Spline Shape Processing: Representation, Learning, and Modeling

Daniel Schmitter

Thèse N° 8174 (novembre 2017)

Thèse présentée à la faculté des sciences et techniques de l'ingénieur
pour l'obtention du grade de docteur ès sciences
et acceptée sur proposition du jury

École polytechnique fédérale de Lausanne—2017

Cover design by Annette Unser
Printing and binding by Repro-EPFL
Typeset with LaTeX

Abstract

In this thesis, we present a novel generic and unifying framework for data-adaptive shape modeling. Our work is motivated by the raising need for powerful geometric modeling kernels that are required for shape characterization in biomedical imaging. The ongoing development of faster and more precise hardware in biomedicine creates new demands regarding the characterization, processing, and analysis of shapes in medical and biological data such as MRI, CT or microscopy. We develop a novel mathematical framework to construct a geometric kernel that is capable to represent deformable shapes in the contexts of segmentation, visualization, and deformation, while also enabling user-interaction, fast optimization as well as the construction of shape dictionaries for shape encoding. Our approach relies on spline-based concepts for shape representation in computer graphics combined with spline-theoretical fundamentals from signal and image processing.

This thesis is organized in two main parts. In the first part, our main contribution is the development of a novel generic framework to encode shapes, learn dictionaries and construct shape projectors onto functional vector spaces defined by spline shapes. Our construction is formulated in the continuous domain and we propose solutions based on L_2-methods as well as sparsity-promoting ℓ_1-based methods. Our contribution is generic in the sense that it is applicable to any spline-based generator that forms a Riesz Basis. We apply our solution to classify shapes in medical ultrasound images, to learn dictionaries of brain structures and to construct robust shape priors for segmentation algorithms.

In the second part of this thesis, our contribution is a novel data-adaptive framework to construct deformable shapes with different kinds of topology that are used in a broad spectrum of applications such as the semi-automatic segmentation of biomedical structures, user-interactive shape modeling, shape deformation and morphing, shape reconstruction from samples as well as the visualization and animation of shapes. We construct novel families of spline-generators that facilitate intuitive user-interaction, shape modeling, and shape characterization, and we provide specific examples and validations of the use of our framework in practice.

The two parts of this thesis together build a unifying theory for the construction of a novel flexible geometric modeling kernel with a wide range of use and applications as we illustrate throughout this thesis with a vast amount of experiments.

Keywords: Shape characterization, shape modeling, splines, interpolation, shape deformation, tensor product surfaces, dictionary learning, sparsity, sparse coding, continuous domain, vector space, shape projectors, geometric kernel, computer-aided design.

Zusammenfassung

Diese Doktorarbeit befasst sich mit der Erarbeitung eines neuartigen mathematischen und geometrischen Kernels zur flexiblen sowie daten-spezifischen Modellierung, Charakterisierung, Analyse sowie Darstellung von Kurven und Flächen, die in biomedizinischen bildgebenden Verfahren generiert und benötigt werden. Die Motivation für die Erarbeitung dieser Doktorarbeit ist die Notwendigkeit neuer geometrischer Kernel, um die geometrischen Formen in den heutzutage generierten Bilddaten in der Biomedizin analysieren und charakterisieren zu können. Die diesbezügliche rasante Entwicklung neuer bildgebender Verfahren erfordert neue Methoden und Algorithmen, um die damit verbundenen daten-spezifischen Herausforderungen bewältigen zu können wie zum Beispiel die geometrische Formanalyse von Organen in grossen Datenmengen mit hoher Auflösung oder die halb-automatische und interaktive 3-dimensionale Formmodelierung zur Segmentierung von Zellen in mikroskopischen Bildern. Unser Ansatz zur Konstruktion eines neuen geometrischen Kernels basiert auf Konzepten der Spline-theorie, welche sowohl in der Computergraphik als auch in der Signal-und Bildverarbeitung angewandt wird.

Diese Doktorarbeit ist aufgeteilt in einen generischen Teil und einen daten-spezifischen Teil. Im ersten Teil erarbeiten wir generische Konzepte zur Konstruktion von Formlexika, Formkodierung und Formanalyse sowie mathematische Projektoren auf Vektorräume, die durch parametrische Kurven und Flächen in Hilberträumen definiert sind. Dieser Teil ist unabhängig von einer spezifischen Splinefunktion unter der Bedingung, dass die Splinefunktion eine Riesz-basis generiert. Wir validieren unser mathematisches Modell mittels Klassifizierung von anatomischen Strukturen in Ultraschall-Bildern, der Erstellung von Formlexika von Gehirnstrukturen, die durch MRI generiert worden sind sowie mittels Segmentierung von biologischen Zellstrukturen in der Mikroskopie.

Im zweiten Teil präsentieren wir neue mathematiche Konstruktionen zur Darstellung von deformierbaren Kurven und Flächen. Wir leiten neue Splinefunktionen her, die nach spezifischen topologischen Anforderungen konstruiert werden können. Wir testen und validieren unsere geometrischen Modelle in verschiedenen Anwendungen, wie zum Beispiel der halb-automatischen Segmentierung von 3D Gehirn- und Aortastrukturen, der computer-gestützten interaktiven Formmodelierung sowie der Deformierung von Flächen, der Flächenrekonstuktion durch Interpollieren oder der Visualisierung und Animierung von geometrischen Formen.

Zusammen bilden die beiden Teile dieser Doktorarbeit ein einheitliches mathematisches Konzept zur Konstruktion eines neuen geometrischen Modellierungskernels.

Stichworte : Formcharakterisierung, Formmodellierung, Splines, Interpollieren, Formdeformierung, Tensorprodukt Flächen, Erlernte Formlexika, Stetigkeit, Vektorraum, Mathematische Projektion, Geometrischer Kernel, Computer-gestützte Modellierung.

There is a vitality, a life force, an energy, a quickening
that is translated through you into action,
and because there is only one of you in all of time,
this expression is unique.
-Martha Graham

To my parents, my sister, and Jihye.

Acknowledgement

First, I would like to thank my thesis advisor, Prof. Michael Unser, for his support and supervision during the last four years. I greatly appreciated his enthusiasm and encouragement to pursue my own ideas as well as his critical thinking. Furthermore, I am deeply grateful to Michael for having given me the opportunity to work on the topic of splines, which over the last four years has become a true passion.

I also thank my thesis co-advisor, Dr. Daniel Sage, who has enthusiastically supported me already when I was still a bachelor student. He gave me the opportunity to work early on on research projects and collaborations and he introduced me to the beautiful research world of EPFL's Biomedical Imaging Group (BIG).

I express my sincere thanks to the president of my thesis committee, Prof. Dimitri Van De Ville, and the jury members, Prof. Mark Pauly, Prof. Chistophe Rabut, and Prof. Jean-Christophe Olivo-Marin.

I especially thank the wonderful and truly exceptional team Mirrakoi: Pablo, Zsuzsanna, and Carlos.

I also thank my great team members at BIG, Kyong Hwan, Harshit, Anais, Emrah, Adrien, and Denis. I further thank fellow past and present BIG members, Arash, Ning, Ricard, Laurene, Julien, Emmanuel, Ulugbek, Stamatis, Ha, Thanh-An, Mike, Masih, Pedram, Ferreol, Martin, Philippe, Virginie, Cedric, John Paul, Luc, Christophe, Shayan. I further thank Claudia, Nadia, Beatrice, and Manuelle for all the help and support with administrative aspects related to my thesis and Mirrakoi as well as EPFL's TTO and VPI for all their support.

I thank my parents and my sister for their unconditional support. I am deeply thankful to Jihye for having brought magic to my life.

Finally, I also want to thank all my friends. *27. October 2017*

Contents

Chapter 1

Introduction

Today's research in biomedical imaging is highly dependent on computer-generated representations of shapes. Curves and surfaces need to be studied and analyzed, which results in an increasing demand for digital geometric models. Hence, today's shape-analysis workflows depend more and more on software for shape representation and modeling.

Shape deformation, interaction, segmentation, morphology, quantification or registration require an accurate representation of an underlying shape structure. Each shape-modeling software has an integrated geometric *kernel*, which implements the underlying mathematics that are used to represent the geometry of a digital shape model. The geometric kernel determines how the user interacts with digital objects, how they are drawn or manipulated, and computed. Overall, the kernel defines the range of possible functionality that can be deployed for digital shape representation, and as a consequence, it also sets the limitations. The increasing demand for cutting-edge software tools directly translates into a rising need for powerful geometric modeling kernels.

In biomedical imaging, the ongoing development of modern imaging devices with high resolution and precision leads to new demands in terms of shape analysis [1, 2, 3, 4, 5]. Simultaneously, the availability of faster and more performant computers, tablets, and workstations allows for the implementation of more complex and computation-intensive models [6, 7, 8, 9]. Finally, it also enables to process large amounts of data at reasonable time scales [10, 11, 12, 13].

The common aspect in many of the shape analysis workflows in the field of biomedical imaging [14, 15] is the need for a specific, yet flexible, shape formulation. When adapted to a specific application, the challenge of obtaining a reliable shape description comes along with the requirement of accurate detection, fast optimization schemes, and user-interactivity. The underlying digital shape representation defines the limitations of implementational feasibility of application-dependent algorithms, such as high-throughput or single structure shape analysis, which also depend on the type of data used such as 2D or 3D.

A typical workflow for shape characterization and analysis in biomedical imaging consists in the following steps: First the data is fed into the pipeline. It can be 2D, 3D or higher dimensional and the structures of interest can have different topologies such as circular, spherical or cylindrical. The next step is the segmentation. It might involve manual or (semi-)automatic outlining and detection of shapes with or without user-interaction including the possible use of shape priors. The last step is the analysis and characterization of the shapes, where measurements, statistical quantities, shape encoding, and learning methods might be applied. These results can be re-used to include prior knowledge in the segmentation step. A schematic representation of such a workflow is given in Figure 1.1.

The execution of a complete shape-characterization pipeline is usually time-consuming and costly. The primary limitations of existing solutions are:

1. *Incompatibilities of sub-modules of the pipeline:* Format conversions between geometry-representations of the different software can be difficult, error-prone as well as compromise accuracy of the shape description. For example, there exist software that might segment 3D data conveniently, whereas another program might visualize it better. Similarly, a tool might be suitable to integrate shape priors in the segmentation process, but unsuitable to perform shape learning and encoding on a whole training set. Also some software might offer fast automatic segmentation but only limited functionality for subsequent user-interaction with a surface object.

2. *Specificity of the type of shape characterization:* Often the underlying research question that is studied requires a very specific functionality when characterizing a shape. For example, if vascular structures in medical images need to be automatically segmented, including the possibility to subsequently modify the result interactively, then a deformable model is needed. It must be

suitable to outline tubular structures, have few parameters to simplify the optimization process and finally be manually editable. It is likely that no implementation of a geometrical model exists that exactly solves this required shape-analysis task. However, there might exist a software which solves a similar problem but for a different topology, for instance, the segmentation of roundish cells in 3D microscopy. A typical development may then involve adapting an existing solution by customizing it and adding new functionality or by implementation of a specific plugin to an existing software platform.

3. *Requirement of profound expertise of the software:* Several software require programming skills or libraries that need to be compiled and installed prior to the proper functioning of the tools. Such circumstances restrict the applicability and interoperability of existing solutions. Not every research unit might have the in-house expertise to run such a software or it might be time-consuming to guarantee its proper functioning.

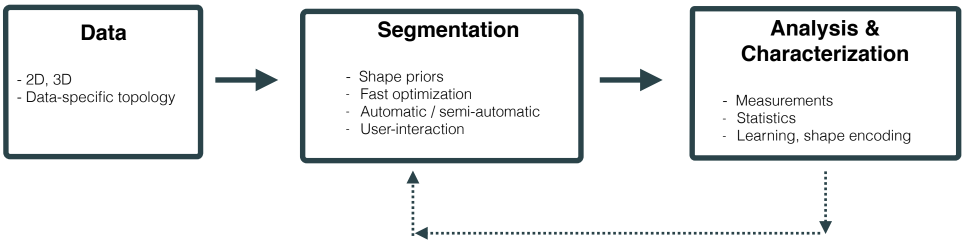


Figure 1.1: Workflow for shape analysis in biomedical imaging.

Existing digital geometric models are most commonly based on splines [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28], subdivision [29, 30, 31, 32, 33, 34, 35, 36, 37, 38], polygon meshes [39, 40, 41, 42, 43] or level sets [44, 45, 46, 47, 48, 49] and intrinsically, except for the latter, they can all be traced back to spline theory. While 'pure' spline-based methods involve continuously-defined models, polygon meshes are discrete. Subdivision is viewed as a hybrid; it is described by an iterative process where operators are applied to a discrete set of points leading to a continuous formulation in the limit. While polygon models are non-parametric and allow for more topological flexibility, continuous parametric models facilitate smooth, 'organic' or free-form modeling, as well as the exact evaluation of differential geometric quantities such as derivatives, normals, tangents and curvature or surface and volume computations.

1.1 Unifying Generic Model for Data-adaptive Shape Characterization

At the core of each implementation used to describe a shape is the geometric kernel. It is the fundament of any shape analysis pipeline and any kind of restriction or limitation at this stage will necessarily have implications in some of the three aspects listed above.

In this thesis, we attempt to address the general problem of characterizing shapes from a geometrical point of view while also focusing on an efficient software implementation. From the typical workflow described in Figure 1.1, we infer the necessary conditions and requirements that have to be directly imposed on a mathematical shape model in order to construct a unifying generic geometric framework for data-adaptive shape characterization. These requirements are categorized as

- Intuitive user-interaction
- Data-specific topology and its deformation
- Fast shape optimization in the context of deformation
- Shape priors and integrable shape knowledge.

The range of applications that a unifying geometric model needs to cover is categorized as

- Interactive shape modeling and manual segmentation
- (Semi-) automatic segmentation
- Shape characterization through measurements and statistical inference
- Dictionary learning and shape encoding.

The two categorizations above can be seen as input and output, respectively, of a data-adaptive generic shape formulation as depicted in Figure 1.2. The goal of this thesis is to construct a complete mathematical framework for data-adaptive generic shape modeling and its application, and in the broadest possible sense, to provide the mathematical basis for an efficient digital implementation of the shape model.

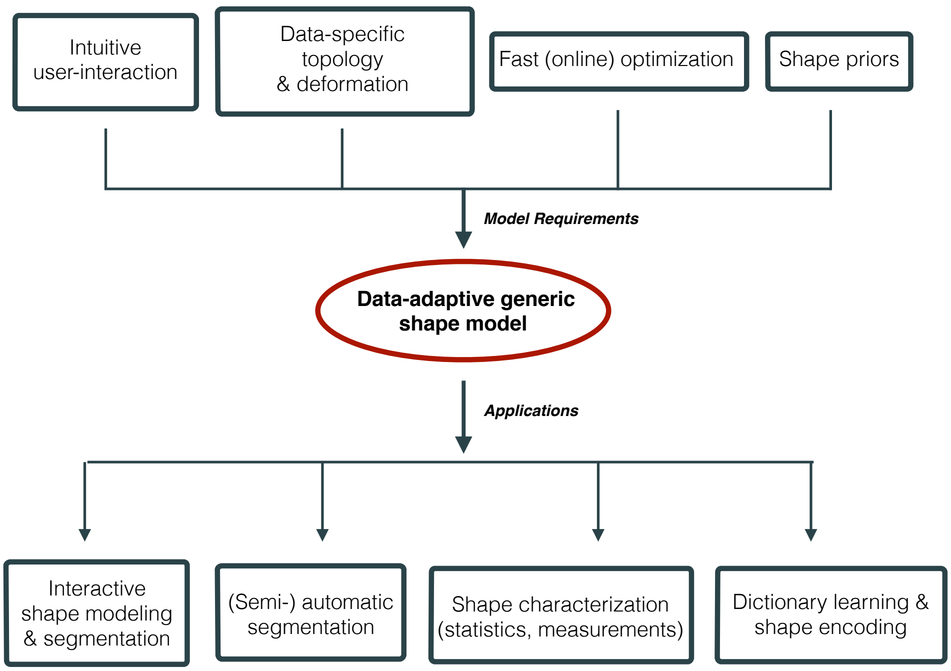

Intuitive user-interaction
Data-specific topology & deformation
Fast (online) optimization
Shape priors
Model Requirements
Data-adaptive generic shape model
Applications
Interactive shape modeling & segmentation
(Semi-) automatic segmentation
Shape characterization (statistics, measurements)
Dictionary learning & shape encoding

Figure 1.2: Unifying generic geometric model for data-adaptive shape characterization.

1.1.1 Spline-based Formulation in the Continuous Domain

We formulate our framework in the continuous domain, which allows us to *exactly* describe a shape in theory. Such a model is also suitable in a discrete pipeline since any continuously-defined shape can always be resampled with arbitrary precision. Furthermore, we choose a *parametric* shape description which simplifies the evaluation of geometric quantities such as surface- or volume-integrals and lends itself well to iterative optimization. Moreover, we describe shapes from a signal-processing perspective. It allows us to deploy the theory of signal and image processing which links discrete measurements to an underyling continuous-domain signal and gives us access to the vast theory related to Hilbert spaces and Fourier analysis [50]. In this regard, we formulate our framework in its most general way using spline-based basis functions [51, 52, 53, 54, 55] and thereby, exploit the best of two worlds: the spline-related theory for shape modeling [56, 57] combined with the spline-theoretic fundamentals used in signal and image processing [58, 59, 60, 61, 62, 63, 64, 65].

1.2 Contribution

This thesis is divided into two main parts; a generic and a data-adapive part.

I **Generic shape modeling**
We first propose a general construction of (functional) vector spaces for parametric shapes. We then deploy their Hilbert-space structure to derive shape projection operators which we use to build the generic aspect of our framework. It includes a novel characterization of shape priors, shape alignment, dictionary learning, and sparse shape encoding. The primary contributions of this part are as follows:

(a) *A new characterization and implementation of shape projectors and registration.* Given a generic reference shape, we define vector spaces that allow the reference to deform *only* according to a subclass of geometric transformations that belongs to the affine family. We derive a closed-form solution that is applicable to any particular type of transformation of the affine family [66]. We use our results to construct shape priors that can be used in segmentation problems and we also show how to register shapes using our shape projectors [67, 68]. Our framework covers the classical case described in procrustean analysis [69, 70, 71, 72, 73] with the difference that we are able to retrieve the sought-after similarity transformation in one step.

(b) *A novel framework for dictionary learning and continuous-domain sparse shape coding.* Given a training set of shapes, we construct a theory for shape encoding and dictionary learning in the continuous domain [74, 75]. We first present an unbiased method for data alignment and derive a functional shape principal-component analysis based on L_2-minimization. Then, we propose an ℓ_1-based framework for sparse dictionary learning of shapes which is robust w.r.t. imbalanced data as well as to outliers. Moreover, we present a formula for a spline-based $L_2 - \ell_2$ norm equality, which we use to provide an exact implementation of the continuous-domain solution at no additional cost compared to the discrete case.

The parametric description of shapes that we propose is independent of the specific choice of basis functions as long as the underlying spline-based generator function satisfies certain criteria. The only constraint is that the generator should be compactly supported and that its integer shifts should form a Riesz basis in order to guarantee a unique and stable representation of a shape.

II **Data-adaptive shape modeling**
In this part, we present and derive data-specific theoretical aspects of our framework and propose solutions for implementations. The contributions are

(a) *The construction of new interpolating spline-based generators for applied problems.* We construct novel families of interpolators and derive general results that guarantee stability and certain shape reproduction properties [76, 77, 78]. The construction follows a systematic approach taking into account practical considerations related to shape modeling such as smoothness, user-interaction, geometry or shape resolution. Then, we show how specific interpolators are constructed according to shape topology and geometric requirements of the curves or surfaces that want to be represented.

(b) *Novel formulation and implementation of deformable shapes in practice.* In this part, we present the power of our framework with a broad spectrum of applications. It includes automatic shape segmentation in biomedical imaging [79] including subsequent user-interaction [80], interactive shape modeling in computer graphics, shape design, surface reconstruction from point clouds as well as the animation and morphing of textured shapes [27, 28].

1.3 Organization of the Thesis

This thesis follows the organization depicted in Figure 1.3. In Chapter 2, we introduce fundamental aspects of spline-theory related to shape representation. We review the general formulation of parametric spline curves and surfaces whose coordinate functions can be interpreted as uniformly sampled one-dimensional signals. We formulate the corresponding Hilbert-space structure and state the fundamental properties that spline-based basis functions need to fulfill. Moreover, we provide formulas for an efficient computation and implementation of inner-products between spline shapes; a frequently used operation in continuous-domain spline-shape processing. In Chapter 3, we present the construction of projection operators onto functional vector spaces that are defined by spline shapes. We further show how shape priors are constructed. In Chapter 4, we present a complete theory for continuous-domain dictionary learning and shape encoding, which covers the L_2- and ℓ_1-based cases for spline curves. In Chapter 5, we present an extension of our shape-encoding framework to 3D surfaces. In Chapter 6, we construct families of interpolating spline-based generators that can be adapted to application-specific criteria. Finally, in Chapter 7, we present a panorama of applications of our framework including shape characterization in biomedical imaging, computer graphics, and shape modeling.

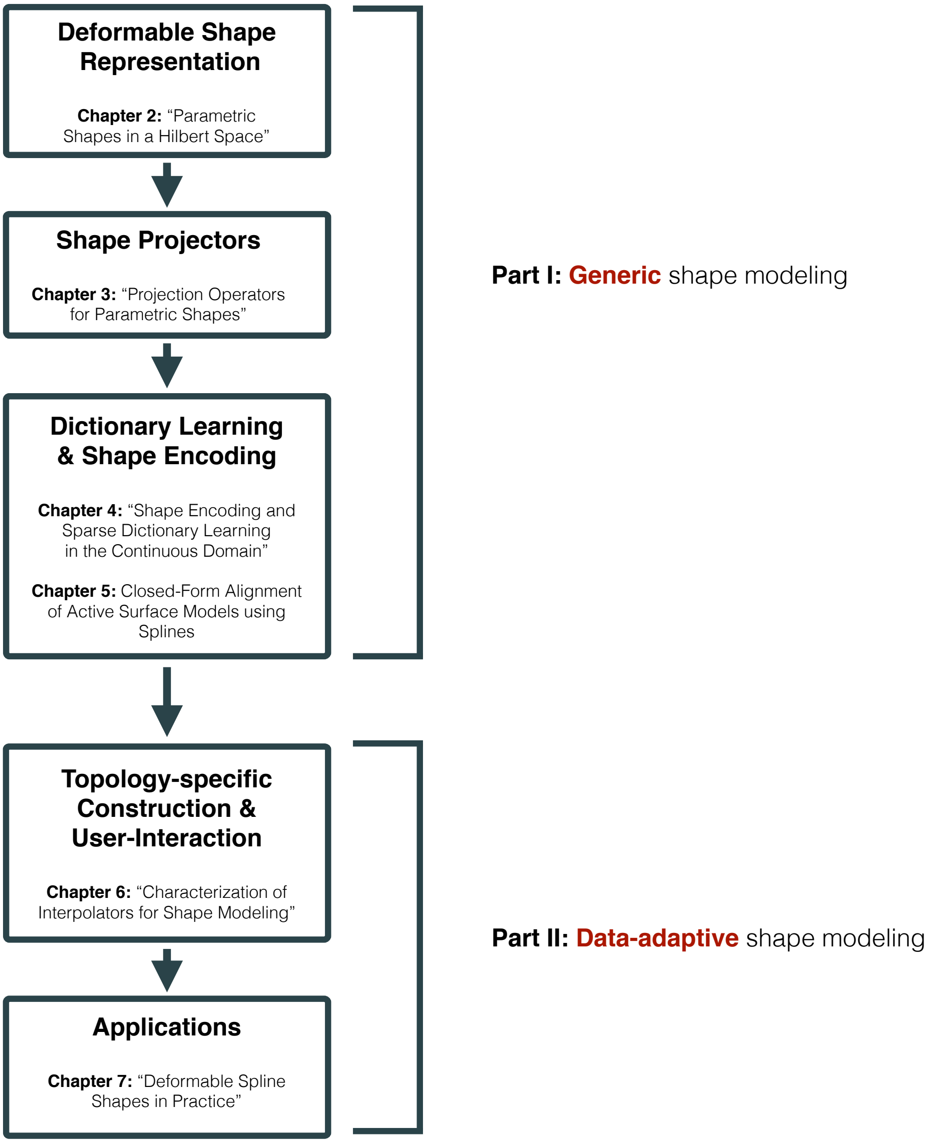

Deformable Shape Representation
Chapter 2: "Parametric Shapes in a Hilbert Space"
Shape Projectors
Chapter 3: "Projection Operators for Parametric Shapes"
Dictionary Learning & Shape Encoding
Chapter 4: "Shape Encoding and Sparse Dictionary Learning in the Continuous Domain"
Chapter 5: Closed-Form Alignment of Active Surface Models using Splines
Topology-specific Construction & User-Interaction
Chapter 6: "Characterization of Interpolators for Shape Modeling"
Applications
Chapter 7: "Deformable Spline Shapes in Practice"
Part I: Generic shape modeling
Part II: Data-adaptive shape modeling

Figure 1.3: Organization of the thesis. Part I consists in the fundamentals to construct a generic shape modeling framework. In Part II, we derive the data-adaptive aspects of the model and related applications.

Chapter 2

Parametric Spline Shapes in a Hilbert Space

We consider a shape as a parametric vector-valued function whose coordinates are continuous-domain signals sampled on the (regular) integer grid. The signal is then expressed as a weighted sum of shifted basis functions, which are versions of a single *generator* [81]. In this chapter, we review the general construction of parametric shapes using cardinal basis functions and show how related inner products are efficiently computed and implemented [82].

2.1 Parametric Curves and Surfaces

2.1.1 Notation

Throughout this thesis we largely adopt the notation from the signal processing community. We describe a vector $\boldsymbol{v}$ using **bold** font. The imaginary complex unit j satisfies $\mathrm{j}^2 = -1$, while the Fourier transform of a function f is denoted by $\widehat{f}$. The convolution between two functions f and g is specified by $f * g$.

2.1.2 Spline Curves

We consider 2D and 3D parametric curves $\boldsymbol{r}(t) = (x(t), y(t), z(t))$ that are described by the coordinate functions $x(t)$, $y(t)$, and $z(t)$, with $t \in \mathbb{R}$. If only a 2D (*i.e.*, planar) curve representation is required, then the z-coordinate can be set to a constant and hence, is ignored. The coordinate functions are parameterized by a suitable linear combination of integer-shifted basis functions that are derived from a spline-based *generator* φ [83]. Using this model, a parametric curve is described by

$$\boldsymbol{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \sum_{k \in \mathbb{Z}} \boldsymbol{c}[k] \varphi(t - k), \tag{2.1}$$

where the coefficients $\boldsymbol{c}[k] = (c_x[k], c_y[k], c_z[k])$ with $k \in \mathbb{Z}$ are called *control points* [20, 76]. In applications, φ is usually chosen such that it is *compactly* (*i.e.*, finitely) supported. This implies that the curve can be modified locally by changing the position of a single control point. The shapes that $\boldsymbol{r}$ can adopt (*e.g.*, polynomial, circular, elliptic) depend on the properties of the generator [84, 85]. In practice, we consider the sum in (2.1) to be finite and the support of φ to be integer [86]. It follows that we express (2.1) for $t \in [0, M]$, where M is a positive integer. In certain contexts it is convenient to normalize the domain of $\boldsymbol{r}$ such that t lies in the unit interval. Then, the curve (2.1) is expressed as

$$\boldsymbol{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \sum_{k \in \mathbb{Z}} \boldsymbol{c}[k] \varphi(Mt - k), \tag{2.2}$$

where $t \in [0, 1]$. Throughout this thesis, we might also use the more general notation

$$\boldsymbol{r}(t) = \sum_{k \in \mathbb{Z}} \boldsymbol{c}[k] \varphi_k(t) \tag{2.3}$$

with $\varphi_k(t) = \varphi(t - k)$ in the cardinal setting.

2.1.3 Tensor-Product Spline Surfaces

The curve model (2.1) is extended to 3D in order to construct surfaces that can be represented by a separable parameterization [77]. In this case, a surface $\boldsymbol{\sigma}$ is parameterized by $(u,v) \in \mathbb{R}^2$ as

$$\begin{aligned}
\boldsymbol{\sigma}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} &= \begin{pmatrix} x_1(u)\cdot x_2(v) \\ y_1(u)\cdot y_2(v) \\ z_1(u)\cdot z_2(v) \end{pmatrix} \\
&= \sum_{k\in\mathbb{Z}} \boldsymbol{c}_1[k]\varphi_1(u-k) \times \sum_{l\in\mathbb{Z}} \boldsymbol{c}_2[l]\varphi_2(v-l) \qquad (2.4) \\
&= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \underbrace{\boldsymbol{c}_1[k]\times\boldsymbol{c}_2[l]}_{\boldsymbol{c}[k,l]} \varphi_1(u-k)\varphi_2(v-l),
\end{aligned}$$

where "$\times$" denotes the element-wise multiplication of two vectors and $\boldsymbol{c}[k,l] = (c_x[k,l], c_y[k,l], c_z[k,l])$ with $(k,l) \in \mathbb{Z}^2$. The expression developed in (2.4) can then be used to generally express surfaces with non-separable parameterizations as

$$\boldsymbol{\sigma}(u,v) = \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \boldsymbol{c}[k,l]\varphi_1(u-k)\varphi_2(v-l). \qquad (2.5)$$

2.2 Properties of the Generator φ

It is crucial that the model (2.1) be independent from the location and orientation of the curve $\boldsymbol{r}$ [87]. Therefore, (2.1) needs to be *affine invariant*, which implies that the generator φ needs to be capable of reproducing constants as specified by Proposition 1.

Proposition 1. *For any* $\mathbf{A} \in \mathbb{R}^{3\times 3}, \boldsymbol{b} \in \mathbb{R}^3$ *and a curve* $\boldsymbol{r}$ *as defined in* (2.3),

$$1 \in \operatorname{span}\{\varphi_k\} \Leftrightarrow \mathbf{A}\,\mathbf{r}(t) + \boldsymbol{b} \in \operatorname{span}\{\varphi_k\}$$

holds.

The proof follows immediately by noticing that for any constant $b \in \mathbb{R}$ the relation $1 \in \operatorname{span}\{\varphi_k\} \Leftrightarrow b \in \operatorname{span}\{\varphi_k\}$ holds. Hence, any translation vector $\boldsymbol{b} \in \mathbb{R}^3$ can be represented by (2.3) while the linearity of the model also ensures that $\mathbf{A}\boldsymbol{r}(t) \in \operatorname{span}\{\varphi_k\}$.

2.2.1 Affine Invariance

Definition 1. For $\mathbf{A} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{b} \in \mathbb{R}^3$, the curve representation (2.3) is affine invariant if

$$\mathbf{A}\,\mathbf{r}(t) + \boldsymbol{b} = \sum_{k\in\mathbb{Z}} \left(\mathbf{A}\,\mathbf{c}[k] + \boldsymbol{b}\right)\, \varphi_k(t).$$

Similarly, the surface representation (2.4) is affine invariant if

$$\mathbf{A}\boldsymbol{\sigma}(u,v) + \boldsymbol{b} = \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} (\mathbf{A}\mathbf{c}[k,l] + \boldsymbol{b})\varphi_k(u)\varphi_l(v). \tag{2.6}$$

It is easy to show that affine invariance is guaranteed if and only if φ satisfies the *partition of unity* condition $\sum_{k\in\mathbb{Z}} \varphi_k(t) = 1$ for all $t \in \mathbb{R}$ [88]. Note that in the literature affine invariance is sometimes referred to as *affine covariance*.

2.2.2 Riesz Basis

We consider the space

$$V(\varphi) = \left\{ \sum_{k\in\mathbb{Z}} c[k]\varphi_k,\ \ c \in \ell_2(\mathbb{Z}) \right\} \tag{2.7}$$

of functions that is generated by the shifts of φ. The Riesz basis property ensures that the representation of a function in $V(\varphi)$ is stable and unique [81].

Definition 2. The set $\{\varphi_k\}_{k\in\mathbb{Z}}$ forms a Riesz basis if

$$A\|c\|_{\ell_2(\mathbb{Z})} \le \left\| \sum_{n\in\mathbb{Z}} c[n]\varphi_n \right\|_{L_2(\mathbb{R})} \le B\|c\|_{\ell_2(\mathbb{Z})} \tag{2.8}$$

for some constants $A, B > 0$ and any sequence $c = (c[k])_{k\in\mathbb{Z}} \in \ell_2(\mathbb{Z})$.

When $\varphi_k = \varphi(\cdot - k)$, (2.8) is equivalent to the Fourier-domain condition

$$A^2 \le \sum_{k\in\mathbb{Z}} |\widehat{\varphi}(\omega - 2k\pi)|^2 \le B^2 \tag{2.9}$$

for any $\omega \in \mathbb{R}$ [81].

2.3 The Hilbert Space $\mathcal{H}$ Containing All Parametric Shapes

2.3.1 Curves

For parametric curves $\boldsymbol{r}(t) = (x(t), y(t), z(t))$, we denote by $\mathcal{H} : L_2(\mathbb{R}, \mathbb{R}^3)$ the Hilbert space associated with the L_2-inner product $\langle \boldsymbol{r}_k, \boldsymbol{r}_l \rangle := \int_{\mathbb{R}} \boldsymbol{r}_k^{\mathsf{T}}(t)\boldsymbol{r}_l(t)\mathrm{d}t$ that contains all parametric curves. The corresponding norm is defined as $\|\boldsymbol{r}\|_{L_2} := \sqrt{\langle \boldsymbol{r}, \boldsymbol{r} \rangle}$. If the domain of $\boldsymbol{r}$ is normalized we denote $\mathcal{H} : L_2([0,1], \mathbb{R}^3)$.

2.3.2 Surfaces

For parametric surfaces $\boldsymbol{\sigma}(u, v) = (x(u, v), y(u, v), z(u, v))$, we denote by $\mathcal{H} : L_2(\mathbb{R}^2, \mathbb{R}^3)$ the Hilbert space associated with the L_2-inner product $\langle \boldsymbol{\sigma}_k, \boldsymbol{\sigma}_l \rangle := \int_{\mathbb{R}^2} \boldsymbol{\sigma}_k^{\mathsf{T}}(u, v)\boldsymbol{\sigma}_l(u, v)\mathrm{d}u\mathrm{d}v$ that contains all parametric surfaces. The corresponding norm is defined as $\|\boldsymbol{\sigma}\|_{L_2} := \sqrt{\langle \boldsymbol{\sigma}, \boldsymbol{\sigma} \rangle}$. If the domain of $\boldsymbol{r}$ is normalized we denote $\mathcal{H} : L_2([0,1] \times [0,1], \mathbb{R}^3)$ or $\mathcal{H} : L_2([0,1]^2, \mathbb{R}^3)$.

2.3.3 Inner Product of Spline-Based Shapes

We use a powerful expression to compute the L_2-inner product $\langle \boldsymbol{r}_1, \boldsymbol{r}_2 \rangle$ between spline-based shapes. We first compute it for the 1D case and then generalize it to higher dimensions.

1D Inner Product

We consider spline-based (coordinate) functions of the form $x(t) = \sum_{k=0}^{K} c_x[k]\varphi_k(t)$. For $t \in [0, 1]$, the L_2-inner product is then expressed as

$$\begin{aligned} \langle x_1, x_2 \rangle &= \int_0^1 x_1(t)x_2(t)\mathrm{d}t \\ &= \sum_{k=0}^{K}\sum_{l=0}^{K} c_{1x}[k]c_{2x}[l] \int_0^1 \varphi_k(t)\varphi_l(t)\mathrm{d}t. \end{aligned} \tag{2.10}$$

We collect all the coefficients of the function x_i in the vector of length K, $\mathbf{c}_{ix} = (c_{ix}[0], \ldots, c_{ix}[K])$ with $i = 1$ or 2. We then define

$$[\mathbf{\Phi}]_{k,l} := \int_0^1 \varphi_k(t)\varphi_l(t)\mathrm{d}t. \tag{2.11}$$

Next, (2.10) is expressed as $\langle x_1, x_2\rangle = \mathbf{c}_{1x}^{\mathsf{T}}\mathbf{\Phi}\mathbf{c}_{2x}$, where $\mathbf{\Phi}$ is the $(K \times K)$ *correlation matrix* of φ_k. For an implementation (2.11) can be crucial. The entries of the matrix $\mathbf{\Phi}$ can be pre-computed. Hence, the evaluation of the integrals (2.11) associated with the inner product (2.10) boils down to a matrix-vector multiplication, which reduces the computational time considerably.

2D and 3D Inner Product for Curves

Similarly, we simplify the 2D inner product, first, for planar curves, *i.e.*, for $\boldsymbol{r}(t) = (x(t), y(t))$. By expressing $\mathbf{c}_{iy}$ the same way as we have expressed $\mathbf{c}_{ix}$ above, for a 2D curve $\boldsymbol{r}_i(t) = (r_{ix}(t), r_{iy}(t))$ we define

$$\mathbf{c}_i = (\mathbf{c}_{ix}, \mathbf{c}_{iy}), \tag{2.12}$$

which is a vector of length $2N$. The corresponding inner product is now expressed as

$$\langle \boldsymbol{r}_1, \boldsymbol{r}_2\rangle = \mathbf{c}_1^{\mathsf{T}}\mathbf{\Psi}\mathbf{c}_2 = \langle \mathbf{c}_1, \mathbf{c}_2\rangle_{\mathbf{\Psi}}, \tag{2.13}$$

where

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{\Phi} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi} \end{bmatrix} \tag{2.14}$$

and $\mathbf{0}$ is a zero matrix with the same dimensions as $\mathbf{\Phi}$ defined by (2.11). Observe that we use regular bold font to describe the $\mathbf{c}$ in (2.13) as opposed to italic bold font to describe the different object $\boldsymbol{c}$ in (2.1). We show in [82] how (2.11) is computed for the case where the curves $\boldsymbol{r}$ are periodic [89]. The extension of (2.13) to 3D (*i.e.*, non-planar) curves is straightforward.

2.3.4 Inner Product of Spline Surfaces

We consider parametric surfaces of the form (2.4) with K control points. Then, the inner product between two surfaces $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ is expressed as

$$\langle \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \rangle = \sum_{\{k,l,p,q\}=0}^{K} \boldsymbol{c}_1^{\mathsf{T}}[k,l]\boldsymbol{c}_2[p,q] \underbrace{\int \varphi_1(u-k)\varphi_2(u-p)\mathrm{d}u}_{\alpha_{k,p}^{(1)}} \underbrace{\int \varphi_1(v-l)\varphi_2(v-q)\mathrm{d}v}_{\alpha_{l,q}^{(2)}} . \tag{2.15}$$

Next, we define

$$\boldsymbol{\gamma}^{(l)} = \begin{pmatrix} c_x[0,l] \\ c_x[1,l] \\ \vdots \\ c_x[K,l] \\ c_y[0,l] \\ \vdots \\ c_y[K,l] \\ c_z[0,l] \\ \vdots \\ c_z[K,l] \end{pmatrix} \tag{2.16}$$

and

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\gamma}^{(0)} \\ \vdots \\ \boldsymbol{\gamma}^{K} \end{pmatrix} . \tag{2.17}$$

We define the autocorrelation matrix

$$\mathbf{A}^{(1)} = \begin{pmatrix} \alpha_{0,0}^{(1)} & \dots & \alpha_{0,K}^{(1)} \\ \alpha_{1,0}^{(1)} & \dots & \alpha_{1,K}^{(1)} \\ \vdots & \dots & \vdots \\ \alpha_{K,0}^{(1)} & \dots & \alpha_{K,K}^{(1)} \end{pmatrix} \tag{2.18}$$

and

$$\mathbf{B}^{(1)} = \begin{pmatrix} \mathbf{A}^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}^{(1)} \end{pmatrix} \tag{2.19}$$

and

$$\boldsymbol{\beta} = \begin{pmatrix} \alpha_{0,0}^{(2)}\mathbf{B}^{(1)} & \dots & \alpha_{0,K}^{(2)}\mathbf{B}^{(1)} \\ \vdots & \dots & \vdots \\ \alpha_{K,0}^{(2)}\mathbf{B}^{(1)} & \dots & \alpha_{K,K}^{(2)}\mathbf{B}^{(1)} \end{pmatrix}. \tag{2.20}$$

Now the inner product (2.15) is expressed as

$$\langle \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \rangle = \boldsymbol{\Gamma}_1^{\mathsf{T}} \boldsymbol{\beta} \boldsymbol{\Gamma}_2. \tag{2.21}$$

Note that in (2.19), $\mathbf{0}$ corresponds to a null-matrix of size $(K \times K)$.

2.4 Summary

In this chapter, we have presented the fundamental properties and aspects that we use in this thesis to represent shapes. The presented concepts as well as the formulas to compute inner products are generic and independent from the specific generator φ that is used. Depending on the particular application and generator that is used to construct shapes in practice, the presented formulas might be even further simplified by exploiting spline-related properties of the basis functions.

Chapter 3

Projection Operators for Parametric Shapes

Overview

In this chapter [1], we present a generic method to construct orthogonal projection operators for landmark-based parametric spline shapes. We construct vector spaces by a given transformation that belongs to the affine family (*e.g.*, affine, similarity, uniform scaling) and that is applied to a reference curve. These vector spaces contain all parametric curves up to the chosen transformation. We define the vector spaces implicitly through an orthogonal projection operator and present a theorem that characterizes the projector for landmark-based spline curves which are popular for the user-interactive analysis of biomedical images. Finally, we show how shape priors are constructed with the spline projector and provide examples of applications for the segmentation of microscopy images in biology.

3.1 Introduction

We propose a generic solution to solve the simple but widely studied problem

[1]The chapter is based on our publications [66, 67, 68]

$$\min_{\mathbf{A},\boldsymbol{b}} \|\mathbf{A}\boldsymbol{r}^{\text{ref}} - \boldsymbol{b} - \boldsymbol{r}\|_{L_2}^2,$$

where the (2×2) matrix $\mathbf{A}$ and $\boldsymbol{b} \in \mathbb{R}^2$ define some linear transformation of the known 2D parametric reference curve $\boldsymbol{r}^{\text{ref}}$ and $\boldsymbol{r}$ is a *query* shape. This problem arises in image segmentation settings [90, 45, 72, 91] when prior knowledge is integrated [46, 92, 93] such that the solution should be close to a given shape up to a linear transformation [71, 73]. For specific transformations, where a particular structure is enforced on $\mathbf{A}$ and $\boldsymbol{b}$ (*e.g.*, similarity, affine), several algorithms have been proposed [94, 95, 68, 67]. In this paper, we provide a generic solution by defining a finite-dimensional vector space that contains all possible shapes obtained by a given linear transformation of a reference shape. The presented framework is formulated in the continuous domain and is valid for *any* kind of linear transformation. The idea is to generically characterize a vector space as a subspace that contains all shapes that are related to a reference shape by a specific transformation. The vector space itself is implicitly characterized by its orthogonal projector. This allows us to compute the “best match” among curves defined by a subspace w.r.t. an arbitrary shape. Intuitively speaking, projecting a query shape onto the subspace defined by the reference shape amounts to “choosing” the reference up to a specific transformation that is closest to the input shape. We consider 2D parametric curves as reference shapes and propose solutions for continuously defined spline-based curves which are popular for user-interactive applications.

3.2 Vector Spaces Defining Affine Shape Spaces

3.2.1 Vector Spaces as Subspaces of $\mathcal{H}$

We define a subspace as the space that contains all allowable geometric transformations of a *reference* curve $\boldsymbol{r}^{\text{ref}}$. Such a subspace can be defined as a finite-dimensional vector space $\mathcal{S}^{\text{ref}}$ of dimension I, whose basis consists of elements $\{\boldsymbol{e}_i^{\text{ref}}\}_{i=1,\dots,I}$, which themselves are curves that depend on $\boldsymbol{r}^{\text{ref}}$. Hence, every element (*i.e.*, curve) living in $\mathcal{S}^{\text{ref}}$ can be expressed as a linear combination of the basis elements. Thus,

$$S^{\text{ref}} = \left\{ \sum_{i=1}^{I} u_i \boldsymbol{e}_i^{\text{ref}}(\cdot) : u_i \in \mathbb{R} \right\} \tag{3.1}$$

is a subspace of the Hilbert space $\mathcal{H}$.

Example - Affine Vector Space

An affine transformation (without translation) of a 2D curve can be expressed as $\mathbf{A}\boldsymbol{r}$, where $\mathbf{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ is a (2×2) matrix with elements $a_i \in \mathbb{R}$. By explicitly evaluating the matrix-vector product, we obtain

$$\mathbf{A}\boldsymbol{r}(t) = a_1 \begin{pmatrix} r_x(t) \\ 0 \end{pmatrix} + a_2 \begin{pmatrix} r_y(t) \\ 0 \end{pmatrix} + a_3 \begin{pmatrix} 0 \\ r_x(t) \end{pmatrix} + a_4 \begin{pmatrix} 0 \\ r_y(t) \end{pmatrix} .$$

Therefore, the affine shape space associated to the 2D reference curve $\boldsymbol{r}^{\text{ref}}$ is a four-dimensional vector space (*i.e.*, $I = 4$) whose basis is given by

$$\{\boldsymbol{e}_i^{\text{ref}}\}_{i \in [1,\ldots,4]} = \left\{ \begin{pmatrix} r_x^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_y^{\text{ref}} \end{pmatrix} \right\},$$

where we have omitted the continuous parameter t to simplify the notation. Note that the choice of the basis is not unique. However, different bases w.r.t. to a given transformation describe the same space.

3.2.2 Construction of Vector Spaces

The vector spaces that appear to be the most useful for applications are summarized in Table 7.2, where we present examples of bases $\{\boldsymbol{e}_i\}_{i=1,\ldots,I}$ that can be used to construct a vector space $\mathcal{S}^{\text{ref}}$ for specific 2D transformations. Taking a reference curve $\boldsymbol{r}^{\text{ref}} = (r_x^{\text{ref}}, r_y^{\text{ref}})$ and choosing a transformation given in Table 3.1, the corresponding vector space is given by the indicated basis.

Table 3.1: Bases for 2D vector spaces.

Transformation	Degrees of Freedom	Basis $\{\boldsymbol{e}_i^{\text{ref}}\}$ w.r.t. $\boldsymbol{r}^{\text{ref}}$
uniform scaling	1	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}\right\}$
non-uniform scaling	2	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ r_y^{\text{ref}} \end{pmatrix}\right\}$
translation	2	$\left\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$
uniform scaling + rotation	2	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} -r_y^{\text{ref}} \\ r_x^{\text{ref}} \end{pmatrix}\right\}$
similarity	4	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} -r_y^{\text{ref}} \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$
affine	4	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_y^{\text{ref}} \end{pmatrix}\right\}$
affine + translation	6	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$
horizontal shear + uniform scaling	2	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ 0 \end{pmatrix}\right\}$
horizontal shear + uniform scaling + translation	4	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$
vertical shear + uniform scaling	2	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\text{ref}} \end{pmatrix}\right\}$
vertical shear + uniform scaling + translation	4	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$
reflection + uniform scaling + rotation	2	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ -r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ r_x^{\text{ref}} \end{pmatrix}\right\}$
reflection + similarity	4	$\left\{\begin{pmatrix} r_x^{\text{ref}} \\ -r_y^{\text{ref}} \end{pmatrix}, \begin{pmatrix} r_y^{\text{ref}} \\ r_x^{\text{ref}} \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right\}$

3.2.3 Orthogonal Projectors

We now consider a projection operator $\mathcal{P}^{\text{ref}} : \mathcal{H} \to \mathcal{S}^{\text{ref}}$ that projects an arbitrary curve in $\mathcal{H}$ onto the vector space $\mathcal{S}^{\text{ref}}$ such that $\boldsymbol{r} \mapsto \mathcal{P}^{\text{ref}}\boldsymbol{r}$. This implies that a vector space can either be explicitly defined by $\mathcal{S}^{\text{ref}}$ or implicitly by $\mathcal{P}^{\text{ref}}$. A projection operator that projects a curve $\boldsymbol{r}$ from $\mathcal{H}$ onto a vector space $\mathcal{S}$ with basis $\{\boldsymbol{e}_i\}_{i=1,\ldots,I}$ and dimension I is expressed in its most general way as $\mathcal{P}\boldsymbol{r}(t) = \sum\limits_{i=1}^{I} \boldsymbol{e}_i(t)\langle \tilde{\boldsymbol{e}}_i, \boldsymbol{r}\rangle$, where $\{\tilde{\boldsymbol{e}}_i\}_{i=1,\ldots,I} \in \mathcal{S}$ is the unique dual basis with respect to $\{\boldsymbol{e}_i\}_{i=1,\ldots,I}$ such that $\langle \boldsymbol{e}_i, \tilde{\boldsymbol{e}}_j\rangle = \delta_{i-j}$ with δ_{i-j} being the Kronecker delta.

Orthogonal projectors are of special interest to us because they minimize the distance between the query curve $\boldsymbol{r} \in \mathcal{H}$ and its projection $\mathcal{P}\boldsymbol{r}$ onto $\mathcal{S}$ w.r.t. the norm induced by the L_2-inner product (see Figure 3.1). Proposition 2 provides a mean to directly compute the orthogonal projector $\mathcal{P}$ given a basis $\{\boldsymbol{e}_i\}_{i=1,\ldots,I}$ of the vector space $\mathcal{S}$.

Proposition 2. *The orthogonal projector $\mathcal{P}^{\text{ref}} : \mathcal{H} \to \mathcal{S}^{\text{ref}}$ that minimizes the distance between the curve $\boldsymbol{r} \in \mathcal{H}$ and the I-dimensional vector space $\mathcal{S}^{\text{ref}}$ is specified by*

$$\mathcal{P}^{\text{ref}}\boldsymbol{r}(t) = \langle \boldsymbol{K}_{\mathcal{P}^{\text{ref}}}(t,\cdot), \boldsymbol{r}\rangle,$$

where $\boldsymbol{K}_{\mathcal{P}^{\text{ref}}}(t,s) = \sum_{i=1}^{I} \boldsymbol{e}_i^{\text{ref}}(t) \otimes \tilde{\boldsymbol{e}}_i^{\text{ref}}(s)$ is the kernel of the operator $\mathcal{P}^{\text{ref}}$ and $\{\tilde{\boldsymbol{e}_i}^{\text{ref}}\}_{i=1,\ldots,I}$ the dual basis of $\{\boldsymbol{e}_i^{\text{ref}}\}_{i=1,\ldots,I}$. It is given by

$$\tilde{\boldsymbol{e}_i}^{\text{ref}} = {\mathbf{G}_{i,1}^{\text{ref}}}^{-1}\boldsymbol{e}_1^{\text{ref}} + \cdots + {\mathbf{G}_{i,I}^{\text{ref}}}^{-1}\boldsymbol{e}_I^{\text{ref}},$$

where $\mathbf{G}^{\text{ref}}$ is the Gram matrix of the basis $\{\mathbf{e}_i^{\text{ref}}\}_{i=1,\ldots,I}$. Here, $\otimes$ denotes the tensor product between two vectors and is defined as $\mathbf{e}_i(t) \otimes \mathbf{e}_j(s) = \mathbf{e}_i(t)\mathbf{e}_i^{\mathrm{T}}(s)$.

The derivation of Proposition 2 is provided in Appendix 3.7.1. We say that $\mathcal{P}^{\text{ref}}$ projects $\boldsymbol{r} \in \mathcal{H}$ onto the I-dimensional *invariant* subspace $\mathcal{S}^{\text{ref}}$. In particular, we have that for any $\boldsymbol{r}^{\mathcal{S}^{\text{ref}}} \in \mathcal{S}^{\text{ref}} \Rightarrow \boldsymbol{r}^{\mathcal{S}^{\text{ref}}} = \mathcal{P}^{\text{ref}}\boldsymbol{r}^{\mathcal{S}^{\text{ref}}}$.

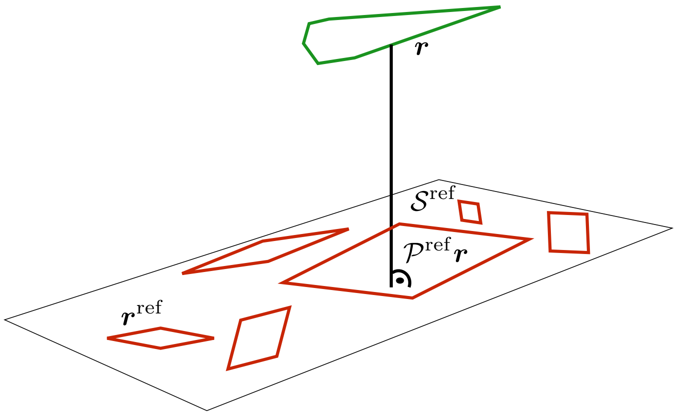

$\boldsymbol{r}$
$\mathcal{S}^{\mathrm{ref}}$
$\mathcal{P}^{\mathrm{ref}}\boldsymbol{r}$
$\boldsymbol{r}^{\mathrm{ref}}$

Figure 3.1: Illustration of an orthogonal projection onto a vector space. The plane denoted by $\mathcal{S}^{\mathrm{ref}}$ represents the subspace defined by the reference shape $\boldsymbol{r}^{\mathrm{ref}}$. $\mathcal{S}^{\mathrm{ref}}$ represents a subspace that contains all curves $\boldsymbol{r}^{\mathrm{ref}}$ up to a class of transformations (*e.g.*, rotations, scaling, and translations of $\boldsymbol{r}^{\mathrm{ref}}$). Projecting a query curve $\boldsymbol{r}$ (green curve) orthogonally onto $\mathcal{S}^{\mathrm{ref}}$ amounts to identify the rotated, scaled, and translated quadrilateral $\boldsymbol{r}^{\mathrm{ref}}$ that is "closest" to $\boldsymbol{r}$ w.r.t. a chosen distance measure. The curve obtained by the orthogonal projection is denoted as $\mathcal{P}^{\mathrm{ref}}\boldsymbol{r}$.

3.3 Spline-based Implementation

3.3.1 Construction of Vector Spaces Using Spline Curves

Example - Affine Transformation Combined with Translation

If we include a translation $\boldsymbol{b} \in \mathbb{R}^2$ in the transformation described in Section 3.2.1, then we obtain $\mathbf{A}\boldsymbol{r} + \boldsymbol{b}$, where $\mathbf{A}$ is defined as in Section 3.2.1 and $\boldsymbol{b} = (b_1, b_2)$. The transformation has six degrees of freedom and it is easy to see that a basis for the affine shape space of the reference curve $\boldsymbol{r}^{\mathrm{ref}}$ is given by

$$\{\boldsymbol{e}_i^{\mathrm{ref}}\}_{i\in[1,\ldots,6]} = \left\{ \begin{pmatrix} r_x^{\mathrm{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} r_y^{\mathrm{ref}} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ r_x^{\mathrm{ref}} \end{pmatrix}, \begin{pmatrix} 0 \\ r_y^{\mathrm{ref}} \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\},$$

which corresponds to

$$\{\mathbf{c}_{\boldsymbol{e}_i^{\text{ref}}}\}_{i\in[1,\dots,6]} = \left\{ \begin{pmatrix}\mathbf{c}_x\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{c}_y\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\ \mathbf{c}_x\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\ \mathbf{c}_y\end{pmatrix}, \begin{pmatrix}\mathbf{1}\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{0}\\ \mathbf{1}\end{pmatrix} \right\},$$

where $\mathbf{0}$ and $\mathbf{1}$ correspond to the vectors of size N (which is the size of $\mathbf{c}_x$ or $\mathbf{c}_y$) and whose elements are all 0 or 1, respectively. Here, $\mathbf{c}_x$ and $\mathbf{c}_y$ are the vectors that contain the coordinates of the control points. (This notation has been defined in Chapter 2.)
Using the simplified expression (2.13) to compute inner products of spline curves, we can now specify the projection operator. A fundamental aspect of our construction is not only that the curve $\boldsymbol{r}$ that is being projected is a spline curve, but also that the basis $\{\boldsymbol{e}_i^{\text{ref}}\}$ of the subspace $\mathcal{S}^{\text{ref}}$ consists of spline curves of the form given by (2.1). Hence, each of these curves $\boldsymbol{r} \in \mathcal{H}$ is uniquely determined by its corresponding vector of control points $\mathbf{c} \in \mathbb{R}^{2N}$.

In this chapter, we provide the expression for 2D curves, noting that the extension to 3D is straightforward. We first define the matrix

$$\mathbf{C}^{\text{ref}} = [\mathbf{c}_{\boldsymbol{e}_1^{\text{ref}}} \cdots \mathbf{c}_{\boldsymbol{e}_I^{\text{ref}}}]. \tag{3.2}$$

It has the dimension $(2N \times I)$ and contains the control points of the curves $\{\boldsymbol{e}_i^{\text{ref}}\}_{i=1,\dots,I}$ that define a basis of $\mathcal{S}^{\text{ref}}$. To simplify the notation, we collect all basis functions in the vector

$$\boldsymbol{\varphi}(t) := (\varphi_0(t), \dots, \varphi_{N-1}(t)). \tag{3.3}$$

The corresponding (orthogonal) spline projector $\mathcal{P}^{\text{ref}} : \mathcal{H} \to \mathcal{S}^{\text{ref}}$ that minimizes the distance between the curve $\boldsymbol{r} \in \mathcal{H}$ and the I-dimensional vector space $\mathcal{S}^{\text{ref}}$ is specified by Theorem 1.

Theorem 1. *Let $\boldsymbol{r}(t) = \boldsymbol{\varphi}(t)^{\mathsf{T}}\mathbf{c}$. Then,*

$$\mathcal{P}^{ref}\boldsymbol{r}(t) = \begin{pmatrix}\boldsymbol{\varphi}(t) & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\varphi}(t)\end{pmatrix}^{\mathsf{T}} \mathbf{P}^{\text{ref}}\mathbf{c},$$

where $\mathbf{P}^{\text{ref}} \in (\mathbb{R}^{2N} \times \mathbb{R}^{2N}) : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ is the $(2N \times 2N)$ projection matrix defined as

$$\mathbf{P}^{\text{ref}} = \mathbf{C}^{\text{ref}}({\mathbf{C}^{\text{ref}}}^T \boldsymbol{\Psi} \mathbf{C}^{\text{ref}})^{-1} {\mathbf{C}^{\text{ref}}}^T \boldsymbol{\Psi}.$$

The proof is provided in Appendix 3.7.2. Theorem 1 provides a direct method to compute the control points of the projected curve. Note that the projection of the vector $\mathbf{c}$ of control points is itself not orthogonal. However, it corresponds to the orthogonal projection of $\boldsymbol{r}(t)$ in the L_2-sense. Therefore, we have $(\mathbf{P}^{\mathrm{ref}})^2 = \mathbf{P}^{\mathrm{ref}}$ and $\mathbf{P}^{\mathrm{ref}^T} \neq \mathbf{P}^{\mathrm{ref}}$. Hence, Theorem 1 shows that $\mathbf{P}^{\mathrm{ref}}$ is an *oblique* projector from $\mathbb{R}^{2N}$ onto the I-dimensional invariant subspace of $\mathbb{R}^{2N}$ defined by the basis $\{\mathbf{c}_{\boldsymbol{e}_1^{\mathrm{ref}}}\}_{i=1,\dots,I}$. This means that $\mathcal{P}^{\mathrm{ref}} : \mathcal{H} \to \mathcal{S}^{\mathrm{ref}}$, which is the orthogonal projector in the L_2-sense, is efficiently implemented via the oblique projector $\mathbf{P}^{\mathrm{ref}} \in (\mathbb{R}^{2N} \times \mathbb{R}^{2N}) : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$.

3.3.2 Construction of Spline Projectors

We provide examples which illustrate how the projectors that correspond to the vector spaces listed in Table 3.1 are implemented using splines, in accordance with Theorem 1.

Scaling Projector (without translation)

The scaling projector can be expressed by solving $\min\limits_{a} \|a\boldsymbol{r}^{\mathrm{ref}} - \boldsymbol{r}\|_{L_2}^2$ such that $\mathcal{P}^{\mathrm{ref}}\boldsymbol{r}(t) = a\boldsymbol{r}^{\mathrm{ref}}$, where $a \in \mathbb{R}$ and $\boldsymbol{r}^{\mathrm{ref}}$ is the reference curve that defines the vector space. Its well known solution is $a = \frac{\langle \boldsymbol{r}^{\mathrm{ref}}, \boldsymbol{r}\rangle}{\langle \boldsymbol{r}^{\mathrm{ref}}, \boldsymbol{r}^{\mathrm{ref}}\rangle}$. Using (2.13), the corresponding spline projector is specified by $\mathbf{P}^{\mathrm{ref}} = \mathbf{c}^{\mathrm{ref}} \frac{\mathbf{c}^{\mathrm{ref}^{\mathsf{T}}} \boldsymbol{\Psi}}{\langle \mathbf{c}^{\mathrm{ref}}, \mathbf{c}^{\mathrm{ref}}\rangle_{\boldsymbol{\Psi}}}$, which corresponds to the solution obtained by the direct application of Theorem 1.

Similarity

The similarity transform is defined scaling of a curve $\boldsymbol{r}$ by a factor a combined with a rotation described by the rotation matrix $\mathbf{R}_\theta$ (applied to $\boldsymbol{r}^{\mathrm{ref}}$) and a translation given by $\boldsymbol{b} = (b_1, b_2)$. It is expressed as

$$\begin{aligned} a\mathbf{R}_\theta \boldsymbol{r}^{\mathrm{ref}} + \boldsymbol{b} &= \begin{pmatrix} a\cos\theta r_x^{\mathrm{ref}} - a\sin\theta r_y^{\mathrm{ref}} + b_1 \\ a\sin\theta r_x^{\mathrm{ref}} + a\cos\theta r_y^{\mathrm{ref}} + b_2 \end{pmatrix} \\ &= \alpha \begin{pmatrix} r_x^{\mathrm{ref}} \\ r_y^{\mathrm{ref}} \end{pmatrix} + \beta \begin{pmatrix} -r_y^{\mathrm{ref}} \\ r_x^{\mathrm{ref}} \end{pmatrix} + b_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \end{aligned}$$

where $a \in \mathbb{R}$, $\alpha = a\cos\theta$ and $\beta = a\sin\theta$. To construct the corresponding projector, we choose $\boldsymbol{e}_1^{\mathrm{ref}} = (r_x^{\mathrm{ref}}, r_y^{\mathrm{ref}})$, $\boldsymbol{e}_2^{\mathrm{ref}} = (-r_y^{\mathrm{ref}}, r_x^{\mathrm{ref}})$, $\boldsymbol{e}_3^{\mathrm{ref}} = (1,0)$ and $\boldsymbol{e}_4^{\mathrm{ref}} = (0,1)$ which corresponds to the basis $\mathbf{c}_{\boldsymbol{e}_1^{\mathrm{ref}}} = (\mathbf{c}_x^{\mathrm{ref}}, \mathbf{c}_y^{\mathrm{ref}})$, $\mathbf{c}_{\boldsymbol{e}_2^{\mathrm{ref}}} = (-\mathbf{c}_y^{\mathrm{ref}}, \mathbf{c}_x^{\mathrm{ref}})$, $\mathbf{c}_{\boldsymbol{e}_3^{\mathrm{ref}}} = (\mathbf{1},\mathbf{0})$ and $\mathbf{c}_{\boldsymbol{e}_4^{\mathrm{ref}}} = (\mathbf{0},\mathbf{1})$.

3.3.3 Example

We compare the affine space with the space defined by the similarity transform. We construct the two corresponding projectors w.r.t. the reference spline curve that represents the *white matter* structure of the brain, as shown in the left of Figure 3.2.

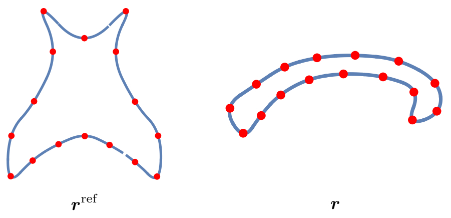


Figure 3.2: Left: representation of a "white matter" segment of a brain. Right: contour of a *corpus callosum* (brain structure). The blue contour is represented as a spline curve and the red dots are its landmarks (*i.e.*, the 2D spline coefficients given by $\{\boldsymbol{c}[k]\}_{k\in\mathbb{Z}}$).

We then project the curve shown in the right of Figure 3.2 (the *corpus callosum*) separately onto the affine, as well as onto the vector space defined by the similarity transform. Among all shapes enclosed by the given subspace defined by the reference shape (*i.e.*, white matter), the projector "chooses" the one closest to the *corpus callosum* (see Figure 3.3).

3.4 Application: Shape Priors for Segmentation

In segmentation algorithms, it is advantageous if prior knowledge about shapes can be integrated [46, 92, 96], for instance in active contour models [90, 94, 49, 1, 22]. Our proposed formulation of a projector allows us to compute the distance between

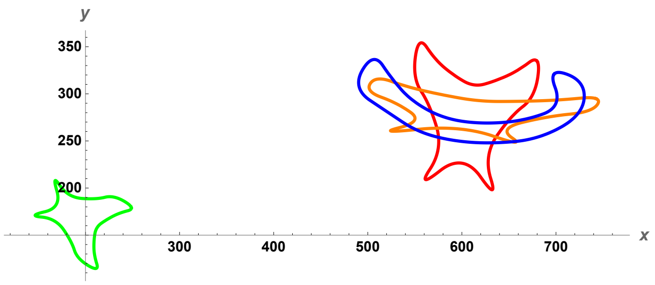


Figure 3.3: Affine vs. similarity: The *corpus callosum* (blue) is registered onto the white matter (green). The orange curve is the "closest" affine transform of the white matter (green) w.r.t. the *corpus callosum* (blue). The red curve is the "closest" deformed white matter (green) w.r.t. the *corpus callosum* (blue) using a similarity transform.

a shape and the vector space given by a reference shape. Hence, we can penalize the cases where a shape is distant from the given vector space defined by the reference shape $\boldsymbol{r}^{\text{ref}}$. To illustrate this concept, we consider the example of spline-based snakes [83]. We denote by $E(\Omega)$ the standard energy term that usually needs to be minimized w.r.t. the snake-defining parameters described by Ω. The minimization of $E(\Omega)$ attracts the contour of the snake towards the boundary of the object of interest. We propose to add a prior term to E. We define it as

$$E_{\text{prior}} = \gamma \|\boldsymbol{r} - \mathcal{P}^{\text{ref}}\boldsymbol{r}\|_{L_2}^2, \tag{3.4}$$

where $\gamma \in \mathbb{R}$ controls the contribution of the prior energy term. We develop E_{prior} as

$$\begin{aligned}
\|\boldsymbol{r} - \mathcal{P}^{\text{ref}}\boldsymbol{r}\|_{L_2}^2 &= \|(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{r}\|_{L_2}^2 \\
&= \langle(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{r}, (\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{r}\rangle_{L_2} \\
&= \langle\boldsymbol{r}, (\mathcal{I} - \mathcal{P}^{\text{ref}})^*(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{r}\rangle_{L_2} \\
&= \langle\boldsymbol{r}, (\mathcal{I} - \mathcal{P}^{\text{ref}})^2\boldsymbol{r}\rangle_{L_2} \\
&= \langle\boldsymbol{r}, (\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{r}\rangle_{L_2},
\end{aligned} \tag{3.5}$$

where $\mathcal{I}$ denotes the identity operator and where we have used the fact that, since $\mathcal{P}^{\mathrm{ref}}$ is orthogonal and hence, self-adjoint, then $(\mathcal{I} - \mathcal{P}^{\mathrm{ref}})$ is also an orthogonal projector. We use (2.13) to express the inner product and Theorem 1 to compute the projector. Then, (3.5) is developed as

$$\begin{aligned} \|\boldsymbol{r} - \mathcal{P}^{\mathrm{ref}}\boldsymbol{r}\|_{L_2}^2 &= \mathbf{c}^{\mathsf{T}}\boldsymbol{\Psi}(\mathbf{I} - \mathbf{P}^{\mathrm{ref}})\mathbf{c} \\ &= \mathbf{c}^{\mathsf{T}}\boldsymbol{\Psi}(\mathbf{I} - \mathbf{C}(\mathbf{C}^{\mathsf{T}}\boldsymbol{\Psi}\mathbf{C})^{-1}\mathbf{C}^{\mathsf{T}}\boldsymbol{\Psi})\mathbf{c} \\ &= \mathbf{c}^{\mathsf{T}}\mathbf{S}\mathbf{c}, \end{aligned} \tag{3.6}$$

where $S = \boldsymbol{\Psi}(\mathbf{I} - \mathbf{C}(\mathbf{C}^{\mathsf{T}}\boldsymbol{\Psi}\mathbf{C})^{-1}\mathbf{C}^{T}\boldsymbol{\Psi})$. In Figure 3.5, we illustrate how such prior knowledge improves robustness in a segmentation setting.

3.5 Robustness

An artificial image (8 bit, i.e. intensity values between 0 and 255) simulating a culture of rod-shaped cells has been created as shown in figure 3.5. To objectively validate the quality of the proposed energy term, the cells have different sizes, intensities and orientation. Because isolated cells are easier to segment we have also simulated cell clumps. The snakes have been initialized with different levels of overlap with the cells and they were optimized with and without the prior energy term. In an additional experiment the segmentation was repeated on the image corrupted by additive Gaussian white noise (std = 25, SNR = 0.44). The results shown in figure 3.5 clearly show the advantage of using the proposed shape prior for constructing snake energies. Most cells were correctly segmented using our approach. Besides, in the proposed implementation the user can manually correct inaccurate results.

3.6 Summary

We provide an explicit formulation to compute the minimal distance between an arbitrary query curve and a vector space defined by a reference. Our solution is generically characterized for landmark-based spline curves as a projection operator once the basis of the vector space is defined. This allows us to compute the

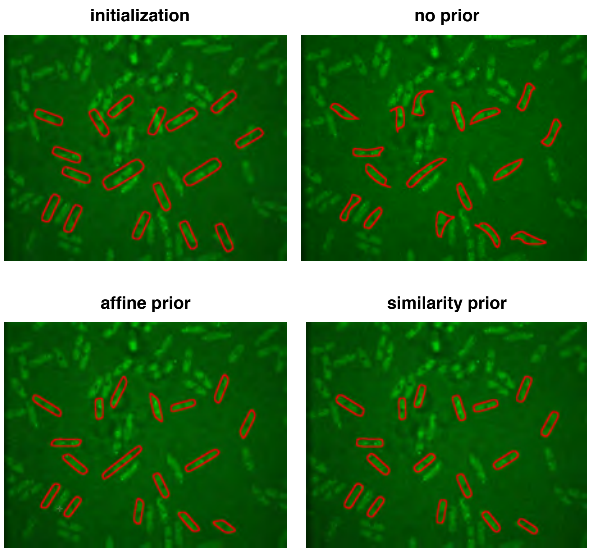


Figure 3.4: Segmentation of rod-shaped yeast cells [5]. Several snakes are initialized (top left) and a standard image-energy term is optimized without (top right) and with an affine (bottom left) as well as similarity (bottom right) prior. The shape prior corresponds to an approximate rod-shape.

continuous-domain distance as a fast matrix-vector operation. It can be used to efficiently characterize shape priors for landmark-based segmentation models. The spline-based solution has the additional advantage that the proposed construction can also be applied to curves that are defined by a set of discrete points or landmarks, by simply interpolating them with a linear B-spline. In Appendix 3.7.3 we show how the framework is extended to compute shape priors for 3D parametric spline surfaces.

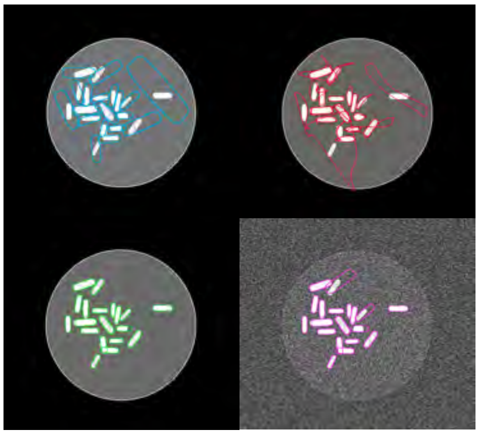


Figure 3.5: Snake segmentation using shape priors. Top left: Initialization of the snakes; top right: result of segmentation without shape prior and (bottom left) result of segmentation with shape prior; bottom right: result of segmentation with shape prior on noisy data.

3.7 Appendix

3.7.1 Derivation of Proposition 2

Proposition 2 follows from a standard result in functional analysis which states that the kernel of $\mathcal{P}$ is computed by

$$\begin{aligned}\mathcal{P}\boldsymbol{\phi}(t) &= \sum_{i=1}^{I} \boldsymbol{e}_i(t)\langle \tilde{\boldsymbol{e}}_i, \boldsymbol{\phi}\rangle = \langle \sum_{i=1}^{I} \boldsymbol{e}_i(t)\tilde{\boldsymbol{e}}_i^{\mathsf{T}}(\cdot), \boldsymbol{\phi}\rangle \\ &= \langle \boldsymbol{K}_{\mathcal{P}}(t,\cdot), \boldsymbol{\phi}\rangle\end{aligned} \tag{3.7}$$

and therefore, $\boldsymbol{K}_{\mathcal{P}}(t,s) = \sum_{i=1}^{I} \boldsymbol{e}_i(t) \otimes \tilde{\boldsymbol{e}}_i(s)$.

3.7.2 Proof of Theorem 1

Proof. If $\mathcal{P}$ is an orthogonal projector w.r.t. $\text{span}\{\boldsymbol{e}_i\}_{i=1,\ldots,I}$, then a curve $\boldsymbol{r}$ can always be decomposed as

$$\boldsymbol{r}(t) = \mathcal{P}\boldsymbol{r}(t) + \underbrace{(\mathcal{I} - \mathcal{P})\boldsymbol{r}(t)}_{\text{error}}, \tag{3.8}$$

where $\mathcal{I}$ is the identity operator and the error between the curve and the projective plane is orthogonal to the projective plane, so that error $\perp \text{span}\{\boldsymbol{e}_i\}$. By expressing $\mathcal{P}\boldsymbol{r} = \sum_i \langle \tilde{\boldsymbol{e}}_i, \boldsymbol{r}\rangle_{L_2} \boldsymbol{e}_i(t) = \sum_i u_i \boldsymbol{e}_i(t)$ in (3.8) and taking the inner product on both sides w.r.t. $\boldsymbol{e}_k$, we obtain the *normal* equation

$$\langle \boldsymbol{r}, \boldsymbol{e}_k\rangle_{L_2} = \sum_i u_i \langle \boldsymbol{e}_i, \boldsymbol{e}_k\rangle_{L_2} + \underbrace{\langle (\mathcal{I} - \mathcal{P})\boldsymbol{r}(t), \boldsymbol{e}_k\rangle_{L_2}}_{0}. \tag{3.9}$$

Evaluating (3.9) for all elements of the basis $\{\boldsymbol{e}_k\}$ and writing all the equations in matrix form, we obtain

$$\begin{pmatrix} \langle \boldsymbol{r}, \boldsymbol{e}_1\rangle_{L_2} \\ \vdots \\ \langle \boldsymbol{r}, \boldsymbol{e}_I\rangle_{L_2} \end{pmatrix} = \mathbf{G}\mathbf{u} \Leftrightarrow \mathbf{u} = \mathbf{G}^{-1} \begin{pmatrix} \langle \boldsymbol{r}, \boldsymbol{e}_1\rangle_{L_2} \\ \vdots \\ \langle \boldsymbol{r}, \boldsymbol{e}_I\rangle_{L_2} \end{pmatrix}.$$

Using the notation $\mathbf{E} = [\boldsymbol{e}_1 \cdots \boldsymbol{e}_I]$, $\mathbf{G} = \mathbf{C}^\mathsf{T}\mathbf{\Psi}\mathbf{C}$ being the Gram matrix with respect to the basis $\{\boldsymbol{e}_i\}$ and $\boldsymbol{g} = (\langle \boldsymbol{e}_1, \boldsymbol{r}\rangle_{L_2}, \ldots, \langle \boldsymbol{e}_I, \boldsymbol{r}\rangle_{L_2})$, the orthogonal projection of $\boldsymbol{r}$ is expressed as

$$\begin{aligned}\mathcal{P}\boldsymbol{r}(t) &= \mathbf{E}\mathbf{G}^{-1}\boldsymbol{g}\\ &= \sum_{i=1}^{I} \boldsymbol{e}_i(t)\, \underbrace{\langle \mathbf{G}^{-1}_{i,1}\boldsymbol{e}_1 + \cdots + \mathbf{G}^{-1}_{i,I}\boldsymbol{e}_I, \boldsymbol{r}\rangle_{L_2}}_{\langle \tilde{\boldsymbol{e}}_i, \boldsymbol{r}\rangle_{L_2}},\end{aligned} \tag{3.10}$$

where $\{\tilde{\boldsymbol{e}_i}\}$ forms the dual basis of $\{\boldsymbol{e}_i\}$ and is defined as $\tilde{\mathbf{E}} = \mathbf{E}(\mathbf{G}^{-1})^\mathsf{T} = [\tilde{\boldsymbol{e}_1} \cdots \tilde{\boldsymbol{e}_I}]$. Because φ generates a Riesz basis, each coordinate function of a spline curve $\boldsymbol{r}$ given by (2.1) is uniquely specified by its control points $\{\mathbf{c}[k]\}_{k\in\mathbb{Z}}$. This implies that there is a one-to-one relation between the coordinate functions of the curve and its spline coefficients. Hence, the matrix $\mathbf{E}$ in (3.10) that defines the basis $\{\boldsymbol{e}_i\}$ for the subspace $\mathcal{S}$ (not to be confused with the Riesz basis generated by φ) is fully specified by $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_I]$ which is the matrix that contains all the control points of the basis $\{\boldsymbol{e}_i\}$. Using (2.13), we rewrite (3.10) as

$$\mathcal{P}\boldsymbol{r}(t) = \mathbf{E}\mathbf{G}^{-1}\boldsymbol{g} = \mathbf{E}\mathbf{G}^{-1}\begin{pmatrix}\langle \boldsymbol{e}_1, \boldsymbol{r}\rangle_{L_2}\\ \vdots \\ \langle \boldsymbol{e}_I, \boldsymbol{r}\rangle_{L_2}\end{pmatrix},$$

which is equivalent to

$$\mathbf{C}\mathbf{G}^{-1}\begin{pmatrix}\mathbf{c}_1^\mathsf{T}\mathbf{\Psi}\\ \vdots \\ \mathbf{c}_I^\mathsf{T}\mathbf{\Psi}\end{pmatrix}\mathbf{c} = \mathbf{C}\mathbf{G}^{-1}\mathbf{C}^\mathsf{T}\mathbf{\Psi}\mathbf{c} = \mathbf{P}\mathbf{c}.$$

Hence, $\mathbf{P} = \mathbf{C}(\mathbf{C}^\mathsf{T}\mathbf{\Psi}\mathbf{C})^{-1}\mathbf{C}^\mathsf{T}\mathbf{\Psi}$. It is easily verified that $\mathbf{P}$ is a projector, characterized by the idempotent relation $\mathbf{P}^2 = \mathbf{P}$. $\square$

3.7.3 Orthogonal Spline Projectors for Parametric Surfaces

We refer to the notation defined in Section 2.3.4. Then, the matrix $\mathbf{E}$ that defines the basis $\{\boldsymbol{e}_i\}$ for the subspace $\mathcal{S}$ is fully specified by $\mathbf{C} = [\mathbf{\Gamma}_1 \ldots \mathbf{\Gamma}_I]$ which is the matrix that contains all the control points of the basis $\{\boldsymbol{e}_i\}$. Using the simplified expression (2.15) to compute spline-based inner products, the expression (3.10), *i.e.*,

$$\mathcal{P}\boldsymbol{\sigma}(u,v) = \mathbf{E}\mathbf{G}^{-1}\boldsymbol{g} = \mathbf{E}\mathbf{G}^{-1}\begin{pmatrix}\langle \boldsymbol{e}_1, \boldsymbol{\sigma}\rangle \\ \vdots \\ \langle \boldsymbol{e}_I, \boldsymbol{\sigma}\rangle\end{pmatrix}$$

is equivalent to

$$\mathbf{C}\mathbf{G}^{-1}\begin{pmatrix}\mathbf{\Gamma}_1^{\mathsf{T}}\boldsymbol{\beta} \\ \vdots \\ \mathbf{\Gamma}_I^{\mathsf{T}}\boldsymbol{\beta}\end{pmatrix}\mathbf{\Gamma} = \mathbf{C}\mathbf{G}^{-1}\mathbf{C}^{\mathsf{T}}\boldsymbol{\beta}\mathbf{\Gamma} = \mathbf{P}\mathbf{\Gamma},$$

where $\mathbf{G} = \mathbf{C}^{\mathsf{T}}\boldsymbol{\beta}\mathbf{C}$ is the Gram matrix of the basis $\{\boldsymbol{e}_i\}$. Hence, $\mathbf{P} = \mathbf{C}(\mathbf{C}^{\mathsf{T}}\boldsymbol{\beta}\mathbf{C})^{-1}\mathbf{C}^{\mathsf{T}}\boldsymbol{\beta}$. It is easily verified that $\mathbf{P}$ is a projector characterized by the idempotent relation $\mathbf{P}^2 = \mathbf{P}$.

Prior Shape Energies for Surfaces

We develop the term that depends on the surface $\boldsymbol{\sigma}$ in E_{prior} as

$$\begin{aligned}\|\boldsymbol{\sigma} - \mathcal{P}^{\text{ref}}\boldsymbol{\sigma}\|_{L_2}^2 &= \|(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{\sigma}\|_{L_2}^2 \\ &= \langle(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{\sigma}, (\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{\sigma}\rangle \\ &= \langle\boldsymbol{\sigma}, (\mathcal{I} - \mathcal{P}^{\text{ref}})^*(\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{\sigma}\rangle \\ &= \langle\boldsymbol{\sigma}, (\mathcal{I} - \mathcal{P}^{\text{ref}})^2\boldsymbol{\sigma}\rangle \\ &= \langle\boldsymbol{\sigma}, (\mathcal{I} - \mathcal{P}^{\text{ref}})\boldsymbol{\sigma}\rangle,\end{aligned} \tag{3.11}$$

where $\mathcal{I}$ denotes the identity operator and where we have used the fact that, since $\mathcal{P}^{\text{ref}}$ is orthogonal and, hence, self-adjoint, then $(\mathcal{I} - \mathcal{P}^{\text{ref}})$ is also an orthogonal projector. By using (2.15) to compute a spline-based inner product and combining it with the expression for a spline projector, (3.11) simplifies to

$$\begin{aligned}\|\boldsymbol{\sigma} - \mathcal{P}^{\mathrm{ref}}\boldsymbol{\sigma}\|_{L_2}^2 &= \mathbf{\Gamma}^\mathsf{T}\boldsymbol{\beta}(\mathbf{I} - \mathbf{P}^{\mathrm{ref}})\mathbf{\Gamma} \\ &= \mathbf{\Gamma}^\mathsf{T}\boldsymbol{\beta}(\mathbf{I} - \mathbf{C}(\mathbf{C}^\mathsf{T}\boldsymbol{\beta}\mathbf{C})^{-1}\mathbf{C}^\mathsf{T}\boldsymbol{\beta})\mathbf{\Gamma} \\ &= \mathbf{\Gamma}^\mathsf{T}\mathbf{S}\mathbf{\Gamma},\end{aligned} \tag{3.12}$$

where $S = \boldsymbol{\beta}(\mathbf{I} - \mathbf{C}(\mathbf{C}^\mathsf{T}\boldsymbol{\beta}\mathbf{C})^{-1}\mathbf{C}^T\boldsymbol{\beta})$.

Chapter 4

Shape Encoding and Sparse Dictionary Learning in the Continuous Domain

Overview

We provide a generic framework to *learn* shape dictionaries for landmark-based curves that are defined in the continuous domain [1]. We first present an unbiased alignment method that involves the construction of a *mean* shape as well as training sets whose elements are subspaces that contain all affine transformations of the training samples. The alignment relies on the orthogonal projection operators that we have constructed in Chapter 3 and that have a closed form. Secondly, we present algorithms to learn shape dictionaries according to the structure of the data that needs to be encoded: a) projection-based functional principal component analysis for homogeneous data and b) continuous-domain sparse shape encoding to learn dictionaries that contain imbalanced data, outliers or different types of shape structures. We then provide a detailed and *exact* implementation of our method by making use of parametric spline curves. We demonstrate that our method requires fewer parameters than pure discrete methods, and that it is computationally more

[1]The chapter is based on our publication [74]

efficient and accurate. We illustrate the use of our framework to learn dictionaries on biomedical images as well as for shape analysis in bioimaging.

4.1 Introduction

Given a training set of K parametric curves $\{\boldsymbol{r}_k\}_{k=1,\dots,K}$, $\boldsymbol{r}_k(t) \in L_2([0,1],\mathbb{R}^2)$, with K shapes defined by a set of landmarks, we aim at learning a dictionary whose *atoms* best capture the shape variability of the training set. We first define for each curve $\boldsymbol{r}_k$ a subspace $S_k = \{\mathbf{A}\boldsymbol{r}_k + \boldsymbol{b} : \mathbf{A} \in \mathbb{R}^{2\times 2}, \boldsymbol{b} \in \mathbb{R}^2\}$ that contains all *allowable* affine or similarity transformations of $\boldsymbol{r}_k$. Next, we compute the *mean* shape $\boldsymbol{r}_{\text{mean}}$ that is closest to all subspaces S_k and project it back onto each S_k (see Figure 4.1) to obtain an *aligned* training set $\{\tilde{\boldsymbol{r}}_k = \mathcal{P}_k\boldsymbol{r}\}_{k=1,\dots,K}$, where $\mathcal{P}_k : L_2([0,1],\mathbb{R}^2) \to S_k$ is the orthogonal projection operator that projects a *query* curve $\boldsymbol{r}$ onto S_k. We use the aligned training data to learn dictionaries by either computing a continuous-domain *functional* principal component analysis (fPCA) or for *sparse shape encoding*, depending on the structure of the data. Our approach allows one to construct dictionaries that contain atoms that are *invariant* to the specific affine transformation that is used. For instance, if the geometric transformation is chosen to be a *similarity* transform, the resutling PCA does not depend on the location, size, or orientation of the original curves $\{\boldsymbol{r}_k\}_{k=1,\dots,K}$. [23].

4.1.1 Contribution

Construction of Mean Shape and Alignment of Curves

We use the projectors defined in Chapter 3 to compute a mean shape, which we use to align a training set by "removing" from the data the corresponding affine transformation that is used to construct the vector space. The specificity of our alignment method is that it does not depend on the particular choice of a *reference* shape or *template* and in that sense, is unbiased. We also provide a closed-form solution instead of an iterative method.

Dictionary Learning with Projection-based functional PCA

We show how to compute a functional PCA for parametric curves with the aligned training set. The principal components are used as atoms to construct the learned

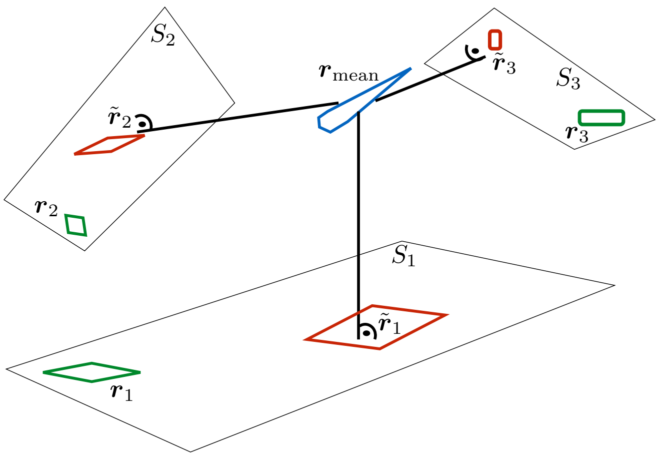

S_2
$\tilde{\boldsymbol{r}}_2$
$\boldsymbol{r}_2$
$\boldsymbol{r}_{\text{mean}}$
$\tilde{\boldsymbol{r}}_3$
S_3
$\boldsymbol{r}_3$
S_1
$\tilde{\boldsymbol{r}}_1$
$\boldsymbol{r}_1$

Figure 4.1: Unbiased shape alignment of curves. For each curve $\boldsymbol{r}_k$ the vector space S_k is computed w.r.t. an allowable geometric transformation. The closest shape, $\boldsymbol{r}_{\text{mean}}$, to all subspaces S_k, is computed and projected back to each subspace, which yields the aligned shapes $\tilde{\boldsymbol{r}}_k$ which define the data that we use to compute the shape dictionary

dictionary.

Exact Implementation using Spline Curves

We provide formulas for the exact implementation of our continuous-domain framework using splines. We derive the equivalent spline-based representation of the fPCA and show how our model is implemented at now additional cost compared to a purely discrete approach. Yet, we benefit from the fact that spline curves need fewer parameters than common landmark-based methods to accurately describe a shape.

Sparse Shape Encoding

We present a method that enforces sparsity to learn dictionaries that can be applied to training data unsuitable to be analyzed with L_2 methods. We provide formulas to express the continuous-domain L_2 norm of any spline curve as a discrete l_2 norm. We show how to exploit these forumlas to convert the continuous domain $L_2 - l_1$ *sparse coding* problem into a discrete $l_2 - l_1$ optimization problem; the crucial step for sparse shape encoding.

4.2 Related Work

4.2.1 Sparse Learning Methods

Sparse signal representation models that typically involve the minimum of an l_1-norm provide more flexibility than l_2-based methods for the encoding of training data because, 1) unlike PCA-related methods, they do not enforce orthogonality on the basis vectors, and 2), they are less sensitive to outliers or inhomogeneous data [97]. Methods for sparse dictionary learning, such as sparse PCA [98, 99, 100] have been proposed for image denoising [101] or to solve image classification tasks [97]. In the context of shape analysis, sparse learning methods have been applied to medical imaging [102, 103]. However, since these algorithms are formulated in the discrete domain they are also limited by the compromise that is requried to achieve an accurate description of the shape while keeping the number of shape descriptors, *i.e.*, landmarks, low.

4.2.2 Statistical Shape Models

l_2-based learning methods to characterize shape data and capture shape variability can be traced back to the classical *point distribution model* (PDM), which is the basis of the *active shape model* (ASM) [71, 73]. Landmark-based curves are aligned by minimizing the variance of the distance between corresponding points. Originally the ASM was introduced to segment images: its main difference with *active contour models* [90, 1, 49] is that it only allows for deformation to fit data such that it is consistent with the training set. The ASM and related statistical shape models [72] usually require that the training set be aligned or registered to a common reference prior to the statistical analysis. Iterative methods, such as the popular *procrustes* *analysis* [70] are used to compute a mean shape from a properly aligned set of training data. A PCA is then applied to the renormalized training data to compute the modes that describe the variation within the data. Although different alignment strategies exist, it remains a challenge to reduce the bias that is introduced when computing the mean shape [104]. Furthermore, these algorithms are iterative which can be a limitation if fast online methods are required. Moreover, they do not allow for a flexible choice of the particular geometric transformation (*e.g.*, rigid-body, similarity, scaling) that is removed when renormalizing which makes the models only applicable to a specific class of shapes. The methods mentioned above are considered as discrete methods. Attempts to construct statistical shape models in the continuous domain have been proposed by making use of B-splines [105]; however, they do not fully exploit the L_2 Hilbert-space structure of parametric spline shapes.

Statistical shape models are closely related to shape analysis [106] or segmentation models because they are often used to incorporate prior information about shapes into an algorithm [107, 46, 108, 92]. In this context, spline-based curve representations are convenient because they enable to implement smooth shapes in the continuous domain [21, 20, 22] with only few parameters.

4.3 Curve Projectors

Given a training set $\{\boldsymbol{r}_k\}_{k=1,\ldots,K}$ of curves, it is necessary to first align the shapes in order to construct a dictionary. This step corresponds to the centering of the data vectors in a classical PCA. To guarantee an unbiased alignment, we propose to associate to each sample curve $\boldsymbol{r}_k$ a subspace S_k as defined in Chapter 3 that contains all allowable affine transformation of $\boldsymbol{r}_k$ as described in Section 3.2.1. Then, we compute the curve $\boldsymbol{r}_{\text{mean}}$ that is the closest to all subspaces, and project it back to them to obtain the aligned curves $\{\tilde{\boldsymbol{r}}_k\}_{k=1,\ldots,K}$ (see Figure 4.1).

4.4 Mean Shape and Alignment

In the case where we are dealing with several reference curves (*i.e.*, a *training* set of reference shapes), we define one vector space $S_k := \mathcal{S}_k^{\text{ref}}$ for each curve $\boldsymbol{r}_k := \boldsymbol{r}_k^{\text{ref}}$. Merging all these subspaces results in a large space of transformations of different curves. Since in the training set some shape configurations occur more frequently than others, we want to construct the dominant or *mean* shape given the training data and a class of transformations. We assume that all the subspaces have the same dimension I and formalize the problem as finding the curve that is closest to all the subspaces S_k, each being specified by its corresponding projector $\mathcal{P}_k := \mathcal{P}_k^{\text{ref}} : \boldsymbol{r} \mapsto \sum_{i=1}^{I} \boldsymbol{e}_i^k(t)\langle \tilde{\boldsymbol{e}}_i^k, \boldsymbol{r}\rangle$ (see Figure 4.1). This problem can be formulated in a variational form if we impose the condition that the mean shape should have unit norm. Although this condition is arbitrary, it does not influence the result; in practice, we are only interested in the shape up to a scaling factor. The principal curve is determined as follows.

The curve $\boldsymbol{r}_{\text{mean}}$ that is closest to all subspaces S_k for $k = 1, \dots, K$ is obtained by solving

$$\arg\max_{\boldsymbol{r}_{\text{mean}}} \sum_{k=1}^{K} \|\mathcal{P}_k \boldsymbol{r}_{\text{mean}}\|_{L_2}^2 \quad \text{s.t.} \quad \|\boldsymbol{r}_{\text{mean}}\|_{L_2}^2 = 1, \tag{4.1}$$

which is equivalent to the eigenvalue problem

$$\sum_{k=1}^{K} \mathcal{P}_k \boldsymbol{r}_{\text{mean}}(t) = \lambda \boldsymbol{r}_{\text{mean}}(t) \quad \text{s.t.} \quad \langle \boldsymbol{r}_{\text{mean},p}, \boldsymbol{r}_{\text{mean},q}\rangle = \delta_{p-q}, \tag{4.2}$$

where we have used the fact that all the $\mathcal{P}_k$ are orthogonal, which implies that $\mathcal{P}_k^* \mathcal{P}_k = \mathcal{P}_k$, where $\mathcal{P}_k^*$ is the adjoint of $\mathcal{P}_k$.

Solutions of the Eigenequation

To solve (4.1), we invoke Propositon 2 and reformulate problem (4.2) as

$$\sum_{k=1}^{K} \mathcal{P}_k \boldsymbol{r}_{\text{mean}}(t) = \sum_{k=1}^{K} \langle \boldsymbol{K}_{\mathcal{P}_k}(t, \cdot), \boldsymbol{r}_{\text{mean}}\rangle = \lambda \boldsymbol{r}_{\text{mean}}(t). \tag{4.3}$$

Equation (4.3) is a Volterra equation whose kernel $\boldsymbol{K}_{\mathcal{P}}$ consists of a finite sum. In Theorem 2, we characterize the solutions of (4.3) as the principal components of the eigenequation (4.2).

Theorem 2. *Let the* $(K \cdot I) \times (K \cdot I)$ *matrix* $\mathbf{\Gamma}$ *be defined as*

$$[\mathbf{\Gamma}]_{(k-1)\cdot I+i,(j-1)\cdot K+l} = \langle \tilde{\boldsymbol{e}}_i^{(k)}, \boldsymbol{e}_j^{(l)} \rangle, \tag{4.4}$$

where $k, l \in [1, \ldots, K]$ *and* $i, j \in [1, \ldots, I]$*. Then, the pth eigencurve of* (4.2) *is given as*

$$\boldsymbol{r}_{\text{mean},p}(t) = \sum_{k=1}^{K} \sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)} \gamma_{ik}^{(p)}, \tag{4.5}$$

where $\gamma_{ik}^{(p)}$ *is the entry indexed by* $(i-1) \cdot K + k$ *of the pth eigenvector of the matrix* $\mathbf{\Gamma}$*.*

The proof is given in Appendix 4.11.1. We show in Appendix 4.11.2 how to interpret this result in practice.

4.4.1 Unbiased Curve Alignment

Now we associate to the training set $\{\boldsymbol{r}_k\}_{k=1,\ldots,K}$ the *aligned* curves

$$\{\tilde{\boldsymbol{r}}_k = \mathcal{P}_k \boldsymbol{r}_{\text{mean}}\}_{k=1,\ldots,K} \tag{4.6}$$

as illustrated in Figure 4.1 , where $\boldsymbol{r}_{\text{mean}}$ is the mean shape computed above. It is worth noticing that the proposed method for aligning the curves is independent of the location of each member of the training set within each subspace $\mathcal{S}_k$.

4.5 Projection-Based Functional PCA for Curves

We now construct a *functional* PCA on the aligned training set (4.6). Since the curves $r \in \mathcal{H}$ are defined in the continous domain it is not possible to apply a discrete-domain PCA to our data. In the discrete domain a training set with K curves - each curve defined by Q landmarks or samples given by their x and y coordinates - is represented as a $2Q \times K$ data matrix and then a discrete domain PCA is performed [109]. Here, our data is of dimension "$2\infty \times K$" and therefore, we use operators instead of matrices to perform a *functional* PCA.

Definition 3. We define the (compact) data operator $\mathbf{X} : \mathbb{R}^K \to L_2([0,1],\mathbb{R}^2)$ as the operator whose kernel consists of K aligned curves as

$$\mathbf{X} = [\tilde{\boldsymbol{r}}_1(t) \cdots \tilde{\boldsymbol{r}}_K(t)],$$

where $\tilde{\boldsymbol{r}}_k$ is defined in (4.6). The adjoint $\mathbf{X}^* : L_2([0,1],\mathbb{R}^2) \to \mathbb{R}^K$ satisfies

$$\langle \boldsymbol{r}, \mathbf{X}\mathbf{v} \rangle_{L_2(\mathbb{R},\mathbb{R}^2)} = \langle \mathbf{X}^*\boldsymbol{r}, \mathbf{v} \rangle_{l_2(\mathbb{R}^K)}, \tag{4.7}$$

with $\mathbf{v} \in \mathbb{R}^K$ and $\boldsymbol{r} \in L_2([0,1],\mathbb{R}^2)$ and we emphasize that the two inner products in (4.7) have two distinct definitions.

We are looking for the optimal orthogonal base curves $\{\boldsymbol{\xi}_1(t),\dots,\boldsymbol{\xi}_K(t)\}$, $\boldsymbol{\xi}_k \in \mathcal{H}$ for $k = 1,\dots,K$, that decorrelate the training set. They are given by the eigencurves of the *scatter operator* $\mathbf{X}\mathbf{X}^* : L_2([0,1],\mathbb{R}^2) \to L_2([0,1],\mathbb{R}^2)$. Analogous to the discrete PCA, we can exploit the property that

- the non-zero eigenvalues of the scatter operator $\mathbf{X}\mathbf{X}^*$ and the *Gram matrix* $\mathbf{X}^*\mathbf{X} \in \mathbb{R}^{K\times K}$ (which corresponds to the *correlation matrix* in discrete PCA) are identical,
- the eigencurves $\{\boldsymbol{\xi}_k(t)\}_{k=1,\dots,K}$ of $\mathbf{X}\mathbf{X}^*$ are immediately obtained from the eigenvectors $\boldsymbol{v} \in \mathbb{R}^K$ as specified in Proposition 3.

Proposition 3. *The eigencurves $\boldsymbol{\xi}_k \in L_2([0,1],\mathbb{R}^2)$ of the scatter operator $\mathbf{X}\mathbf{X}^*$: $L_2([0,1],\mathbb{R}^2) \to L_2([0,1],\mathbb{R}^2)$ are specified by*

$$\mathbf{X}\mathbf{X}^*\{\boldsymbol{\xi}_k\}(t) = \lambda_k \boldsymbol{\xi}_k(t)$$

and are related to the eigenvectors $\boldsymbol{v}_k \in \mathbb{R}^K$ of the Gram matrix $\mathbf{X}^*\mathbf{X}$ by

$$\boldsymbol{\xi}_k = \frac{1}{\sqrt{\lambda_k}}\mathbf{X}\boldsymbol{v}_k$$
$$\boldsymbol{v}_k = \frac{1}{\sqrt{\lambda_k}}\mathbf{X}^*\boldsymbol{\xi}_k,$$

such that

$$(\mathbf{X}^*\mathbf{X})\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k,$$

where the λ_k are the non-zero eigenvalues of $\mathbf{X}^*\mathbf{X}$ which are identical to the non-zero eigenvalues of $\mathbf{X}\mathbf{X}^*$. Furthermore, the relation

$$\boldsymbol{v}_k^{\mathrm{T}}\boldsymbol{v}_l = \langle \boldsymbol{\xi}_k, \boldsymbol{\xi}_l \rangle = \delta_{k-l}$$

holds.

The Gram matrix of size $K \times K$ is computed as

$$\mathbf{X}^*\mathbf{X} = \begin{pmatrix} \langle \tilde{\boldsymbol{r}}_1, \tilde{\boldsymbol{r}}_1 \rangle & \dots & \langle \tilde{\boldsymbol{r}}_1, \tilde{\boldsymbol{r}}_K \rangle \\ \vdots & \ddots & \vdots \\ \langle \tilde{\boldsymbol{r}}_K, \tilde{\boldsymbol{r}}_1 \rangle & \dots & \langle \tilde{\boldsymbol{r}}_K, \tilde{\boldsymbol{r}}_K \rangle \end{pmatrix}.$$

Now, we can easily compute the principal curves by specifying the data array $\mathbf{Z}$ as

$$\mathbf{Z} = \mathbf{X}\mathbf{V}, \tag{4.8}$$

where

$$\mathbf{Z} = [\boldsymbol{z}_1(t) \cdots \boldsymbol{z}_K(t)]$$

and $\mathbf{V} = [\boldsymbol{v}_1 \dots \boldsymbol{v}_K]$ is the orthogonal matrix containing the eigenvectors of the Gram matrix. They can also be computed via the relation

$$\mathbf{Z} = [\sqrt{\lambda}_1 \boldsymbol{\xi}_1(t) \dots \sqrt{\lambda}_I \boldsymbol{\xi}_K(t)].$$

For a more in-depth description of functional PCA using compact operators we refer the reader to [110].

4.6 Implementation with Landmark-based Spline Curves

We now illustrate how the presented framework can be implemented using spline curves. For simplicity, we consider that the curves all have the same number N of control points and are constructed with the same basis function φ.

4.6.1 Mean Spline Shape

To compute the mean shape $\boldsymbol{r}_{\text{mean}}$ using splines, we take advantage of the fact that the spline coefficients uniquely specify a spline curve (Riesz basis property). We directly compute the vector of control points that defines $\boldsymbol{r}_{\text{mean}}$. Proposition 4 characterizes spline-based solution that corresponds to the eigenvalue problem stated in (4.1) in the case where the training set consists of spline curves.

Proposition 4. *Assume a training set of K spline curves $\boldsymbol{r}_k^{\text{ref}}$ of the form (2.1), where each curve defines a vector space $\mathcal{S}_k^{\text{ref}}$ through the spline coefficients given by the $(2N \times I)$ matrix $\mathbf{C}_k := \mathbf{C}_k^{\text{ref}}$ as specified by (3.2). Then, the vector of control points $\mathbf{c}_{\text{mean}}$ of the spline curve $\boldsymbol{r}_{\text{mean}}$ is given as the solution of the eigenequation*

$$\sum_{k=1}^{K} \mathbf{C}_k(\mathbf{C}_k^{\mathrm{T}}\boldsymbol{\Psi}\mathbf{C}_k)^{-1}\mathbf{C}_k^{\mathrm{T}}\boldsymbol{\Psi}\mathbf{c}_{\text{mean}} = \lambda\mathbf{c}_{\text{mean}}.$$

The proof is provided in Appendix 4.11.3.

4.6.2 Functional PCA for Spline Curves

Since $\{\varphi_n\}_{n=0,\dots,N-1}$ forms a Riesz basis, the data array $\mathbf{X}$ defined in Definition 4 is fully specified by the matrix of control points

$$\boldsymbol{\Omega} = [\mathbf{c}_1 \dots \mathbf{c}_K] \tag{4.9}$$

that define the curves $\{\tilde{\boldsymbol{r}}_k\}_{k=1,\dots,K}$. Using (2.13), the Gram matrix of $\mathbf{X}$ is computed as

$$\mathbf{X}^*\mathbf{X} = \boldsymbol{\Omega}^{\mathrm{T}}\boldsymbol{\Psi}\boldsymbol{\Omega} \tag{4.10}$$

and hence, the $(2N \times K)$-matrix $\boldsymbol{\Omega}_{\mathbf{Z}}$ that contains the control points of the principal curves is immediately computed by

$$\boldsymbol{\Omega}_{\mathbf{Z}} = \boldsymbol{\Omega}\mathbf{V}, \tag{4.11}$$

where as detailed in Section 4.5, $\mathbf{V}$ is the orthogonal matrix that contains the eigenvectors of the Gram matrix. The principal curves $\boldsymbol{z}_k(t)$ are finally obtained by

$$\boldsymbol{z}_k(t) = ([\mathbf{\Omega_Z}]_{\mathrm{All},k})^{\mathrm{T}} \begin{pmatrix} \varphi(t) & \mathbf{0} \\ \mathbf{0} & \varphi(t) \end{pmatrix}, \tag{4.12}$$

where we have adopted *mathematica* notation to describe $[\mathbf{\Omega_Z}]_{\mathrm{All},k}$ as the kth column of $\mathbf{\Omega_Z}$. More generally, we have

$$\mathbf{Z} = \mathbf{\Omega}_\mathbf{Z}^{\mathrm{T}} \begin{pmatrix} \varphi(t) & \mathbf{0} \\ \mathbf{0} & \varphi(t) \end{pmatrix}. \tag{4.13}$$

4.7 Sparse Shape Encoding

PCA uses all of the data curves to compute the principal curves. This makes it prone to outliers which might compromise the robustness when learning a shape dictionary. We propose a dictionary learning approach that only uses a sparse subset of the data to encode the shapes. For this purpose, we first derive a property specific to the spline representation of curves, which allows us to express the continuous domain L_2 norm as a discrete domain l_2 norm.

4.7.1 L_2-l_2 Norm Equality

Theorem 3. *For any data array* $\mathbf{D} = [\boldsymbol{d}_1(t) \dots \boldsymbol{d}_K(t)]$ *whose elements are parametric spline curves described by the matrix of control points* $\mathbf{\Omega_D} = [\mathbf{c}_{\boldsymbol{d}_1} \dots \mathbf{c}_{\boldsymbol{d}_K}]$ *and any spline curve* $\boldsymbol{r}$ *specified by the vector of control points* $\mathbf{c}$*, we have the norm equality*

$$\|\boldsymbol{r} - \mathbf{D}\boldsymbol{\alpha}\|_{L_2}^2 = \|\tilde{\boldsymbol{c}} - \tilde{\mathbf{D}}\boldsymbol{\alpha}\|_{l_2},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^K$,

$$\tilde{\boldsymbol{c}} = \mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^{-1}\mathbf{c}, \tag{4.14}$$

$$\tilde{\mathbf{D}} = \mathbf{Q}\mathbf{\Lambda}^{1/2}\mathbf{Q}^{-1}\mathbf{\Omega_D} \tag{4.15}$$

and

$$\mathbf{\Psi} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}, \tag{4.16}$$

such that $\mathbf{Q}$ *is an orthogonal matrix, whose columns are the eigenvectors of* $\boldsymbol{\Psi}$*, and* $\boldsymbol{\Lambda}$ *is the diagonal matrix that contains the eigenvalues of* $\boldsymbol{\Psi}$ *defined by* (2.14)*.*

Proof. We develop the L_2 norm as follows.

$$\begin{aligned}\|\boldsymbol{r} - \mathbf{D}\boldsymbol{\alpha}\|_{L_2}^2 &= \left\| \begin{pmatrix} \boldsymbol{\varphi}(t) & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(t) \end{pmatrix}^{\mathsf{T}} (\mathbf{c} - \boldsymbol{\Omega}_{\mathbf{D}}\boldsymbol{\alpha}) \right\|_{L_2}^2 \\ &= (\mathbf{c} - \boldsymbol{\Omega}_{\mathbf{D}}\boldsymbol{\alpha})^{\mathsf{T}} \boldsymbol{\Psi} (\mathbf{c} - \boldsymbol{\Omega}_{\mathbf{D}}\boldsymbol{\alpha}),\end{aligned} \tag{4.17}$$

where $\boldsymbol{\varphi}$ is defined in (3.3). Since $\boldsymbol{\Psi}$ is a positive-definite symmetric matrix it admits an eigendecomposition of the form

$$\boldsymbol{\Psi} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}, \tag{4.18}$$

where $\mathbf{Q}$ is an orthogonal matrix, *i.e.*, $\mathbf{Q}^{-1} = \mathbf{Q}^{T}$, whose columns are the eigenvectors of $\boldsymbol{\Psi}$, and $\boldsymbol{\Lambda}$ is the diagonal matrix that contains the eigenvalues of $\boldsymbol{\Psi}$. Therefore, we have

$$\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1} = \left(\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}\right)^{\mathsf{T}}, \tag{4.19}$$

which allows us to re-express (4.17) as

$$\begin{aligned}&\|(\mathbf{c} - \boldsymbol{\Omega}_{\mathbf{D}}\boldsymbol{\alpha})^{\mathsf{T}}\boldsymbol{\varphi}\|_{L_2}^2 \\ &= \|\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}(\mathbf{c} - \boldsymbol{\Omega}_{\mathbf{D}}\boldsymbol{\alpha})\|_{l_2}^2 \\ &= \|\underbrace{\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}\mathbf{c}}_{\tilde{\mathbf{c}}} - \underbrace{\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}\boldsymbol{\Omega}_{\mathbf{D}}}_{\tilde{\mathbf{D}}}\boldsymbol{\alpha}\|_{l_2}^2.\end{aligned} \tag{4.20}$$

□

4.7.2 Continuous-Domain Sparse Dictionary Learning

The projection-based functional PCA described in Section 4.5 is a pure L_2-based method. It is well known that such methods are sensitive to outliers, as well as imbalanced, or inhomogeneous data sets. Hence, there exist practical settings where those models are less suitable. Another limitation of the fPCA is the orthogonality constraint on the eigencurves, which might be unnecessary and too restrictive in certain scenarios.

Here, again we consider a training set of parametric curves $\mathbf{X} = [\boldsymbol{x}_1(t) \dots \boldsymbol{x}_K(t)]$ that are defined in the continuous domain as specified in Theorem 3. However, now we aim at constructing a dictionary $\mathbf{D} = \mathbf{D}(t) = [\boldsymbol{d}_1(t) \dots \boldsymbol{d}_J(t)]$ with $J <= K$ and where the $\{\boldsymbol{d}_j(t)\}_{j=1,\dots,J}$ are parametric curves, such that $\mathbf{D}$ yields the optimal value of the continuous domain *sparse coding* problem which is defined in analogy to its discrete counterpart [111, 112] as

$$\operatorname*{arg\,min}_{\boldsymbol{\alpha}\in\mathbb{R}^J}\left\{\frac{1}{2}\|\boldsymbol{x}_k - \mathbf{D}\boldsymbol{\alpha}_k\|^2_{L_2} + \lambda\|\boldsymbol{\alpha}_k\|_{l_1}\right\} \tag{4.21}$$

for all the $\boldsymbol{x}_k(t)$ in the training set, where $\lambda \in \mathbb{R}$ is a regularization parameter. The problem (4.21) is well studied [113] and known as the *Lasso* [114] method or *basis pursuit* [115]. If we enforce orthonormality on $\boldsymbol{\alpha}$ instead of sparsity, *i.e.*, $\langle \boldsymbol{\alpha}_k, \boldsymbol{\alpha}_l, \rangle = \delta_{k-l}$ and $\lambda = 0$, we recover the exact functional PCA solution (4.8) with $\boldsymbol{\alpha}_k = \boldsymbol{v}_k$. On the other hand, for $\lambda > 0$ we obtain a sparse vector $\boldsymbol{\alpha}_k$.

However, here the goal in the construction of $\mathbf{D}$ is that it yields an accurate approximation of a shape $\boldsymbol{x}(t) \approx \mathbf{D}(t)\boldsymbol{\alpha}$ such that each curve $\boldsymbol{x}$ only uses a few elements of $\mathbf{D}$ in its representation. By making use of spline curves we invoke Theorem 3, which allows us to formulate the continuous domain *sparse coding* problem in the discrete domain as

$$\operatorname*{arg\,min}_{\boldsymbol{\alpha}\in\mathbb{R}^J}\left\{\frac{1}{2}\|\boldsymbol{x}_k - \mathbf{D}\boldsymbol{\alpha}_k\|^2_{L_2} + \lambda\|\boldsymbol{\alpha}_k\|_{l_1}\right\} \tag{4.22}$$

$$= \operatorname*{arg\,min}_{\boldsymbol{\alpha}\in\mathbb{R}^J}\left\{\frac{1}{2}\|\tilde{\boldsymbol{x}}_k - \tilde{\mathbf{D}}\boldsymbol{\alpha}_k\|^2_{l_2} + \lambda\|\boldsymbol{\alpha}_k\|_{l_1}\right\}, \tag{4.23}$$

where

$$\tilde{\boldsymbol{x}}_k = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}[\boldsymbol{\Omega}]_{\text{All},k} \tag{4.24}$$

with $[\boldsymbol{\Omega}]_{\text{All},k}$ being the vector of control points of the kth curve of $\mathbf{X}$ as specified in (4.9) and

$$\tilde{\mathbf{D}} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\mathbf{Q}^{-1}\boldsymbol{\Omega}_{\mathbf{D}} = [\tilde{\boldsymbol{d}_1} \dots \tilde{\boldsymbol{d}_J}], \tag{4.25}$$

with $\mathbf{\Omega_D}$ being the matrix of control points that describe the parametric curves, *i.e.*, the atoms that form the continuous domain dictionary $\mathbf{D}(t)$.

To solve the discrete domain *sparse coding* problem we prevent $\tilde{\mathbf{D}}$ from becoming arbitrarily large by enforcing the l_2-norm of its column vectors to be smaller or equal to one. As suggested in [112, 97] this allows us to define the convex set of possible dictionaries as

$$\mathcal{C} := \{\tilde{\mathbf{D}} \in \mathbb{R}^{2N\times J} \quad \text{s.t.} \quad \|\tilde{\boldsymbol{d}}_j\|_{l_2} \leq 1, j = 1, \dots, J\}, \tag{4.26}$$

where N is the number of control points used to describe a spline curve (2.1). Now $\tilde{\mathbf{D}}$ is found by solving the joint optimization problem

$$\underset{\tilde{\mathbf{D}}\in\mathcal{C},\boldsymbol{\alpha}\in\mathbb{R}^J}{\arg\min} \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{2}\|\tilde{\boldsymbol{x}}_k - \tilde{\mathbf{D}}\boldsymbol{\alpha}_k\|_{l_2}^2 + \lambda\|\boldsymbol{\alpha}_k\|_{l_1}\right), \tag{4.27}$$

which is convex w.r.t. the two variables $\tilde{\mathbf{D}}$ and $\boldsymbol{\alpha}$ when one of them is fixed. Finally, from (4.25), we see that

$$\mathbf{\Omega_D} = \mathbf{Q}\mathbf{\Lambda}^{-1/2}\mathbf{Q}^{-1}\tilde{\mathbf{D}}$$

and therefore, the continuous-domain dictionary is computed through

$$\mathbf{D}(t) = \begin{pmatrix} \boldsymbol{\varphi}(t) & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(t) \end{pmatrix}^{\mathsf{T}} \mathbf{\Omega_D} \tag{4.28}$$

with $\boldsymbol{\varphi}$ as defined in (3.3).

Optimization

The joint optimization problem (4.27) can be solved by alternating methods which keep one variable fixed while minimizing the other one as described in [116, 117, 111]. Here we make use of the online optimization algorithm which is based on stochastic approximations [118, 119] and implemented in the popular SPAMS library written by Mairal et al. [112, 97]. It minimizes sequentially a quadratic local approximation of the expected cost function and is well suited for efficient and fast solving of online dictionary optimization problems with large training sets. Since the focus of this article is not the optimization itself, we refer the reader to the references mentioned above for a detailed description of the algorithm and implementation details.

4.8 Comparison with Existing Methods

4.8.1 Existing Linear Construction Methods for Dictionaries

The classical approach to learn dictionaries is to consider K shapes that are described by an ordered set of N points or landmarks in $\mathbb{R}^2$ [71]. The shapes themselves are represented as one large vector $\boldsymbol{r}_k \in \mathbb{R}^{2N}$, where $k \in [1, \ldots K]$. They are geometrically normalized by aligning them to a common reference in order to remove some effects of rigid-body transforms. The alignment to a reference shape $\boldsymbol{r}^{\text{ref}}$, is computed as $\tilde{\boldsymbol{r}}_k = \mathbf{A}\boldsymbol{r}_k + \mathbf{b}$, where $\mathbf{A}$ is an affine transformation matrix and $\boldsymbol{b} \in \mathbb{R}^2$ a translation vector such that they solve $\min\limits_{\mathbf{A},\boldsymbol{b}} \|\boldsymbol{r}^{\text{ref}} - \mathbf{A}\boldsymbol{r}_k - \boldsymbol{b}\|_{l_2}^2$. A standard PCA is then applied to the set of aligned shapes $\{\tilde{\boldsymbol{r}}_k\}_{k=1,\ldots K}$. Aside from only operating with discrete data, the standard approach has the drawback of being potentially biased because distances between normalized shapes generally differ from distances between non-normalized shapes. The fundamental difference between the classical approach and our method lies in the different concepts that define projective geometry and affine geometry. We exploit the fact that the solution of $\min\limits_{\mathbf{A},\boldsymbol{b}} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$ can be expressed (in closed form) as the orthogonal projection $\mathbf{P}^x\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{b}$, a property that holds for both discrete and continuous domain curves. This allows us to express the affine transformation as a projection onto a space which does not depend on the specific element $\boldsymbol{x}$ that lives in that space.

4.8.2 Closed-Form Solution for Continuous and Discrete Curves

The formulas that we present in this manuscript provide solutions in the continuous domain. In some applications, however, curves are defined by a discrete set of points. In this case, the solutions for spline-based curves can be applied because a discretized curve can always be expressed as a parametric curve using the linear B-spline [20] as basis function (see Section 4.9.1 for an example).

Equivalent Spline Solution Using Uniform Samples

One of the benefits of using a spline-based representation of curves is that it allows one to represent curves in the continuous domain with a small number N of control points. This becomes apparent when noticing that, for a uniformly discretized curve $\boldsymbol{r}$ given by the ordered set of points $\{\boldsymbol{r}(\frac{q}{Q})\}_{q=0,\dots,Q}$ with $(Q+1)$ samples, we have that

$$\begin{aligned}&\lim_{Q\to\infty}\frac{1}{Q}\sum_{q=0}^{Q}\left|\boldsymbol{r}_1\left(\frac{q}{Q}\right)-\mathbf{A}\boldsymbol{r}_2\left(\frac{q}{Q}\right)-\boldsymbol{b}\right|^2\\&=\int_0^1|\boldsymbol{r}_1(t)-\mathbf{A}\boldsymbol{r}_2(t)-\boldsymbol{b}|^2\mathrm{d}t.\end{aligned}$$

We see that, while the continuous curve $\boldsymbol{r}(t)$ is expressed with N control points which corresponds to a projection matrix $\mathbf{P}$ of size $(2N\times 2N)$, the discrete curve $\boldsymbol{r}(\frac{q}{Q})$ is described with $Q\gg N$ points whose corresponding projection matrix is of size $(2Q\times 2Q)$. This shows that a continuous-domain spline-based vector space can be implemented at no additional cost compared to a discrete curve described with N points, although the continuous curve is equivalent to a discrete setting where the number of points tends towards infinity. Hence, to be equivalent, we would have to use many more discrete points.

4.9 Validation and Experiments

4.9.1 Shape Analysis of Biological Structures

In biology often images acquired through microscopy are studied. Typically, the different samples of the same organism are studied in an image as for instance a colony of cells or bacteria. Being able to have representative shapes of such colonies is important to study for instance the reaction of an organism when exposed to a certain type of drug or chemical substance or to observe the behavior in specific environments. Next, we provide an example for shape analysis using real biological data.

Learning Shape Priors

We have manually outlined the 20 chromosomes that are shown in the microscopic image in Figure 4.2 (top). The outlining has been done by placing 12 landmarks on the contours of the chromosomes and interpolating them with the basis functions proposed in [76, 77]. This procedure allows us to obtain a spline-based curve description of each chromosome with landmarks that are corresponding throughout the data set.

The chromosomes share a similar symmetric approximate rod-shaped structure; however, they differ in their size, orientation, and location. Using our proposed framework, we first compute the aligned training set $\{\tilde{\boldsymbol{r}}_k\}_{k=1,\ldots,20}$ and then compute $\boldsymbol{r}_{\text{mean}}$ as given by Theorem 4. The resulting *learned* shape (Figure 4.2, red shape in middle row) can be further used for classification (see Section 4.9.2) or also as a *trained* shape-prior for segmentation problems [68, 67]. It characterizes the shape population in terms of its geometry and hence, can be viewed as an "average" shape.

Learned Shape vs. functional PCA vs. Point Distribution Model

To test the accuracy of the learned shape-prior, we compare it to the first eigenshape obtained through the projection-based functional PCA described in Section 4.5 and the *mean* shape obtained with the classical Point Distribution Model (PDM, see Section 4.2 for its description). Since the PDM is a discrete method and is only based on linear interpolation between landmarks we have computed two corresponding *mean* shapes; one with a low resolution (LR) that corresponds to the number of landmarks used for the two continuous-domain models and a second with a high resolution (HR), where we have increased the number of samples by a factor of 50, *i.e.*, 50 samples between each original landmark (see Figure 4.2, middle row).

For each of the three models we have computed the normalized correlation $\frac{\langle \boldsymbol{r}_{\text{model}}, \boldsymbol{r}_{\text{data}} \rangle}{\|\boldsymbol{r}_{\text{model}}\|_{L_2} \|\boldsymbol{r}_{\text{data}}\|_{L_2}}$ between the most representative shape $\boldsymbol{r}_{\text{model}}$ obtained with the corresponding method and each curve $\boldsymbol{r}_{\text{data}}$ in the dataset. Here, $\boldsymbol{r}_{\text{model}}$ stands for either 1) $\boldsymbol{r}_{\text{mean}}$, 2) the mean shape obtained with the PDM or 3) the first eigenshape ("fPCA1") obtained with the fPCA. The results are shown in Table 4.1. We see that our method to compute the *learned* shape $\boldsymbol{r}_{\text{mean}}$ as the curve being closest to all the subspaces generated by the shapes of the dataset, captures best the shape variability. Further, the continuous domain methods seem to yield higher accuracy than the PDM. However, by increasing the resolution of the PDM we can approach the accuracy of continuous domain models according to the theoretical argument provided in Section 4.8.2.

Shape Reconstruction: Projection-based Functional PCA vs. Point Distribution Model

We reconstruct the shapes of the dataset by using our proposed projection-based fPCA and compare it to the shape approximation procedure given by the PDM. From (4.8), we see that fPCA allows for a perfect shape reconstruction if all the eigencurves are used. However, here we approximate the data as $\boldsymbol{r}_{\text{data}}(t) \approx \boldsymbol{r}_{\text{recon}}^{\text{fPCA}}(t) = \sum_{i=1}^{4} a_i \boldsymbol{z}_i^{\text{fPCA}}(t)$ using the first four eigenvectors of the fPCA. The $a_i \in \mathbb{R}$ are the coefficients that allow for the optimal approximation. For comparison, we compute the approximation obtained with the high resolution PDM model, also using the first four eigenvectors. The PDM model is expressed as $\boldsymbol{r}_{\text{data}} \approx \boldsymbol{r}_{\text{recon}}^{PDM} = \overline{\boldsymbol{r}} + \sum_{i=1}^{4} b_i \boldsymbol{z}_i^{\text{PDM}}(t)$ with $b_i \in \mathbb{R}$ being the optimal approximation coefficients and $\overline{\boldsymbol{r}}$ the mean shape computed with the PDM. Since the PDM is a discrete model, we can obtain an equivalent continuous-domain representation by interpolating the landmarks with the uniform linear B-spline. This allows us to compute and compare the L_2 reconstruction errors as reported in Table 4.2. Again, the results suggest that the continuous domain model, *i.e.*, the functional PCA yields higher accuracy and thus, captures shape variability more efficiently than the PDM. The reconstructed shapes are shown in Figure 4.3.

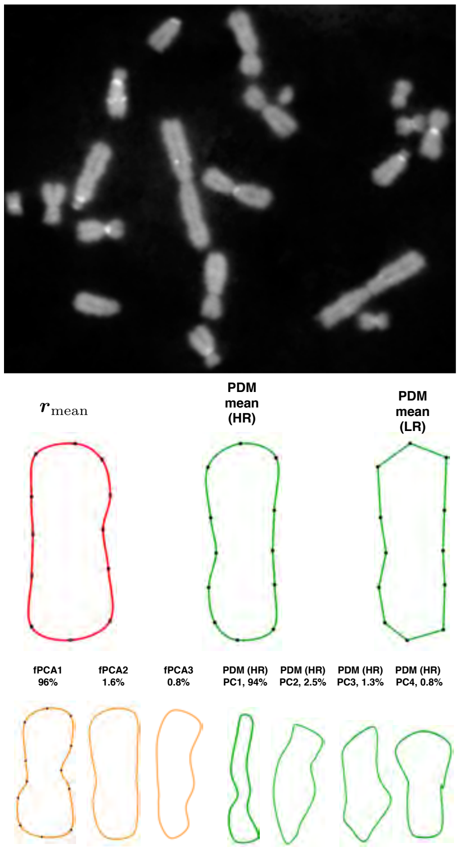


Figure 4.2: Shape analysis of chromosome data. The data set that consists of 20 chromosomes is shown in the top. The chromosomes have been manually outlined by placing landmarks on the contours followed by spline interpolation. Theorem 4 yields $\boldsymbol{r}_{\text{mean}}$; the red shape (middle row, curve on the left). The orange curve, "fPCA1" (bottom row) corresponds to the first eigenshape obtained through fPCA. It captures 96% of the shape variability found in the data set, computed as $\lambda_1 / \sum \lambda_i$, where λ_i denotes the eigenvalues obtained through fPCA for the i-th eigenshape. The two green shapes in the middle row represent the *mean* shapes obtained with the PDM with high (HR) and low (LR) resolution. The bottom row shows the result of the eigenanalysis w.r.t. fPCA (orange curves) as well as the first 4 eigenshapes obtained with the high resolution (HR) PDM. Again, the percentage numbers indicated the shape variability captured by each eigenshape as given by the corresponding eigenvalues.

Table 4.1: Normalized correlation between principal shapes and chromosome data.

Data	$\boldsymbol{r}_{\text{mean}}$	fPCA	PDM (HR)	PDM (LR)
1	**0.958**	0.955	0.954	0.844
2	0.986	**0.988**	0.955	0.845
3	**0.986**	0.972	0.956	0.845
4	**0.984**	0.962	0.955	0.845
5	0.970	**0.974**	0.953	0.840
6	**0.982**	0.974	0.954	0.843
7	**0.995**	0.985	0.957	0.847
8	**0.987**	0.983	0.955	0.843
9	**0.985**	0.984	0.956	0.843
10	0.985	**0.986**	0.954	0.844
11	**0.981**	0.974	0.953	0.842
12	**0.960**	0.946	0.954	0.844
13	**0.973**	0.969	0.952	0.839
14	**0.965**	0.948	0.953	0.841
15	**0.981**	0.980	0.957	0.846
16	0.973	**0.983**	0.954	0.842
17	**0.977**	0.955	0.956	0.844
18	0.986	**0.987**	0.957	0.846
19	**0.996**	0.989	0.958	0.847
20	**0.994**	0.984	0.958	0.847
Mean	**0.980**	0.974	0.955	0.844
STD	0.010	0.013	0.001	0.002

Table 4.2: Reconstruction error $\|\boldsymbol{r}_{\text{data}} - \boldsymbol{r}_{\text{recon}}\|^2_{L_2} / (\|\boldsymbol{r}_{\text{data}}\|_{L_2} \|\boldsymbol{r}_{\text{recon}}\|_{L_2})$ for chromosome data.

Data	fPCA (4)	PDM (4,HR)
1	**0.016**	0.085
2	**0.004**	0.047
3	**0.004**	0.036
4	**0.009**	0.036
5	**0.013**	0.072
6	**0.007**	0.052
7	**0.001**	0.022
8	**0.005**	0.042
9	**0.002**	0.050
10	**0.002**	0.050
11	**0.002**	0.057
12	**0.016**	0.075
13	**0.012**	0.064
14	**0.014**	0.065
15	**0.005**	0.048
16	**0.012**	0.075
17	**0.015**	0.047
18	**0.006**	0.042
19	**0.003**	0.021
20	**0.002**	0.019
Mean	**0.008**	0.050
STD	0.005	0.018

4.9.2 Shape Classification

If different “groups” of shapes are compared with each other, then the learned shape as described in Section 4.4 can be used to compute and compare principal shapes that represent the different groups. In a standard shape-classification setting, the computation of the mean shape $\boldsymbol{r}_{\text{mean}}$ can be viewed as a “trained” shape, where the curves used to compute this shape constitute the training set.

Classification of Shapes in Medical Imaging

This experiment is part of a clinical study where the structural and potential functional changes of the pelvic floor hiatus (PFH) are examined after a woman has given birth to one or several children [120]. 3D ultrasound volumes of 245 women were acquired and grouped into 61 *nulliparae* (women who did not give birth to children) and 184 *multiparae* (women who gave birth to one or several children). For both groups, images were acquired when the women were “at rest” and while “contracting” the PFH. The PFH is outlined on a specific 2D section of the ultrasound volume using the following procedure: A clinician draws key points on the image which have particular anatomical meaning. Curves are then computed by interpolating the ordered set of keypoints using spline interpolators [76, 77], as shown in Figure 4.4 (top row).

The interpolation property of a basis function is particularly useful in user-interactive applications. Aside from comparing surface area and perimeter of the closed curves, a qualitative analysis w.r.t. shape differences is performed. This comparison between different patient groups is important to clinicians because it reveals similarities (or differences) while at the same time removing within-group variability that is “absorbed” by the linear transform that is used. In this case, we constructed spline-based affine vector spaces using the affine transformation of the spline curves combined with translation (*i.e.*, vector space of dimension six). The mean shapes are computed for the four subgroups (*nulliparae* and *multiparae*, “at rest” or “contraction”). They are shown in the bottom row of Figure 4.4 and strongly indicate that the shape of the PFH probably does not change after giving birth to one or several children although its size, perimeter, and surface do [121].

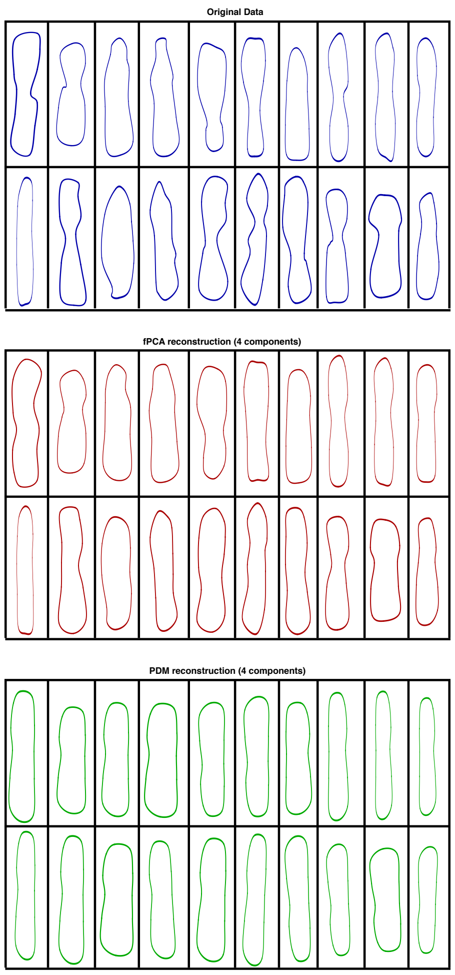


Figure 4.3: Reconstruction of chromosome data set. In the top (first and second row) the original data is shown (blue). The orange-colored shapes (3rd and 4th row) represent the shapes that have been reconstructed using our proposed projection-based functional PCA. The green shapes (last two rows) correspond to the reconstruction using the PDM.

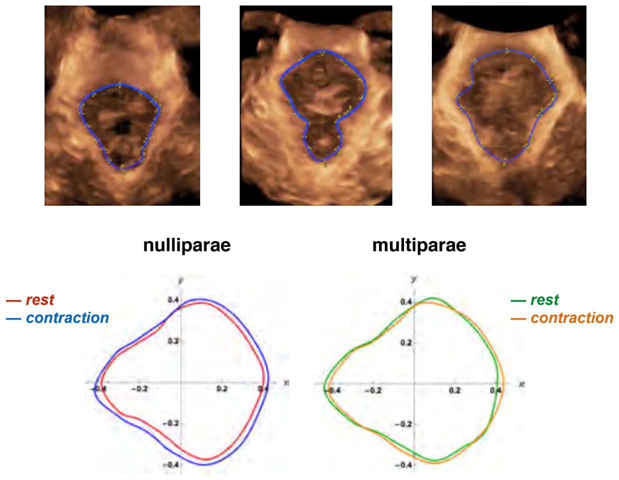


Figure 4.4: Top row: Examples of 3D ultrasound volumetric data. The top-left image shows the PFH area of a patient at contraction, whereas the middle and right images show two different patients' PFH area at rest. The blue curves represent the outline of the PFH that has been constructed by spline interpolation of an ordered set of points drawn by a clinician on the image. Bottom row: The comparison of nulliparae vs. mutliparae women reveals that there is no qualitative difference in the shapes $\boldsymbol{r}_{\text{mean}}^{\text{nulliparae}}$ and $\boldsymbol{r}_{\text{mean}}^{\text{mutliparae}}$ between the two groups (although the sizes are different), independently from the state ("at rest" or "contraction") of the PFH area. (Image courtesy Dr. med. Sylvain Meyer, Urogynaecology Unit and Obstetrics Department, CHUV Lausanne, EHC, Morges, Switzerland.)

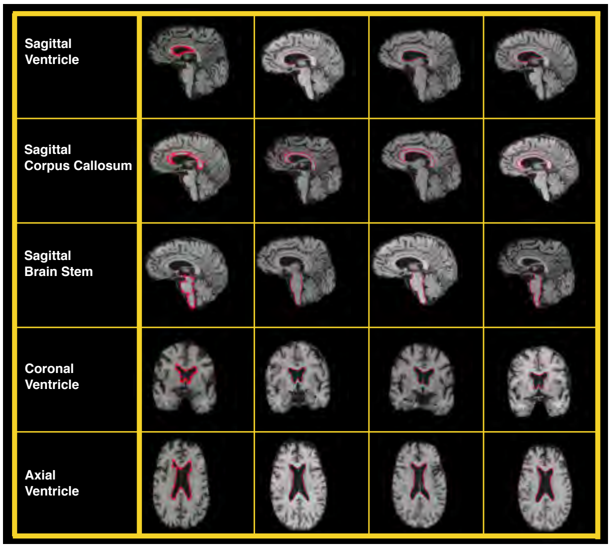


Figure 4.5: Shape library representing different brain structures. Each row represents a shape *type*. In order to illustrate the shape variability that occurs within a group of the same type, four samples per *type* are shown.

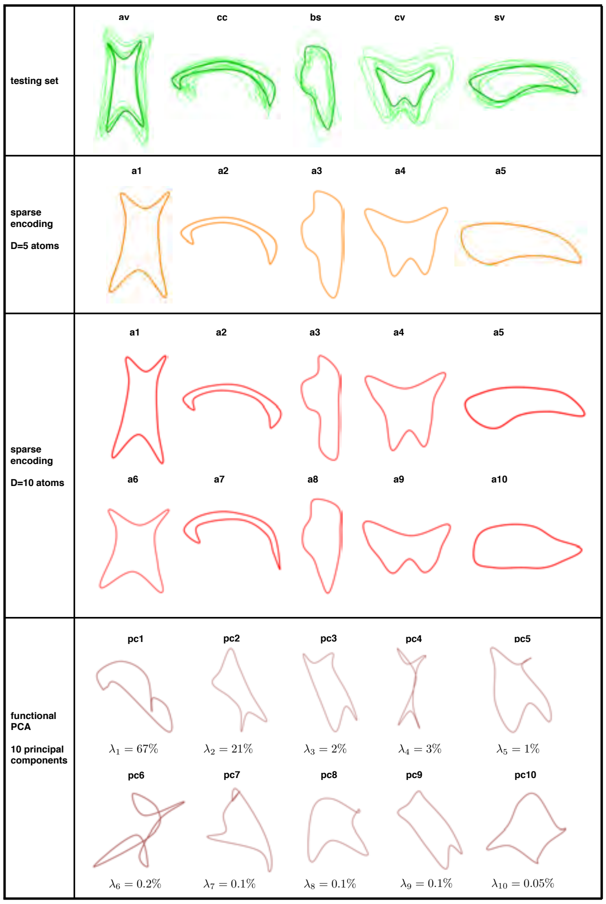


Figure 4.6: Atoms of the *learned* shape dictionaries. Some samples of the *training* set are shown (green, top row) together with the *atoms*, a_i of both dictionaries, D5 (orange, second row) and D10 (red, third and fourth row). The last two rows (purple) correspond to the principal components (pc) obtained with the projection-based fPCA. The values of the λ_i are computed as described in the caption of Figure 4.2

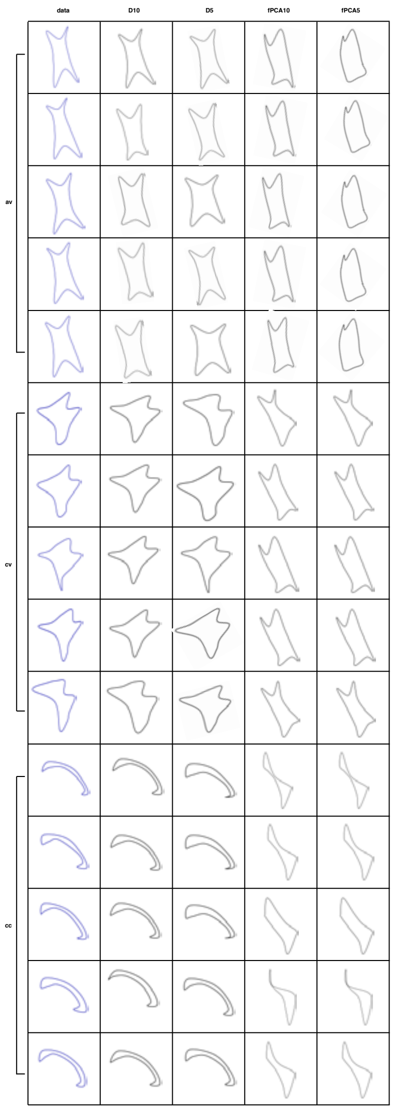


Figure 4.7: Reconstructed *testing* set. The first column shows the *testing* set, whereas the second (D10) and third (D5) columns show the reconstruction with the sparse methods. The last two columns correspond to the reconstruction using fPCA.

Table 4.3: Correlation between the best atom of the learned dictionary and the testing set (rounded values).

		test 1	test 2	test 3	test 4	test 5	Mean	STD
SV	D5	0.99	**0.98**	**0.98**	**0.98**	**0.99**	**0.98**	0.00
	D10	**0.99**	0.97	0.97	0.97	0.98	0.98	0.01
	fPCA	0.85	0.86	0.88	0.89	0.87	0.87	0.01
CC	D5	**0.99**	0.99	0.98	0.99	**0.99**	0.99	0.00
	D10	0.99	**0.99**	**0.99**	**0.99**	0.98	**0.99**	0.00
	fPCA	0.80	0.78	0.84	0.81	0.76	0.80	0.02
BS	D5	0.99	**0.97**	**0.99**	**0.98**	**0.99**	**0.98**	0.01
	D10	**0.99**	0.96	0.99	0.98	0.99	0.98	0.01
	fPCA	0.90	0.89	0.91	0.92	0.90	0.90	0.01
CV	D5	0.98	0.98	**0.99**	**0.99**	0.99	0.99	0.00
	D10	**0.99**	**0.98**	0.99	0.99	**0.99**	**0.99**	0.00
	fPCA	0.94	0.97	0.95	0.95	0.95	0.95	0.01
AV	D5	0.99	0.99	**0.99**	**0.99**	**0.99**	**0.99**	0.00
	D10	**0.99**	**0.99**	0.99	0.99	0.99	0.99	0.00
	fPCA	0.98	0.97	0.97	0.97	0.98	0.97	0.00

4.9.3 Sparse Dictionary Learning in Medical Imaging

We are interested in the construction of a shape dictionary which encodes curves that represent several *types* of shapes. We have constructed a *training* set that contains 150 outlines of brain structures, each representing one among the following five different *types* of shapes: *Sagittal Ventricle (sv), Sagittal Corpus Callosum (cc), Sagittal Brain Stem (bs), Coronal Ventricle (cv), Axial Ventricle (av).* Samples of each brain structure are shown in Figure 4.5. The data set consists of 30 samples per brain structure. Within each group represented by a particular type we have correspondence between landmarks. However, between shapes that belong to different groups, such a correspondence is no longer guaranteed. Furthermore, the types that represent the *Coronal Ventricle* and the *Axial Ventricle* appear to be similar up to a (anisotropic) scaling factor and a rotation (see Figure 4.5). Hence, the data set can also be considered as *imbalanced* besides being *inhomogeneous.* It is well known that L_2-based methods are error-prone when dealing with imbalance or inhomogeneity in a data set. Thus, sparse or l_1-based methods tend to be more efficient in such cases. We have applied our method to learn a dictionary for sparse shape encoding, where we compute a dictionary (D5) with only 5 atoms and a second one with 10 atoms (D10). Since the method allows for sparse shape encoding, we expect to find a dicitonary D5, where each atom resembles one of the five shape types. The atoms of the two dictionaries, D5 and D10, are shown in Figure 4.6. The regularization parameter λ has been chosen empirically. As a control experiment we also perform a (L_2-based) functional PCA and use it to construct a dictionary that consists of the first 10 eigencurves.

To validate our method we have built a *testing* set which consists in 25 shapes which all differ from the shapes of the training set. Each group represented by a shape type contains 5 samples (denoted as “test 1”, “test 2”, . . . , “test 5”). In a first step, we compute the correlation between the “best” atom, *i.e.*, the most similar atom of the dictionary (for the three dictionaries D5, D10, and fPCA) and each of the 25 samples. The results are summarized in Table 4.3. It becomes apparent that the L_2-method fails when dealing with inhomogeneous data as expected. The accuracy of the D5 and the D10 dictionary is similar and both qualitatively (Figure 4.6) and quantitatively high.

In a second step we reconstruct 15 shapes among the testing set corresponding to three different types; 5 axial ventricles, 5 coronal ventricles, and 5 corpi callosi. We use the learned dictionaries and the corresponding sparse codes for the reconstruction process. We compute the reconstruction error $\|\boldsymbol{r}_{\text{data}} - \boldsymbol{r}_{\text{recon}}\|_{L_2}^2$ and also compare it to the "pure" L_2 method, *i.e.*, the projection-based fPCA, where we use 5 as well as 10 eigenshapes for the approximation. The reconstructed testing set is shown in Figure 4.7 (appended to this article) and the errors are listed in Table 4.4. We notice that the reconstruction with D10 tends to yield more accurate results as with D5, which is expected. Again, the projection-based fPCA fails to yield satisfying results.

4.10 Summary

We have presented a unified framework for dictionary learning in the continuous domain for shape encoding where the data consists of landmark-based parametric curves. We provide closed-form solutions for the unbiased alignment of the training data and show how shapes and dictionaries are learned for different types of applications such as the characterization of homogeneous, inhomogeneous or imbalanced data. The alignment is based on a new method to compute mean shapes, which can also be used to construct shape priors for further use in segmentation problems. We derive formulas for an exact and fast implementation of the proposed framework using spline curves. Our examples and validation experiments highlight the advantages of our model compared to state-of-the-art discrete frameworks. Furthermore, our model can be easily extended to 3D parametric curves that are defined by landmarks.

Table 4.4: Reconstruction error $\|\boldsymbol{r}_{\text{data}} - \boldsymbol{r}_{\text{recon}}\|^2_{L_2} / (\|\boldsymbol{r}_{\text{data}}\|_{L_2} \|\boldsymbol{r}_{\text{recon}}\|_{L_2})$.

data	D10	D5	fPCA5	fPCA10
av1	**0.007**	0.063	0.223	0.777
av2	**0.013**	0.054	0.213743	0.65489
av3	0.0195	**0.004**	0.241	0.711
av4	**0.006**	0.127	0.196	0.558
av5	**0.008**	0.029	0.259	0.755
av mean	**0.010**	0.056	0.226	0.691
av std	0.005	0.046	0.024	0.087
cv1	**0.008**	0.065	0.218	0.218
cv2	0.010	**0.003**	0.205	0.205
cv3	0.022	**0.002**	0.214	0.214
cv4	**0.009**	0.011	0.191	0.191
cv5	**0.009**	0.057	0.447	0.447
cv mean	**0.011**	0.028	0.255	0.255
cv std	0.005	0.030	0.107	0.107
cc1	0.005	**0.003**	0.664	0.664
cc2	0.010	**0.004**	0.705	0.705
cc3	**0.009**	0.035	0.736	0.736
cc4	**0.012**	0.024	0.365	0.365
cc5	**0.008**	0.021	0.637	0.637
cc mean	**0.009**	0.018	0.621	0.621
cc std	0.002	0.013	0.148	0.148

4.11 Appendix

4.11.1 Proof of Theorem 2

Proof. The eigenequation (4.2) is developed as

$$\begin{aligned}\sum_k \mathcal{P}_k \boldsymbol{\phi}(t) &= \langle \sum_{k=1}^{K} \boldsymbol{K}_{\mathcal{P}_k}(t,\cdot), \boldsymbol{\phi}\rangle \\ &= \langle \sum_{k=1}^{K}\sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)} \otimes \tilde{\boldsymbol{e}}_i^{(k)}(\cdot), \boldsymbol{\phi}\rangle = \lambda \boldsymbol{\phi}(t).\end{aligned}$$

We identify

$$\begin{aligned}\boldsymbol{\phi}(t) &= \frac{1}{\lambda}\langle \sum_{k=1}^{K}\sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)} \otimes \tilde{\boldsymbol{e}}_i^{(k)}(\cdot), \boldsymbol{\phi}\rangle \\ &= \frac{1}{\lambda}\sum_{k=1}^{K}\sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)} \langle \tilde{\boldsymbol{e}}_i^{(k)}, \boldsymbol{\phi}\rangle = \sum_{k=1}^{K}\sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)} \gamma_{ik},\end{aligned}$$

where $\gamma_{ik} = \frac{\langle \tilde{\boldsymbol{e}}_i^{(k)}, \boldsymbol{\phi}\rangle}{\lambda}$. Hence,

$$\lambda\gamma_{ik} = \langle \tilde{\boldsymbol{e}}_i^{(k)}, \sum_{l=1}^{K}\sum_{j=1}^{I} \boldsymbol{e}_j(\cdot)^{(l)}\gamma_{jl}\rangle = \sum_{l=1}^{K}\sum_{j=1}^{I} \gamma_{jl}\langle \tilde{\boldsymbol{e}}_i^{(k)}, \boldsymbol{e}_j^{(l)}\rangle.$$

This leads to the eigenvalue problem given by (4.29) whose solution is $\boldsymbol{\phi}(t) = \sum_{k=1}^{K}\sum_{i=1}^{I} \boldsymbol{e}_i(t)^{(k)}\gamma_{ik}$. □

$$
K \times I \left\{ \underbrace{\overbrace{\begin{bmatrix}
\langle \tilde{\boldsymbol{e}}_1^{(1)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_1^{(1)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_1^{(1)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_1^{(1)}, \boldsymbol{e}_I^{(K)} \rangle \\
\langle \tilde{\boldsymbol{e}}_2^{(1)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_2^{(1)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_2^{(1)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_2^{(1)}, \boldsymbol{e}_I^{(K)} \rangle \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots\vdots & \ddots & \vdots \\
\langle \tilde{\boldsymbol{e}}_I^{(1)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_I^{(1)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_I^{(1)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_I^{(1)}, \boldsymbol{e}_I^{(K)} \rangle \\
\langle \tilde{\boldsymbol{e}}_1^{(2)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_1^{(2)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_1^{(2)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_1^{(2)}, \boldsymbol{e}_I^{(K)} \rangle \\
\langle \tilde{\boldsymbol{e}}_2^{(2)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_2^{(2)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_2^{(2)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_2^{(2)}, \boldsymbol{e}_I^{(K)} \rangle \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots\vdots & \ddots & \vdots \\
\langle \tilde{\boldsymbol{e}}_I^{(2)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_I^{(2)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_I^{(2)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_I^{(2)}, \boldsymbol{e}_I^{(K)} \rangle \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots\vdots & \ddots & \vdots \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots\vdots & \ddots & \vdots \\
\langle \tilde{\boldsymbol{e}}_1^{(K)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_1^{(K)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_1^{(K)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_1^{(K)}, \boldsymbol{e}_I^{(K)} \rangle \\
\langle \tilde{\boldsymbol{e}}_2^{(K)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_2^{(K)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_2^{(K)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_2^{(K)}, \boldsymbol{e}_I^{(K)} \rangle \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots\vdots & \ddots & \vdots \\
\langle \tilde{\boldsymbol{e}}_I^{(K)}, \boldsymbol{e}_1^{(1)} \rangle & \dots & \langle \tilde{\boldsymbol{e}}_I^{(K)}, \boldsymbol{e}_1^{(K)} \rangle & \langle \tilde{\boldsymbol{e}}_I^{(K)}, \boldsymbol{e}_2^{(1)} \rangle & \dots & & \dots & \langle \tilde{\boldsymbol{e}}_I^{(K)}, \boldsymbol{e}_I^{(K)} \rangle
\end{bmatrix}}^{K \times I}}_{\boldsymbol{\Gamma}} \right.
\underbrace{\begin{bmatrix} \gamma_{1,1} \\ \gamma_{1,2} \\ \vdots \\ \gamma_{1,K} \\ \gamma_{2,1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \gamma_{I,K} \end{bmatrix}}_{\boldsymbol{\gamma}}
= \lambda
\underbrace{\begin{bmatrix} \gamma_{1,1} \\ \gamma_{1,2} \\ \vdots \\ \gamma_{1,K} \\ \gamma_{2,1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \gamma_{I,K} \end{bmatrix}}_{\boldsymbol{\gamma}}
\tag{4.29}
$$

4.11.2 Vector Space Including a Translation

If in the construction of the K projectors a basis that includes a translation $\boldsymbol{b}$ given by $\{\boldsymbol{e}_{b_x}, \boldsymbol{e}_{b_y}\} = \{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\}$ is used, then both $\boldsymbol{e}_{b_x}$ and $\boldsymbol{e}_{b_y}$ are eigencurves and hence, solutions of the eigenequation (4.2) with eigenvalue equal to K. This is easy to see, since for such a projector, $\mathcal{P}\boldsymbol{e}_{b_x} = 1 \cdot \boldsymbol{e}_{b_x}$ and, therefore, $\sum_{k=1}^{K} \mathcal{P}_k \boldsymbol{e}_{b_x} = K \cdot \boldsymbol{e}_{b_x}$. The same result holds true for $\boldsymbol{e}_{by}$. In this case, $\boldsymbol{r}_{\text{mean}}$ is chosen to be the third eigencurve, since the first two are constants, *i.e.*, 2D points.

4.11.3 Proof of Theorem 4

Using (2.13) we develop

$$\begin{aligned}&\sum_{k=1}^{K} \langle \mathcal{P}_k \boldsymbol{r}, \boldsymbol{r} \rangle = \lambda \langle \boldsymbol{r}, \boldsymbol{r} \rangle \Leftrightarrow \sum_{k=1}^{K} \mathbf{c}^{\mathrm{T}} \mathbf{P}_k^{\mathrm{T}} \boldsymbol{\Psi} \mathbf{c} = \lambda \mathbf{c}^{\mathrm{T}} \boldsymbol{\Psi} \mathbf{c} \\ &\Leftrightarrow \sum_{k=1}^{K} \mathbf{P}_k^{\mathrm{T}} \boldsymbol{\Psi} \mathbf{c} = \lambda \boldsymbol{\Psi} \mathbf{c} \Leftrightarrow \boldsymbol{\Psi}^{-1} \sum_{k=1}^{K} \mathbf{P}_k^{\mathrm{T}} \boldsymbol{\Psi} \mathbf{c} = \lambda \mathbf{c},\end{aligned} \tag{4.30}$$

where $\mathbf{c}$ is the vector of control points of $\boldsymbol{r}$. Maximizing (4.30) w.r.t. $\mathbf{c}$ and using the expression provided by Theorem 1 for the spline projector, (4.30) boils down to the eigenvalue problem $\sum_{k=1}^{K} \mathbf{C}_k (\mathbf{C}_k \boldsymbol{\Psi} \mathbf{C}_k^{\mathrm{T}})^{-1} \mathbf{C}_k^{\mathrm{T}} \boldsymbol{\Psi} \mathbf{c} = \lambda \mathbf{c}$. ■

Chapter 5

Closed-form Alignment of Active Surface Models Using Splines

Overview

In this chapter [1], we propose a new formulation of the active surface model in 3D. Thereby, we use the surface projectors constructed in Section 3.7.3 and extend our formulation of the functional PCA (see Chapter 4) to 3D surfaces. Instead of aligning a shape dictionary through the similarity transform as is the case for the classical active shape model, we consider more flexible affine transformations and use our alignment method that is unbiased in the sense that it implicitly constructs a common reference shape. Our formulation is expressed in the continuous domain and we provide an algorithm to exactly implement the framework using spline-based parametric surfaces. We test our model on real 3D MRI data. A comparison with the classical active shape model shows that our method allows us to capture shape variability in a dictionary in a more precise way.

[1] The chapter is based on our publication [75]

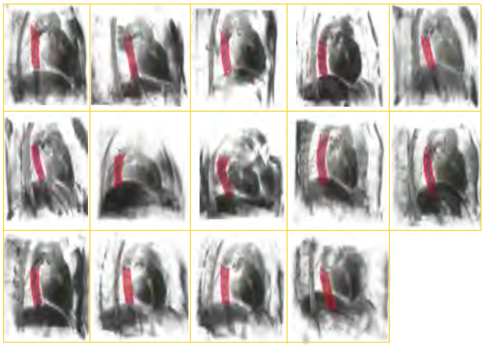


Figure 5.1: Real data set consisting of 14 3D MRI scans. The red meshes outline the segmented descending thoracic aorta.

5.1 Introduction

The classical active shape model (ASM) [71, 73] is a popular method to align discrete 2D curves given by an ordered set of points. It is typically used to characterize shape variability and to construct shape dictionaries. Thereby, the curves of the dictionary $\{\boldsymbol{r}_i\}$ are first aligned to a reference shape $\overline{\boldsymbol{r}}$ by optimally rotating and translating each $\boldsymbol{r}_i$ w.r.t. $\overline{\boldsymbol{r}}$, which amounts to removing a similarity transform. Afterwards, a principal component analysis (PCA) is computed with the aligned curves in order to statistically analyze the data set. The computation of the reference shape is done iteratively: first, all the shapes are aligned to the first shape of the data set, then the mean shape is computed, and, in a third step, all the aligned shapes are realigned again with the mean shape. This process is repeated until convergence. Variants exist to compute the reference shape, the main challenge being to reduce the bias of the model that is caused by the choice of an initial reference. A shape in the dictionary is then approximated as $\boldsymbol{r}_i \approx \mathbf{V}\boldsymbol{w}$, where $\mathbf{V}$ is the matrix containing a subset of eigenvectors computed through the PCA and $\boldsymbol{w}$ is a corresponding vector of weights.

The same methodology is also applicable to the analysis of 3D shapes [122, 123]. Such active shape models are widely used to characterize structures in medical images [124] as well as for biomedical image segmentation [125]. There exist also variants of the technique that make use of non-linear algorithms [126].

In this chapter, we propose a novel 3D active surface model as an extension of the classical ASM. Instead of applying a similarity transform to normalize the data, we consider a more general and flexible affine transformation while formulating the surface registration problem in the continuous domain. Our method is unbiased because it does not require to "choose" a reference shape to start the alignment process. We further propose an implementation using spline surfaces; this has the advantage that a shape is fully specified in the continuous domain by a discrete set of control points. Hence, no discretization of the surfaces is needed. We then express the PCA in the continuous domain as a functional surface PCA and use splines to derive a closed-form solution, which lends itself to a direct implementation. We have applied our method to real data in order to characterize a set of segmented descending thoracic aortas in 3D MRI (Figure 5.1). The experimental comparison with the classical ASM suggests that our model captures shape variability more accurately.

The theoretical concepts presented in this chapter are based on extending the theory presented in Chapter 4. Therefore, we focus on the presentation of results and we do not derive all the corresponding proofs; they can easily be obtained by adapting the proofs presented in Chapter 4 to the case of surfaces.

5.2 Unbiased Alignment

Prior to performing the functional surface PCA, the surfaces need to be aligned. This corresponds to the centering of the data vectors in the classical (discrete) PCA. To guarantee an unbiased alignment of the surfaces, we first specify the subspace S_i that is spanned by all the affine transformations (including translation) of any given surface $\boldsymbol{\sigma}_i$ within our initial shape dictionary $\{\boldsymbol{\sigma}_i\}$ (Figure 5.2). Then we compute the surface $\boldsymbol{\sigma}_{\text{ref}}$ which is closest to all subspaces S_i and project it back onto S_i to obtain the dictionary which contains the aligned surfaces $\{\tilde{\boldsymbol{\sigma}}_i\}$ (Figure 5.3). Instead of *explicitly* characterizing the space of affine transformations as the collection of all parametric surfaces $\boldsymbol{\sigma} \in \mathbb{R}^3$, $\{\boldsymbol{\sigma} | \mathbf{A}\boldsymbol{\sigma}(u,v) + \boldsymbol{b}, u, v \in \mathbb{R}, \mathbf{A} \in \mathbb{R}^{3\times 3}, \boldsymbol{b} \in \mathbb{R}^3\}$, we *implicitly* characterize the affine spaces by the orthogonal projection onto them. A basis to construct a projector onto an affine shape space of a surface $\boldsymbol{\sigma}$ is given by

$$\begin{aligned}&\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_{12}\}\\ &= \{ \begin{pmatrix}\boldsymbol{\sigma}_x\\0\\0\end{pmatrix}, \begin{pmatrix}\boldsymbol{\sigma}_y\\0\\0\end{pmatrix}, \begin{pmatrix}\boldsymbol{\sigma}_z\\0\\0\end{pmatrix}, \begin{pmatrix}0\\\boldsymbol{\sigma}_x\\0\end{pmatrix}, \begin{pmatrix}0\\\boldsymbol{\sigma}_y\\0\end{pmatrix}, \begin{pmatrix}0\\\boldsymbol{\sigma}_z\\0\end{pmatrix},\\ &\begin{pmatrix}0\\0\\\boldsymbol{\sigma}_x\end{pmatrix}, \begin{pmatrix}0\\0\\\boldsymbol{\sigma}_y\end{pmatrix}, \begin{pmatrix}0\\0\\\boldsymbol{\sigma}_z\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix} \}.\end{aligned} \tag{5.1}$$

The orthogonal projection of $\boldsymbol{\sigma}$ onto S_i is expressed as

$$\mathcal{P}_i\boldsymbol{\sigma}(u,v) = \sum_{k=1}^{K} \boldsymbol{e}_k(u,v)\langle\tilde{\boldsymbol{e}_k}, \boldsymbol{\sigma}\rangle, \tag{5.2}$$

where $\{\tilde{\boldsymbol{e}_k}\}$ is the dual basis of $\{\boldsymbol{e}_k\}$ such that $\langle\boldsymbol{e}_k, \tilde{\boldsymbol{e}_l}\rangle = \delta_{k-l}$, where δ_{k-l} denotes the Kronecker Delta. Hence, the projector $\mathcal{P}_i$ in (5.2) orthogonally projects an arbitrary *query* shape $\boldsymbol{\sigma}$ onto the affine space given by a surface $\boldsymbol{\sigma}_i$ (Figure 5.3).

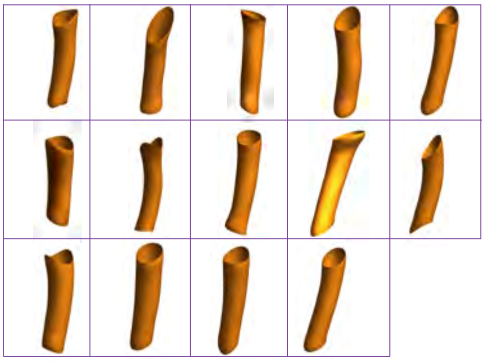


Figure 5.2: 3D Renderings of the segmented surfaces shown in Figure 5.1. The orange surfaces correspond to the dictionary $\{\boldsymbol{\sigma}_i\}$.

5.2.1 Aligned Shape Dictionary

To the shape dictionary $\{\boldsymbol{\sigma}_i\}$ we can associate the dictionary $\{\mathcal{P}_i\boldsymbol{\sigma}\}$, where $\boldsymbol{\sigma}$ is an arbitrary *query* surface and whose elements are the best fitting affine transformations of the $\boldsymbol{\sigma}_i$.

Definition 4. We define the data array $\mathbf{X} \in L_2(\mathbb{R}^2, \mathbb{R}^3) \times \mathbb{R}^I$ which takes a query surface $\boldsymbol{\sigma}$ and outputs I projections as

$$\begin{aligned}\mathbf{X} &= [\mathcal{P}_1\boldsymbol{\sigma}(u,v) \cdots \mathcal{P}_I\boldsymbol{\sigma}(u,v)]\\ &= [\tilde{\boldsymbol{\sigma}}_1(u,v) \cdots \tilde{\boldsymbol{\sigma}}_I(u,v)],\end{aligned}$$

where $\mathcal{P}_i$ is the projector onto the subspace S_i, $\boldsymbol{\sigma}$ is a parametric surface and $\tilde{\boldsymbol{\sigma}}_i = \mathcal{P}_i\boldsymbol{\sigma}$.

Next, we define $\boldsymbol{\sigma}_{\text{ref}}$, which is closest to all subspaces S_i.

Definition 5. The optimal surface $\boldsymbol{\sigma}_{\text{ref}}$ that is closest to all subspaces S_i in the L_2-sense is obtained by maximizing the Frobenius norm of $\mathbf{X}$ subject to $\|\boldsymbol{\sigma}\| = 1$, *i.e.*,

$$\operatorname*{arg\,min}_{\mathbf{A}_i, \boldsymbol{b}_i, \boldsymbol{\sigma}_{\text{ref}}} \sum_{i=1}^{I} \|\boldsymbol{\sigma}_{\text{ref}} - \mathbf{A}_i\boldsymbol{\sigma}_i - \boldsymbol{b}_i\|_{L_2}^2 = \operatorname*{arg\,max}_{\boldsymbol{\sigma}_{\text{ref}}} \sum_{i=1}^{I} \|\mathcal{P}_i\boldsymbol{\sigma}_{\text{ref}}\|_{L_2}^2$$

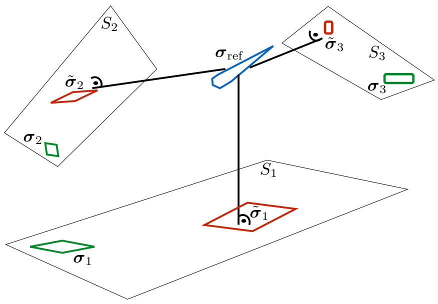
subject to $\|\boldsymbol{\sigma}_{\text{ref}}\| = 1$.

The exact computation of $\boldsymbol{\sigma}_{\text{ref}}$ is specified by Proposition 5.

Proposition 5. *The optimal surface $\boldsymbol{\sigma}_{\text{ref}}$ in the sense of Definition 5 is given as the solution of the eigenequation*

$$\sum_{i=1}^{I} \mathcal{P}_i\boldsymbol{\sigma}_{\text{ref}} = \lambda\boldsymbol{\sigma}_{\text{ref}}, \tag{5.3}$$

where $\lambda \in \mathbb{R}$ is the largest eigenvalue of the eigenequation (5.3)*.*



S_2
S_3
S_1
$\boldsymbol{\sigma}_{\text{ref}}$
$\tilde{\boldsymbol{\sigma}}_2$
$\tilde{\boldsymbol{\sigma}}_3$
$\tilde{\boldsymbol{\sigma}}_1$
$\boldsymbol{\sigma}_2$
$\boldsymbol{\sigma}_3$
$\boldsymbol{\sigma}_1$

Figure 5.3: Unbiased shape alignment. For each surface $\boldsymbol{\sigma}_i$ (green) in the dictionary $\{\boldsymbol{\sigma}_i\}$, the space S_i containing all its affine transformations is defined. The surface $\boldsymbol{\sigma}_{\text{ref}}$ (blue) that is closest to all spaces S_i is computed and projected back onto the S_i to obtain the dictionary of aligned shapes $\{\tilde{\boldsymbol{\sigma}}_i\}$ (red).

5.3 Application: Functional Surface PCA

We compute the aligned data array $\mathbf{X}$ specified by Definition 4, where we chose $\boldsymbol{\sigma} := \boldsymbol{\sigma}_{\text{ref}}$ as illustrated in Figures 5.3 and 5.4. Since we consider surfaces that are described in the continuous domain, we apply a functional (instead of a discrete) PCA to $\mathbf{X}$. For this purpose, we define $\mathbf{X}^* : L_2(\mathbb{R}^2, \mathbb{R}^3) \to \mathbb{R}^I$, which satisfies $\langle \boldsymbol{\sigma}, \mathbf{X}\boldsymbol{v} \rangle = \langle \mathbf{X}^*\boldsymbol{\sigma}, \boldsymbol{v} \rangle$, where $\boldsymbol{v} \in \mathbb{R}^I$. The eigensurfaces $\boldsymbol{\phi}_i \in L_2(\mathbb{R}^2, \mathbb{R}^3)$ of the *scatter operator* $\mathbf{XX}^* : L_2(\mathbb{R}^2, \mathbb{R}^3) \to L_2(\mathbb{R}^2, \mathbb{R}^3)$ are then specified by

$$\mathbf{XX}^*\{\boldsymbol{\phi}_i\} = \lambda_i \boldsymbol{\phi}_i,$$

where the λ_i are the non-zero eigenvalues of $\mathbf{X}^*\mathbf{X}$. The derivation of these results will be presented elsewhere.

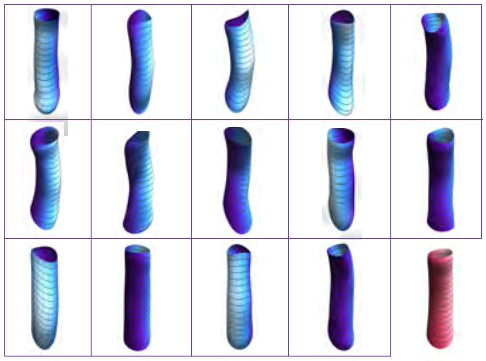


Figure 5.4: Aligned shape dictionary $\{\tilde{\boldsymbol{\sigma}}_i\}$. The pink surface in the bottom right corresponds to the reference shape, $\boldsymbol{\sigma}_{\mathrm{ref}}$, computed with Proposition 5.

5.4 Spline-based Implementation

We consider surfaces of the form

$$\boldsymbol{\sigma}(u,v) = \sum_k \sum_l \boldsymbol{c}[k,l]\varphi(u-k)\psi(v-l), \tag{5.4}$$

where the sums in (5.4) are finite and the total number, M, of *control points*, $\boldsymbol{c}[k,l] = (c_x[k,l], c_y[k,l], c_z[k,l]) \in \mathbb{R}^3$ is finite, and φ and ψ are spline basis functions that satisfy the Riesz basis condition [81]. For the explicit expressions of φ and ψ to construct a cylindrical surface, we refer the reader to [80]. Expression (5.4) allows us to fully represent a surface in the continuous domain with only few control points. Further, we can express the projector (5.2) w.r.t. to the spline control points as specified by Proposition 6.

Proposition 6. *The matrix* $\mathbf{P} \in (\mathbb{R}^{3M} \times \mathbb{R}^{3M}) : \mathbb{R}^{3M} \mapsto \mathbb{R}^{3M}$ *is a* $(3M \times 3M)$ *projection matrix defined as* $\mathbf{P} = \mathbf{C}\mathbf{G}^{-1}\mathbf{C}^{\mathrm{T}}\boldsymbol{\beta}$*, where* $\mathbf{C}$ *is a matrix containing the control points of a spline surface* (5.4)*,* $\mathbf{G}$ *is the Gram matrix w.r.t. the basis* (5.1) *describing the affine shape space of a surface* $\boldsymbol{\sigma}$*, and* $\boldsymbol{\beta}$ *is a sparse matrix containing* *the autocorrelation integrals of the integer-shifted basis functions φ and ψ as detailed in Section 3.7.3.*

It can be shown that if all the control points of an arbitrary surface $\boldsymbol{\sigma}$ are arranged in a vector $\mathbf{\Gamma}$, then the control points of the projected surface are computed as $\tilde{\boldsymbol{\sigma}}_i = \mathcal{P}_i\boldsymbol{\sigma} = \mathbf{P}_i\mathbf{\Gamma}$. A direct consequence of Proposition 6 is that the continuous-domain principal shapes can be computed through an exact but simple matrix-vector multiplication as given by Proposition 7.

Proposition 7. *The matrix that contains the control points of the principal shapes is computed by*

$$\mathbf{C}_{\mathbf{Z}} = \mathbf{C}\mathbf{V}, \tag{5.5}$$

where $\mathbf{V}$ *contains the eigenvectors of* $\mathbf{X}^*\mathbf{X} = \mathbf{C}^{\mathsf{T}}\boldsymbol{\beta}\mathbf{C}$.

5.5 Results and Conclusion

Since $\mathbf{V}$ is orthogonal, any shape in the dictionary can be approximated as $\mathbf{C} \approx \mathbf{C}_{\mathbf{Z}}\tilde{\mathbf{V}}$, where $\tilde{\mathbf{V}}$ only contains a subset of eigenvectors. We compare our method with the classical ASM, where the shapes in the dictionary are approximated as $\boldsymbol{\sigma}_i \approx \tilde{\mathbf{V}}^{\text{ASM}}\boldsymbol{w}$ and computed with the iterative algorithm described in the introduction, where a similarity transform is removed from the original surfaces and $\tilde{\mathbf{V}}^{\text{ASM}}$ is the subset of eigenvectors computed with the corresponding PCA. Our data set consists of 14 MRI scans where the descending thoracic aorta has been segmented (Figure 5.1). We measure the L_2-error between the aligned shapes and the first principal shape $\boldsymbol{\phi}_1$, for both the classical ASM and our method. The results in Table 5.1 show that our method captures shape variability with a higher accuracy than the classical method which is due to the fact that only removing a similarity in 3D is too restrictive to align the shapes in a precise way. In Figure 5.5, we show a comparison of the first five principal surfaces. In the case of the ASM, the higher modes do not capture the shape variability in an intuitive way and make it difficult to interpret their physical meaning. Furthermore, our method is formulated in the continuous domain, which allows us to describe the shapes in an exact way; thereby the spline-based implementation keeps the computational cost low compared to a prior discretization of the shapes as is the case for the classical ASM. As a next step,

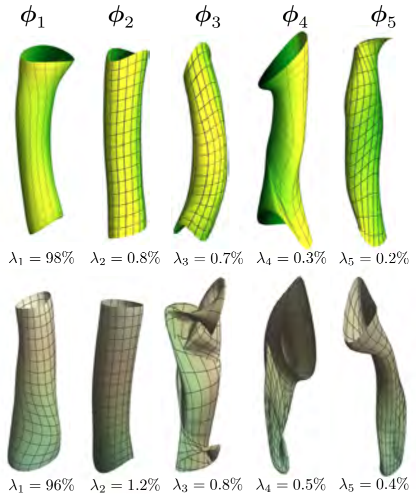


Figure 5.5: Principal surfaces computed with our proposed method (top) and the classical ASM model (bottom). The λ_i are the normalized eigenvalues. They describe the amount of information that the corresponding ϕ_i carries.

Table 5.1: L_2-error comparison

$\|\boldsymbol{\phi}_1 - \tilde{\boldsymbol{\sigma}}_i\|_{L_2}^2$	$\|\boldsymbol{\phi}_1^{\text{ASM}} - \tilde{\boldsymbol{\sigma}}_i^{\text{ASM}}\|_{L_2}^2$
0.0223	0.0388
0.0212	0.0395
0.0216	0.0392
0.0199	0.0395
0.0217	0.0393
0.0209	0.0392
0.0205	0.0391
0.0217	0.0387
0.0210	0.0389
0.0240	0.0383
0.0227	0.0391
0.0206	0.0396
0.0216	0.0396
0.0224	0.0393

we will use the presented results to include prior knowledge into 3D segmentation algorithms in order to increase robustness and speed in interactive settings.

Chapter 6

Characterization of Interpolators for Shape Modeling

Overview

In this chapter we present the construction of interpolators to be used in interactive applications. The chapter is based on our following three publications:

- In Section 6.1 we generalize the Keys interpolation kernel [127] to be able to represent trigonometric shapes such as ellipses and spheres [1].
- In Section 6.2 we present a family of interpolators [2], which is an extension of the work presented in Section 6.1
- Finally, in Section 6.3 we go one step further and construct families of interpolators that allow one to vary the resolution of the reconstructed shapes [3].

[1]Our related publication is [76].
[2]Our related publication is [77]
[3]Our related publication is [78]

6.1 Trigonometric Interpolation Kernel for Deformable Shapes

We present a new trigonometric generator function that is capable of perfectly reproducing circles, spheres and ellipsoids while at the same time being interpolatory. Such basis functions have the advantage that they allow to construct shapes through a sequence of control points that lie on their contour (2D) or surface (3D) which facilitates user-interaction, especially in 3D. Our piece-wise exponential basis function has finite support, which enables local control for shape modification. We derive and prove all the necessary properties of the kernel to represent shapes that can be smoothly deformed and show how idealized shapes such as ellipses and spheres can be constructed.

6.1.1 Introduction

Shape representation and deformation is an ongoing research topic in the fields where shapes need to be constructed, visualized, approximated or segmented. Related research domains include shape modeling for industrial design [128, 129, 130], or segmentation in biomedical imaging [131, 132, 13], such as the design of active contour models [83]. Applications involving such modeling are often user-interactive allowing a user to modify the shape by directly interacting with it. Desirable properties of such models are summarized as follows: 1) Intuitive user-interaction: the shape must be deformable by letting a user to directly interact with its boundary in a simple manner; *e.g.* dragging with a computer mouse the contour of a curve or surface. 2) Local deformation: through user-interaction the shape should only deform in the neighborhood where the interaction takes place. 3) Smooth deformation: a small perturbation of the shape must result in a small deformation of the shape. 4) Reproduction of particular shapes: typically the model must be able to represent particular types of idealized shapes (*e.g.* polynomial curves, ellipses, spheres) and have good approximation properties. 5) Continuity: depending on the application it can be required that the shape be everywhere differentiable. 6) Numerically stable implementation: usually this requires that the underlying mathematical functions are well-defined. 7) Allowing fast optimization: in semi-automatic applications, optimization schemes related to shape deformation might need to be included in the model.

Because it is not always possible to satisfy all of the constraints, in practice usually a trade-off between the above mentioned requirements needs to be done. Existing methods can generally be categorized by either using a discrete or a continuous-domain model. Discrete approaches use polygonal meshes [40, 42] or subdivision-based models [32, 38, 133] to represent shapes. They show high flexibility, but require a large amount of parameters for shape modeling, which is a drawback in optimization schemes. On the other hand, continuous-domain models are mostly based on Bézier curves [24, 54], spherical harmonics [134] or on compactly supported basis functions such as B-splines [53, 51, 52, 25, 135] which have an explicit analytical expression. However, B-splines are only able to represent polynomial shapes [55, 136] and therefore, do not allow for the construction of ellipses and spheres.

We present a new trigonometric basis function that enables the parametric representation and deformation of idealized shapes, such as ellipses and spheres. It allows to generate shape models that meet all of the requirements listed above and is particularly useful for simplified user-interaction because it is compactly supported and verifies the interpolation property. This means that the control points of the shape, which are accessible to the user, directly lie on its boundary.

The main contributions of this article are the derivation of the proposed interpolation kernel together with all the necessary properties to construct deformable models for shape representation. The motivation behind this work is the construction of 2D [90, 83] and 3D [137, 43, 26, 86] active contour models for the segmentation of sphere-like structures in biomedical images such as roundish cells [86] or organs [79](Figure 6.1).

6.1.2 Construction of the Interpolator

In [84], ellipse-reproducing basis functions are proposed that are defined as $\psi_M(t) = \sum_{n=0}^{2} \lambda_n \frac{\mathrm{d}^n}{\mathrm{d}t^n} \beta_{\boldsymbol{\alpha}_1}(t)$, where M is the number of control points used to construct a given function and $\beta_{\boldsymbol{\alpha}_1}$ is the 3-th order causal exponential B-spline defined in the Fourier domain as $\hat{\beta}_{\boldsymbol{\alpha}_1}(\omega) = \prod_{k=1}^{3} \frac{1-e^{\alpha_k - j\omega}}{j\omega - \alpha_k}$. Thereby, the vector $\boldsymbol{\alpha}_1 = (0, \frac{j2\pi}{M}, -\frac{j2\pi}{M})$ specifies the poles of the B-spline. We have shown that ψ_M is of minimal support and can be either smooth or interpolatory but not both at the same time. Finite support is important to implement fast optimization schemes [138].

Proposed Interpolator

Definition 6. Our piece-wise exponential basis function ϕ_M is expressed in its causal form as

$$\phi_M = \beta_{\boldsymbol{\alpha}_1} * \phi_M^0, \tag{6.1}$$

where $\phi_M^0 = \gamma_1(M)\beta_{(0)} + \gamma_2(M)(\delta + \delta(\cdot - 1))$ is a smoothing kernel of unit support, $\beta_{(0)}$ is the zero*th*-degree B-spline, and γ_1, γ_2 are chosen such that the centered generator $\phi_M(\cdot + a)$, satisfies the interpolation condition

$$\phi_M(k + a) = \delta_k. \tag{6.2}$$

Here, a is the appropriate shifting constant to center the generator, the dot in the argument of a function is a placeholder for its parameter, and δ_k is the Kronecker delta. Because $\beta_{\boldsymbol{\alpha}_1}$ is of support equal to 3 and ϕ_M^0 has unit support, the resulting support of ϕ_M is equal to 4 and hence $a = 2$. The unique weights γ_1 and γ_2 used to compute ϕ_M^0 in (6.1) are computed by solving (7.13) while additionally enforcing ϕ_M to be symmetric, *i.e.*, $\phi_M(t + a) = \phi_M(-t + a)$; a property that is especially convenient in practice. We find

$$\gamma_1(M) = \frac{\pi^3 \sec^2\left(\frac{\pi}{M}\right)}{M^2\left(M \tan\left(\frac{\pi}{M}\right) - \pi\right)}$$

and

$$\gamma_2(M) = \frac{\pi^2 \csc\left(\frac{\pi}{M}\right) \csc\left(\frac{2\pi}{M}\right)\left(M - 2\pi \csc\left(\frac{2\pi}{M}\right)\right)}{M^2\left(M \sec\left(\frac{\pi}{M}\right) - \pi \csc\left(\frac{\pi}{M}\right)\right)}.$$

The explicit expression of the proposed interpolator $\varphi_M(t) = \phi_M(t + a)$ is given by (6.3), where the interpolator φ_M is the centered (*i.e.*, shifted) version of the causal generator ϕ_M. In the following we derive the properties of the proposed interpolator that are useful in practical applications.

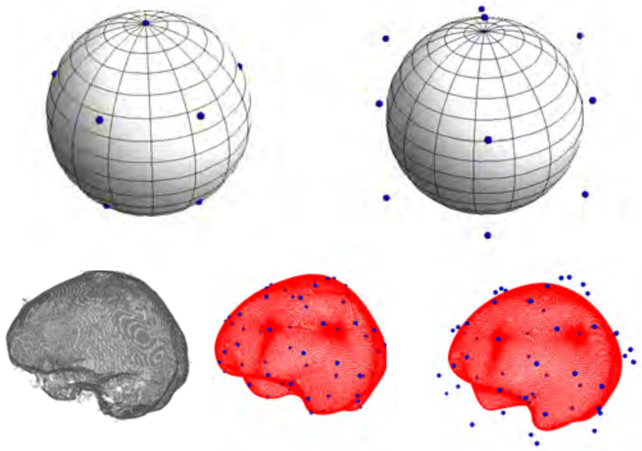


Figure 6.1: Reproduction and approximation of shapes. Top row: Reproduction of the exact sphere using the proposed interpolatory (left) and the non-interpolatory function (right) from [84]. Bottom row: Segmentation of a brain volume. On the left a rendering of a brain is shown that has been extracted from a 3D MRI scan as described in [79]. The 3D brain structure has been segmented with a deformable model using our interpolatory (middle) as well as the non-interpolatory (right) basis function [84]. In both cases the results are $\mathcal{C}^1$-diffeomorphic to the sphere, which has been used to initialize them. The blue dots are the control points. They directly lie on the shape boundary for the interpolatory scheme. The segmented brain shape on the bottom right shows that user-interactive shape modification is difficult and non-intuitive for the non-interpolatory case, because it is unclear which part of the surface is affected by moving a control point.

$$\varphi_M(t) = \begin{cases} \dfrac{\sin^2\left(\frac{\pi}{M}\right)\left(M\sin\left(\frac{2\pi(t-2)}{M}\right)-2\pi(t-2)\right)+\left(M\sin\left(\frac{2\pi}{M}\right)-2\pi\right)\sin^2\left(\frac{\pi(t-2)}{M}\right)}{2\left(\cos\left(\frac{2\pi}{M}\right)-1\right)\left(-M\sin\left(\frac{2\pi}{M}\right)+\pi\cos\left(\frac{2\pi}{M}\right)+\pi\right)} & 1 < t < 2 \\ \dfrac{M\left(\sin\left(\frac{2\pi(t-2)}{M}\right)-2\sin\left(\frac{2\pi(t-1)}{M}\right)+\sin\left(\frac{2\pi(t+1)}{M}\right)+\sin\left(\frac{2\pi}{M}\right)-\sin\left(\frac{4\pi}{M}\right)\right)+2\pi t\cos\left(\frac{2\pi}{M}\right)-2\pi(t-1)\cos\left(\frac{4\pi}{M}\right)-2\pi\cos\left(\frac{2\pi t}{M}\right)}{4\left(\cos\left(\frac{2\pi}{M}\right)-1\right)\left(-M\sin\left(\frac{2\pi}{M}\right)+\pi\cos\left(\frac{2\pi}{M}\right)+\pi\right)} & 0 < t \leq 1 \\ -\dfrac{M\left(\sin\left(\frac{2\pi(t-1)}{M}\right)-2\sin\left(\frac{2\pi(t+1)}{M}\right)+\sin\left(\frac{2\pi(t+2)}{M}\right)-\sin\left(\frac{2\pi}{M}\right)+\sin\left(\frac{4\pi}{M}\right)\right)+2\pi\left(t\cos\left(\frac{2\pi}{M}\right)-(t+1)\cos\left(\frac{4\pi}{M}\right)+\cos\left(\frac{2\pi t}{M}\right)\right)}{4\left(\cos\left(\frac{2\pi}{M}\right)-1\right)\left(-M\sin\left(\frac{2\pi}{M}\right)+\pi\cos\left(\frac{2\pi}{M}\right)+\pi\right)} & -1 < t \leq 0 \\ \dfrac{\sin^2\left(\frac{\pi}{M}\right)\left(2\pi(t+2)-M\sin\left(\frac{2\pi(t+2)}{M}\right)\right)+\left(M\sin\left(\frac{2\pi}{M}\right)-2\pi\right)\sin^2\left(\frac{\pi(t+2)}{M}\right)}{2\left(\cos\left(\frac{2\pi}{M}\right)-1\right)\left(-M\sin\left(\frac{2\pi}{M}\right)+\pi\cos\left(\frac{2\pi}{M}\right)+\pi\right)} & -2 < t \leq -1 \\ 0 & |t| \geq 2 \end{cases} \tag{6.3}$$

Reproduction of Trigonometric Functions

To be able to represent shapes that are deformations and translations of ellipses or spheres our generator φ_M must reproduce sines and cosines, as well as constants.

Proposition 8. *The interpolatory basis function φ_M reproduces constants as well as $\cos(\frac{2\pi\cdot}{M})$ and $\sin(\frac{2\pi\cdot}{M})$ independently of the number of control points $M \geq 3$.*

This result follows from [63, Proposition 2], where it was shown that reproduction properties are preserved through convolution. In [84] it was shown that $\beta_{\boldsymbol{\alpha}_1}$ reproduces the functions stated in Proposition 19 if and only if $M \geq 3$ and hence, by (6.1) φ_M inherits from $\beta_{\boldsymbol{\alpha}_1}$ its ellipse-reproducing properties.

Smoothness and Regularity

A fundamental requirement for the construction of the basis function is that it must be everywhere differentiable. This is important w.r.t. shape deformation in order to avoid discontinuities when perturbing an ideally reproduced shape, such as an ellipse or sphere.

Proposition 9. *The interpolator φ_M belongs to $\mathcal{C}^1$ and has bounded second derivatives.*

Proof: We re-express (6.1) in terms of exponential B-splines of different order as

$$\begin{aligned}\varphi_M &= \beta_{\boldsymbol{\alpha}_1} * \varphi_M^0 \\ &= \beta_{\boldsymbol{\alpha}_1} * \Big(\gamma_1(M)\beta_{(0)} + \gamma_2(M)(\delta + \delta(\cdot - 1))\Big) \\ &= \gamma_1(M)\beta_{\boldsymbol{\alpha}_1 \cup (0)} + \gamma_2(M)(\beta_{\boldsymbol{\alpha}_1} + \beta_{\boldsymbol{\alpha}_1}(\cdot - 1)),\end{aligned} \tag{6.4}$$

where we have used the property that the convolution between two exponential B-splines specified by $\beta_{\boldsymbol{\alpha}_m}$ and $\beta_{\boldsymbol{\alpha}_n}$ yields another exponential B-spline specified by $\boldsymbol{\alpha} = \boldsymbol{\alpha}_m \cup \boldsymbol{\alpha}_n$, *i.e.*, the union of the sets of poles defining the functions to be convolved. Now we use the fact that the order N of an exponential B-spline corresponds to the number of poles defining it. The number of times a B-spline is everywhere continuously differentiable is equal to $N-2$. From (6.4) we see that the lowest order of B-spline involved in the construction of φ_M is $N=3$ and hence, due to the linearity of the derivative $\varphi_M \in \mathcal{C}^1$. The second part of Proposition 9 follows from the fact that (exponential) B-splines of order N are Hölder-continuous of order $N-1$ with bounded derivatives [63]. ■

Convergence and Order of Approximation

The order of approximation of the interpolator is of importance because it describes how fast an approximated function or shape converges towards the object being interpolated.

Proposition 10. *The interpolator φ_M converges to the (polynomial) Keys interpolator [127] (which in computer graphics is known as the Catmull-Rom spline [16]). It is given by*

$$\phi_{\text{Keys}} = 3\beta^3 - (\beta^2 + \beta^2(\cdot - 1)) \tag{6.5}$$

and has an order of approximation of $L=3$. Thereby, β^3 and β^2 are the cubic and quadratic polynomial B-splines.

Proof: We observe that as M grows large, *i.e.*, $M \to \infty$ the 3rd order exponential B-spline defined by $\boldsymbol{\alpha}_1$ converges to the 3rd order polynomial B-spline that is defined by its poles $\boldsymbol{\alpha} = (0,0,0)$. Since $\lim_{M\to+\infty} \gamma_1(M) = 3$ and $\lim_{M\to+\infty} \gamma_2(M) = -1$ we obtain $\lim_{M\to+\infty} \varphi_M = \phi_{\text{Keys}}$. ■
Furthermore, φ_M is able to approximate any curve with arbitrary precision by chosing M sufficiently large [139].

Riesz Basis

It is desirable that the curves and surfaces given by (7.1) and (7.2) are uniquely specified by their sequence of control points $\{\boldsymbol{c}[k]\}_{k\in\mathbb{Z}}$ and $\{\boldsymbol{c}[k,l]\}_{k,l\in\mathbb{Z}}$ respectively. Therefore, the shifted basis functions $\{\varphi_M(\cdot - k)\}_{k\in\mathbb{Z}}$ should be linearly independent. Additionally, for practical reasons, the interpolation process must be numerically stable. These requirements are fullfilled if the generating function satisfies the Riesz-basis condition [63].

Proposition 11. *For $M \geq 3$ the function φ_M generates a Riesz basis, i.e., there exist two constants $0 < A \leq B < \infty$, such that*

$$0 \leq A\|c\|^2_{l_2(\mathbb{Z})} \leq \|\sum_{k\in\mathbb{Z}} c[k]\varphi_M(\cdot - k)\|^2_{L_2(\mathbb{R})} \leq B\|c\|^2_{l_2(\mathbb{Z})} < \infty \tag{6.6}$$

for all $c \in l_2(\mathbb{Z})$.

We only outline a sketch of proof. It is based on the fact that (6.6) is expressed in the Fourier domain as $A \leq \sum_{k=-\infty}^{\infty} |\hat{\varphi}_M(\omega + 2\pi k)|^2 \leq B$, and that using (6.4) the Fourier transform of φ_M is given by $\hat{\varphi}_M(\omega) = \gamma_1\hat{\beta}_{\boldsymbol{\alpha}_1\cup(0)}(\omega) + \gamma_2(\hat{\beta}_{\boldsymbol{\alpha}_1})(\omega)(1+e^{-j\omega})$. The complete proof is similar to the one of [140, Theorem 6.2].

Affine Invariance and Partition of Unity

Affine invariance holds if and only if $\sum_{k\in\mathbb{Z}} \varphi_M(t-k) = 1$, which is called *partition of unity*. Furthermore, from Proposition 19 we know that φ_M reproduces constants. As a consequence and using the fact that φ_M is an interpolator the partition of unity is verified and hence, also affine invariance.

6.1.3 Reproduction of Ellipses

A direct consequence of Proposition 1 is that φ_M allows us to construct ellipses independently from the number of control points $M \geq 3$. In this section we explicitly show how ellipses can be reproduced using the proposed basis functions. Because ellipses can be constructed by applying an affine transformation to a circle and using the property of our model to be affine invariant it suffices to show that we can generate circles.

Proposition 12. *Using the generator φ_M the unit circle is parametrized as*

$$\boldsymbol{r}(t) = \begin{pmatrix}\cos(2\pi t)\\ \sin(2\pi t)\end{pmatrix} = \sum_{k=0}^{M-1} \begin{pmatrix} c^c_M[k] \\ c^s_M[k]\end{pmatrix} \varphi_{M,\mathrm{per}}(Mt-k), \tag{6.7}$$

where $\varphi_{M,\mathrm{per}} = \sum_{n=-\infty}^{+\infty} \varphi_M(t - Mn - k)$ is the M-periodization of φ_M, $c^c_M[\cdot] = \cos[\frac{2\pi\cdot}{M}]$, $c^s_M[\cdot] = \sin[\frac{2\pi\cdot}{M}]$, and $t \in [0,1)$.

Proof: Using Proposition 19 combining with the fact that φ_M is an interpolator and $\cos(\frac{2\pi t}{M})$ is M-periodic we write

$$\begin{aligned}\cos(\frac{2\pi t}{M}) &= \sum_{k\in\mathbb{Z}} \cos[\frac{2\pi k}{M}]\varphi_M(t-k)\\ &= \sum_{k=0}^{M-1}\sum_{n=-\infty}^{+\infty}\cos[\frac{2\pi(Mn+k)}{M}]\varphi_M(t-Mn-k)\\ &= \sum_{k=0}^{M-1}\cos[\frac{2\pi k}{M}]\underbrace{\sum_{n=-\infty}^{+\infty}\varphi_M(t-Mn-k)}_{\varphi_{M,\mathrm{per}}(t-k)}\\ \Rightarrow \cos(2\pi t) &= \sum_{k=0}^{M-1}\varphi_{M,\mathrm{per}}(Mt-k).\end{aligned} \tag{6.8}$$

Combining (6.8) with a similar derivation for $\sin(\frac{2\pi t}{M})$ proves the claim. ■
Plots of the reconstructed trigonometric functions are shown in Figure 7.21 as well as the unit circle that has been reconstructed by φ_M with the smallest possible number of control points $M = 3$ (Riesz-basis condition).

6.1.4 Reproduction of spheres

In this section, we outline our proposed construction of the sphere. Its parameterization as a tensor-product surface (7.2) results as a corollary from Proposition 12.

Corollary 1. *Using the generator φ_M the unit sphere is parameterized as*

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \begin{pmatrix}x(u,v)\\ y(u,v)\\ z(u,v)\end{pmatrix} &= \begin{pmatrix}\cos(2\pi u)\sin(\pi v)\\ \sin(2\pi u)\sin(\pi v)\\ \cos(\pi v)\end{pmatrix}\\ &= \sum_{k=0}^{M_1-1}\sum_{l=-1}^{M_2+1}\boldsymbol{c}[k,l]\varphi_{M_1,\mathrm{per}}(M_1u-k)\varphi_{2M_2}(M_2v-l),\end{aligned} \tag{6.9}$$

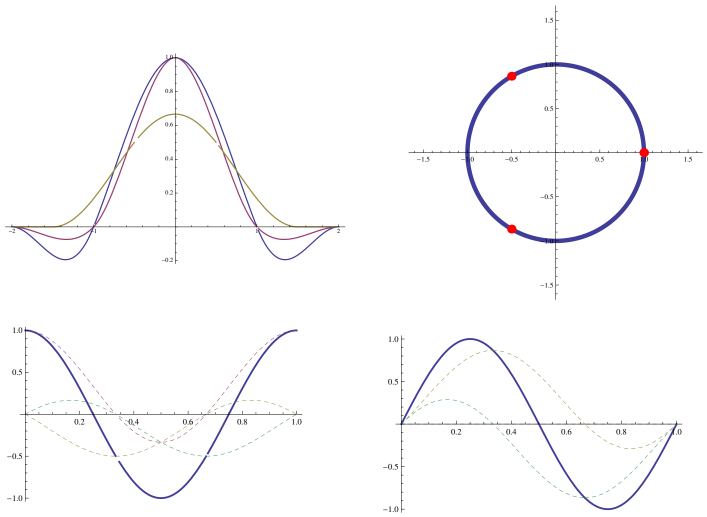


Figure 6.2: Top left: The proposed trigonometric interpolator (blue), Keys interpolator (purple) and the non-interpolatory ellipse reproducing basis function from [84] (yellow). Top right: the circle obtained with the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$. Bottom: $\cos(2\pi t)$ (left) and $\sin(2\pi t)$ (right) are shown together with the basis functions (colored dashed lines) corresponding to $c^c_M[k]$ and $c^s_M[k]$, for $M = 3$ and $k \in [0 \dots M-1]$ respectively.

where $u, v \in [0,1)$, and the control points are given by

$$\boldsymbol{c}[k,l] = \begin{pmatrix} c^c_{M_1}[k] c^s_{2M_2}[l] \\ c^s_{M_1}[k] c^s_{2M_2}[l] \\ c^c_{2M_2}[l] \end{pmatrix}. \tag{6.10}$$

The limits of the second sum in (7.4) are due to the fact that $v \in [0,1)$ and the support of φ_{2M_2} is limited to the interval $[-2,2]$. Therefore, we have $l \notin [-1,\dots,M_2+1] \Leftrightarrow \varphi_{2M_2}(M_2 v - l) = 0$. The other terms in (7.4) follow from inserting $M = 2M_2$ in (6.8). Plots of a reconstructed sphere and a deformation of it are shown in Figure 6.1, as well as a comparison with shapes constructed with the non-interpolatory basis function from [84].

6.1.5 Conclusion

We present a new trigonometric kernel to construct deformable shape models. We show that the kernel satisfies the necessary requirements needed for their construction such as: interpolation condition, compact support, smoothness, reproduction properties, Riesz-basis condition. The main advantage of the proposed basis functions is that they are smooth while also being interpolatory, therefore allowing the control points of a constructed shape to lie directly on the shape boundary; a feature that allows for intuitive shape manipulation in user-interactive applications. We explicitly show how to construct idealized shapes such as circles and spheres. We illustrate the use of such models in demo-videos showing 2D and 3D user-interactive deformable models that are constructed using (7.5) and (7.4). They are avialble at *http://bigwww.epfl.ch/demo/interpolated-shapes/*.

6.2 A Family of Smooth Interpolators

In this section, we generalize the result presented in the previous Section 6.1 to construct interpolators of any order.

Interpolatory basis functions are helpful to specify parametric curves or surfaces that can be modified by simple user-interaction. Their main advantage is a characterization of the object by a set of control points that lie on the shape itself (*i.e.*, curve or surface). Here, we characterize a new family of compactly supported piece-wise-exponential basis functions that are smooth and satisfy the interpolation property. They can be seen as a generalization and extension of the Keys interpolation kernel using cardinal exponential B-splines. The proposed interpolators can be designed to reproduce trigonometric, hyperbolic, and polynomial functions or combinations of them. We illustrate the construction and give concrete examples on how to use such functions to construct parametric curves and surfaces.

6.2.1 Introduction

The representation of shapes using parametric curves and surfaces is widely used in domains that make use of computer graphics [141, 54] such as industrial design [128, 129, 130], the animation industry [30], as well as for the analysis of biomedical images [68, 79, 86]. In that context, it is often important to be able to interactively change the shape of the curve or surface. The spline-based representation of parametric shapes has proven to be a convenient choice to include user interactivity in shape modeling due to the underlying control-point-based nature of spline functions. If the basis functions are compactly supported, the change of position of a control point modifies the shape only locally. This allows for a local control by the user. Commonly used basis functions such as NURBS or B-splines have this locality property but are in general not interpolatory (except for example zeroth and first degree B-splines, which are not smooth) [25]. This has the disadvantage that the control points do not directly lie on the contour or surface of the shape. Especially in 3D applications, this can be inconvenient because it is no longer intuitive to interactively modify complex shapes. More recently a method to construct piecewise polynomial interpolators has been presented in [55, 142].

In this paper, we propose a new family of piecewise exponential basis functions that are interpolatory and are at least in $\mathcal{C}^1$. They are compactly supported and their order can be chosen to be arbitrarily high. We show that they are able to reproduce exponential polynomials which include the pure polynomials as a subset. This convenient property is particularly relevant for the exact rendering of conic sections such as circles, ellipses, or parabolas, as well as other trigonometric and hyperbolic curves and surfaces [84]. In its absence, one must resort to subdivision to tackle this aspect [32, 35, 38, 133, 143]. However, existing comparable subdivision schemes usually rely on basis functions that are defined as a limit process and do not have a closed-form expression [144].

Our proposed family generalizes the piecewise-polynomial Keys interpolator [127, 16, 145] to higher degrees and can be seen as its extension using exponential B-splines [63, 146].

The paper is organized as follows. In section 2 we give a brief review on exponential B-splines and their relation with differential operators. This is needed to understand the reproduction properties of our proposed interpolators since they are based on exponential B-splines. In Section 3 we present the proposed family of interpolators. We present the relevant properties and prove that they reproduce exponential polynomials. We also provide a generic algorithm to construct specific interpolators that belong to the proposed family. In Section 4 we give specific examples of interpolators and we explicitly show how parametric curves and surfaces with desirable reproduction properties are constructed.

6.2.2 Exponential B-splines

In this section, we briefly review the link between exponential B-splines and differential operators which is crucial to understand the reproduction properties of the proposed spline family. (For a more in-depth characterization of exponential B-splines, we refer the reader to [63, 146].) These reproduction properties are needed for the exact representation of elementary shapes (see Section 6.3.3 for examples) and are automatically enforced by our construction.

Notation

We describe the list of roots $\alpha_1, \dots, \alpha_N$ using the vector notation $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_N)$. To assert the inclusion of a list of roots $\boldsymbol{\alpha}_1$ into another list $\boldsymbol{\alpha}_2$, we use the set notation $\boldsymbol{\alpha}_1 \subset \boldsymbol{\alpha}_2$. If $\boldsymbol{\alpha}_1$ must be excluded from $\boldsymbol{\alpha}_2$, we write $\boldsymbol{\alpha}_2 \setminus \boldsymbol{\alpha}_1$. Similarly, we denote the union of the two lists of roots $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ by $\boldsymbol{\alpha}_1 \cup \boldsymbol{\alpha}_2$. Likewise, we write $\alpha_n \in \boldsymbol{\alpha}$ to signify that one of the components of $\boldsymbol{\alpha}$ is α_n. Furthermore, the nth-order derivative operator is denoted by $D^n = \frac{\mathrm{d}^n}{\mathrm{d}t^n}$ with $D^0 = I$ (identity operator).

Operator Properties and Reproduction of Null-Space Components

Consider the generic differential operator L of order N

$$L = D^N + a_{N-1}D^{N-1} + \cdots + a_0 I. \tag{6.11}$$

Its characteristic polynomial with variable $s \in \mathbb{C}$ is given by

$$L(s) = s^N + a_{N-1}s^{N-1} + \cdots + a_0 = \prod_{n=1}^{N} (s - \alpha_n). \tag{6.12}$$

By evaluating $L(s)$ at $s = \mathrm{j}\omega$, where $\mathrm{j}^2 = -1$, we obtain that the frequency response of the differential operator is $\hat{L}(\mathrm{j}\omega) = \prod_{n=1}^{N}(\mathrm{j}\omega - \alpha_n)$. This allows us to factorize the operator L as

$$L_{\boldsymbol{\alpha}} := L = (D - \alpha_1 I)(D - \alpha_2 I)\cdots(D - \alpha_N I). \tag{6.13}$$

It follows that the nullspace, which contains all the solutions of the homogeneous differential equation $L_{\boldsymbol{\alpha}}\{f_0\}(t) = 0$, is given by

$$N_{L_{\boldsymbol{\alpha}}} = \mathrm{span}\{t^{n-1}\mathrm{e}^{\alpha_{(m)}t}\}_{m=1,\ldots,N_d;n=1,\ldots,n_{(m)}}, \tag{6.14}$$

where the N_d distinct roots of the characteristic polynomial are denoted by $\alpha_{(1)}, \ldots, \alpha_{(N_d)}$ with the multiplicity of $\alpha_{(m)}$ being $n_{(m)}$ and $\sum_{m=1}^{N_d} n_{(m)} = N$. There exists a unique causal Green's function $\rho_{\boldsymbol{\alpha}}$ ($\rho_{\boldsymbol{\alpha}}(t) < 0$ for $t < 0$) associated to the operator $L_{\boldsymbol{\alpha}}$ that satisfies $L_{\boldsymbol{\alpha}}\{\rho_{\boldsymbol{\alpha}}\}(t) = \delta(t)$, where δ is the Dirac distribution.
Its explicit form is

$$\rho_{\boldsymbol{\alpha}}(t) = \sum_{m=1}^{N_d}\sum_{n=1}^{n_{(m)}} c_{m,n}\frac{t_+^{n-1}}{(n-1)!}\mathrm{e}^{\alpha_{(m)}t}, \tag{6.15}$$

with suitable constants $c_{m,n}$. We see that (6.15) is a causal exponential polynomial. The discrete counterpart of $L_{\boldsymbol{\alpha}}$ is denoted by $\Delta_{\boldsymbol{\alpha}}$. It is specified by its symbol $\hat{\Delta}_{\boldsymbol{\alpha}}(z) = \prod_{n=1}^{N}(1 - \mathrm{e}^{\alpha_n}z^{-1})$. An exponential B-spline is then defined as $\beta_{\boldsymbol{\alpha}}(t) = \Delta_{\boldsymbol{\alpha}}\{\rho_{\boldsymbol{\alpha}}\}(t)$, which is equivalent to the Fourier-domain definition

$$\hat{\beta}_{\boldsymbol{\alpha}}(\omega) = \frac{\hat{\Delta}_{\boldsymbol{\alpha}}(\mathrm{e}^{\mathrm{j}\omega})}{\hat{L}_{\boldsymbol{\alpha}}(\mathrm{j}\omega)} = \prod_{k=1}^{n}\frac{1 - \mathrm{e}^{\alpha_k - \mathrm{j}\omega}}{\mathrm{j}\omega - \alpha_k}. \tag{6.16}$$

Since $\Delta_{\boldsymbol{\alpha}}$ is defined on the integer grid, the exponential B-splines reproduce the causal Green's function (6.15) associated to $L_{\boldsymbol{\alpha}}$

$$\rho_{\boldsymbol{\alpha}}(t) = \Delta_{\boldsymbol{\alpha}}^{-1}\{\beta_{\boldsymbol{\alpha}}\}(t) = \sum_{k=0}^{+\infty} p_{\boldsymbol{\alpha}}[k]\beta_{\boldsymbol{\alpha}}(t-k), \tag{6.17}$$

where $p_{\boldsymbol{\alpha}}$ is a unique causal sequence as has been shown in [63]. Extrapolating the Green's function (6.15) for $t < 0$ is equivalent to extrapolating the sum in (6.17) for negative k, which results in the reproduction of an exponential polynomial. More generally, it can be shown that $\beta_{\boldsymbol{\alpha}}$ is able to reproduce any component $P_0(t) \in N_L$ that is in the null space of $L = L_{\boldsymbol{\alpha}}$.

6.2.3 Construction of Interpolatory Basis Functions

Desirable Properties of the Basis Functions

We want to construct an interpolator based on a suitable linear combination of exponential B-splines of different orders. The following characteristics should be met:

- *Smoothness*
  We want the interpolation functions to be at least continuously differentiable and, hence, the minimum order of the B-spline involved is 3 (*i.e.*, degree 2).

- *Support*
  The interpolator should be compactly supported and the support of the function should not be larger than the support of the B-spline of highest order N involved. Therefore, the support of the resulting function is an integer and is equal to N.

- *Symmetry*
  We want the interpolator to be symmetric. This can be achieved if the non-zero poles of the exponential spline are grouped in pairs of opposite sign [63]. Furthermore, except for the highest-order B-spline involved, the B-splines in the sum have to come in pairs and be shifted accordingly.

- *Interpolation Condition*
  The constructed function has to satisfy the interpolation condition

  $$\varphi(t)|_{t=k} = \delta[k], \quad k \in \mathbb{Z}, \tag{6.18}$$

  where $\delta[k]$ represents the Kronecker delta.

- *Reproduction of Exponential Polynomials*
  We are interested in representing shapes that do not only rely on polynomial but also on trigonometric and hyperbolic coordinate functions. Thus, the interpolators must reproduce exponential polynomials.

Characterization of the Family of Interpolators

Taking all of the above considerations into account we characterize an N^{th} order smooth and piece-wise exponential interpolator as

$$\varphi(t) := \lambda_N \beta_{\boldsymbol{\alpha}_N}(t+\frac{N}{2}) + \sum_{n=n_0}^{N-1} \lambda_n \Big(\beta_{\boldsymbol{\alpha}_n}(t+\frac{N}{2}) + \beta_{\boldsymbol{\alpha}_n}(t-\frac{N}{2}+n) \Big), \tag{6.19}$$

where $\boldsymbol{\alpha}_{n_0}$ has at least $n_0 = 3$ poles (smoothness constraint) and $N \geq 2(n_0-1)$ is an integer that defines the highest-order exponential B-spline involved. Furthermore, in order for φ to reproduce exponential polynomials, we enforce $\boldsymbol{\alpha}_n \subset \boldsymbol{\alpha}_N$ for $n \in [n_0, N-1]$ (see Section 6.2.3). Here, the notation $\boldsymbol{\alpha}_n$ implies that the list of poles $\boldsymbol{\alpha}_n$ contains n elements. Hence, using the fact that the support of an exponential B-spline is equal to the number of poles that specifies it, we see that φ is of support equal to N.
The weights λ_n are computed by making use of the symmetry of the interpolator, *i.e.*, $\varphi(t) = \varphi(-t)$ and by imposing the interpolation constraints (7.13), which we achieve by solving the system of equations

$$\begin{cases} 1 = \varphi(0) \\ 0 = \varphi(1) \\ \vdots \\ 0 = \varphi(\lfloor N/2 \rfloor). \end{cases} \tag{6.20}$$

From (6.20) we see that, $N/2$ (even case) respectively $N/2+1$ (odd case) interpolation constraints have to be met to construct the function. This follows from the fact that φ can only be non-zero within the interval $[-N/2, N/2]$. If N is even, $N/2$ is integer and since the interpolator is smooth $\varphi(N/2) = 0$ and hence, does not explicitly need to be imposed in (6.20).

Construction of Basis Functions

The proposed interpolatory basis functions are constructed as follows:

1. Define an exponential spline type with desirable reproduction properties; that is, select the list $\boldsymbol{\alpha}_{n_0}$ that contains the featured poles and whose total number of elements is $n_0 \geq 3$.
2. Given $\boldsymbol{\alpha}_{n_0}$, N must be no smaller than $N_{\min} = 2(n_0 - 1)$. This restriction is directly related to the interpolation constraints.
3. The N*th*-order interpolator is given by (6.19), where $\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n_0} \cup \mathbf{0}_{n-n_0}$ with $n > n_0$ and $\mathbf{0}_K$ is the K-element vector filled with zeros.
4. The weights λ_n can be computed by solving (6.20).

The system (6.20) is over-determined when $N > N_{\min}$. In that case, the weights λ_n for $(N - N_{\min})$ terms in (6.19) can be chosen arbitrarily (see Section 6.2.4 for examples). Conversely, we see that, in order to satisfy the interpolation constraints for a given order N, the smallest possible number of poles is $n_0 = \lfloor N/2 \rfloor + 1$. Otherwise, when the system of equations is overdetermined, one can always construct $\boldsymbol{\alpha}_{n'_0} = \boldsymbol{\alpha}_{n_0} \cup \mathbf{0}_{(\lfloor N/2 \rfloor + 1) - n_0}$ with n'_0 poles. The resulting exponential B-spline $\beta_{\boldsymbol{\alpha}_{n'_0}}$ preserves the reproduction properties of $\beta_{\boldsymbol{\alpha}_{n_0}}$ but has increased order of approximation and regularity [63] (see Section 6.2.3) Based on our experiments we conjecture that the system of equations (6.20) always has a solution.

Reproduction Properties and Regularity

Reproduction of Exponential Polynomials.

Proposition 13. *The interpolator defined by* (6.19) *reproduces exponential polynomials up to degree q and exponent α if and only if $\boldsymbol{\alpha}_{n_0}$ contains $q+1$ copies of α.*

Proof. We first show that exponential polynomials can be reproduced with exponential B-splines and then conclude that φ preserves these reproduction properties.

An exponential polynomial of exponent α and degree q can always be written as a linear combination of exponential monomials $Q^n_\alpha(t) = \mathrm{e}^{\alpha t} t^n$ for $n = 0, \dots, q$. The exponential polynomial is expressed as

$$P^q_\alpha(t) = \sum_{k=0}^{q} a_k \mathrm{e}^{\alpha t} t^n. \tag{6.21}$$

For $n \geq n_0$ in (6.19) and $n \in [n_0, N]$, every exponential B-spline $\beta_{\boldsymbol{\alpha}_n}$ is defined through a list $\boldsymbol{\alpha}_n$ that contains $\boldsymbol{\alpha}_{n_0}$. Furthermore, from [63] we know that, for $\alpha \in \boldsymbol{\alpha}_{n_0}$ of multiplicity $q+1$, there exist sequences p_n such that

$$\mathrm{e}^{\alpha t} t^n = \sum_{k \in \mathbb{Z}} p_n[k] \beta_{\boldsymbol{\alpha}_{n_0}}(t-k) \tag{6.22}$$

for $n = 0, \ldots, q$, which is equivalent to saying that $\beta_{\boldsymbol{\alpha}_{n_0}}$ reproduces exponential monomials up to degree q and exponent α. The shifted exponential B-splines in (6.19) also have the same reproduction property. By combining (6.21) and (6.22) and considering an arbitrary shift m, we see that

$$\begin{aligned} P_\alpha^n(t-m) &= \sum_{k=0}^{n} a_k Q_\alpha^k(t-m) \\ &= \sum_{k=0}^{n} a_k \sum_{l \in \mathbb{Z}} p_k[l] \beta_{\boldsymbol{\alpha}_{n_0}}(t-m-l) \\ &= \sum_{k=0}^{n} a_k \mathrm{e}^{\alpha(t-m)} (t-m)^k \\ &= \mathrm{e}^{\alpha t} \sum_{k=0}^{n} a_k \mathrm{e}^{-\alpha m} \sum_{l=0}^{k} \binom{k}{l} t^l (-1)^{k-l} m^{k-l}, \end{aligned} \tag{6.23}$$

which is a linear combination of polynomials in t of degree up to n that are multiplied by $\mathrm{e}^{\alpha t}$. Thus, we can collect all the factors multiplying t^k and rewrite them as b_k to express (6.23) as

$$P_\alpha^n(t-m) = \mathrm{e}^{\alpha t} \sum_{k=0}^{n} b_k t^k := P_{\alpha,m}^n(t) \tag{6.24}$$

for $n = 0, \ldots, q$, which is also an exponential polynomial of exponent α and degree n.

$\square$

In the next step, we first show that exponential polynomials can be reproduced if φ is composed of exponential B-splines of identical degree and containing the same poles. Then we conclude that, because the reproduction of exponential polynomials is preserved through convolution, φ also reproduces these exponential polynomials.

By (6.22) and (6.24) and using exponential B-splines of the same degree, we write

$$\sum_{k=0}^{p} a_k \sum_{l\in\mathbb{Z}} p_k[l]\left(\lambda_N \beta_{\boldsymbol{\alpha}_{n_0}}(t+\frac{N}{2}-l) + \sum_{n=\lfloor N/2\rfloor+1}^{N-1} \lambda_n\big(\beta_{\boldsymbol{\alpha}_{n_0}}(t+\frac{N}{2}-l)+\beta_{\boldsymbol{\alpha}_{n_0}}(t-\frac{N}{2}-l)\big)\right)$$
$$= P^p_{\alpha,-\frac{N}{2}}(t)\left(\lambda_N + \sum_{n=\lfloor N/2\rfloor+1}^{N-1}\lambda_n\right) + \sum_{n=\lfloor N/2\rfloor+1}^{N-1}\lambda_n P^p_{\alpha,\frac{N}{2}-n}(t) \tag{6.25}$$

for $p = 0, \dots, q$, which is also an exponential polynomial with the same degree and exponent as its constituents.

The next step of the proof relies on a proposition originally stated by Unser and Blu in [63], which we recall here for the sake of completeness.

Proposition 14 (Unser and Blu [63])**.** *Let ψ_α be a function that reproduces the exponential polynomials in* $\mathrm{span}\{\mathrm{e}^{\alpha t}, \dots, t^p\mathrm{e}^{\alpha t}\}$*. Then, for any compactly supported function ψ such that $\int_{\mathbb{R}}\psi(t)\mathrm{e}^{-\alpha t}\mathrm{d}t \neq 0$, the composite function $\psi * \psi_\alpha$ also reproduces these exponential polynomials (where $*$ denotes the convolution product).*

Using this proposition, we deduce that, for $n \in [n_0, N]$, the convolution product

$$\beta_{\boldsymbol{\alpha}_n\backslash\boldsymbol{\alpha}_{n_0}} * \beta_{\boldsymbol{\alpha}_{n_0}} \tag{6.26}$$

preserves the exponential reproduction properties of $\beta_{\boldsymbol{\alpha}_{n_0}}$. Note that in (6.26), the term $\beta_{\boldsymbol{\alpha}_{n_0}\backslash\boldsymbol{\alpha}_{n_0}} = 1$.

From the definition of the interpolatory basis function (6.19) and by combining (6.25) and (6.26), we obtain

$$\sum_{k=0}^{p} a_k \sum_{l\in\mathbb{Z}} \tilde{p}_k[l]\varphi(t-l) =$$
$$\sum_{k=0}^{p} a_k \sum_{l\in\mathbb{Z}} \tilde{p}_k[l]\Big(\lambda_N \beta_{\boldsymbol{\alpha}_N}(t+\frac{N}{2}-l) + \sum_{n=\lfloor N/2\rfloor+1}^{N-1} \lambda_n\big(\beta_{\boldsymbol{\alpha}_n}(t+\frac{N}{2}-l) + \beta_{\boldsymbol{\alpha}_n}(t-\frac{N}{2}-l)\big)\Big)$$
$$= \sum_{k=0}^{p} a_k \sum_{l\in\mathbb{Z}} \tilde{p}_k[l]\Big(\lambda_N\big(\beta_{\boldsymbol{\alpha}_N\setminus\boldsymbol{\alpha}_{n_0}} * \beta_{\boldsymbol{\alpha}_{n_0}}\big)(t+\frac{N}{2}-l)$$
$$+ \sum_{n=\lfloor N/2\rfloor+1}^{N-1} \lambda_n\Big(\beta_{\boldsymbol{\alpha}_n\setminus\boldsymbol{\alpha}_{n_0}} * \big(\beta_{\boldsymbol{\alpha}_{n_0}} + \beta_{\boldsymbol{\alpha}_{n_0}}(\cdot - N)\big)\Big)(t+\frac{N}{2}-l)\Big), \tag{6.27}$$

where $\tilde{p}_k$ is a suitable sequence of coefficients. Therefore, from (6.27) we see that φ also reproduces the exponential polynomials given by (6.25) up to degree q and exponent α, where $\alpha \in \boldsymbol{\alpha}_{n_0}$ is of multiplicity $q+1$.

Regularity The regularity of the proposed interpolator depends on the exponential B-spline of lowest order that is involved in the construction of φ. Hence, φ belongs to $\mathcal{C}^{n_0-2}$.

Order of Approximation. If the poles of the constructed interpolators are of the form $\alpha = \frac{\pi x}{M}$, $x \in \mathbb{C}$, and if M is related to the number of control points, then the definition of the interpolator (6.19) implies that, as $M \to \infty$, φ converges to piecewise-polynomial interpolators that have an $(n_0)th$ order of approximation (by the Strang-Fix equivalence [147, 61]). Such interpolators are of special interest for the construction of particular shapes (see Section 6.3.3).

In the following, we give concrete examples of interpolatory basis functions that are derived from (6.19).

6.2.4 Examples of Interpolators and Applications

Polynomial Bases

If the pole vector entirely consists of zeroes, the basis function is a sum of polynomial B-splines and hence is piecewise polynomial. For example, the 4*th*-order basis corresponds to the Keys interpolation kernel [127]. These basis functions are all symmetric. Some examples are shown in Figure 6.3.

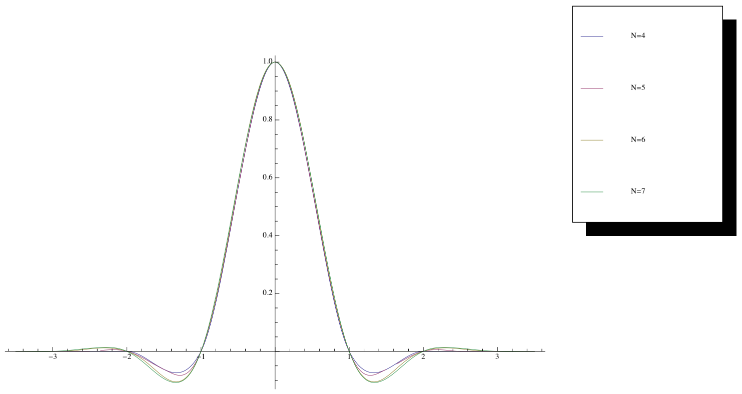


Figure 6.3: Examples of polynomial-reproducing interpolators. Their poles are all equal to zero and their respective order corresponds to $N = 4, \ldots, 7$, and is equal to their support. The basis function that corresponds to $N = 4$ represents the Keys interpolator [127].

Trigonometric and Hyperbolic Bases

Trigonometric and hyperbolic functions take special relevance within computational geometry. Exponential splines that are able to reproduce (hyberbolic) sines and cosines can be used to construct the desired interpolatory basis functions. Because the exponents involved in the representation of (hyperbolic) sinusoidal functions come in pairs of opposite sign, the resulting basis functions are symmetric as a consequence of the symmetry of their elementary constituents. Some hyperbolic and trigonometric interpolators are shown in Figure 6.4.

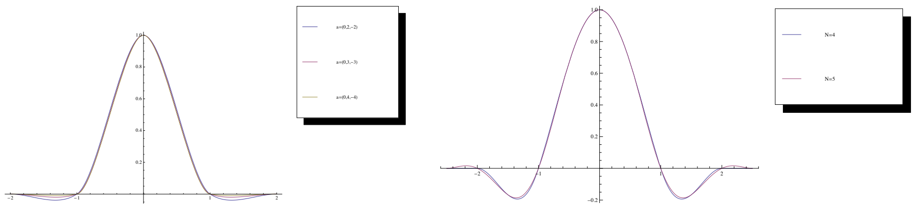


Figure 6.4: Hyperbolic (left) and trigonometric (right) interpolators. Left: $4th$-order interpolating functions are shown that were constructed with different lists of poles indicated by $\boldsymbol{\alpha}$. Right: Trigonometric interpolators. The two interpolators correspond to $4th$ and $5th$ order with $\boldsymbol{\alpha}_{n_0} = (0, \frac{2\mathrm{j}\pi}{3}, -\frac{2\mathrm{j}\pi}{3})$.

Larger Support Interpolators

For the sake of completeness, we also provide an example of how to construct interpolators by choosing n_0 and N such that the corresponding system of equations is over-determined (see Section 6.2.3). Such situations arise if either for a given $\boldsymbol{\alpha}_{n_0}$ a corresponding $N > N_{\min} = 2(n_0 - 1)$ is chosen or if for a given N a corresponding $n_0 < \lfloor N/2 \rfloor + 1$ is chosen. In both cases, the weights λ_n for $(N - N_{\min})$ terms in (6.19) can be chosen arbitrarily. Our experiments show that such interpolators that are constructed by solving an overdetermined system of equations tend to oscillate more than their "smallest" support counterparts. Examples are shown in Figure 6.6 (middle and right), where we constructed $6th$- and $7th$-order interpolators.

6.2.5 Applications

In this section, we show how idealized parametric curves and surfaces (such as ellipses and ellipsoids) can be reproduced using the proposed interpolators. Such shapes can be constructed independently of the number of control points, which makes them particularly useful for deformable models where, when starting from an initial configuration, it is desirable to approximate shapes with arbitrary precision [84, 85, 86, 79]. We construct symmetric interpolators that have the smallest support given $\boldsymbol{\alpha}_{n_0}$ as described in Section 6.2.3.

Reproduction of Parametric Curves

Reproduction of Ellipses. Here we explicitly show how ellipses are reproduced within our framework. We consider the lowest order, which is $N = 4$. The condition for being able to reconstruct an ellipse with M control points is $\boldsymbol{\alpha}_{n_0} = \boldsymbol{\alpha}_3 := (0, \frac{\mathrm{j}2\pi}{M}, -\frac{\mathrm{j}2\pi}{M})$. Therefore, by applying (6.19), the interpolator is

$$\varphi(t) = \lambda_4 \beta_{\boldsymbol{\alpha}_4}(t+2) + \lambda_3 \big(\beta_{\boldsymbol{\alpha}_3}(t+2) + \beta_{\boldsymbol{\alpha}_3}(t+1)\big) \tag{6.28}$$

with $\boldsymbol{\alpha}_4 = (0, 0, \frac{\mathrm{j}2\pi}{M}, -\frac{\mathrm{j}2\pi}{M})$ and the λ_n which are found by solving (6.20). Specifically, we end up with the system of equations

$$\begin{cases} 0 = \lambda_4 \beta_{\boldsymbol{\alpha}_4}(1) + \lambda_3 \big(\beta_{\boldsymbol{\alpha}_3}(1) + \beta_{\boldsymbol{\alpha}_3}(0)\big) = \lambda_4 \beta_{\boldsymbol{\alpha}_4}(1) + \lambda_3 \beta_{\boldsymbol{\alpha}_3}(1) \\ 1 = \lambda_4 \beta_{\boldsymbol{\alpha}_4}(2) + \lambda_3 \big(\beta_{\boldsymbol{\alpha}_3}(2) + \beta_{\boldsymbol{\alpha}_3}(1)\big) = \lambda_4 \beta_{\boldsymbol{\alpha}_4}(2) + 2\lambda_3 \beta_{\boldsymbol{\alpha}_3}(2) \end{cases} \tag{6.29}$$

whose solution is

$$\lambda_3(M) = \frac{\pi^2 \csc\left(\frac{\pi}{M}\right) \csc\left(\frac{2\pi}{M}\right) \left(M - 2\pi \csc\left(\frac{2\pi}{M}\right)\right)}{M^2 \left(M \sec\left(\frac{\pi}{M}\right) - \pi \csc\left(\frac{\pi}{M}\right)\right)} \tag{6.30}$$

and

$$\lambda_4(M) = \frac{\pi^3 \sec^2\left(\frac{\pi}{M}\right)}{M^2 \left(M \tan\left(\frac{\pi}{M}\right) - \pi\right)}. \tag{6.31}$$

To reproduce $\cos(\frac{2\pi}{M}\cdot)$, we take advantage of the interpolation property which yields

$$\cos(\frac{2\pi}{M}t) = \sum_{k\in\mathbb{Z}} \frac{\mathrm{e}^{\mathrm{j}\frac{2\pi}{M}k} + \mathrm{e}^{-\mathrm{j}\frac{2\pi}{M}k}}{2} \varphi(t-k), \tag{6.32}$$

where the coefficients are the integer samples of the curve. Normalizing the period of the cosine and using the M-periodized basis functions

$$\varphi_M(Mt-k) = \sum_{n=-\infty}^{+\infty} \varphi(M(t-n)-k), \tag{6.33}$$

we express the cosine as

$$\cos(2\pi t) = \sum_{k=0}^{M-1} \cos\left[\frac{2\pi k}{M}\right]\varphi_M(Mt-k). \tag{6.34}$$

In a similar way we obtain

$$\sin(2\pi t) = \sum_{k=0}^{M-1} \sin\left[\frac{2\pi k}{M}\right]\varphi_M(Mt-k). \tag{6.35}$$

Plots of the trigonometric functions are shown in Figure 6.6 as well as the circle obtained through the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$.
Due to the choice of $\boldsymbol{\alpha}_{n_0}$, we see that, as we increase the number M of control points, φ converges to the $4th$-order polynomial basis, which corresponds to the Keys interpolator.

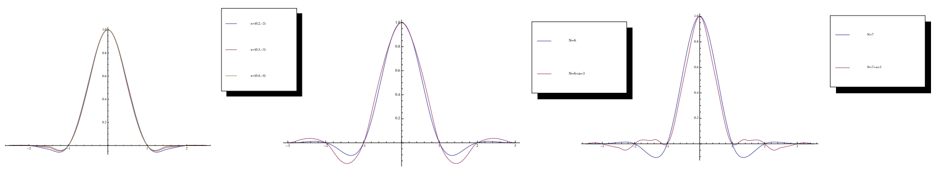


Figure 6.5: Larger support interpolators. Left: Hyperbolic interpolators; 5*th*-order interpolators are shown that were constructed with different lists of poles $\boldsymbol{\alpha}$. Regarding the number of elements in $\boldsymbol{\alpha}$ which is equal to 3, their "smallest" support counterparts (which are shown in Figure 2) are constructed with $N_{\min} = 2(3-1) = 4$, whereas here $N = 5$ was chosen. Middle and right: Effect of including lower-order B-splines in the construction of the interpolating function. If the order of the interpolator is N, then the lowest-order B-spline involved in its construction must be at least $n = \lfloor N/2 \rfloor + 1$ (blue curves). If splines of lower order than $n = \lfloor N/2 \rfloor + 1$ are used in the construction, the interpolating function shows an oscillatory behaviour (red and magenta curves respectively). Here we have used $N = 6$ (middle) and $N = 7$ (right) respectively and hence, the required minimum-order B-spline involved corresponds to $n = n_0' = \lfloor 6/2 \rfloor + 1 = \lfloor 7/2 \rfloor + 1 = 4$ and the corresponding list of poles is $\boldsymbol{\alpha} = (0,0,0,0) = \boldsymbol{\alpha}_{n_0'} = \boldsymbol{\alpha}_{n_0} \cup \mathbf{0}_{(\lfloor N/2 \rfloor + 1) - n_0} = \boldsymbol{\alpha}_{n_0} \cup \mathbf{0}_{4-3} = (0,0,0) \cup (0)$. In the construction of the red and magenta interpolators we have additionally used a B-spline of order $n = 3$ which corresponds to $\boldsymbol{\alpha} = (0,0,0) = \boldsymbol{\alpha}_{n_0}$.

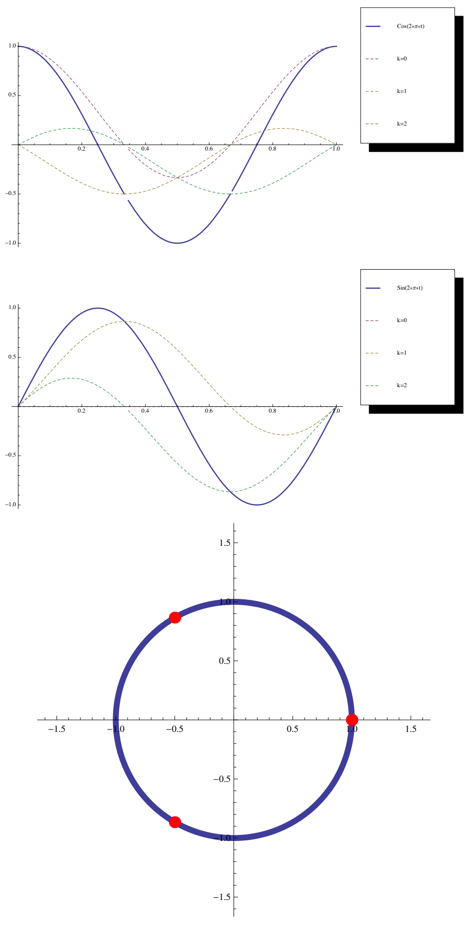


Figure 6.6: The functions $\cos(2\pi t)$ (top left) and $\sin(2\pi t)$ (top right) are shown together respectively (blue curves) with the underlying shifted basis functions (dashed curves) that correspond to $M = 3$ and the shift k. Note that in the construction of the sine, the contribution of the basis function corresponding to the shift $k = 0$ is zero because in (6.72) it is computed through $\sin\left[\frac{2\pi k}{M}\right]\varphi_M(Mt - k) = 0 \cdot \varphi_M(Mt) = 0$. Bottom: Circle obtained with the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$.

Reproduction of Parametric Surfaces

Sphere. We can also reproduce spheres or ellipsoids by using the basis function defined in (6.28). Similar to [86], a possible parameterization of the sphere is given by

$$\begin{aligned}\boldsymbol{\sigma}(u,v) &= \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \begin{pmatrix} \cos(2\pi u)\sin(\pi v) \\ \sin(2\pi u)\sin(\pi v) \\ \cos(\pi v) \end{pmatrix} \\ &= \sum_{k=0}^{M_1-1}\sum_{l=-1}^{M_2+1} \boldsymbol{c}[k,l]\varphi_{M_1}(M_1 u - k)\varphi(M_2 v - l)\end{aligned} \tag{6.36}$$

where $u, v \in [0,1]$ and the control points are given by

$$\boldsymbol{c}[k,l] = \begin{pmatrix} \cos\left[\frac{2\pi k}{M_1}\right]\sin\left[\frac{2\pi l}{2M_2}\right] \\ \sin\left[\frac{2\pi k}{M_1}\right]\sin\left[\frac{2\pi l}{2M_2}\right] \\ \cos\left[\frac{2\pi l}{2M_2}\right] \end{pmatrix}. \tag{6.37}$$

We choose $M_1 = 2M_2$ because the term that depends on u is 1-periodic while the term that dependends on v is 2-periodic. The fact that l only needs to run from -1 to M_2+1 is due to the support of φ, which is equal to 4. The result is shown in the top-left image of Figure 6.7. For comparison, we are also displaying the solutions obtained by the non-interpolatory scheme described in [86]. While the displayed shapes are the same in both cases, the essential difference is that the control points of our proposed interpolators lie on the surface; a property that is useful if the shape needs to be modified interactively. Because each control point is associated to a limited number of compactly supported basis functions, moving its location results in a local modification of the surface.

Torus. The torus can be reproduced in a way similar to the sphere. Again, using the same basis function of our working example (6.28) and the standard parameterization of the torus, we obtain

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} &= \begin{pmatrix} (R + r\cos(2\pi v))\cos(2\pi u) \\ (R + r\cos(2\pi v))\sin(2\pi u) \\ r\sin(2\pi v) \end{pmatrix} \\ &= \sum_{k=0}^{M-1}\sum_{l=0}^{M-1} \boldsymbol{c}[k,l]\varphi_M(Mu-k)\varphi_M(Mv-l),\end{aligned} \tag{6.38}$$

where $u, v \in [0,1]$ and the control points are obtained by sampling as

$$\left.\begin{pmatrix} (R + r\cos(2\pi v))\cos(2\pi u) \\ (R + r\cos(2\pi v))\sin(2\pi u) \\ r\sin(2\pi v) \end{pmatrix}\right|_{u=\frac{k}{M}, v=\frac{l}{M}}. \tag{6.39}$$

The radii R and r of the torus can be chosen in an arbitrary way, without affecting the shape but only the size of the surface. The resulting surface is shown in the middle-left image in Figure 6.7.

"Figure 8" Immersion. The so-called "figure 8" immersion has a slightly more complicated parameterization than the two previous examples. It is given by

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} &= \begin{pmatrix} (r + \cos(\pi u)\sin(2\pi v) - \sin(\pi u)\sin(4\pi v))\cos(2\pi u) \\ (r + \cos(\pi u)\sin(2\pi v) - \sin(\pi u)\sin(4\pi v))\sin(2\pi u) \\ \sin(\pi u)\sin(2\pi v) + \cos(\pi u)\sin(4\pi v) \end{pmatrix} \\ &= \begin{pmatrix} r\cos(2\pi u) + \frac{1}{2}\sin(2\pi v)\big(\cos(\pi u) + \cos(3\pi u)\big) - \frac{1}{2}\sin(4\pi v)\big(\sin(3\pi u) - \sin(\pi u)\big) \\ r\sin(2\pi u) + \frac{1}{2}\sin(2\pi v)\big(\sin(\pi u) + \sin(3\pi u)\big) - \frac{1}{2}\sin(4\pi v)\big(\cos(\pi u) - \cos(3\pi u)\big) \\ \sin(\pi u)\sin(\pi v) + \cos(\pi u)\sin(4\pi v) \end{pmatrix}\end{aligned} \tag{6.40}$$

where $u, v \in [0,1]$ and $r > 2$ is a constant. Hence, we notice that the frequencies associated with the parameter u are π, 2π, and 3π, whereas the frequencies associated with v are 2π and 4π. Therefore, we construct two interpolators, φ_1 with $\boldsymbol{\alpha}_{\varphi_1} = (\frac{\mathrm{j}\pi}{M}, -\frac{\mathrm{j}\pi}{M}, \frac{\mathrm{j}2\pi}{M}, -\frac{\mathrm{j}2\pi}{M}, \frac{\mathrm{j}3\pi}{M}, -\frac{\mathrm{j}3\pi}{M})$ and φ_2 with $\boldsymbol{\alpha}_{\varphi_2} = (\frac{\mathrm{j}2\pi}{M}, -\frac{\mathrm{j}2\pi}{M}, \frac{\mathrm{j}4\pi}{M}, -\frac{\mathrm{j}4\pi}{M})$.

The expression for the tensor-product spline surface is then given by

$$\boldsymbol{\sigma}(u,v) = \sum_{k=0}^{M_1-1}\sum_{l=0}^{M_2-1} \boldsymbol{c}[k,l]\varphi_{1,M_1}(M_1u-k)\varphi_{2,M_2}(M_2v-l), \tag{6.41}$$

where φ_1 and φ_2 have been periodized and $\boldsymbol{c}[k,l] = \boldsymbol{\sigma}(u,v)\big|_{u=k,v=l}$.
The resulting surface is shown at the bottom left in Figure 6.7.

6.2.6 Conclusion

We have characterized a new family of compactly supported interpolators that are based on exponential B-splines. We have shown that they reproduce exponential polynomials while being interpolating. We have illustrated how different members of the family, such as polynomial, trigonometric, or hyperbolic interpolators of different orders can be constructed according to desirable reproduction properties. We have also shown how the proposed interpolators can be used to represent parametric curves and surfaces. The interpolation property ensures that the control points lie on the curve or surface itself. This property is particularly useful for shape representation or manipulation in user-interactive applications. The proposed family of interpolating functions can be seen as a generalization of the polynomial Keys interpolator to higher orders as well as its extension with respect to the reproduction of exponential polynomials.

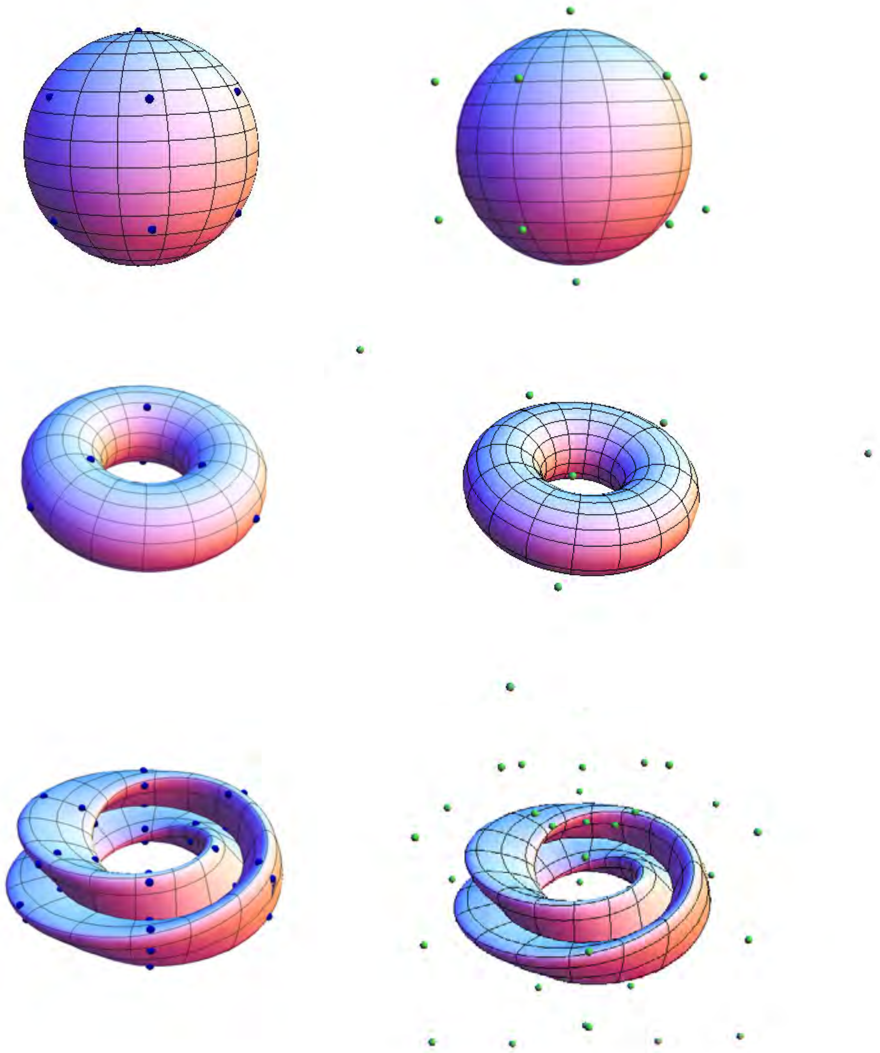


Figure 6.7: Shape comparison. Interpolatory (left, blue) and non-interpolatory control points (right, green). The basis functions that were used to construct the non-interpolatory surfaces correspond to the ones presented in [84]. For the non-interpolatory surfaces it is difficult and non-intuitive to associate a given control point to the specific surface patch that is modified when moving the control point through user-interaction.

6.3 Interpolators for Shape Modeling with Varying Resolution

In this section, we present a modified version of the spline generators constructed in the previous Section 6.2 that allow to modify the resolution of a shape.

In applications that involve interactive curve and surface modeling, the intuitive manipulation of shapes is crucial. For instance, user interaction is facilitated if a geometrical object can be manipulated through control points that interpolate the shape itself. Additionally, models for shape representation often need to provide local shape control and they need to be able to reproduce common shape primitives such as ellipsoids, spheres, cylinders, or tori. We present a general framework to construct families of compactly supported interpolators that are piece-wise-exponential polynomial. They can be designed to satisfy regularity constraints of any order. They enable one to build parametric deformable shape models by suitable combinations of interpolators that form a Riesz basis. The proposed generators are obtained as a sum of exponential B-splines on the half-integer grid. They allow to change the resolution of shapes based on the refinability of B-splines. Our proposed family of generators can be designed to reproduce trigonometric, hyperbolic, and polynomial functions, or combinations of them. We illustrate their use on examples to construct shape models that involve curves and surfaces with applications to interactive modeling and character design.

6.3.1 Introduction

The interactive modeling of curves and surfaces is often desirable in applications that involve the visualization of shapes. Related domains include computer graphics [134, 54, 143, 148, 141, 42], image analysis in biomedical imaging [86, 43, 26, 79], industrial shape design [128, 129, 130] or the modeling of animated surfaces [30]. Shape-modeling frameworks that allow for user interaction can usually be categorized in either *discrete* or *continuous-domain* models. Discrete models are typically based on interpolating polygon meshes or subdivision [32, 149, 35, 38, 133, 150] and they easily allow to locally refine a shape. Subdivision models are also considered as hybrids between discrete and continous-domain models because they iteratively define continuous functions in the limit. However, the limit functions do not always have a closed-form expression [144]. Continuous-domain models allow for *organic* shape modeling and consist of Bézier shapes or spline-based models such as NURBS [17, 25, 151]. They allow one to control shapes locally due to their compactly supported basis functions. However, NURBS generally cannot be smooth and interpolating at the same time, which leads to a non-intuitive manipulation of shapes because NURBS control points do not lie on the boundary of the object.

Motivation and Contribution

Our motivation is the practical need for interpolating functions to be used in user-interactive applications[4] (see Figures 6.8 and 6.9). In this article, we present a general framework that combines the best of the discrete and continuous world: *smooth* and *compactly supported* basis functions, which are defined in the *continuous* domain satisfying the *interpolation condition* and allowing to *vary* the resolution of a constructed shape. In interactive shape modeling these properties allow for the following key attributes:

- Organic shape modeling: smoothness enables a continuously defined tangent plane and Gaussian curvature at any point on the surface, which facilitates realistic texturing and rendering of shapes;
- Local shape control: compact support combined with the interpolation property of the basis functions guarantees precise and direct shape interaction and an intuitive modeling process.
- Detailed surfacing: few parameters are required at the initial stage of modeling, while varying the resolution of the shape allows the user to increase the number of control points when more details are to be modeled.

[4]Videos that illustrate the use and advantage of our proposed framework can be found at http://bigwww.epfl.ch/demo/varying-resolution-interpolator/.

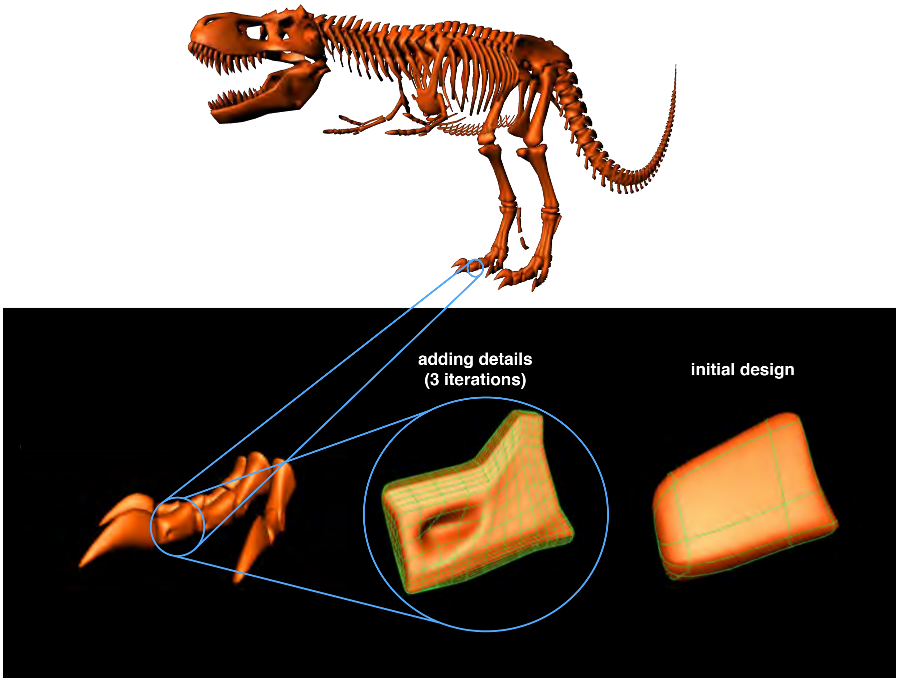
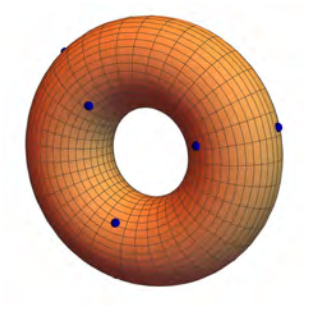
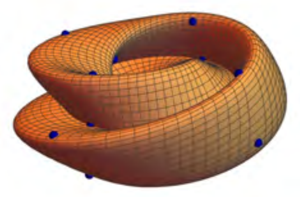
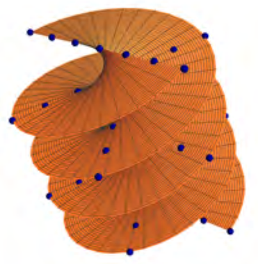


Figure 6.8: Interactive shape modeling for character design. Remodeling of the foot of the “T-rex” is shown. A bone of the middle toe of the right foot is modeled; first, an initial design is achieved with few control points that interpolate the shape (bottom, right). Then, the resolution is increased by applying three refinement iterations in order to have more flexibility to add details to the bone (bottom, middle). Due to convergence of our modified refinement scheme, after three iterations it behaves interpolatory-like.

The “T-rex” has been remodeled after the character designed by Joel Anderson, source: http://joel3d.com/


(a) Torus.


(b) "Figure 8" immersion.


(c) Helicoid.

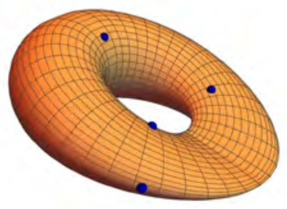

(d) Pinched torus.

Figure 6.9: Parametric surfaces constructed with the proposed family of interpolators. If the parameterization of a shape is known, we provide the formulae to construct the corresponding interpolator in order to represent the shape as detailed in Section 6.3.3. The interpolation property ensures that the control points (blue points) interpolate the surface. This property is particularly useful in user-interactive applications, where a surface is modified by dragging control points (*e.g.* as previously demonstrated in [76, 77, 27, 28])

Our framework consists of a new family of compactly supported interpolators that are linear combinations of shifted exponential B-splines on the half-integer grid. This allows us to harness useful properties of B-splines which can be transferred to the interpolators. We first derive general results and define the construction problem together with necessary constraints and conditions. We then establish relevant reproduction properties and show that, under suitable conditions, the integer shifts of the generators form a Riesz basis, which guarantees a unique and stable representation of the parametric shapes used in practice. The generators are compactly supported. Their degree of regularity can be increased at will.

We further propose an algorithm to change the resolution of the generators which, in turn, allows us to change the resolution of the shapes. This demands that the generators be expressed as a linear combination of finer-resolution basis functions. For this purpose, we propose a refinement scheme associated to our generators by introducing a "pre-refinement" step such that the resulting refinement converges to the interpolator itself. We finally apply the derived general results to characterize a family of symmetric and smooth interpolators that are at least in $\mathcal{C}^1$ and have minimum support.

Finally, we present detailed applications that involve character design (Figure 6.8) as well as the design of idealized parametric shapes (Figure 6.9).

More specifically, Sections 6.3.3 and 6.3.3 are the main technical contributions, whereas in Section 6.3.3 we present practical applications which motivate this article.

Related Work

Recently, a method to build piecewise-polynomial interpolators has been presented in [55, 142]. The present work is the continuation of our previous efforts to, first, generalize the popular Catmull-Rom [16] and Keys [127, 145] interpolators for practical applications [76, 77, 80, 27, 28] and, next, to go one step further and construct families of interpolators that allow for varying the resolution of a shape [152, 153]. Here, the novelty w.r.t. [77] is that the presented framework allows one to vary the resolution of shapes which facilitates shape design in practice as illustrated in Section 6.3.3.

6.3.2 Exponential B-Splines

As in Section 6.2.2, again we exploit the link between exponential B-splines and differential operators. This is crucial to understand the properties of the proposed family of splines. For a more in-depth characterization of exponential B-splines, we refer the reader to [63].

Operators and Reproduction of Null-Space Components

We use a similar notation as introduced in Section 6.2.2. With a slight abuse of notation we denote by $\beta_{\boldsymbol{\alpha}}$ the *centered* (hence, non-causal) exponential B-spline, whose support is $[-n_0/2, n_0/2]$. We have therefore

$$\beta_{\boldsymbol{\alpha}}(t) = \beta^{+}_{\boldsymbol{\alpha}}(t + n_0/2), \tag{6.42}$$

with $\beta^{+}_{\boldsymbol{\alpha}}$ being the *causal* B-spline defined in (6.16). Similarly, we denote by $\Delta_{\boldsymbol{\alpha}}$ the centered discrete operator, whose impulse response is equal to the one of its causal counterpart $\Delta^{+}_{\boldsymbol{\alpha}}$ shifted by $(-n_0/2)$. The reason for introducing centered B-splines and operators is that we shall define interpolators that are symmetric around the origin and, hence, centered.

6.3.3 General Characterization of the Interpolator

We consider generators that are constructed as a sum of half-integer shifted versions of a given exponential B-spline $\beta_{\boldsymbol{\alpha}}$.

Definition 7. For a sequence $\lambda \in \ell_1(\mathbb{Z})$ and $\boldsymbol{\alpha}$ a vector of roots, we define

$$\phi_{\lambda,\boldsymbol{\alpha}}(t) := \sum_{n\in\mathbb{Z}} \lambda[n] \beta_{\boldsymbol{\alpha}}\left(t - \frac{n}{2}\right). \tag{6.43}$$

In the frequency domain, we then have

$$\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega) = \left(\sum_{n\in\mathbb{Z}} \lambda[n] \mathrm{e}^{-\mathrm{j}\omega n/2}\right) \widehat{\beta}_{\boldsymbol{\alpha}}(\omega). \tag{6.44}$$

In what follows, we state the desired mathematical properties that the generator $\phi_{\lambda,\boldsymbol{\alpha}}$ should satisfy.

I The generator $\phi_{\lambda,\boldsymbol{\alpha}}$ is interpolatory, in the sense that, for any function $f \in \text{span}\{\phi_{\lambda,\boldsymbol{\alpha}}(\cdot - k)\}_{k\in\mathbb{Z}}$, we have $f(t) = \sum_{k\in\mathbb{Z}} f(k)\phi_{\lambda,\boldsymbol{\alpha}}(t-k)$. This is equivalent to the interpolation condition

$$\phi_{\lambda,\boldsymbol{\alpha}}(t)\big|_{t=k} = \delta[k], \tag{6.45}$$

where $\delta[k]$ represents the Kronecker delta.

II The generator $\phi_{\lambda,\boldsymbol{\alpha}}$ is compactly supported, which implies that λ has a finite number of non-zero values.

III The function $\phi_{\lambda,\boldsymbol{\alpha}}$ is smooth with at least a continuous derivative.

IV The family of the integer shifts of the generator $\{\phi_{\lambda,\boldsymbol{\alpha}}(\cdot - k)\}_{k\in\mathbb{Z}}$ forms a Riesz basis.

V The generator $\phi_{\lambda,\boldsymbol{\alpha}}$ preserves the reproduction properties of the associated exponential B-spline $\beta_{\boldsymbol{\alpha}}$, in the sense that it is capable of reproducing the null-space components of the operator $\mathrm{L}_{\boldsymbol{\alpha}}$ defined in (6.13).

VI The generator $\phi_{\lambda,\boldsymbol{\alpha}}$ allows one to represent shapes at various resolutions.

We choose equi-spaced half-integer shifts of the exponential B-splines in Definition 7. The reason is that our problem has no solution using only integer shifts under Conditions I), II), and III): There is no smooth and compactly supported interpolator of the form $\sum_{k\in\mathbb{Z}} \lambda[k]\beta_{\boldsymbol{\alpha}}(t-k)$. This can easily be verified, for example by plugging any polynomial B-spline into Definition 7 and using integer shifts while imposing the interpolation conditions: It turns out that there are not enough degrees of freedom to solve the problem due to the compact support of the B-splines as well as the smoothness condition, which forces the degree of the B-spline to be greater than 1. Furthermore, by using half-integer shifts, we guarantee that our solution lives in the spline space of the next finer resolution; a property that can be exploited in practice, as detailed in Section 6.3.3.

Riesz Basis

We consider the space

$$V(\phi_{\lambda,\boldsymbol{\alpha}}) = \left\{ \sum_{n\in\mathbb{Z}} c[n]\phi_{\lambda,\boldsymbol{\alpha}}(\cdot - n), \ c \in \ell_2(\mathbb{Z}) \right\} \tag{6.46}$$

of functions that is generated by the integer shifts of $\phi_{\lambda,\boldsymbol{\alpha}}$. The Riesz basis property ensures that the representation of a function in $V(\phi_{\lambda,\boldsymbol{\alpha}})$ is stable and unique. Thus, the family of functions $\{\phi_{\lambda,\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ should form a Riesz basis of $V(\phi_{\lambda,\boldsymbol{\alpha}})$. We show in this section that this is the case if $\{\beta_{\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ is itself a Riesz basis and if $\phi_{\lambda,\boldsymbol{\alpha}}$ is interpolatory.

Definition 8. The family $\{\varphi_n\}_{n\in\mathbb{Z}}$ of functions forms a Riesz basis if

$$A\|c\|_{\ell_2(\mathbb{Z})} \leq \left\| \sum_{n\in\mathbb{Z}} c[n]\varphi_n \right\|_{L_2(\mathbb{R})} \leq B\|c\|_{\ell_2(\mathbb{Z})} \tag{6.47}$$

for some constants $A, B > 0$ and any sequence $c = (c[n])_{n\in\mathbb{Z}} \in \ell_2(\mathbb{Z})$.

When $\varphi_n = \varphi(\cdot - n)$, (6.47) is equivalent to the Fourier-domain condition

$$A^2 \leq \sum_{k\in\mathbb{Z}} |\widehat{\varphi}(\omega - 2k\pi)|^2 \leq B^2 \tag{6.48}$$

for any $\omega \in \mathbb{R}$ [81]. The family $\{\beta_{\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis when $\boldsymbol{\alpha}$ is such that $\alpha_n - \alpha_m \neq 2k\pi\mathrm{j}$, $k \in \mathbb{Z}$, for any pair of distinct purely imaginary roots $\alpha_m, \alpha_n \in \boldsymbol{\alpha}$ [63, Theorem 1].

Proposition 15. *Let $\boldsymbol{\alpha}$ be such that $\alpha_n - \alpha_m \neq 2k\pi\mathrm{j}$, $k \in \mathbb{Z}$, for any pair of distinct purely imaginary roots $\alpha_m, \alpha_n \in \boldsymbol{\alpha}$. For any sequence $\lambda \in \ell_1(\mathbb{Z})$, if the basis function $\phi_{\lambda,\boldsymbol{\alpha}}$ is interpolatory, then the family $\{\phi_{\lambda,\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis.*

The proof is given in 6.4.1 as well as an estimate of the Riesz Bounds.

Reproduction Properties

Proposition 16. *Let $\boldsymbol{\alpha}$ be a vector of roots. We assume that $\lambda \in \ell_1(\mathbb{Z})$ satisfies the conditions*

$$\sum_{n\in\mathbb{Z}} |\lambda[n]| \mathrm{e}^{-\alpha n/2} < \infty, \tag{6.49}$$

$$\sum_{n\in\mathbb{Z}} \lambda[n] \mathrm{e}^{-\alpha n/2} \neq 0 \tag{6.50}$$

for every $\alpha \in \boldsymbol{\alpha}$. Then, the basis function $\phi_{\lambda,\boldsymbol{\alpha}}$ has the same reproduction properties as the corresponding exponential B-spline $\beta_{\boldsymbol{\alpha}}$. In particular, it reproduces the exponential polynomials

$$t^{n-1}\mathrm{e}^{\alpha_{(m)}t} \tag{6.51}$$

for $m = 1, \ldots, n_d$ and $n = 1, \ldots, n_{(m)}$, where we use the notations defined in Section 6.2.2.

Note that (6.49) is always satisfied as soon as $\phi_{\lambda,\boldsymbol{\alpha}}$ is compactly supported. The proof of Proposition 16 is given in Appendix 6.4.2.

Regularity

From Definition 7, it immediately follows that $\phi_{\lambda,\boldsymbol{\alpha}}$ has the same regularity as the exponential B-spline $\beta_{\boldsymbol{\alpha}}$ if $\lambda \neq 0$. Hence, $\phi_{\lambda,\boldsymbol{\alpha}}$ belongs to $\mathcal{C}^{n_0-2}$ [63, Section III-A].

Varying the Resolution of the Generator

The causal exponential B-spline $\beta^+_{\boldsymbol{\alpha}}$ is refinable, in the sense that its dilation by an integer m can be expressed as a linear combination of $\beta^+_{\boldsymbol{\alpha}/m}(\cdot - k)$. This is what we refer to as the *resolution* of the basis function. We shall see how this property translates for the function $\phi_{\lambda,\boldsymbol{\alpha}}$. For this purpose, we first revisit the m-scale relation for exponential B-splines. For convenience, we express the corresponding terms with respect to causal (non-centered) B-splines. In practice, we always consider symmetric interpolators $\phi_{\lambda,\boldsymbol{\alpha}}$ with support $[-(n_0-1), n_0-1]$ (see Section 6.3.3). Therefore, we define the shifted and causal version of the interpolator as

$$\phi^+_{\lambda,\boldsymbol{\alpha}}(t) = \phi_{\lambda,\boldsymbol{\alpha}}(t-(n_0-1)). \tag{6.52}$$

Every causal formula is easily adapted to the centered case by applying a shift operation similar to (6.52). We follow the notations of [63], where an in-depth discussion on the refinability of exponential B-splines can be found.

As shown in [63, Section IV-D], the dilation by an integer $m \in \mathbb{N}\setminus\{0\}$ of an exponential B-spline is expressed in the space domain as

$$\beta^+_{\boldsymbol{\alpha}}\left(\frac{t}{m}\right) = \sum_{k\in\mathbb{Z}} h_{\frac{\boldsymbol{\alpha}}{m},m}[k]\beta^+_{\frac{\boldsymbol{\alpha}}{m}}(t-k), \tag{6.53}$$

where the *refinement filter* $h_{\boldsymbol{\alpha},m}$ is specified by its Fourier transform

$$H_{\boldsymbol{\alpha},m}(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{m^{n_0-1}}\prod_{n=1}^{n_0}\left(\sum_{k=0}^{m-1}\mathrm{e}^{\alpha_n k}\mathrm{e}^{-\mathrm{j}k\omega}\right). \tag{6.54}$$

As we shall see, it is impossible to establish a similar relation for the interpolator $\phi^+_{\lambda,\boldsymbol{\alpha}}$. However, we can exploit the refinability of the corresponding spline $\beta^+_{\boldsymbol{\alpha}}$ to express the dilation of $\phi^+_{\lambda,\boldsymbol{\alpha}}$.
For $\boldsymbol{\alpha}$ a vector of roots, $\lambda \in \ell_1(\mathbb{Z})$, and m_0 an *even* integer, we define the digital *pre-filter* $g_{\lambda,\boldsymbol{\alpha},m_0}$ by its Fourier transform

$$G_{\lambda,\boldsymbol{\alpha},m_0}(\mathrm{e}^{\mathrm{j}\omega}) = \mathrm{e}^{-\mathrm{j}\omega m_0(n_0/2-1)}\left(\sum_{n\in\mathbb{Z}}\lambda[n]\mathrm{e}^{-\mathrm{j}\omega n m_0/2}\right)H_{\frac{\boldsymbol{\alpha}}{m_0},m_0}(\mathrm{e}^{\mathrm{j}\omega}). \tag{6.55}$$

The term $\mathrm{e}^{-\mathrm{j}\omega m_0(n_0/2-1)}$ is due to the fact that $\beta_{\boldsymbol{\alpha}}$ and $\phi_{\lambda,\boldsymbol{\alpha}}$ do not have the same support in general. The pre-filter allows us to express $\phi^+_{\lambda,\boldsymbol{\alpha}}$ dilated by m_0 as a linear combination of the refined shifted B-splines $\beta^+_{\frac{\boldsymbol{\alpha}}{m_0}}(\cdot-k)$. Note that $G_{\lambda,\boldsymbol{\alpha},m_0}$ is a valid Fourier transform of a digital filter (*i.e.*, a function of $\mathrm{e}^{\mathrm{j}\omega}$) only for even m_0.

Proposition 17. *Let $\boldsymbol{\alpha}$ be a vector of roots, $\lambda \in \ell_1(\mathbb{Z})$, and m_0 be an even integer. Then, we have*

$$\phi^+_{\lambda,\boldsymbol{\alpha}}\left(\frac{t}{m_0}\right) = \sum_{k\in\mathbb{Z}} g_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0}[k]\beta^+_{\frac{\boldsymbol{\alpha}}{m_0}}(t-k). \tag{6.56}$$

The proof is given in Appendix 6.4.3.

Modified Refinement Scheme Based on Exponential B-Splines. Using Proposition 17, we are able to express a function which is constructed with the interpolator $\phi^+_{\lambda,\boldsymbol{\alpha}}$ in an exponential B-spline basis. Starting with the samples $c[k] = f(t)|_{t=k\in\mathbb{Z}}$ of a continuously defined function $f(\cdot)$ that can be *perfectly* reconstructed, *i.e.*, $f \in \mathrm{span}\{\phi^+_{\lambda,\boldsymbol{\alpha}}(\cdot - k)\}_{k\in\mathbb{Z}}$, we have

$$\begin{aligned} f(t) &= \sum_{k\in\mathbb{Z}} c[k]\phi^+_{\lambda,\boldsymbol{\alpha}}(t-k) \\ &= \sum_{k\in\mathbb{Z}} c[k] \sum_{l\in\mathbb{Z}} g_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0}[l]\beta^+_{\frac{\boldsymbol{\alpha}}{m_0}}(m_0(t-k)-l) \\ &= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} c[k]g_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0}[l]\beta^+_{\frac{\boldsymbol{\alpha}}{m_0}}(m_0t-m_0k-l) \\ &= \sum_{l\in\mathbb{Z}} c_0[l]\beta^+_{\frac{\boldsymbol{\alpha}}{m_0}}(m_0t-l) \end{aligned} \tag{6.57}$$

with

$$c_0[l] = \left(c_{\uparrow m_0} * g_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0}\right)[l], \tag{6.58}$$

where $\uparrow m_0$ denotes the upsampling by a factor m_0.

Equation (6.57) shows that a function that is originally expressed in the basis generated by $\phi^+_{\lambda,\boldsymbol{\alpha}}$ can be expressed in a corresponding exponential B-spline basis with respect to a finer grid. This suggests that, after having performed the change of basis described by (6.57), the resolution of f can be further refined by applying the standard iterative B-spline refinement rules. At this point, it is interesting to take a deeper look into the relation between the interpolated function f and the sequence

c of samples as we iteratively refine it. As will become apparent in the application-oriented Section 6.3.3, a parametric shape is described by coordinate functions whose samples build 2D or 3D vectors of *control points.* Repositioning of these control points allows us to locally modify the shape, while the iterative refinement of the control points allows us to iteratively increase the local control over the shape. Hence, for practical purposes, it is convenient to study the convergence of the refinement process as the number of iterations becomes large. Proposition 18 describes the refinement scheme and provides the corresponding convergence result.

Proposition 18. *Let $\boldsymbol{\alpha}$ be a vector of roots and $\lambda \in \ell_1(\mathbb{Z})$. For a continuous function f with samples $f(t)|_{t=k\in\mathbb{Z}} = c[k]$ and the integers m, m_0, with m_0 being even, we consider the iterative scheme specified by*

1. *pre-filter step: $c_0[k] = (g_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0} * c_{\uparrow m_0})[k]$;*

2. *iterative steps: for $n \geq 1$, $c_n[k] = (h_{\frac{\boldsymbol{\alpha}}{m_0 m^n},m} * (c_{n-1})_{\uparrow m})[k]$.*

Then, the iterative scheme is convergent, in the sense that

$$\sum_{k\in\mathbb{Z}} c_n[k]\delta\left(m_0 m^n t - k\right) \underset{n\to\infty}{\longrightarrow} f(t). \tag{6.59}$$

The proof is given in Appendix 6.4.4.

Example. We illustrate how to refine the resolution of a circle by applying Proposition 18. To efficiently take advantage of the interpolation property, we apply the "pre-refinement" step (6.58) at the first iteration. For the subsequent iterations, we apply the standard refinement given by (6.54) as described by Proposition 18. By doing so, we see that the iterative scheme converges towards the circle $\boldsymbol{r}(t) = \sum_{l\in\mathbb{Z}} \boldsymbol{c}_0[l]\beta^{+}_{\frac{\boldsymbol{\alpha}}{m_0}}(m_0 t - l) = \sum_{k\in\mathbb{Z}} \boldsymbol{r}[k]\phi^{+}_{\boldsymbol{\alpha}}(t-k)$. The result of the algorithm is shown in Figure 6.10. In Section 6.3.3, we provide the exact details on how to reconstruct the circle with our framwork.

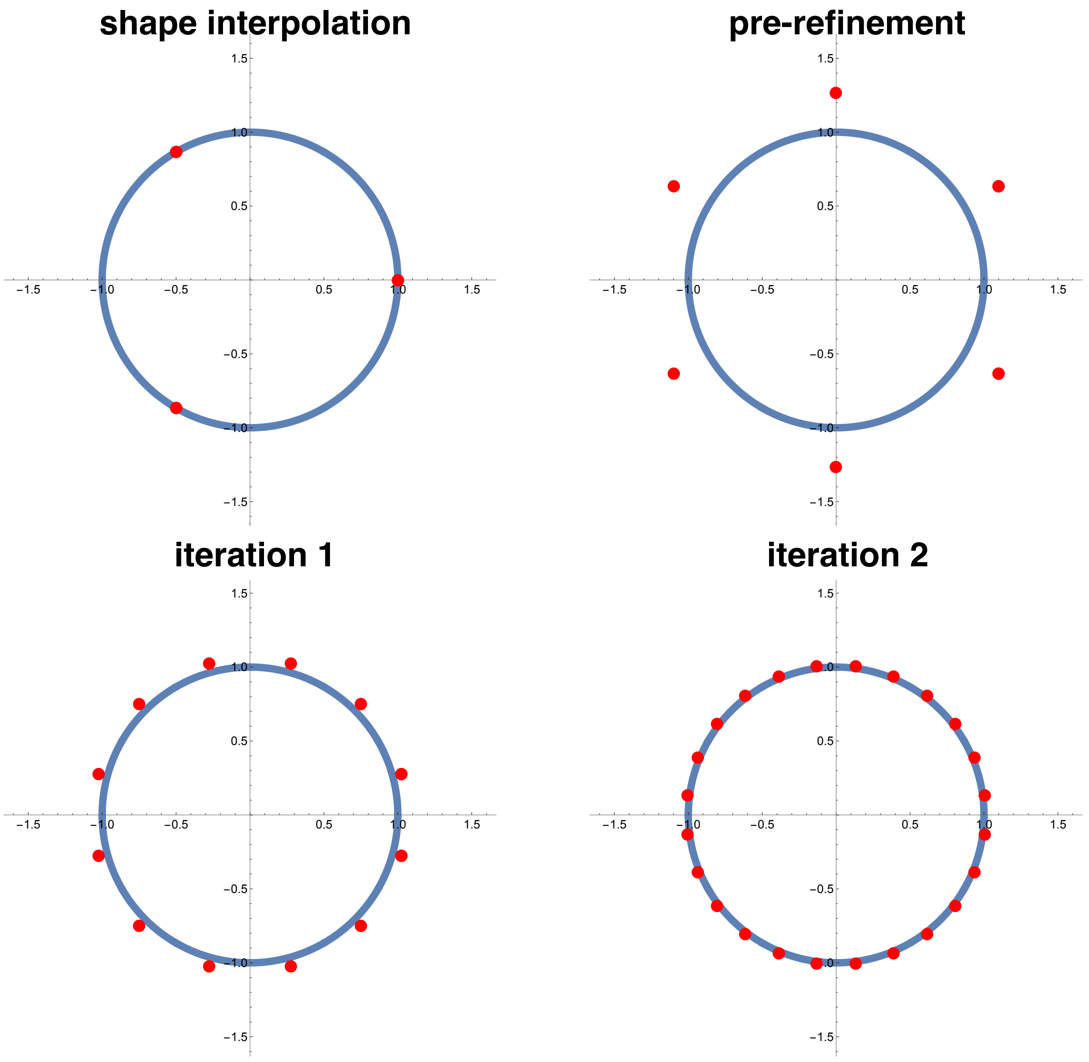


Figure 6.10: Refined circle. The parametric circle is first constructed using the proposed interpolator and $\boldsymbol{\alpha} = (0, \frac{2j\pi}{3}, -\frac{2j\pi}{3})$ (top left). At the first iteration, the "pre-refinement" mask is applied to the initial control points according to (6.58) (top right), whereas at the subsequent iterations the standard refinement mask for exponential B-splines (6.54) is applied (bottom, from left to right). In the bottom right, we see how the iterative process converges towards the continuously defined circle.

Construction of a Family of Compactly Supported Interpolators in Practice

There exists no exponential B-spline $\beta_{\boldsymbol{\alpha}}$ that is interpolatory and smooth (*i.e.*, at least in $\mathcal{C}^1$) at the same time. Our goal is to construct a compactly supported generator function that has the same smoothness and reproduction properties as $\beta_{\boldsymbol{\alpha}}$ while also being interpolatory. In order to meet the smoothness constraints, we require the number of elements of $\boldsymbol{\alpha}$ to be $n_0 \geq 3$ in accordance with Section 6.3.3. Furthermore, we want the interpolator to be symmetric, which implies that the elements of $\boldsymbol{\alpha}$ are either zero or come in pairs with opposite signs [63]. Using Definition 7 and the conditions described in Section 6.3.3, we are looking for the interpolator with minimal support.

Introductory Example: The Quadratic B-Spline. We illustrate the concept with a simple example that uses quadratic polynomial B-splines, which are constructed with $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 = (0,0,0)$ in (6.16) and whose support is of size 3. The interpolation constraint combined with the half-integer shifts demand that λ contains at least three non-zero values to have enough degrees of freedom. This also implies that the minimum-support interpolator is constructed with no more than three non-zero elements of λ. To satisfy the symmetry constraints, we center the shifted B-splines around the origin and enforce $\lambda[1] = \lambda[-1]$. Hence, our generator must take the form

$$\begin{aligned}\phi_{\lambda,\boldsymbol{\alpha_0}}(t) &= \lambda[1]\beta_{\boldsymbol{\alpha_0}}(t-\frac{1}{2}) + \lambda[0]\beta_{\boldsymbol{\alpha_0}}(t) + \lambda[-1]\beta_{\boldsymbol{\alpha_0}}(t+\frac{1}{2}) \\ &= \lambda[0]\beta_{\boldsymbol{\alpha_0}}(t) + \lambda[1]\big(\beta_{\boldsymbol{\alpha_0}}(t-\frac{1}{2}) + \beta_{\boldsymbol{\alpha_0}}(t+\frac{1}{2})\big).\end{aligned} \tag{6.60}$$

Since $\boldsymbol{\alpha}_0$ has $n_0 = 3$ elements, the support of the interpolator is $N = 2(n_0 - 1) = 4$. The interpolator itself is supported in $[-(n_0-1),(n_0-1)] = [-2,2]$. The interpolation condition is expressed as

$$\begin{cases}\phi_{\lambda,\boldsymbol{\alpha_0}}(0) = 1 \\ \phi_{\lambda,\boldsymbol{\alpha_0}}(1) = 0.\end{cases}$$

We define the matrix

$$\mathbf{A}_{\alpha_0} = \begin{pmatrix} \beta_{\boldsymbol{\alpha}_0}(0) & \beta_{\boldsymbol{\alpha}_0}(-1/2) + \beta_{\boldsymbol{\alpha}_0}(1/2) \\ \beta_{\boldsymbol{\alpha}_0}(1) & \beta_{\boldsymbol{\alpha}_0}(1-1/2) + \beta_{\boldsymbol{\alpha}_0}(1+1/2) \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & 1 \\ \frac{1}{8} & \frac{1}{2} \end{pmatrix}$$

and rewrite the interpolation constraint as $(\lambda[0], \lambda[1]) = \mathbf{A}_{\boldsymbol{\alpha}_0}^{-1}(1,0) = (1, -\frac{1}{2})$. The resulting interpolator is shown in Figure 6.11.

The General Case. In what follows, we only consider vector of poles $\boldsymbol{\alpha}$ for which $\alpha_n - \alpha_m \neq 2k\pi\mathrm{j}$, $k \in \mathbb{Z}$ for all pairs of distinct, purely imaginary roots $\alpha_m, \alpha_n \in \boldsymbol{\alpha}$ (Riesz Basis property). We generalize the above example to construct symmetric and compactly supported interpolators of any order and that are of the form

$$\phi_{\lambda,\boldsymbol{\alpha}}(t) := \lambda[0]\beta_{\boldsymbol{\alpha}}(t) + \sum_{n=1}^{n_0-2} \lambda[n] \left(\beta_{\boldsymbol{\alpha}}(t-n/2) + \beta_{\boldsymbol{\alpha}}(t+n/2)\right), \tag{6.61}$$

whose support is included[5] in $[-N/2, N/2] = [-(n_0-1), n_0-1]$. We easily pass from the general representation (6.43) to (6.61), adapted to the symmetric and compactly supported case, by setting $\lambda[n] = 0$ when $|n| \geq n_0 - 1$ (support condition) and $\lambda[-n] = \lambda[n]$ for every n (symmetry condition).

The function $\phi_{\lambda,\boldsymbol{\alpha}}$ is interpolatory if and only if

$$\phi_{\lambda,\boldsymbol{\alpha}}(0) = 1 \text{ and } \phi_{\lambda,\boldsymbol{\alpha}}(1) = \cdots = \phi_{\lambda,\boldsymbol{\alpha}}(n_0-2) = 0. \tag{6.62}$$

This defines a linear system with (n_0-1) unknown non-zero elements of λ, $\{\lambda[0], \ldots, \lambda[n_0-2]\}$, and (n_0-1) equations. The system (6.62) has a solution if the matrix $\mathbf{A}_{\boldsymbol{\alpha}} \in \mathbb{R}^{(n_0-1)^2}$ defined for $k, l = 0, \ldots, (n_0-2)$ by

$$[\mathbf{A}_{\boldsymbol{\alpha}}]_{k+1,l+1} = \begin{cases} \beta_{\boldsymbol{\alpha}}(k) & \text{if } \quad l = 0 \\ \beta_{\boldsymbol{\alpha}}(k-l/2) + \beta_{\boldsymbol{\alpha}}(k+l/2) & \text{else} \end{cases} \tag{6.63}$$

is invertible. In this case, we have

$$\lambda = (\lambda[0], \ldots, \lambda[n_0-2]) = \mathbf{A}_{\boldsymbol{\alpha}}^{-1}(1, 0, \ldots, 0). \tag{6.64}$$

[5]The support is exactly $[-N/2, N/2]$ when $\lambda[n]$ is non-zero for $n = 0, \ldots, (n_0-2)$, which is always the case in the examples we have considered.

Knowing $\boldsymbol{\alpha}$, we can easily check if the matrix $\mathbf{A}_{\boldsymbol{\alpha}}$ is invertible, which is the case for all the examples that we tested (we have already seen that it is true for $\boldsymbol{\alpha} = (0, 0, 0)$ is Section 6.3.3). From (6.64), we see that λ is completely determined by $\boldsymbol{\alpha}$. This motivates Definition 9.

Definition 9. Let $\boldsymbol{\alpha}$ be a vector of roots whose elements are either zero or come in pairs with opposite signs. If the matrix $\mathbf{A}_{\boldsymbol{\alpha}}$ defined in (6.63) is invertible, then the interpolatory basis function $\phi_{\boldsymbol{\alpha}}$ is defined as

$$\phi_{\boldsymbol{\alpha}} := \phi_{\lambda,\boldsymbol{\alpha}}, \tag{6.65}$$

with λ defined by (6.64).

We conjecture that the matrix $\mathbf{A}_{\boldsymbol{\alpha}}$ is always invertible, and that we always can define an interpolator $\phi_{\boldsymbol{\alpha}}$ for any list of roots $\boldsymbol{\alpha}$. In the remaining of this article, we assume that $\mathbf{A}_{\boldsymbol{\alpha}}$ is invertible and, therefore, that $\phi_{\boldsymbol{\alpha}}$ is well-defined. Under this assumption, the unicity of the vector λ ensures that the interpolator $\phi_{\boldsymbol{\alpha}}$ in Definition 9 has minimal support among the interpolators of the form (6.43).

In practice, the type of interpolator that needs to be constructed depends on the parametric shape that is represented. For instance, for a rectangular surface, a polynomial interpolator is required and the vector $\boldsymbol{\alpha}$ of roots will have to consist of zeros. If instead we aim at representing circles, spheres, or ellipsoids (see Section 6.3.3), whose coordinate functions are trigonometric, we need to construct interpolators that preserve sinusoids. Therefore, $\boldsymbol{\alpha}$ will contain pairs of purely imaginary roots with opposite signs. Similarly, we can reproduce hyperbolic shapes by picking an $\boldsymbol{\alpha}$ that contains pairs of real roots with opposite signs. If an interpolator is required to reproduce both trigonometric and polynomial shapes, *e.g.*, to construct a cylinder, then the corresponding polynomial and trigonometric root vectors are concatenated to construct $\boldsymbol{\alpha}$. Examples of different interpolators are shown in Figure 6.11.

We now summarize the properties of the generator $\phi_{\boldsymbol{\alpha}}$ for $\boldsymbol{\alpha}$ a vector of roots of size $n_0 \geq 3$ such that $\alpha_n - \alpha_m \neq 2k\pi\mathrm{j}$, $k \in \mathbb{Z}$, for any pair of distinct purely imaginary roots $\alpha_m, \alpha_n \in \boldsymbol{\alpha}$. These properties are in accordance with Conditions I to VI in Section 6.3.3.

- The function $\phi_{\boldsymbol{\alpha}}$ is interpolatory.
- The function $\phi_{\boldsymbol{\alpha}}$ is compactly supported in $[-(n_0 - 1), n_0 - 1]$.

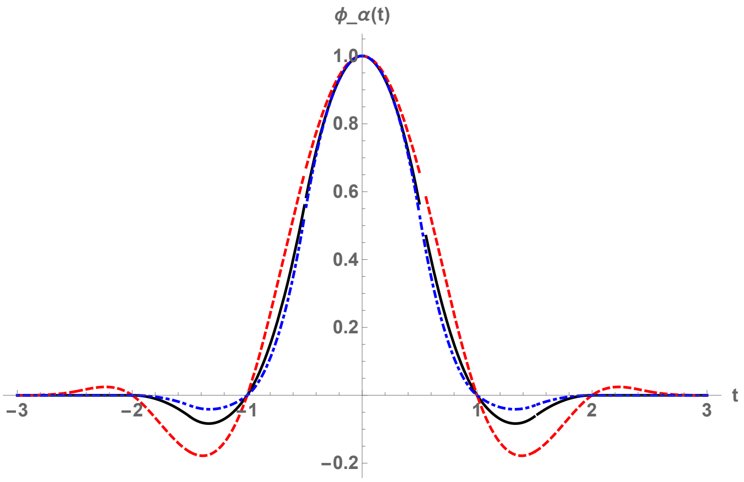


Figure 6.11: Different types of interpolators: polynomial interpolator (black, solid curve) with $\boldsymbol{\alpha} = (0, 0, 0)$. The number of poles is equal to 3. Trigonometric interpolator (red, dashed curve): the non-zero poles are purely imaginary and come in pairs with opposite signs (*e.g.*, $\boldsymbol{\alpha} = (0, 0, \frac{\mathrm{j}2\pi}{3}, -\frac{\mathrm{j}2\pi}{3}))$. Hyperbolic interpolator (blue, dot-dashed curve): the non-zero poles are real and come in pairs with opposite signs (*e.g.*, $\boldsymbol{\alpha} = (0, \frac{2\pi}{3}, -\frac{2\pi}{3}))$.

- The function $\phi_{\boldsymbol{\alpha}}$ has the minimal support among the interpolators that are linear combinations of shifted exponential B-splines on the half-integer grid.
- The function $\phi_{\boldsymbol{\alpha}}$ is in $\mathcal{C}^{n_0-2}$ and therefore, at least in $\mathcal{C}^1$.
- The family $\{\phi_{\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ is a Riesz basis.
- The family $\{\phi_{\boldsymbol{\alpha}}(\cdot - n)\}_{n\in\mathbb{Z}}$ reproduces the null-space components of the operator $\mathrm{L}_{\boldsymbol{\alpha}}$ (see Section 6.3.2).
- The function $\phi_{\boldsymbol{\alpha}}$ is refinable in the sense explained in Section 6.3.3.

Remark. The presented interpolators are not (entirely) positive (see Figure 6.11) and thus, do not satisfy the convex-hull-property. However, the popularity of the Catmull-Rom splines [16] in computer graphics shows that in interactive shape modeling, one prefers to use interpolators at the expense of the convex-hull property. As a side note, our experiments have shown that when using more than 90'000 control points, oscillations on the surface of a shape might start to appear. Since in a typical interactive shape modeling process the number of control points is much smaller, *e.g.*, between 10 and 500, the oscillating phenomena are negligible.

Applications

In this section, we show how parametric curves and surfaces are constructed using the proposed spline bases. Such shapes can be constructed independently of the number of control points. This makes them particularly useful for deformable models where, starting from an initial configuration, one aims at approximating a target shape with arbitrary precision [81].

Reproduction of Idealized Shapes. We consider curves and surfaces that are described by the coordinate functions $r_x(t)$, $r_y(t)$, and $r_z(t)$, with $t \in \mathbb{R}$. The coordinate functions are expressed by a linear combination of weighted integer shifts of the generator $\phi_{\boldsymbol{\alpha}}$. Due to the interpolation property of the generator, the weights simply correspond to the samples of the coordinate functions. Such a parametric curve is expressed as

$$\boldsymbol{r}(t) = \begin{pmatrix} r_x(t) \\ r_y(t) \\ r_z(t) \end{pmatrix} = \sum_{k\in\mathbb{Z}} \boldsymbol{r}[k]\phi_{\boldsymbol{\alpha}}(t-k), \tag{6.66}$$

where the coefficients $\boldsymbol{r}[k] = (r_x[k], r_y[k], r_z[k])$ with $k \in \mathbb{Z}$ are the *control points*. The curve (7.1) can be locally modified by changing the position of a single control point. The shapes that $\boldsymbol{r}$ can adopt (*e.g.*, polynomial, circular, elliptic) depend on the properties of the generator.

One can also extend the curve model (7.1) to represent separable tensor-product surfaces. In this case, a surface $\boldsymbol{\sigma}$ is parameterized by $u, v \in \mathbb{R}$ as

$$\begin{aligned}\boldsymbol{\sigma}(u,v) &= \begin{pmatrix}\boldsymbol{\sigma}_x(u,v)\\ \boldsymbol{\sigma}_y(u,v)\\ \boldsymbol{\sigma}_z(u,v)\end{pmatrix} = \begin{pmatrix} r_{1,x}(u)\cdot r_{2,x}(v)\\ r_{1,y}(u)\cdot r_{2,y}(v)\\ r_{1,z}(u)\cdot r_{2,z}(v)\end{pmatrix}\\ &= \sum_{k\in\mathbb{Z}} \boldsymbol{r}_1[k]\phi_{\boldsymbol{\alpha}_1}(u-k) \times \sum_{l\in\mathbb{Z}} \boldsymbol{r}_2[l]\phi_{\boldsymbol{\alpha}_2}(v-l) \qquad (6.67)\\ &= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \underbrace{\boldsymbol{r}_1[k]\times\boldsymbol{r}_2[l]}_{\boldsymbol{\sigma}[k,l]} \phi_{\boldsymbol{\alpha}_1}(u-k)\phi_{\boldsymbol{\alpha}_2}(v-l),\end{aligned}$$

where "×" denotes the element-wise multiplication of two vectors. Finally, one generalizes (7.2) to represent surfaces with a non-separable parameterization as

$$\boldsymbol{\sigma}(u,v) = \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \boldsymbol{\sigma}[k,l]\phi_{\boldsymbol{\alpha}_1}(u-k)\phi_{\boldsymbol{\alpha}_2}(v-l). \qquad (6.68)$$

Reproduction of Ellipses. We now explicitly show how ellipses can be reproduced using our proposed interpolatory basis functions. To construct the ellipses as a function of the number of control points M, we choose $\boldsymbol{\alpha} = \left(0, \frac{j2\pi}{M}, -\frac{j2\pi}{M}\right)$ and, hence, $n_0 = 3$. The interpolator is obtained by Definition 9 and by solving the corresponding system of equations (6.62). The non-zero values of the sequence λ are

$$\lambda[0] = \frac{\pi^2 \csc^2\left(\frac{\pi}{2M}\right)\sec\left(\frac{\pi}{M}\right)}{4M^2}$$

and

$$\lambda[1] = \lambda[-1] = -\frac{\pi^2 \csc\left(\frac{\pi}{M}\right)\csc\left(\frac{2\pi}{M}\right)}{M^2}.$$

To reproduce $\cos\left(\frac{2\pi}{M}\cdot\right)$, we take advantage of the interpolation property, which yields

$$\cos\left(\frac{2\pi}{M}t\right) = \sum_{k\in\mathbb{Z}} \frac{\mathrm{e}^{\mathrm{j}\frac{2\pi}{M}k} + \mathrm{e}^{-\mathrm{j}\frac{2\pi}{M}k}}{2}\phi_{\boldsymbol{\alpha}}(t-k), \qquad (6.69)$$

where the coefficients are the integer samples of the curve. Normalizing the period of the cosine and using the M-periodized basis functions

$$\phi_{\boldsymbol{\alpha},M}(t) = \sum_{k\in\mathbb{Z}} \phi_{\boldsymbol{\alpha}}(t - Mk), \tag{6.70}$$

we express the cosine as

$$\cos(2\pi t) = \sum_{k=0}^{M-1} \cos\left(\frac{2\pi k}{M}\right) \phi_{\boldsymbol{\alpha},M}(Mt - k). \tag{6.71}$$

In a similar way we obtain

$$\sin(2\pi t) = \sum_{k=0}^{M-1} \sin\left(\frac{2\pi k}{M}\right) \phi_{\boldsymbol{\alpha},M}(Mt - k). \tag{6.72}$$

Plots of the trigonometric functions are shown in Figure 6.12 as well as the circle obtained through the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$. Ellipses can be constructed by simply applying an affine transformation to the circle $\boldsymbol{r}$. In order to guarantee a representation that does not depend on the location and orientation of the curve, it must be *affine invariant*. This is ensured if the interpolator satisfies the partition of unity $\sum_{k\in\mathbb{Z}} \phi_{\boldsymbol{\alpha},M}(\cdot - k) = 1$, which implies that it must reproduce zero-degree polynomials (*i.e.*, the constants). Hence, we need that $0 \in \boldsymbol{\alpha}$.

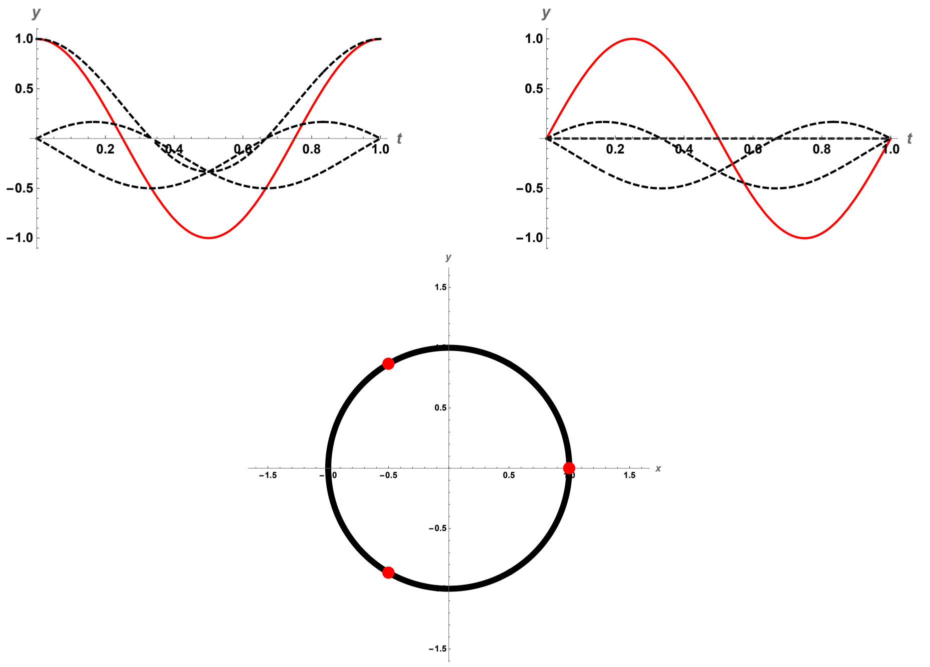


Figure 6.12: Top row: reproduction of the cosine (left) and sine (right) for $M = 3$. The weighted and shifted basis functions are represented by dashed lines. The reconstructed parametric circle is shown in the bottom row (black) with the interpolatory control points (shown in red on the boundary of the circle).

Reproduction of the Roman surface. The standard parameterization of the Roman surface is specified by

$$\boldsymbol{\sigma}(u,v) = \begin{pmatrix} \frac{1}{2}r^2\cos(2\pi u)\sin(4\pi v) \\ \frac{1}{2}r^2\sin(2\pi u)\sin(4\pi v) \\ r^2\cos(2\pi u)\sin(2\pi u)\cos^2(2\pi v) \end{pmatrix} \tag{6.73}$$

$$= \begin{pmatrix} \frac{1}{2}r^2\cos(2\pi u)\sin(4\pi v) \\ \frac{1}{2}r^2\sin(2\pi u)\sin(4\pi v) \\ \frac{1}{4}r^2\sin(4\pi u)(1+\cos(4\pi v)) \end{pmatrix}, \quad (u,v)\in\mathbb{R}^2. \tag{6.74}$$

We parameterize (6.73) as a tensor-product surface of the form (7.2) and denote by M_1 and M_2 the number of control points related to $\phi_{\boldsymbol{\alpha}_1}$ and $\phi_{\boldsymbol{\alpha}_2}$, respectively. The surface is trigonometric in u and v. Hence, we choose to construct the interpolators $\phi_{\boldsymbol{\alpha}_1}$ and $\phi_{\boldsymbol{\alpha}_2}$ with $\boldsymbol{\alpha}_1 = \left(\frac{2j\pi}{M_1}, \frac{-2j\pi}{M_1}, \frac{4j\pi}{M_1}, \frac{-4j\pi}{M_1}\right)$ and $\boldsymbol{\alpha}_2 = \left(0, \frac{4j\pi}{M_2}, \frac{-4j\pi}{M_2}\right)$ to express (6.73) as

$$\boldsymbol{\sigma}(u,v) = \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \boldsymbol{\sigma}[k,l]\phi_{\boldsymbol{\alpha}_1}(M_1u-k)\phi_{\boldsymbol{\alpha}_2}(M_2v-l). \tag{6.75}$$

In order to satisfy the relation $\alpha_n - \alpha_m \neq 2k\pi\mathrm{j}$, $k\in\mathbb{Z}$ for all pairs of distinct, purely imaginary roots, we choose $M_1 = M_2 = 5$. To construct $\phi_{\boldsymbol{\alpha}_1}$, we see that $n_0 = 4$ and $N = 2(n_0-1) = 6$. Hence, the support of $\phi_{\boldsymbol{\alpha}_1}$ is of size 6. Following (6.61), the interpolator is expressed as

$$\phi_{\boldsymbol{\alpha}_1}(t) = \lambda[0]\beta_{\boldsymbol{\alpha}_1}(t) + \lambda[1]\Big(\beta_{\boldsymbol{\alpha}_1}(t-\frac{1}{2}) + \beta_{\boldsymbol{\alpha}_1}(t+\frac{1}{2})\Big) + \lambda[2]\Big(\beta_{\boldsymbol{\alpha}_1}(t-1) + \beta_{\boldsymbol{\alpha}_1}(t+1)\Big).$$

By solving the corresponding system of equations (6.62) for the non-zero entries of λ, we find $\lambda[0] = 18.118$, $\lambda[1] = -10.128$, and $\lambda[2] = 1.730$. For the construction of $\phi_{\boldsymbol{\alpha}_2}$, we have that $n_0 = 3$ and $N = 2(n_0-1) = 4$. The support of $\phi_{\boldsymbol{\alpha}_2}$ is therefore equal to 4 and the interpolator is expressed as

$$\phi_{\boldsymbol{\alpha}_2}(t) = \lambda[0]\beta_{\boldsymbol{\alpha}_2}(t) + \lambda[1]\Big(\beta_{\boldsymbol{\alpha}_2}(t-\frac{1}{2}) + \beta_{\boldsymbol{\alpha}_2}(t+\frac{1}{2})\Big).$$

By solving (6.62), we find that $\lambda[0] = 7.396$ and $\lambda[1] = -2.825$.

Since the generator is an interpolator, the control points of the surface are given by its samples, specified by

$$\boldsymbol{\sigma}(u,v)\big|_{u=k,v=l} = \begin{pmatrix} \frac{1}{2}r^2\cos(\frac{2\pi k}{M_1})\sin(\frac{4\pi l}{M_2}) \\ \frac{1}{2}r^2\sin(\frac{2\pi k}{M_1})\sin(\frac{4\pi l}{M_2}) \\ r^2\cos(\frac{2\pi k}{M_1})\sin(\frac{2\pi k}{M_1})\cos^2(\frac{2\pi l}{M_2}) \end{pmatrix}.$$

We choose $(u,v) \in [0,1]^2$ and $r = 3$. Then, the sums in (7.2) are finite due to the compact support of the generators. The parameterization of the surface is given by

$$\boldsymbol{\sigma}(u,v) = \sum_{k=-2}^{M_1+2}\sum_{l=-1}^{M_2+1} \boldsymbol{\sigma}[k,l]\phi_{\boldsymbol{\alpha}_1}(M_1 u - k)\phi_{\boldsymbol{\alpha}_2}(M_2 v - l).$$

The Roman surface is illustrated in Figure 6.13.

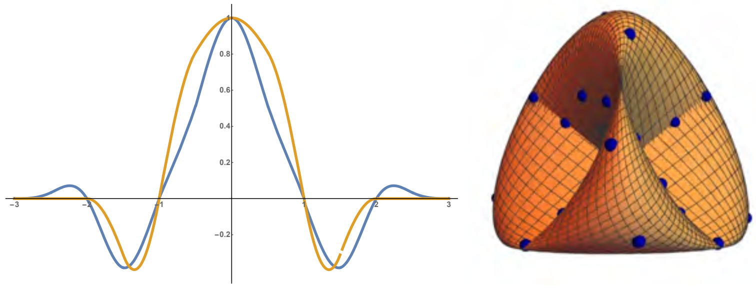


Figure 6.13: Roman surface. The interpolators $\phi_{\boldsymbol{\alpha}_1}$ (blue) and $\phi_{\boldsymbol{\alpha}_2}$ (yellow) are shown as well as the reconstructed surface (right). The interpolatory control points are shown as blue dots on the surface.

Reproduction of a Hyperbolic Paraboloid. A parameterization of a hyperbolic paraboloid is given by

$$\boldsymbol{\sigma}(u,v) = \begin{pmatrix} au\cosh(v) \\ bu\sinh(v) \\ hu^2 \end{pmatrix}, \quad (u,v) \in \mathbb{R}^2, \tag{6.76}$$

where a, b, and h are constants. The paraboloid (6.76) is polynomial in u and hyperbolic in v. Hence, we choose $\boldsymbol{\alpha}_1 = (0,0,0)$ and $\boldsymbol{\alpha}_2 = \left(0, \frac{1}{M_2}, \frac{-1}{M_2}\right)$ when expressing (6.76) as the tensor-product surface

$$\boldsymbol{\sigma}(u,v) = \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} \boldsymbol{\sigma}[k,l]\phi_{\boldsymbol{\alpha}_1}(M_1u-k)\phi_{\boldsymbol{\alpha}_2}(M_2v-l).$$

To construct $\phi_{\boldsymbol{\alpha}_1}$, we have that $n_0 = 3$ and its support is equal to $N = 2(n_0-1) = 4$. The interpolator is expressed as

$$\phi_{\boldsymbol{\alpha}_1}(t) = \lambda[0]\beta_{\boldsymbol{\alpha}_1}(t) + \lambda[1]\Big(\beta_{\boldsymbol{\alpha}_1}(t-\frac{1}{2}) + \beta_{\boldsymbol{\alpha}_1}(t+\frac{1}{2})\Big).$$

Solving (6.62), we obtain $\lambda[0] = 2$ and $\lambda[1] = -\frac{1}{2}$.
For the construction of $\phi_{\boldsymbol{\alpha}_2}$, we see that $n_0 = 3$, $N = 2(n_0-1) = 4$, and its support is also of size 4. The interpolator is given by

$$\phi_{\boldsymbol{\alpha}_2}(t) = \lambda[0]\beta_{\boldsymbol{\alpha}_2}(t) + \lambda[1]\Big(\beta_{\boldsymbol{\alpha}_2}(t-\frac{1}{2}) + \beta_{\boldsymbol{\alpha}_2}(t+\frac{1}{2})\Big).$$

Solving (6.62) yields $\lambda[0] = 1.968$ and $\lambda[1] = -0.489$. As in the previous example, the control points are obtained by sampling the surface, which leads to

$$\boldsymbol{\sigma}(u,v)\big|_{u=k,v=l} = \begin{pmatrix} a\frac{k}{M_1}\cosh(\frac{l}{M_2}) \\ b\frac{k}{M_1}\sinh(\frac{l}{M_2}) \\ h(\frac{k}{M_1})^2 \end{pmatrix}.$$

We choose $(u,v) \in [-1,1]^2$, $M_1 = M_2 = 3$, $a = b = 4$ and $h = 8$. The corresponding parameterization is

$$\boldsymbol{\sigma}(u,v) = \sum_{k=-M_1-1}^{M_1+1} \sum_{l=-M_2-1}^{M_2+1} \boldsymbol{\sigma}[k,l]\phi_{\boldsymbol{\alpha}_1}(M_1u-k)\phi_{\boldsymbol{\alpha}_2}(M_2v-l).$$

The hyperbolic paraboloid is illustrated in Figure 6.14.

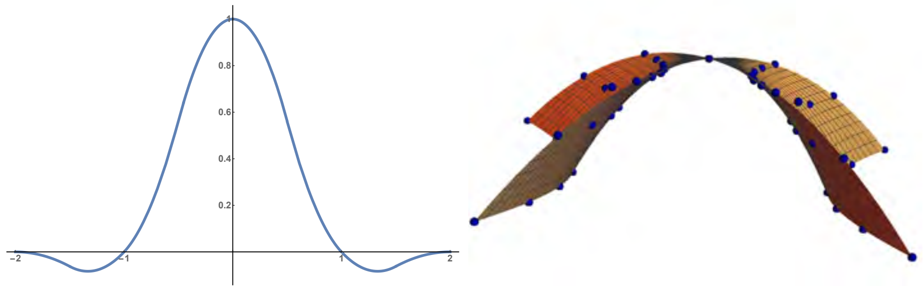


Figure 6.14: Hyperbolic paraboloid. On the left the interpolator $\phi_{\boldsymbol{\alpha}_2}$ is shown. ($\phi_{\boldsymbol{\alpha}_1}$ is shown in Figure 6.11.) On the right the reconstructed hyperbolic paraboloid with its interpolatory control points (blue dots) is shown.

Interactive Shape Modeling. The presented interpolators are well suited to be implemented in an interactive shape modeling framework, for instance, for CAD design. The key properties in such a context are

- *interpolation property:* it allows to easily interact with the surface by displacing control points with a computer mouse;
- *varying resolution:* once the "rough" outline of the shape is designed, the details are modeled by increasing the resolution at specific locations.

Example: Character Design. The interpolation property is convenient to design complex shapes as shown in the first column in Figure 6.8 in order to obtain a low resolution model. To increase the level of detail of the shape, we increase the resolution of the surface by first applying the pre-refinement step (6.58) and then the standard refinement mask for (exponential) B-splines (6.54). These two steps increase the number of control points, however, at the expense of being interpolatory. This increase in the number of control points allows one to have more flexibility in the modeling process. Furthermore, after few iterations the convergence of the proposed modified refinement scheme allows for an interpolatory-like behavior (see Figure 6.8).

6.3.4 Summary

We have presented a general framework to construct interpolators as linear combinations of exponential B-splines of the same order n_0. The interpolators are compactly supported and their integer shifts form a Riesz basis whenever the corresponding B-spline does. Since the underlying building blocks are exponential B-splines, we can exploit the refinability property of the B-splines to resample the model. Based on these general properties, we have constructed a new family of interpolators to represent parametric shapes. The new interpolators are smooth and they can be designed to perfectly reproduce polynomial, trigonometric, and hyperbolic shapes. We provide explicit examples of such generators and show in detail how idealized parametric curves and surfaces are constructed. The reconstructed shapes have the property that the control points directly lie on their boundary. This enables an intuitive manipulation of shapes by changing the location of a control point. Since the interpolators have compact support, this displacement of control points allows one to *locally* control the deformation of a shape[6]. In a next step, we plan to further investigate the refinability properties for practical applications such as real-time rendering or zooming of images.

6.4 Appendix

6.4.1 Proof of Proposition 15

Proof. We split the proof into two parts: the existence of an upper bound, relying on the one for the corresponding exponential B-spline, and the lower bound, based on the fact that the function is interpolatory.

Upper Bound We first show that one can find $B_{\boldsymbol{\alpha}} < \infty$ such that, for every $\omega \in \mathbb{R}$,

$$\sum_{k\in\mathbb{Z}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 \leq B_{\boldsymbol{\alpha}}^2. \tag{6.77}$$

[6]Demo videos illustrating an implementation of our framework are found at http://bigwww.epfl.ch/demo/varying-resolution-interpolator/.

This result is well-known (see for instance [63, Theorem 1]); we prove it for the sake of completeness. A more precise estimation of $B_{\boldsymbol{\alpha}}$ is given in [63, Proposition 3]. The function $\beta_{\boldsymbol{\alpha}} * \beta_{\boldsymbol{\alpha}}^{\vee}$, where $\beta_{\boldsymbol{\alpha}}^{\vee}(t) = \beta_{\boldsymbol{\alpha}}(-t)$, is continuous and compactly supported. Therefore, the sequence $c = (c[n])_{n\in\mathbb{Z}} = (\beta_{\boldsymbol{\alpha}} * \beta_{\boldsymbol{\alpha}}^{\vee}(n))_{n\in\mathbb{Z}}$ of its samples is in $\ell_1(\mathbb{Z})$. Since the Fourier transform of $\beta_{\boldsymbol{\alpha}} * \beta_{\boldsymbol{\alpha}}^{\vee}(t)$ is $|\widehat{\beta}_{\boldsymbol{\alpha}}(\omega)|^2$, we have that

$$\sum_{k\in\mathbb{Z}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 = \sum_{k\in\mathbb{Z}} c[k]\mathrm{e}^{-\mathrm{j}\omega k} \leq \|c\|_{\ell_1(\mathbb{Z})} := B_{\boldsymbol{\alpha}}^2 < \infty. \tag{6.78}$$

Using (6.44), we moreover have that

$$\sum_{k\in\mathbb{Z}} |\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 = \sum_{k\in\mathbb{Z}} \left(\sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}(\omega-2k\pi)n/2} \right)^2 |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2. \tag{6.79}$$

By splitting the sum with respect to k odd or even and since $\mathrm{e}^{-\mathrm{j}(\omega-2k\pi)n/2} = ((-1)^k)^n \mathrm{e}^{-\mathrm{j}\omega n/2}$, we have that

$$\sum_{k\in\mathbb{Z}} |\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 = |G_0(\omega)|^2 \sum_{k \text{ even}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 + |G_1(\omega)|^2 \sum_{k \text{ odd}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 \tag{6.80}$$

with $G_0(\omega) = \sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}\omega n/2}$ and $G_1(\omega) = \sum_{n\in\mathbb{Z}} (-1)^n \lambda[n]\mathrm{e}^{-\mathrm{j}\omega n/2}$. Clearly, for $i = 0, 1$, $|G_i(\omega)| \leq \sum_{n\in\mathbb{Z}} |\lambda[n]| = \|\lambda\|_{\ell_1(\mathbb{Z})}$ and thus,

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} |\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 &\leq \|\lambda\|_{\ell_1(\mathbb{Z})}^2 \left(\sum_{k \text{ even}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 + \sum_{k \text{ odd}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 \right) \\
&= \|\lambda\|_{\ell_1(\mathbb{Z})}^2 \sum_{k\in\mathbb{Z}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2 \\
&\leq \|\lambda\|_{\ell_1(\mathbb{Z})}^2 B_{\boldsymbol{\alpha}}^2,
\end{aligned}$$

so that the constant $B_{\lambda,\boldsymbol{\alpha}} = \|\lambda\|_{\ell_1(\mathbb{Z})} B_{\boldsymbol{\alpha}} < \infty$ acts as an upper bound in (6.48).

Lower Bound The function $\phi_{\lambda,\boldsymbol{\alpha}}$ is assumed to be interpolatory; in the frequency domain, this condition is expressed as

$$\sum_{k\in\mathbb{Z}} \widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega - 2k\pi) = 1 \quad \text{for all} \quad \omega \in \mathbb{R}. \tag{6.81}$$

Moreover, the functions $\omega \mapsto \sum_{k\in\mathbb{Z}} |\widehat{\beta}_{\boldsymbol{\alpha}}(\omega - 2k\pi)|^2$, G_0, and G_1 above are also continuous and periodic (for G_0 and G_1, this comes from $\lambda \in \ell_1(\mathbb{Z})$). Therefore, the function $\omega \mapsto \sum_{k\in\mathbb{Z}} |\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega - 2k\pi)|^2$ is also continuous and periodic. As such, it reaches its minimum at some frequency $\omega_0 \in [0, 2\pi]$. Further, the inequality $A^2_{\lambda,\boldsymbol{\alpha}} := \sum_{k\in\mathbb{Z}} |\widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega_0 - 2k\pi)|^2 \geq 0$ holds. Assume now that $A_{\lambda,\boldsymbol{\alpha}} = 0$, then we have $\widehat{\phi}_{\boldsymbol{\alpha}}(\omega_0 - 2k\pi) = 0$ for every $k \in \mathbb{Z}$, and therefore, $\sum_{k\in\mathbb{Z}} \widehat{\phi}_{\lambda,\boldsymbol{\alpha}}(\omega_0 - 2k\pi) = 0$, which contradicts (6.81). Hence, $A_{\lambda,\boldsymbol{\alpha}} > 0$ acts as a lower bound in (6.48).

□

Remark. Based on (6.80), we deduce the following estimates for the Riesz constants $A_{\lambda,\boldsymbol{\alpha}}$ and $B_{\lambda,\boldsymbol{\alpha}}$ associated to $\phi_{\lambda,\boldsymbol{\alpha}}$:

$$A_{\lambda,\boldsymbol{\alpha}} = A_{\boldsymbol{\alpha}} \min_{[0,2\pi]} |\hat{\lambda}(\mathrm{e}^{\mathrm{j}\omega})|, \tag{6.82}$$

$$B_{\lambda,\boldsymbol{\alpha}} = B_{\boldsymbol{\alpha}} \max_{[0,2\pi]} |\hat{\lambda}(\mathrm{e}^{\mathrm{j}\omega})|, \tag{6.83}$$

where $A_{\boldsymbol{\alpha}}$ and $B_{\boldsymbol{\alpha}}$ are the constants for the Riesz basis condition for $\beta_{\boldsymbol{\alpha}}$ (given in Proposition 4 and Proposition 3 in [63]), and $\hat{\lambda}(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}\omega n/2}$ is the discrete Fourier transform of λ.

6.4.2 Proof of Proposition 16

Proof. This follows from Proposition 2 in [63] which states that reproduction properties are preserved through convolution. More precisely, if f is such that $\int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\alpha t}\mathrm{d}t \neq 0$ for all $\alpha \in \boldsymbol{\alpha}$, then $f * \beta_{\boldsymbol{\alpha}}$ inherits the reproduction properties of $\beta_{\boldsymbol{\alpha}}$. In our case, we have $\phi_{\lambda,\boldsymbol{\alpha}}(t) = (f * \beta_{\boldsymbol{\alpha}})(t)$ with $f(t) = \sum_{n\in\mathbb{Z}} \lambda[n]\delta(t - n/2)$. Then, for every $\alpha \in \boldsymbol{\alpha}$,

$$\int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\alpha t}\mathrm{d}t = \sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\alpha n/2}, \tag{6.84}$$

which is bounded and nonzero by assumption. □

6.4.3 Proof of Proposition 17

Proof. For the causal generator, we use (6.42) and (6.52) to express (6.44) as

$$\widehat{\phi}^{+}_{\lambda,\boldsymbol{\alpha}}(\omega) = \mathrm{e}^{-\mathrm{j}\omega(n_0/2-1)} \left(\sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}\omega n/2} \right) \widehat{\beta}^{+}_{\boldsymbol{\alpha}}(\omega). \tag{6.85}$$

Then, we have

$$\begin{aligned} m_0\widehat{\phi}^{+}_{\lambda,\boldsymbol{\alpha}}(m_0\omega) &= \mathrm{e}^{-\mathrm{j}\omega m_0(n_0/2-1)} \left(\sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}m_0\omega n/2} \right) m_0\widehat{\beta}^{+}_{\boldsymbol{\alpha}}(m_0\omega) \\ &= \mathrm{e}^{-\mathrm{j}\omega m_0(n_0/2-1)} \left(\sum_{n\in\mathbb{Z}} \lambda[n]\mathrm{e}^{-\mathrm{j}m_0\omega n/2} \right) H_{\frac{\boldsymbol{\alpha}}{m_0},m_0}(\mathrm{e}^{\mathrm{j}\omega})\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0}}(\omega) \\ &= G_{\lambda,\frac{\boldsymbol{\alpha}}{m_0},m_0}(\mathrm{e}^{\mathrm{j}\omega})\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0}}(\omega), \end{aligned} \tag{6.86}$$

where we used the relation (6.53) expressed in the frequency domain. Finally, we take the inverse Fourier transform of (6.86) and obtain (6.56) in the time domain. □

6.4.4 Proof of Proposition 18

Proof. Equation (6.59) is equivalent to the frequency domain relation

$$\lim_{n\to\infty} \frac{1}{m_0 m^n} C_n(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^n}}) = \widehat{f}(\omega). \tag{6.87}$$

The iterative step between c_n and c_{n-1} in the frequency domain becomes

$$C_n(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^n}}) = H_{\frac{\boldsymbol{\alpha}}{m_0 m^n},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^n}}) C_{n-1}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^{n-1}}}). \tag{6.88}$$

Iterating this relation, we obtain

$$C_n(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^n}}) = \left(\prod_{k=1}^{n} H_{\frac{\boldsymbol{\alpha}}{m_0 m^k},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^k}}) \right) C_0(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0}}). \tag{6.89}$$

By expressing (6.53) iteratively in the frequency domain and replacing $\boldsymbol{\alpha}$ by $\boldsymbol{\alpha}/m_0$, we see that

$$\begin{aligned}\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0}}(\omega) &= \frac{1}{m} H_{\frac{\boldsymbol{\alpha}}{m_0 m},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m}})\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0 m}}\left(\frac{\omega}{m}\right) \\ &= \left(\prod_{k=1}^{n} \frac{1}{m} H_{\frac{\boldsymbol{\alpha}}{m_0 m^k},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m^k}})\right)\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0 m^n}}\left(\frac{\omega}{m^n}\right) \\ &= \lim_{n\to\infty} \prod_{k=1}^{n} \frac{1}{m} H_{\frac{\boldsymbol{\alpha}}{m_0 m^k},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m^k}}),\end{aligned} \tag{6.90}$$

where in the last line we have used the well-known convergence result from spline theory [35, 31, 154]

$$\lim_{n\to\infty} \widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0 m^n}}\left(\frac{\omega}{m^n}\right) = \widehat{\beta}_{\underbrace{(0,\dots,0)}_{n_0 \text{times}}}(0) = \operatorname{sinc}^{n_0}(0) = 1. \tag{6.91}$$

Expressing (6.57) in the frequency domain, we finally have

$$\begin{aligned}\widehat{f}(\omega) &= \frac{1}{m_0} C_0(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0}})\widehat{\beta}^{+}_{\frac{\boldsymbol{\alpha}}{m_0}}\left(\frac{\omega}{m_0}\right) \\ &= \lim_{n\to\infty} \frac{1}{m_0 m^n}\left(\prod_{k=1}^{n} H_{\frac{\boldsymbol{\alpha}}{m_0 m^k},m}(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^k}})\right) C_0(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0}}) \\ &= \lim_{n\to\infty} \frac{1}{m_0 m^n} C_n(\mathrm{e}^{\frac{\mathrm{j}\omega}{m_0 m^n}}),\end{aligned} \tag{6.92}$$

where we have used (6.91) and (6.89) for the second and third equality, respectively. □

Chapter 7

Deformable Spline Shapes in Practice

Overview

We present the construction, use, and implementation of spline shapes in practice using the theory developed in Chapter 6.

- In Section 7.1, we present applications that involve deformable spheres [1].
- In Section 7.2, we specifically show how the spherical model presented in Section 7.1 is applied for the automatic and atlas-free segmentation of the brain in 3D MRI [2]
- In Section 7.3 we construct a deformable model with cylindrical topology and use it to segment the aorta and spinal cord in 3D MRI [3].

[1]Our related publications are [27, 28].
[2]Our related publication is [79].
[3]Our related publication is [80].

7.1 Shapes with Spherical Topology: Modeling, Deformation, Interaction

Existing shape models with spherical topology are typically designed either in the *discrete* domain using *interpolating* polygon meshes or in the continuous domain using *smooth* but *non-interpolating* schemes such as subdivision or NURBS. Both polygon models and subdivision methods require a large number of parameters to model smooth surfaces. NURBS need fewer parameters but have a complicated rational expression and non-uniform shifts in their formulation. We present a new method to construct deformable closed surfaces, which includes the exact sphere, by combining the best of two worlds: a *smooth* and *interpolating* model with a continuously varying tangent plane and well-defined curvature at every point on the surface. Our formulation is considerably simpler than NURBS and requires fewer parameters than polygon meshes. We demonstrate the generality of our method with applications including intuitive user-interactive shape modeling, continuous surface deformation, shape morphing, reconstruction of shapes from parameterized point clouds, and fast iterative shape optimization for image segmentation. Comparisons with discrete methods and with non-interpolating approaches highlight the advantages of our framework.

7.1.1 Introduction

The representation of shapes with spherical topology has been an ongoing research topic in computer graphics for more than three decades. The principal reason is the massive demand for closed genus-zero surfaces in industrial, architectural, and animation design as well as in biomedical imaging. Designing spherical models that are simultaneously optimal with respect to several different shape characteristics still remains a challenge. Depending on whether an application involves user interaction, shape deformation or optimization schemes, different aspects of a model are more important than others.

In user-interactive applications, a fundamental requirement is the ability to intuitively manipulate the shape. Typically, this requirement presupposes an easy way to interact directly with the surface as well as to control shapes locally. It is linked to the topic of surface deformation because the deformation should be stable: a small perturbation of the surface should result in a small change of the

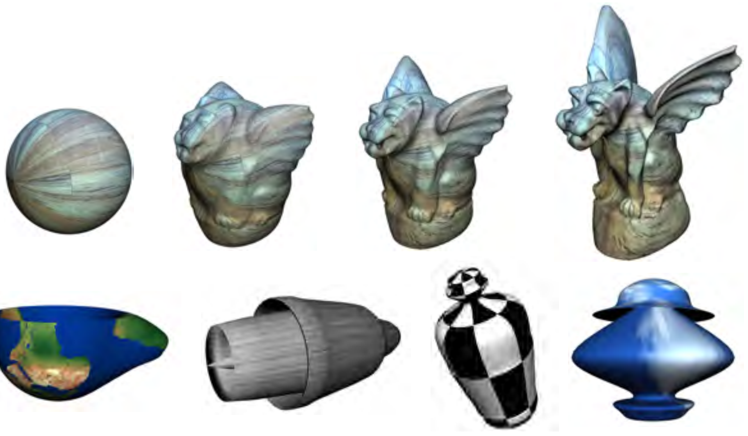


Figure 7.1: Smooth modeling of shapes with spherical toplogy. A continuous deformation of the sphere into the Gargoyle is shown in the top row, where a wood texture has been added to the surface. The shapes in the bottom row consist of a single surface patch and are constructed through interactive deformation of the sphere. The interpolating structure of the model allows us to intuitively design surfaces that can adopt shapes beyond the classical spherical topology. Our framework is inherently smooth, which facilitates natural texturing.

shape. Numerical stability is crucial too: a theoretical model must remain useful in practice. On the other hand, the application might involve shape deformation as an optimization process. For example, in real-time shape recognition, approximation, and segmentation, the fast evaluation of derivative- and integral-based quantities in iterative settings is required. Further, the smoothness of the surface and the number of parameters that are involved can also play an important role. Usually, it is impossible to find a model that is optimal with respect to all of these requirements. In practice, a compromise is made favoring the most important needs for a specific application. Existing models are based on polygon meshes, NURBS or subdivision.

Overview and Contribution. In this article, we present the full theory of a model to construct deformable shapes with spherical topology, along with applications. This work was first introduced in a condensed form as a "technical brief" at SIGGRAPH ASIA 2016 [27]. Our framework is based on interpolatory control points, similar to polygon meshes, while at the same time being smooth and formulated in the continuous domain as is the case with NURBS. The resulting surface allows for local control, is everywhere differentiable, and has a continuously varying tangent plane at every point on the surface as well as a well-defined Gaussian curvature. A major contribution is the explicit formulation of necessary conditions for the poles of the sphere to remain closed and smooth when deforming. We illustrate the use of our method with several applications: 1) User-interactive shape modeling: because the basis functions satisfy the interpolation property, the control points lie directly on the surface of the object, which facilitates intuitive shape modeling. The basis is also finitely supported to enable local surface control and allows us to model a broad range of shapes by deforming a single spherical surface patch; 2) Smooth surface reconstruction from parameterized point clouds: if the underlying spherical parameterization of the samples is known, they can be easily interpolated with our model to reconstruct a smooth surface; 3) Efficient surface deformation: by exploiting the affine invariance of our model, we illustrate how a fast implementation of deformation algorithms is achieved in the case of minimum-energy deformation; 4) Fast iterative optimization of deformation algorithms: we show how the iterative evaluation of surface and volume integrals is implemented in an efficient way for real-time optimization and provide an example of a segmentation algorithm for 3D medical images.
Our construction involves a class of smooth basis functions that are non-rational and have uniform shifts, which leads to a considerably simpler formulation than for traditional parametric methods. A control-point-based structure allows us to use fewer parameters than polygon or subdivision methods to achieve smoothness. Examples of the use of our method are shown in Figure 7.1.

7.1.2 Related Work

Continuous Closed Surfaces. The most widely used technique to construct deformable spheres in the continuous domain is NURBS [17, 25] which are a subfamiliy of T-splines [56]. Parametric NURBS surfaces are based on polynomial B-splines and are defined by a set of control points which allow local shape control [53, 51, 52, 25, 135]. The main reason for using the rational NURBS expression instead of the (non-rational) polynomial B-splines is that NURBS are able to exactly reproduce conic sections [19]. Conceptually, this property is equivalent to reproducing trigonometric functions, which is a necessary requirement for constructing spheres. There exist several ways to construct NURBS spheres, *e.g.*, by constructing quarter or half circles and exploiting the properties of tensor-product splines or by constructing surfaces of revolution [151]. NURBS typically involve the explicit characterization of non-uniform knot vectors with double knots. A drawback of NURBS is the rational form, which leads to complicate expressions of related integrals and derivatives [135]. Furthermore, the NURBS formulation depends on additional weight parameters, which have no intuitive interpretation. Other constructions to approximate sphere-like surfaces based on B-splines have been studied in [155, 156], whereas in [86] an exact approach using exponential splines is proposed. Other models use (rational) Bézier surfaces [24], which are also related to splines [54].

Discrete Closed Surfaces. Popular discrete methods are based on polygon meshes [40, 42]. With these models it is possible to represent shapes of arbitrary topology and hence, closed surfaces with spherical topology can be easily generated. There exists a vast amount of literature on mesh optimization, processing and discretizing continuous-domain operators (*e.g.*, see [62] and [41]). Polygon models are usually interpolating, where the control points coincide with the vertices of the mesh; this property implies that the shape is modified by points which directly lie on the boundary of the object. Related to polygon models are subdivision methods [29, 32] used to construct surfaces [157, 30, 18, 33, 34]. These methods are characterized by refinement operations that are iteratively applied to a set of points leading to a continuous limit surface with a certain regularity. Hence, subdivision can be seen as a hybrid method combining the discrete and the continuous-domain approach. Although in theory they are continuous, in practice, a finite number of iterations are applied, leading to a discrete mesh (which is the reason why we list subdivision as a discrete method). As opposed to polygon mesh models, subdivision methods do not necessarily have interpolating control points. Different methods that are based on non-stationary refinement rules have been proposed to approximate a sphere using subdivision [31, 38, 133]. One drawback of polygon and subdivision methods is that they require a large number of parameters which can be a challenge when computational speed is required (*e.g.*, in finite element models [158]).

Spherical Parameterization. The problem of finding a parameterization of an object with spherical topology is not trivial and has been tackled in [159, 160]. It is linked to the problem of ordering an unorganized set of points or a point cloud, which in 3D is significantly harder than in 2D. In [134] a method is presented to generate a spherical parameterization of closed surfaces in the continuous domain by expressing them in a basis of spherical harmonics. A related problem is surface reconstruction from a point cloud [161, 162].

Interpolation. A widely used interpolating spline in computer graphics is the Catmull-Rom spline [16]. However, its nature is polynomial and hence, it cannot be used to exactly parameterize the sphere. A variant of the Catmull-Rom spline used in signal processing is the Keys cubic convolution interpolator [127] which has been generalized by [76, 77] to construct a trigonometric interpolation kernel that is able to reproduce conic sections. Other variants have been presented in [163, 55, 80]. An interpolating subdivision scheme was originally introduced by Deslaurier and Dubuc [58]. Variants of this scheme have been proposed in [164] which have also been used to construct conic sections [165].

7.1.3 Parametric Shape Representation

Notation. We use boldfont for vectors and regular font for scalars, *e.g.*, $\boldsymbol{c} = (c_x, c_y, c_z)$. To denote partial derivatives, we use the notation $\frac{\partial \boldsymbol{\sigma}(u,v)}{\partial u} = \boldsymbol{\sigma}_u(u,v)$.

Note that, throughout this article we will use the terms "spherical topology" and "closed surface" to describe the same kind of objects, namely connected surfaces without holes or boundaries. Using these terms to describe equivalent objects makes sense in computer graphics because in a digital environment, even continuous-domain objects can only be represented by a discretized approximation. However, in the field of mathematical topology a more rigorous definition of these terms would be required.

Tensor-product Surfaces. We construct parametric shapes using integer shifts of a (non-rational) generator function φ. A 3D curve $\boldsymbol{r}(t)$ that is described by the coordinate functions $x(t)$, $y(t)$ and $z(t)$ with $t \in \mathbb{R}$ is then represented by a linear combination of integer shifts of φ as

$$\boldsymbol{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \sum_{k\in\mathbb{Z}} \boldsymbol{c}[k]\varphi(t-k), \tag{7.1}$$

where the $\{\boldsymbol{c}[k] = (c_x[k], c_y[k], c_z[k])\}_{k\in\mathbb{Z}}$ are the 3D *control points*. The model (7.1) is extended in order to construct a separable parametric *tensor product* surface $\boldsymbol{\sigma}(u,v)$ with $u, v \in \mathbb{R}$, that is represented as the component-wise product (described by the symbol "$\times$") of two curves $\boldsymbol{r}_1 \times \boldsymbol{r}_2$, *i.e.*,

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} &= \begin{pmatrix} x_1(u)\cdot x_2(v) \\ y_1(u)\cdot y_2(v) \\ z_1(u)\cdot z_2(v) \end{pmatrix} \\ &= \underbrace{\sum_{k\in\mathbb{Z}} \boldsymbol{c}_1[k]\varphi_1(u-k)}_{\boldsymbol{r}_1} \times \underbrace{\sum_{l\in\mathbb{Z}} \boldsymbol{c}_2[l]\varphi_2(v-l)}_{\boldsymbol{r}_2} \\ &= \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} \underbrace{\boldsymbol{c}_1[k]\times\boldsymbol{c}_2[l]}_{\boldsymbol{c}[k,l]}\varphi_1(u-k)\varphi_2(v-l).\end{aligned} \tag{7.2}$$

Based on this equation, an arbitrary non-separable surface with control points $\boldsymbol{c}[k,l]$ can be constructed whose expression corresponds to the last line of (7.2).

Generator function φ. The shapes that the model (7.2) can adopt depend on the generator φ. For example, if φ is a B-spline the resulting shapes are polynomial. In our case, we are interested in generating trigonometric shapes in order to be able to construct exact spheres. For this purpose, we use the piecewise exponential generator proposed by [76], which reproduces sines and cosines. It is defined as $\varphi = \beta * \psi$, where β is a third order exponential B-spline, ψ is an appropriate smoothing kernel and "$*$" denotes a convolution. We provide the explicit expression of φ in (6.3). The relevant characteristics of φ for our construction, besides its sphere-reproduction property, are that it is twice differentiable with bounded second derivatives and satisfies the interpolation property $\varphi(t=k) = \delta_k$, where δ_k denotes the Kronecker delta, $t \in \mathbb{R}$ and $k \in \mathbb{Z}$. Our generator constitutes a partition of unity, *i.e.*, $\sum_k \varphi(t-k) = 1$, which is a necessary and sufficient condition for the represented shapes to be *affine invariant*. Because this generator depends on the number M of control points that are used to construct a curve $\boldsymbol{r}$, we use the notation φ_M instead of φ. The support of φ_M is equal to 4.

Definition 10. As a simplification to indicate the M_1-periodized basis function, we write

$$\phi_1(t) := \varphi_{M_1,\mathrm{per}}(t) = \sum_{n=-\infty}^{+\infty} \varphi_{M_1}(t - M_1 n) \tag{7.3}$$

and $\phi_2 := \varphi_{2M_2}$. To denote the integer shifts of the basis functions on the normalized parameter domain we use $\phi_{1,k}(t) := \phi_1(M_1 t - k)$ and $\phi_{2,k}(t) := \phi_2(M_2 t - k)$.

7.1.4 Spherical Parameterization

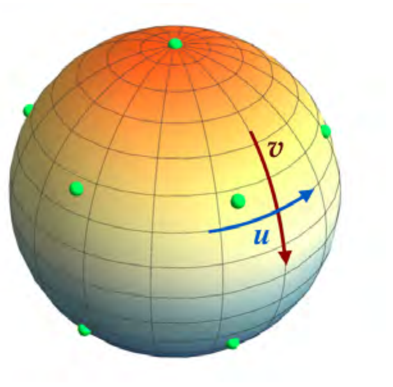


Figure 7.2: Reconstructed sphere with the interpolatory control points shown in light green. The two directions of the parameterizations are indicated by the blue and red arrow.

In this section, we outline our proposed construction of the deformable sphere.

Without loss of generality, we parameterize its surface as

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} &= \begin{pmatrix} \cos(2\pi u)\sin(\pi v) \\ \sin(2\pi u)\sin(\pi v) \\ \cos(\pi v) \end{pmatrix} \\ &= \boldsymbol{r}_1(u) \times \boldsymbol{r}_2(v)\end{aligned} \tag{7.4}$$

with $u, v \in [0,1]$. In [76], it has been shown that

$$\boldsymbol{r}_1(\frac{u}{M_1}) = \begin{pmatrix} \cos(\frac{2\pi u}{M_1}) \\ \sin(\frac{2\pi u}{M_1}) \\ 1 \end{pmatrix} = \sum_{k=0}^{M_1-1} \begin{pmatrix} \cos\left(\frac{2\pi k}{M_1}\right) \\ \sin\left(\frac{2\pi k}{M_1}\right) \\ 1 \end{pmatrix} \phi_{1,k}(\frac{u}{M_1}). \tag{7.5}$$

The periodization of ϕ_1 as defined in (7.3) allows us to express the u-dependent 1-periodic trigonometric functions in (7.4) using a finite sum and $M_1 \in \mathbb{Z}$ control points. The v-dependent trigonometric functions in (7.4) are not periodic and are expressed as

$$\boldsymbol{r}_2(\frac{v}{M_2}) = \begin{pmatrix} \sin(\frac{\pi v}{M_2}) \\ \sin(\frac{\pi v}{M_2}) \\ \cos(\frac{\pi v}{M_2}) \end{pmatrix} = \sum_{l=-1}^{M_2+1} \begin{pmatrix} \sin\left(\frac{\pi k}{M_2}\right) \\ \sin\left(\frac{\pi k}{M_2}\right) \\ \cos\left(\frac{\pi k}{M_2}\right) \end{pmatrix} \phi_{2,l}(\frac{v}{M_2}). \tag{7.6}$$

Because the support of φ_M is equal to 4, for $v \in [0,1]$, we have $\varphi_{M_2}(v-l) = 0$ if $l \notin [-1, \dots, M_2+1]$, which explains the limits of the sum in (7.6). Following the construction given by (7.2), we finally parameterize the sphere as

$$\begin{aligned}\boldsymbol{\sigma}(u,v) &= \boldsymbol{r}_1(u) \times \boldsymbol{r}_2(v) \\ &= \sum_{k=0}^{M_1-1} \sum_{l=-1}^{M_2+1} \boldsymbol{c}[k,l]\phi_{1,k}(u)\phi_{2,l}(v),\end{aligned} \tag{7.7}$$

where the control points of the surface are given by its samples

$$\boldsymbol{c}[k,l] = \begin{pmatrix} c_x[k,l] \\ c_y[k,l] \\ c_z[k,l] \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{2\pi k}{M_1}\right)\sin\left(\frac{\pi l}{M_2}\right) \\ \sin\left(\frac{2\pi k}{M_1}\right)\sin\left(\frac{\pi l}{M_2}\right) \\ \cos\left(\frac{\pi l}{M_2}\right) \end{pmatrix}. \tag{7.8}$$

Note that, M_1 and M_2 are the number of control points used in the u- and v-directions. Hence, this representation allows us to construct a perfect sphere with any number of control points M_1, M_2. The only condition for the integer shifts of φ_M to form a stable basis, *i.e.*, to guarantee a stable implementation, is $M \geq 3$ [76]. The reconstructed sphere with interpolatory control points is shown in Figure 7.2.

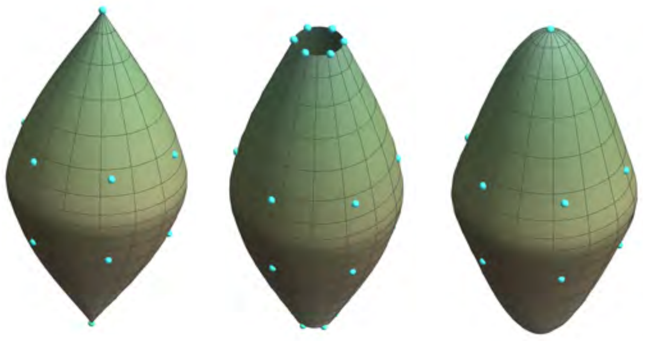


Figure 7.3: Closed and smooth deformable sphere. If no smoothness conditions are imposed the surface is non-differentiable at the poles (left). If no pole-interpolation conditions are imposed the surface looses its spherical topology (middle) when deforming. On the right, the closed and deformed sphere is shown with smoothly varying tangent planes at the poles.

Smoothness Conditions at the Poles. Since $\varphi \in \mathcal{C}^1$, continuity is guaranteed nearly everywhere on the surface as long as the control points do not overlap. However, for the deformed sphere, smoothness is not guaranteed at the poles of the sphere unless we take appropriate measures. In [155], it is shown that continuity at the poles is ensured if the deformable sphere is constructed with continuously varying tangent planes at these points. This condition is expressed mathematically as

$$\boldsymbol{\sigma}_v(u, v)|_{v=0} = \boldsymbol{T}_{1,N} \cos(2\pi u) + \boldsymbol{T}_{2,N} \sin(2\pi u), \tag{7.9}$$

for the north pole and

$$\boldsymbol{\sigma}_v(u,v)|_{v=1} = \boldsymbol{T}_{1,S}\cos(2\pi u) + \boldsymbol{T}_{2,S}\sin(2\pi u), \tag{7.10}$$

for the south pole, where $\boldsymbol{T}_{1,N}$, $\boldsymbol{T}_{2,N}$, $\boldsymbol{T}_{1,S}$, and $\boldsymbol{T}_{2,S}$ are vector parameters that can be freely chosen. In Appendix 7.4.1, we show that both sides of the equality in (7.9) can be simplified independently and we end up with the condition

$$\boldsymbol{c}[k,-1] = \frac{\boldsymbol{T}_{1,N}\cos\left(\frac{2\pi k}{M_1}\right) + \boldsymbol{T}_{2,N}\sin\left(\frac{2\pi k}{M_1}\right)}{M_2\varphi'_{2M_2}(1)} + \boldsymbol{c}[k,1]. \tag{7.11}$$

Similarly, (7.10) simplifies to

$$\boldsymbol{c}[k,M_2+1] = \boldsymbol{c}[k,M_2-1] - \frac{\boldsymbol{T}_{1,S}\cos\left(\frac{2\pi k}{M_1}\right) + \boldsymbol{T}_{2,S}\sin\left(\frac{2\pi k}{M_1}\right)}{M_2\varphi'_{2M_2}(1)}. \tag{7.12}$$

The tangent plane at the poles is then spanned by the vectors $\boldsymbol{T}_{1,N}$, $\boldsymbol{T}_{2,N}$ and $\boldsymbol{T}_{1,S}$, $\boldsymbol{T}_{2,S}$. In Figure 7.3, the effect of imposing the smoothness conditions at the poles is illustrated.

Interpolation Conditions at the Poles. The sphere needs to remain closed when deforming in order to maintain spherical topology. Again, special attention needs to be paid to the poles; we require that all the circles of longitude of the original sphere originate and end at the poles of the surface. In accordance with the parameterization (7.4), this condition is expressed as

$$\begin{aligned} \boldsymbol{\sigma}(u,0) &= \boldsymbol{c}_N \quad \text{(north pole)},\\ \boldsymbol{\sigma}(u,1) &= \boldsymbol{c}_S \quad \text{(south pole)}. \end{aligned}$$

In Appendix 7.4.2, we show that condition (7.13) translates directly into

$$\begin{aligned} \boldsymbol{c}[k,0] &= \boldsymbol{c}_N \quad \text{(north pole)},\\ \boldsymbol{c}[k,M_2] &= \boldsymbol{c}_S \quad \text{(south pole)} \end{aligned} \tag{7.13}$$

$\forall k \in [0\dots M_1-1]$. In Figure 7.3, we compare a deformed sphere with and without imposing the closeness conditions at the poles.

Main Result. We combine all of the above considerations together in order to state the main result of this article: A locally and smoothly deformable sphere is expressed by the parameterization (7.7) subject to the smoothness conditions (7.11) and (7.12) and the closeness condition (7.13).

Useful Properties of $\boldsymbol{\sigma}$ in Practice. Our deformable sphere $\boldsymbol{\sigma}$ is affine invariant. Hence, its construction is independent of location and orientation, *i.e.*,

$$\mathbf{A}\boldsymbol{\sigma}(u,v)+\boldsymbol{b}=\sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}}(\mathbf{A}\boldsymbol{c}[k,l]+\boldsymbol{b})\phi_{1,k}(u)\phi_{2,l}(v),$$

where $\mathbf{A}$ is a 3×3 matrix and $\boldsymbol{b}$ a constant vector in 3D.
Furthermore, since ϕ is twice differentiable, the surface has everywhere a well-defined tangent plane and Gaussian curvature. This property, for instance, allows us to compute the normal vector at any point on the surface; an important requirement to render a textured surface.

7.1.5 Results and Applications

Interactive Modeling

It is crucial in interactive shape modeling that the modeling process is intuitive. Standard modeling applications allow a user to modify a shape by dragging its control points with the mouse in order to displace them. If the control points lie directly on the surface of the shape, the modeling task is significantly simplified. This is the case for polygon models, but then the underlying shape is not smooth. On the other hand, NURBS allow for the construction of smooth shapes, but the control points do not interpolate the shape. This makes the modeling task less intuitive. Local shape control is difficult as the surface becomes more complex because it is no longer clear which part of the surface is affected by a specific control point. Our proposed construction solves this problem since φ_M satisfies the interpolation condition and is also smooth. Hence, even if the modeled surface is of great complexity the modeling process remains intuitive and simple since the control points always lie on the boundary of the shape. Furthermore, thanks to the compact support of φ_M, local shape control is guaranteed. In Figure 7.4, we illustrate the interactive shape modeling process.

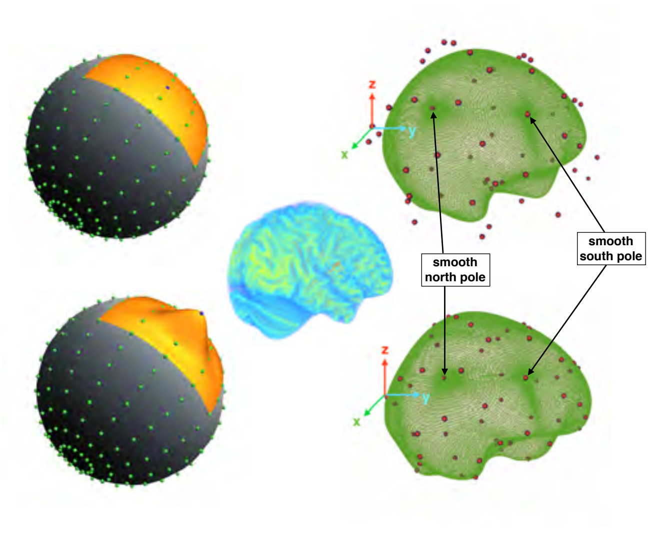


Figure 7.4: Interactive Modeling. On the left, the region (yellow) that is affected by moving a single control point (blue) is shown, which corresponds to a patch of size 4×4 due to the support of the generator φ_M. On the right, a brain (green) has been modeled using our interpolatory construction (bottom) and compared to the process where a non-interpolatory basis function is used (top) similar to NURBS. The coordinate system indicates a control point that is about to be interactively displaced in 3D space. In the top right image, it is no longer clear which region of the surface is controlled by a certain control point. The two poles are indicated in the figure to show the importance of the smoothness property in practice. In the middle, a smooth brain model has been rendered based on the modeling process illustrated on the right.

Intuitive User Interaction. Our proposed framework can be exploited to make user-interactive shape modeling more intuitive and compelling. It is ideally suited to be implemented in interactive shape modeling software, where the user modifies the shape by displacing the control points with the computer mouse. With relatively few control points, complex structures are easily constructed and modified. In Figure 7.5, we show examples of the use of our framework in an interactive modeling environment. Final renderings, where texture is added to a shape, are achieved without discretization artifacts independent of the number of control points chosen since the underlying structure is smooth, (see Figure 7.1, bottom row). In Figure 7.6, the effect of enforcing the poles to be smooth is illustrated in the case of interactive modeling.

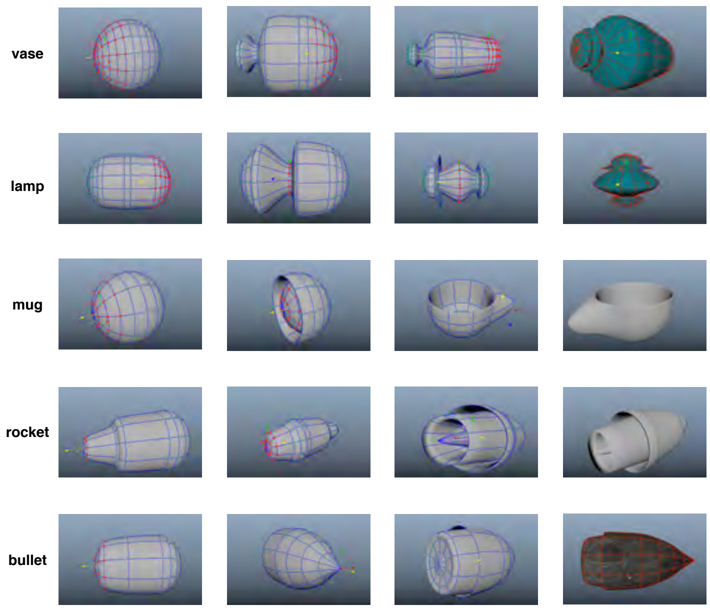


Figure 7.5: Implementation of the framework in a shape modeling environment. Different shapes are interactively designed starting from a sphere (from left to right). The interpolatory control points allow us to easily model surfaces that can adopt shapes beyond traditional spherical topology, such as the mug, rocket or bullet. The last two rows show shapes where only the closeness condition has been imposed in order to allow for the construction of sharp kinks.

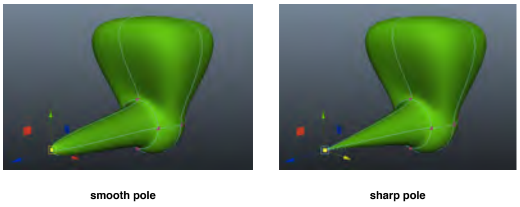


Figure 7.6: Poles with continuously varying tangent plane. The effect of imposing the smoothness condition on the poles in interactive shape modeling is illustrated. Left: smooth pole; right: sharp discontinuity at the pole resulting in a singularity.

Shape Interpolation

When dealing with a parameterized point cloud whose points correspond to the samples of a surface with spherical topology, our formulation allows for an immediate reconstruction of the smooth shape. Several algorithms have been proposed to obtain such a parameterization (*c.f.* Section 7.1.2). In this case, for each point $\boldsymbol{p} \in \mathbb{R}^3$ of the point cloud, a pair (u_k, v_l) of coordinates is assigned in the parameter domain and we can establish the relation $\boldsymbol{\sigma}(u_k, v_l) = \boldsymbol{p}_{k,l} = \boldsymbol{c}[k, l]$. For a fixed number of points, M_1, M_2, in the u- and v-direction, the parametric coordinates for the normalized parametric domain, *i.e.*, $u, v \in [0, 1]$, are given by $u_k = \frac{k}{M_1}$ and $v_l = \frac{l}{M_2}$. The resulting continuously defined surface $\boldsymbol{\sigma}(u, v)$ is immediately reconstructed since it is fully specified by its control points subject to the smoothness and pole-interpolation conditions described above. An example is shown in Figure 7.7.

Smooth Modeling at Arbitrary Resolution. Because our construction of $\boldsymbol{\sigma}$ is inherently smooth, even with few control points the tangent plane and Gaussian curvature are everywhere well-defined. This property can be an advantage to construct textured models with few parameters, for example, in applications involving real time rendering. As an example, we have parameterized the point cloud of the Gargoyle model using the algorithm described by [159], which allows us to

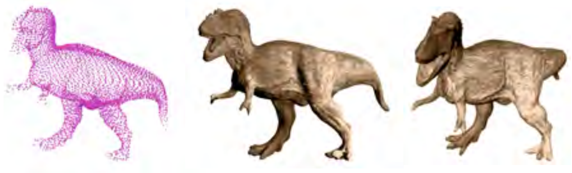


Figure 7.7: Interpolation of a parameterized point cloud. The dinosaurs (middle and right) are smooth reconstructions obtained by interpolating the point cloud on the left. Our surface construction is affine invariant and hence, rotating the shape is simply obtained by rotating the point cloud.

reconstruct a smooth surface by interpolating the points. Additionally, we have subsampled the point cloud at different resolutions to obtain an approximation of the Gargoyle with varying levels of accuracy. In Figure 7.8, we illustrate the result and show a comparison with polygons.

Compression. Related to the previous example is the problem of shape compression. Typically, the fewer coefficients are used in order to compress a shape, the more discontinuous its representation becomes, which influences texturing and rendering of smooth surfaces. The advantage of our model is that smoothness is always preserved, even with few coefficients as shown in Figure 7.8.

Efficient Shape Deformation

An advantage of using continuous-domain models that are based on control points is that the shapes are described by a finite number of control points, whereas the corresponding coordinate functions x, y, and z live in an infinite dimensional space; this allows us to describe a shape deformation process in the continuous domain just by displacing the control points. In the following, we provide two examples that illustrate how the minimization of distance criteria in the continuous domain can be efficiently formulated as conditions with respect to the control points. Other deformation criteria which can be minimized in a similar way were studied in [166, 167] and [168].

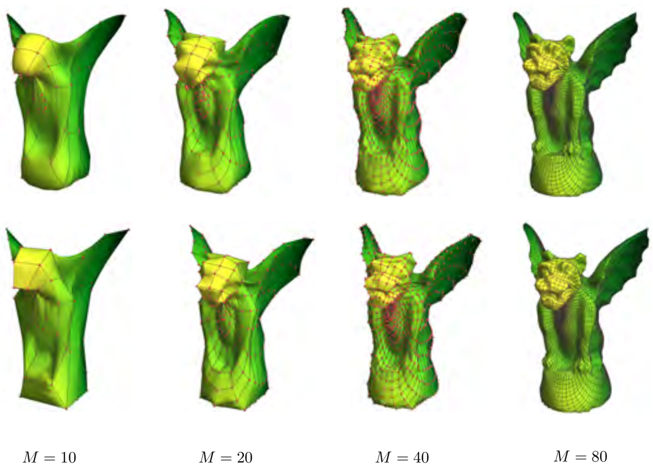


Figure 7.8: Interpolation of shapes with spherical topology: Smooth Gargoyle reconstructions at different resolutions. The same number of control points is used in both directions of the parameter domain, *i.e.*, $M = M_1 = M_2$. In the top row, the results obtained with our construction are shown, whereas in the bottom row a (linear) polygon reconstruction method is applied. Note that with our approach the smoothness of the model does not depend on the number of parameters.

Minimum-Energy Deformation

We illustrate two deformation processes which correspond to minimum-energy deformation in $L_2([0,1]^2, \mathbb{R}^3)$. Both processes are formulated entirely with respect to the control points. Thereby, we can parameterize the path, which describes the deformation in the space that contains all parametric shapes. An immediate application is the construction of interpolated or extrapolated shapes, where the terms "interpolated" and "extrapolated" refer to a shape lying on the path in some *shape space*. Typically, such a shape space is described by a metric that provides a notion of "distance" between two points that lie in the space. Hence, in a given shape space, a shape is treated as a single point. Here we are interested in describing the deformation such that a minimum amount of *energy* is required in order to deform one shape into another. This translates into describing the deformation as the shortest path between two points in the shape space according to its underlying metric.

The Hilbert Plane as a Shape Space. Given two surfaces $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ living in the Hilbert plane $L_2([0,1]^2, \mathbb{R}^3)$, the shortest path connecting them can be parameterized by the "intermediate" surface $\boldsymbol{\sigma}$, that minimizes

$$\boldsymbol{F}(\tau, \boldsymbol{\sigma}) = \tau \|\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}\|_{L_2}^2 + (1-\tau)\|\boldsymbol{\sigma}_2 - \boldsymbol{\sigma}\|_{L_2}^2 \tag{7.14}$$

for a given $\tau \in [0,1]$. We see immediately that, for $\tau = 0$, the minimizer is $\boldsymbol{\sigma} = \boldsymbol{\sigma}_2$, whereas for $\tau = 1$ it is $\boldsymbol{\sigma} = \boldsymbol{\sigma}_1$. For values of $\tau \in \mathbb{R} \setminus [0,1]$, the path $\boldsymbol{F}$ describes extrapolated shapes, *i.e.*, shapes that do not lie between the two surfaces $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$. The L_2-norm in (7.14) is induced by the L_2-inner product

$$\langle \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \rangle = \int_{\mathbb{R}} \int_{\mathbb{R}} \boldsymbol{\sigma}_1(u,v)^{\mathsf{T}} \boldsymbol{\sigma}_2(u,v) \mathrm{d}u \mathrm{d}v. \tag{7.15}$$

Using the property that our parameterization is affine invariant, it is easy to show that the solution of $\min\limits_{\boldsymbol{\sigma}} \boldsymbol{F}(\tau, \boldsymbol{\sigma})$ is given by

$$\mathbf{C}(\tau) = \tau \mathbf{C}_1 + (1-\tau)\mathbf{C}_2, \tag{7.16}$$

where $\mathbf{C}, \mathbf{C}_1, \mathbf{C}_2$ are the matrices which contain all the control points of the corresponding surfaces. As an example that illustrates the deformation process and also the effect of imposing the closeness-condition on the poles, we have deformed a disk into a sphere. In Figure 7.9, we illustrate this process and compare it to the case where no pole-interpolation conditions are imposed. In Figure 7.1, the deformation of a sphere into a Gargoyle is shown.

The Hilbert Sphere as a Shape Space. Every parametric shape can be projected onto the unit Hilbert sphere by normalizing it such that $\|\boldsymbol{\sigma}\|_{L_2} = \sqrt{\langle \boldsymbol{\sigma}, \boldsymbol{\sigma} \rangle} = 1$. The shortest distance between two points $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ on the sphere, lies on the

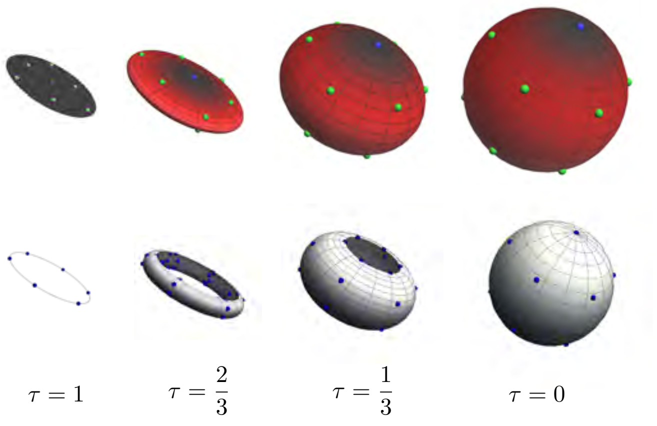


Figure 7.9: Minimum-energy deformation in the Hilbert plane. In the top row, a disk is deformed into a sphere through equation (7.16). In the bottom row, the same process is illustrated but without imposing the pole-interpolation conditions given by (7.13). Hence, the surface does not remain closed when deforming and (7.16) describes the deformation between a circle and a sphere.

great circle that passes through them. A possible parameterization of this great circle is

$$\mathbf{\Gamma}(\tau) = \frac{1}{\sin(\theta)}[\sin(\theta(1-\tau))\boldsymbol{\sigma}_1 + \sin(\theta\tau)\boldsymbol{\sigma}_2] \tag{7.17}$$

where $\theta = \cos^{-1}(\langle\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2\rangle)$ and $\mathbf{\Gamma}(0) = \boldsymbol{\sigma}_1$ and $\mathbf{\Gamma}(1) = \boldsymbol{\sigma}_2$. Again, if $\tau \in]0,1[$, equation (7.17) describes interpolated shapes, whereas for $\tau \in \mathbb{R} \setminus [0,1]$, τ describes extrapolated shapes. As in the previous example, we exploit the affine invariance of our parameterization in order to describe the deformation as a function of the control points. The interpolating control points are given by

$$\mathbf{C}(\tau) = \frac{1}{\sin(\theta)}[\sin(\theta(1-\tau))\mathbf{C}_1 + \sin(\theta\tau)\mathbf{C}_2]. \tag{7.18}$$

An example invoking this deformation is shown in Figure 7.10.

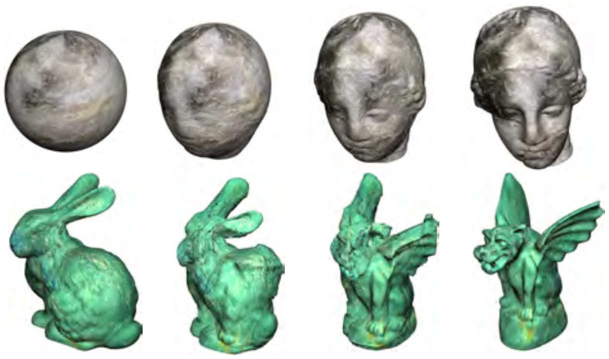


Figure 7.10: Minimum-energy deformation on the Hilbert sphere. The sphere and Venus (top row) as well as the Stanford Bunny and Gargoyle (bottom row) have been mapped through normalization onto the Hilbert sphere. A deformation is then described by equation (7.18), where the values of τ correspond to $\tau = 0, \frac{1}{3}, \frac{2}{3}, 1$ (from left to right).

Morphing. Computing morphs between two or several shapes is similar to computing the deformation between shapes. The difference is that the deformation is expressed as a parameterized weighted linear combination between two shapes, whereas a morph corresponds to a particular instance of the parameterized function. Concretely, if equation (7.16) or (7.18) is evaluated for a specific value of τ, we obtain a morph between $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$. Examples of such smooth morph constructions are shown in Figure 7.10, which correspond to morphed point clouds similar to the ones shown in Figure 7.11.

Parameterization. An important aspect to consider when using our model is that the parameterization which describes the shape is not unique. This is natural in the case of surfaces with spherical topology and originates from the fact that there exist an infinite number of ways to place the two poles (with the constraint that they must be opposite to each other). However, w.r.t. (7.16) and (7.18) we see that there is a unique correspondence between two given spherical parameterizations, which implies that given two surfaces, $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, each control point $\boldsymbol{c}_1[k_0, l_0]$ will be transformed into $\boldsymbol{c}_2[k_0, l_0]$. If a different parameterization is chosen for at least one of the two surfaces (*i.e.*, if the poles are placed differently), then the resulting deformation will inevitably be different. This process is illustrated in Figure 7.11. Insights into finding an optimal correspondence between shapes can be found in [169] and [170].

Fast Computation of Surface and Volume Integrals

In certain applications that require iterative optimization, it is necessary to rapidly compute surface or volume integrals efficiently. An example is the deformation of a surface guided by optimizing an energy functional in real time.

Flux Across Surface

We illustrate how a flux E across a surface S, parameterized by $\boldsymbol{\sigma}(u, v)$, is computed rapidly and efficiently. Given a vector field $\boldsymbol{f}$, one way of expressing the flux E is by

$$E(\boldsymbol{\sigma}) = \oiint_S \boldsymbol{f} \cdot \mathrm{d}\boldsymbol{S} = \int_0^1 \int_0^1 g^x(\boldsymbol{\sigma}) \mathrm{d}y \wedge \mathrm{d}z, \tag{7.19}$$

where $\mathrm{d}\boldsymbol{S}$ represents the vector differential of the surface area, $\wedge$ denotes the wedge product, and $g^x(x, y, z) = \int_{-\infty}^{x} \mathrm{div}\boldsymbol{f}(\tau, y, z)\mathrm{d}\tau$ is the pre-integrated divergence of the vector field $\boldsymbol{f}$ along the x-dimension. Typically, $\boldsymbol{f}$ does not depend on the surface and hence, g^x can be precomputed and stored in a look-up table to significantly speed up the computation. We derive (7.19) in Appendix 7.4.3. The use of pre-integrated functions is only possible because we define the surface $\boldsymbol{\sigma}$ in the continuous domain. Next, the flux E can be efficiently optimized by computing the gradient of E w.r.t. the control points using a gradient-based iterative method. An

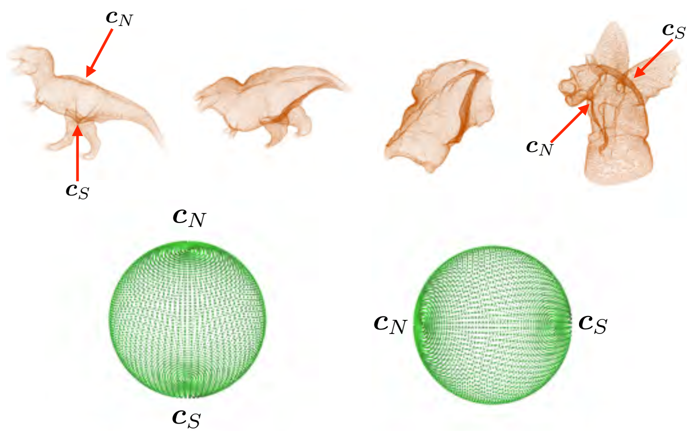


Figure 7.11: Influence of the parameterization on the deformation. The point clouds (with $M = M_1 = M_2 = 270$) that define the control points of the dinosaur and the Gargoyle are parameterized and the location of the poles is indicated with red arrows. The dinosaur (top left) is deformed into the Gargoyle (top right). The two intermediate shapes in the top row illustrate the deformation. In the bottom row, we illustrate this concept by pointing out that the poles on the sphere can be placed at different locations. For instance, if a different parameterization of the dinosaur is chosen such that the the north pole $\boldsymbol{c}_N$ and south pole $\boldsymbol{c}_S$ are exchanged, then the deformation process is different.

explicit expression of the gradient can be obtained easily and hence, implemented in an exact way.

Example. We illustrate the above computation by segmenting the surface of a human brain in a 3D MRI image. We first compute an edgemap of the 3D image using a standard surface extraction algorithm [171] and construct an energy functional E that depends on the gradient of the edge-map. Hence, in (7.19), the gradient becomes $\boldsymbol{f}$. By minimizing (7.19), $\boldsymbol{\sigma}$ deforms iteratively in order to approximate the edge map as shown in Figure 7.12. The result can easily be manually adjusted by a clinician (see Figure 7.4), which is an additional advantage of our algorithm compared to existing methods.

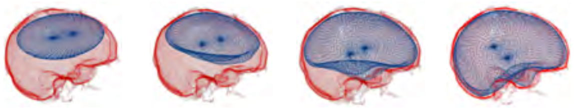


Figure 7.12: Brain segmentation in a 3D medical MRI image. The red surface is a rendered edgemap that has been extracted from medical data. An ellipsoidal surface is initialized inside the brain surface (left) and evolves by iteratively minimizing (7.19) (from left to right). The final result (right) corresponds to a smooth and continuous closed surface shape.

Exact Volume Computation

For $k_i \in [0, \dots, M_1 - 1]$ and $l_i \in [-1, \dots, M_2 + 1]$ the volume enclosed by the surface $\boldsymbol{\sigma}$ is computed by

$$\begin{aligned}\mathrm{Vol}(\boldsymbol{\sigma}) = &\sum_{\substack{k_1,k_2,k_3\\l_1,l_2,l_3}} c_x[k_1,l_1]c_y[k_2,l_2]c_z[k_3,l_3] \\ &\times \alpha(k_1,k_2,k_3,l_1,l_2,l_3)\end{aligned} \tag{7.20}$$

where c_x, c_y and c_z are the x, y, and z coordinates of the control points of $\boldsymbol{\sigma}$ and

$$\begin{aligned}&\alpha(k_1,k_2,k_3,l_1,l_2,l_3)\\ &= M_1M_2\Big(\int_0^1 \phi_{1,k_1}\phi'_{1,k_2}\phi_{1,k_3}\mathrm{d}u \int_0^1 \phi_{2,l_1}\phi_{2,l_2}\phi'_{2,l_3}\mathrm{d}v \\ &- \int_0^1 \phi_{1,k_1}\phi_{1,k_2}\phi'_{1,k_3}\mathrm{d}u \int_0^1 \phi_{2,l_1}\phi'_{2,l_2}\phi_{2,l_3}\mathrm{d}v\Big).\end{aligned} \tag{7.21}$$

Since α does not depend on the control points $\boldsymbol{c}$, it can be precomputed and stored in a look-up table in order to quickly evaluate the volume in interactive optimization schemes. Furthermore, because ϕ and its derivative ϕ' have compact support, the number of non-zero elements in the sum (7.20) is small, which additionally simplifies the computation. We derive the formula for the volume in Appendix 7.4.4. The integrals in (7.21) can be further simplified and exactly evaluated using techniques from spline theory similar to the approaches described in [172, 82].

7.1.6 Implementation

In this section, we describe some important details regarding the implementation.

Choice of Free Parameters

Exact Sphere. The orientation of the sphere is given by (7.4) and therefore, the coordinates of the poles are $\boldsymbol{c}_N = (0, 0, 1)$ for the north pole and $\boldsymbol{c}_S = (0, 0, -1)$ for the south pole. Since by construction, the vectors $\boldsymbol{T}_{1,N}$ and $\boldsymbol{T}_{2,N}$ span the tangent plane at the north pole of the sphere, a natural choice is to set $\boldsymbol{T}_{1,N} = (1, 0, 1) - \boldsymbol{c}_N = (1, 0, 0)$ and $\boldsymbol{T}_{2,N} = (0, 1, 1) - \boldsymbol{c}_N = (0, 1, 0)$. With the same approach we also obtain $\boldsymbol{T}_{1,S} = (1, 0, 0)$ and $\boldsymbol{T}_{2,S} = (0, 1, 0)$ for the south tangent plane. Note that because our construction is affine invariant, for a sphere with a different size or orientation the new coordinates are found by applying the corresponding affine transformation to the existing control points.

Arbitrary Shape with Spherical Topology. A simple method to estimate the tangent plane is to compute the plane that best approximates the points lying on the first circle of latitude next to the north or south pole. Any two vectors spanning this plane can be chosen as $\boldsymbol{T}_{1,N}$, $\boldsymbol{T}_{2,N}$ and $\boldsymbol{T}_{1,S}$ and $\boldsymbol{T}_{2,S}$.

Discretization of Basis Functions

Because our construction is formulated in the continuous domain, the shape representation can be discretized with arbitrary precision in order to implement it. An efficient way is to discretize the interpolator φ rather than the surface, which becomes highly beneficial, for example, in interactive applications where the shapes to be constructed are not known beforehand. By discretizing φ prior to surface construction, the samples of the interpolator can be stored in a look-up table to speed up the surface reconstruction. Thus, the sampling rate is freely chosen. In

Figures 7.13 and 7.14, we show the effect of the different sampling rates. A sampling rate equal to one corresponds to a polygon model (*i.e.*, linear interpolation between points), which means only the blue sample in Figure 7.13 is considered to be non-zero. The higher the sampling rate, the closer the approximation of the continuous domain model is. Its effect on the surface reconstruction is shown in Figure 7.14. In practice, if a large number of control points is used one can already obtain satisfactory smoothness of the surface with a low sampling rate, whereas with a small number of control points, the sampling rate must be higher to obtain a smooth surface.

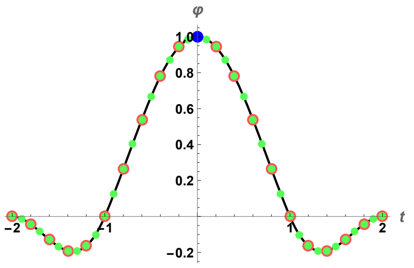


Figure 7.13: Sampling of the interpolator φ. Because φ is formulated in the continuous domain, it can be discretized with arbitrary precision. If only one sample is considered (blue sample), then the result corresponds to linear interpolation, which is equivalent to a polygon model. The samples denoted by orange circles correspond to a lower sampling rate than the green samples.

7.1.7 Discussion and Future Work

Comparison with NURBS. There are several advantages of our formulation compared to an approach using NURBS. With the NURBS formulation, a sphere can only be represented using multiple surfaces patches. The NURBS formulation requires not only more control points to represent a sphere, but also more total parameters due to the weights used in their formulation. Further, because the basis functions of NURBS are rational terms, the computation of derivatives

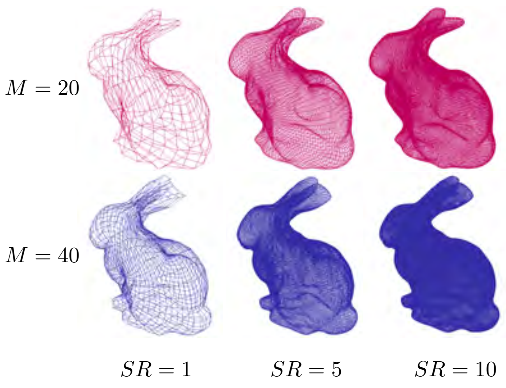


Figure 7.14: Effect of different sampling rates. The surfaces are constructed with $M = M_1 = M_2$. The red wireframe corresponds to a lower number of control points used ($M = 20$) to reconstruct the bunny compared to the blue wireframe ($M = 40$) and from left to right the sampling rate (SR) is increased.

and integrals result in complicated expressions. This can become a problem when integral-dependent quantities need to be computed, such as in the evaluation of surface or volume integrals in optimization schemes. Also, the optimization itself must be carried out simultaneously with respect to the control points and the weight parameters, which introduces additional complexity. For interactive shape design, the interpolation property of our framework makes the modeling task more intuitive. Especially, complex shapes that require more detail, and hence, more control points, are modeled more easily with our solution (see Figure 7.4). Furthermore, interpolating parameterized point clouds with spherical topology is difficult with NURBS due to their non-interpolatory nature; it involves complex NURBS approximation techniques or inverse filtering, which is not straight-forward because of the smoothness conditions at the poles. The only NURBS that are interpolatory are zero and first degree NURBS, which are non-smooth.
Moreover, our formulation allows for a shape representation using only integer

shifts. NURBS usually have non-uniform shifts. The advantage of considering integer shifts is that it allows us to use convolution and filtering techniques as well as frequency domain calculus. This can be useful when doing surface resampling, projections onto other spline spaces, and evaluation of inner products, for example to compute L_2-distances between surfaces. It also allows for a simpler formulation of the surface by specifying its control points instead of non-uniform knot vectors including double knots.

Comparison with Polygon Models. Polygon models are inherent interpolatory schemes because the control points coincide with the vertices of the mesh. Similar to subdivision schemes, these models require more parameters than our model in order to achieve a higher degree of smoothness (see Figure 7.8). Geometric operators and quantities, such as tangent planes, normals, curvatures, or the Laplacian have to be approximated by polygon mesh processing techniques. The same holds true for integral and derivative-based quantities. However, polygon meshes do not require an underlying parameterization of the model.

Comparison with the Catmull-Rom Interpolator. The Catmull-Rom [16] or Keys interpolator [127] are interpolating and smooth. Because they are polynomial it is not possible to construct the exact sphere with these functions. However, a construction of a model with spherical topology (which excludes exact spheres and ellipsoids) is possible by replacing φ in our framework with the Catmull-Rom spline. Because its support is the same as for φ our formulation for the smoothness and interpolation conditions at the poles can easily be adapted to the purely polynomial case.

Extending the Framework to Other Topologies. Our concept can be extended to surfaces with other topologies (*e.g.*, cylindrical or rectangular) in order to create a unifying framework for smooth shape modeling with interpolatory control points. These topologies do not require special attention to poles and are easier to parameterize using tensor-products and a suitable interpolator. One way to parameterize the rectangle is with the polynomial Keys interpolator [127], whereas the cylinder is parameterized using φ for the trigonometric part (*i.e.*, the circles in one direction) and the Keys interpolator for the linear part (*i.e.* the direction of the axis). Another example is the torus which is easily parameterized using φ since it is periodic in u and v. In Figure 7.15, examples of these topologies are shown as well as how they can be smoothly deformed by exploiting the interpolation property in interactive settings.

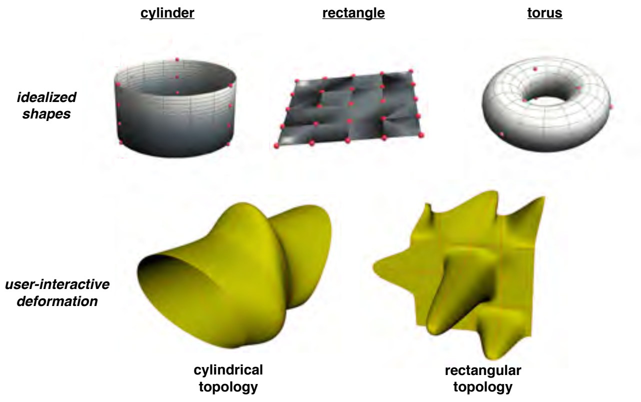


Figure 7.15: Smooth modeling of different topologies with interpolatory control points. In the top row, the idealized shapes that define the topologies are shown. The red points indicate the interpolatory control points. In the bottom row, a smooth deformation of the shapes is illustrated.

7.1.8 Conclusion

The standard method for smooth, parametric shape modeling in industry is NURBS. In this paper, we presented an alternative method to model smooth shapes with spherical topology. The fundamental difference with the existing standard is that our basis functions are interpolatory and non-rational and we only use uniform shifts. Our formulation is simpler than NURBS and thus, has several advantages in practical applications including immediate reconstruction of smooth surfaces by interpolating parameterized point clouds, more intuitive shape modeling or a simplified formulation of optimization algorithms that involve integral- and derivative-dependent quantities. Our framework indicates promising future directions by extending it to a richer family of topologies. A video illustrating the use of our framework in practice is available at *http://bigwww.epfl.ch/demo/siggraph2016/*.

7.2 Atlas-free Brain Segmentation in MRI

We present a new method for the atlas-free brain segmentation of proton-density-like 3D MRI images. We show how steerable filters can be efficiently combined with parametric spline surfaces to produce a fast and robust 3D brain segmentation algorithm. The novelty lies in the computation of brain edge maps through optimal steerable surface detectors which provide efficient energies for the rapid optimization of snakes. Our experimental results show the promising potential of the method for fast and accurate brain extraction.

7.2.1 Introduction

Brain-segmentation algorithms are extensively used to examine disease-related structural and morphological changes that occur in the brain. Such methods tend to be computationally expensive because 3D volumes need to be processed. Most of the algorithms rely on atlas-based registration methods, which make the overall algorithm computationally expensive [173, 174]. Furthermore, they might bias the outcome if either the patient scan or the registration algorithm do not match the template image well [175].

Active contours and surfaces (a.k.a. snakes) provide an alternative to atlas-based segmentation. They have been widely used to segment simple biomedical structures in 2D [176, 177]. However, snakes often require user interaction, which makes snakes less suitable for 3D medical imaging. We propose to make use of a 3D parametric spline snake for the atlas-free segmentation of the brain surface. Its parameterization allows us to implement a fast algorithm that has been proven to be competitive with the state of the art [86]. The segmentation is formulated as an energy-minimization problem [90]. Defining an efficient energy function is crucial for fast segmentation because it determines the speed of the optimization process as well as the accuracy of the result. In 3D, edge maps provide a convenient way to compute energy terms because they allow one to bypass the expensive evaluation of volume integrals at each iteration by replacing them by surface-based terms [86]. We show how steerable filters [178] combined with Canny-like criteria [179] can be used to compute edge maps through the implementation of optimal 3D steerable surface detectors [180, 171]. We have tested the efficiency of our proposed framework of steerable filters and parametric spline snakes for brain segmentation on realistic brain phantoms [181] and show its capability to be fast and robust.

7.2.2 Feature Detection With 3D Steerable Filters

To compute the 3D edge maps we make use of steerable filters. They were first introduced in [178] as a family of filters that can be efficiently rotated by representing them through a linear combination of appropriate basis filters. Therefore, steerable filters provide a convenient framework for rotation-invariant feature detection. We use Mth order steerable derivative-based filters whose impulse response takes the form

$$h(\boldsymbol{x}) = \sum_{m=1}^{M} \sum_{n=0}^{m} \sum_{p=0}^{m-n} \alpha_{m,n,p} \underbrace{\frac{\partial^n}{\partial x^n} \frac{\partial^p}{\partial y^p} \frac{\partial^{m-n-p}}{\partial z^{m-n-p}} g(\boldsymbol{x})}_{h_{m,n,p}(\boldsymbol{x})}$$

where g is an isotropic 3D Gaussian function and $\alpha_{m,n,p}$ are the weights of the basis functions. In this paper, we use $\boldsymbol{x}$ to describe a point (x, y, z) in 3D space.

Defining the rotation matrix $\boldsymbol{R}_{\theta,\phi}$, a feature with a particular orientation located in 3D space can be detected by a rotated version of the feature template $h(-\boldsymbol{x})$ through estimation of its Euler angles by

$$(\theta^*(\boldsymbol{x}), \phi^*(\boldsymbol{x})) = \underset{\theta,\phi}{\operatorname{argmax}}(f(\boldsymbol{x}) * h(\boldsymbol{R}_{\theta,\phi}\boldsymbol{x})).$$

The response of the filter is given by

$$r^*(\boldsymbol{x}) = f(\boldsymbol{x}) * h(\boldsymbol{R}_{\theta^*,\phi^*}\boldsymbol{x}).$$

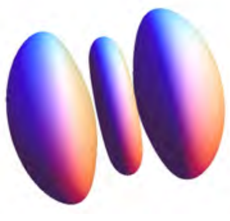


Figure 7.16: Isosurface representation of the optimal surface detector.

Designing the Optimal Surface Detector

The generalized optimality criterion to derive a feature detector was originally proposed in 2D in [180] and extended to 3D in [171]. We use it to derive an optimal 2nd-order surface detector. We choose $f_0(\boldsymbol{x}) = \delta(x)$ as our idealized surface template, where δ denotes the Dirac delta. The response of the filter to the surface template centered at the origin is given by

$$S = (f_0 * h)(\mathbf{0}) = \int_{\mathbb{R}^3} f_0(\boldsymbol{x}) h(-\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

The localization error (due to the presence of noise) in the direction orthogonal to the surface is quantified by

$$Loc = -\int_{\mathbb{R}^3} f_0(\boldsymbol{x}) \frac{\partial^2}{\partial x^2} h(-\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

We maximize $(S \cdot Loc)$ using Lagrangian optimization, while imposing unit energy on the filter as $\int_{\mathbb{R}^3} |h(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} = 1$.

This yields the optimal surface detector

$$h(\boldsymbol{x}) = \frac{\sigma}{8\pi\sqrt{3}} (\Delta g(\|\boldsymbol{x}\|) - 5 g_{xx}(\boldsymbol{x})) \tag{7.22}$$

where Δ denotes the Laplacian operator, σ is the standard deviation of the Gaussian and $g_{xx} = \frac{\partial^2 g}{\partial x^2}$. An isosurface representation of (7.22) is shown in Figure 7.16.

Surface Detection

To express the rotated version of (7.22), we make use of the property $\mathcal{D}_v^2 f = \boldsymbol{v}^T \boldsymbol{H}_f \boldsymbol{v}$, where $\mathcal{D}_v$ denotes the operator describing the directional derivative, $\boldsymbol{H}_f$ is the 3D Hessian matrix of f, and $\boldsymbol{v} = (\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi)$ is a unit vector specifying an arbitrary orientation in 3D. Thus, $h(\boldsymbol{R}_{\theta,\phi}\boldsymbol{x}) \propto \Delta g(\|\boldsymbol{x}\|) - 5\boldsymbol{v}^T \boldsymbol{H}_g(\boldsymbol{x})\boldsymbol{v} = \boldsymbol{\beta}_g(\boldsymbol{x})$ and

$$(f * h(\boldsymbol{R}_{\theta,\phi}\cdot))(\boldsymbol{x}) \propto \boldsymbol{v}^T \boldsymbol{\beta}_{f*g}(\boldsymbol{x})\boldsymbol{v}. \tag{7.23}$$

Applying the constraint $\boldsymbol{v}^T\boldsymbol{v} = 1$ on the unit vector $\boldsymbol{v}$ and maximizing (7.23), we obtain $\boldsymbol{\beta}_{f*g}\boldsymbol{v} = \lambda\boldsymbol{v}$. The optimal orientation is given by the eigenvector $\boldsymbol{v}_{max}$

corresponding to the largest eigenvalue λ_{max} of $\boldsymbol{\beta}_{f*g}$, which also yields the maximum response of the detector. The result of the surface detection is shown in Figure 7.17 (top right).

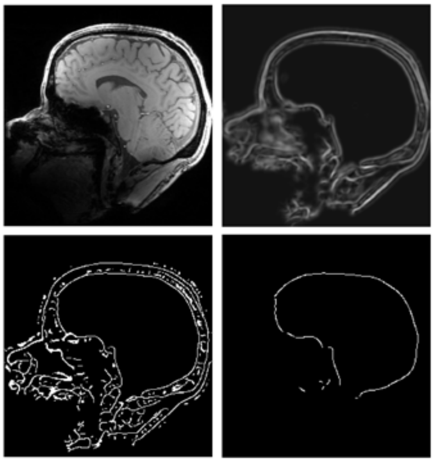


Figure 7.17: 2D sagittal cross section of the detected 3D surface. Top row: Original proton-density image (left) and result of steerable filtering (right). Bottom row: Result of NMS and thresholding (left) and extracted largest component (right).

Surface Refinement

In order to better delineate the surface, a classical non-maximum suppression (NMS) in the direction orthogonal to the surface is applied followed by a thresholding step in order to obtain a binary image (Figure 7.17, bottom right). The direction orthogonal to the surface is given by $\boldsymbol{v}_{max}$ described above. In Figure 7.17 (bottom left) we see that the brain appears as the innermost surface and is almost closed. To ensure additional robustness, we extract this inner surface before segmenting it. For this purpose we roughly estimate the center of the brain. Then, we iterate over all foreground voxels, and keep only the ones closest to the center (in the direction orthogonal to the surface). Among the retained voxels we extract the largest component, whose 2D representation is shown in Figure 7.17 (bottom right).

7.2.3 3D Parametric Spline Snake

We use the continuously defined 3D spline snake proposed in [86] to segment the extracted brain surface. It is a parametric surface whose expression is

$$\boldsymbol{\sigma}(u,v) = \sum_{i=0}^{M_1-1} \sum_{j=-1}^{M_2+1} \mathbf{c}[i,j]\phi_{1,per}(M_1 u - i)\phi_2(M_2 v - j) \tag{7.24}$$

where $\phi_{1,per}(u) = \sum_{n=-\infty}^{\infty} \phi_1(u - M_1 n), \forall u \in \mathbb{R}$, and $\mathbf{c}[i,j] \in \mathbb{R}^3$ are the control points in 3D. In (7.24), the basis functions ϕ are made of exponential B-splines and are defined as

$$\phi_1(u) = \begin{cases} \frac{\cos\left(\frac{2\pi|u|}{M_1}\right)\cos\left(\frac{\pi}{M_1}\right) - \cos\left(\frac{2\pi}{M_1}\right)}{1-\cos\left(\frac{2\pi}{M_1}\right)} & 0 \le |u| < \frac{1}{2} \\ \frac{1-\cos\left(\frac{2\pi(3/2-|u|)}{M_1}\right)}{2\left(1-\cos\left(\frac{2\pi}{M_1}\right)\right)} & \frac{1}{2} \le |u| < \frac{3}{2} \\ 0 & \frac{3}{2} \le |u| \end{cases}$$

$$\phi_2(v) = \begin{cases} \frac{\cos\left(\frac{\pi|v|}{M_2}\right)\cos\left(\frac{\pi}{2M_2}\right) - \cos\left(\frac{\pi}{M_2}\right)}{1-\cos\left(\frac{\pi}{M_2}\right)} & 0 \le |v| < \frac{1}{2} \\ \frac{1-\cos\left(\frac{\pi(3/2-|v|)}{M_2}\right)}{2\left(1-\cos\left(\frac{\pi}{M_2}\right)\right)} & \frac{1}{2} \le |v| < \frac{3}{2} \\ 0 & \frac{3}{2} \le |v|. \end{cases}$$

Considering additional conditions on the poles of the surface, a total of $M_1(M_2 - 1) + 4$ control points are necessary to define the snake in a continuous way. Due to its spline-based structure, the snake surface can adopt the shape of any kind of closed surface with arbitrary precision.

Fast Optimization

The segmentation process is formulated as an energy-minimization problem. The refined surface image (Figure 7.17, bottom right) provides an ideal edge map to guide the snake towards the desired boundary. We use it to calculate the gradient energy proposed in [86], which is defined as

$$\begin{aligned} E_{grad} &= -\oiint_S \boldsymbol{\nabla} f \cdot \mathrm{d}\boldsymbol{S} = -\oiint_S \left(\boldsymbol{\nabla} f \cdot \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|} \right) \mathrm{d}S \\ &= -\iiint_V \mathrm{div}(\boldsymbol{\nabla} f)\mathrm{d}V = \iiint_V -\Delta f \mathrm{d}V \\ &= -\oiint_{\partial V} (\Delta f)^x \mathrm{d}y \wedge \mathrm{d}z \end{aligned} \tag{7.25}$$

where $(\Delta f)^x = \int_{-\infty}^{x} \Delta f(\tau, y, z)\mathrm{d}\tau$ and $\wedge$ denotes the wedge product. In the last step of (7.25), Gauss' theorem has been used. The quantity $(\Delta f)^x$ can be pre-computed and stored in a look-up table to allow fast energy computation. The optimization process is visualized in Figure 7.18.

Snake Initialization

We initialized the snake as an ellipsoid lying completely within the brain surface (Figure 7.18, top row). Therefore, during optimization, the snake will primarily expand rather than shrink.

7.2.4 Experiments

We validated our algorithm on the BrainWeb PD phantom [181], where the ground truth is known. Our method was tested with the original bias- and noise-free image, as well as with increasing radio-frequency (RF) intensity non-uniformities of 20% and 40% and increasing additive white Gaussian noise levels ranging from 1 to 9%. Measures of overlap between the segmented brain masks and the gold standard were calculated according to the Dice and Jaccard similarity indices. Additionally, the inter-subject variability was evaluated by computing similarity coefficients between the corrupted images and the bias- and noise-free image. The initial position of the surface snake was the same throughout the experiments. We used $M_1 = M_2 = 9$ to evaluate (7.24). Therefore, 76 control points were used. The experiments were run without user interaction.

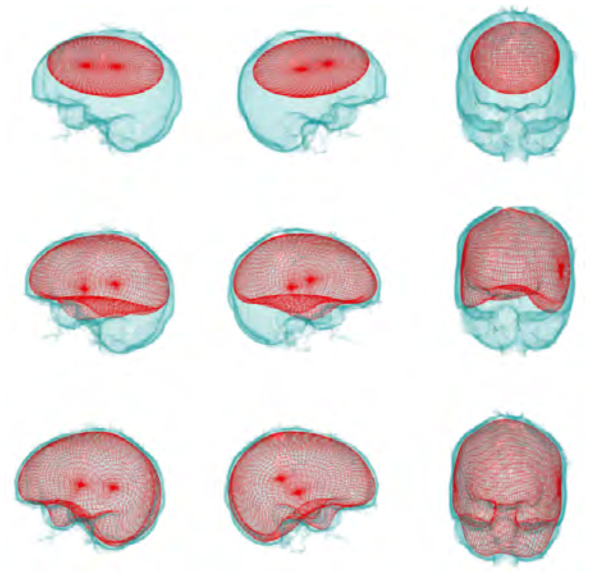


Figure 7.18: Wireframe representation of the parametric surface. The 3D spline snake is initialized as an ellipsoid (top row). Through minimization of the energy term (7.25), it segments the brain surface (middle row). The final segmentation result is shown in the bottom row.

Robustness, Accuracy, and Computational Aspects

The results of the measures of similarity are given in Table 7.1 and illustrated in Figure 7.19. The high degree of overlap with respect to the gold standard confirms the selectivity of the surface detector. Its capacity to detect surface elements is not hampered by bias or noise in the image. This is due to its property of being an optimal 'matched' detector. The measures for inter-subject variability are even higher and show additional robustness of the method with respect to different scanning conditions. Worsening the image quality did not significantly affect the final segmentation, which indicates the reproducibility of the method. Besides, the algorithm was also tested on 12 real subjects with a satisfying outcome (data not shown). Furthermore, our algorithm is fast enough to be run in a doctor-patient encounter. The implementation of our algorithm executes in less than 70 seconds on average on a standard computer (3.3 GHz, 16GB RAM).

	gold standard	inter subject
Dice	0.9522 ± 0.0011	$0.9806\pm$ 0.0047
Jaccard	0.9087 ± 0.0020	0.9620 ± 0.0090

Table 7.1: Dice and Jaccard similarity coefficients.

7.2.5 Conclusion

We provide a new solution for fast brain segmentation in 3D MRI proton-density-like images. Our study shows how 3D parametric spline snakes can be used for this purpose. Our method relies on 3D edge maps which allow fast snake optimization through the computation of surface integrals, thereby avoiding tedious integration of volumes. Based on steerable filters, we show how an optimal surface detector can be used to compute such edge maps. We have demonstrated the robustness of the proposed method with respect to radio-frequency-induced intensity inhomogeneities, as well as noise. Our algorithm is fast and atlas-free. No user interaction is required. It therefore shows the potential to satisfy the requirements needed to be run in clinical routine.

7.3 Medical Segmentation of Structures with Cylindrical Topology

We propose a new parametric 3D snake with cylindrical topology. Its construction is based on interpolatory spline bases which facilitates user-interaction because the control points of the snake directly lie on the surface of the deformable cylinder. We prove that the basis function exactly reproduce a cylinder and propose a new parametrization as a tensor-product spline surface. We provide explicit formulas for the energy function which allows to compute a closed-form expression of its gradient enabling a fast implementation of the optimization algorithm. We have

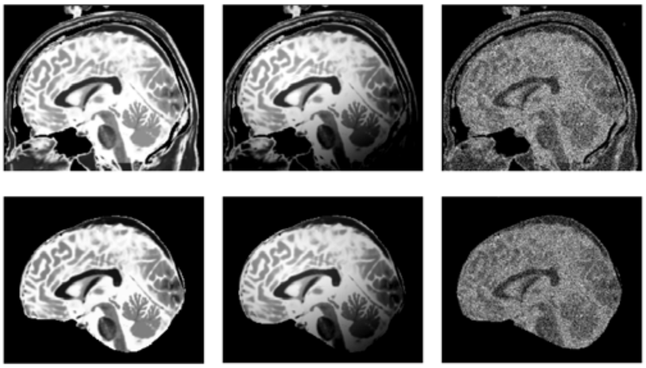


Figure 7.19: 2D sagittal cross section of 3D simulated Brainweb data (top row) and the corresponding segmented images (bottom row). Left column: noise and bias free images; middle: biased images (40% RF intensity non-uniformity); right: noisy images (9% additive Gaussian white noise).

implemented the proposed framework as a freely available open-source plugin for the bioimaging platform Icy. Its utility has been tested on artificial data as well as on real 3D data to segment the spinal cord and the ascending aorta.

7.3.1 Introduction

The assessement of quantitative parameters of medical structures using 3D imaging modalities is an active field of research. Accurate measurements of physiological structures are crucial for correct diagnosis of diseases, risks or malformations. We propose a new method for the segmentation of structures with cylinder-like topology such as the aorta or the spinal cord. Most algorithms that have been proposed for the aorta segmentation so far target computed tomography (CT) images as opposed to magnetic resonance images (MRI) because of the significantly better contrast [182, 183]. However, the MRI modality does not expose the patients to dangerous radiation. Furthermore, methods to segment cardiovascular structures are usually fully automatic which means that wrong segmentation results can not be corrected and intricate self-correction routines are necessary for quality check which can be time consuming [184]. Besides, algorithms that are atlas-based need to deal with the problem of how to construct an accurate atlas representing a whole patient population [185]. Our new proposed method is atlas-free and accounts for the possibility of user-interaction. It relies on a new parameterization of the cylinder using compactly supported basis functions. We show that the basis perfectly reproduces the cylinder as a tensor-product spline surfaces. Furthermore, it is interpolatory which implies that the control points of the shape, that can be modified by the user, directly lie on the surface; a property that allows intuitive and easy user-interaction. Using the new proposed parametric surface we construct a 3D snake and provide an explicit expression for a contour-based energy function that attracts the snake towards the boundary of interest [86]. The provided energy function enables an explicit computation of its gradient; a property that can be exploited for an efficient implementation. Finally, we have tested the robustness w.r.t. to noise on test data and we have performed an evaluation on real data using a cohort of 14 healthy subjects [186] to segment the ascending thoracic aorta, a region that is can be of interest to measure hemodynamics after thoracic endovascular aortic repair as well as aneurisms in this region [187].

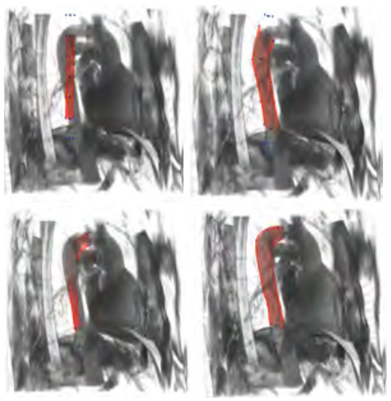


Figure 7.20: Segmentation of the aorta with the cylinder snake in 3D MRI. The red wireframe represents the surface of the snake and the blue dots are the control points.

7.3.2 New Parameterization of the Cylinder

In this section we present a new parametrization of the cylinder as a tensor-product spline surface. For this purpose our basis functions need to be able to reproduce circles and ellipses as well as first degree polynomials.

Reproduction of circles and ellipses

We consider the basis function φ_1 of the following form, which belongs to the family of interpolators presented in Section 6.2:

$$\varphi_1(t) = \lambda_1 \beta_{\boldsymbol{\alpha}_1}(t) - \lambda_2[\beta_{\boldsymbol{\alpha}_2}(t) + \beta_{\boldsymbol{\alpha}_2}(t-1)], \tag{7.26}$$

where

$$\lambda_1(M) = \frac{2\pi^3}{M^2\left(M\sin\left(\frac{2\pi}{M}\right) - \pi\cos\left(\frac{2\pi}{M}\right) - \pi\right)} \tag{7.27}$$

$$\lambda_2(M) = \frac{\pi^2\left(2\pi - M\sin\left(\frac{2\pi}{M}\right)\right)}{M^2\left(\cos\left(\frac{2\pi}{M}\right) - 1\right)\left(M\sin\left(\frac{2\pi}{M}\right) - \pi\cos\left(\frac{2\pi}{M}\right) - \pi\right)} \tag{7.28}$$

and $\hat{\beta}_{\boldsymbol{\alpha}}(\omega) = \prod_{k=1}^{n} \frac{1-e^{\alpha_k - j\omega}}{j\omega - \alpha_k}$ is the n-th order causal exponential B-spline defined in the Fourier domain and the poles are given by $\boldsymbol{\alpha}_1 = (0, 0, \frac{j2\pi}{M}, -\frac{j2\pi}{M})$ and $\boldsymbol{\alpha}_2 = (0, \frac{j2\pi}{M}, -\frac{j2\pi}{M})$.

Proposition 19. *The basis ϕ_1 is an interpolator and is capable of reproducing the complex exponentials $e^{j2\pi t}$ and $e^{-j2\pi t}$ independent of the number of control points $M \geq 3$.*

Proof: In order to show that ϕ_1 is an interpolator we notice that the n-th order exponential B-spline has support n. Thus, φ_1 has a support equal to 4. By imposing the corrsponding interpolation conditions on (7.26) and solving for λ_1 and λ_2 we obtain the weights given by (7.27) and (7.28). In order to prove the reproduction properties of φ_1 we use the exponential reproduction properties from the exponential B-spline, *i.e.* if $\alpha \in \boldsymbol{\alpha}$, then there exists a sequence $p[k]$ such that $e^{\alpha t} = \sum_{k\in\mathbb{Z}} p[k]\beta_{\boldsymbol{\alpha}}(t-k)$ [63]. Therefore, considering $\alpha \in \boldsymbol{\alpha}_2 \in \boldsymbol{\alpha}_1$ we obtain

$$\begin{aligned}&\sum_{k\in\mathbb{Z}} p[k]\Big[\lambda_1\beta_{\boldsymbol{\alpha}_2}(t-k) - \lambda_2(\beta_{\boldsymbol{\alpha}_2}(t-k) + \beta_{\boldsymbol{\alpha}_2}(t-k-1))\Big]\\ &= \lambda_1 e^{\alpha t} - \lambda_2(e^{\alpha t} + e^{\alpha(t-1)})\\ &= e^{\alpha t}(\lambda_1 - \lambda_2(1+e^{-\alpha}))\end{aligned} \tag{7.29}$$

From [63] (Proposition 2) we know that if a function ψ reproduces exponential polynomials then $\tilde{\psi} * \psi$ also reproduces these exponential polynomials if $\tilde{\psi}$ satisfies some mild conditions. Thus, since $\beta_{\boldsymbol{\alpha}_1} = \beta_0 * \beta_{\boldsymbol{\alpha}_2}$ and

$$\begin{aligned}&\sum_{k\in\mathbb{Z}} \varphi_1(t-k) =\\ &\sum_{k\in\mathbb{Z}} p[k]\Big[\lambda_1(\beta_0 * \beta_{\boldsymbol{\alpha}_2})(t-k) - \lambda_2(\beta_{\boldsymbol{\alpha}_2}(t-k) + \beta_{\boldsymbol{\alpha}_2}(t-k-1))\Big]\end{aligned} \tag{7.30}$$

we see that φ_1 also reproduces the exponential polynomials given by (7.29). The last pont of Proposition 1, $M \geq 3$ is due to the fact that φ_1 needs to form a Riesz basis in order to guarantee a unique and stable representation of the resulting parametric surface. It has been shown in [63] that this is only verified if for all pairs of distinct purely imaginary elements of $\boldsymbol{\alpha}$ we have $\alpha_m - \alpha_n \neq j2\pi k, k \in \mathbb{Z}$. Hence, from the definition of α_1 and α_2 we directly see that $M \geq 3$ ■

Corollary 2. *The interpolator* φ_1 *reproduces* $\cos(2\pi t)$ *and* $\sin(2\pi t)$ *independent of the number of control points* $M \geq 3$.

Proof: By exploiting the exponential reproduction property of B-splines and the property of φ_1 of being an interpolator we write

$$\cos(\frac{2\pi t}{M}) = \frac{e^{\frac{j2\pi t}{M}} + e^{-\frac{j2\pi t}{M}}}{2} = \sum_{k\in\mathbb{Z}} \left[\frac{e^{\frac{j2\pi k}{M}} + e^{-\frac{j2\pi k}{M}}}{2}\right]\varphi_1(t-k) \tag{7.31}$$

and therefore

$$\cos(2\pi t) = \sum_{k\in\mathbb{Z}} \cos[2\pi k]\varphi_1(Mt-k) \tag{7.32}$$

In a similar way we obtain $\sin(2\pi t)$. ■

Plots of the reconstructed trigonometric functions are shown in Figure 7.21 as well as the circle $\boldsymbol{r}$ obtained through the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$.

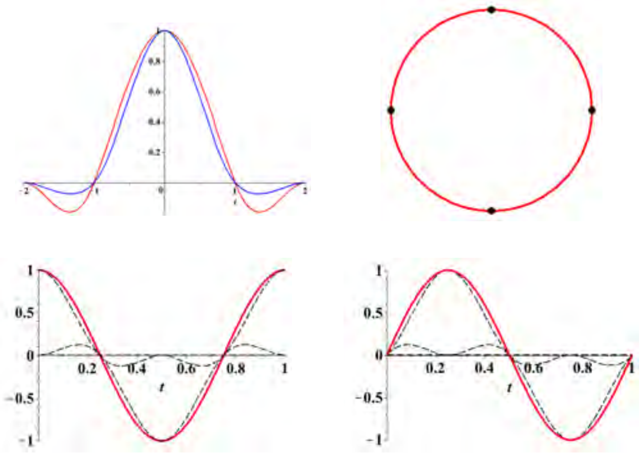


Figure 7.21: Top left: Keys interpolator (blue) and the proposed ellipse reproducing interpolator (red). Top right: the circle obtained with the parametric equation $\boldsymbol{r}(t) = (\cos(2\pi t), \sin(2\pi t))$. Bottom: $\cos(2\pi t)$ (left) and $\sin(2\pi t)$ (right) are shown together with the basis functions for $M = 3$.

Reproduction of 1st degree polynomials

The parametric representation of the cylinder as a spline surface requires that at least one basis function reproduces 1st degree polynomials. We use the Keys interpolator which is in $\mathcal{C}^1$ and reproduces 2nd degree polynomials. It is given by

$$\varphi_2(t) = \begin{cases} \frac{1}{2}\left(2 - t^2(3t+5)\right) & -1 < t \leq 0 \\ -\frac{1}{2}(t-2)^2(t-1) & 1 \leq t < 2 \\ \frac{1}{2}(t+1)(t+2)^2 & -2 < t \leq -1 \\ \frac{1}{2}\left((3t-5)t^2+2\right) & 0 < t < 1 \end{cases}. \tag{7.33}$$

Its support is also equal to 4 as for φ_1. A plot of the line that has been reproduced with φ_2 is shown in figure 7.22.

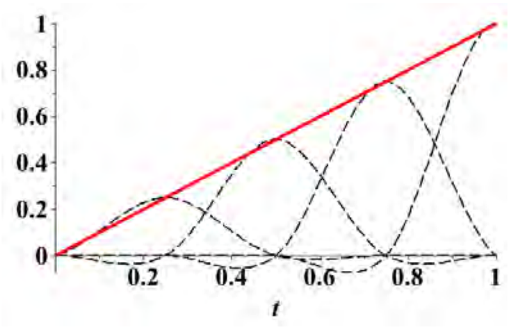


Figure 7.22: Reproduction of the line and weighted basis functions.

Reproduction of the cylinder

Proposition 20. *The normalized cylinder surface can be expressed as*

$$\boldsymbol{\sigma}(u,v) = \sum_{k=0}^{M-1} \sum_{k=-1}^{M+1} \mathbf{c}[k,l]\varphi_{1,M}(Mu-k)\varphi_2(Mv-l), \tag{7.34}$$

where $u, v \in [0,1]$ *and* $\varphi_{1,M}$ *is the* M*-periodization of* φ_1 *and the control points of the surface are given by*

$$\boldsymbol{c}[k,l] = \begin{pmatrix} \cos[\frac{2\pi k}{M}] \\ \sin[\frac{2\pi k}{M}] \\ \frac{l}{M} \end{pmatrix}. \tag{7.35}$$

Proof: By using the $M-$periodized basis function of φ_1

$\varphi_{1,M}(Mt-k) = \sum\limits_{n=-\infty}^{+\infty} \varphi_1(M(t-n)-k)$, we can re-express the cosine in (7.32) as

$\cos(2\pi t) = \sum\limits_{k=0}^{M-1} \cos[\frac{2\pi k}{M}]\varphi_{1,M}(Mt-k)$. In a similar way we obtain

$\sin(2\pi t) = \sum\limits_{k=0}^{M-1} \sin[\frac{2\pi k}{M}]\varphi_{1,M}(Mt-k)$. The reproduction of the straight line can be expressed as $\sum_{k\in\mathbb{Z}} \frac{k}{M}\varphi_2(Mt-k)$. If we enforce t to lie in the interval $[0,1]$ and because the support of φ_2 is limited to $[-2,2]$ the summation can be restricted to the indices $k \in [-1, M+1]$. Since φ_1 and φ_2 satisfy the partition of unity condition, *i.e.* $\sum_{k=-\infty}^{\infty} \varphi(t-k) = 1$ (proof ommitted) we can develop the standard parametrization of the cylinder for $u, v \in [0,1]$ as follows:

$$\begin{aligned}
\boldsymbol{\sigma}(u,v) &= \begin{pmatrix} \cos(2\pi u) \\ \sin(2\pi u) \\ v \end{pmatrix} \\
&= \sum_{k=0}^{M-1} \begin{pmatrix} \cos[\frac{2\pi k}{M}] \\ \sin[\frac{2\pi k}{M}] \\ 1 \end{pmatrix} \varphi_{1,M}(Mu-k) \cdot \sum_{k=-1}^{M+1} \begin{pmatrix} 1 \\ 1 \\ \frac{k}{M} \end{pmatrix} \varphi_2(Mv-k) \qquad (7.36) \\
&= \sum_{k=0}^{M-1} \sum_{k=-1}^{M+1} \begin{pmatrix} \cos[\frac{2\pi k}{M}] \\ \sin[\frac{2\pi k}{M}] \\ \frac{k}{M} \end{pmatrix} \varphi_{1,M}(Mu-k)\varphi_2(Mv-l)
\end{aligned}$$

■

An important property of the surface that we use to construct the snake is that it must guarantee to be able to outline shapes irrespective of their size, orientation and position. It must therefore be invariant to affine transformations, *i.e.*

$$\mathbf{A}\boldsymbol{\sigma}(u,v)+\mathbf{b} = \sum_{k=0}^{M-1}\sum_{l=-1}^{M+1}(\mathbf{A}\boldsymbol{c}[k,l]+\mathbf{b})\varphi_{1,M}(Mu-k)\varphi_2(Mv-l), \tag{7.37}$$

where $\mathbf{A}$ is a 3×3 matrix and $\mathbf{b}\in\mathbb{R}^3$. It is easy to see that equation (7.37) is automatically satisfied if the basis functions φ_1 and φ satisfy the partition of unity. A plot of the resulting surface is shown in figure 7.23.

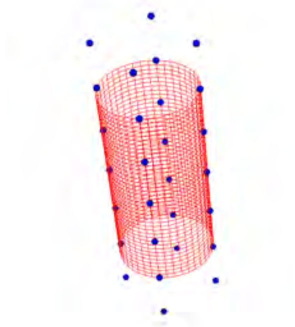


Figure 7.23: Wireframe representation of the cylinder created as a tensor-product spline surface given by Proposition 2. The blue points are the control points. Note that the few non-interpolatory control points are due to the fact that the boundary conditions for the reproduction of the z-coordinate in (7.34) are not periodic.

7.3.3 3D Parametric Spline Snake

To construct the snake we need to define an energy functional that can be minimized in order to attract the snake surface towards the boundary of interest. We use a gradient-based energy similar to the one proposed by [86]. It is given by

$$\begin{aligned} E_{\text{grad}} &= -\iint_{\partial\Omega} \Delta^x \mathrm{d}y \wedge \mathrm{d}z \\ &= -\int_0^1 \int_0^1 \Delta^x(\boldsymbol{\sigma}(u,v)) \left(\frac{\partial y}{\partial u}\frac{\partial z}{\partial v} - \frac{\partial y}{\partial v}\frac{\partial z}{\partial u} \right) \mathrm{d}u\mathrm{d}v, \end{aligned} \tag{7.38}$$

where $\Delta^x(x,y,z) = \int_{-\infty}^{x} \Delta(\tau,y,z)\mathrm{d}\tau$ can be pre-integrated and stored in a lookup table in order to speed up the computational process. The partial derivatives in (7.38) can be computed through

$$\frac{\partial \boldsymbol{\sigma}}{\partial u}(u,v) = M \sum_{k=0}^{M-1} \sum_{l=-1}^{M+1} \mathbf{c}[k,l] \dot{\varphi}_{1,M}(Mu-k)\varphi_2(Mv-l) \tag{7.39}$$

$$\frac{\partial \boldsymbol{\sigma}}{\partial u}(u,v) = M \sum_{k=0}^{M-1} \sum_{l=-1}^{M+1} \mathbf{c}[k,l] \varphi_{1,M}(Mu-k)\dot{\varphi}_2(Mv-l), \tag{7.40}$$

where $\frac{\partial \boldsymbol{\sigma}}{\partial u} = (\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u})$ and $\frac{\partial \boldsymbol{\sigma}}{\partial v} = (\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v})$.

Equation (7.38) allows to compute a closed-form expression of its gradient w.r.t. to the contorl points, which allows an efficient implementation of the optimization algorithm.

7.3.4 Experiments

We have implemented and validated our proposed framework on artificial as well as on real 3D MRI. For the artificial data we have created a perfect 3D hollow cylinder and corrupted the image with increasing levels of additive Gaussian white noise. The overlap between the initial position of the snake and the perfect cylinder corresponds to a Jaccard index of .12. The signal-to-noise ratios (SNR) and resulting overlap measures are shown in table 7.2.

In order to validate the snake on real data we have manually segmented the thoracic ascending aorta on 14 scans taken from healthy subjects. The segmentation was carried out by an expert clinician and represents the gold standard. The mean overall measures w.r.t. to the initial position and segmentation results are shown in table 7.3 as well as the standard deviations (std).

Table 7.2: Jaccard indices for segmentation of (noisy) data.

SNR [dB] (stdd)	Jaccard index
∞ (-)	0.94
9.91 (10)	0.94
0.27 (30)	0.94
-4.17 (50)	0.94
-7.10 (70)	0.92
-9.21 (90)	0.92

Table 7.3: Mean Jaccard indices for segmentation of real data.

–	mean Jaccard index (std)
initialization	0.23 (0.23)
result	0.96 (0.02)

The optimization is carried out by a Powell-like line-search method [188]. The segmentation took less than 4 seconds on average on a 2.3 GHZ processor with 8 GB RAM. Furthermore, we have also successfully tested the framework for computed tomography data and for the segmentation of the spinal cord in 3D MRI (results will be published elsewhere). An illustration of a real 3D MRI scan where the spinal cord and the thoracic ascending aorta are segmented are shown in figure 7.24.

7.3.5 Conclusion

We have proposed a novel parameterization of the cylinder in order to construct a 3D snake. We have shown how to perfectly reproduce the cylindrical topology using interpolatory basis functions. They have the advantage that a tensor-product spline surface can be constructed where the control points lie on the surface itself; an advantage for user-interactive applications. We provide an explicit formulation for a gradient energy and the results obtained on real data are promising. Furthermore,

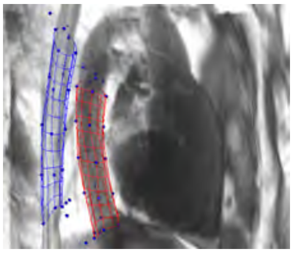


Figure 7.24: Simultaneous segmentation of the spinal cord (blue) and the thoracic ascending aorta (red).

our experiments on test data show that the proposed algorithm is robust to noise.

7.4 Appendix

7.4.1 Smoothness Conditions at Poles

The left-hand-side of (7.9) is developed as

$$
\begin{aligned}
&\left.\frac{\partial \boldsymbol{\sigma}(u,v)}{v}\right|_{v=0} = \\
&M_2 \sum_{k=0}^{M_1-1} \sum_{l=-1}^{M_2+1} \boldsymbol{c}[k,l]\varphi_{M_1,\mathrm{per}}(M_1u-k)\varphi'_{2M_2}(M_2v-l)\Big|_{v=0} \\
&= M_2 \sum_{k=0}^{M_1-1} \varphi_{M_1,\mathrm{per}}(M_1u-k) \sum_{l=-1}^{M_2+1} \boldsymbol{c}[k,l]\varphi'_{2M_2}(-l) \\
&= M_2 \sum_{k=0}^{M_1-1} \varphi_{M_1,\mathrm{per}}(M_1u-k) \\
&\times \Big(\boldsymbol{c}[k,-1]\varphi'_{2M_2}(1) + \boldsymbol{c}[k,0]\varphi'_{2M_2}(0) + \boldsymbol{c}[k,1]\varphi'_{2M_2}(-1)\Big) \\
&= M_2 \sum_{k=0}^{M_1-1} \varphi_{M_1,\mathrm{per}}(M_1u-k) \\
&\times \Big(\varphi'_{2M_2}(-1)(\boldsymbol{c}[k,-1] - \boldsymbol{c}[k,1])\Big),
\end{aligned}
\tag{7.41}
$$

where we have used the fact that φ' is odd (φ is even). The right-hand-side of (7.9) is expressed as

$$\begin{aligned}
&\boldsymbol{T}_{1,N}\cos(2\pi u)+\boldsymbol{T}_{2,N}\sin(2\pi u)\\
&=\boldsymbol{T}_{1,N}\sum_{k=0}^{M_1-1}\cos\big(\frac{2\pi k}{M_1}\big)\varphi_{M_1,\mathrm{per}}(M_1u-k)\\
&+\boldsymbol{T}_{2,N}\sum_{k=0}^{M_1-1}\sin\big(\frac{2\pi k}{M_1}\big)\varphi_{M_1,\mathrm{per}}(M_1u-k)\\
&=\sum_{k=0}^{M_1-1}\Big(\boldsymbol{T}_{1,N}\cos\big(\frac{2\pi k}{M_1}\big)+\boldsymbol{T}_{2,N}\sin\big(\frac{2\pi k}{M_1}\big)\Big)\\
&\times\varphi_{M_1,\mathrm{per}}(M_1u-k).
\end{aligned}\tag{7.42}$$

By equating (7.41) to (7.42) and by identifying the coefficients, we obtain (7.11) and (7.12).

7.4.2 Interpolation Conditions at Poles

At the north pole, we compute

$$\boldsymbol{\sigma}(u,0)=\sum_{k=0}^{M_1-1}\sum_{l=-1}^{M_2+1}\mathbf{c}[k,l]\varphi_{M_1,\mathrm{per}}(M_1u-k)\varphi_{2M_2}(-l).\tag{7.43}$$

Since φ_M satisfies the interpolation condition, the term that depends on l is always zero unless $l=0\Leftrightarrow\varphi_{2M_2}(l=0)=1$. Hence, (7.43) simplifies to

$$\boldsymbol{\sigma}(u,0)=\sum_{k=0}^{M_1-1}\mathbf{c}[k,0]\varphi_{M_1,\mathrm{per}}(M_1u-k):=\boldsymbol{c}_N.\tag{7.44}$$

Because the integer shifts of φ_{M_1} build a basis [76] and φ_{M_1} satisfies the partition of unity, (7.44) only holds if $\mathbf{c}[k,0]=\boldsymbol{C}$, with $\boldsymbol{C}$ a constant vector for all k. Thus,

$$\begin{aligned}\boldsymbol{\sigma}(u,0) &= \sum_{k=0}^{M_1-1} \boldsymbol{C}\cdot\varphi_{M_1,\mathrm{per}}(M_1u-k)\\ &= \boldsymbol{C}\cdot\sum_{k=0}^{M_1-1} \underbrace{\varphi_{M_1,\mathrm{per}}(M_1u-k)}_{=1}\\ &= \boldsymbol{C}\\ &= \boldsymbol{c}[k,0] = \boldsymbol{c}_N \quad \forall k\in[0\ldots M_1-1].\end{aligned}$$

A similar derivation leads to the interpolation condition at the south pole.

7.4.3 Flux Across Surface

We denote by $\boldsymbol{n}$ the normal vector to the surface and make use of the divergence theorem to compute

$$\begin{aligned}E(\boldsymbol{\sigma}) &= \oiint_S \boldsymbol{f}\cdot\mathrm{d}\boldsymbol{S} = \oiint_S \Big(\boldsymbol{f}\cdot\frac{\boldsymbol{n}}{\|n\|}\Big)\mathrm{d}S\\ &= \iiint_V \underbrace{\mathrm{div}\boldsymbol{f}}_{g}\,\mathrm{d}V = \oiint_{\partial V=S} g^x\mathrm{d}y\wedge dz\\ &= \oiint_{\partial V=S} g^y\mathrm{d}x\wedge dz = \oiint_{\partial V=S} g^z\mathrm{d}x\wedge dy,\end{aligned}$$

where g^x, g^y, g^z are the pre-integrated functions along the dimensions x, y or z. The wedge operator is defined as

$$\mathrm{d}y\wedge dz = \frac{\partial y}{\partial u}\frac{\partial z}{\partial v} - \frac{\partial y}{\partial v}\frac{\partial z}{\partial u} \tag{7.45}$$

and is explicitly computed using $\frac{\partial \boldsymbol{\sigma}}{\partial u} = (\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u})$ and $\frac{\partial \boldsymbol{\sigma}}{\partial v} = (\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v})$ with

$$\begin{aligned}&\frac{\partial \boldsymbol{\sigma}}{\partial u}(u,v) = \\ &M_1 \sum_{k=0}^{M_1-1} \sum_{l=-1}^{M_2+1} \mathbf{c}[k,l] \varphi'_{M_1,\mathrm{per}}(M_1 u - k) \varphi_{2M_2}(M_2 v - l)\end{aligned} \tag{7.46}$$

and

$$\begin{aligned}&\frac{\partial \boldsymbol{\sigma}}{\partial v}(u,v) = \\ &M_2 \sum_{k=0}^{M_1-1} \sum_{l=-1}^{M_2+1} \mathbf{c}[k,l] \varphi_{M_1,\mathrm{per}}(M_1 u - k) \varphi'_{2M_2}(M_2 v - l).\end{aligned} \tag{7.47}$$

7.4.4 Volume Computation

By the divergence theorem, the volume of a parametric surface is computed as

$$\mathrm{Vol}(\boldsymbol{\sigma}) = \oiint_S (x,0,0) \cdot \boldsymbol{n} \mathrm{d}S = \oiint_S x \mathrm{d}y \wedge dz. \tag{7.48}$$

By applying the same simplifications to compute the wedge operator (7.45) as in Appendix 7.4.3 and using

$$x(u,v) = \sum_{k=0}^{M_1-1} \sum_{l=-1}^{M_2+1} \mathbf{c}[k,l] \varphi_{M_1,\mathrm{per}}(M_1 u - k) \varphi_{2M_2}(M_2 v - l),$$

the volume computation simplifies to

$$\begin{aligned}&\mathrm{Vol}(\boldsymbol{\sigma}) \\ &= \int_0^1 \int_0^1 x(u,v) \Big(\frac{\partial y(u,v)}{\partial u} \frac{\partial z(u,v)}{\partial v} - \frac{\partial y(u,v)}{\partial v} \frac{\partial z(u,v)}{\partial u} \Big) \mathrm{d}u \mathrm{d}v.\end{aligned}$$

Because only the basis functions depend on u and v, the integral-dependent terms can be isolated and precomputed to obtain (7.20).

Chapter 8

Conclusion

In this thesis, we have presented a novel unifying and generic framework for data-adaptive shape characterization using splines. Our framework is competitive and applicable to a wide range of practical settings, which includes biomedical imaging, computer graphics, and shape modeling in general. In this last chapter, we summarize the main contributions and results of this thesis and provide an outlook for further directions of research and applications.

8.1 Summary of Results

Novel construction of shape projectors onto vector spaces. We provided a novel generic construction of shape projectors onto functional vector spaces. The continuous-domain framework is applicable to parametric shapes and we presented an exact spline-based implementation which can be readily implemented using vector-matrix multiplications at a low computational cost. Our solution is valid for any spline-based generator whose integer shifts form a Riesz basis. An advantage of our construction is that it does not depend on a specific reference shape living in the vector space; a crucial property to compute unbiased shape alignments.

We show how data sets consisting in curves and surfaces are aligned using our algorithm and how shape priors are constructed for segmentation of biomedical images. Our results demonstrate that our shape priors provide additional robustness in noisy segmentation settings compared to the state of the art and that the flexibility of the chosen transformation of the affine family used to construct the projection operator improves the segmentation results.

New framework for continuous-domain sparse shape encoding and dictionary learning. We presented a new generic theory for dictionary learning and shape encoding of spline shapes. Our construction can be applied to datasets which are imbalanced or which contain outliers. A key element of our method is the derivation of an L_2-ℓ_2 norm equality which allows to express the continuous-domain L_2 norm of a spline-shape using its vector of control points and matrices which contain the correlation integrals of the basis functions. This direct continuous-discrete domain correspondance related to spline shapes allows us to deploy sparse ℓ_1-based methods w.r.t. spline coefficients and thus, to apply sparsity-methods to shapes defined in the continuous-domain. We also presented an L_2-based functional projector-based PCA for spline shapes, which is a special case of our sparse dictionary-learning model. We apply and validate our method to classify shapes in medical imaging, to analyze inhomogeneous datasets of brain structures, and to construct dictionaries of anatomical medical shapes.

New spline interpolators for data-adaptive and topology-specific shape representation. We constructed novel families of spline interpolators for smooth shape modeling and shape characterization. Thereby, we keep ease-of-use and user-interactivity as a main focus. We show how interpolators are constructed to represent shapes of different topology, such as cylindrical, rectangular, spherical or toroidal. We provided specific algorithms that allow one to immediately construct an interpolator if the parameterization of a shape is known. Besides user-interactivity our interpolators are suitable for local shape control and fast optimization due to their compact support. Furthermore, we presented an extension of the family to construct interpolators that allow one to vary the resolution of a shape, thereby enabling to add detail in interactive shape design. All our proposed interpolators form a Riesz basis and we have shown how they are used in practice.

Deformable spline shapes in practice. We have presented several applications, where we used our new framework in practice. We have provided examples of the 3D segmentation of medical structures such as the brain, the aorta, and the spinal cord. We illustrated how our algorithms are implemented, thereby combining semi-automatic optimization algorithms with user-interactivity. Furthermore, we have shown how our model is used in general shape modeling contexts, such as interactive computer-aided design, shape reconstruction from samples and point clouds or shape morphing and deformation. We have demonstrated that our framework is applicable beyond biomedical imaging with potential use in fields related to computer graphics or computer-aided design.

8.2 Outlook

3D extension of continuous-domain sparse dictionary learning. Currently, discrete methods for sparse coding and dictionary learning are well explored for 1D signals and images. In this thesis, we have presented a theory for continous-domain shape encoding and sparse dictionary learning for 2D curves. Our solution can be fully extended to 3D spline surfaces by noting that the inner product between spline surfaces can also be expressed as a matrix vector multiplication. Methods for 3D sparse dictionary learning can offer new possibilities [189], for instance, to analyze 3D data sets of medical structures or for automatic real-time object recognition in practice. Thereby, the use of spline shapes allows to represent them in an exact manner but with few coefficients which enables fast processing times and low computational costs.

Extended segmentation frameworks based on sparse dictionary learning and shape priors. We have shown how our shape priors and shape encoding methods are used in practice to improve robustness and to analyze data sets of shapes. These two models can be combined to construct a segmentation framework which is robust regarding noise but also flexible with respect to segmenting different types of shapes encoded in a dictionary. For example, if dictionaries are constructed, which contain several anatomical structures, then related shape priors can be computed, each representing a population of shape structures. Structure-specific segmentation energies can be computed using iterative algorithms in an online segmentation setting, which immediately associate the structure of interest to a shape in the dictionary. Furthermore, our framework can be combined with tracking algorithms which are related to segmentation with prior knowledge [13, 190].

Software implementation for unifying shape characterization in biomedical imaging. The motivation of this thesis has been to develop a complete theory for the construction of a geometric kernel that builds the basis for a software implementation, which allows biologists and clinicians to analyze, segment, and characterize their data in a single software environment regardless of the shape of their data and its dimension. We have constructed a unifying theory for shape characterization which can be used to process different types of shapes, such as 2D and 3D with different topology, user interaction, optimization and we provide means to construct shape priors and shape dictionaries. Each aspect of our theory has been tested, validated, and implemented as a software prototype. From the perspective of biomedical imaging, the next step should be to provide a full software implementation of the framework and to make it available to clinicians and biologists on popular software platforms such as ImageJ [191] or Icy [7].

Bibliography

[1] A. Blake and M. Isard, *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*, Springer-Verlag New York, Inc., 1st edition, 1998.

[2] E. Meijering, "Cell segmentation: 50 years down the road," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 140–145, September 2012.

[3] G. Myers, "Why bioimage informatics matters," *Nature Methods*, vol. 6, no. 7, pp. 659–660, July 2012.

[4] A. Cardona and P. Tomancak, "Current challenges in open-source bioimage informatics," *Nature Methods*, vol. 9, no. 7, pp. 661–665, July 2012.

[5] D. Schmitter, P. Wachowicz, D. Sage, A. Chasapi, I. Xenarios, V. Simanis, and M. Unser, "A 2D/3D image analysis system to track fluorescently labeled structures in rod-shaped cells: Application to measure spindle pole asymmetry during mitosis," *Cell Division*, vol. 8, no. 6, pp. 1–13, April 2013.

[6] R. Murphy, E. Meijering, and G. Danuser, "Guest editorial," *Image Processing, IEEE Transactions on*, vol. 14, no. 9, pp. 1233–1236, Sept 2005.

[7] F. de Chaumont, S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Le Montagner, T. Lagache, A. Dufour, and J.-C. Olivo-Marin, "Icy: An open bioimage informatics platform for extended reproducible research," *Nature Methods*, vol. 9, no. 7, pp. 690–696, July 2012.

[8] A. Munoz-Barrutia, J. Kovacevic, M. Kozubek, E. Meijering, and B. Parvin, "Quantitative bioimaging: Signal processing in light microscopy [from the guest editors]," *Signal Processing Magazine, IEEE*, vol. 32, no. 1, pp. 18–19, Jan 2015.

[9] C. Ortiz-de Solorzano, A. Munoz-Barrutia, E. Meijering, and M. Kozubek, "Toward a morphodynamic model of the cell: Signal processing for cell modeling," *Signal Processing Magazine, IEEE*, vol. 32, no. 1, pp. 20–29, Jan 2015.

[10] Baek Hwan H. Cho, Ivan Cao-Berg, Jennifer Ann A. Bakal, and Robert F. Murphy, "OMERO.searcher: content-based image search for microscope images.," *Nature methods*, vol. 9, no. 7, pp. 633–634, July 2012.

[11] Satwik Rajaram, Benjamin Pavie, Lani F Wu, and Steven J Altschuler, "Phenoripper: software for rapidly profiling microscopy images," *Nature methods*, vol. 9, no. 7, pp. 635637, July 2012.

[12] D. Schmitter, A. Roche, B. Maréchal, D. Ribes, A. Abdulkadir, M. Bach-Cuadra, A. Daducci, C. Granziera, S. Klöppel, P. Maeder, R. Meuli, and G. Krueger, "An evaluation of volume-based morphometry for prediction of mild cognitive impairment and Alzheimer's disease," *NeuroImage: Clinical*, vol. 7, pp. 7–17, 2015.

[13] A.C. Dufour, L. Tzu-Yu, C. Ducroz, R. Tournemenne, B. Cummings, R. Thibeaux, N. Guillen, A.O. Hero, and J.-C. Olivo-Marin, "Signal processing challenges in quantitative 3-d cell morphology: More than meets the eye," *Signal Processing Magazine, IEEE*, vol. 32, no. 1, pp. 30–40, Jan 2015.

[14] H. Peng, "Bioimage informatics: a new area of engineering biology," *Bioinformatics*, vol. 24, no. 17, pp. 1827–1836, July 2008.

[15] H. Peng, A. Bateman, A. Valencia, and J.D. Wren, "Bioimage informatics: a new category in *Bioinformatics*," *Bioinformatics*, vol. 28, no. 8, pp. 1057, 2012.

[16] E. Catmull and R. Rom, "A class of local interpolating splines," *Computer aided geometric design. Academic Press.*, pp. 317–326, 1974.

[17] L. Piegl, "On nurbs: A survey," *IEEE Comput. Graph. Appl.*, vol. 11, no. 1, pp. 55–71, Jan. 1991.

[18] E. Catmull and J. Clark, "Seminal graphics," chapter Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes, pp. 183–188. ACM, New York, NY, USA, 1998.

[19] G. E. Farin, *NURBS: From Projective Geometry to Practical Use*, A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 1999.

[20] P. Brigger, J. Hoeg, and M. Unser, "B-Spline snakes: A flexible tool for parametric contour detection," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1484–1496, September 2000.

[21] M.A.T. Figueiredo, J.M.N. Leitão, and A.K. Jain, "Unsupervised contour representation and estimation using B-splines and a minimum description length criterion," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 1075–1087, June 2000.

[22] F. Precioso and M. Barlaud, "B-spline active contours for fast video segmentation," in *Proceedings of the 2001 International Conference on Image Processing*, Thessaloniki, Greece, October, 7-10, 2001, pp. 777–780.

[23] F. Precioso and M. Barlaud, "B-spline active contour with handling of topology changes for fast video segmentation.," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 6, pp. 555–560, January 2002.

[24] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-Spline Techniques*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[25] L. Piegl and W. Tiller, *The NURBS Book*, Springer Berlin Heidelberg, second edition, 2010.

[26] D. Barbosa, T. Dietenbeck, J. Schaerer, J. D'hooge, D. Friboulet, and O. Bernard, "B-Spline explicit active surfaces: An efficient framework for real-time 3-D region-based segmentation," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 241–251, January 2012.

[27] D. Schmitter, P. García-Amorena, and M. Unser, "Smoothly deformable spheres: Modeling, deformation, and interaction," in *Proceedings of the 2016 ACM Special Interest Group on Computer Graphics and Interactive Techniques Conference Asia: Technical Briefs (SIGGRAPH-TB'16)*, Macau, Macao Special Administrative Region of the People's Republic of China, December 5-8, 2016, paper no. 2.

[28] D. Schmitter, P. García-Amorena, and M. Unser, "Smooth shapes with spherical topology: Beyond traditional modeling, efficient deformation, and interaction," *Computational Visual Media*, vol. 3, no. 3, pp. 199–215, September 2017.

[29] M. Lounsbery, T. DeRose, and J. Warren, "Multiresolution analysis for surfaces of arbitrary topological type," *ACM Trans. Graph.*, vol. 16, no. 1, pp. 34–73, Jan. 1997.

[30] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1998, SIGGRAPH '98, pp. 85–94, ACM.

[31] Joe Warren and Henrik Weimer, *Subdivision Methods for Geometric Design: A Constructive Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2001.

[32] N. Dyn and E. Farkhi, "Spline subdivision schemes for compact sets. A survey," *Serdica Mathematical Journal*, vol. 28, pp. 349–360, 2002.

[33] Jos Stam and Charles T. Loop, "Quad/triangle subdivision.," *Comput. Graph. Forum*, vol. 22, no. 1, pp. 79–86, 2003.

[34] Charles Loop and Scott Schaefer, "Approximating catmull-clark subdivision surfaces with bicubic patches," *ACM Trans. Graph.*, vol. 27, no. 1, pp. 8:1–8:11, Mar. 2008.

[35] K. Conti, C. Hormann, "Polynomial reproduction for univariate subdivision schemes of any arity," *Journal of Approximation Theory*, vol. 163, no. 4, pp. 413 – 437, 2011.

[36] "Interpolatory blending net subdivision schemes of dubucdeslauriers type," *Computer Aided Geometric Design*, vol. 29.

[37] M. Charina and C. Conti, "Polynomial reproduction of multivariate scalar subdivision schemes," *Journal of Computational and Applied Mathematics*, vol. 240, no. 0, pp. 51 – 61, 2013.

[38] M. Charina, C. Conti, and L. Romani, "Reproduction of exponential polynomials by multivariate non-stationary subdivision schemes with a general dilation matrix," *Numerische Mathematik*, vol. 127, no. 2, pp. 223–254, 2014.

[39] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel, "Interactive multi-resolution modeling on arbitrary meshes," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1998, SIGGRAPH '98, pp. 105–114, ACM.

[40] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "Openmesh: A generic and efficient polygon mesh data structure," 2002, OpenSG Symposium 2002.

[41] Olga Sorkine, "Differential representations for mesh processing," *Computer Graphics Forum*, vol. 25, no. 4, pp. 789–807, 2006.

[42] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*, AK Peters, 2010.

[43] A. Dufour, R. Thibeaux, E. Labruyere, N. Guillen, and J.-C. Olivo-Marin, "3-D Active meshes: Fast discrete deformable models for cell tracking in 3-D time-lapse microscopy," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1925–1937, July 2011.

[44] R. Malladi, J.A. Sethian, and B.C. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, 1995.

[45] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[46] M.E. Leventon, W.E.L. Grimson, and O. Faugeras, "Statistical shape influence in geodesic active contours," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 2000, vol. 1, pp. 316–323 vol.1.

[47] T.F. Chan and L.A. Vese, “Active contours without edges,” *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, February 2001.

[48] X. Han, C. Xu, and J.L. Prince, “A topology preserving level set method for geometric deformable models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 755–768, June 2003.

[49] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher, “Fast global minimization of the active contour/snake model,” *Journal of Mathematical Imaging and Vision*, vol. 28, no. 2, pp. 151–167, 2007.

[50] Martin Vetterli, Jelena Kovačević, and Vivek K Goyal, *Foundations of signal processing*, Cambridge University Press, 2014.

[51] T. Lyche and L.L. Schumaker, “Local spline approximation methods,” *Journal of Approximation Theory*, vol. 15, no. 4, pp. 294 – 325, 1975.

[52] T. Lyche, “Discrete cubic spline interpolation,” *BIT Numerical Mathematics*, vol. 16, no. 3, pp. 281–290, 1976.

[53] L. L. Schumaker, *Spline functions : basic theory*, Pure and applied mathematics : a Wiley-Interscience series of texts, monographs, and tracts. J. Wiley & Sons, 1981.

[54] L. Romani and M.A. Sabin, “The conversion matrix between uniform B-spline and Bézier representations,” *Computer Aided Geometric Design*, vol. 21, no. 6, pp. 549 – 560, 2004.

[55] C. V. Beccari, G. Casciola, and L. Romani, “Construction and characterization of non-uniform local interpolating polynomial splines.,” *J. Computational Applied Mathematics*, vol. 240, pp. 5–19, 2013.

[56] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri, “T-splines and t-nurccs,” in *ACM SIGGRAPH 2003 Papers*, New York, NY, USA, 2003, SIGGRAPH ’03, pp. 477–484, ACM.

[57] Thomas W. Sederberg, David L. Cardon, G. Thomas Finnigan, Nicholas S. North, Jianmin Zheng, and Tom Lyche, “T-spline simplification and local refinement,” in *ACM SIGGRAPH 2004 Papers*, New York, NY, USA, 2004, SIGGRAPH ’04, pp. 276–283, ACM.

[58] Gilles Deslauriers and Serge Dubuc, "Symmetric iterative interpolation processes," *Constructive Approximation*, vol. 5, no. 1, pp. 49–68, 1989.

[59] Markus H. Gross, Roger Gatti, and Oliver Staadt, "Fast multiresolution surface meshing," in *Proceedings of the 6th Conference on Visualization '95*, Washington, DC, USA, 1995, VIS '95, pp. 135–, IEEE Computer Society.

[60] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, November 1999, IEEE Signal Processing Society's 2000 magazine award.

[61] T. Blu, P. Thévenaz, and M. Unser, "Complete parameterization of piecewise-polynomial interpolation kernels," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1297–1309, November 2003.

[62] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, New York, NY, USA, 2004, SGP '04, pp. 175–184, ACM.

[63] M. Unser and T. Blu, "Cardinal exponential splines: Part I—Theory and filtering algorithms," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1425–1438, April 2005.

[64] S. Derrode, M. A. Charmi, and F. Ghorbel, "Fourier-based invariant shape prior for snakes," in *in Proc. of the IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, 2006, pp. 14–19.

[65] Mark Pauly, Leif P. Kobbelt, and Markus Gross, "Point-based multiscale surface representation," *ACM Trans. Graph.*, vol. 25, no. 2, pp. 177–193, Apr. 2006.

[66] D. Schmitter and M. Unser, "Shape projectors for landmark-based spline curves," *accepted in IEEE Signal Processing Letters.*

[67] D. Schmitter and M. Unser, "Similarity-based shape priors for 2D spline snakes," in *Proceedings of the Twelfth IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'15)*, Brooklyn NY, USA, April 16-19, 2015, pp. 1216–1219.

[68] R. Delgado-Gonzalo, D. Schmitter, V. Uhlmann, and M. Unser, "Efficient shape priors for spline-based snakes," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3915–3926, November 2015.

[69] D. G. Kendall, "Shape manifolds, procrustean metrics, and complex projective spaces," *Bulletin of the London Mathematical Society*, vol. 16, no. 2, pp. 81–121, 1984.

[70] Colin Goodall, "Procrustes methods in the statistical analysis of shape (disc: p321-339)," *Journal of the Royal Statistical Society, Series B: Methodological*, vol. 53, pp. 285–321, 1991.

[71] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[72] I.L. Dryden and K.V. Mardia, *Statistical Shape Analysis*, John Wiley & Sons, 1998.

[73] T.F. Cootes, G.J. Edwards, and C.J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, June 2001.

[74] D. Schmitter and M. Unser, "Landmark-based shape encoding and sparse-dictionary learning in the continuous domain," *submitted.*

[75] D. Schmitter and M. Unser, "Closed-form alignment of active surface models using splines," in *Proceedings of the Fourteenth IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'17)*, Melbourne, Commonwealth of Australia, April 18-21, 2017, pp. 219–222.

[76] D. Schmitter, R. Delgado-Gonzalo, and M. Unser, "Trigonometric interpolation kernel to construct deformable shapes for user-interactive applications," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 2097–2101, November 2015.

[77] D. Schmitter, R. Delgado-Gonzalo, and M. Unser, "A family of smooth and interpolatory basis functions for parametric curve and surface representation," *Applied Mathematics and Computation*, vol. 272, no. 1, pp. 53–63, January 1, 2016.

[78] D. Schmitter, J. Fageo, A. Badoual, P. García-Amorena, and M. Unser, "Compactly-supported smooth interpolators for shape modelling with varying resolution," *submitted.*

[79] D. Schmitter, R. Delgado-Gonzalo, G. Krueger, and M. Unser, "Atlas-free brain segmentation in 3D proton-density-like MRI images," in *Proceedings of the Eleventh IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'14)*, Beijing, People's Republic of China, April 29-May 2, 2014, pp. 629–632.

[80] D. Schmitter, C. Gaudet-Blavignac, D. Piccini, and M. Unser, "New parametric 3D snake for medical segmentation of structures with cylindrical topology," in *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP'15)*, Québec QC, Canada, September 27-30, 2015.

[81] M. Unser, "Sampling—50 Years after Shannon," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 569–587, April 2000.

[82] A. Badoual, D. Schmitter, and M. Unser, "An inner-product calculus for periodic functions and curves," *IEEE Signal Processing Letters*, vol. in press, 2016.

[83] R. Delgado-Gonzalo, V. Uhlmann, D. Schmitter, and M. Unser, "Snakes on a plane: A perfect snap for bioimage analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 1, pp. 41–48, January 2015.

[84] Ricard Delgado-Gonzalo, Philippe Thevenaz, Chandra Sekhar Seelamantula, and Michael Unser, "Snakes with an ellipse-reproducing property," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 1258–1271, 2012.

[85] R. Delgado-Gonzalo, P. Thévenaz, and M. Unser, "Exponential splines and minimal-support bases for curve representation," *Computer Aided Geometric Design*, vol. 29, no. 2, pp. 109–128, February 2012.

[86] R. Delgado-Gonzalo, N. Chenouard, and M. Unser, "Spline-based deforming ellipsoids for interactive 3d bioimage segmentation," *IEEE Transactions on Image Processing*, vol. 22, no. 10, pp. 3926–3940, October 2013.

[87] R.D. Gonzalo, *Segmentation and Tracking in High-Throughput Bioimaging*, EPFL thesis no. 5657 (2013), 186 p., Swiss Federal Institute of Technology Lausanne (EPFL), March 15, 2013, 2013 research award of the Swiss Society for Biomedical Engineering.

[88] C. de Boor and R. DeVore, "Partitions of unity and approximation," *Proceedings of the American Mathematical Society*, vol. 93, no. 4, pp. 705–709, April 1985.

[89] M. Jacob, T. Blu, and M. Unser, "Sampling of periodic signals: A quantitative error analysis," *IEEE Transactions on Signal Processing*, vol. 50, no. 5, pp. 1153–1159, May 2002.

[90] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, January 1987.

[91] B. Li and S. T. Acton, "Active contour external force using vector field convolution for image segmentation," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2096–2106, Aug 2007.

[92] M. Gastaud, M. Barlaud, and G. Aubert, "Combining shape prior and statistical features for active contour segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 5, pp. 726–734, May 2004.

[93] Y. Chen, H.D. Tagare, S. Thiruvenkadam, F. Huang, D. Wilson, K.S. Gopinath, R.W. Briggs, and E.A. Geiser, "Using prior shapes in geometric active contours in a variational framework," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 315–328, December 2002.

[94] A. Foulonneau, P. Charbonnier, and F. Heitz, "Affine-invariant geometric shape priors for region-based active contours," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 8, pp. 1352–1357, Aug 2006.

[95] M.-A. Charmi, S. Derrode, and F. Ghorbel, "Fourier-based geometric shape prior for snakes," *Pattern Recognition Letters*, vol. 29, no. 7, pp. 897 – 904, 2008.

[96] G. Charpiat, P. Maurel, J.-P. Pons, R. Keriven, and O. Faugeras, "Generalized gradients: Priors on minimization flows," *International Journal of Computer Vision*, vol. 73, no. 3, pp. 325–344, July 2007.

[97] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.

[98] Hui Zou, Trevor Hastie, and Robert Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, pp. 2006, 2004.

[99] Ron Zass and Amnon Shashua, "Nonnegative sparse pca," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 2006, NIPS'06, pp. 1561–1568, MIT Press.

[100] Daniela M. Witten, Robert Tibshirani, and Trevor Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 7 2009.

[101] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Trans. Img. Proc.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[102] Shaoting Zhang, Yiqiang Zhan, and Dimitris N Metaxas, "Deformable segmentation via sparse representation and dictionary learning," *Medical Image Analysis*, vol. 16, no. 7, pp. 1385–1396, 2012.

[103] Shaoting Zhang, Yiqiang Zhan, Yan Zhou, Mustafa Uzunbas, and Dimitris N Metaxas, "Shape prior modeling using sparse representation and online dictionary learning," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2012, pp. 435–442.

[104] F.James Rohlf, "Bias and error in estimates of mean shape in geometric morphometrics," *Journal of Human Evolution*, vol. 44, no. 6, pp. 665 – 683, 2003.

[105] D. Rueckert, A. F. Frangi, and J. A. Schnabel, "Automatic construction of 3-d statistical deformation models of the brain using nonrigid registration," *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 1014–1025, Aug 2003.

[106] Martin Styner and Guido Gerig, *Medial Models Incorporating Object Variability for 3D Shape Analysis*, pp. 502–516, Springer Berlin Heidelberg, 2001.

[107] A. Kelemen, G. Székely, and G. Gerig, "Elastic Model-Based Segmentation of 3D Neuroradiological Data Sets," *IEEE Trans. Med. Imaging*, vol. 18, pp. 828–839, Oct. 1999.

[108] Daniel Cremers, Florian Tischhäuser, Joachim Weickert, and Christoph Schnörr, "Diffusion snakes: Introducing statistical shape knowledge into the mumford-shah functional," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 295–313, 2002.

[109] M. Unser, B.L. Trus, and A.C. Steven, "Normalization procedures and factorial representations for classification of correlation-aligned images: A comparative study," *Ultramicroscopy*, vol. 30, no. 3, pp. 299–310, July-August 1989.

[110] C. Vonesch, F. Stauber, and M. Unser, "Steerable PCA for rotation-invariant image recognition," *SIAM Journal on Imaging Sciences*, vol. 8, no. 3, pp. 1857–1873, 2015.

[111] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng, "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems 19*, P. B. Schölkopf, J. C. Platt, and T. Hoffman, Eds., pp. 801–808. MIT Press, 2007.

[112] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA, 2009, ICML '09, pp. 689–696, ACM.

[113] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb 2006.

[114] Robert Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[115] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, Jan. 2001.

[116] Kjersti Engan, Sven Ole Aase, and John Hakon Husoy, “Frame based signal compression using method of optimal directions (mod).,” in *ISCAS (4)*. 1999, pp. 1–4, IEEE.

[117] M. Aharon, M. Elad, and A. Bruckstein, “Svdd: An algorithm for designing overcomplete dictionaries for sparse representation,” *Trans. Sig. Proc.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[118] Léon Bottou, “On-line learning in neural networks,” chapter On-line Learning and Stochastic Approximations, pp. 9–42. Cambridge University Press, 1998.

[119] Olivier Bousquet and Léon Bottou, “The tradeoffs of large scale learning,” in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., pp. 161–168. Curran Associates, Inc., 2008.

[120] D. Baud, Sy. Meyer, Y. Vial, P. Hohlfeld, and C. Achtari, “Pelvic floor dysfunction 6 years post-anal sphincter tear at the time of vaginal delivery,” *International Urogynecology Journal*, vol. 22, no. 9, pp. 1127–1134, 2011.

[121] G. A. van Veelen, K. J. Schweitzer, and C. H. van der Vaart, “Ultrasound imaging of the pelvic floor: changes in anatomy during and after first pregnancy,” *Ultrasound in Obstetrics and Gynecology*, vol. 44, no. 4, pp. 476–480, 2014.

[122] C. J. Twining and C. J. Taylor, “Kernel principal component analysis and the construction of non-linear active shape models,” in *In Proceedings British Machine Vision Conference (BMVC*, 2001.

[123] Tom Huysmans, Bart Haex, Tom De Wilde, Remy Van Audekercke, Jos Vander Sloten, and Georges Van der Perre, “A 3d active shape model for the evaluation of the alignment of the spine during sleeping,” *Gait & posture*, vol. 24, no. 1, pp. 54–61, 2006.

[124] Andreas Wimmer, Grzegorz Soza, and Joachim Hornegger, *A Generic Probabilistic Active Shape Model for Organ Segmentation*, pp. 26–33, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[125] S. Dambreville, Y. Rathi, and A. Tannenbaum, "A framework for image segmentation using shape models and kernel space shape priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1385–1399, Aug 2008.

[126] Matthias Kirschner, Meike Becker, and Stefan Wesarg, *3D Active Shape Model Segmentation with Nonlinear Shape Priors*, pp. 492–499, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[127] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, Dec 1981.

[128] M.A. Audette, A. Chernikov, and N. Chrisochoides, "A review of mesh generation for medical simulators," *Handbook of Real-World Applications in Modeling and Simulation*, vol. 2, pp. 261, 2012.

[129] A. Garg, A.O. Sageman-Furnas, B. Deng, Y. Yue, E. Grinspun, M. Pauly, and M. Wardetzky, "Wire mesh design," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 66:1–66:12, 2014.

[130] B. Deng, S. Bouaziz, M. Deuss, A. Kaspar, Y. Schwartzburg, and M. Pauly, "Interactive design exploration for constrained meshes," *Computer-Aided Design*, vol. 61, pp. 13–23, 2015.

[131] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91 – 108, 1996.

[132] T. Heimann and H.-P. Meinzer, "Statistical shape models for 3d medical image segmentation: A review," *Medical Image Analysis*, vol. 13, no. 4, pp. 543–563, 2009.

[133] P. Novara and L. Romani, "Building blocks for designing arbitrarily smooth subdivision schemes with conic precision," *Journal of Computational and Applied Mathematics*, vol. 279, no. 0, pp. 67 – 79, 2015.

[134] C. Brechbuehler, G. Gerig, and O. Kuebler, “Parametrization of closed surfaces for 3-d shape description,” *Computer Vision and Image Understanding*, vol. 61, no. 2, pp. 154 – 170, 1995.

[135] C. Manni, F. Pelosi, and L.M. Sampoli, “Generalized B-splines as a tool in isogeometric analysis,” *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 58, pp. 867 – 881, 2011.

[136] T. Dokken, T. Lyche, and K.F. Pettersen, “Polynomial splines over locally refined box-partitions,” *Computer Aided Geometric Design*, vol. 30, no. 3, pp. 331 – 356, 2013.

[137] G. Gerig, M. Styner, and G. Szekely, “Statistical shape models for segmentation and structural analysis,” in *Proceedings of the 2002 IEEE International Symposium on Biomedical Imaging.*, 2002, pp. 18–21.

[138] M. Jacob, T. Blu, and M. Unser, “Efficient energies and algorithms for parametric snakes,” *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1231–1244, September 2004.

[139] T. Blu and M. Unser, “Quantitative Fourier analysis of approximation techniques: Part I—Interpolators and projectors,” *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2783–2795, October 1999.

[140] M. Unser and P.D. Tafti, *An Introduction to Sparse Stochastic Processes*, Cambridge University Press, Cambridge, UK, 2014.

[141] E. Cohen, T. Lyche, and R. Riesenfeld, “Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics,” *Computer Graphics and Image Processing*, vol. 14, no. 2, pp. 87 – 111, 1980.

[142] M. Antonelli, C. Beccari, and G. Casciola, “A general framework for the construction of piecewise-polynomial local interpolants of minimum degree,” *Advances in Computational Mathematics*, vol. 40, no. 4, pp. 945–976, 2014.

[143] C. Conti, L. Romani, and M. Unser, “Ellipse-preserving Hermite interpolation and subdivision,” *Journal of Mathematical Modelling and Algorithms in Operations Research*, vol. in press.

[144] N. Dyn, L. David, and A. Luzzatto, "Exponentials reproducing subdivision schemes," *Foundations of Computational Mathematics*, vol. 3, no. 2, pp. 187–206, 2003.

[145] E. Meijering and M. Unser, "A note on cubic convolution interpolation," *IEEE Transactions on Image Processing*, vol. 12, no. 4, pp. 477–479, April 2003.

[146] M. Unser, "Cardinal exponential splines: Part II—Think analog, act digital," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1439–1449, April 2005.

[147] G. Strang and G. Fix, "A Fourier analysis of the finite element variational method," *Constructive Aspects of Functional Analysis*, p. 793840, 1973.

[148] N. Dyn, D. Levin, and J. A. Gregory, "A 4-point interpolatory subdivision scheme for curve design," *Computer Aided Geometric Design*, vol. 4, no. 4, pp. 257–268, 1987.

[149] Nira Dyn, David Levin, and Jungho Yoon, "Analysis of univariate nonstationary subdivision schemes with application to Gaussian-based interpolatory schemes," *SIAM Journal on Mathematical Analysis*, vol. 39, no. 2, pp. 470–488, 2007.

[150] A. Badoual, D. Schmitter, V. Uhlmann, and M. Unser, "Multiresolution subdivision snakes," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1188–1201, March 2017.

[151] David F. Rogers, *An Introduction to NURBS: With Historical Perspective*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[152] A. Badoual, D. Schmitter, and M. Unser, "Locally refinable parametric snakes," in *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP'15)*, Québec QC, Canada, September 27-30, 2015, paper no. TEC-P21.2.

[153] A. Badoual, D. Schmitter, and M. Unser, "Local refinement for 3d deformable parametric surfaces," in *Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP'16)*, Phoenix AZ, USA, September 25-28, 2016, pp. 1086–1090.

[154] C. Vonesch, T. Blu, and M. Unser, "Generalized daubechies wavelet families," *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4415–4429, September 2007.

[155] P. Dierckx, "Algorithms for smoothing data on the sphere with tensor product splines," *Computing*, vol. 32, no. 4, pp. 319–342, April 1984.

[156] R.H.J. Gmelig Meyling and P.R. Pfluger, "B-spline approximation of a closed surface," *IMA Journal of Numerical Analysis*, vol. 7, no. 1, pp. 73–96, January 1987.

[157] D. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points," *Computer-Aided Design*, vol. 10, no. 6, pp. 356 – 360, 1978.

[158] T. Hughes, *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation, 2012.

[159] Emil Praun and Hugues Hoppe, "Spherical parametrization and remeshing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 340–349, July 2003.

[160] Arul Asirvatham, Emil Praun, and Hugues Hoppe, *Computational Science – ICCS 2005: 5th International Conference, Atlanta, GA, USA, May 22-25, 2005. Proceedings, Part II*, chapter Consistent Spherical Parameterization, pp. 265–272, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[161] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, Aire-la-Ville, Switzerland, Switzerland, 2006, SGP '06, pp. 61–70, Eurographics Association.

[162] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva, "State of the Art in Surface Reconstruction from Point Clouds," in *Eurographics 2014 - State of the Art Reports*, Sylvain Lefebvre and Michela Spagnuolo, Eds. 2014, The Eurographics Association.

[163] Ren-Jiang Zhang, "Uniform interpolation curves and surfaces based on a family of symmetric splines," *Comput. Aided Geom. Des.*, vol. 30, no. 9, pp. 844–860, Dec. 2013.

[164] Carolina Vittoria Beccari, Giulio Casciola, and Lucia Romani, “Non-uniform interpolatory curve subdivision with edge parameters built upon compactly supported fundamental splines,” *BIT Numerical Mathematics*, vol. 51, no. 4, pp. 781–808, 2011.

[165] C. Conti, L. Gemignani, and L. Romani, “Exponential pseudo-splines: Looking beyond exponential b-splines,” *Journal of Mathematical Analysis and Applications*, vol. 439, no. 1, pp. 32–56, 2016.

[166] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer, “Elastically deformable models,” *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 205–214, Aug. 1987.

[167] D. Terzopoulos and D. N. Metaxas, “Dynamic 3d models with local and global deformations: deformable superquadrics.,” in *ICCV*. 1990, pp. 606–615, IEEE.

[168] Mario Botsch and Olga Sorkine, “On linear variational surface deformation methods,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, Jan. 2008.

[169] Rhodri H. Davies, Tim F. Cootes, and Chris J. Taylor, *Information Processing in Medical Imaging: 17th International Conference, IPMI 2001 Davis, CA, USA, June 18–22, 2001 Proceedings*, chapter A Minimum Description Length Approach to Statistical Shape Modelling, pp. 50–63, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[170] Martin Styner, Kumar T. Rajamani, Lutz-Peter Nolte, Gabriel Zsemlye, Gbor Szkely, Christopher J. Taylor, and Rhodri H. Davies, “Evaluation of 3d correspondence methods for model building.,” in *IPMI*, Christopher J. Taylor and J. Alison Noble, Eds. 2003, vol. 2732 of *Lecture Notes in Computer Science*, pp. 63–75, Springer.

[171] F. Aguet, M. Jacob, and M. Unser, “Three-dimensional feature detection using optimal steerable filters,” in *Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP’05)*, Genova, Italian Republic, September 11-14, 2005, vol. II, pp. 1158–1161.

[172] M. Jacob, T. Blu, and M. Unser, “An exact method for computing the area moments of wavelet and spline curves,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 633–642, June 2001.

[173] J. Ashburner, “A fast diffeomorphic image registration algorithm,” *NeuroImage*, vol. 38, no. 1, pp. 95–113, October 2007.

[174] A. M. Dale, B. Fischl, and M. I. Sereno, “Cortical surface-based analysis: I. Segmentation and surface reconstruction,” *NeuroImage*, vol. 9, no. 2, pp. 179 – 194, February 1999.

[175] B. Moeller and S. Posch, “An integrated analysis concept for errors in image registration,” *Pattern Recognition and Image Analysis*, vol. 18, no. 2, pp. 201–206, June 2008.

[176] C. Zimmer, E. Labruyere, V. Meas-Yedid, N. Guillen, and J.-C. Olivo-Marin, “Segmentation and tracking of migrating cells in videomicroscopy with parametric active contours: A tool for cell-based drug testing,” *IEEE Trans. Medical Imaging*, vol. 21, no. 10, pp. 1212–1221, October 2002.

[177] Y.-L. Fok, J. C. K. Chan, and R.T. Chin, “Automated analysis of nerve-cell images using active contour models,” *IEEE Trans. Medical Imaging*, vol. 15, no. 3, pp. 353–368, June 1996.

[178] W. T. Freeman and E. H. Adelson, “The design and use of steerable filters,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, September 1991.

[179] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, November 1986.

[180] M. Jacob and M. Unser, “Design of steerable filters for feature detection using Canny-like criteria,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1007–1019, August 2004.

[181] R.K.-S. Kwan, A.C. Evans, and G.B. Pike, “MRI simulation-based evaluation of image-processing and classification methods,” *IEEE Trans. Medical Imaging*, vol. 18, no. 11, pp. 1085–1097, November 1999.

[182] M. Feuerstein, T. Kitasaka, and K. Mori, “Adaptive model based pulmonary artery segmentation in 3d chest ct,” 2010, vol. 7623, pp. 76234S–76234S–9.

[183] T. Kitasaka, J. Hasegawa, K. Mori, and J. Toriwaki, “A method for automated extraction of aorta and pulmonary artery in the mediastinum using medial line models from 3d chest x-ray ct images without contrast materials,” in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2002, vol. 3, pp. 273–276 vol.3.

[184] S. Kurugol, R. San Jose Estepar, J. Ross, and G.R. Washko, “Aorta segmentation with a 3d level set approach and quantification of aortic calcifications in non-contrast chest ct,” in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, Aug 2012, pp. 2343–2346.

[185] I. Isgum, A. Rutten, M. Prokop, M. Staring, S. Klein, J.P.W. Pluim, M.A. Viergever, and B. van Ginneken, “Automated aortic calcium scoring on low-dose chest computed tomography,” *Medical Physics*, vol. 37, no. 2, pp. 714–723, February 2010.

[186] D. Piccini, P. Monney, C. Sierro, S. Coppo, G. Bonanno, R.B. van Heeswijk, J. Chaptinel, G. Vincenti, J. de Blois, Simon C. Koestner, T. Rutz, A. Littmann, M.O. Zenge, J. Schwitter, and M. Stuber, “Respiratory self-navigated postcontrast whole-heart coronary mr angiography: Initial experience in patients,” *Radiology*, vol. 270, no. 2, pp. 378–386, 2014.

[187] G.H.W. van Bogerijen, F. Auricchio, M. Conti, A. Lefieux, A. Reali, A. Veneziani, J.L. Tolenaar, F.L. Moll, V. Rampoldi, and S. Trimarchi, “Aortic hemodynamics after thoracic endovascular aortic repair, with particular attention to the bird-beak configuration,” *Journal of Endovascular Therapy*, vol. 21, no. 6, pp. 791–802, January 2014.

[188] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, third edition, 1986.

[189] Y. Lin, M. Song, D. T. P. Quynh, Y. He, and C. Chen, “Sparse coding for flexible, robust 3d facial-expression synthesis,” *IEEE Computer Graphics and Applications*, vol. 32, no. 2, pp. 76–88, March 2012.

[190] Alexandre Dufour, Vasily Shinin, Shahragim Tajbakhsh, Nancy Guillén-Aghion, J-C Olivo-Marin, and Christophe Zimmer, “Segmenting and tracking fluorescent cells in dynamic 3-d microscopy with coupled active surfaces,” *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1396–1410, 2005.

[191] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri, “Nih image to imagej: 25 years of image analysis,” *Nature methods*, vol. 9, no. 7, pp. 671–675, 2012.

Curriculum Vitæ

Daniel Schmitter, PhD

Contact Information

Mirrakoi SA
EPFL Innovation Park
Building C
CH-1015 Lausanne
Switzerland

Office: +41 21 693 1136

E-mail:
daniel.schmitter@mirrakoi.com
daniel.schmitter@epfl.ch

Personal

born 9 June 1986, Swiss & Canadian nationality

Work Experience

- since 2018: **Founder & CEO, Mirrakoi SA**
- 2015 - 2017: **Head EPFL Start-up Project Mirrakoi**
  Biomedical Imaging Group, EPFL
- 2013 - 2017: **Doctoral Assistant**
  Biomedical Imaging Group, EPFL
- 2011 - 2013: **Software Engineer**
  Biomedical Imaging Group, EPFL
- 2012 - 2013: **R&D Engineer CAD/CAE**
  Siemens Healthcare, Advanced Clinical Imaging Technology Group
- 2010 - 2012: **Software Engineer**
  Laboratory of Stem Cell Bioengineering, EPFL
- 2010: **Software Engineer**
  CIBM EPFL.

Education

- 2017: PhD in mathematical image processing and computer-aided design
- 2013: MSc bioengineering, minor in biomedical technologies, EPFL
- 2011: BSc in life sciences and technology, EPFL

Patents

- 3 U.S. patents filed (main inventor)
- 1 PCT application filed (main inventor)

Grants & Awards

- 2017
  - 130'000 CHF: SNF-Bridge Proof-of-Concept grant awarded for EPFL's start-up project **Mirrakoi**
  - 100'000 CHF: FIT/Innovaud Innogrant awarded for EPFL's start-up project **Mirrakoi**. Obtained highest score among more than 900 EPFL start-up projects since the creation of the Innogrant in 2005.
  - 30'000 CHF: Venturekick Stage I & II grant awarded for start-up project **Mirrakoi**. (Stage III decision pending.)
  - 6'000 CHF EPFL Enable intern grant awarded for start-up project **Mirrakoi**.
  - "Top 25"-ranked in the 2017 Swiss "Venture Business Plan Competition" for the start-up project **Mirrakoi**.
- 2016
  - 60'000 CHF EPFL Enable grant awarded for start-up project **Mirrakoi**.
- 2015
  - Best student paper award (1st rank) at the IEEE International Conference on Image Processing 2015 (ICIP'15), Quebec, Canada, September 2015.

Publications

- **25 scientific peer-reviewed publications** (**13 as first author**)
  Available at: http://bigwww.epfl.ch/publications/
  Google scholar profile: "Daniel Schmitter"

University Teaching

- 2016 - 2017: **Head of teaching assistants of 2 EPFL master courses**
  Head of teaching assistants of 2 EPFL master level courses, 250 students, 2 professors, 12 teaching assistants.
- 2010 - 2017: **EPFL Teaching Assistant**
  Image Processing I & II (master level courses)
  Analysis I & II for engineers (bachelor level courses)
- 2014-2017: **EPFL bachelor & master project supervision**
  11 students, 3 full-time interns