one another.

### 4.6.2 MPEG7 Database

A very challenging shape database is the MPEG7-database (cf. Figure 4.18). It consists of 70 different shape classes which are represented by 20 different shapes each. Some of the retrieving results are presented in Figure 4.18.

For the so called *Bull's Eye Test*, one calculates the 40 closest shapes to a given shape according to the used distance function. If among these shapes, there are $k$ shapes of the same class as the query shape, the retrieval rate is $\frac{k}{20}$. The mean of all 1400 retrieval rates is the retrieval rate of the database. As we can see in Table 4.1, the retrieval rate depends on the discretization size. For 500 shape points of every shape, the retrieval rate increases up to almost 76%. Besides this important retrieval rate, we are also interested in how good the

| Shape Points | Bull's Eye | Worst Case | Optimal Classes |
|---|---|---|---|
| 100 | 67.07% | 15.5 % | 13 |
| 200 | 73.35% | 24.5 % | 14 |
| 300 | 75.04% | 25.7 % | 15 |
| 400 | 75.71% | 25.0 % | 15 |
| 500 | 75.97% | 26.7 % | 16 |

**Table 4.1: MPEG7 retrieval rate.** Some retrieval rates for the MPEG7 shape database are provided. The results are given with respect to the used discretization size (1st row). Besides the retrieval rate of the Bull's Eye Test (2nd row), the average retrieval rate of the worst performing class (3rd row) and the numbers of optimal performing classes (4th) is given.

retrieval for the most challenging class works. For 100 shape points the worst average retrieval rate within one class is 15.5% whereas this values increases to 26.6% for a fine discretization size of 500 shape points. Also the amount of classes that provide for a 100% retrieval rate with respect to the *Bull's Eye test* increases from 13 to 16 classes. Overall, we see that increasing the discretization size improves the performance of a shape matching in general. For the *Bull's Eye Test*, almost 2 million pairwise shape distances have to be computed. Hence, efficient shape matching methods like the one presented in this section are necessary to obtain good retrieval rates. To perform the Bull's Eye Test on a standard PC with 500 shape points, this method took about 84 hours which is considerably faster than a purely DTW based method.

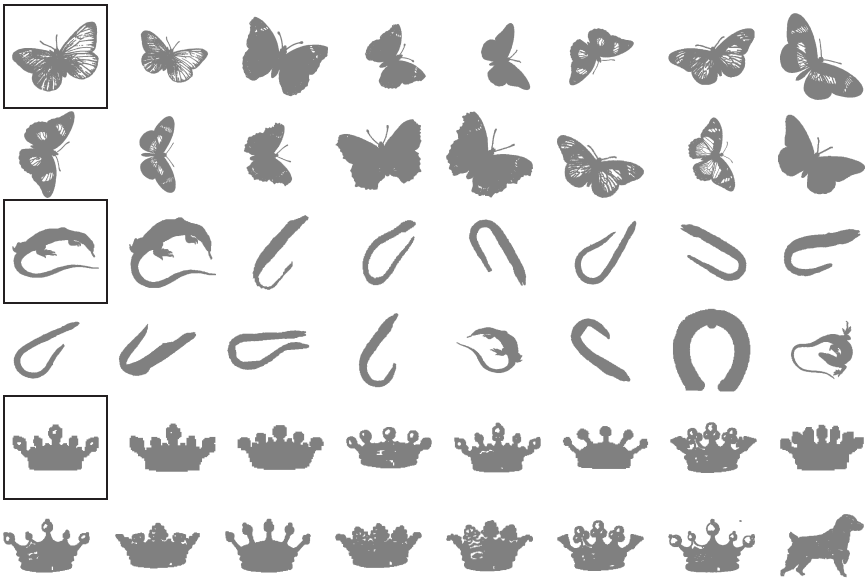


**Figure 4.18: MPEG7 retrieval samples.** While for shape classes that are easily confused, retrieval rates can drop below 50% (shape class *Lizard*), the average retrieval rate is above 75%. The proposed graph cut method allows to substantially accelerate the required shape distance computation.

# Chapter 5

# Conclusion

## 5.1 Summary

In this work we presented certain efficient methods to solve classical problems of Shape Analysis. In order to do so, we presented three different definitions of *shape*. In Chapter 2, we introduced the concept of stochastic shapes that assign to every pixel of the image domain $\Omega \subset \mathbb{R}^2$ the probability that this pixel is part of the presented object. As a consequence, this shape model is a region-based model. In the following chapters, we focused on contour-based shape models.

In Chapter 3, we introduced group operations under which a shape should be invariant. A shape was then defined as the set of all smooth curves (parameterized via immersions) divided by these group operations. Here we differentiated between two different group operations. The first group operated as a reparameterization group and the other group modeled rigid body transformation in the image domain. Dividing both groups out of the set of smooth curves, we received the set of shapes. This very technical approach towards the concept of

shapes has the advantage that every shape can be represented via a function $\theta : [0; 2\pi] \to \mathbb{R}$ and all these functions form an orbifold on which we can compute a geodesic. Nonetheless, this model allows only curves that are parameterized by arc-length. This restriction was corrected with the shape concept that we used for shape matching.

In Chapter 4 the problem of abstracting from a contour was solved by using a certain feature space that describes a contour in a way that this description is invariant with respect to rigid body transformation. Instead of abstracting from the whole set of contours $c : \mathbb{S}^1 \to \mathbb{C}$, the abstraction happens at every point of the contour. Every point $c(s)$ is described with respect to its neighboring points such that this local descriptor does not change if a rigid body transformation is applied to the whole curve. We called these shapes *feature loops*.

For all these different concepts of shape, we presented methods to solve classical Computer Vision problems. The focus of this work was on the efficiency of these methods and in certain cases we could also prove that the presented methods always provide a global optimal solution with respect to certain energy functionals.

## Shape Acquisition

The problem of acquiring a shape from an image was solved via a functional that combines classical image segmentation functionals with shape prior models. As purely image segmentation functionals, Mumford-Shah like functionals and Geodesic Active Contour like functionals are possible. We also presented three very classical shape priors that can be used in order to obtain a functional that is convex with respect to the used stochastic shape model. This shape model is as far as we know one of the first convex shape models every designed for shape prior driven image segmentation.

The advantage of the presented method over other methods lies in the fact that we can quickly find the global optimum of the used functional with respect to possible deformations. This is possible because the considered functional is a convex functional over the convex domain of possible shapes.

Secondly, we could benefit from the fact that the considered energy functional is Lipschitz continuous with respect to rigid body transformations. Thus, a Lipschitz optimization technique could be used in order to globally optimize the given functional also over possible rigid body transformations. In general, a Lipschitz optimization is slower than gradient descent methods. Thus, it makes sense to put much effort into convexifying the shape model, because afterwards only an optimizing over a relatively slow space has to be considered, namely the 3-dimensional space of rigid body transformations.

Knowing that we always find the global optimum of the specified energy functional helped us to detect occlusions by simply considering the data term of the involved image segmentation neglecting the part of the energy functional that controls the shape prior.

Concluding, we were able to globally optimize a quite general variational approach for image acquisition which is governed by shape prior and can reliably detect occlusions.

### Shape Morphing

In order to solve the problem of shape morphing, we presented a completely new method of finding a geodesic on an arbitrary submanifold. This quite general approach rejects the commonly used *shooting method* and proposes instead a *path-shortening* method. This method applies a gradient descent method on an energy functional. Thus, it is not necessary to exactly solve either a *partial differential equation* or an *ordinary differential equation*. Instead we just shorten a given path until a geodesic is found.

One important advantage of the path-shortening method is its efficiency. We showed that it is faster by a factor of 1000 depending on the used resolution. This efficiency boost is in fact not the only advantage of the proposed method. We also showed that not only the formulation of the method but also the computed distances on the considered shape space is symmetric. Thus, it makes sense to compute the induced *shape metric* with this method in order to obtain a symmetric distance function.

### Shape Matching

For the problem of shape matching we proposed two different approaches which both resulted in a worst-case complexity of $O(N^2 \log(N))$ if both shapes are given as $N$ discrete, ordered points. This is an important contribution since a lot of researchers of the Computer Vision community still use methods that have a worst-case complexity of $O(N^3)$. The first approach that we presented was based on a DTW graph and the problem that had to be solved was the shortest path problem. In fact, not only one but $N$ different paths in a grid of size $O(N^2)$ had to be computed. The presented method is an extension on the method of Maes who proposed to subsequently subdivide the set of initial match-points and the involved graph. Our method has the advantage that it performs not only a subdivision of the graph, but it also cuts off certain areas of the graph and thus, improves the run-time dramatically. As a result, a shape matching for two shapes that are discretized each by $N = 1000$ shape-points could be computed in about 400 milliseconds. We also demonstrated that the proposed method does not only provide the best worst-case complexity with respect to all previously proposed methods. It is also in practice much faster than probabilistically motivated methods like the popular Branch-and-Bound method.

The second approach that we proposed addresses the problem of

shape matching as a graph cut problem. We could show that this formulation is equivalent to the shortest path problem in the DTW graph. But instead of computing multiple shortest paths, we have only to consider *one single* graph cut. Thus, the prior separation between initial point-match and the actual matching computation does not exist any more for the graph cut framework that we proposed. This is an important improvement in the area of shape matching. Nonetheless, this does not mean that shape matching can really be computed more efficiently than the cubic run-time based DTW based shape matching.

In order to improve the runtime of the graph cut problem, we proposed a graph cut method that exploits the planarity of the involved graph. The presented method is an extension of a recent work of Borradaile and Klein. While the original work used two quite sophisticated data structures in order to reduce the run-time for any planar graph, we were able to drop one of these data structures and obtained a more efficient method in the process. As a result, also the graph cut formulation computes the shape matching in sub-cubic runtime, namely in $O(N^2 \log(N))$.

## Shape Classification

We showed the result of two different shape classification methods, namely the method of shape clustering and the method of shape retrieval in the form of the Bull's Eye Test for the challenging MPEG7 shape database. We showed that the retrieval rate depends highly on the resolution of the shape points. Thus, it is very important to improve the efficiency of shape matching methods. Instead of just performing shape matching for 100 shape points, we could compute the retrieval rate for up to 500 shape points in justifiable time. With a standard PC, the two million comparison of the MPEG7 database could be computed within less than four days.

## 5.2 Future Work

### Shape Acquisition

In this work we showed how the quite simple shape metric for stochastic shapes, namely the region-based $L^2$ distance, can be efficiently used for shape acquisition. In general, this distance is not descriptive enough in order to emulate human notion of object recognition. An important extension of the presented shape prior driven shape acquisition would be the incorporation of more sophisticated shape dissimilarity measure into shape acquisition. Two of such measures were presented which were either induced by the problem of shape morphing or the problem of shape matching. The main challenge of such an approach lies in the fact, that robust shape acquisition in normally defined as a region based energy functional while robust shape distance functions are usually defined as an edge based energy functional. To combine these two approaches is very challenging and should be addressed in the future.

### Shape Morphing

One important drawback of the presented shape morphing lies in the fact that arc-length represented shapes were used. We believe that the good retrieval results that we obtained with shape matching lies in the flexibility of allowing different parameterization of the shapes. Note that a matching mapping $m : \mathbb{S}^1 \rightarrow \mathbb{S}^1$ between two shapes reparametrizes the second shape in order to find a good match with respect to the first shape. Therefore we think that allowing different parameterizations for one shape should improve the results we obtain with shape morphing.

### Shape Matching

We presented two different methods to compute a shape matching between two shapes represented as loops in a preselected feature space. Both methods provided for a very low worst-case run-time complexity. Thus, also shapes which are given at a quite high resolution could be matched in a justifiable time. Future work should be focused on exploiting the fact that shapes given at a fine discretization can be matched. One important task would be the improving of features for these finely represented shapes. Hence, it would make sense to apply a more sophisticated measure on involved histograms like the two presented shape contexts. Besides this very classical approach of shape matching, future work should also be focused on a more general form of shape matching, namely partial shape matching. In practice, we often have to deal with occlusion. Future work should not only focus on detecting this occlusion like we did in Chapter 2, but also detecting partial occlusion should be addressed in future work. A major challenge is to automatically divide a shape into its parts and to perform a shape matching of these parts. To provide a general framework that solves this problem automatically should be addressed in the future.

### Shape Clustering

While clustering with respect to distance driven methods work quite well for some shapes, it may fail for other shapes. This is especially then the case if the involved distance function does not handle all presented shapes correctly. As a result, outliers with respect to the involved distance function should be handled more appropriately. Thus not only distances but also stochastic tools like averaging shapes or the computation of a shape covariances should be addressed in the future.

# Bibliography


[1] B. C. Appleton. *Globally minimal contours and surfaces for image segmentation.* PhD thesis, University of Queensland, Australia, December 2004.

[2] M. Bakircioglu, M. Grenander, N. Khaneja, and M. I. Miller. Curve matching on brain surfaces using frenet distances. *Human Brain Mapping*, pages 329–333, 1998.

[3] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[4] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and objects recognition using shape contexts. *IEEE PAMI*, 24(4):509–522, 2002.

[6] M. Bergtholdt and C. Schnörr. Shape priors and online appearance learning for variational segmentation and object recognition in static scenes. In *Pattern Recognition (Proc. DAGM).* LNCS, 2005.

[7] F. L. Bookstein. *The Measurement of Biological Shape and Shape Change*, volume 24 of *Lect. Notes in Biomath.* Springer, New York, 1978.

[8] F. L. Bookstein. Principal warps: thin–plate splines and the decomposition of deformations. *IEEE PAMI*, 11:567–585, 1989.

[9] G. Borradaile. *Exploiting Planarity for Network Flow and Connectivity Problems*. PhD thesis, Brown University, May 2008.

[10] G. Borradaile and P. N. Klein. An $o(n \log n)$ algorithm for maximum $st$-flow in a directed planar graph. In *17th ACM-SIAM Symp. on Discrete Algorithms*, 2006.

[11] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *Intl. Conf. on Medical Image Computing and Comp. Ass. Intervention (MICCAI)*, volume 1935 of *LNCS*, pages 276–286. Springer, 2000.

[12] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. In A. K. Jain M. Figueiredo, J. Zerubia, editor, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 2134 of *LNCS*, pages 359–374. Springer, 2001.

[13] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE PAMI*, 26(9):1124–1137, 2004.

[14] P. Buergisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[15] J. Buhmann and H. Kuehnel. Complexity optimized data clustering by competitive neural networks. *Neural Computation*, 5:75–88, 1993.

[16] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces based object segmentation. *IEEE PAMI*, 19(4):394–398, 1997.

[17] A. Chambolle. Total variation minimization and a class of binary mrf models. In *Int. Conf. on Energy Minimization Methods for Computer Vision and Pattern Recognition*, number 3757 in LNCS, pages 136–152. Springer, 2005.

[18] T. Chan, S. Esedoḡlu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

[19] T. F. Chan and L. A. Vese. A level set algorithm for minimizing the Mumford–Shah functional in image processing. In *IEEE Workshop on Variational and Level Set Methods*, pages 161–168, Vancouver, CA, 2001.

[20] G. Charpiat, O. Faugeras, and R. Keriven. Approximations of shape metrics and application to shape warping and empirical shape statistics. *Journal of Foundations Of Computational Mathematics*, 5(1):1–58, 2005.

[21] S. Chern, W. Chen, and K. S. Lam. *Lectures on Differential Geometry*. World Scientific, 1999.

[22] I. Cohen, N. Ayache, and P. Sulger. Tracking points on deformable objects using curvature information. In *Europ. Conf. on Computer Vision*, pages 458–466, 1992.

[23] T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. Use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12(6):355–365, 1994.

[24] D. Cremers. Dynamical statistical shape priors for level set based tracking. *IEEE PAMI*, 28(8):1262–1273, August 2006.

[25] D. Cremers. Nonlinear dynamical shape priors for level set segmentation. *Journal of Scientific Computing*, 35(2-3):132–143, June 2008.

[26] D. Cremers, T. Kohlberger, and C. Schnörr. Shape statistics in kernel space for variational image segmentation. *Pattern Recognition*, 36(9):1929–1943, 2003.

[27] D. Cremers, F. R. Schmidt, and F. Barthel. Shape priors in variational image segmentation: Convexity, Lipschitz continuity and globally optimal solutions. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Anchorage, Alaska, June 2008.

[28] D. Cremers and S. Soatto. Variational space-time motion segmentation. In B. Triggs and A. Zisserman, editors, *IEEE Int. Conf. on Computer Vision*, volume 2, pages 886–892, Nice, Oct. 2003.

[29] A. Delong and Y. Boykov. A scalable graph-cut algorithm for n-d grids. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Anchorage, Alaska, June 2008.

[30] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[31] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[32] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis*. Wiley, Chichester, 1998.

[33] A. Duci and A. C. Mennucci. Banach-like metrics and metrics of compact sets, 2007.

[34] N. Duta, A. Sonka, and Jain. A. K. Learning shape models from examples using automatic shape clustering and procrustes analysis. In A. Kuba, M. Samal, and A. Todd-Pokropek, editors, *Proc. Inf. Proc. in Med. Imaging*, volume 1613 of *LNCS*, pages 370–375. Springer, 1999.

[35] D. Farin, M. Pfeffer, P. H. N. de With, and W. Effelsberg. Corridor Scissors: A semi-automatic segmentation tool employing minimum-cost circular paths. In *IEEE ICIP*, Singapore, October 2004.

[36] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[37] G. Galilei. *Discorsi e dimostrazioni matematiche, informo a due nuoue scienze attenti alla mecanica i movimenti locali.* appresso gli Elsevirii; Opere VIII. (2), 1638.

[38] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classication of silhouettes. *IEEE PAMI*, 21(12):1312–1328, 1999.

[39] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking and matching deformable contours. *IEEE PAMI*, 17(3):294–302, 1995.

[40] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE PAMI*, 6(6):721–741, 1984.

[41] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum *a posteriori* estimation for binary images. *J. Roy. Statist. Soc., Ser. B.*, 51(2):271–279, 1989.

[42] U. Grenander, Y. Chow, and D. M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes*. Springer, New York, 1991.

[43] P. Hansen and B. Jaumard. Lipschitz optimization. In P. Horst and P. Pardalos, editors, *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.

[44] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[45] M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.

[46] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[47] O. Juan and Y. Boykov. Capacity scaling for graph cuts in vision. In *IEEE Int. Conf. on Comp. Vis.*, 2007.

[48] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. of Computer Vision*, 1(4):321–331, 1988.

[49] D. G. Kendall. The diffusion of shape. *Advances in Applied Probability*, 9:428–430, 1977.

[50] E. Klassen and A. Srivastava. Geodesics between 3d closed curves using path-straightening. In *Europ. Conf. on Computer Vision*, 2006.

[51] E. Klassen, A. Srivastava, W. Mio, and S. H. Joshi. Analysis of planar shapes using geodesic paths on shape spaces. *IEEE PAMI*, 26(3):372–383, 2003.

[52] M. Klodt, T. Schoenemann, K. Kolev, M. Schikora, and D. Cremers. An experimental comparison of discrete and continuous shape optimization methods. In *European Conference on Computer Vision (ECCV)*, Marseille, France, October 2008.

[53] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE PAMI*, 29(12):2079–2088, 2007.

[54] J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964.

[55] J. Kruskal and M. Liberman. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[56] V. Kwatra, A. Schoedl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 22(3):277–286, July 2003.

[57] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE PAMI*, 22(10):1185–1190, 2000.

[58] M. Leventon, W. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 316–323, Hilton Head Island, SC, 2000.

[59] H. Li and R. Hartley. The 3d-3d registration problem revisited. In *IEEE Int. Conf. on Comp. Vis.*, Rio de Janeiro, 2007.

[60] H. Ling and D. W. Jacobs. Shape classification using the innerdistance. *IEEE PAMI*, 29(02):286–299, 2007.

[61] M. Maes. On a cyclic string-to-string correction problem. *Inf. Process. Lett.*, 35(2):73–78, 1990.

[62] M. Maes. Polygonal shape recognition using string-matching techniques. *Pattern Recognition*, 24(5):433–440, 1991.

[63] S. Manay, D. Cremers, B.-W. Hong, A. Yezzi, and S. Soatto. Integral invariants for shape matching. *IEEE PAMI*, 28(10):1602–1618, 2006.

[64] P. Maurel, R. Keriven, and O. Faugeras. Reconciling landmarks and level sets. In *Proc. International Conference on Pattern Recognition*, pages 69–72, 2006.

[65] R. McConnell, R. Kwok, J.C. Curlander, W. Kober, and S. S. Pang. $\psi - s$ correlation and dynamic time warping: two methods for tracking ice floes in sar images. *IEEE Trans. on Geosc. and Rem. Sens.*, 29:1004–1012, 1991.

[66] P. Michor and D. Mumford. Riemannian geometries on spaces of plane curves. *J. of the European Math. Society*, 2003.

[67] P. Michor and D. Mumford. An overview of the riemannian metrics on spaces of curves using the hamiltonian approach. *Applied and Computational Harmonic Analysis*, 23:74–113, 2007.

[68] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space, 1996.

[69] F. Mokhtarian and A. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE PAMI*, 14:789–805, 1992.

[70] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition, 1995.

[71] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577–685, 1989.

[72] A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM T. on Mathematical Software*, 27(1):27–57, 2001.

[73] A. Pitiot, H. Delingette, A. Toga, and P. Thompson. Learning Object Correspondences with the Observed Transport Shape Measure. In *Information Processing in Medical Imaging*, pages 25–37, July 2003.

[74] J. B. Rosen. The gradient projection method for nonlinear programming part I: linear constraints. *SIAM Journal on Applied Mathematics*, 8:181–217, 1960.

[75] J. B. Rosen. The gradient projection method for nonlinear programming part II: nonlinear constraints. *SIAM Journal on Applied Mathematics*, 9:514–532, 1961.

[76] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[77] M. Rousson and D. Cremers. Efficient kernel density estimation of shape and intensity priors for level set segmentation. In *Intl. Conf. on Medical Image Computing and Comp. Ass. Intervention (MICCAI)*, volume 1, pages 757–764, 2005.

[78] M. Rousson, N. Paragios, and R. Deriche. Implicit active shape models for 3d segmentation in MRI imaging. In *Intl. Conf. on Medical Image Computing and Comp. Ass. Intervention (MICCAI)*, volume 2217 of *LNCS*, pages 209–216. Springer, 2004.

[79] F. R. Schmidt, M. Clausen, and D. Cremers. Shape matching by variational computation of geodesics on a manifold. In *Pattern Recognition (Proc. DAGM)*, volume 4174 of *LNCS*, pages 142–151, Berlin, Germany, September 2006. Springer.

[80] F. R. Schmidt, D. Farin, and D. Cremers. Fast matching of planar shapes in sub-cubic runtime. In *IEEE Int. Conf. on Comp. Vis.*, Rio de Janeiro, Brazil, October 2007.

[81] F. R. Schmidt, E. Töppe, and D. Cremers. Efficient planar graph cuts with applications in computer vision. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Miami, Florida, June 2009.

[82] F. R. Schmidt, E. Töppe, D. Cremers, and Y. Boykov. Efficient shape matching via graph cuts. In *Int. Conf. on Energy Minimization Methods for Computer Vision and Pattern Recognition*, 2007.

[83] F. R. Schmidt, E. Töppe, D. Cremers, and Y. Boykov. Intrinsic mean for semimetrical shape retrieval via graph cuts. In *Pattern Recognition (Proc. DAGM)*, volume 4713 of *LNCS*, pages 446–455, Heidelberg, Germany, September 2007. Springer.

[84] B. Schölkopf. *Support Vector Learning*. Oldenbourg, München, 1997.

[85] T. Sebastian, P. Klein, and B. Kimia. On aligning curves. *IEEE PAMI*, 25(1):116–125, 2003.

[86] E. Sharon and D. Mumford. 2d-shape analysis using conformal mapping. *Int. J. of Computer Vision*, 70(1):55–75, 2006.

[87] D. Sharvit, J. Chan, H. Tek, , and B. Kimia. Symmetry-based indexing of image database, 1998.

[88] S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *IEEE Int. Conf. on Comp. Vis.*, pages 349–356, 2005.

[89] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. In *JCSS*, pages 362–391, 1981.

[90] H. Tagare. Shape-based nonrigid correspondence with application to heart motion analysis. *IEEE Trans Med Imaging*, 18(7):570–579, 1999.

[91] D. W. Thompson. *On Growth and Form.* Cambridge University Press, Cambridge, 1917.

[92] W. Thurston. *The Geometry and Topology of Three-Manifolds.* http://www.msri.org/publications/books/gt3m/, 2002.

[93] A. Trouve and L. Younes. Diffeomorphic matching problems in one dimension: Designing and minimizing matching functions. In *Europ. Conf. on Computer Vision*, pages 573–587, 2000.

[94] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, E. Grimson, and A. Willsky. Model–based curve evolution technique for image segmentation. In *Comp. Vision Patt. Recog.*, pages 463–468, Kauai, Hawaii, 2001.

[95] M. Unger, T. Pock, D. Cremers, and H. Bischof. Tvseg - interactive total variation based image segmentation. In *British Machine Vision Conference (BMVC)*, Leeds, UK, September 2008.

[96] G. Vogiatzis, P. Torr, and R. Cippola. Multi-view stereo via volumetric graph-cuts. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 391–399, 2005.

[97] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal ACM*, 21(1):168–173, 1974.

[98] K. Weihe. Maximum (s,t)-flows in planar networks in O($|V|$ log$|V|$) time. *J. Comput. Syst. Sci.*, 55(3):454–475, 1997.

[99] H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54:150–168, 1932.

[100] X. Yang, X. Bai, L. J. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. In *Europ. Conf. on Computer Vision*, pages 788–801, Marseille, France, October 2008.

[101] L. Younes. Optimal matching between shapes via elastic deformations. *Image and Vision Computing*, 17:381–389, 1999.

[102] L. Younes, P. Michor, J. Shah, and D Mumford. A metric on shape space with explicit geodesics. *Rend. Lincei Mat. Appl.*, 9:25–57, 2008.

[103] A. Yuille. Generalized deformable models, statistical physics, and matching problems. *Neural Comp.*, 2:1–24, 1990.

[104] L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.