3D Shape Analysis based on Feature Curve Networks

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
einer Doktorin der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Anne Gehre, M. Sc. RWTH

aus Neuss, Deutschland

Berichter: Universitätsprofessor Dr. rer. nat. Leif Kobbelt
Professor Mirela Ben-Chen, PhD

Tag der mündlichen Prüfung: 12. Februar 2019

Abstract

For high-level analysis of 3D shapes, we require an abstract representation of geometric data. Typically, this is achieved by developing descriptors on a local pointwise level or globally on the entire shape. Point based descriptors can be very sensitive to local changes of the shape (e.g. noise). In contrast, global descriptors tend to be too coarse, depending on the analysis task. Hence, a desirable representation encodes different levels of abstraction, which range from a geometric point based description over intermediate topological information to high-level structure and is flexible enough to be applied in various shape analysis tasks. In this thesis we show that networks consisting of feature curves are able to capture this information at various levels, and are therefore a well-suited representation for the challenging task of analysis of 3D shapes.

Feature curves trace out salient creases and crests of 3D geometric data. Their computation has been studied intensively in the past and various directions such as patch layouts, slippage-based techniques, or crest line tracing have been established. They provide an abstract representation of salient parts of the geometry and contain topological and global structural information about the shape as well as geometric details. E.g., the geometric information can be exploited in low-level geometry processing tasks such as remeshing or smoothing, where geometric entities should align with surface features to obtain high quality output. In contrast, the more high-level structural information contained in the feature curve network provides important cues, which can be beneficial for the analysis of the geometric data.

However, automatically computed *feature curve networks* on raw data can have various defects such as noise, fragmentation, or missing data. To make the feature curves applicable to downstream applications, we require a set of meaningful feature curves with low fragmentation and without misclassified feature curves (e.g. due to noise). First attempts to obtain a more relevant set of curves from the automatically computed feature curve networks resort to local solutions such as smoothing, filtering, or (curvature based) thresholding of the curves. This however, also removes small-scale or weak feature curves from the network which might describe important details. In this thesis, we combine local (feature curve strength, length, parallelism, etc.) and global (density and symmetry) saliency measures to extract meaningful feature curve networks from raw potentially noisy input networks.

While the pure geometric information from these meaningful feature curve networks can be used directly for downstream applications, the more high-level structural information is not as easily accessible. However, for the analysis of the geometric data, more abstract information is required. Hence, in this thesis we present techniques to obtain structural information from symmetry, reoccurrence, and curve template membership, which can be exploited in shape analysis applications.

Finally, we present several applications, which benefit from the use of meaningful feature curve networks. These range from low-level geometry processing (remeshing, smoothing) to high-level shape analysis (functional maps).

Abstract (Deutsch)

Für die Analyse von 3D Modellen ist eine abstrakte Darstellung der Geometrie notwendig. Typischerweise geschieht dies über die Definition von lokalen Punkt-Deskriptoren oder globalen Signaturen, welche die gesamte Geometrie beschreiben. Punkt-Deskriptoren sind sehr sensitiv bezüglich lokaler Änderungen (z.B. durch Rauschen), während globale Signaturen (abhängig vom Ziel der Analyse) dazu neigen, zu grob zu sein. Eine geeignete Oberflächendarstellung sollte demnach verschiedene Abstraktionsebenen umfassen, welche von einer geometrischen Punkt-basierten Beschreibung über eine topologische Zwischen-Ebene bis hin zu einer globalen Struktur des Objekts reichen. In dieser Dissertation werden wir zeigen, dass Netzwerke, die Feature-Kurven enthalten, Informationen auf verschiedenen Abstraktionsebenen umfassen und damit eine geeignete Repräsentation für die komplexe Aufgabe der Analyse von 3D Modellen darstellen.

Feature-Kurven eines 3D Modells führen entlang auffälliger Linien (typischerweise Senken oder Hügelkämme). Die Berechnung dieser Kurven wurde in den vergangen Jahren ausgiebig erforscht, und verschiedenste Richtungen wie etwa Layout-Techniken, Selbstregistrierungs-Methoden, oder Krümmungslinien-Verfolgung haben sich etabliert. Sie liefern eine abstrakte Repräsentation wichtiger Teile der Geometrie und umfassen topologische und global strukturelle Information über das Modell, sowie geometrische Details. Die geometrischen Details können zum Beispiel in Geometrieverarbeitungs-Algorithmen wie Oberflächen-Glättung oder Dezimierung ausgenutzt werden, bei welchen sich die geometrischen Entitäten entlang der Feature-Kurven ausrichten sollten, um hohe Qualität zu garantieren. Die abstraktere strukturelle Information gibt wichtige Hinweise,

welche für die Analyse von 3D Modellen relevant ist.

Automatisch berechnete Feature-Kurven-Netzwerke auf Rohdaten können diverse Defekte wie etwa Rauschen, Fragmentierung oder das Fehlen wichtiger Daten enthalten. Um sie für Anwendungen nutzbar zu machen, werden relevante Feature-Kurven benötigt mit geringer Fragmentierung und ohne fehlklassifizierte Kurven (z.B. durch Rauschen). Erste Ansätze, welche relevantere Kurven aus automatisch berechneten Feature-Kurven-Netzwerken extrahieren, fallen auf lokale Lösungen wie Glättung, Krümmungs-basiertes Unterdrücken oder Filtern zurück. Dadurch werden jedoch auch schwächere Kurven entfernt, welche potenziell wichtige Details enthalten. In dieser Arbeit kombinieren wir lokale (Krümmungsstärke, Länge, Parallelität, usw.) und globale (Symmetrie und Dichte) Relevanz-Maße, um bedeutungsvolle Feature-Kurven-Netzwerke aus potenziell verrauschten Rohdaten zu extrahieren.

Während die geometrischen Daten dieser relevanten Feature-Kurven direkt genutzt werden können, ist die abstrakte strukturelle Information nicht einfach zugänglich. Allerdings wird letztere für die Analyse von 3D Geometrie benötigt. Deshalb stellen wir in dieser Dissertation Methoden vor, um Struktur aus Symmetrie, Wiederauftreten und Kurven-Template-Zugehörigkeit abzulesen, welche von Modell-Analyse Methoden verwendet werden kann.

Zum Abschluss präsentieren wir einige Anwendungen, welche von relevanten Feature-Kurven-Netzwerken profitieren. Hierbei werden Bereiche der Geometrieverarbeitung (Glättung, Dezimierung) und Modell-Analyse (functional maps) abgedeckt.

Acknowledgements

First of all, I thank Leif Kobbelt for his support over the last five years, which have not only lead to the completion of this thesis but have also given me the opportunity to tackle various exciting challenges in research. I am very grateful that I had the chance to do research with Leif, from whom I have learned incredibly much. Furthermore, I thank Mirela Ben-Chen who reviewed my thesis as well as the other members of the exam committee Wil van der Aalst and Martin Grohe.

This thesis is based on my research as a doctoral student at the Visual Computing Institute of the RWTH Aachen University where I held a position as a research assistant. My work received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n˚[340884].

I feel privileged that I had many great co-authors contribute to my research: Leif Kobbelt, Justin Solomon, David Bommes, Michael Bronstein, Isaak Lim, and Larry Wang. A special thanks goes to my close friend and colleague Isaak Lim, with whom I had intense discussions about research and who gave me strong moral support over the last five years. Also, I would like to express my gratitude to all present and former colleagues of the Visual Computing Institute, especially to the special interest group geometry processing: Max Lyon, Patrick Schmidt, and Janis Born; as well as to Jan Möbius, and two former colleagues and friends Sven Middelberg and H.C. Ebke who have made life very enjoyable at the Visual Computing Institute. Furthermore, I thank my former student assistant Tim Elsner, who had to deal with the worst kinds of feature curve networks.

I am very grateful that I had the chance to do part of my research with the Geometric Data Processing Group at the Massachusetts Institute of Technology under the supervision of Justin Solomon. The collaboration with Justin was amazing and I thank him and his group for all the exciting research discussions. Further, I thank the DAAD for sponsoring the semester abroad with the scholarship FIT weltweit. In this context I would also like to thank my MIT family Ben Arenas, Fernanda Basso, Rodrigo Urruela, Lik Khian Yeo, Thomas Kopfmüller, Idaly Ali, Enzo Peroni, and Teo Jun Hao, who have made the time in Boston incredible and unforgettable.

Finally, I thank my family and friends for their enduring support, especially my parents Margret and Hendrik Kathrein, who have supported me throughout my entire studies. I also thank Christine Kathrein, and Silke and Steffen Gehre for their moral support. Particularly, I would like to thank my husband Dominic Gehre for always being there for me, supporting all my decisions, dealing with all deadline related downsides as well as sharing the enthusiasm of successful research.

Contents

1 Introduction 1
1.1 Thesis Overview and Contributions 4
1.2 Notation 7

2 Fundamentals 11
2.1 Differential Geometry 11
2.1.1 Curves 12
2.1.2 Curvature of a Surface 13
2.1.3 Surface Metric and Geodesics 16
2.2 Data Analysis Fundamentals 16
2.2.1 Probability Theory 17
2.2.2 Clustering 20
2.2.3 Dimensionality Reduction 22
2.2.4 RANSAC 23
2.2.5 Neural Networks 24
2.3 Surface Representations 26
2.4 Estimating Curvature and Curvature Derivatives from Discrete Input 32
2.5 Feature Curve Networks 34
2.5.1 Computing Feature Curve Networks 35
2.5.1.1 Fully Automatic Techniques 35
2.5.1.2 Interactive Techniques 43
2.6 Correspondences and Symmetry Detection 44
2.6.1 Point Signatures 45
2.6.1.1 Signatures Invariant to Rigid Motion 46
2.6.1.2 Isometry Invariant Signatures 47
2.6.1.3 Topological Signatures 51
2.6.1.4 Learned Signatures 53
2.6.1.5 Distance Measures 55
2.6.2 Symmetry Detection 55
2.6.2.1 Symmetry Detection on Meshes 57

2.6.2.2 Symmetry Detection on Graphs 58
2.6.3 Consistent Correspondences on Shapes with Functional Maps 60
2.6.3.1 Point Correspondences from Functional Maps 62

3 Feature Curve Correspondences 65
3.1 Feature Curve Descriptors 66
3.1.1 Heat-Kernel-Signature for Feature Curves 67
3.1.1.1 Distance Computation 68
3.1.1.2 Limitations 68
3.1.2 Geodesic Iso-Curve Signature for Feature Curves 69
3.1.2.1 Geodesic Distances 69
3.1.2.2 Radii Computation 70
3.1.2.3 Distance Computation 71
3.1.2.4 Radius Sampling 72
3.1.2.5 Limitations 73
3.1.3 Learning Curve Correspondences 73
3.1.3.1 Input Representation 74
3.1.3.2 LSTM Network 75
3.1.3.3 Limitations 76
3.1.3.4 Unsupervised Learning 76
3.1.4 Results 77
3.2 Feature Curve Correspondences from Local-Global Voting . . . 83
3.2.1 Problem Statement 84
3.2.2 Partial Feature Curve Matching based on Local Transformation Space Voting 85
3.2.3 Global Transformation Space Voting 88
3.2.4 Orbit Detection 89
3.2.5 Verification 89
3.2.6 Results 90
3.3 Summary 93

4 Meaningful Feature Curve Networks 95
4.1 Meaningful Feature Curves 96
4.2 Feature Curve Co-Completion in Noisy Data 99
4.2.1 System Overview 99
4.2.2 Bayesian Model Selection 101
4.2.2.1 Heuristic Model Selection 104

4.2.3 Feature Curve Co-Completion . . . 104
4.2.3.1 Feature Curve Classification Noise Removal . 104
4.2.3.2 Template Generation . . . 105
4.2.4 Results . . . 107
4.3 Adapting Feature Curve Networks to a Prescribed Scale . . . 113
4.3.1 Problem Statement . . . 115
4.3.2 Scale Conforming Feature Curve Networks . . . 116
4.3.2.1 Quality Criteria . . . 117
4.3.2.2 Achieving Scale Conformity . . . 119
4.3.3 Feature Network Adaptation . . . 121
4.3.3.1 Initial Feature Curve Network . . . 122
4.3.3.2 Proximity Based Conflict Detection . . . 122
4.3.3.3 Feature Curve Selection . . . 125
4.3.4 Results . . . 130
4.4 Summary . . . 134

5 Applications . . . 135
5.1 Low-Level Geometry Processing with Meaningful FCNs . . . 135
5.1.1 Isotropic Remeshing . . . 136
5.1.2 Anisotropic Remeshing: Guiding Constraints for Level-of-Detail Quad Meshing . . . 140
5.1.3 Feature Preserving Smoothing . . . 143
5.2 Curve Constrained Functional Maps . . . 145
5.2.1 Problem Statement . . . 147
5.2.2 Input: Feature Curve Networks . . . 148
5.2.3 Functional Maps with Curve Constraints . . . 149
5.2.4 Iterative Update . . . 151
5.2.5 User Feedback . . . 152
5.2.6 Applications and Results . . . 154
5.2.6.1 Point-to-point correspondences . . . 155
5.2.6.2 Correspondence Accuracy . . . 157
5.2.6.3 Timing . . . 163
5.3 Summary . . . 168

6 Conclusion . . . 169
6.1 Outlook . . . 170

Bibliography . . . 173

1 Introduction

Recent developments in 3D shape acquisition, processing, and modeling have led to a vast amount of available 3D data (e.g. ShapeNet [Sha], Trimble 3D Warehouse [Tri], Thingiverse [Thib], Thingi10K [Thia], CGTrader [CGT], TurboSquid [Tur], ModelNet [Mod], and many more). This is driven by the fact that laser scanners and depth cameras are commonly available. Alongside, CAD (3D modeling) systems are constantly improving and enhanced with latest methodologies, which allows average users to handle geometry with relative ease. Also, the development of low-cost 3D printers have made 3D models appeal to an increasing audience. Hence, the amount of 3D data, which is available via search engines increases steadily.

This mass of digitized 3D geometry demands automatic processing, recognition, and comparison techniques, which remain challenging tasks. 3D geometric data is typically discretized in form of *point clouds* or *mesh* representations. The raw data that emerges from a scanning process or a CAD-design can have various defects, such as noise, high sample density, missing data, incomplete topological information, intersections, etc. Usually, this data cannot be used in downstream applications directly, which make assumptions on the quality, resolution, and completeness of the data. Low-level geometry processing techniques such as smoothing, remeshing, or completion can be applied to enhance the element quality and adapt the resolution of the geometric raw data. However, these algorithms often result in loss of information. E.g. by smoothing or remeshing a surface, high-frequent details are removed. In order to preserve such details, the methods need to be constrained to align with surface features.

Feature curves trace out prominent creases and crests of an object. These give significant hints about its shape and provide a descriptive yet concise representation of the 3D geometry. Feature curve networks encode surface details, which can be used to guide geometry processing applications. Ideally, such a feature curve network (FCN) should capture the global shape but also contain relevant small scale details. However, finding a FCN which only contains relevant information is not straight forward. There exists a body of work on the automatic and semi-automatic computation of feature curve networks (see e.g. [YBS05, BPK98, OBS04, WB01]). These methods are typically based on local curvature information. Hence, the resulting feature curve networks might be densely spaced, fragmented, and contain noise.

Beyond low-level processing, regarding the large amount of geometric data, we also require high-level information about the shapes for tasks such as retrieval, recognition, comparison, or segmentation. This data-driven high-level processing of geometry is referred to as *shape analysis* [XKH*16]. It involves the development of abstraction techniques to compare and analyze geometric data and allows inference about new data samples similar to standard *data analysis* techniques. Instead of describing shapes on a (local) point- or entire shape level, feature curves contain sparse yet salient geometric information with additional topological information given through the curve connectivity. Furthermore, the entire network captures the global structure of a shape. Hence, a meaningful set of feature curves would provide an abstract, sparse, and informative description for the analysis of shapes.

Such feature curve networks give beneficial input to low-level processing as well as high-level analysis. However, the notion of a *meaningful FCN* is quite abstract such that there seems to be no unique definition. First attempts to obtain more relevant feature curve networks from automatically computed input rely on local solutions such as (curvature based) thresholds, smoothing, or filtering of the curves. Hence, an important measure to evaluate the salience of a feature curve is via its

- *Sharpness/strength:* e.g. the magnitude of the curvature across the feature curve.

This however, is a purely local measure, which also removes weak meaningful curves from the network. [MZL*09] obtain feature curve based abstractions on man-made shapes by first fitting an envelope to the 3D shape. Creases and crests are detected on this envelope and the weakest curves are removed iteratively to receive a higher level of abstraction. Although the global shape is captured well by the envelope, small scale details are removed as well. [DGGDV11] couple the information of feature curves with global surface segmentations to combine geometry-driven detail with perceptually-driven global structural information. According to [DGGDV11], important properties of a feature curve network are:

- *Orientation:* Features should align along principal curvature directions.
- *Continuity:* Long curves are more likely to encode structure in a shape and might be more meaningful.
- *Regularity:* Feature curves should be well spaced over the surface and adapt to the surface density.

While the first two properties are local measures, the latter one can effectively only be evaluated with regard to the global structure of the feature curve network. The *Exoskeletons* created in [DGGDV11] again only capture a rough segmentation of the object such that density constraints automatically hold. A meaningful FCN should also contain relevant small scale details. In this case, the structure of the entire FCN needs to be taken into account to abide to density constraints. [SJW*11] present a semisupervised approach to feature-curve detection based on partial feature curve

- *Symmetry/reoccurrence:* Instances that appear multiple times in an object are more likely to be meaningful and should be included in the feature curve network.

Reoccurrence in a feature curve network is also a global property and requires the detection of symmetric feature curve (configurations). In [SJW*11] a user has

to draw instances of the reoccurrences she expects into the geometry. The fully automatic detection of reoccurring feature curve templates provides important hints, such that small scale as well as weak features can be identified as relevant, if they belong to a larger reoccurring template of curves. Furthermore, reoccurring instances can be completed based on the combined information to decrease the fragmentation and missing data in the FCN.

Obviously, there is a set of desirable properties that combine a meaningful feature curve network. While local properties as the feature strength, curve-length, etc. are easy to evaluate on a feature curve, deciding which curves to preserve with global density constraints is quite challenging. Furthermore, finding partial symmetries in a feature curve network, possibly across multiple shapes, can be complex depending on the type of input data due to noise or different types of deformations.

1.1 Thesis Overview and Contributions

In this thesis we develop algorithms to compute globally and locally meaningful feature curve networks. We present techniques to augment these with high-level structural information such as reoccurrence and template membership. Furthermore, we apply and develop methods where such meaningful FCNs can yield enhanced results in the field of low-level geometry processing and high-level shape analysis.

While local measures can be adopted mostly from previous work, to the best of our knowledge there are no related fully automatic techniques which allow to enhance or constrain a FCN based on the global properties introduced above. Hence, our goal is to create meaningful feature curve networks by:

1. Exploiting *reoccurring* feature curve templates to (1) separate relevant reoccurring features from noise. This includes weak and small scale details as well as strong feature curves in the final output. (2) We resolve

fragmentation and missing data issues by completing the FCN based on the feature curve templates.

2. Adapt FCNs to a prescribed global *scale*. Given an input FCN we compute a maximal subset of relevant feature curves which abides to the global density constraints.

We start by introducing fundamentals in differential geometry, data analysis, and feature curve networks, and discuss related work in Chapter 2.

Finding correspondences among feature curves is essential to nearly all following analysis steps. Reoccurrence is a strong indicator for the relevance of a feature curve, i.e. in case a curve is observed multiple times it is unlikely detected due to noise. Depending on the use case we might be facing various challenges. E.g. we may analyze differently articulated shapes, there may be high accuracy requirements in task specific settings, or we could be facing large real world data sets, which are influenced by noise and under-sampling. Hence, in Chapter 3 we present several techniques how to obtain feature curve correspondences.

In case the regarded shapes underlie intrinsic deformations or represent different shapes of the same family of objects, the task of finding corresponding feature curves can become increasingly complex. In this case, we are required to evaluate intrinsic measures on the curves. We present two feature curve descriptors, which can be used to measure feature curve similarity. To the best of our knowledge, there have been no previous approaches to feature curve descriptors which take intrinsic measures into account.

However, handcrafting descriptors is hard and in many cases too complex to obtain high correspondence accuracies. If we would like to find task specific correspondences, e.g. for a certain family of shapes, we can learn a classifier from examples. Hence, we present a supervised technique to learn feature curve correspondences with neural networks, which automatically finds task-dependent representations of feature curves.

Finally, in a setting where real-world data is analyzed a correspondence detection technique is required, which is robust even under the influence of noise in a

large amount of potentially misclassified feature curves. We develop a method to detect feature curves that are symmetric under rigid transformations, which is robust to noise and fragmentation. Our goal is to obtain FCNs for raw input data. As discussed above, this is very challenging due to various defects. Hence, we cannot rely on local information only. In this context we develop a similarity measure denoted as *Feature BLAST* inspired by the *Basic Local Alignment Search Tool* [AGM*90]. Based on this, we present a two-step local-global voting procedure to robustly identify curves that are symmetric under rigid transformations.

With this portfolio of techniques, which allow to find corresponding feature curves, we are able to extract meaningful feature curve networks according to the requirements discussed above. In Chapter 4 we first present an approach, which is able to complete fragmented noisy feature curve networks by identifying reoccurring curve templates. After a symmetry detection phase, several hypotheses on how the feature curve data reoccurs are proposed by the system. We let the entirety of the feature curve data decide which curve templates are most descriptive but least redundant by *Bayesian Model Selection*. With the detected templates, fragmented curves are completed without the need for local heuristics. Furthermore, small scale reoccurring details are captured, while non-reoccurring weak curves are classified as noise. As a result we receive high-level structural information about reoccurring templates, as well as a clean feature curve network containing meaningful feature curves.

Secondly, we present a technique which is able to adapt the scale of a feature curve network to a prescribed resolution. For this, we define properties that a *scale conforming* FCN needs to fulfill in order to detect curve segments that cannot coexist. To find a maximal set of non-conflicting curves, we propose a binary labeling program. We compute a global optimum with the objective to preserve a maximal set of meaningful feature curves that can be represented at a given resolution.

The resulting meaningful FCNs can be leveraged to low-level geometry processing tasks as well as high-level shape analysis applications. In Chapter 5 we

present applications such as feature constrained smoothing and remeshing to obtain high quality surface representations.

Furthermore, we show how feature curve networks can introduce additional topological information to inter-surface mappings based on *functional maps*. Functional maps transport functions from one surface to another. Methods exist which infer point-to-point correspondences from the functional map representation. Standard functional maps, presented in e.g. [OBCS*12, NMR*18], are able to find accurate mappings between isometrically deformed shapes. If the shapes have strong non-isometric deformations, pure geometric cues alone are not sufficient to compute precise maps. We present a technique to include semantic cues provided by corresponding feature curves on the shapes into the functional maps framework. With this technique it is possible to find accurate mappings between highly non-isometric shapes.

Overall, the methods presented in this thesis lead the way from raw input data to abstract high-level information by incorporating global (symmetry and scale) and local (strength, length, etc.) cues into a feature curve network. The combined geometric and topological information is applicable to various utilities as the exemplary ones introduced in this thesis. We will elaborate this in detail in the final Chapter 6.

Some of the material in this thesis has previously been published in [GLK16, GBK16, GLK18, GBKS18].

1.2 Notation

In Table 1.1 we summarize the most important notation, which is used consistently throughout the thesis. In general, we emphasize vectors and matrices in bold letters in contrast to scalar values. We capitalize matrices, while vectors are denoted by lower case letters. The transpose of a matrix is indicated by T. Values that are given by their frequency coefficients are expressed by $\tilde{\cdot}$. Furthermore, we index a vector $\boldsymbol{v}$, e.g. at the i-th element with the notation $\boldsymbol{v}[i]$.

symbol	meaning
$\Gamma = (V, E, C)$	feature curve network
γ, c	continuous and discrete curves
s	curve segments
$\mathcal{S}$	continuous surface
$\boldsymbol{M} = (V, E, F)$	discrete mesh
$g_{\mathcal{S}}, g_{\boldsymbol{M}}$	metric
$\mathcal{N}_k(\cdot)$	k-ring neighborhood
$\boldsymbol{P}$	point set
$\kappa_{\max}, \kappa_{\min}$ and $\mathbf{t}_{\max}, \mathbf{t}_{\min}$	principal curvatures and principal directions
$\boldsymbol{n}$	normals
$\boldsymbol{p}$	surface points
$d(\cdot, \cdot)$	distance measure
$\Delta_{\mathcal{S}}, \boldsymbol{L}_{\boldsymbol{M}}$	Laplace–Beltrami operator
$\boldsymbol{\Phi}_{\mathcal{S}}$	Eigenbasis of the Laplace–Beltrami operator
$\boldsymbol{\phi}$	eigenvector
λ	eigenvalue
$\boldsymbol{S}$	discrete shape operator
$\mathcal{E}$	energy or objective function
β, θ	angles
$l(\cdot)$	length
$A(\cdot)$	area
$w, \boldsymbol{W}$	weights
$\|\cdot\|_2$	Euclidean/2-norm
$\|\cdot\|_{\text{Fro}}$	Frobenius norm
μ	mean
σ	variance
$p_{\text{c}}(\cdot), p_{\text{d}}(\cdot)$	continuous and discrete probabilities
$p_{\text{df}}(\cdot)$	probability density function
$\mathcal{L}(\cdot)$	likelihood function
$\boldsymbol{\mathcal{D}} = \{\mathcal{d}_1, \ldots, \mathcal{d}_k\}$	data

$\mathcal{H}$	hypothesis/model
T	transformation/point map
T_F, $\tilde{F}$	functional map

Table 1.1: A quick reference of the most important notation used throughout this thesis.

2 Fundamentals

Feature curve networks provide valuable abstract information for various types of geometric representations and algorithms. They can guide low-level geometry processing tasks, e.g. feature aware surface reconstruction methods from point clouds, or isotropic and anisotropic remeshing algorithms as well as high-level shape analysis tasks such as symmetry detection, inter-surface correspondence computation, or object segmentation.

Due to the broad variability and applicability of FCNs, we will first build the theoretical foundations required to construct, process, apply, and analyze feature curve networks. After introducing relevant basics in differential geometry and data analysis, we discuss possible surface representations in Section 2.3. Then we formally define FCNs and describe possible methods to compute these (Section 2.5). For the analysis of feature curve networks, we need to make curves comparable. Therefore, we will discuss basics of point-signatures (Section 2.6.1), symmetry detection (Section 2.6.2), and inter-surface correspondence computation (Section 2.6.3).

Some of the material, definitions, and findings in this chapter have previously been published in [GLK16], [GBK16], and [GBKS18].

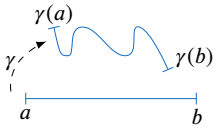
2.1 Differential Geometry

To detect and analyze feature curve networks, it is essential to define a notion of curvature on curves and surfaces. Curves and surfaces will be considered as *Riemannian manifolds* $\left(\mathcal{M}, g_{\mathcal{M}}\right)$, where $\mathcal{M}$ is a smooth (C^∞) manifold. The *metric* $g_{\mathcal{M}}$ is required to measure distances over curved manifolds. While in the

"straight" Euclidean space we can measure distances between two points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ via inner products $\langle \boldsymbol{p}_1 - \boldsymbol{p}_2, \boldsymbol{p}_1 - \boldsymbol{p}_2 \rangle$, it is not directly clear how to measure these in a curved space. There, the surface metric is induced by the inner product on the tangent space of $\mathscr{M}$. In the following we will give a concise overview over important basics in differential geometry.

2.1.1 Curves

In this section we consider curves as smooth 1-manifolds embedded in $\mathbb{R}^3$. In parametric form we describe a curve $\gamma : [a, b] \to \mathbb{R}^3$ as a function that maps a value $u \in \mathbb{R}$ from the interval $[a, b]$ to a point on the curve, i.e., $\gamma(u) = (x(u), y(u), z(u))^\top$. Such a parametric representation is especially useful if it preserves the length of the curve.



Arc length We can compute the length of a curve segment by integrating over its tangent vector. The tangent vector at a point $\gamma(u)$ is given by the first derivative [BKP*10]:

$$\frac{\partial \gamma(u)}{\partial u} = \begin{pmatrix} \frac{\partial x(u)}{\partial u} \\ \frac{\partial y(u)}{\partial u} \\ \frac{\partial z(u)}{\partial u} \end{pmatrix}.$$

The arc length of the curve γ in the interval $[a_0, b_0] \subset [a, b]$ is then computed as:

$$l(a_0, b_0) = \int_{a_0}^{b_0} \|\gamma'\|_2 \, du \quad \text{with } \gamma' = \frac{\partial \gamma}{\partial u}.$$

Since we can measure distances between points on the curve by the arc length of the respective segment it encodes its *metric*. If the parametric mapping is length preserving, we denote it as *arc length parametrization*. This mapping is unique up to orientation and assigns the interval $[a, b]$ to $[0, L = l(a, b)]$.

Curvature The curvature of a curve measures how much it deviates from a straight line. Given a curve γ in arc length parametrization, the curvature at $\gamma(u)$, $u \in [0, L]$ is defined as the norm of the second order derivatives [BKP*10]:

$$\kappa(u) = \left\| \frac{\partial^2 \gamma(u)}{\partial^2 u} \right\|_2 .$$

2.1.2 Curvature of a Surface

Similar to the curve case, we can compute curvature on the surface by taking the parametric representation of a smooth surface $\mathcal{S}$ into account. In this case $\mathcal{S}$ is considered a 2D manifold embedded in $\mathbb{R}^3$. We denote the 2D parameter domain by $\Omega \subset \mathbb{R}^2$. In parametric form $\mathcal{S}$ is given as:

$$\mathcal{s}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}.$$

If we consider directional derivatives on $\mathcal{S}$ at a point $(u_0, v_0) \in \Omega$, we can treat the surface case similar to the curve case. Assume we are given an arbitrary (tangent) direction $\overline{\boldsymbol{\omega}}$ in Ω. We define a straight line as $(u, v) = (u_0, v_0) + t\overline{\boldsymbol{\omega}}$ with its image $\mathcal{s}_\omega(t) = \mathcal{s}\left((u_0, v_0) + t\overline{\boldsymbol{\omega}}\right)$. Again, we can compute the tangent $\boldsymbol{\omega}$ as the derivative of $\mathcal{s}_\omega(t)$ at $t = 0$:

$$\begin{aligned} \boldsymbol{\omega} = \frac{\partial \mathcal{s}_\omega(t)}{\partial t} &= \begin{pmatrix} \frac{\partial \mathcal{s}\left((u_0, v_0) + t\overline{\boldsymbol{\omega}}\right)}{\partial u} & \frac{\partial \mathcal{s}\left((u_0, v_0) + t\overline{\boldsymbol{\omega}}\right)}{\partial v} \end{pmatrix} \cdot \overline{\boldsymbol{\omega}} \\ &\overset{t=0}{=} \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \cdot \overline{\boldsymbol{\omega}} = \boldsymbol{J}(u_0, v_0) \cdot \overline{\boldsymbol{\omega}}. \end{aligned}$$

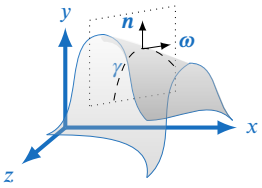
The matrix $\boldsymbol{J}$ denotes the *Jacobian* of the parametrization function $\mathcal{s}$. $\boldsymbol{J}(u_0, v_0)$ transforms a parameter space vector $\overline{\boldsymbol{\omega}}$ at (u_0, v_0) to a tangent vector $\boldsymbol{\omega}$ [BKP*10]. The Jacobian encodes the metric of the surface since it allows to compute differences of angles, lengths, and areas that are induced by the map. E.g. we can compute the angle between two unit-vectors as $\boldsymbol{\omega}_1^\top \boldsymbol{\omega}_2 = \overline{\boldsymbol{\omega}}_1^\top \boldsymbol{J}^\top \boldsymbol{J} \overline{\boldsymbol{\omega}}_2$ or the length

of a vector $\boldsymbol{\omega}^\top \boldsymbol{\omega} = \overline{\boldsymbol{\omega}}^\top \boldsymbol{J}^\top \boldsymbol{J} \overline{\boldsymbol{\omega}}$.

The matrix $\boldsymbol{J}^\top \boldsymbol{J}$ is referred to as the *first fundamental form*. It computes an inner product, that is defined on the tangent space of $\mathcal{S}$, and thus induces a metric on $\mathcal{S}$:

$$\boldsymbol{I} = \boldsymbol{J}^\top \boldsymbol{J}$$

To introduce a notion of curvature on surfaces we need to define the *normal curvature*. A tangent vector $\boldsymbol{\omega}$ at a point $\boldsymbol{p} \in \mathcal{S}$ and the surface normal $\boldsymbol{n}$ at $\boldsymbol{p}$ span a plane, which intersects $\mathcal{S}$. This intersection yields a planar curve γ. The curvature at $\boldsymbol{p}$ of this planar curve defines the normal curvature



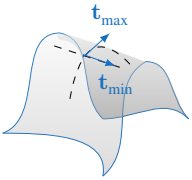
$$\kappa_n(\overline{\boldsymbol{\omega}}) = \frac{\overline{\boldsymbol{\omega}}^\top \boldsymbol{II}\, \overline{\boldsymbol{\omega}}}{\overline{\boldsymbol{\omega}}^\top \boldsymbol{I}\, \overline{\boldsymbol{\omega}}},$$

with the *second fundamental form* also known as *Shape operator* [BKP*10]:

$$\boldsymbol{II} = \begin{bmatrix} \frac{\partial^2 \mathfrak{z}}{\partial u^2}\boldsymbol{n} & \frac{\partial^2 \mathfrak{z}}{\partial u \partial v}\boldsymbol{n} \\ \frac{\partial^2 \mathfrak{z}}{\partial u \partial v}\boldsymbol{n} & \frac{\partial^2 \mathfrak{z}}{\partial v^2}\boldsymbol{n} \end{bmatrix}.$$

To understand this formula let us think of curvature as a change of surface normals. The differential of the surface $\mathfrak{z}$ yields a vector tangent to γ. Taking the differential of this tangent results in the surface normal along $\mathfrak{z}$ [CDGDS13]. By taking the dot-product with the normal $\boldsymbol{n}$ at $\boldsymbol{p}$ we receive the difference in the surface normals with respect to the regarded direction. The denominator, which computes the length of $\boldsymbol{\omega}$, is used for normalization.

For any tangent direction $\boldsymbol{\omega}$ we can compute the normal curvature at $\boldsymbol{p}$. However, we are interested in the extremal, i.e. the *principal* curvature values, called the *maximum curvature* $\kappa_{\max}$ and the *minimum curvature* $\kappa_{\min}$, with the respective *principal curvature directions*



$\mathbf{t}_{max}$ and $\mathbf{t}_{min}$. We can derive these extremal directions and curvatures from $\boldsymbol{II}$, by computing the eigenvectors and corresponding eigenvalues of $\boldsymbol{II}$ [CDGDS13], since $\boldsymbol{II}$ is real and symmetric. On the surface, the extremal directions yield two vector fields. The *integral curves* of a vector field are parametric curves, which are tangent to the vector field. Hence, the integral curves of the extremal direction fields follow the principal curvature directions and are referred to as *curvature lines* [OBS04]. In case $\kappa_{max} \neq \kappa_{min}$ the principal directions are unique and orthogonal to each other. A point where one of the principal curvatures vanishes is called *parabolic*. Points with $\kappa_{max} = \kappa_{min}$ are denoted as *umbilical points*.

Curvature Derivatives Several feature curve detection methods trace out extrema of the maximal and minimal curvature directions. Thus we are also interested in the curvature derivatives in the principal curvature directions, which are referred to as *extremality coefficients* [OBS04]:

$$\kappa'_{max} = \frac{\partial \kappa_{max}}{\partial \mathbf{t}_{max}} = \nabla \kappa_{max} \cdot \mathbf{t}_{max}$$
$$\kappa'_{min} = \frac{\partial \kappa_{min}}{\partial \mathbf{t}_{min}} = \nabla \kappa_{min} \cdot \mathbf{t}_{min}$$

The curvature strength $\max\left(|\kappa_{min}|, |\kappa_{max}|\right)$ indicates the salience of a surface point. However, it also classifies small fluctuations as significant, which might exist due to noise. At this point we introduce the notion of a *Dupin cyclide*. One way to characterize a Dupin cyclide in $\mathbb{R}^3$ is that all its curvature lines are circles, i.e. it is covered by two orthogoanally intersecting families of curves. Dupin cyclides fulfill the following property [YBS05]:

$$|\kappa'_{max}|^2 + |\kappa'_{min}|^2 = 0.$$

I.e. the curvature is constant along the curvature lines. The integral over the *cycledness* $\sqrt{|\kappa'_{max}|^2 + |\kappa'_{min}|^2}$ of a curve c with arc length l along the curve:

$$\mathcal{T} = l \int \sqrt{|\kappa'_{max}|^2 + |\kappa'_{min}|^2} dc \tag{2.1}$$

indicates how much the surface deviates from being part of a Dupin cyclide [YBS05]. For small pertubations, where the surface is close to being a Dupin cyclide, $\mathcal{T}$ is small [YBS05]. This measure is particularly useful to detect salient surface segments. Regions where the curvature changes usually contain more information than those with constant curvature.

2.1.3 Surface Metric and Geodesics

A *metric* $g_{\mathcal{S}}$ allows to measure distances $d_{\mathcal{S}} : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ on the surface [ZZCJ14]. Given a scalar norm $\|\cdot\|_m$ and a curve $\gamma : [0,1] \to \mathcal{S}$ with tangent vectors γ' the length of gamma is measured by

$$d_{\mathcal{S}}(\gamma(0),\gamma(1)) = \int_0^1 g_{\mathcal{S}}(\gamma',\gamma')\,du = \int_0^1 \|\gamma'\|_m\,du.$$

If we use the standard *Euclidean metric* we receive exactly the arclength as defined above. The metric is called *intrinsic* if d always returns the length of the shortest path between two points on the surface. We refer to the intrinsic Euclidean metric as *geodesic distance* between $\gamma(0)$ and $\gamma(1)$. A norm is called *isotropic* if it is direction invariant. In some cases (e.g. when curvature information is given) elliptic norms can be of interest. For *anisotropic norms* each point $\boldsymbol{p} \in \mathbb{R}^3$ is associated with a tensor $\boldsymbol{\mathcal{T}}_m(\boldsymbol{p})$ (e.g. curvature tensor/shape operator or a direction field tensor), such that $\|\boldsymbol{v}\|_{m,\boldsymbol{p}} = \sqrt{\boldsymbol{v}^\top \boldsymbol{\mathcal{T}}_m(\boldsymbol{p})\boldsymbol{v}}$ for $\boldsymbol{v} \in \mathbb{R}^3$[CHK13]. The shortest path along a surface according to an anisotropic metric is denoted as *anisotropic geodesic*.

2.2 Data Analysis Fundamentals

Due to recent developments in 3D acquisition, a large amount of 3D data is available (e.g ShapeNet [Sha], Trimble 3D Warehouse [Tri], Thingiverse [Thib], Thingi10K [Thia], CGTrader [CGT], TurboSquid [Tur],ModelNet [Mod], and many more). By applying shape analysis methods, we are able to infer information obtained from the data at hand. Therefor the field of data analysis naturally lends itself with well established methods. In general, the objective in data

analysis is to find patterns in data [Bis06]. There exists a variety of learning based methods to achieve this, which can roughly be classified into *supervised learning*, *unsupervised learning*, and *reinforcement learning*. In the first case, the applications have access to input examples with specified targets or labels (e.g. classification or regression). In unsupervised scenarios, no corresponding targets are given. Typical unsupervised tasks are *clustering* (finding groups of similar entities in the data), or *dimensionality reduction* (e.g. for the purpose of data visualization or comparison). Reinforcement learning methods make descisions based on rewards that can be earned by interacting with the enviroment. Here, we will only scratch the surface of data analysis methods and introduce techniques relevant for further reading. We refer the interested reader to [Bis06].

2.2.1 Probability Theory

Often, information that we are given (especially from real-world measurements) is incomplete, ambiguous, or noisy. Hence, it is important to introduce the concept of *uncertainty*. For this, let us assume that we are given probabilities $p_{\mathrm{d}}(X)$ for a random variable X over a discrete set of datapoints $\mathcal{D} = \{\mathscr{d}_1, \ldots, \mathscr{d}_k\}$ with the probablity that an event $\mathscr{d}_i$ occurs: $p_{\mathrm{d}}\left(X = \mathscr{d}_i\right) \in [0, 1]$. If two data samples are measured simultaneously we compute the *joint probability*, e.g. $p_{\mathrm{d}}\left(X = \mathscr{d}_i, Y = \mathscr{d}_j\right)$. In case X and Y are *stochastically independant* $p_{\mathrm{d}}(X, Y) = p_{\mathrm{d}}(X)\, p_{\mathrm{d}}(Y)$. The *conditional probability*, i.e. $p_{\mathrm{d}}\left(Y = \mathscr{d}_j | X = \mathscr{d}_i\right)$, measures the probability of $\mathscr{d}_j$ occurring under the assumption that $X = \mathscr{d}_i$. We can compute the *marginal probability* from the joint probabilites by applying the *sum rule* [Bis06]:

$$p_{\mathrm{d}}\left(X = \mathscr{d}_i\right) = \sum_{j=1}^{m} p_{\mathrm{d}}\left(X = \mathscr{d}_i, Y = \mathscr{d}_j\right).$$

Furthermore, joint probabilities are obtained from the *product rule*:

$$p_{\mathrm{d}}(X, Y) = p_{\mathrm{d}}(Y|X)\, p_{\mathrm{d}}(X).$$

With these rules and the symmetry property $p_{\mathrm{d}}(X, Y) = p_{\mathrm{d}}(Y, X)$ it is possible to derive the famous *Bayes' theorem*:

$$p_{\mathrm{d}}(Y|X) = \frac{p_{\mathrm{d}}(X|Y)\, p_{\mathrm{d}}(Y)}{p_{\mathrm{d}}(X)} = \frac{p_{\mathrm{d}}(X|Y)\, p_{\mathrm{d}}(Y)}{\sum_{\mathscr{A}_i} p_{\mathrm{d}}\left(X|Y = \mathscr{A}_i\right) p_{\mathrm{d}}\left(Y = \mathscr{A}_i\right)}.$$

In case the variables are continuous, we regard the *probability density function* $p_{\mathrm{df}}(x)$ over x. Then, the probability that X lies in an interval (a, b) is given by:

$$p_{\mathrm{c}}(X \in (a, b)) = \int_a^b p_{\mathrm{df}}(x)\, dx.$$

A probability density function satisfies the following conditions:

$$p_{\mathrm{df}}(x) \geq 0$$
$$\int_{-\infty}^{\infty} p_{\mathrm{df}}(x)\, dx = 1$$

The rules specified in the discrete case likewise hold in the continuous case. In the following we denote discrete probabilities by $p_{\mathrm{d}}(\cdot)$ and continuous probabilities by $p_{\mathrm{c}}(\cdot)$ with probability density functions $p_{\mathrm{df}}(\cdot)$. In case the presented material applies to both we denote this by $p(\cdot)$.

An effective way to describe a set of data points with function values $f(x)$ based on their probabilities is to compute an average value, called the *mean* or the *expectation* of the function values under the probability density $p_{\mathrm{df}}(x)$:

$$\mu(f) = \int p_{\mathrm{df}}(x)\, f(x) dx.$$

For a finite set of measurements, sampled according to the probability density, the expectation value can be approximated by:

$$\mu(f) \approx \frac{1}{k} \sum_{i=1}^{k} f\left(x_i\right).$$

The *variance* describes the amount of variability (or spread) of a function around the mean and is defined by:

$$\sigma(f) = \mu\left([f - \mu(f)]^2\right)$$

The *covariance* describes the common variance of two random variables. I.e. if they are stochastically independent their covariance equals zero. The covariance is computed by:

$$\sigma\left(\ell_1, \ell_2\right) = \mu\left(\left(\ell_1 - \mu\left(\ell_1\right)\right)\left(\ell_2 - \mu\left(\ell_2\right)\right)\right).$$

We can further ask questions about the data by setting up a hypothesis (e.g. the points are samples of a specific distribution) over the data-set. Let us assume we have one or several hypotheses $\mathcal{H}$ about our data, and we want to evaluate whether these apply. With Bayes' formula we can evaluate the posterior probability of $\mathcal{H}$ given the data $\mathcal{D}$:

$$p\left(\mathcal{H}|\mathcal{D}\right) = \frac{\mathscr{L}\left(\mathcal{D}|\mathcal{H}\right) p\left(\mathcal{H}\right)}{p\left(\mathcal{D}\right)}.$$

For this we need to know the *prior probability* $p\left(\mathcal{H}\right)$ of the hypothesis $\mathcal{H}$, the probability of the data under any possible hypothesis $p\left(\mathcal{D}\right)$, as well as $\mathscr{L}\left(\mathcal{D}|\mathcal{H}\right)$, a likelihood value (i.e. in contrast to probabilities these do not integrate to 1) which describes how likely the data is measured under the hypothesis $\mathcal{H}$. Since the marginal probability $p\left(\mathcal{D}\right)$ can be computed by:

$$p_{\mathrm{d}}\left(\mathcal{D}\right) = \sum_{\mathcal{H}} \mathscr{L}\left(\mathcal{D}|\mathcal{H}\right) \cdot p_{\mathrm{d}}\left(\mathcal{H}\right) \text{ or}$$
$$p_{\mathrm{c}}\left(\mathcal{D}\right) = \int_{\mathcal{H}} \mathscr{L}\left(\mathcal{D}|\mathcal{H}\right) \cdot p_{\mathrm{df}}\left(\mathcal{H}\right) d\mathcal{H}$$

over all possible models $\mathcal{H}$ or model parameters, the likelihood values are normalized such that $p\left(\mathcal{H}|\mathcal{D}\right)$ results in a conditional probability.

In *classical probability theory*, also denoted as *frequentest approach*, probabilities are interpreted as frequencies of randomly reoccurring observations. Here, only one specific hypothesis can be assumed and model parameters (e.g. of the underlying distribution) are optimized according to the observed data (e.g. *regression* to a predefined type of probability distribution). Especially, this means that no prior probability on the hypothesis is required.

In contrast, in the *Bayesian view* uncertainty is modeled as probability distributions over different hypotheses. For this however, the prior probability over the hypothesis needs to be designed, which is not always straightforward. Let us assume that we have k hypotheses or *models* $\mathcal{H}_1, \ldots, \mathcal{H}_k$ about how our data could be distributed. With *Bayesian model selection* we can find the model which is most likely, given the measured data $\mathcal{D} = \{\mathcal{d}_1, \ldots, \mathcal{d}_m\}$. According to Bayes'rule:

$$p\left(\mathcal{H}_i|\mathcal{D}\right) = \frac{p\left(\mathcal{D}|\mathcal{H}_i\right) \cdot p\left(\mathcal{H}_i\right)}{p\left(\mathcal{D}\right)}.$$

If the data is measured independently, each observation can be taken into account separately:

$$p\left(\mathcal{H}_i|\mathcal{d}_1, \ldots, \mathcal{d}_m\right) = \frac{\prod_{j=1}^{m} p\left(\mathcal{d}_j|\mathcal{H}_i\right) \cdot p\left(\mathcal{H}_i\right)}{\prod_{j=1}^{m} p\left(\mathcal{d}_j\right)}.$$

It is possible to avoid the computation of the denominator, by comparing two models via the *Bayes ratio* [KR93]:

$$\mathcal{K} = \frac{p\left(\mathcal{H}_i|\mathcal{D}\right)}{p\left(\mathcal{H}_j|\mathcal{D}\right)} = \frac{p\left(\mathcal{D}|\mathcal{H}_i\right) \cdot p\left(\mathcal{H}_i\right)}{p\left(\mathcal{D}|\mathcal{H}_j\right) \cdot p\left(\mathcal{H}_j\right)}$$

A value of $\mathcal{K} > 1$ means that the model $\mathcal{H}_i$ has stronger support for the data than $\mathcal{H}_j$. According to [Jef98, KR95], $\mathcal{K}$ gives insights on the weight of evidence to select one model over the other. Following their scheme, a value of $\mathcal{K} > 10$ strongly supports the model $\mathcal{H}_i$ over $\mathcal{H}_j$, while a value close to 1 means that the models have a similar descriptive power [GLK18]. In classical hypothesis testing there is typically a model $\mathcal{H}_{\text{null}}$ denoted as *null hypothesis*. This model considers evidence against the other hypotheses. $\mathcal{H}_{\text{null}}$ is used for verification: If $\mathcal{H}_{\text{null}}$ is selected over the other models, none of the proposed hypotheses are significant for the given data.

2.2.2 Clustering

Given a set of data points $\mathcal{D} = \{\mathcal{d}_1, \ldots, \mathcal{d}_m\}$ in multidimensional space, clustering techniques compute groups (*clusters*) of similar samples [Bis06]. Throughout

this thesis we will mention several clustering methods, which we will briefly explain in the following.

k-Means, also called *Lloyds' clustering*, is a non-probabilistic clustering technique, which partitions the data into k clusters. The parameter k is given as input to the algorithm. Initially, k samples are chosen as cluster representatives $\mu_1, \ldots, \mu_k$. These are updated iteratively in a two-step procedure:

1. *Expectation:* Assign all samples to the closest cluster representative. This minimizes the global distance of the data samples to their assigned cluster.

2. *Maximization:* Update the cluster representatives by computing the mean of the assigned cluster samples.

The procedure is iterated until convergence. This clustering algorithm is very simple, and the resulting clusters will always have a convex shape [EC02]. The naming of the two steps derive from the probabilistic version of this algorithm called *Expectation-Maximization* clustering. It assumes that data-points are samples from a mixture model of distributions which is fitted to the data (e.g. a Gaussian mixture model).

Mean-Shift clustering does not require information about the number of expected clusters [FH75]. Furthermore, the computation is based on the density of the data-points, and hence arbitrary cluster shapes can result from this procedure. For each point $\boldsymbol{p}$ the sample density is evaluated within a local radius. Within this radius a kernel function (e.g. a Gaussian kernel) is evaluated that allows to compute a weighted mean $\mu(\boldsymbol{p})$ of the density function. Then $\boldsymbol{p}$ is shifted to $\mu(\boldsymbol{p})$. This procedure is iterated until convergence. The found extrema in the point density are chosen as cluster centers. All points from which the mean-shift procedure converges to the same mode are assigned to the respective cluster. Here, the amount of detected clusters heavily depends on the size of the preselected local radius.

Hierarchical clustering algorithms build cluster hierachies. These can be *agglomerative* (bottom up) or *devisive* (top down) [RM05]. In the first case each data-point is contained in one cluster at the bottom layer (i.e. the leafs of the hierarchy) and these are merged according to a preset metric. For the latter case the root of the hierarchy contains all data-points in one large cluster. This root cluster is split recursively, while moving down the hierarchy. The merging and splitting operations are performed based on a similarity measure. There are three main categories of similarity measures [RM05]:

- *Single-link clustering* measures the smallest distance between any two samples from two clusters.
- *Complete-link clustering* measures the largest distance between any two samples from two clusters.
- *Average-link clustering* computes the average distance between all samples from two clusters.

A clustering can be obtained by cutting the hierarchy at a desired similarity level. In contrast to the previous methods, hierarchical clustering automatically produces multiple partitions of the data at different similarity levels. Furthermore, the shape of the clusters can be modeled by the distance functions. On the other hand, hierarchical clustering can be computationally expensive with a runtime of at least $O(m^2)$. In practice the runtime can be improved by adding topological constraints which reduce the search space (e.g. [YNBT01]).

2.2.3 Dimensionality Reduction

Principal component analysis (PCA) is a technique that computes the main axes of variation of a data-set. The dimensionality of the data can be reduced if it is represented by its projection into the lower dimensional coordinate system spanned by these axes. This is especially useful for data compression, feature extraction, and data visualization [Bis06]. Given a set of data-points $\mathcal{D} = \{\mathscr{d}_1, \ldots, \mathscr{d}_m\}$ in n-dimensional space ($\mathscr{d}_i \in \mathbb{R}^n$), we first compute the k axes of largest variation.

For this we need to compute the variance of the projected data in the following way [Bis06]:

1. Compute the mean $\mu_D = \frac{1}{m}\sum_{i=1}^{m} \mathscr{d}_i$ of the data-points.
2. Compute the covariance matrix $\Sigma_D = \frac{1}{m}\sum_{i=1}^{m} \left(\mathscr{d}_i - \mu_D\right)\left(\mathscr{d}_i - \mu_D\right)^\top$.
3. The variance along an axis $\boldsymbol{u}$ of the projected data is given by

$$\sigma_D^{\boldsymbol{u}} = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{u}^\top \mathscr{d}_i - \boldsymbol{u}^\top \mu_D\right)^2 = \boldsymbol{u}^\top \Sigma_D \boldsymbol{u}.$$

We want to find $\boldsymbol{u}$ that maximizes $\sigma_D^{\boldsymbol{u}}$, with the constraint $\|\boldsymbol{u}\|_2 = 1$. Since Σ_D is symmetric its has an orthonormal eigenbasis with eigenvectors $\phi_1, \ldots, \phi_n$ and eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. If we represent $\boldsymbol{u} = \sum_{i=1}^{n} w_i \phi_i$ as linear combination of the eigenvectors with $\sum_i w_i^2 = 1$, we get $\boldsymbol{u}^\top \Sigma_D \boldsymbol{u} = \left(\sum_{i=1}^{n} w_i \phi_i\right)^\top \cdot \sum_{i=1}^{n} w_i \lambda_i \phi_i = \sum_{i=1}^{n} \lambda_i w_i^2$, due to orthonormality of the eigenvectors. This sum is maximal for $w_n = 1$, hence $\sigma_D^{\boldsymbol{u}}$ is maximal if $\boldsymbol{u}$ is the eigenvector corresponding to the largest eigenvalue of Σ_D. Likewise, we can choose the k axes of largest variation as the eigenvectors corresponding to the largest eigenvalues of Σ_D.

2.2.4 RANSAC

Random sample consensus (RANSAC) is a technique to fit parameters to a model given a set of data-points which contain outliers [FB81]. Iteratively, a random subset of the data-points is selected, which contains the minimal amount of points required to estimate the model parameters. These are denoted as *hypothetical inliers*. Next, the optimal parameters to fit the prescribed model to this subset are computed. Then *inliers* from the entire data-set are identified (based on a distance threshold) and the *outliers* are discarded. If the set of inliers (consensus set) is large enough, the model parameters are re-estimated based on the consensus set. Otherwise, a new set of hypothetical inliers is selected and the entire procedure is iterated. This algorithm assumes that the majority of the points are inliers to the sought model and will build a *consensus* on the model parameters, whereas outliers are unlikely to agree on a common model.

2.2.5 Neural Networks

Neural networks (NN) can in theory be universal approximators [Hor91], which makes them a powerful tool to fit functions to any observed data. In general, a *feed forward* neural network model consists of a series of functional transformations [Bis06]. Given a set of input variables $\boldsymbol{x}_0 = \left(x_1, \ldots, x_{k_0}\right) \in \mathbb{R}^{k_0}$, a linear transformation is applied in form of:

$$\hat{\boldsymbol{x}}_{j+1} = \boldsymbol{W}^j \boldsymbol{x}_j + \boldsymbol{w}_0^j \tag{2.2}$$

with *weights* $\boldsymbol{W}^j \in \mathbb{R}^{k_{j+1} \times k_j}$ and *bias* $\boldsymbol{w}_0^j \in \mathbb{R}^{k_{j+1}}$ of the j-th layer of the NN. The output of this linear function $\hat{\boldsymbol{x}}_{j+1}$ is denoted as *activation*. To be able to model non-linear functions, the outputs of these linear units are handed to an *activation function* $\hat{h}(\cdot)$ which are typically (leaky) *ReLU*, *sigmoid* (denoted by $\hat{\sigma}(\cdot)$), or *tanh* functions [Bis06]. The activation function with $\boldsymbol{x}_{j+1} = \hat{h}\left(\hat{\boldsymbol{x}}_{j+1}\right) \in \mathbb{R}^{k_{j+1}}$ is applied elementwise to $\hat{\boldsymbol{x}}_{j+1}$. The outputs of the activation functions are latent (hidden) representations of the input. The network architecture can consist of multiple such layers (*deep* NN) to be able to model increasingly complex functions.

To model such functions the weights and biases are optimized according to an objective function. This is typically referred to as training. During *forward propagation*, the input is transformed by the sequence of layers. To quantify the output, a *loss function* (objective function) is evaluated, which heavily depends on the targeted application. In this thesis we will discuss NNs in the context of supervised classification and descriptor learning. Here, we split the input data into three sets: (1) *training set* which is used to learn the weights of the NN; (2) *validation set* contains data which is not seen by the network during training, but is used to tune hyper-parameters (e.g. stepsize/learning rate of the gradient based optimizer, regularization, model selection etc.) of the NN; and (3) the *test set*, which is used to evaluate the final performance of the NN. Let us assume we are given N_{label} labels with a ground-truth $\boldsymbol{i}_{\text{label}} \in \mathbb{N}^{N_{\text{td}}}$ which contains a value in $\{1, \ldots, N_{\text{label}}\}$ for each of the N_{td} samples of the training data. We would like to

predict a label for an input $\boldsymbol{x}_0$ with known label i_0. Furthermore, we are given a NN with N_{layer} layers with output $\hat{\boldsymbol{y}} = \hat{h}\left(\hat{\boldsymbol{x}}_{N_{\text{layer}}}\right)$. One possibility to interpret the output is by a probabilistic interpretation of $\hat{\boldsymbol{y}}$ where we treat the predictions as likelihood values for the set of labels. By normalization we receive probabilities for the respective labels $\hat{\boldsymbol{y}}_n \in \mathbb{R}^{N_{\text{label}}}$. The loss can then be evaluated based on the predictions, e.g. by the *negative log-likelihood loss*:

$$\mathcal{E}_{\text{loss}}\left(\hat{\boldsymbol{y}}_n, i_0\right) = -\log\left(\hat{\boldsymbol{y}}_n\left[i_0\right]\right),$$

where the notation $[i]$ refers to the i-th entry in the indexed vector. In order to minimize the loss function, *error back propagation* is applied. Back propagation is used to compute the gradients of the loss function with respect to the weight matrices and biases. Stochastic gradient descent or a variant is a typical optimization procedure that is used for this task. This procedure is iterated over several *epochs* (number of times the NN sees the entire data) to improve the networks performance.

The basic idea of NN has been improved and extended in various different ways. Since, we only want to give a basic understanding required for further reading we refer the interested reader to [Bis06, GBC16].

Convolutional Neural Networks (CNN) are specialized for processing data with a grid topology [GBC16] such as images or time series data. Instead of the linear operation (2.2) a convolution is applied in at least one layer. E.g. assuming that the input $\boldsymbol{x}_0$ has a 1D grid topology the convolution is formulated by:

$$\hat{\boldsymbol{x}}_{j+1} = \left(\boldsymbol{x}_j \star \boldsymbol{W}^j\right)(t) = \sum_a \boldsymbol{x}_j[a]\boldsymbol{W}^j[t-a].$$

During training, the weights of the filter (also referred to as *kernel* or *feature map*) $\boldsymbol{W}^j$ are learned, such that they are able to identify specific patterns in the data. The kernels typically have fewer parameters than the fully connected linear layer, which are shared for different parts of the input. Furthermore, they can be applied to input of variable size [GBC16]. Usually, convolution layers are combined

with activation functions as above and *pooling operations* to become invariant to small local changes. E.g. to find maximal responses of the filters, *max pooling* is applied which returns the maximal value within a neighborhood (e.g. 2×2).

In this thesis we regard feature curves, which are formed by a series of points on a surface. We are particularly interested in NNs which can handle sequential data. *Recurrent neural networks* (RNN) are such a family of networks [GBC16]. Here, weights are shared along each entity of the sequence. This way the RNN can be trained to become invariant to shifted, symmetrically flipped, or different length sequences. One problem of RNNs is the *vanishing (or exploding) gradient problem*. If the number of layers is too large, the gradients (which are computed by chain rule) will be very small for the layers closer to the input, as a consequence they do not have a strong influence on the output. In case of sequence data we can think of each instance of the sequence as one layer, only that the weights between the layers are shared. Hence, depending on the sequence length this can have a strong influence.

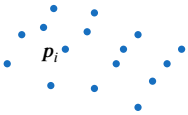
A specialization of RNNs that overcome this problem are the *long-short-term-memory* (LSTM) networks proposed in [HS97]. A *LSTM* cell contains a *memory cell*, an *input gate*, an *output gate* and a *forget gate*, based on which it is possible to steer the output of the LSTM cell by a combination of sums, sigmoidal activations, and products. Since not all of these components are activated by sinusoidal functions, the problem of vanishing gradients has less of an effect. Due to the interaction of the memory cell and the forget gate, it is possible to carry parts of the previous state/input into the next one and thus store information from previous parts of a sequence over a longer period of time. This method has shown to be effective in practice [GBC16].

2.3 Surface Representations

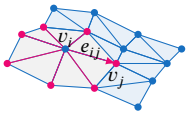
In this thesis we assume that a surface $\mathcal{S}$ is defined as an orientable continuous 2D manifold embedded in $\mathbb{R}^3$. In order to perform computations on $\mathcal{S}$, it is necessary

to discretize the surface by selecting a set of samples from $\mathcal{S}$. Discretizations of a continuous surface can have diverse representations. In this chapter we describe a subset of those representations upon which we can compute, apply, and analyze feature curve networks.

Point Clouds A point cloud $P = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_{|P|}\} \subset \mathbb{R}^3$ is defined as a set of $|P|$ unordered samples taken from the continuous underlying 2D manifold. Point clouds can be obtained by measuring real-world geometry with 3D scanners (e.g. laser- or time of flight scanners). Usually, the detected point-clouds are dense (with several million points) and contain a certain amount of measurement noise. Furthermore, the sample distribution and density heavily depend on the scan position. Surface parts that are not visible from this position are not covered in the scan data. Making this geometric data accessible to downstream applications requires further processing (e.g. denoising and hole-filling techniques).



Triangle Meshes In contrast to point clouds, which have a pure geometric representation, triangle meshes have a geometric and topological component, which specify the connectivity of the point samples [BKP*10]. We define a triangle mesh as a tuple $\boldsymbol{M} = (V, E, F)$ consisting of a set of vertices $V = \{v_1, \dots, v_{|V|}\} \subset \mathbb{Z}$, edges $E = \{e_1, \dots, e_{|E|}\} \subset V \times V$ and triangular faces $F = \{f_1, \dots, f_{|F|}\} \subset V \times V \times V$, which indicate how the vertices are connected to form the surface. Each vertex v_i is associated with its geometric embedding $\boldsymbol{p}_i \in P \subset \mathbb{R}^3$. Edges are given by a tuple of vertices they connect. If an edge has two subindices it is *directed*, this is denoted by the tuple $\boldsymbol{e}_{ij} = (v_i, v_j)$. We denote the geometric embedding of edges (i.e. the respective edge vectors) as $\boldsymbol{e}_{ij} = \boldsymbol{p}_j - \boldsymbol{p}_i$. Vertices in the k-ring neighborhood of a vertex v_i are contained in the set $\mathcal{N}_k(v_i)$. The image above shows the one-ring neighborhood $\mathcal{N}_1(v_i)$ colored in magenta.



There are various methods to obtain a triangle mesh from a point cloud. These methods can be classified into three categories [STJ*17]:

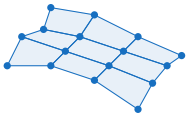
- *High accuracy offline methods* rely on global optimization techniques to recover complex surfaces with millions of points. These methods take points with surface normals as input and approximate a manifold surface. Offline methods can be devided into two main directions.
  Combinatorial algorithms directly triangulate the input surface (e.g. by tetrahedralizing the samples, and extracting the boundary of this volumetric representation, see e.g. [ACK01, HK06]), while techniques based on *implicit functions* interpret scanner points as samples from an (un-)signed distance or indicator function from which the surface is reconstructed (see e.g. [KBH06, BBX95]).

- *Online methods* reconstruct surfaces from point-clouds in real-time (e.g. [NIH*11]). Here the user can interactively move the scanner around an object and leverage the new scan data to the reconstruction system. The object is updated only for the new local data, by using local surface representations (such as truncated signed distance functions). Usually the output quality is lower compared to offline methods.

- *Hybrid techniques* combine benefits from the previous methods to obtain high-quality surface output, while computing the result at interactive rates. [STJ*17] use a multi-resolution hierarchy to represent the surface. Local modifications can be made on the finest level of the hierarchy and are then propagated to the coarse levels. Afterwards, the global update information can be pushed back to the local levels. This technique allows for large-scale changes, while only performing local updates.

The quality of the obtained triangle meshes can be further improved by filling holes [Lie03], removing noise by smoothing the surface [VL08, DMSB99], or by improving the triangles shapes by resampling the geometry with remeshing techniques [BK04, ADVDI03].

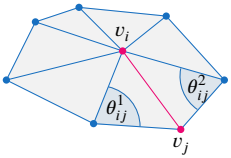
Quadrilateral Meshes Meshes with quadrilateral (quad) faces can have a meaningful orientation, since they are alignable to curvature- or feature-curve orientations. Similar to triangle meshes, quad meshes are defined by a tuple $\boldsymbol{M}_Q = (V, E, F)$, with the difference that $F = \{f_1, \dots, f_{|F|}\} \subset V \times V \times V \times V$, i.e. the faces are quads instead of triangles.



Quad meshes can be computed eighter directly from the input point cloud (e.g. [BL18, STJ*17] or by remeshing a triangle mesh representation of the surface. Popular techniques to compute quad meshes from triangle meshes are based on parametrizing the input surface, i.e. finding a map that projects the input mesh $\boldsymbol{M}$ to $\mathbb{R}^2$. From the 2D domain a grid of integer iso-lines is mapped back to the surface in $\mathbb{R}^3$ (e.g. [ECBK14, BCE*13, EBCK13, BZK09]).

The Laplace Operator and its Spectrum The *Laplace operator* Δ is used for various tasks in geometry processing, such as smoothing, parametrization, or frequency analysis. It is defined as the divergence of the gradient $\Delta = \nabla \cdot \nabla$ [BKP*10]. This operator is generalized to functions defined on surfaces with the *Laplace–Beltrami operator* $\Delta_{\mathcal{S}} = \operatorname{div}_{\mathcal{S}} \nabla_{\mathcal{S}}$. Here, it is essential to define appropriate divergence and gradient operations on the manifold. We can describe a mesh $\boldsymbol{M}$ by the *spectrum* of its Laplace–Beltrami operator up to isometric deformations. The frequency components are given by the eigenbasis of the Laplace–Beltrami matrix $\boldsymbol{L}_M$. Let $\boldsymbol{W}_M$ with $\left(\boldsymbol{W}_M\right)_{ij} = w_{ij}$ denote the weighted adjacency matrix of the mesh vertices. There are several possible choices of weights, depending on desired properties such as symmetry, locality, linear precision, and positivity [WMKG07]. Commonly used weights are the *cotangent weights*, which fulfill the first three properties. They are defined by:

$$w_{ij} = \cot\left(\theta_{ij}^1\right) + \cot\left(\theta_{ij}^2\right).$$

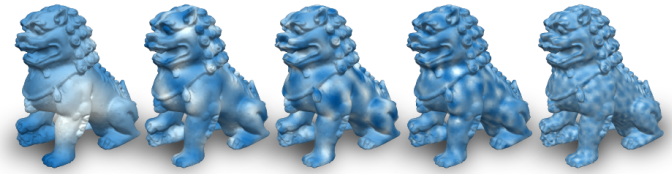


Futhermore, let $\boldsymbol{D}_M = \operatorname{diag}\left(A(v_1), \dots, A(v_n)\right)$ define the *lumped mass matrix*, with area weight $A(\cdot)$,

which measures the size of the dual surface patch at a vertex. Then the disrete Laplace–Beltrami operator is defined by [RBG*09]:

$$\boldsymbol{L}_M := \boldsymbol{D}_M^{-1}\left(\operatorname{diag}\left(\sum_{j\in\mathcal{N}_1(v_1)} w_{1j},\ldots,\sum_{j\in\mathcal{N}_1(v_n)} w_{nj}\right)-\boldsymbol{W}_M\right)$$

The eigenvectors $\boldsymbol{\Phi}_M = \left\{\boldsymbol{\phi}_1^M,\ldots,\boldsymbol{\phi}_k^M\right\}$ of $\boldsymbol{L}_M$ define an orthonormal basis, which represent the frequency components, from which functions can be approximated in the spectral domain. The corresponding eigenvalues $|\lambda_1^M| \le |\lambda_2^M| \le \cdots \le |\lambda_k^M|$ describe the mesh frequencies. In the image below we visualize 5 of the low frequency eigenfunctions. The minimal function value is depicted in white, the maximal value in blue, intermediate color values are interpolated linearly from the eigenfunctions.



As an example, let us assume we want to represent the mesh points in the frequency domain. Given the matrix $\boldsymbol{P} \in \mathbb{R}^{n\times 3}$ containing the stacked vertex embeddings $\boldsymbol{p}_i^\top$. The *spectral coefficients* of $\boldsymbol{P}$ are computed by projection into the eigenbasis $\boldsymbol{\Phi}_M$:

$$\tilde{\boldsymbol{P}} = \boldsymbol{\Phi}_M^\top \cdot \boldsymbol{D}_M \cdot \boldsymbol{P}.$$

We will denote all following mesh-frequency coefficients of a function ℓ by $\tilde{\ell}$. The reconstruction from the *spectral domain* to the *shape space* is performed by:

$$\boldsymbol{P} = \boldsymbol{\Phi}_M \cdot \tilde{\boldsymbol{P}}.$$

Approximate equality ($\approx$) is given in case only a part (e.g. the first k eigenvectors) of the basis is used. Of course, other functions defined over the mesh can be represented in the spectral domain by replacing $\boldsymbol{P}$.

Quality Measures In the following we list some measures that we use to evaluate the quality of the computed meshes:

1. *Hausdorff distance*: Measures the distance between two surfaces. I.e., given two manifold surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$ the Hausdorff distance is defined as:

$$d_{\mathrm{H}}(\mathcal{S}_1, \mathcal{S}_2) = \max\{ \sup_{\boldsymbol{x}\in\mathcal{S}_1} \inf_{\boldsymbol{y}\in\mathcal{S}_2} \|\boldsymbol{x}-\boldsymbol{y}\| , \sup_{\boldsymbol{y}\in\mathcal{S}_2} \inf_{\boldsymbol{x}\in\mathcal{S}_1} \|\boldsymbol{y}-\boldsymbol{x}\| \}. \tag{2.3}$$

2. *Target complexity*: The target complexity can be evaluated by measuring the amount of entities (points, edges, faces) or the resolution (e.g. edge lengths, point densities) of a surface. Measures as the Hausdorff-distance should always be evaluated under regard of the target complexity. Typically, a high accuracy (low Hausdorff-distance) with a low target complexity is desirable.

3. *Triangle Shape*: Usually, isotropic triangles are preferable over pointy needles or caps. Isotropic triangles are computationally more stable and can thus be handled better in downstream applications.

4. *Scaled Jacobian for quad meshes*: In general, the Jacobian can be computed for any type of mesh and indicates the anisotropy. For quad meshes we can use it to measure the deviation of a quad $f = (v_1, v_2, v_3, v_4)$ to a square. First, we compute the area enclosed by the unit (outward-pointing) edge-vectors $\boldsymbol{e}_{ij}$, $\boldsymbol{e}_{ik}$ at each corner v_i of the quad. The scaled Jacobian is the minimum over all corners:

$$A_i = \left\| \boldsymbol{e}_{ij} \times \boldsymbol{e}_{ik} \right\|_2$$
$$SJ_f = \min_{1\le i\le 4} A_i$$

   Ideally, this value should be close to one. Values close to zero indicate strongly anisotropic quads with poor quality [DSSC08, Knu00].

2.4 Estimating Curvature and Curvature Derivatives from Discrete Input

Feature curve detection involves finding extrema in the minimum and maximum curvature fields. Hence, it is essential to accurately compute curvature (second order surface derivatives) and curvature derivatives in discrete settings, e.g. on a mesh or a point cloud. However, computing higher order derivatives is not straightforward due to the pointwise or piecewise linear structure of the surface discretization.

Different strategies have been proposed to accurately estimate the curvature and its derivatives. One line of research fits global or local smooth functions, e.g. implicit [OBS04] or polynomial functions [YBS05] to the geometry. From these smooth surface representations, derivatives can be computed easily. However, higher-order surface fitting can involve computationally expensive preprocessing. If we are given the discretized surface in form of a triangle mesh, it is possible to exploit discrete differential operators to estimate the curvature.

Curvature from Global Implicit Surface Fitting [OBS04, OBS03] present a method to estimate curvature and its derivatives based on implicit surface fitting. An implicit surface is defined by the zero-iso-surface (i.e. $f(x) = 0$) of a function $f : \mathbb{R}^3 \to \mathbb{R}$. As input, [OBS04, OBS03] take a set of point samples, which is cast into a multi-scale hierarchy. On the lowest level the points are approximated by a parallelepiped. This parallelepiped is subdivided into eight octants, which are again devided into an octree to group the points into subsets. The implicit surface is constructed in a bottom up approach. At each level the implicit function $f_k(x)$ is obtained by adding the implicit function from the previous (more coarse) level $f_{k-1}(x)$ to an offset function. The offset approximates the detail added by the next level in the hierarchy. These offsets are obtained via least squares minimization to a quadratic [OBS03] or linear [OBS04] approximation of the regarded point sub-set (i.e. weighted with a radial basis function).
The top level implicit function can now be used to estimate the curvature at a point $\boldsymbol{p}$. First, $\boldsymbol{p}$ is projected onto the implicit surface by moving it into normal direction,

which is obtained by $\boldsymbol{n} = \frac{\nabla f}{\|\nabla f\|}$. The non-zero eigenvectors and eigenvalues of $\nabla \boldsymbol{n}$ approximate $\mathbf{t}_{\max}$ and $\mathbf{t}_{\min}$ and their corresponding curvature values $\kappa_{\max}$ and $\kappa_{\min}$. According to [OBS04], the principal curvature derivatives are given by:

$$\begin{aligned}\kappa' &= \nabla\kappa \cdot \boldsymbol{t} \\ &= \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{l=1}^{3} \frac{\partial^3 f}{\partial \boldsymbol{p}[i]\partial \boldsymbol{p}[j]\partial \boldsymbol{p}[l]} \cdot \boldsymbol{t}[i]\boldsymbol{t}[j]\boldsymbol{t}[l] + 3\kappa \sum_{i=1}^{3}\sum_{j=1}^{3} \frac{\partial^2 f}{\partial \boldsymbol{p}[i]\partial \boldsymbol{p}[j]} \cdot \boldsymbol{t}[i]\boldsymbol{n}[j]\end{aligned}$$

for $\mathbf{t}_{\min}$, $\kappa_{\min}$ and $\mathbf{t}_{\max}$, $\kappa_{\max}$ respectively, where $[i]$ indicates the i-th entry of the given vector.

Curvature from Local Bivariate Polynomial Fitting To achieve higher computational performance, it can be beneficial to locally approximate the geometry by smooth surfaces. E.g. [YBS05] fit local bivariate polynomials to samples from the surface $\mathcal{S}$. Given the surface normal $\boldsymbol{n}$ at a point $\boldsymbol{p}$, the neighboring points of $\boldsymbol{p}$ are transformed into a local coordinate frame centered at $\boldsymbol{p}$ with the positive z-direction corresponding to $\boldsymbol{n}$. The z-coordinates are then expressed as a function in x and y:

$$z(x, y) = \frac{1}{2}\left(w_0 x^2 + 2w_1 xy + w_2 y^2\right) + \frac{1}{6}\left(w_3 x^3 + 3w_4 x^2 y + 3w_5 x y^2 + w_6 y^3\right)$$

The weights $w_i, i \in \{0, \ldots, 6\}$ are computed with the adjacent-normal cubic approximation method [GI04]. Curvature values and derivatives can be computed from this smooth surface representation as discussed above (cf. Chapter 2.1.2).

Curvature from Discrete Differential Operators [HPW05] use discrete differential operators to estimate the surface curvature. First, a discrete version of the shape operator is computed in the following way:

1. Set up the shape operator for each edge. The eigenvectors of the shape operator point into the prinicipal curvature directions. For each edge e we know that $\mathbf{t}_{\min} = \frac{e}{\|e\|}$ with $\kappa_{\min} = 0$. Let $\boldsymbol{n}_e$ be the average normal of the two adjacent faces and θ_e their dihedral angle. Then $\mathbf{t}_{\max} = \mathbf{t}_{\min} \times \boldsymbol{n}_e$ and

$\kappa_{\max}$ is chosen proportional to θ_e, e.g. $w_{\kappa_{\max}} = \|\boldsymbol{e}\|_2 \cos\left(\frac{\theta_e}{2}\right)$ [HPW05]. The shape operator at e can then be defined as:

$$\boldsymbol{S}(e) = w_{\kappa_{\max}} \cdot \mathbf{t}_{\max} \cdot \mathbf{t}_{\max}^{\top}$$

2. Compute the vertex based shape operator by averaging over the adjacent edges (or edges within a certain region $\mathcal{R}$ from the vertex v):

$$\boldsymbol{S}(v) = \sum_{e \in \mathcal{R}} \langle \boldsymbol{n}_v, \boldsymbol{n}_e \rangle \boldsymbol{S}(e)$$

Principal curvatures and curvature directions are obtained from $\boldsymbol{S}$ by computing the eigenvectors corresponding to the largest absolute eigenvalues of $\boldsymbol{S}$.

To compute the curvature derivative [HPW05] interpolate a piecewise linear function based on the vertex-wise values of $\kappa_{\min}$ and $\kappa_{\max}$. The gradients $\nabla\kappa(f)$ of these functions are piecewise constant on each triangle face f. The curvature derivatives at a vertex v are computed as an area-weighted average of adjacent triangles of the terms $\nabla\kappa(f) \cdot \boldsymbol{t}$.

2.5 Feature Curve Networks

Feature curves typically trace out important creases and crests of a 3D surface. Mathematically, they are defined as points where either the maximal principal curvature attains a local maximum or the minimum curvature has a negative minimum [Dem09]. Since this kind of surface abstraction can be used for various algorithms as discussed above, a large body of work on the detection of feature curve networks exists. After introducing a formal definition for feature curve networks, we will present techniques for automatic and interactive feature curve computation.

Definition 2.1 (Feature Curve Network (FCN) [GLK16]) *For a given mesh $\boldsymbol{M}$ we define a **feature curve network** as a 3-tuple $\Gamma = (V, E, C)$ where the geometric embedding of the vertices $v \in V$ lie on the continuous surface defined by $\boldsymbol{M}$ but not necessarily at a vertex position of $\boldsymbol{M}$. According to their valence in*

Γ*, we split the set of vertices* V *into two disjoint subsets* $\bar{V}$ *and* V^* *that contain the regular vertices (valence* $= 2$*) and the extraordinary vertices (valence* $\neq 2$*) respectively.* E *is the set of edges, which connect the vertices in the feature network. The set* $\mathbf{C}$ *consists of (mutually disjoint) curves or arcs. An arc* $c \in \mathbf{C}$ *is a sequence of feature edges that either connects two extraordinary vertices or forms a closed loop.*

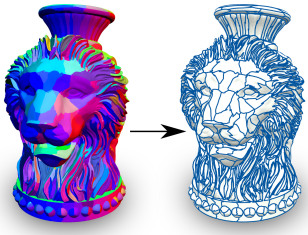
2.5.1 Computing Feature Curve Networks

There exists a large body of work on the detection of feature curve networks. Here, we will mainly focus on *view-independent* algorithms. *View-dependent* methods extract aesthetic curves based on a specific viewing direction (see e.g. [KST09, KST08, ZHX*11]), which is less useful in view-independant downstream applications as presented here. *View-independent* techniques can be divided into two main categories: *fully automatic* and *interactive* methods. Fully automatic feature curve detection algorithms are based on local surface properties such as the surface curvature or normal deviation. In contrast, interactive techniques take input from the user, who indicates relevant curves. The system usually tries to align the hints of the user to the curvature maxima/minima. While in the first case, the detected curves are not necessarily meaningful, curves selected by a user typically are, and do not always require further processing. We will cover examples from both categories in the following sections.

2.5.1.1 Fully Automatic Techniques

Typically, FCNs are computed on triangle meshes, since the topological information given allows for more accurate curvature estimation and line tracing. Such surface based techniques can be roughly divided into four categories: (1) patch based FCNs, (2) curve tracing based FCNs, (3) patches computed from input curves, and (4) FCNs based on slippage analysis. Furthermore, it is also possible to compute curve networks from point clouds directly. These techniques have some additional challenges due to the lack of topological information. In the following, we will give examples for each of the categories.

Patch based FCNs Patch based techniques divide the points on the surface into disjoint sets, the patches. This partitioning is usually performed by assigning the faces of the corresponding mesh M to one of the patches. Typically, it is preferable if a patch only has one connected boundary and contains no islands. The patch-boundaries represent the feature curves. E.g. the image above (left) shows patches computed with [CSAD04] and the respective patch-boundary feature curves (right).



Several methods have been presented to compute surface partitions, from which we can extract feature curve networks:

Variational shape approximation (VSA) [CSAD04] is an example for a method based on primitive fitting. It iteratively finds a surface partitioning which minimizes the normal variance in a surface patch. For each patch a *proxy* is stored, which represents the normal of the respective region. They employ *Lloyd's* algorithm to update the patches:

1. *Initialization:* First k random faces $\{f_1, \dots, f_k\}$ are selected with normals $\{\boldsymbol{n}_1, \dots, \boldsymbol{n}_k\}$ on the input mesh $\boldsymbol{M}$. These normals are the initial proxies.

2. *Distortion minimizing flooding:* For each patch the face with the normal closest to the proxy is selected as seed. Then the regions are iteratively grown by adding adjacent faces with the lowest distance between the normals. The growing process guarantees connected, non-overlapping patches.

3. *Update proxy:* The updated proxy is computed as the smallest eigenvector of the covariance matrix of the points in the patch-region. With the new proxy we go back to step (2) and iterate.

Other primitives that can be used for fitting have been suggested, e.g. CAD primitives [WK05] or parametrized shapes [VSHJ01]. While these methods are straight forward and easy to implement, obviously, the extracted patch boundaries are highly dependent on the selected primitive. E.g. in the case of VSA (with normals as primitives) we require a lot of patches to capture all the details as in the example shown above.

A different line of research is based on *discrete Morse theory* [For98]. The patches are built from a topological skeleton of a scalar function $f : \mathcal{S} \rightarrow \mathbb{R}$ defined over the surface. Three different kinds of critical points can be detected in the gradient ∇f: (1) minima, (2) maxima, and (3) saddle points. For each saddle point there exist four outgoing tangent curves, which connect to two maxima and two minima. These curves are denoted *separatrices* and the resulting graph structure yields the *topological skeleton* [WG09]. To compute the topological skeleton, a scalar input function is required. E.g. [WG09] use $\kappa_{\max}$ and $\kappa_{\min}$ and combine both graph structures to a FCN. [SWPL08] measure the *curvedness* $f(\boldsymbol{p}) = 0.5(\kappa^2_{\max}(\boldsymbol{p}) + \kappa^2_{\min}(\boldsymbol{p}))^{\frac{1}{2}}$. After computing the initial skeleton, some of the separatrices are filtered out. This can be based on the persistence, i.e. the locally smallest deviation of a saddle point to its adjacent extrema [ELZ00, CSEH05, EHZ01]. Since removal based on local persistence removes entire separatrices, it does not take into account that the strength of a feature may vary along a curve. Thus, follow-up work use continiouous measures to evaluate the relevance of a curve [SWPL08, WG09].

Patch based techniques rather capture the global structure of the surface than measuring small fluctuations, e.g. by finding a globally optimal primitive fitting or computing a consistent topological skeleton of the object. While these methods are often more robust to noise and avoid computing derivatives of the curvature, they neglect weaker features in the global structure. However, weak features can also be meaningful. Methods that trace ridges and valleys can detect such weak features, since they are based on local measurements.

FCNs based on Ridges and Ravines FCNs that are composed from tracing *ridges* and *ravines* capture both salient shape features as well as small shape variations [OBS04]. Below an example of feature curves computed with [YBS05] is visualized. The blue curves indicate the ridges, while the magenta curves show the ravines. We start by giving a mathematical definition of rigdes and ravines according to [BPK98]:

Definition 2.2 (Ridges and Ravines) *A non-umbilic point $\boldsymbol{p} \in \mathcal{S}$ is called a **ridge point** if $\kappa_{\max}$ attains a local positive maximum at $\boldsymbol{p}$ along the associated curvature line. A non-umbilic point $\boldsymbol{p} \in \mathcal{S}$ is called a **ravine point** if $\kappa_{\min}$ attains a local negative minimum at $\boldsymbol{p}$ along the associated curvature line.*

[BPK98, OBS04, HPW05, YBS05] further characterize these extrema as zero crossings of κ':

Theorem 2.1 *Ridge points are characterized by the following conditions*

$$\kappa'_{\max} = 0, \quad \frac{\partial \kappa'_{\max}}{\partial \mathbf{t}_{\max}} < 0, \quad \kappa_{\max} > |\kappa_{\min}|.$$

Likewise, ravine points are characterized by the conditions

$$\kappa'_{\min} = 0, \quad \frac{\partial \kappa'_{\min}}{\partial \mathbf{t}_{\min}} > 0, \quad \kappa_{\min} < -|\kappa_{\max}|.$$

Typically, methods that trace crest lines follow the scheme described below (see e.g. [YBS05, HPW05, OBS04]):

1. *Estimate principal curvatures, prinicipal curvature directions, and extremality coefficients:* For this [OBS04] exploit implicit surface fitting, [YBS05] use local polynomial fitting, and [HPW05] make use of discrete differential operators.

2. *Establish consistent orientation of principal curvature directions:* For non-umbilic points the prinicipal curvature directions are unique up to 180° flips. Obtuse angles of principal curvature directions between adjacent vertices in a 1-ring neighborhood are avoided by performing consistent flips [OBS04].

3. *Compute curvature extrema:* Zero crossings in κ' can be found by detecting sign changes at the vertices of edges [YBS05, OBS04], approximating the sign of the derivatives of the extremality coefficients and finding the extrema by linear interpolation along the line. Alternatively, in case the extremality coefficients are given as piecewise linear functions as in [HPW05], the conditions in Theorem 2.1 can be evaluated directly.

4. *Connect points to obtain feature lines:* In case we are only given feature points, instead of line segments these need to be connected to feature arcs. There are three possible cases how to connect the curves: In the *regular case* two edges of a triangle face contain a feature point. Then these points are connected by a line. In the *trisector case* all three edges of a triangle have a feature point, these are connected to the barycenter of the face. No further processing is relevant for the *start/end case*, where only one edge contains a feature point.

5. *Postprocessing:* To enhance the quality of the feature curves, the authors present several postprocessing techniques:

    - *Thresholding:* A typical method to reduce the amount of noise in a feature curve network is to filter out weak features according to a predefined threshold. Different measures have been proposed to threshold the feature curves. E.g. [OBS04, HPW05] compute the integral over the principal curvature along the curve, while [YBS05] present a threshold based on the cycledness along the curve.
    - *Smoothing:* To reduce the surface noise, a standard technique is to smooth the surface in advance to processing. It is also possible to smooth extremality coefficients or the obtained feature curves.

- *Establish local connectivity:* [YBS05] avoid strong fragmentation of the feature curves by locally reconnecting the segments that roughly follow the same direction.

FCNs that derive from tracing ridges and ravines capture small surface fluctuations, weak features, as well as salient surface curves. However, these FCNs do not contain a boundary representation (B-rep) as the patch based methods do, which is required for applications like surface segmentation, parametrization, or curve based modeling.

Patches from Tracing To exploit the benefits from both techniques, a line of research focuses on the computation of FCNs which transform traced creases and crests to patch layouts. These techniques take disconnected feature curves as input. From these initial features regions are detected.

[LPRM02] first grows regions dual to the feature curves, which divides the surface into patches containing the respective curve. To include the feature curves into the patch layout, each region is cut by its feature curve by tracing the steepest-descent path until the boundary is reached.

[CYW15] extend the existing curves by evaluating a cost function on the vertices adjacent to the end points of the feature curves, which measures the curvature, continuity of principal directions, and smoothness. They iteratively elongate the arcs by selecting the extension with minimum cost until a predefined cost threshold is reached. The remaining gaps are closed by continuing the crest lines as straight as possible until they intersect another curve. This technique is more sensitive to the mesh quality than the previous method, since it strongly depends on local surface properties for the curve tracing.

[GSV*17] present a method which produces curve networks similar to what an artist would create to easily perceive the shape. The curve networks not only contain high-curvature lines, but also orthogonal strands which convey the shape of the object. Gori et al. show that these networks can be used to reconstruct the 3D surface. The algorithm is based on computing flowlines (paths of vertex adjacent edges) on a quad-dominant mesh. First, a set of flowlines is computed by

clustering mesh edges. Then, an initial curve network is computed with [YBS05], which is used to partition the input surface. Next, flowlines are added until all regions comply with a descriptiveness threshold. As a final step, superfluous flowlines with high cost are removed from the network.

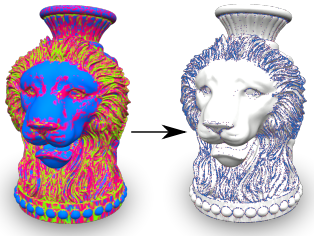
The resulting FCNs of the presented techniques have very different properties, and it is application dependant which method is best suited. E.g. [LPRM02] targets surface parametrization, while [GSV*17] can be used for surface reconstruction.

FCNs from Slippable Motions Instead of regarding feature curves as curvature extrema it is also possible to view them in terms of slippable motions in a self-alignment problem.

Definition 2.3 (Slippable Motions, [GG04]) *A **rigid motion** consists of 2 time dependent components: a rotation $\rho = (\rho_x, \rho_y, \rho_z)$ indicating the rotations around the respective axis, and a translation $\tau \in \mathbb{R}^3$. The velocity at a given time step at $\boldsymbol{p}$ is computed by:*

$$\vec{\boldsymbol{v}}(\boldsymbol{p}) = \rho \times \boldsymbol{p} + \tau.$$

*A rigid motion is called a **slippable motion** of a surface $\mathcal{S}$ if the velocity vector is tangent to $\mathcal{S}$ at each point of $\mathcal{S}$.*



Feature curves are composed of points, which are 1-slippable, i.e. where the surface can locally be slided against itself into only one direction. E.g. in the image on the right (left) we visualize 1-slippable (green), 2-slippable (pink), and 3-slippable (blue) motion. To compute slippable directions we want to find a rigid transformation that has minimal motion into the

normal direction and moves tangent to the surface if we apply it to a point $\boldsymbol{p}$. This can be measured by the following functional:

$$\min_{\rho,\tau} \sum_{i=1}^{n} ((\rho \times \boldsymbol{p} + \tau) \cdot \boldsymbol{n})^2,$$

which is minimal for motions orthogonal to the surface normal. The minimum of this linear least squares problem is the null space of its derivative [GG04].

[BBW*09] detect feature lines from the slippage information. First, they identify points which are 1-slippable by translation in a small local neighborhood. Since points in the vicinity can have a similar slippage direction, this results in regions of 1-slippalbe points. To transform these into lines, the points with maximum mean curvature in the direction orthogonal to the slippage vector are detected (see figure above (right)).

As the patch based methods, this technique does not rely on curvature derivatives and is thus more robust to noise. However, to apply this technique several heuristics and thresholds need to be set (e.g. which points of the curve to connect or when a motion is only 1-slippable) which can be challenging if they are set for the entire mesh.

FCNs from Point Clouds The detection of feature curve networks in point clouds is a challenging and ongoing research topic. Due to the lack of topological information, the normals and curvature values need to be approximated based on local neighborhoods (e.g. k-nearest neighbors or an ϵ-region). Selecting the correct neighborhood size is not straight forward. If it is too small, the detected features can be too dense and noisy, while if the neighborhood radius is chosen too large, important features may be smoothed out. Furthermore, assumptions have to be made about the structure of the curve, e.g. that all points lie on a nearly straight line, so that the feature curve topology can be established.

First, feature curve candidates need to be detected. This can be done via principal component analysis [GWM01, PKG03] of the local neighborhood of the points, RANSAC based plane fitting [NLNZ16], or slippage analysis [BBW*09].

Next, the connectivity of the feature curve segment is established. Often, the choice when to connect feature points is based on local decisions and heuristics. Possible criteria to connect feature points are proximity, density, and smoothness of the feature direction [NLNZ16].

Of course, feature curves in real-world geometry do not necessarily follow straight lines, and complex structures will not be covered well by such heuristics. The resulting curves can be fragmented and details may be missing. Hence, especially curves generated from noisy point cloud data require further processing to extract a globally meaningful network.

2.5.1.2 Interactive Techniques

Automatically computed FCNs can have several flaws. The curves can be fragmented or disconnected. There may be false positive curves, i.e. points that are classifed as features at high frequency details that exist due to noise. Also, false negatives can occur when features are too weak, depending on the thresholds used in the respective detection algorithm. Furthermore, curves can be jagged in case of surface noise. These defects can be avoided by interactive techniques, where a user indicates the relevant curves. However, there is a tradeoff between controlability and ease of use [ZZCJ14]. Requiring a user to draw the entire curve can be very tedious. Hence, interactive systems usually ask for partial input in form of a *click-and-connect* interface, where some points along the curve are selected by the user and the intermediate parts are found automatically.

[LL02] present geometric *snakes* (snakes are also known as *active contour models*). The user inputs an initial curve/snake according to the click-and-connect paradigm. Then, an energy is minimized, which is composed of two terms. The first term maintains the smoothness of the curve by analyzing the curves' derivatives, which supports that the curve locally behaves like a spline. The second term draws the snake-points closer to points with high normal deviation to neighboring faces, i.e. to the sharp features of the mesh. The optimization process is performed by iterative gradient descent in the parameter domain (so that no re-projection to

the surface needs to be applied during the optimization).

A more interactive interface was presented with the *live-wires* in [ZZCJ14]. Here, a start-point is selected by a click, then the curve is updated automatically by moving the position of the cursor. Since the extent of the entire curve is not known during the interactive process, a parametrization based approach would be computationally intensive (it would be required to constantly update the local paramter domain). Instead, the intermediate points are found by shortest path search on anisotropic geodesics [CHK13, ZZCJ14] (cf. Chapter 2.1.3). In order for the geodesics to align with high curvature regions of the mesh, the tensor which is required to compute the anisotropic norm needs to be chosen, such that the eigenvectors align with the principal curvature directions and the (largest) eigenvalues λ_1 and λ_2, $\lambda_1 \geq \lambda_2$ with the curvature strength (e.g. the shape operator $\boldsymbol{S}$), since

$$\|\boldsymbol{v}\|_{m,p} = \sqrt{\boldsymbol{v}^\top \boldsymbol{\mathcal{T}}_m(p)\boldsymbol{v}} = \|\boldsymbol{v}\| \sqrt{\lambda_1 \cos(\theta)^2 + \lambda_2 \sin(\theta)^2},$$

where θ denotes the angle between the vector $\boldsymbol{v}$ and the eigenvector corresponding to λ_1. Accordingly, geodesics will tend to align with the the eigenvector corresponding to λ_2. The computation based on geodesic shortest paths is especially useful, since efficient algorithms exist to compute these at interactive rates.

If an entire curve network needs to be selected by a user, a sparser input can be beneficial. E.g. in [GBKS18] the user selects only the start and end point of the curves. Again, the intermediate part of the curve is found by the shortest path on a user-defined metric, as for example isotropic and anisotropic geodesics.

2.6 Correspondences and Symmetry Detection

In contrast to typical low-level geometry processing tasks, where raw geometry representations are viewed in isolation, for shape analysis we consider 3D geometry in relation to other data (e.g. shape collections, points or segments of the

shape), since the entire information is used to infer knowledge about the given data-set. Hence, an essential task in 3D shape analysis is the identification of correspondences within the regarded data-set.

On the lowest level we can identify corresponding points via point signatures, which locally describe a point on the surface (cf. Chapter 2.6.1). Based on these descriptors, methods were developed, which compute larger consistently corresponding patches within the same surface by symmetry detection (cf. Chapter 2.6.2) and across different surfaces (cf. Chapter 2.6.3). To make feature curves accessible to shape analysis tasks, we will show how to find correspondences among feature curves in the same shape and across different shapes. First, we will introduce the basic foundations from point-based correspondence and symmetry detection, which we present in the following.

2.6.1 Point Signatures

In this chapter we will only discuss point signatures/descriptors which are functions $f : M \to \mathbb{R}^n$, that assign points on the surface of the mesh M to n-dimensional real-valued vectors. Point signatures describe local or global surface properties, e.g. the relation or distance to other points on the shape. Similarity between the points is then evaluated by comparing these descriptors based on a distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

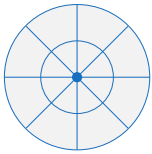
An essential component involved in the design of such a similarity measure is to achieve invariance to a specific class of deformations $\mathcal{D}$, such that $f(\mathcal{D}\boldsymbol{p}) = f(\boldsymbol{p})$ [RPC10]. Here, we will mainly discuss two of these categories: those that are invariant against *rigid motion* and signatures that are invariant to *isometric* deformations. The set of rigid transformations includes rotations, translations, and reflections. Intuitively, isometric deformations are articulated motions that can be performed on a model without stretching the surface. Mathematically, a mapping $T : \mathcal{S}_1 \to \mathcal{S}_2$ is called isometric if it preserves distances between two points under T [BBK06]. Our surface is a Riemannian manifold where distances are measured by geodesics. Hence, descriptors that are invariant to isometric deformations are typically based on the geodesic distances.

There is a large body of work on point signatures. Hence, here we can only discuss a few representatives, which are commonly used in literature.

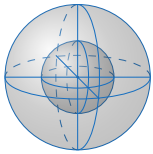
2.6.1.1 Signatures Invariant to Rigid Motion

Rigid motions are a subset of isometric transformations. However, since these descriptors are not required to preserve an intrinsic measure, there is more flexibility in the descriptor design. This is especially useful if the signature is tailored to a specific task. In the following we will name a few signatures which are invariant to rigid motion, each with different application areas:

The *shape diameter function* $\ell_{\mathrm{SD}} : \boldsymbol{M} \to \mathbb{R}$ presented in [SSCO08] estimates the diameter at a point $\boldsymbol{p}$ on $\boldsymbol{M}$. In the discrete setting, a cone is constructed at each vertex v_i of the respective mesh $\boldsymbol{M}$, which has its opening angle in the direction of the inward pointing normal at $\boldsymbol{p}_i$. Subsequently, directions are sampled in this cone, which are used to shoot rays and compute the intersection of this ray with the surface. The distance between $\boldsymbol{p}_i$ and the intersection point indicates the diameter. The final diameter estimate is computed by averaging over the selected sample directions. Close surface points are likely to have a similar diameter ℓ_{SD}. Hence, this signature is often used for rough segmentations of objects rather than identifying small scale details on the shape.



To describe small scale variations, methods that were inspired by image-based techniques, such as the well known *SIFT*-descriptor [Low99], have been proposed. These are based on local histograms. E.g. the *shape context* presented in [BMP01] with extensions proposed in e.g. [KPNK03, KBLB12] computes 2D local radial grids around each point $\boldsymbol{p}$ containing the histograms (see image above). The vectors pointing to the other points within the radial grid are computed and stored in the histogram of the respective grid cell.



A similar idea was presented with the *Signature of Histograms of OrienTations (SHOT)* [TSDS10]. Here, the grid is created by subdividing a 3D sphere. Instead of relative positions, the normal variations to the normal at $\boldsymbol{p}$ are stored in the histogram bins. Local surface details are captured by the normal deviation of the points. By measuring differences in local orientations, points that vary by small scale surface details can be disambiguated. Surface descriptors that capture this amount of detail are often required in tasks such as consistent inter-surface mapping.

In contrast to the previous methods, *spin images* [JH99] describe a more global view of the object. Given a point $\boldsymbol{p}$ that is described by the spin image and its normal $\boldsymbol{n}$, a plane of width r is spanned from $\boldsymbol{p}$ in normal direction and into some tangent direction at $\boldsymbol{p}$. This plane is subdivided into bins. Subsequently, it is rotated around the normal and all points that are intersected by a bin of the plane are accumulated. Hence, all points within the cylinder erected along the normal of radius r are captured in this histogram. The width r of the spin image impacts the locality of the descriptor. In case the spin image captures enough global information of the geometry, it can be used for tasks such as retrieval (e.g. [DBP05]).

2.6.1.2 Isometry Invariant Signatures

To compare points on articulated shapes, it is beneficial if the signature is invariant to isometric deformations. As described above, isometric transformations preserve geodesic distances on the manifold $\mathcal{S}$. Thus, an obvious choice is to base the signature design on the geodesics of the surface.

The *histogram of geodesics* (HOG) [RPC10] $f_{\text{HOG}} : \boldsymbol{M} \rightarrow \mathbb{R}^k$ measures the distribution of geodesic distances $f_{\text{HOG}}(\boldsymbol{p})$ from a surface point $\boldsymbol{p}$. From $\boldsymbol{p}$ the geodesic distances to a set of samples are computed. These distances are stored in a histogram which is divided into k distance bins. A variant of this method was

presented in [TBW*11]. Here, the lengths of geodesic iso-curves are measured for a number of k distance samples in a small local radius. Descriptors based on geodesic distances can be very accurate (depending on the number of samples). However, the calculation of geodesic distances is computationally heavy, which makes the descriptor computation quite costly for a large number of distance samples.

To overcome this issue, several descriptors based on the spectrum of the Laplace–Beltrami matrix have been proposed, which itself is intrinsic and allows to approximate the geodesic distances by the distribution of heat or the motion of particles on the surface.

The *heat kernel siganture* (HKS) [SOG09] and the *heat kernel map* (HKM) [OMMG10] measure the distribution of heat on the surface. They are based on the heat equation:

$$\Delta_{\mathcal{S}} u(\boldsymbol{p}, t) = -\frac{\partial u(\boldsymbol{p}, t)}{\partial t},$$

which describes the heat diffusion process over $\mathcal{S}$. Let $f_{\text{heat}} : \mathcal{S} \to \mathbb{R}$ denote an initial heat distribution and $\mathcal{H}_t \cdot f_{\text{heat}}$ the heat distribution at time t. It can be shown that $\mathcal{H}_t \cdot f_{\text{heat}} = \exp(-t\Delta_{\mathcal{S}}) \cdot f_{\text{heat}}$ satisfies the heat equation for all t and that $\mathcal{H}_t$ has the same eigenfunctions $\phi_1, \ldots, \phi_k$ as $\Delta_{\mathcal{S}}$ with eigenvalues $\exp(-\lambda_1)t, \ldots, \exp(-\lambda_k)t$ [SOG09]. If we select a *Dirac-delta-function*

$$\delta_{\boldsymbol{p}_1}(\boldsymbol{p}_2) = \begin{cases} 1 \text{ if } \boldsymbol{p}_2 = \boldsymbol{p}_1 \\ 0 \text{ else} \end{cases}$$

as initial heat distribution, then a single point $\boldsymbol{p}_1$ emits heat. The heat diffusion from a single heat source is also denoted as *heat kernel*:

$$f_{\text{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2) = \mathcal{H}_t \cdot \delta_{\boldsymbol{p}_1}(\boldsymbol{p}_2)$$

Let us compute $\delta_{\boldsymbol{p}_1}$ as a reconstruction from the spectral domain:

$$\delta_{\boldsymbol{p}_1} = \sum_{i=1}^{\infty} \langle \delta_{\boldsymbol{p}_1}, \phi_i \rangle \cdot \phi_i = \sum_{i=0}^{\infty} \phi_i(\boldsymbol{p}_1) \cdot \phi_i.$$

Then the heat kernel can be expressed as:

$$\begin{aligned} f_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2) &= \mathscr{H}_t \cdot \delta_{\boldsymbol{p}_1}\left(\boldsymbol{p}_2\right) \\ &= \left(\mathscr{H}_t \cdot \sum_{i=0}^{\infty} \phi_i(\boldsymbol{p}_1) \cdot \phi_i\right)\left(\boldsymbol{p}_2\right) = \left(\sum_{i=0}^{\infty} \phi_i(\boldsymbol{p}_1) \cdot \mathscr{H}_t \cdot \phi_i\right)\left(\boldsymbol{p}_2\right) \\ &= \left(\sum_{i=0}^{\infty} \exp(-\lambda_i t)\phi_i(\boldsymbol{p}_1) \cdot \phi_i\right)\left(\boldsymbol{p}_2\right). \end{aligned} \tag{2.4}$$

It is sufficient to compute the first k Laplace–Beltrami eigenfunctions and eigenvalues to estimate the heat kernel, since high frequency oscillations usually transport a certain amount of noise. Intuitively, the distribution of heat from a single heat source approximates the geodesic distances from that point [CWW13]. In contrast to geodesic distance computations, efficient algorithms exist to compute the first k eigenfunctions of the mesh Laplacian. However, in this case numerical issues can arise. E.g. when the mesh quality is poor, the eigenvector computation is less stable. The HKM describes a point $\boldsymbol{p}$ by the heat distribution at q time samples, i.e.:

$$f_{\mathrm{HKM}}(\boldsymbol{p}) = \begin{pmatrix} f_{\mathrm{HK}_{t_1}}(\boldsymbol{p}, \cdot) \\ \vdots \\ f_{\mathrm{HK}_{t_q}}(\boldsymbol{p}, \cdot) \end{pmatrix}$$

In the discrete setting $f_{\mathrm{HKM}}(\boldsymbol{p}) \in \mathbb{R}^{q \times |V|}$. Unfortunately, this descriptor can only be used to compare discrete mesh vertices from different shapes if they have the same number of vertices. To avoid this issue, the HKS computes the amount of heat that is distributed from a point to itself within a certain amount of time:

$$f_{\mathrm{HKS}}(\boldsymbol{p}) = \begin{pmatrix} f_{\mathrm{HK}_{t_1}}(\boldsymbol{p}, \boldsymbol{p}) \\ \vdots \\ f_{\mathrm{HK}_{t_q}}(\boldsymbol{p}, \boldsymbol{p}) \end{pmatrix}.$$

This has the benefit that $f_{\mathrm{HKS}}(\boldsymbol{p}) \in \mathbb{R}^q$ is independent of the mesh tessellation. Furthermore, according to [SOG09] important properties such as symmetry, isometric invariance, informativeness, and stability hold for both the HKM as well

as the HKS.

A variant of the HKS was proposed by [ASC11] with the *wave kernel signature* (WKS). Here, the motion of a quantum mechanical particle along the surface is measured by the *Schrödinger* equation:

$$\frac{\partial u(\boldsymbol{p},t)}{\partial t} = \imath \Delta_{\mathcal{S}} u(\boldsymbol{p},t),$$

where $\imath$ is the imaginary unit, and u is a wave function along which the particle moves. While the HKS averages over eigenfunctions with different frequencies, the WKS allows for more precise control by separating the influence of different frequencies. Especially, high frequency information is represented more accurately, which allows for a more precise descriptor matching. Similar to the heat kernel (cf. Equation 2.4), the wave function, which satisfies the Schrödinger equation can be derived. Let f_μ^2 denote the energy probability distribution of a quantum particle with expectation value μ, then its wave function is given by:

$$f_{\mathrm{WK}_\mu}(\boldsymbol{p},t) = \sum_{k=0}^{\infty} \exp(\imath \lambda_k t) \cdot \phi_k(\boldsymbol{p}) \cdot f_\mu\left(\lambda_k\right). \tag{2.5}$$

The probability that a particle is located at a point $\boldsymbol{p}$ at time t can then be computed by $|f_{\mathrm{WK}_\mu}(\boldsymbol{p},t)|^2$. The wave kernel signature is defined as an average probability over time:

$$f_{\mathrm{WKS}}(\mu,\boldsymbol{p}) = \lim_{t_0\to\infty} \frac{1}{t_0} \int_0^{t_0} |f_{\mathrm{WK}_\mu}(\boldsymbol{p},t)|^2 = \sum_{k=0}^{\infty} \phi_k(\boldsymbol{p})^2 f_\mu\left(\lambda_k\right)^2.$$

This formulation is independent of the time component. Instead the WKS can be computed for different energy values, which relate to the oscillation frequencies [ASC11]. This separation of frequencies allows for more accurate descriptor comparison as discussed above.

The descriptors mentioned so far all take the relation to other points on the shape into account, e.g. by describing a diffusion process or measuring the distances to other points. [MOR*18] take this approach one step further with the *discrete*

time evolution process descriptor (DEP). The DEP measures the propagation of information on a shape. For this an initial distribution $\ell_0 \in \mathbb{R}^{|V|}$ is specified at the mesh vertices. Furthermore, a relation function $\boldsymbol{R} \in \mathbb{R}^{|V|\times|V|}$ is required which describes the information transfer between surface points (e.g. geodesic distances, kernels, or diffusion distances). At each time step of the evolution process, the relation function is applied to the current distribution (with a certain amount of regularization), thus describing higher order relations. In contrast, the previous descriptors describe one step of such an evolution process. However, it is possible to formulate previous methods in terms of the DEP and apply several steps of the time discrete evolution, leading to increased correspondence accuracy.

2.6.1.3 Topological Signatures

Most point signatures describe the local geometry at a surface point, but do not capture the global connectivity structure of the shape [COO15]. Such global topological information can be used to distinguish locally similar points.

E.g. [COO15] describes a topological signature for points on 3D shapes. The global surface connectivity structure, as seen from a point on the surface, is captured by a *persistence diagram* (PD). PDs are a tool of topological data analysis and can be used to describe the lifespan of a topological feature. The topological features used in [COO15] are holes in geodesic balls. To measure these, geodesic balls are grown around a surface point $\boldsymbol{p}$ and intersected with the surface. If the number of holes in the intersected surface changes, the respective radius is stored in the PD. The x-axis of the PD indicates at which radius a new hole is created, the y-axis when it disappears. [COO15] show that augmenting local descriptors with this topological information improves matching and supervised shape labeling results.

In [GBK16] we present the *Geodesic Iso-Curve Signature* (GICS), which by design captures both local surface properties and topological events. For this we extend the method presented in [TBW*11]. Given a manifold surface mesh $\boldsymbol{M}$,

the geodesic iso-curve signature is based on the computation of lengths $l_p(r_i)$ of isocontours around a surface point $\boldsymbol{p}$ on $\boldsymbol{M}$, at n different geodesic radii r_i over the entire surface mesh $\boldsymbol{M}$:

$$\not{f}_{\text{GICS}}(\boldsymbol{p}) = [l_p(r_1), l_p(r_2), \dots, l_p(r_n)] \in \mathbb{R}^n$$

with radii

$$r_1 \leq r_2 \leq \dots \leq r_n, r_i \in \mathbb{R}.$$

The function $l_p(r_i)$ maps the radius r_i to the length of the corresponding iso-curve:

$$l_{p \in M} : [0, d_{\max}(\boldsymbol{p})] \rightarrow \mathbb{R},$$

where $d_{\max}(\boldsymbol{p}_1) = \max\{d_M(\boldsymbol{p}_1, \boldsymbol{p}_2) | \forall \boldsymbol{p}_2 \text{ on } \boldsymbol{M}\}$ denotes the maximal geodesic distance on $\boldsymbol{M}$ from a surface point $\boldsymbol{p}_1$. Upon topological events, geodesic isocontours split, merge, or parts of the isocontour vanish (when a geometric feature ends). When the isocontours that grow smoothly around a surface point meet, they are merged, or respectively split when they grow around separate topological features. In many cases, such topological events cause a sign change in the gradient of the signature function, such that they can be detected as peaks. Furthermore, when the radius exceeds the length of a geometrical feature (e.g. the fingers of a human), the length of the radius decreases and drops to zero for the respective geometric part. Hence, these events induce local maxima in the local continuous length-function of the isocontours for prominent topological features. This way, characteristic topological changes of a shape are captured. Furthermore, we allow for precise sampling of the distance measures, thereby controlling the influence of local surface properties vs. global relations. In [GBK16] we show that the additional topological information strongly increases point-matching performance compared to state of the art point signatures as the WKS or HOG. We will give more details on the computation of this signature in Chapter 3, where we will show its extension to feature curves.

2.6.1.4 Learned Signatures

While previous methods apply for general shape matching applications, often in a task specific setting it can be beneficial to learn signatures or correspondences from examples. In this case, we are typically given a training set with target labels or correspondences, based on which it is possible to learn descriptors optimal for the given setting. To learn signatures, it is relevant to identify weights or parameters that can be learned, set up an objective function/loss, and minimize this objective. In the following we discuss a set of representative techniques which are able to learn task specific descriptors.

[WVR*14] present the *optimal intrinsic descriptor*. Windheuser et al. observe that the matching accuracy of the HKS and WKS heavily depend on the chosen parameters (i.e. time or energy samples). Hence they propose a method to learn the optimal parameters. For this they find a formulation such that the optimization procedure can choose from an infinite set of parameter values. For a set of known corresponding points, a siamese energy function is introduced, which optimizes for descriptors with low distance values for corresponding points and high distances in case of negatives.

Litman et al. further observe that the HKS (which acts as a low-pass filter) has low sensitivity and poor localiztation. In contrast, the WKS (which behaves like a band-pass filter) maps to many unrelated matching regions (low specificity) but is well localized and thus has a high sensitivity [LB14]. Hence, they propose the *optimal spectral descriptors* as a generalization, where the frequency responses are learned from examples for a specific task. As in [WVR*14] an energy is constructed which minimizes the intra-class- and maximizes the inter-class distance. They show how this can be formulated as a closed form solution.

Previous methods are based on the spectral representation of the mesh. The following methods can take arbitrary representations as input and are thus more flexible.

[MBBV15] present a generalization of CNNs to non-Euclidean manifolds. They introduce a geodesic convolution with a filter, which corresponds to a geodesic disc on the surface. This is denoted as *geodesic convolutional neural networks* (GCNN). The geodesic disc is divided into radial and angular bins. Continuous function values within the bins are interpolated from discrete samples by Gaussian kernels. The filter weights for each bin can be trained similar to standard CNNs. Masci et al. present how to learn descriptors and correspondences. First, signatures are computed on the input data such as the WKS or HKS. In order to compute descriptors, a siamese loss is introduced as in [WVR*14]. To find correspondences, the *negative log-likelihood loss* is used (cf. Section 2.2.5). [BMRB16] present *anisotropic convolutional neural networks* which follow a similar principal. Instead of constructing patches around a vertex, anisotropic heat kernels are introduced as weighting functions which are aligned along the principal curvature directions. This avoids the rotational degree of freedom in [MBBV15] and is more robust to the surface tessellation. The learned filter weights correspond to the scale and the orientation of the heat kernel. This method can be generalized even further by avoiding handcrafted kernels for interpolation (see e.g. [MBM*17]).

A technique which avoids resampling and blending by kernels was proposed in [LDCK18]. Lim et al. present a neighborhood encoding operation (similar to the effect of a convolution) in form of a spiral operator, which follows a sequence of the mesh vertices around the k-rings of the center vertex. Since the vertices are given in form of a sequence, this data is handed to a LSTM architecture. The weights of the LSTM-cells can be learned for tasks such as point-correspondence computation as above. The results show that applying LSTM networks to the given sequence data outperforms previous work in terms of correspondence accuracy.

2.6.1.5 Distance Measures

To compare point signatures, we evaluate a distance measure. A simple technique to quantify the similarity of a descriptor $f(\boldsymbol{p}_1) \in \mathbb{R}^n$, and $f(\boldsymbol{p}_2) \in \mathbb{R}^n$ evaluated at two points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is to compute the (squared) Euclidean distance:

$$d_{\text{Euclidean}}(f(\boldsymbol{p}_1), f(\boldsymbol{p}_2)) = \|f(\boldsymbol{p}_1) - f(\boldsymbol{p}_1)\|_2,$$
$$d_{\text{Euclidean,sq}}(f(\boldsymbol{p}_1), f(\boldsymbol{p}_2)) = \|f(\boldsymbol{p}_1) - f(\boldsymbol{p}_1)\|_2^2.$$

Unfortunately, the Euclidean distance is not robust to outliers and sensitive to differently scaled function values. Hence, a typical method is to normalize the descriptor values [ASC11]:

$$d_{\text{Euclidean, normalized}}(f(\boldsymbol{p}_1), f(\boldsymbol{p}_2)) = \sum_{i=1}^{n} \frac{|f(\boldsymbol{p}_1)[i] - f(\boldsymbol{p}_2)[i]|}{|f(\boldsymbol{p}_1)[i] + f(\boldsymbol{p}_2)[i]|}.$$

Again, a version of this distance where the nominator is squared exists, which is denoted as the X^2-distance. A further possibility is to normalize over the n dimensions of the descriptor. This can be useful if the n descriptor entries significantly vary in scale, as it is the case for the heat-kernel-signature [SOG09]:

$$d_{\text{HKS}}(f(\boldsymbol{p}_1), f(\boldsymbol{p}_2)) = \sum_{i=1}^{n} \frac{|f(\boldsymbol{p}_1)[i] - f(\boldsymbol{p}_2)[i]|}{\sum_{p \in P} f(\boldsymbol{p})[i]}.$$

Note that this normalization assumes that a common set P exits which contains all surface points. There are various other measures to compare vectors (e.g. angle distance, statistical methods). Here, we only mention those relevant for further reading. Note, that it is highly application dependent which measure is most suitable.

2.6.2 Symmetry Detection

With local descriptors we can find similar points by evaluating a distance measure on the respective signatures. However, the computed point correspondences are treated completely independent from each other. Hence, it is possible that several points have the same best match. If we want to find globally consistent

correspondences (e.g. the neighborhood of a point is mapped to the neighborhood of its correspondence), we need to exploit global adjacency relations. In this section we will discuss approaches to *partial symmetry detection* that use such relations to detect similar regions on the same surface.

Definition 2.4 (Symmetry) *A surface $\mathcal{S}$ or a subset of this surface $\bar{\mathcal{S}} \subset \mathcal{S}$ is* ***symmetric*** *with respect to a transformation/mapping T, if it is invariant under the application of T, i.e., $T(\bar{\mathcal{S}}) = \bar{\mathcal{S}}$. If $T(\mathcal{S}) = \mathcal{S}$ for the entire shape $\mathcal{S}$, T is referred to as a* ***global*** *symmetry. Otherwise, if $T(\bar{\mathcal{S}}) = \bar{\mathcal{S}}$, T represents a* ***partial*** *symmetry of $\mathcal{S}$. A symmetry group defines a set of transformations that satisfy the following axioms [MPWC13]:*

1. ***Closure:*** *If a surface is symmetric with respect to the transformations T_1 and T_2 it is also symmetric with respect to their composition $T_1 \cdot T_2$.*
2. ***Identity element:*** *The surface is symmetric with respect to the identity* **Id**.
3. ***Inverse element:*** *The inverse T^{-1} of a symmetry transformation T, with $T \cdot T^{-1} = \mathbf{Id}$ is also a symmetry transformation.*
4. ***Associativity:*** *For symmetry transformations T_1, T_2, and T_3: $(T_1 \cdot T_2) \cdot T_3 = T_1 \cdot (T_2 \cdot T_3)$ holds true.*

In general, symmetry detection methods can be categorized based on the following criteria [MPWC13]:

1. *Input surface representation:* e.g. points, graphs, volume, or mesh.
2. *Global vs. partial symmetry:* In the first case the entire object is symmetric with respect to some transformation. Partial symmetries only refer to a local surface patch.
3. *Approximate vs. exact symmetry:* The definition given above describes an exact symmetry. However, in real-world surface measurements it is unlikely that the repeated geometric parts are identical. Approximate

symmetries on a mesh $\boldsymbol{M}$ are measured via a distance function $d(\boldsymbol{M}, \boldsymbol{T}(\boldsymbol{M}))$. $\boldsymbol{M}$ and $\boldsymbol{T}(\boldsymbol{M})$ are $\epsilon-$symmetric if $d(\boldsymbol{M}, \boldsymbol{T}(\boldsymbol{M})) < \epsilon$.

4. *Intrinsic vs. extrinsic:* This category is directly related to the previous one and defines how the distance function d is computed. Similar to the observations made for surface descriptors above, we can use extrinsic measures as the Euclidean distance between points. If we measure distances on shapes undergoing isometric transformations, an intrinsic measure (e.g. geodesics) should be used.

There is a large body of work regarding different combinations of the criteria discussed above. We refer the interested reader to [MPWC13] for a concise overview. Here, we will discuss the approaches most related to our work which compute partial, approximate, extrinsic symmetries on meshes and graphs.

2.6.2.1 Symmetry Detection on Meshes

A popular technique to compute partial symmetries in a mesh $\boldsymbol{M}$ is based on *transformation space voting* [MGP06, PMW*08, SAD*16]. These algorithms follow a five-step procedure:

1. *Feature detection:* First, salient points on the surface are identified, e.g. by selecting points with high absolute curvature.

2. *Feature point matching:* From the set of feature points, correspondences are extracted according to a point signature.

3. *Transformation between local frames:* For the points (with index i and j) of each correspondence pair a local frame is computed, constituent of the surface normal, and the principal curvature directions. Then the transformations $\boldsymbol{T}_{ij}$ that align these local frames are computed for all point pairs. These are 7-dimensional *similarity transformations* composed of a rotational, translational, and scale component.

4. *Voting:* The transformations obtained from the previous step are stored in a global 7-dimensional voting space. The goal is to find those transformations that reoccur frequently, which corresponds to finding cluster-maxima in the voting space. These are computed by applying mean-shift clustering.

5. *Spatial verification:* In the transformation space, the spatial relation of the surface is lost. Hence, as a final step random seed pairs are selected, from which surface patches are grown which match under the same transformation.

[PMW*08] extend this algorithm by finding structure in the voting space. Regular grids that are generated by combining k equal transformations are denoted as *orbits*. In the voting space, these occur as regular 1D or 2D grids. The corresponding axes of the *generating transformation* are called *generators* $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$. Initially, the grid generators are estimated with a RANSAC based grid fitting procedure. Next, an energy minimization is applied which iteratively aligns the data points (the transformations) to the grid by evaluating a bi-directional distance function.

Symmetry detection based on transformation space voting has proved to work well in practice for clean input. In case of noise, e.g. for real-world scanned objects, where surface descriptors are less reliable, it can occur that the voting space becomes very dense, making it hard to obtain cluster maxima. Especially, since the voting space is global and contains transformations from the entire mesh, cluster boundaries are hard to find. Furthermore, often real-world scans are large data sets making this process computationally expensive.

2.6.2.2 Symmetry Detection on Graphs

To overcome some of the problems mentioned above, [BBW*09, BWM*11] propose symmetry detection based on graphs from feature curve networks. In the presence of noise, local surface descriptors are not reliable. Feature curves have the advantage that their geometric structure and connectivity can be compared,

containing more global information, which is more robust in a setting with noise. We give a rough overview over the required steps:

1. *Feature curve detection:* First, an initial feature curve network is computed (e.g. by slippage analysis). Since, these algorithms operate on noisy input data, the extracted curves are often fragmented.

2. *Feature curve enhancement:* To make a feature graph applicable for graph matching, feature segments are enhanced by locally reconnecting fragmented curves or applying morphological operations. Furthermore, weak features (with low curvature) are neglected from the curve network.

3. *Graph representation:* In this point the approaches choose different representations: [BBW*09] choose k-nearest-neighbor graphs. [BWM*11] use *affine shape spaces*, which encode the reoccurring structures.

4. *Graph matching:* The sub-graphs are compared by applying *iterative closest line* and RANSAC based fitting, or by comparing the sub-graph topology.

5. *Geometric validation:* Dense surface correspondences can be found and verified by transporting the transformations found in the feature graph onto the surface.

The benefit of these techniques is that all computations are performed on a graph structure of the feature curves, and no additional descriptors are required, which is more robust in the presence of noise. Furthermore, the use of feature curves restricts to the relevant points and gives a sub-sampling of the input for analysis. On the downside, feature curves detected in noisy real-world data are usually heavily fragmented and jagged and require a lot of enhancement, if they are to be used for the algorithms discussed above.

2.6.3 Consistent Correspondences on Shapes with Functional Maps

If multiple shapes are involved, e.g. in tasks where attributes are transferred (e.g. texture, surface attributes, normals, etc.) from one shape to the other or corresponding point locations need to be known (e.g. for simultaneous object deformation, pose transfer, co-segmentation, etc.) consistent correspondences need to be established. For example, in the image above it possible to transfer the texture from the zebra to the elephant and the cat due to consistent correspondence information.

Finding consistent correspondences between different non-rigidly deformed shapes is a very challenging task. Given two shapes $\mathcal{S}_1$ and $\mathcal{S}_2$, the goal is to find a consistent assignment $\boldsymbol{T}$ that maps points on $\mathcal{S}_1$ to those of $\mathcal{S}_2$. Solving this directly results in a large combinatorial problem, which is computationally expensive. [OBCS*12] have proposed *functional maps* to overcome this complexity, by formulating a function correspondence problem, which can be expressed in a lower dimensional Laplace–Beltrami eigenspace.

Assume we are given the mapping $\boldsymbol{T} : \mathcal{S}_1 \to \mathcal{S}_2$ and real-valued corresponding functions $\mathcal{g}_1 : \mathcal{S}_1 \to \mathbb{R}$ and $\mathcal{g}_2 : \mathcal{S}_2 \to \mathbb{R}$. Now, functions are easily mapped from one shape to the other by $\mathcal{g}_2(\boldsymbol{p}) = \boldsymbol{T}_F\left(\mathcal{g}_1\right)(\boldsymbol{p}) = \mathcal{g}_1\left(\boldsymbol{T}^{-1}(\boldsymbol{p})\right)$. If $\boldsymbol{T}$ is bijective, the original mapping can be recovered from the functional map $\boldsymbol{T}_F$ by transporting an indicator function. Now, the goal is to find a functional map $\mathcal{g}_2 = \boldsymbol{T}_F \circ \mathcal{g}_1$ without knowledge of $\boldsymbol{T}$.

From the previous chapters we know that functions can be defined on both shapes that allow for comparison of surface points. Point signatures (discussed in Chapter 2.6.1) with similar parameter settings can be computed on both shapes and represent corresponding functions. From these, the functional map $\boldsymbol{T}_F$ is

estimated. It is common to represent the $\mathcal{g}_i$ in the eigenbasis of the *Laplace–Beltrami operator* with $\Delta_{\mathcal{S}_i}\phi_j^{\mathcal{S}_i} = \lambda_j^{\mathcal{S}_i}\phi_j^{\mathcal{S}_i}$, such that:

$$\mathcal{g}_i = \sum_{j\geq 1} \underbrace{\langle \phi_j^{\mathcal{S}_i}, \mathcal{g}_i \rangle}_{\tilde{\mathcal{g}}_i} \phi_j^{\mathcal{S}_i}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Usually, only the first k coefficients are used to reduce the complexity of the representation and avoid high-frequency noise. When the functional map is applied to $\mathcal{g}_1$ we get:

$$\mathcal{g}_2 = T_F(\mathcal{g}_1) = T_F\left(\sum_{j=1}^{k} \tilde{\mathcal{g}}_1 \phi_j^{\mathcal{S}_1}\right) = \sum_{j=1}^{k} \tilde{\mathcal{g}}_1 T_F\left(\phi_j^{\mathcal{S}_1}\right).$$

$T_F\left(\phi_j^{\mathcal{S}_1}\right)$ is a function on $\mathcal{S}_2$ and can also be written in basis notation as:

$$T_F\left(\phi_j^{\mathcal{S}_1}\right) = \sum_{l=1}^{k} \tilde{\boldsymbol{F}}_{lj} \phi_l^{\mathcal{S}_2}.$$

Hence, for the entire map we get:

$$\mathcal{g}_2 = \sum_{l=1}^{k} \tilde{\mathcal{g}}_2 \phi_l^{\mathcal{S}_2} = \sum_{j=1}^{k}\sum_{l=1}^{k} \tilde{\mathcal{g}}_1 \tilde{\boldsymbol{F}}_{lj} \phi_l^{\mathcal{S}_2}. \tag{2.6}$$

The only unknown in this term is $\tilde{\boldsymbol{F}}_{lj}$. To compute $\tilde{\boldsymbol{F}} \in \mathbb{R}^{k\times k}$ let us consider the discrete setting where $\mathcal{S}_i$ is represented by a mesh $\boldsymbol{M}_i$, with a set of m corresponding functions $\boldsymbol{G}_1 = \left(\boldsymbol{g}_1^1, \ldots, \boldsymbol{g}_1^m\right) \in \mathbb{R}^{|V_{M_1}|\times m}$ and $\boldsymbol{G}_2 = \left(\boldsymbol{g}_2^1, \ldots, \boldsymbol{g}_2^m\right) \in \mathbb{R}^{|V_{M_2}|\times m}$ defined at the vertices of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ respectively. With equation 2.6, $\tilde{\boldsymbol{F}}$ is a matrix in $\mathbb{R}^{k\times k}$ which can be estimated via a linear least squares optimization:

$$\tilde{\boldsymbol{F}} = \arg\min_{\tilde{\boldsymbol{F}}} \left\| \tilde{\boldsymbol{F}}\tilde{\boldsymbol{G}}_1^\top - \tilde{\boldsymbol{G}}_2^\top \right\|_{\text{Fro}}^2 \tag{2.7}$$

where $\|\cdot\|_{\text{Fro}}$ denotes the *Frobenius norm*. This objective can be further extended by soft constraints. E.g. we can obtain smoother maps by finding a functional

map that commutes with the Laplace–Beltrami operator. Then the following term is added to the objective function:

$$\left\| \tilde{\boldsymbol{F}} \boldsymbol{\Lambda}_{M_1} - \boldsymbol{\Lambda}_{M_2} \tilde{\boldsymbol{F}} \right\|_{\text{Fro}} \tag{2.8}$$

with $\boldsymbol{\Lambda}_{M_i} = \text{diag}(\lambda_1^{M_i}, \dots, \lambda_k^{M_i})$. The crucial part of the functional maps framework is the requirement to provide pairs of corresponding functions. Automatically constructing such functions, by computing local descriptors, is often practically limited to the near-isometric setting.

Various extensions and improvements to this initial approach have been presented. [KBB*13, EKB*15] use joint diagonalization to develop compatible bases on the two surfaces in the near-isometric case. [PBB*13] use ideas from sparse recovery to improve the functional mapping pipeline and associated bases. [ERGB16] use coupled functional maps to obtain consistent mappings from the source to the target shape and vice versa. Also, partial functional maps [RCB*17] overcome the problem of missing data in case a full scan of the source or target is not provided. [HWG14] extend to collections of more than two shapes, computing consistent maps in shape collections. Finally, descriptor preservation can be improved by promoting commutativity with pointwise multiplication operators [NO17, WGBS18] and by approximately preserving products of functions [NMR*18].

2.6.3.1 Point Correspondences from Functional Maps

As stated above, it is possible to recover point-to-point correspondences from the functional maps. However, simply mapping an indicator function via the functional map might not lead to a unique maximum in case the two shapes are not identical. To map indicator functions, [OBCS*12] project these into the eigenbasis $\boldsymbol{\Phi}_{M_1}$ and map them with the functional map: $\overline{\boldsymbol{\Phi}}_{M_1} = \left(\tilde{\boldsymbol{F}} \boldsymbol{\Phi}_{M_1}^\top \right)^\top$. Applying $\tilde{\boldsymbol{F}}$ to $\boldsymbol{\Phi}_{M_1}$ approximates $\boldsymbol{\Phi}_{M_2}$ by a linear combination of the basis vectors of $\boldsymbol{\Phi}_{M_1}$. Hence a correspondence for each vertex of M_1 (rows of $\overline{\boldsymbol{\Phi}}_{M_1}$) can be found as the nearest neighbors in the rows of $\boldsymbol{\Phi}_{M_2}$. However, this technique does not

take any neighborhood information for the map recovery into account. Even though the computed maps are smooth, this property does not have to extend to the point-to-point reconstruction.

Several techniques were proposed which take the geodesic neighborhood information into account to compute point-to-point correspondences from functional maps [ESBC19, EBC17, RMC17]. Exemplary, we will discuss the most recent method presented in [ESBC19]. They minimize a geodesic harmonic energy of a map from $M_1 = (V_1, E_1, F_1)$ to $M_2 = (V_2, E_2, F_2)$ (forward map T_{12}) and vice versa (backward map T_{21}), and a reversibility term which relates both maps. The geodesic harmonic energy of the maps are given by [ESBC19]:

$$\begin{aligned}\mathcal{E}_{gh}\left(T_{12}\right) &= \sum_{(v_i,v_j)\in E_{M_1}} w_{(v_i,v_j)} d^2_{M_2}\left(T_{12}\left(p_i\right), T_{12}\left(p_j\right)\right)\\ \mathcal{E}_{gh}\left(T_{21}\right) &= \sum_{(v_i,v_j)\in E_{M_2}} w_{(v_i,v_j)} d^2_{M_1}\left(T_{21}\left(p_i\right), T_{21}\left(p_j\right)\right),\end{aligned}$$

where $d_M(\cdot,\cdot)$ denotes the geodesic distance and $w_{(v_i,v_j)}$ is the cotangent weight (cf. Section 2.3) of the edge $\left(v_i, v_j\right)$.

The energy $\mathcal{E}_{gh}$ approximates the *Dirichlet energy* $\frac{1}{2}\int_M \|\mathrm{d}T\|^2\, dv$ (with the map differential d). With the Diriclet energy, the map smoothness is evaluated. Harmonic maps between smooth surfaces are defined as minima of this energy [ESBC19]. The forward and backward energy are combined by an area-weighted average to a common energy term $\mathcal{E}_{gd}$ [ESBC19], which will be minimized in the following.

If we merely optimize for $\mathcal{E}_{gh}$, the map will collapse into a single point. Hence, a reversibility term is added, which first applies the forward and secondly the

backward map and measures the geodesic distance to the original point location [ESBC19]:

$$\mathcal{E}_r\left(T_{ij}, T_{ji}\right) = \sum_{i,j\in\{1,2\}, i\neq j} \frac{1}{A_i^2} \sum_{p\in P_i} d^2_{M_i}\left(T_{ji}\left(T_{ij}(\boldsymbol{p})\right), \boldsymbol{p}\right) \cdot A_i(\boldsymbol{p})$$
$$A_i = \sum_{p\in P_i} A_i(\boldsymbol{p}),$$

where $A_i(p)$ denotes the area weight of the point p. The final energy is given by [ESBC19]:

$$\mathcal{E}_T\left(T_{12}, T_{21}\right) = \alpha\mathcal{E}_{gd}\left(T_{12}, T_{21}\right) + (1-\alpha)\,\mathcal{E}_r\left(T_{12}, T_{21}\right).$$

In the discrete setting T_{ij} is a matrix in $\mathbb{R}^{|V_i|\times|V_j|}$, where each row contains at most 3 non-zero entries indicating the barycentric coordinates onto which a point is mapped on the other surface. The maps can be initialized with one of the previously mentioned methods and updated in an iterative optimization procedure, which minimizes the energy $\mathcal{E}_T$ while abiding the constraints on T_{ij}. While this method is far more computationally expensive in comparison to previously mentioned techniques, the resulting maps are smooth and consistent, such that they can be used in downstream applications such as texture mapping, co-segmentation, or classification.

3 Feature Curve Correspondences

As observed above (cf. Section 2.6), a consistent set of point correspondences is essential for various shape analysis tasks. However, points are isolated entities, while feature curves characterize connected salient segments of the surface. The additional topological information provides consistent and continuous correspondences along the entire curves. This intermediate level, which can be placed between pointwise matches and entire shape maps, facilitates shape analysis tasks, which require global reoccurrence information. E.g. global reoccurrence is a strong indicator to select parts of shapes that are relevant and meaningful. Furthermore, this intermediate correspondence level can ease the computation of entire surface to surface maps. Details are elaborated in Chapter 5.

In this chapter we discuss different methods to obtain feature curve correspondences. Depending on the application we might be facing different kinds of challenges. E.g. finding correspondences on differently articulated shapes, having high accuracy requirements in task specific settings, or facing large real-world data sets, which are influenced by noise or incomplete coverage.

The task of finding corresponding curves becomes increasingly complex in the case where several (different) meshes are taken into account, which underlie isometric deformations. Similar to the point-descriptors we will show possible feature curve signatures, based on which we are able to compare feature curves even under (near-)isometric deformations of the shapes. While curve signatures provide a fully automatic approach to compare feature curves, in task specific settings we might be facing high accuracy requirements, which are hard to satisfy in a pure model based approach. Hence we will show how to learn feature curve

correspondences with a data-driven approach by training a neural network based on long-short-term-memory (LSTM) cells. A further challenge arises when real-world data is processed. In this case we need to be able to find correspondences robustly from a large amount of noisy, fragmented, and potentially misclassified feature curves. Thus in the final section, we will present a robust local-global transformation space voting scheme, which finds feature curve correspondences that are induced by rigid transformations.

Some of the material, techniques, and findings presented in this chapter have previously been published in [GBK16] and [GLK18].

3.1 Feature Curve Descriptors

In Chapter 2 we introduced point descriptors as a method to compare surface points and find point-to-point correspondences. Especially, signatures that are invariant to isometric deformations are useful in case the regarded objects are differently articulated shapes. To compare feature curves, we exploit this idea and extend point signatures to curve descriptors. In this chapter we will explore two possibilities: (1) we extend the *Heat-Kernel-Signature* (HKS) to describe the heat flow from the curve to itself and (2) present the *Geodesic Iso-Curve Signature* (GICS) on curves. Both descriptors are based on intrinsic measures and are thus invariant to isometric deformations. While the HKS is computationally more efficient, since it is based on the Laplace–Beltrami eigenspectrum, the GICS captures additional topological information on the global structure of the shape. In the context of a specific task, we can further improve matching accuracies by learning feature curve correspondences from prelabeled examples. We will present an LSTM based neural network as a supervised variant to learn curve correspondences. We first present all three methods and conclude with an overall comparison in Section 3.1.4.

3.1.1 Heat-Kernel-Signature for Feature Curves

In Section 2.6.1 we introduced the Heat-Kernel-Signature for isolated points, which describes the heat-flow from a point to itself after a certain amount of time. To extend this definition to curves, we would like to formulate a HKS for curves, which measures the amount of heat that flows from the curve back to itself. Given a FCN $\Gamma = (V, E, C)$, which is computed on a surface mesh M, we define the heat-kernel-signature for a curve $c \in C$ as:

$$f_{\mathrm{HKS}}(c) = \begin{pmatrix} f_{\mathrm{HK}_{t_1}}(c,c) \\ \vdots \\ f_{\mathrm{HK}_{t_q}}(c,c) \end{pmatrix},$$

for q time-steps $t_1, \ldots, t_q$. To compute $f_{\mathrm{HK}_t}(c,c)$, first we need to define the HKM with an initial heat distribution

$$\delta_c(\boldsymbol{p}) = \begin{cases} 1 \text{ if } \boldsymbol{p} \text{ lies on } c \\ 0 \text{ else.} \end{cases}$$

i.e. the entire curve emits heat at $t = 0$. We apply the operator $\mathcal{H}_t$ to the initial distribution. From the point-based HKS we know that [SOG09]:

$$\mathcal{H}_t \cdot f(\boldsymbol{p_1}) = \int_M f_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2) \cdot f(\boldsymbol{p}_2)\, d\boldsymbol{p}_2.$$

Hence for $f = \delta_c$ we get:

$$\begin{aligned} f_{\mathrm{HK}_t}(c, \boldsymbol{p_1}) = \mathcal{H}_t \cdot \delta_c(\boldsymbol{p}_1) &= \int_M f_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2) \cdot \delta_c(\boldsymbol{p}_2)\, d\boldsymbol{p}_2 \\ &= \int_c f_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2)\, d\boldsymbol{p}_2. \end{aligned}$$

$f_{\mathrm{HK}_t}(c, \boldsymbol{p_1})$ computes the amount of heat that travels from c to a point $\boldsymbol{p}_1$. To obtain the heat-kernel-signature $f_{\mathrm{HK}_t}(c,c)$, we need to accumulate the heat that travels from c to all points on c. This is achieved by integration over c:

$$f_{\mathrm{HK}_t}(c,c) = \int_c \int_c f_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2)\, d\boldsymbol{p}_2 d\boldsymbol{p}_1.$$

In the discrete setting we approximate the integral by:

$$\mathscr{f}_{\mathrm{HK}_{t_q}}(c,c) = \sum_{v_1 \in c} \sum_{v_2 \in c} w_l(v_1) \cdot w_l(v_2)\, \mathscr{f}_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2)$$

with weights $w_l(v)$ which denote the normalized lengths of the curve segment corresponding to a vertex v lying on a curve c and with

$$\mathscr{f}_{\mathrm{HK}_t}(\boldsymbol{p}_1, \boldsymbol{p}_2) = \sum_{i=0}^{\infty} exp(-\lambda_i t)\, \phi_i[v_1]\, \phi_i[v_2]$$

as discussed in Section 2.6.1. Hence it is possible to compute the HKS on curves solely from the eigenfunctions and eigenvalues of the Laplace–Beltrami operator, which makes the computation very efficient.

3.1.1.1 Distance Computation

For two meshes $\boldsymbol{M}$ and $\boldsymbol{M}'$ with FCNs Γ and Γ' we compute the descriptor distance of two curves $c \in \boldsymbol{C}$ and $c' \in \boldsymbol{C}'$. Since the values of the curve based HKS can have strongly varying scales, we choose a normalized Euclidean distance measure as proposed in [ASC11]:

$$d_{\mathrm{HKS,curves}}(\mathscr{f}_{\mathrm{HKS}}(c), \mathscr{f}_{\mathrm{HKS}}(c')) = \sum_{i=1}^{q} \frac{|\mathscr{f}_{\mathrm{HKS}}(c)[i] - \mathscr{f}_{\mathrm{HKS}}(c')[i]|}{|\mathscr{f}_{\mathrm{HKS}}(c)[i] + \mathscr{f}_{\mathrm{HKS}}(c')[i]|}.$$

This distance measure is robust to outliers and not sensitive to different scales of the descriptor entries.

3.1.1.2 Limitations

The HKS is based on the spectrum of the Laplace–Beltrami eigenfunctions. In case of degenerate mesh entities, it can occur that the computation of the eigenvectors becomes unstable. Furthermore, due to the integration over the curve the heat-kernel values are blended along the curve, which leads to less accurate matching results (cf. Section 3.1.4). However, the descriptor can be computed quite efficiently and it can be useful in combination with other (e.g. topological) data.

3.1.2 Geodesic Iso-Curve Signature for Feature Curves

In Section 2.6.1 we discussed the Geodesic Iso-Curve Signature for points. Strong topological changes in the iso-curves such as splits, merges, or collapses are captured in the signature as extrema. This additional global information improves the matching accuracy in case of point correspondences. Hence, we formulate the GICS for curves in the following. Given a manifold surface mesh $\boldsymbol{M}$ and a FCN $\Gamma = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{C})$, we define the GICS for feature curves as:

$$\not\ell_{\mathrm{GICS}}(c) = [l_c(r_1), l_c(r_2), \dots, l_c(r_n)] \in \mathbb{R}^n$$

for a curve $c \in \boldsymbol{C}$, i.e., the zero-set of the geodesic distance computation is the entire feature curve, and with radii

$$r_1 \leq r_2 \leq \dots \leq r_n, r_i \in \mathbb{R}.$$

As above, the function $l_c(r_i)$ maps the radius r_i to the length of the corresponding iso-curve:

$$l_{c \in C} : [0, d_{\max}(c)] \to \mathbb{R}.$$

For the computation of the signature, several design choices have to be made: (1) an algorithm to compute the geodesic distances is selected, and (2) a method for the approximation of the isocontours of the geodesic discs has to be chosen. In addition, (3) a distance metric has to be defined on the signatures for comparison. Finally, (4) we need to decide how to sample the geodesic radii on $\boldsymbol{M}$. In the following we will discuss these design choices.

3.1.2.1 Geodesic Distances

Diverse algorithms exist to compute geodesic distances on a triangular surface mesh $\boldsymbol{M}$ (e.g. [CDGDS13, CHK13, BK07, SSK*05]). If performance is the main objective, the fast heat method [CDGDS13] would be a good choice, and depending on the type of input data, there might other preferable options. However, we use approximate exact geodesic distances [BK07], because of their robustness.

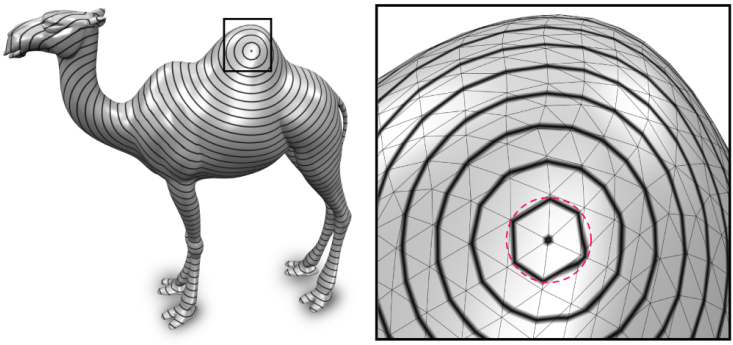


Figure 3.1: Approximation of the length of an isocontour (red) for a single point as zero-set. The sub arcs of the isocontour are approximated by the line which connects the two points at which the iso-contour intersects the triangle of a mesh M (left).

Furthermore, [BK07] present an extension to the exact geodesics computation in [SSK*05], which computes geodesic distances from isolated point sets. Bommes et al. exended their work for arbitrary, possibly open, polygons on the mesh, which define the zero set of the distance field. Hence, arbitrary sources can be set for the computation of geodesic distances. E.g. we can choose point sets, curves, polygonal graphs or even a mix of the afore mentioned as sources. Then, the iso-contours of the geodesic radii can be computed for these sets as well, so that we can generate signatures for sets of primitives. In our case these primitives are the feature curves $c \in C$.

3.1.2.2 Radii Computation

We compute the geodesic distance from a seed curve $c \in C$ on M. To approximate the length $l_c(r_i)$ of an isocontour corresponding to a radius r_i, we sum up the lengths of the line segments between the two intersection points of each triangle intersecting the isocontour (see Figure 3.1). Let $Tr = (v_0, v_1, v_2)$ denote a triangle

that is intersected by the isocontour along the edges (v_0, v_1) and (v_0, v_2) for a radius r_i. The segment s_{Tr} of the isocontour that intersects this triangle is approximated as:

$$l_{s_{Tr}}(r_i) = \left\| \overline{\boldsymbol{p}}_1 - \overline{\boldsymbol{p}}_2 \right\|_2$$

with

$$\overline{\boldsymbol{p}}_j = (1 - w_j) \cdot \boldsymbol{p}_0 + w_j \cdot \boldsymbol{p}_j \quad \text{and} \quad w_j = \frac{|r_i - d_M^0|}{|d_M^j - d_M^0|},$$

where d_M^j denotes the geodesic distance between c and $\boldsymbol{p}_j$. The sum of the lengths of the segments $l_{s_{Tr}}(r_i)$ approximates the length of the respective isocontour.

3.1.2.3 Distance Computation

For two shapes M and M', we compute the distance of the signatures of two curves c on M and c' on M'. Normalized distances have the benefit of being more robust to outliers. Hence, similarly as above we define a distance using the L^1 norm of normalized iso-curve length distances at the same radii to compare the geodesic signatures.

$$d_{\text{GICS}}(c, c') = \sum_{i=0}^{i_{\max}} \frac{|l_c(r_i) - l'_{c'}(r_i)|}{|l_c(r_i) + l'_{c'}(r_i)|}$$

The lengths $l'_{c'}(r_i)$ on the shape M' are interpolated linearly from the two radii r'_j and r'_{j+1} with $r'_j \leq r_i \leq r'_{j+1}$, since the radii are not necessarily sampled at the same values (cf. Section 3.1.2.4). From each curve the maximal geodesic distance $d_{\max}(c)$ on M can be different. Hence, we compute a maximal index $i_{\max}$ for which both signatures have defined values. Hence $i_{\max}$ is defined as

$$i_{\max} = \max\{i | r_i \leq \min(d_{\max}(c), d_{\max}(c'))\}$$

Finally, to make this distance symmetric, we compute the descriptor distance as:

$$d_{\text{GICS, sym}}(c, c') = \frac{D_{\text{GICS}}(c, c') + D_{\text{GICS}}(c', c)}{2}.$$

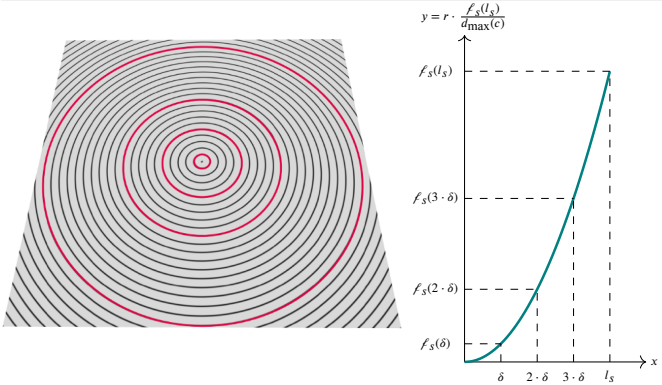


Figure 3.2: Radius sampling. To capture local shape properties and guarantee local shape awareness, we adjust the distribution of samples (geodesic radii) in the domain of the length function according to a monotonically increasing sampling function (right). On the left we depict the respective sample distribution on a plane (red).

3.1.2.4 Radius Sampling

The function $l_{c \in C} : [0, d_{\max}(c)] \to \mathbb{R}$, which describes the length of an isocontour at radius r, needs to be sampled at discrete values $r_1, \ldots, r_n$ in order to obtain a descriptor of size n. An obvious choice would be to uniformly sample r between 0 and $d_{\max}(c)$. Unfortunately, then the local shape is not captured well by the signature. To achieve local shape awareness, we need to increase the density of local samples. Hence, we introduce a continuous monotonically increasing sampling function $f_s : [0, l_s] \to \mathbb{R}$. The domain of this sampling function is fixed to preset values $[0, l_s]$ on which we uniformly sample (see x-axis in Figure 3.2 for $n = 4$). This domain is equal for each shape, since we want to apply the same sampling for each signature. Hence, the following increment is computed in the domain:

$$\delta = \frac{l_s}{n}.$$

Since the maximum function value is $\ell_s(l_s)$, in order to obtain radius values, the function values need to be rescaled by $\frac{d_{\max}(c)}{\ell_s(l_s)}$, such that the radii can be computed as:

$$r_i = \ell_s(i \cdot \delta) \cdot \frac{d_{\max}(c)}{\ell_s(l_s)}$$

Examples for possible sampling functions are $\ell_s(x) = x$, $\ell_s(x) = x^2$, and $\ell_s(x) = x^4$. Overall, we found that applying the sampling function $\ell_s(x) = x^4$ allows a more accurate distinction between local surface properties in case of point-correspondences. However, we did not measure a significant difference for feature-curve correspondences.

3.1.2.5 Limitations

To compute the GICS, we need to compute geodesic distances from all sample curves. This affects the computational efficiency. Possibly, this can be improved by choosing a more efficient method to compute geodesics (e.g. geodesics in heat [CDGDS13] or [SRGB14]). A further limitation of our approach is that this signature is not scale invariant in its current form. We believe that it can be transformed into a scale invariant version by scaling the meshes by an average geodesic distance before the computation.

3.1.3 Learning Curve Correspondences

Since the descriptors presented above have a general and flexible formulation, they can be computed fully automatically on arbitrary shapes. This however also implies that, since they are not optimized for a specific task, they can result in less accurate correspondences. Defining a model for a specific application can be very complicated and it is often not possible to handcraft such a descriptor effectively, since the underlying functions might be highly complex. However, as discussed in Section 2.6.1.4 it is possible to learn descriptors or correspondences, if we are given a set of exemplary instances with target labels and an appropriate objective.

In this Section we present a method to learn feature curve correspondences from examples. Given a set of shapes $\{\mathcal{S}_1, \dots, \mathcal{S}_k\}$ where we assume an upright orientation (e.g. with [FCODS08, LZL16]) with FCNs $\Gamma_1, \dots \Gamma_k$, our goal is to find corresponding curves among the curves of the FCNs. To learn from the examples, we assume that the curves $c \in \boldsymbol{C}_i$ of Γ_i are indexed with one of N_{label} labels. In the following we discuss our design choices to learn correspondences for this data. First we discuss the input representation. Inspired by previous work in the field (e.g. [LDCK18]), we learn the task specific correspondences by handing curves as a sequence of pointwise information to a neural network.

3.1.3.1 Input Representation

Initially, the feature curves are given by a sequence of vertices. Since corresponding feature curves on different shapes can have different lengths, we first resample the curves uniformly to sequences of N_v vertices, to be less sensitive to varying lengths. To construct a more descriptive representation, we compute the DEP (cf. Section 2.6.1) for each vertex. The DEP describes the intrinsic properties of the shape. Alone however, it is invariant to mirror symmetries and cannot capture the complete structure of the shape. Furthermore, local details are not represented well. Hence, we augment it by relative orientation information. At each feature point $\boldsymbol{p}$ we compute the normal and an orientation vector $(\boldsymbol{p} - \boldsymbol{\mu}_i)$ where $\boldsymbol{\mu}_i$ denotes the center of gravity of the shape $\mathcal{S}_i$. While the normal captures local surface orientations the direction vector encodes the relative global structure of the curve network. Both representations are invariant to translations but not to mirror symmetries. However, the direction vectors are not rotation invariant. To overcome this issue, the orientation vectors of a shape are rotated by a random angle (consistently over the entire shape) around $\boldsymbol{\mu}_i$, where the rotation vector corresponds to the shapes upright direction. A new random rotation is computed each time before the shape's curves are seen by the neural network. This representation of the local frame allows to differ between mirror symmetric curves but is translational and rotational invariant. Combined with the intrinsic information

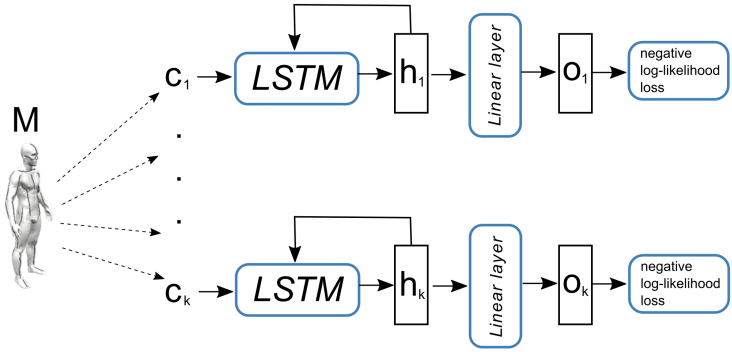


Figure 3.3: LSTM Network for feature curve classification. Each curve c_i is encoded by a sequence of descriptor values $(d_1, \dots d_{N_v})$ measured along the curve. These are passed to an LSTM cell and a linear layer. The output o_i predicts the class of a feature curve, which is evaluated based on a negative log-likelihood loss function.

provided by the DEP, we receive a descriptive representation of the feature curves, which captures both local orientations and the global shape properties.

3.1.3.2 LSTM Network

The input to the network is given by a sequence of descriptors. As discussed above, the information that this sequence encodes provides additional topological cues in contrast to isolated points. Ideally, we want to train a neural network that is specialized for sequence data and computes a compact representation of this sequence. In Section 2.2.5 we discussed that this is the case for LSTM networks. These are especially interesting, since they have the potential to be trained to become invariant to shifts, flips, and cut off sequences, which are common for automatically detected FCNs. In contrast, it is hard to model a handcrafted descriptor, which is robust to these properties. Inspired by the work of [LDCK18], where the application of LSTM networks on sequence data

achieves accurate results for the task of finding point-to-point correspondence, we propose to give our input data to an LSTM network for the task of finding curve correspondences. The architecture we use is a bidirectional LSTM cell (i.e. the sequence is processed forwards and backwards) with a 125 dimensional hidden state, combined with a linear layer, which maps the hidden layer to N_{label} dimensions (see Figure 3.3). We choose a batchsize of 10 to train the network. For the task of finding correspondences, we apply the negative log-likelihood loss discussed in Section 2.6.3. As optimization method we apply stochastic gradient descent with a learning rate of 0.001.

3.1.3.3 Limitations

While the learned feature curve representation shows high accuracies for the task specific geometry (cf. Section 3.1.4), a network has to be trained anew for each application. Learning the parameters of a network can take several hours. In contrast, descriptors as the HKS and GICS are computed within seconds. Furthermore, for each new task a labeled training set is required. Hence, learning feature curve correspondences in a supervised setting is useful for applications where a lot of data is available and we want to infer information from a subset of this data to the entire set or to new unseen data.

3.1.3.4 Unsupervised Learning

While handcrafted descriptors are less accurate than learned representations, they can be computed without requiring any labeled training data. Ideally, we would obtain high accuracy results without requiring labeled input. This can be achieved by designing a loss function, which does not rely on the target labels, but can evaluate the quality of the correspondences directly. In Chapter 5.2 we will present a method to compute inter-surface maps from curve correspondences at interactive rates. By evaluating the distortion of these maps (e.g. [ROA*13, HLR*18]) the quality of the curve correspondences can be rated. A possible direction how to obtain curve correspondences could consist of training a neural network which predicts a permutation matrix (e.g. based on *Sinkhorn iterations* [MBLS18] or

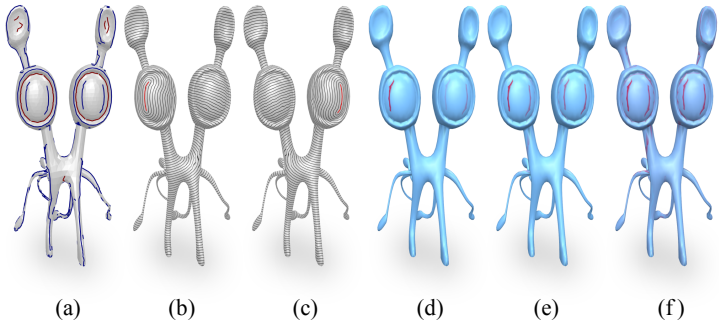


Figure 3.4: GICS visualized on feature lines. Geodesic isocontours are computed for radii growing around a curve segment. We computed feature curves with [YBS05] (a) and illustrate the iso-contours around a representative curve (b) and its best match in descriptor space (c). (d), (e), and (f) encode the distance values of the feature curves in descriptor space, where the maximal descriptor distance is cut off for increasing values (left to right). The red color indicates a high matching score (i.e. low descriptor distance).

sampling a permutation matrix with reinforcement learning). However, issues that can arise are vanishing gradients due to multiple applications of the *softmax* function as well as designing an appropriate loss function with an equal range of values for all shape pairs. We leave this highly interesting and challenging problem for future work.

3.1.4 Results

Figure 3.4 illustrates an example of the GICS on feature lines for a ”tele alien” shape. The initial feature lines are computed with [YBS05] (a). We computed the distance of the red feature curve in (b) to all other line features of the mesh and show its best matching feature curve in (c). (d), (e), and (f) encode the distance values, where we cut off the distance threshold at increasing values (left to right).

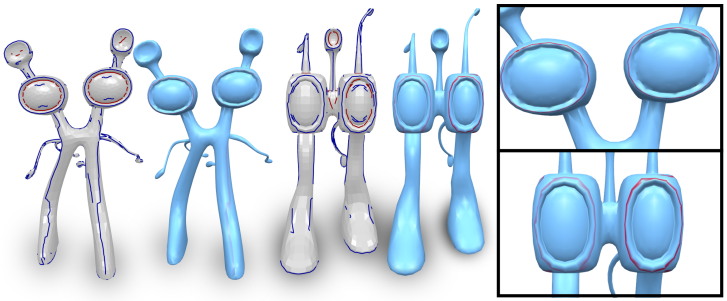


Figure 3.5: Across shape feature line matching. The circular shaped feature curve around the left eye of the first tele alien is selected as the source feature. The distance values to the other features are encoded in red (low distance) and blue (high distance). The best matches on the second shape represent circular arcs around the tele aliens eyes as well.

Furthermore, we show an example for across shape feature line matching in Figure 3.5. Here, the circular shaped feature around the left eye of the first tele alien is selected as the source feature. The distance values to the other features are encoded in red (low distance) and blue (high distance). The best matches on the second shape also represent circular arcs around the tele aliens eyes.

Comparison HKS and GICS In the following we present a quantitative comparison of the curve based HKS and GICS. Unfortunately, large scale data sets for evaluation of correspondence accuracies are only available for point correspondences. To overcome this issue, we compute the curve based HKS and GICS on feature curves detected on the FAUST [BRLB14] data set. The FAUST data set consists of a hundred meshes with ten humans, which were scanned in ten different poses each. For this data set, point-to-point correspondence information is given. We compute a set of 126 feature curves on the first mesh with [YBS05]. The corresponding feature curves on the other meshes are inferred from the point

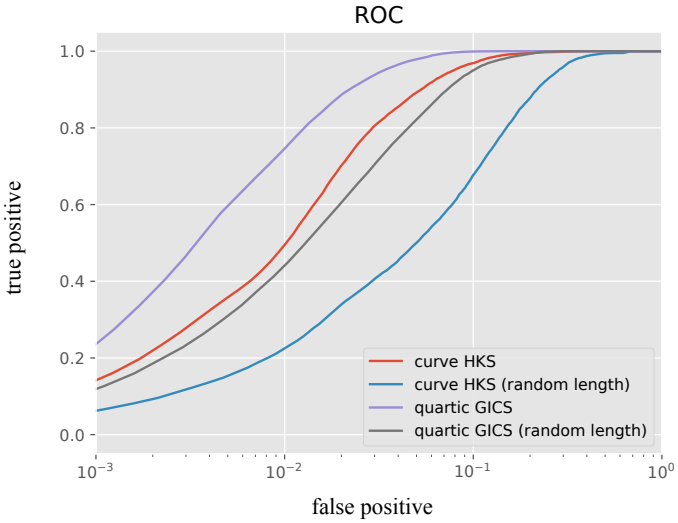


Figure 3.6: ROC on the FAUST inter data set for the curve based GICS (with sampling function $\ell(x) = x^4$) and HKS. The ROC (in log scale) is shown for correspondence tests with random length and full length curves.

correspondences. To add some additional variance, we randomly cut off up to a quarter of the curve at the beginning and the end. We evaluate the correspondence accuracy both on the full set of curves and on the curves of random length.

In Figure 3.6 we evaluate the *receiver-operator characteristic* (ROC), which measures the percentage of positive and negative matches. I.e. given two FCNs Γ_1 and Γ_2 for positive $c^+ = (c_1^+ \in C_1, c_2^+ \in C_2) \in C^+$ and negative matches $c^- = (c_1^- \in C_1, c_2^- \in C_2) \in C^-$, the false positive and true positive rates are measured as:

$$N_{tp} = \frac{|\{(c_1^+, c_2^+) \in C^+ | d(c_1^+, c_2^+) \leq t_\mathrm{d}\}|}{|C^+|}$$
$$N_{fp} = \frac{|\{(c_1^-, c_2^-) \in C^- | d(c_1^-, c_2^-) \leq t_\mathrm{d}\}|}{|C^-|},$$

Descriptor	FAUST Intra	FAUST Inter	FAUST Intra (random length)	FAUST Inter (random length)
GICS	0.96	0.9	0.68	0.58
HKS	0.76	0.64	0.44	0.37

Table 3.1: Results of Hungarian matching algorithm. We compute the GICS and HKS feature curve descriptors on curves detected on the FAUST data set.

where $d(c_1, c_2)$ measures the descriptor distance of the curves $c_1 \in \boldsymbol{C}_1$ and $c_2 \in \boldsymbol{C}_2$. The value t_d defines a distance threshold in the descriptor space, which is increased from zero to the maximal descriptor distance. Figure 3.6 shows the ROC for the GICS and the HKS with the full curves and the curves of random length. In this experiment we compute correspondences from the first shape to all other shapes. In general, a steeper ROC-curve indicates a higher matching performance. We can observe that in both cases the GICS computes more accurate curve correspondences.

Furthermore, we compute an optimal one-to-one assignment with the *Hungarian algorithm*. Given a distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$, which contains the descriptor distances, i.e. in our setting $(\boldsymbol{D})_{ij} = d(c_i, c_j)$, the Hungarian algorithm minimizes the following objective:

$$\begin{aligned} \min_{b_{ij}} \sum_i \sum_j (\boldsymbol{D})_{ij}\, b_{ij}, \qquad & i, j \in \{1, \dots, n\} \\ \text{s.t.} \sum_j b_{ij} = 1 \qquad & \forall i \in \{1, \dots, n\} \\ \sum_i b_{ij} = 1 \qquad & \forall j \in \{1, \dots, n\} \end{aligned}$$

where b_{ij} is a binary variable which indicates whether the curves c_i and c_j correspond ($b_{ij} = 1$) or not ($b_{ij} = 0$). In Table 3.1 we list the measured accuracies (i.e. the proportion of correct matches). We run the Hungarian algorithm on the FAUST intra (the first shape of each human is compared to its other poses) and on FAUST inter (the first shape of the data set is compared to all other shapes). Furthermore, we evaluate the accuracy on the curves of full length and those of random length. As in the previous test, we can observe that the GICS provides a

Set	test accuracy	validation accuracy	training accuracy	epochs
1 (random length)	0.78	0.8	1	3500
2 (random length)	0.83	0.78	1	3300
3 (random length)	0.8	0.8	1	4500
4 (random length)	0.78	0.78	1	3500
5 (random length)	0.78	0.78	1	3500

Table 3.2: 5-fold cross validation on the curves of random length of the FAUST data set. The sets are split into 60 training samples, 20 validation samples, and 20 test samples.

significantly higher accuracy than the HKS. The GICS shows a high accuracy of 0.96 and 0.9 for the full length curves. Matching curves of random length is far more challenging, still the GICS outperforms the HKS in terms of accuracy. Although less accurate, the HKS is computationally more efficient and can be useful if efficiency is the objective.

Learned Correspondences In Table 3.2 we show the accuracy of the curve correspondences which are learned with the LSTM network. We perform 5-fold cross validation, by splitting the FAUST data set with curves of random length into a training set (60 meshes), a validation set (20 meshes), and a test set (20 meshes) to show the robustness of our approach on different test sets. Hereby, in set 1 meshes $1-60$ are contained in the training set, $61-80$ in the validation set, and $81-100$ in the test set. In the next set we shift by 20 meshes such that meshes $21-80$ are contained in the training set, $81-100$ in the validation set, and $1-20$ in the test set. We repeat this procedure for a total of 5 sets for cross validation. From the results we observe that the measured matching accuracies are similar all for five partitions.

Comparison to Learned Correspondences In Table 3.3 we compare the results on the FAUST data set of the learned feature curve correspondences with the LSTM network (top) to the HKS and GICS (bottom table). To train the

Method	test accuracy	validation accuracy	training accuracy	epochs
LSTM (full curves)	0.91	0.91	1	6000
LSTM (random length)	0.78	0.8	1	3500
Method	**validation accuracy**	**test accuracy**	**validation accuracy (Hungarian)**	**test accuracy (Hungarian)**
GICS	0.74	0.9	0.91	0.91
GICS (random length)	0.48	0.52	0.6	0.65
HKS	0.54	0.48	0.37	0.47
HKS (random length)	0.33	0.31	0.37	0.4

Table 3.3: Comparison of learned correspondences between feature curves to via an LSTM network (top) with feature curve correspondences obtained from descriptor comparison of the GICS and the HKS (bottom). Correspondence accuracies are measured for the curves of full length and those of random length for the training and validation set.

LSTM network, we split the FAUST data into a training set (meshes 1-60), a validation set (meshes 61-80), and a test set (meshes 81 - 100). We show the measured accuracies (portion of correctly classified labels) for all three sets on the curves of full length and those of random length. The last column indicates the number of epochs required for training. The table below shows the accuracies computed with the feature curve descriptors HKS and GICS for the validation and test set. Here, accuracies are measured by comparing the curves of two shapes as in the tests above. Note, that since we only measure the distances of the first shape to all other shapes, the classification task the LSTM has to solve is slightly more complex. To find correspondences between two shapes we (columns 1 + 2) take the closest curve in descriptor space as correspondence and (columns 3 + 4) apply the Hungarian algorithm to obtain an optimal permutation as above. We observe that the task specific correspondences are significantly more accurate compared to the general descriptors, especially for the curves of random length. Finding correspondences for curves of random length with hand crafted features is particularly challenging since it requires partial matches of the curves. Here, a LSTM network which parses the sequence data can be very effective, since it is trained to be invariant to this type of variation in the curves.

3.2 Feature Curve Correspondences from Local-Global Voting

Due to the increasing accessibility of 3D scanners, a large amount of available geometric data is influenced by multiple forms of data uncertainty and inaccuracy:

1. *Geometric inaccuracy:* The measured (real-world) geometry is not equal at each perceived reoccurrence.

2. *Measurement uncertainty:* The measurements can suffer from noise.

3. *Missing data/coverage:* Depending on the scan position, not all parts of the geometry are scanned equally.

4. *Feature curve uncertainty/Classification noise:* In a local setting it is hard to evaluate whether automatically computed feature curves are merely noise or whether they represent meaningful parts of an object.

Hence, real-world objects require further processing to improve the surface quality and make them accessible to analysis. Co-occurrence information provides strong evidence how the surface can be completed and denoised. Especially, if feature curve correspondences are available, feature preserving surface enhancement algorithms can be applied. However, while FCNs obtained from man-made shapes are usually clean, real-world data is typically dense and noisy, leading to fragmented, jagged, and noisy feature curve networks.

Previous feature line based symmetry detection algorithms such as [BBW*09, BWM*11] rely on a preprocessing step which obtains a clean feature graph from the initial FCN by generating longer connected segments via local operations. In a setting with noise connectivity and filtering heuristics might not be valid because local measures are unreliable and the detected FCN can be very dense. Instead of altering the input curves, we present a robust technique to find reoccurring feature curves from the input FCN, without making strong assumptions about its structure.

Symmetry detection based on transformation space voting was proposed in [MGP06, PMW*08, SAD*16] (cf. Section 2.6). As stated above, these approaches strongly rely on the quality of the initial point matches. In the presence of noise, local point signatures tend to be inexact, hence the search space can become very large and dense, such that the false-positive rate in the set of point correspondences increases. This makes it hard to find structure in such a transformation space. With the feature curve information that is provided on an intermediate (more global) level, we are able to extend the previously presented global voting scheme to a robust local-global feature curve based transformation space voting. Therefor we propose a robust partial curve matching technique (cf. Section 3.2.2), which avoids the computation of unreliable point signatures and is thus more resilient to noise. Robust feature curve matching allows to decrease the false-positive rate of potential matches and simplifies the identification of reoccurring instances of the same geometry. Furthermore, considering only the points that lie on the feature curves of a shape greatly reduces the complexity of the search space.

3.2.1 Problem Statement

Given a surface $\mathcal{S}$ that is approximated by a mesh $\boldsymbol{M}$ with an automatically computed FCN $\Gamma = (V, E, C)$, our goal is to identify groups of similar feature curves that reoccur under the same rigid transformations $T_0, \dots, T_n$. Above, we discussed issues of the global transformation space voting presented in [MGP06], which arise due to the various sources of uncertainty. To avoid these, we need to pose a set of requirements:

1. *Avoid over-reliance on local decisions:* Local point descriptors are unreliable in the presence of noise, which makes it hard to find structure based on these (e.g. in the transformation space). We find correspondences on a more global level by taking the geometry of the feature curve into account.

2. *Soft feature curve identification:* We only leverage feature curves with strong evidence (which are long or which have high average curvature)

for the co-occurrence analysis. This avoids a high density transformation space deriving from classification noise.

3. *Partial feature curve matching:* Correspondences between feature curve pairs need to be described as partial matches (Section 3.2.2), since feature curves are traced locally and can thus be fragmented or overlap at different reoccurrences.

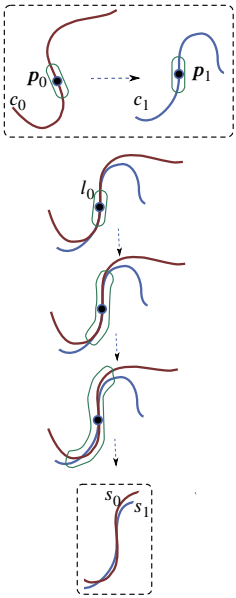
4. *Transformations clustering:* To identify transformations that appear multiple times, a cluster analysis is performed rather than testing for strict equality.

These requirements are met by a two-step local-global voting procedure. In the first (local) step, we identify similar feature curves (Section 3.2.2) by voting for the best transformations between the vertices of a pair of curves. In a second (global) step, similar transformations between entire feature curves are grouped into clusters (cf. Section 3.2.3). Finally, we detect orbits in the transformation space (i.e. identifying whether subsets of the reoccurring transformations can be described by a grid), as in [PMW*08, SAD*16].

3.2.2 Partial Feature Curve Matching based on Local Transformation Space Voting

Due to measurement and data uncertainty, it is not possible to find unique correspondences between entire feature curves. Since co-occurring feature segments can overlap, it is crucial to find partial matches between curves. This is a challenging task, since we would have to match all pairs of partial curves. To solve this problem, we present a novel ”sub-string matching” technique to identify partial matches between feature curve segments. Given a pair of feature curves $c_0 \in C$ and $c_1 \in C$, this two-step procedure first computes point correspondences between vertices on c_0 and c_1. In a second step, the partial matches are obtained from the point matches by voting for the most likely transformation between c_0 and c_1.

Feature BLAST In the presence of noise, local descriptors and respective point matches can become unreliable. Therefore, we exploit the more global information provided by the feature curve geometry to establish robust point- and partial curve matches. Hence, we avoid the computation of unreliable descriptors in the presence of noise. To obtain point correspondences between a point $\boldsymbol{p}_0$ on c_0 and $\boldsymbol{p}_1$ on c_1 (i.e. points lying on feature curves, see image below), we have developed a method which is based on a similar idea as the *Basic Local Alignment Search Tool* (BLAST) [AGM*90]. This method is used in genetics for DNA sequencing. Instead of matching DNA strands, we compute partial matches between feature curves. Algorithm 3.2.2 describes the steps of *Feature BLAST*.



As an input we provide the minimal match length $l_0 \in \mathbb{R}$ (which is encoded by the green ellipsoid in the image on the right) of two feature curve sequences as well as two hysteresis thresholds $t_{\text{low}} \in \mathbb{R}$ and $t_{\text{high}} \in \mathbb{R}$, with $t_{\text{low}} < t_{\text{high}}$. The lower threshold t_{low} evaluates whether an initial match was found (line 4 of Algorithm 3.2.2) by measuring the distance of two feature line segments s_0 and s_1 that are traced from $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ with the minimal match length l_0. The distance function is evaluated by translating $\boldsymbol{p}_0$ onto $\boldsymbol{p}_1$ and computing the best rotation between the two point sets. A match is only found if t_{low} is greater than the computed error. While the distance value d is less than the greater threshold t_{high}, the length of the match is extended by Δl (third image from the top) until either d exceeds t_{high} (fourth image from the top) or the feature curve ends. In our experiments we set the parameters l_0 to $20-30$ times the average edge length of the underlying triangle mesh, t_{low} to the average edge length, and $t_{\text{high}} = 2 \cdot t_{\text{low}}$. Δl (cf. Algorithm 3.2.2) is set to 5 times the average edge length. Note that we only use curves with strong evidence in this step, which are longer than l_0 or have

an average curvature along the curve which is higher than the $0.5 \cdot \kappa_{\text{avg}}$, where κ_{avg} denotes the average maximal absolute curvature, averaged only along the feature curves.

As a result, we receive point correspondences between the vertices of the two feature curves with the respective match length as well as an optimal rigid transformation and an affinity value (both are used to vote for partial feature curve matches in the next step):

$$\text{FeatureBLAST}(\boldsymbol{p}_0, \boldsymbol{p}_1) = l \cdot \exp\left(-\frac{d}{0.5 \cdot t_{\text{high}}}\right)$$

between the two points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ associated with the curve segments s_0 and s_1. In our experiments we compute this value for all pairs of points lying on the curves c_0 and c_1 and leverage these to vote for an optimal transformation between c_0 and c_1.

Voting for Transformations To find an optimal transformation for a pair of feature curves, we exploit transformations between the point correspondences found in the previous step, by letting the point correspondences vote for the best transformation. We represent each rigid transformation in form of a seven dimensional vector. The first three entrees compose the translation $(t_x, t_y, t_z)^\top$ and the last four represent the rotation in quaternion notation:

$$\left(\cos\left(\frac{\theta}{2}\right), n_x \sin\left(\frac{\theta}{2}\right), n_y \sin\left(\frac{\theta}{2}\right), n_z \sin\left(\frac{\theta}{2}\right)\right)^\top$$

with the rotation axis (n_x, n_y, n_z) and rotation angle θ. The 7-dimensional voting space is binned and the affinity values of the point matches whose transformations fall into the same bin are summed up. In our experiments we use bin sizes of twice the average edge length of the underlying geometry for the translations and an angle difference of 1.5° for rotations in unit quaternion representation (i.e. lying on the unit 3-hypersphere). We extract the maxima from the voting space as best transformations between two feature curves. Note, that this second step is only applied if a correspondence is detected in the previous step.

```
1:  function FeatureBLAST(l_0, t_low, t_high, p_0, p_1, Δl)
2:      l ← l_0
3:      distance ← distance(l_0, p_0, p_1)
4:      if distance < t_low then
5:          while distance < t_high do
6:              l ← l + Δl
7:              distance ← distance(l, p_0, p_1)
8:      return l · exp(−distance/(0.5 · t_high))
9:  function distance(l, p_0, p_1)
10:     s_0 ← CurveSegment(p_0, c_0, l)
11:     s_1 ← CurveSegment(p_1, c_1, l)
12:     s_0^t ← s_0 − p_0 + p_1
13:     R ← ICP(s_0^t, s_1)
14:     return averageEuclideanDistance(R(s_0^t), s_1)
```

Algorithm 3.1: FeatureBLAST - Hysteresis thresholding algorithm that computes the affinity of two points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ lying on two feature curve segments s_0 and s_1. A curve segment s_i of the curve c_i with length l starting at $\boldsymbol{p}_i$ is returned by the function CurveSegment($\boldsymbol{p}_i, c_i, l$). Furthermore, the optimal rotation between the curve segments is computed (similar to the iterative closest points (ICP) algorithm [BM92]). The hysteresis thresholds t_{low} and t_{high} bound the average Euclidean distance between the curve segments. If the distance between the transformed segment s_0^t and s_1 of initial length l_0 is less than t_{low}, an iterative growing procedure (the length of the segments is extended by Δl) is applied until the distance exceeds t_{max}.

3.2.3 Global Transformation Space Voting

Finally, to find reoccurrences of similar transformations, we need to cluster the transformations that derive from the previous (local voting) step over all pairs of feature curves. We use an agglomerative hierarchical clustering approach. As similarity measure we apply an average-link strategy. I.e. initially each transfor-

mation composes one cluster. As soon as two clusters are fused, a new average representative transformation is computed. This strategy has the benefit that it avoids $m \cdot k$ comparisons between all $m + k$ elements of the two merged clusters, especially in the light of large data sets. We stop clustering when the distance of the next element is larger than a predefined maximal error of the transformations (we use twice the average edge length of the underlying geometry for the translations and an angle of 1.5° between the unit quaternion representation of the rotations in our examples).

3.2.4 Orbit Detection

Reoccurring geometric entities that form a regular grid can be described by applying the same generating transformation T multiple times, i.e. $T^k = T \cdot T \cdot \cdots \cdot T$. The transformations $T^1, T^2, \ldots, T^k$ also appear as a regular grid in the transformation space. The goal of this step is to group these different transformations into one cluster. In our setting the grids can have a translational and a rotational component. Since all transformations derive from the generating transformation T, those with a rotational component need to be clustered according to their rotation axis. The generator is described by the rotation angle θ. In case of translations, the generators correspond to the translation vectors. Apart from 1D grids, combinations of different transformations can lead to 2D or 3D grids. In order to detect these orbits, we apply the method presented in [PMW*08], which we briefly outlined in Section 2.6.

3.2.5 Verification

From the previous step we know which feature curves are transformed under the same transformations. It is possible that curves reoccur under several different transformations. In Chapter 4.2 we will discuss how to select the best subset among the available explanations of the data and how to extract generating feature curve templates from these. Furthermore, we can identify which feature curves are reoccurrences of another by evaluating whether the curves share point

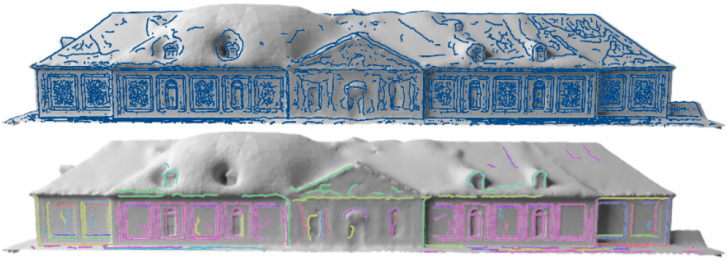


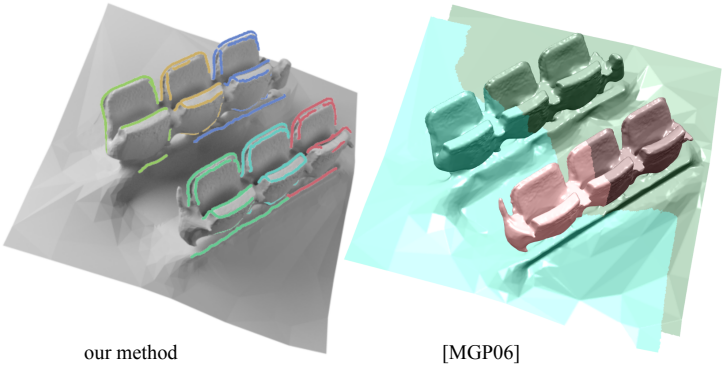
Figure 3.7: Scan of a manor house, with the initial dense FCN generated with [YBS05] (top). The bottom row shows the reoccurring feature curves in the same color. Note, that we are able to detect corresponding feature curves even though these are very jaggy given the high amount of noise in the data.

correspondences. Note, that it is also possible to verify this for the entire noisy set of curves. More details will be given in Chapter 4.2.

3.2.6 Results

In the following, we will show results of the co-occurrence analysis. Figure 3.7 depicts co-occurring feature curves on a scanned object with high amount of noise. Feature curves that share the same color represent reoccurring instances of the same set of curves (in this example we do not show the entire clusters of curves that are grouped to the same transformation). Note, that although the input feature curves are very dense and noisy, we are able to identify reoccurrences of the same curve configurations. More results will be given in Chapter 4.2.

Comparison In Figure 3.8 (left) we show results on a part of a scan from a theater. We compare our method to symmetry detection with [MGP06] in Figure 3.8 (right). With [MGP06] we find two translational (left to right) reoccurences of the chairs (green and red). The heat kernel signature [SOG09] is used as underlying descriptor for point matching. Our method benefits from the connectivity

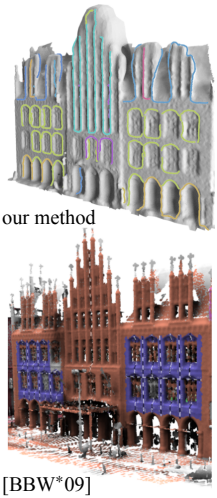


our method [MGP06]

Figure 3.8: Comparison to symmetry detection with [MGP06]. Two translational reoccurences (one in the front row and one in the back row, both left to right, red and green colored geometry indicate the two different translations) of the chairs have been detected with the approach of [MGP06]. Our method benefits from the detection of feature curve correspondences, which avoids the computation of unreliable local point descriptors (the underlying descriptor used for [MGP06] is the heat kernel signature [SOG09]). Our method (left) detects and divides the six instances of the co-occurring chairs exactly (each instance is indicated in a different color). This is crucial for the definition of a feature curve template.

of the feature curves, which allows to find each reoccurrence. Hence, our method detects all instances of the chairs and is robust to noise. In contrast, local descriptors can become unreliable under the influence of noise, which leads to a scattered transformation space in which it is hard to find meaningfull cluster centers. For [MGP06] we measured a computation time of 1511.31s for the point-signatures and 174.21s for the symmetry detection. With our method it took 74.59s.

We compare our results to the feature line based symmetry detection from [BBW*09] in the image on the right. In the top row we show curves that are grouped to the same cluster of transformations in the same color. Although we do not find all reoccurrenes of the yellow windows, we are able to separate the components more distinctly compared to [BBW*09]. This is crutial for tasks as template extraction and completion of FCNs, where more detailed information is required. Furthermore, we do not perform any preprocessing and use the input curves that are computed fully automatically with [YBS05].


our method

[BBW*09]

Furthermore, we compare our results to [BWM*11] in Figure 4.4 (3), (4), and bottom row. Berner et al. are able to find reoccurring feature curve groups of the windows of the church (purple lines) from the input FCN (yellow lines). However, the detection method relies on a similar topology of the reoccurrences. In the presence of noise this cannot be ensured, such that no feature curve correspondence is detected in this case. E.g. the topology of the pillars next to the windows varies (cf. Figure 4.4 (bottom row)), hence they are not included into the mapping. In these cases a user has to define corresponding points in the reoccurring instances. The feature curve input of our method can be very dense, jagged, and noisy, i.e. we do not make strong assumptions on the connectivity of the input FCN.

Limitations In cases with extreme noise levels where only very small fragmented feature segments are traced, our method will not be able to detect structure, since we will have no evidence for the co-occurrence analysis. I.e. BLAST will not find reliable matches, such that a transformation space clustering will not lead to reasonable results. However, in our results we show that our method finds a large set of reoccurrences even under a high influence of noise.

The reoccurrences found by this technique can be used for shape analysis tasks such as template extraction and completion, as we will show in Chapter 4.2. In contrast to previous work, our method detects reoccurrence information even under strong influence of noise. However, we only take rigid transformations into account. If shapes underlie isometric deformations, we cannot detect the reoccurrence. In the following we will present methods which handle these cases.

3.3 Summary

In this chapter we have presented a set of methods to compute correspondences between feature curves. Which method is most suited strongly depends on the application, the input data (resolution, noise), the amount of observed data (as well as the knowledge about this data, e.g. ground-truth), and which type of symmetries we expect. Correspondence and symmetry relations provide global structural information about the feature curve networks. As stated above, this is one important hint that distinguishes meaningful feature curves from noise. We will discuss this in more detail in Chapter 4. Furthermore, symmetry information encodes high-level structure (e.g. to complete feature curve networks, as we will show in Chapter 4.2.3), but also provides cues on a semantic level. Semantic correspondence information can be exploited in shape analysis tasks. We will demonstrate its usefulness exemplary for the computation of functional maps between highly non-isometric shapes in Chapter 5.2.

4 Meaningful Feature Curve Networks

Automatically detected FCNs are usually fragmented and contain a certain amount of *measurement-* and *classification noise*. Measurement noise occurs e.g. in the scanning process, when points are shifted in the viewing direction of the camera. In the FCN this measurement noise causes jagged curves and can lead to fragmentation. Classification noise covers curves which are labeled as features but are false positives as well as false negatives, i.e. feature curves that were neglected. These FCNs contain both meaningful and irrelevant information. For further processing and downstream applications it is therefore of interest to identify the meaningful set of curves and close gaps to avoid fragmentation. In previous work the relevance of feature curves has been evaluated by its local salience. However, to quantify the relevance of a curve, the feature curve network should be regarded as a whole. Here, we present two methods which evaluate the importance of curves motivated by global properties of the FCN. First, we will evaluate the meaningfulness of a curve based on its reoccurrence. Not only does it allow to identify prominent curves. By finding reoccurring curve segments, we are able to complete the feature curve network leading to less fragmented results. Secondly, we evaluate the meaningfulness of a feature curve based on a global scale parameter. Usually, geometric data is discretized with a specified resolution. We present a method which globally maximizes the amount of relevant feature curve information, which can be represented at a given scale.

Some of the material, techniques, and findings presented in this chapter have previously been published in [GLK16] (and the corresponding supplemental material) and in [GLK18].

4.1 Meaningful Feature Curves

There are various heuristics to evaluate the meaningfulness of a feature curve locally and globally. As described in Definition 2.1, we define $\Gamma = (V, E, C)$, where each curve $c \in C$ consists of a set of connected edges from $e \in E$. The relevance of a feature edge e can depend on several different properties. Hereby, the edges can be evaluated individually (single-edge quality criteria) or in combination with other edges (combined-edge quality criteria) [GLK16]:

- **Strength/Sharpness:** Most feature detection algorithms evaluate across edge curvature to specify the significance of a feature edge (e.g. for feature suppression). The factor $w_\kappa(c)$ measures the local sharpness by averaging $\kappa(\boldsymbol{p}) = \max\left(|\kappa_{\max}(\boldsymbol{p})|, |\kappa_{\min}(\boldsymbol{p})|\right)$, the maximum (across-feature) absolute curvature along the segment boundary polygon $\boldsymbol{p}_i, \ldots, \boldsymbol{p}_j$ which composes c:

$$w_\kappa(c) = \frac{\sum_{k=i}^{j-1} \|\boldsymbol{p}_{k+1} - \boldsymbol{p}_k\| \left(\kappa\left(\boldsymbol{p}_k\right) + \kappa\left(\boldsymbol{p}_{k+1}\right)\right)/2}{\sum_{k=i}^{j-1} \|\boldsymbol{p}_{k+1} - \boldsymbol{p}_k\|}.$$

- **Length:** In general, a feature edge is embedded in a connected feature curve and its length hints at the relevance of respective edges [DGGDV11]. Hence, we define $l(e)$, which measures the length of the feature curve, to which e belongs. A simple choice would be to take the length of the respective arc in Γ. However, this would ignore the fact that feature curves can geometrically (and "semantically") extend beyond extraordinary vertices. Several feature curve detection algorithms (e.g. [YBS05] and [ZZCJ14]) provide us with an appropriate segment membership. In the case of patch based methods (e.g. VSA), we compute the length of a supporting feature curve for each feature edge. Starting with the arc to which e belongs, we extend this curve at both ends by adding that arc, which continues the existing feature curve in the most straight (i.e. tangent continuous) direction. The extension stops when the straightest arc no longer corresponds to a boundary of the same patch as the starting arc. This stopping criterion

restricts the extension to T-type-vertices in the feature network and makes sure that each feature curve remains a subset of a patch segment boundary.

- **Closed loops:** A feature curve that is connected to a closed loop describes a perceptual component, which should be preferred:

$$w_{\text{Loop}}(e) = \begin{cases} \alpha_{\text{loop}} & \text{, if } e \text{ lies on a loop} \\ 1 & \text{, otherwise.} \end{cases}$$

- **Smoothness:** The geometric quality of the result can be improved by not only considering the weight of individual feature edges, but by also regarding pairs of edges. Smoothly connected segments convey the shape of an object better than highly curved segments and are considered more relevant [DGGDV11]. In order to promote connected curves over fragmented segments and interleaved edges approximating parallel curves, we introduce a smoothness measure $w_s(e_i, e_j)$. For this we consider all pairs of edges $e_i, e_j \in E$ which are adjacent to a common feature vertex. We choose:

$$w_s(e_i, e_j) = \begin{cases} \cos^p(2 \cdot \theta_{i,j}) & \text{, for } \theta_{i,j} \leq \frac{\pi}{4} \\ 0 & \text{, otherwise} \end{cases}$$

  where $\theta_{i,j}$ is the angle between e_i and e_j. $w_s(e_i, e_j)$ smoothly decreases from 1 to 0 as $\theta_{i,j}$ increases from 0 to 45 degrees. The parameter p controls the fall-off.

- **Orthogonality and parallelism:** For the purpose of quad-meshing, feature segments should be continued smoothly and intersect orthogonally to establish high quality meshes [BCE*13]. We can evaluate parallelism and orthogonality by introducing a combined weight $w_{||,\perp}$ between all edges that are within a prescribed radius:

$$w_{||,\perp}(e_i, e_j) = \cos^{2p}(2 \cdot \theta_{i,j})$$

  $w_{||,\perp}$ is close to 1 for angles near multiples of 90 degrees, which favors orthogonal configurations and parallel edge configurations.

- **Reoccurrence/Symmetry:** Symmetry is an important perceptual feature to identify a shape [MZL*09]. Hence, symmetric curve segments are more relevant than non-symmetric ones, i.e.:

$$w_{\text{Sym}}(e) = \begin{cases} \alpha_{\text{sym}} & \text{, if } e \text{ has symmetric edge } e' \\ 1 & \text{, otherwise} \end{cases}$$

  where α_{sym} is a constant factor. In contrast to the previous local properties symmetry is a global property and finding symmetric feature curves is a challenging task, which we discussed in Chapter 3. However, if symmetric feature curves are detected in a preprocessing step, this information can be used to detect relevant curves and complete the feature curve network.

- **Density/Scale:** Feature curves on surface meshes are usually defined solely based on local shape properties such as dihedral angles and principal curvatures. From the application perspective, however, the meaningfulness of a network of feature curves also depends on a global scale parameter that takes the distance between feature curves into account, i.e., on a coarse scale, nearby feature curves should be merged or suppressed if the surface region between them is not representable at the given scale/resolution. To maximize the amount of meaningful curve information, it is crucial to find a global optimum, which takes the relevance of the curves (i.e. previously listed properties) into account.

As discussed in Chapter 2, previous methods often resort to local solutions to extract meaningful curve networks and improve the quality by applying smoothing and local completion operations. In the following we will consider the global context given by the entire feature curve network to make decisions on the relevance of the curves.

4.2 Feature Curve Co-Completion in Noisy Data

Global co-occurrence information of feature curve groups provide valuable hints how to suppress noisy weak features and complete missing data in the FCN [GLK18]. A common (local) denoising technique is to smooth the surface or threshold the feature curves according to their curvature [BBW*09]. This also removes weak feature curves, which can be essential to describe the surface. Likewise, fragmentation is circumvented by locally reconnecting the curves. However, in a setting with noise where alot of feature curve information is given, it is often not clear which curves should be reconnected.

Feature curve configurations that reappear multiple times have a high probability of being meaningful, and are less likely due to noise. By detecting co-occurring groups of curves, this global feature information can be used to identify relevant curves and complete the FCN. This task is especially challenging due to various sources of measurement and data uncertainty. Due to this uncertainty, previous works in the field [SJW*11, BWM*11] rely on user input (e.g. user specified curve templates, or point correspondences) in the presence of noise.

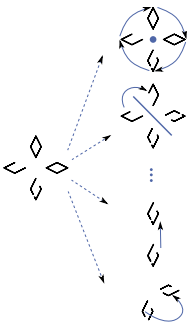
We present a fully automatic approach to feature curve co-completion in noisy data. Assume we are given a set of globally reappearing curve configurations (we discussed how to obtain these in Chapter 3). With these we are able to consolidate the co-occurrences to a completed feature curve template. A template which describes multiple reoccurrences of the same feature curves provides us with the means to complete the feature curve information as well as neglect sporadic short and weak features which are not observed multiple times. Furthermore, we obtain high-level shape information to describe and abstract the geometry.

4.2.1 System Overview

Our goal is to extract *feature curve templates* from a fragmented FCN computed on noisy data. We construct such templates by identifying reoccurring feature

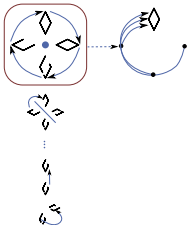
curves, which can then be grouped together. Due to the more global information encoded in a template, we are able to complete noisy FCNs. In the following we give a system overview of our approach.

Co-occurrence Analysis: In the *co-occurrence analysis*, sets of reappearing feature curves are identified, with the objective to find rigid transformations that generate reoccurring parts of the observed data. E.g. the image to the right shows the feature curves of a reoccurring rhomb. Several transformations (rightmost) can be used to explain parts of the data. We define a sub-set of such transformations as a model or a hypothesis that explains the given data.



We assume that we are given a set of describing transformations $\mathcal{T}_{\text{all}} = \{T_0, \ldots, T_n\}$ obtained from the co-occurrence analysis discussed in Chapter 3.2.

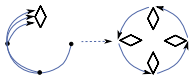
Feature Curve Co-Completion: In the second phase, the *co-completion*, a descriptive yet concise subset of these transformations needs to be selected. Since feature curves can be generated by multiple transformations, the goal is to extract a sparse subset of transformations which describes the set of reoccurring feature curves well. E.g. in the example to the right, the topmost 90° rotation around the blue rotation center is selected, since it provides the least redundant description of the data. The selected generating transformation can be used to group the reoccurrences into one template (rightmost).



If we had perfect knowledge of the reoccurrences of feature curves, we would be able to evaluate measures as the minimum description length [Ris83] for model selection. However, we have to identify which transformations describe the entire FCN containing weak and noisy feature curves best. Therefore, we

cannot simply minimize the description length of a set of models. Instead, we take the inherent uncertainty in the data into account and assign soft likelihood values to each feature curve, which describes how likely it is that the feature curve has been generated by a given set of transformations (model). With Bayesian model selection (cf. Section 2.2.1) we are able to identify a concise and descriptive set of models.

We use the generated templates to *complete* the feature curve information, by applying the inverse transformation and inserting the feature template at every location where a reoccurrence was observed.



4.2.2 Bayesian Model Selection

From the co-occurrence analysis we receive a set of reoccurring transformations $\mathcal{T}_{\text{all}} = \{T_0, \dots, T_n\}$ from the data $\mathcal{D} = C = \{c_1, \dots, c_m\}$. I.e. the data points are curves contained in the noisy FCN. We assume that the feature curves reoccur under a subset of these transformations (hypothesis). Hence, if a hypothesis is selected, we automatically select a subset of $\mathcal{T}_{\text{all}}$. The set of possible hypotheses $\{\mathcal{H}_1, \dots, \mathcal{H}_{|\mathcal{P}(\mathcal{T}_{\text{all}})|}\}$ has $|\mathcal{P}(\mathcal{T}_{\text{all}})|$ many entries, where $\mathcal{P}(\cdot)$ denotes the powerset. Due to the different sources of uncertainty in the data (the feature curves), we can mere assign soft likelihood values based on the observed evidence. In this probabilistic setting, Bayesian model selection allows to identify the model that maximizes the support for the data, as described in Section 2.2.1.

Given a set of models $\{\mathcal{H}_1, \dots \mathcal{H}_k\}$ that describe the data $C = \{c_0, \dots, c_m\}$ (the entire noisy FCN), we are able to choose a model that describes the data well while providing low redundancy based on Bayesian model selection. The output of this model selection step is a model $\mathcal{H}_i$ which is associated with a set of transformations $\{T_0^i, \dots T_q^i\}$ that assigns a transformation to each reoccurring feature curve and neglects the curves that are not supported by $\mathcal{H}_i$.

Figure 4.1: Scan of a museum (left) and a townhall (right), with (1) the initial dense feature network generated with [YBS05]. (2) shows the completed FCN. (3) depicts the connected components. (4) shows the feature curve groups that fall into the same template. Dark blue curves indicate the reoccurring templates, while strong (non-reoccurring) curves are visualized in light blue.

To evaluate the Bayes ratio $\mathcal{K} = \frac{p(\mathcal{H}_i|C)}{p(\mathcal{H}_j|C)}$ (cf. Section 2.2.1), we assume a uniform distribution as prior $p_{\mathrm{d}}(\mathcal{H})$ (all models are equally likely). We describe

the likelihood values of the feature curves given a model $\mathcal{H}_i$, i.e. $\mathscr{L}\left(c_k|\mathcal{H}_i\right)$ by testing whether the feature curve c_k was generated from other feature curves in $\boldsymbol{C}$ by applying one of the transformations $\boldsymbol{T}_j^i \in \mathcal{H}_i$. Therefor, we apply the inverse transformation $(\boldsymbol{T}_j^i)^{-1}$ for all $\boldsymbol{T}_j^i \in \mathcal{H}_i$ to the points $\boldsymbol{p}_{c_k}$ that compose the curve c_k ($\boldsymbol{p}_{c_k}^{\text{tr}} = (\boldsymbol{T}_j^i)^{-1} \cdot \boldsymbol{p}_{c_k}$). Then, we compute the average affinity value $w_{\text{affinity}}(\boldsymbol{T}_j^i, c_k)$ over the vertices of the feature curve. The affinity values are computed by Feature BLAST, which we introduced in Section 3.2.2, that computes the arclength l at which two curves c_k and c_k^{nn} are similar, starting at the points $\boldsymbol{p}_{c_k}$ on curve c_k and $\boldsymbol{p}_{c_k^{\text{nn}}}$ on c_k^{nn}. The affinity value is evaluated with the formula:

$$\text{FeatureBLAST}\left(\boldsymbol{p}_{c_k}, \boldsymbol{p}_{c_k^{\text{nn}}}\right) = l \cdot \exp\left(-\frac{d}{0.5 \cdot t_{\text{high}}}\right),$$

where d is the average distance of the two curve segments with length l. The corresponding feature curve for the BLAST matching is found as the curve on which the nearest neighbor $\boldsymbol{p}_{c_k^{\text{nn}}}$ of $\boldsymbol{p}_{c_k}^{\text{tr}}$ lies. In case the transformation describes a grid, we sum up the average affinity values for all reoccurrences.

We add $\boldsymbol{T}_{null}$ to each model, which encodes that a feature curve was not generated by any of the given transformations, and set

$$w_{\text{affinity}}(\boldsymbol{T}_{null}, c_k) = l_0 \cdot \exp\left(-\frac{t_{\text{high}}}{0.5 \cdot t_{\text{high}}}\right) = l_0 \cdot e^{-2},$$

which is the smallest possible affinity value if the minimum match length l_0 is met. The likelihood of feature curve c_k given the model $\mathcal{H}_i$ is then computed as

$$\mathscr{L}\left(c_k|\mathcal{H}_i\right) = \max_{\boldsymbol{T}_j^i \in \mathcal{H}_i} w_{\text{affinity}}(\boldsymbol{T}_j^i, c_k). \tag{4.1}$$

For numerical stability, all computations are performed in log-space.

Note that after model selection, point correspondences on the reoccurring curve instances are automatically given by the association of $\boldsymbol{p}_{c_k}$ and $\boldsymbol{p}_{c_k^{\text{nn}}}$. In case the curves on which $\boldsymbol{p}_{c_k}$ and the reoccurrences $\boldsymbol{p}_{c_k^{\text{nn}}}$ lie are assigned to the same transformation and the corresponding BLAST score exceeds the matching threshold t_{low}, the points $\boldsymbol{p}_{c_k^{\text{nn}}}$ are considered as reoccurrences of $\boldsymbol{p}_{c_k}$.

4.2.2.1 Heuristic Model Selection

It is infeasible to compare all possible models ($2^{|\mathcal{T}_{\text{all}}|}$ many). Hence we propose a heuristic to neglect redundant transformations. We start with the model that holds the entire set of transformations, i.e. $\mathcal{H}^0 = \mathcal{T}_{\text{all}}$ and recursively remove transformations from this set until the Bayes ratio $\frac{\mathcal{H}^i}{\mathcal{H}^{i+1}}$ between the models $\mathcal{H}^i$ and $\mathcal{H}^{i+1}$ shows strong support for the model $\mathcal{H}^i$ (i.e. $\mathscr{K} > 10$). From a subset of models with lowest Bayes ratio we greedily remove the transformation with least support for the data (i.e. with minimal $\mathscr{L}(C|\mathcal{H}) \cdot p(\mathcal{H})$). Note that T_{null} is excluded from this removal. If $\mathscr{K} > 10$ after removing a transformation we know that the model $\mathcal{H}^i$ was of significance. Hence, we can find a descriptive model (w.r.t. the definition of [Jef98, KR95]) with $\mathcal{H}^i$ after at most $|\mathcal{T}_{\text{all}}|$ iterations (since one transformation is removed in each iteration), i.e. after at most $|\mathcal{T}_{\text{all}}|^2$ computations of the Bayes ratio.

4.2.3 Feature Curve Co-Completion

In the final FCN, feature curve classification noise is discarded (Section 4.2.3.1) and reoccurring groups of feature curves are grouped into templates, which are completed based on their co-occurrence information (Section 4.2.3.2). The result of Bayesian model selection is a set of transformations $\mathcal{H}_i = \{T_0^i, \dots, T_q^i, T_{null}\}$ with an assignment of each feature curve c to one of the transformations T_c in $\mathcal{H}_i$. T_c is selected as arg max for equation (4.1). The mapping $B_{T_c \in \mathcal{H}_i} : C \to C$ is then defined between pairs of feature curves under the selected transformation.

4.2.3.1 Feature Curve Classification Noise Removal

Curves that are assigned to the model T_{null} do not reoccur with respect to one of the transformations in the selected model $\mathcal{H}_i$. Hence, we assume that weak non-reoccurring curves were classified as false positives by the feature detection method. These are removed from the network. Only significant curves are retained. For these non-reoccurring curves, we set a high threshold so that no classification noise is added (curves that are longer than $3 \cdot l_0$ times the average

edge length or have an average curvature higher than κ_{avg} (the absolute maximal curvature averaged along all the curves) and are longer than l_0 times the average edge length). In our results we indicate these curves in light blue.

4.2.3.2 Template Generation

The remaining curves correspond with respect to one of the transformations $T_0^i, \ldots, T_q^i$. These are grouped into templates, which are extracted for each of the transformations $T_0^i, \ldots, T_q^i$, separately.

Connected Component Computation First, all feature curves that belong to the same component are identified. We define a graph $\mathcal{G} = \{C, E\}$, where C represents the set of feature curves (the nodes) and $E = \{(c, B_{T_c}(c)) | c \in C\}$ the set of undirected edges. In this graph, nodes are connected by an edge if and only if their corresponding curves are mapped onto each other by the transformation T, i.e. if point-correspondences exist on the respective curves (cf. Section 4.2.2). Hence, the connected components of this graph represent reoccurrences of the same feature curve. We show the connected components as third image in Figures 4.1-4.4. Note that these (exact) feature curve correspondences with point-to-point reoccurrence information are essential for the FCN completion step (otherwise it would be hard to separate and complete close curves individually).

Connected Component Completion To compute the templates, we overlay the reoccurrences of each connected component by applying the inverse transformation. To consolidate the curves (i.e. select one alternative of the multiple reoccurrences) we implement a shortest path search on the possible (reoccurring) paths through the FCN. Note that for this step the identification of corresponding feature curve segments is crucial (i.e. previous related symmetry detection methods do not provide this information). Only those segments where point-correspondences on other instances exist are considered in the graph. To identify alternative paths, we project the end-points of the same connected components onto their nearest neighbors of the reoccurrences in an ϵ-radius. The new start-

and end-points of this network are all vertices that are projected to end-points only (not onto an intermediate point of the feature curve). In this network we iteratively find the longest *shortest path* from one start-point to an end-point and remove all redundant paths.

Grid Origin Computation Different connected components that are assigned to the same transformation and are seeded at the same origin should be grouped into one template. The transformations alone do not provide information on where detected grids are seeded. In the following we describe a method to locate the origins by using the curve information only.

In case we have detected a grid in the orbit detection, a seed point for the grid needs to be found. To find this point, we project the feature curves belonging to the same connected component onto their generators. In case the grid is spanned by two translation vectors $\boldsymbol{g}_0$ and $\boldsymbol{g}_1$ and these are perfectly orthogonal, we can simply project onto $\boldsymbol{g}_0$ and $\boldsymbol{g}_1$. Otherwise, we need to account for the shear and compute two projection directions $\overline{\boldsymbol{g}}_0$ and $\overline{\boldsymbol{g}}_1$:

$$\begin{aligned} \boldsymbol{n} &= \boldsymbol{g}_{0_n} \times \boldsymbol{g}_{1_n} \\ \boldsymbol{g}_0^{\perp} &= \boldsymbol{g}_{1_n} \times \boldsymbol{n} \\ \overline{\boldsymbol{g}}_0 &= \langle \boldsymbol{g}_0, \boldsymbol{g}_0^{\perp} \rangle \cdot \boldsymbol{g}_0^{\perp} \end{aligned}$$

where $\boldsymbol{g}_{i_n}$ are the normalized generators, $\times$ denotes the cross product and $\langle \cdot, \cdot \rangle$ the dot product between the respective vectors. The same computation can be performed for $\overline{\boldsymbol{g}}_1$. In case the generator is a rotation angle, we project the feature lines onto a circle around the respective center of rotation. The projection directions (or circle) are binned and the amount of projected curves is stored in these bins. The optimal offset for the grid with generator $\boldsymbol{g}$ is found by computing the minimal value for the offset index z:

$$\min_z \sum_{i=0}^{k} \text{binentry}(z + i \cdot \parallel \boldsymbol{g} \parallel).$$

Connected components that fall into the same grid are then grouped together into one template. Furthermore, this allows us to identify gaps in the grids, which

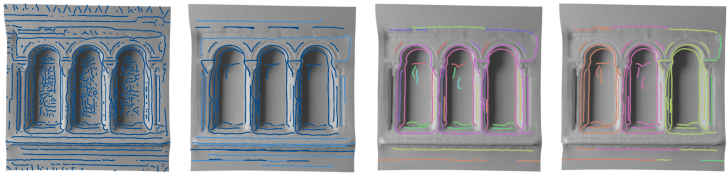


Figure 4.2: Our method takes a noisy feature curve network (FCN) as input and produces a completed and denoised output FCN. This is done by performing co-occurrence analysis on the feature curves with which we identify corresponding curves and omit noise. Reoccurring groups are merged into a template, which is used for co-completion of the network. In the completed network, dark blue curves indicate the reoccurring templates, while strong (non-reoccurring) curves are visualized in light blue. The example shown above is computed on an excerpt of a scan of a gothic cathedral.

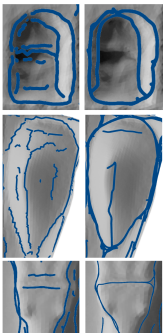
are detected if at least two neighboring grid positions contain no evidence of the curve template. At these points we split the grid for the template extraction. In the fourth images of Figures 4.2 and 4.3 we encode the curves that were assigned to the same grid cell (i.e. reoccurring group of curves) with the same color. The final templates are obtained by transforming the completed connected components that are assigned to the same template into a common grid cell with respect to the defined grid origin.

For the resulting FCN we copy the computed templates to each position where we have measured an occurrence of the respective template. In addition, strong non-reoccurring curves are added to the network as described in Section 4.2.3.1.

4.2.4 Results

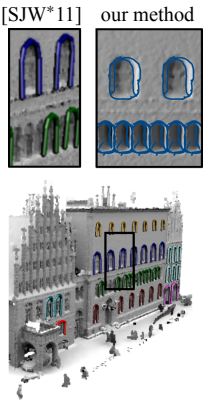
In the following, we will show results of the feature curve co-completion. Figures 4.1-4.4 (1) show the dense input feature curves. Note that all FCNs are

generated on noisy point cloud data. The initial curves are generated automatically with the method presented in [YBS05]. (2) visualizes the completed FCNs. To complete the network, we insert the extracted templates at each detected reoccurrence (dark blue) as well as strong features, which were not detected as part of the reoccurring configurations (light blue) into the network. (3) shows the connected components (described in Section 4.2.3.2). Figures 4.2 and 4.3 (4) depict the groups which where assigned to the same grid position. In Figures 4.1 and 4.4 (4) we encode the feature curves that are grouped into the same template in the same color.



With our method, we are able to identify co-occurring configurations of feature curves and use this information to complete the FCNs. E.g. in the images above we show close-ups of the original noisy and fragmented curves (left) of the models. Our approach is able to complete significant curves and remove the noisy features (right).

Feature-Completion In the image on the right we compare with the supervised feature line learning approach presented in [SJW*11]. The feature completion approach by Sunkel et al. requires a user to draw the curve templates onto instances of the geometry. The system automatically finds further reoccurrences of the templates. In contrast, our method is unsupervised and finds the reoccurring templates automatically. See the figure on the right (top right) for comparison.



We can also take the method proposed in [BWM*11] into account for completion. However, their input FCN only contains high curvature feature lines. This is sufficient to detect the co-occurrences, but to complete the FCN, the identification of corresponding weaker curves is crucial. In contrast, we analyze the entire FCN,

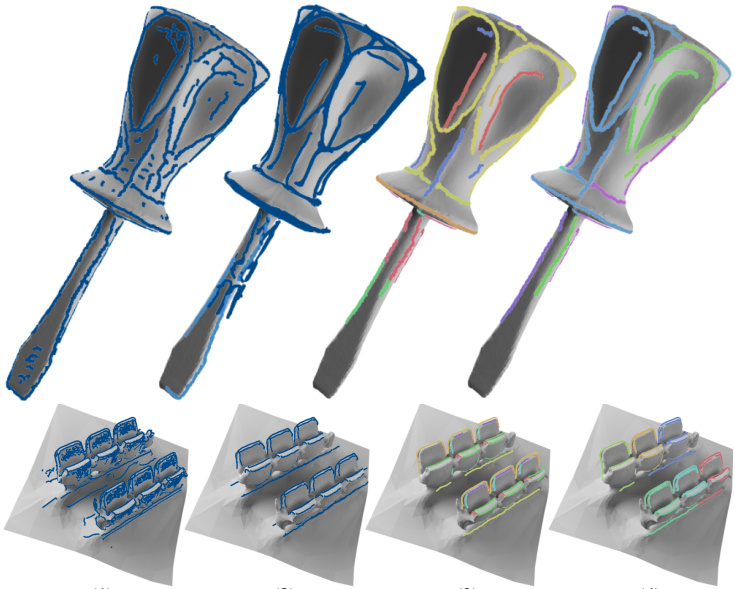


Figure 4.3: Screwdriver (top) and scan of theater seats (bottom), with the initial dense feature network generated with [YBS05] (1), the FCN which was completed with our method (2), the connected components (3), and the feature curve groups that fall into the same orbit (4). Dark blue curves indicate the reoccurring templates, while strong (non-reoccurring) curves are visualized in light blue.

since this is relevant to include weak and strong feature curves into the templates for completion (cf. Figure 4.4).

Curvature Threshold A local method to denoise feature curves is to threshold these, e.g. based on their curvature. An example is given in Figure 4.5, which shows an excerpt of the museum depicted in Figure 4.1. By increasing the curvature threshold, the number of curves is reduced. Note that even in the example

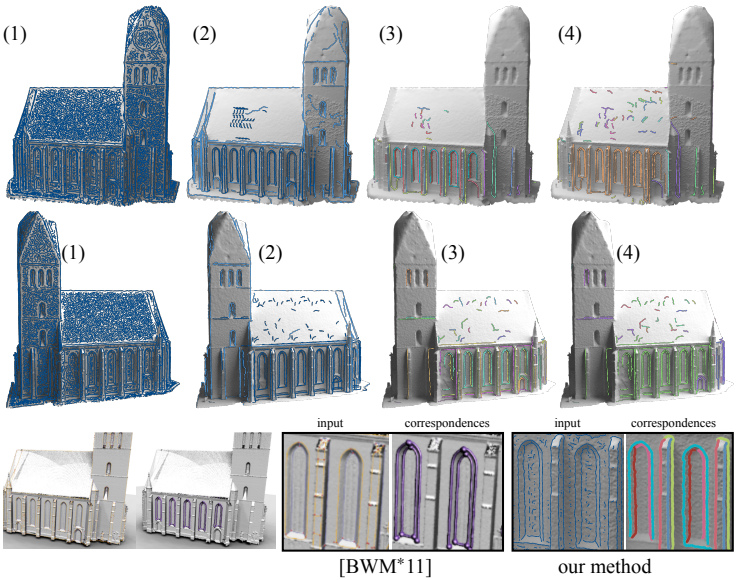


Figure 4.4: Comparison to [BWM*11]. The top and middle row show results of our feature curve co-completion with (1) the input FCN, (2) the co-completed FCN, (3) the connected components, and (4) the feature curve groups that fall into the same template. The bottom row depicts the results of [BWM*11]. Note that our method does not rely on topological correspondence of the curves and does not require a user to select reoccurrences in the presence of noise.

with the highest curvature threshold (column 4) not all noise is removed, while relevant curves are missing. I.e. usually it is not possible to set one threshold for the entire geometry. With our method (rightmost), relevant features are detected by co-occurrence. Additionally, reoccuring groups are completed (e.g. the missing part of the window in the bottom right).

Figure 4.5: Comparison of local feature curve denoising with three different curvature thresholds to our method. By discarding the input feature curves (left) based on their average curvature, weak but relevant features are suppressed before all the noise is removed. The co-occurrence analysis in our method (right most) allows to identify reoccurring curve configurations and complete feature curves that are not in the input feature set as well as neglect noisy features that do not reoccur.

Feature Curve Input In Figure 4.6 we demonstrate that the method is flexible with respect to the FCN input. We compute three different kinds of FCNs (for which the software is available online): [YBS05], CGALs [CP05] implementation, and [OBS04] with the curvature computation [Rus04]. In each example, the reoccurring window configurations are completed similarly.

Limitations It can occur that a detected reoccurrence falls into a subgrid. E.g. the handle of the screwdriver in Figure 4.3 has repeated feature curves in a 60˚ degree angle, while the shaft repeats in a 180˚ angle. The 180˚ grid falls into the 60˚ grid. So when the feature curves are transformed back for completion, they are repeated at every grid location. Another limitation derives from very jagged curves. In this case the feature curve consolidation can fail and multiple arcs representing the same feature occur in the template. Since we assume that feature curves are arbitrarily shaped, this cannot be avoided in our setting without making heuristics about the shape of the curves.

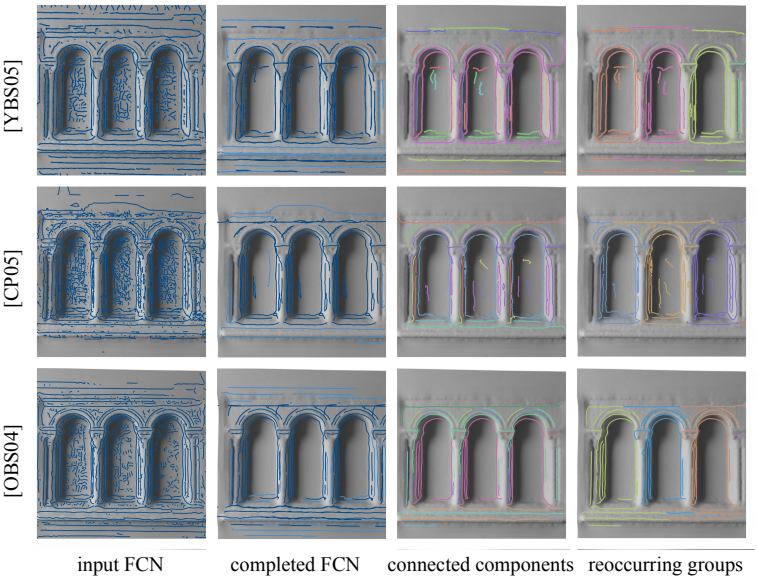


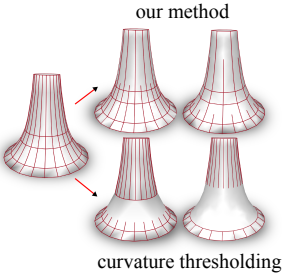
Figure 4.6: Different feature detection methods. Initial feature curves are computed with [YBS05], [CP05], and [OBS04]. Although the input curves are different, similar connected components and reoccurring groups are detected.

The feature curve networks that result from our procedure contain only meaningful information, including globally reoccurring curve configurations and locally salient (long and high curvature) curves. These networks can be applied in various geometry processing tasks, e.g. to apply feature preserving remeshing or smoothing of the noisy geometry. The structure information can be used to complete the geometry and to support various shape analysis tasks such as retrieval, procedural modeling, or segmentation, which are more challenging on real-world data. Our method provides valuable input to these approaches in the presence of noise.

4.3 Adapting Feature Curve Networks to a Prescribed Scale

Typically, geometry processing algorithms as well as application areas like simulation, medical applications, and visualization techniques are provided with a surface representation at a user-specified target edge length or complexity. The original raw data is often too dense, which can affect run-time and storage restrictions. Depending on the application, complexity constraints can range from noise suppression, to removing small scale details (e.g. to reduce computation times in simulations), or to depicting up to very coarse resolutions of the mesh (e.g. when the image resolution or available storage is low). Various processing techniques exist that establish good element quality at the required complexity (e.g. [ESCK16, ECBK14, CJL11, BZK09, BK04, ADVDI03]) from raw input data. The surface approximation is considered high quality if elements have a high regularity and align to surface features.

The required features or guiding constraints need to be computed in advance and are given as input to the respective methods. As discussed above, current feature curve extraction approaches (e.g. [NSP10, WG09, YBS05, OBS04]) only take local shape properties into account, where the number of selected feature curves is controlled by filtering or by setting thresholds on curvature values.



However, none of the previously mentioned feature extraction algorithms take the spatial feature density into account. Suppose there are two feature curves nearby, one with a relatively high ”sharpness” and the other one with lower. Then a filter and threshold approach would be able to suppress the less sharp feature. At the same time, however, it would also unnecessarily suppress other feature curves with the same (lower) sharpness even if they are not close to another feature

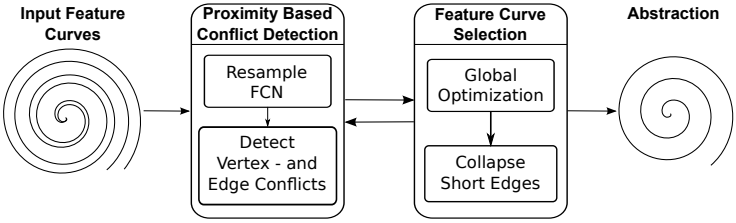


Figure 4.7: Overview over our feature curve network adaptation scheme: As input we receive an initial feature network. In the *proximity based conflict detection*, we check whether the network conforms to the prescribed scale. For this we have to find a discrete representation via resampling. We can then describe scale-conformity as vertex and edge conflicts. In the *feature curve selection* the detected conflicts are resolved. Since edges are weighted according to their relevance, we resolve conflicts between edges by suppressing the less significant edges in a global optimization step. Remaining edges that are shorter than the minimal scale are collapsed. This can change the network topology, which requires further conflict detection. We iterate this process until no conflicts exist and the transformed feature network conforms to the prescribed scale.

(see image above). This example clearly shows that feature selection is a global optimization problem and the decision to suppress a feature should be dependent on the existence of other features in the vicinity. Hence, using the aforementioned mechanisms, we cannot preserve the maximal set of feature curves representable at a given resolution.

In this section we present an approach that adapts FCNs to a prescribed scale by looking at the network as a whole and not treating each feature curve separately, i.e., we preserve all features which can be represented at the given target resolution, irrespectively of their sharpness.

4.3.1 Problem Statement

Conceptually, our scale adaptation approach is quite different from previous ones. Existing approaches that are based on a (pre-)filter operation eliminate features by locally blurring the sharpness until it falls below a given threshold. Hence, the decision whether a feature curve is eliminated is made just locally without considering other features in the vicinity. This leads to situations where more features than necessary are removed, since even a less pronounced feature (which is smoothed out by the filter) could be valuable on a given scale if neighboring features are sufficiently far away. In contrast to this local filter approach, we are following a globally optimal sub-sampling approach where feature curves are eliminated only if there is another (more relevant) conflicting feature curve nearby.

Given an initial set of feature curves on a surface mesh $\boldsymbol{M}$, we generate abstractions in form of FCNs, which are guaranteed to comply with a prescribed global proximity constraint. This constraint is given in form of a scale parameter r_{min}, which limits the target resolution (e.g. the minimum edge-lengths of the respective approximation). To extract the final FCN we follow the workflow depicted in Figure 4.7, which consists of two main building blocks.

The first module involves a *proximity based conflict detection* procedure, which identifies curve segments that cannot coexist at the given scale. In this continuous setting, finding appropriate segments is intractable (because there exist infinitely many possible segments). Thus, we discretize this problem by introducing the definition of a *scale conforming* FCN in Section 4.3.2, which resamples feature arcs into scale conforming segments. This discretization allows for the identification of conflicting curve segments (which we denote as edges of the FCN in the following) and the formulation of an objective function measuring the quality of the edges.

The identified set of conflicting features serves as input to the second component: the *feature curve selection*. Feature selection is performed with the goal

to best preserve the structure of the initial network. In particular, we resample features such that the geometric error is minimal and remove a minimal set of edges so that the remaining (non-conflicting) edges preserve as much as possible of the original feature network.

Due to the discrete representation of the feature curves, we are able to evaluate the relevance of each edge. Since the requirements imposed upon a feature curve network are highly application dependent, we allow the user to combine different relevance criteria. For this we use the salience measures introduced in Section 4.1. The first set of criteria evaluates the quality of a *single* edge based on its (mostly) local properties: across edge *sharpness*, *curve length*, *symmetry*, and whether the feature edge belongs to a closed *loop*. Secondly, we also regard *combined* quality criteria, which rate pairs of edges according to their *smoothness*, *orthogonality*, and *parallelism*. Exact details will be described in Section 4.3.3.3. Depending on the application, any other combination of criteria can also be integrated into our approach (e.g. different saliency based edge weights).

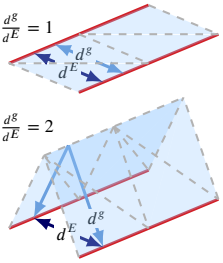
For *feature curve selection*, the described quality criteria are incorporated in a functional. In a global optimization scheme, we select those curve segments of the FCN which maximize this function, while abiding proximity constraints.

4.3.2 Scale Conforming Feature Curve Networks

State of the art remeshing algorithms produce high quality surface approximations based on a set of features $\Gamma = (V, E, C)$ which is provided as input. Moreover, a target scale can be prescribed to control the mesh complexity. In order to abide to such a density constraint, mesh elements can only align along a subset of surface features which are not contradictory to the prescribed target edge length. We denote such a subset of Γ as a *scale conforming FCN*.

The input scale parameter r_{min} provides a lower bound for the sampling rate across the continuous surface defined by $\boldsymbol{M}$. Hence, a remeshing procedure

should adapt the vertex positions to this lower bound, such that all edges are longer than $r_{\min}$. Accordingly, we denote a FCN as *scale conforming* to a given scale $r_{\min}$ if it can be extended to a manifold triangle mesh $\boldsymbol{M}_{sc}$ (by adding vertices and edges) that fulfills the following conditions:



1. Each edge e of $\boldsymbol{M}_{sc}$ has its length $\|\boldsymbol{e}\|_2 \geq r_{\min}$.
2. The ratio of geodesic distance d^g_M to Euclidean distance d^E_M between nearby feature curves is bounded by some $q < 2$. This condition ensures that the surface $\boldsymbol{M}_{sc}$ approximates $\boldsymbol{M}$ well, avoiding high frequency surface oscillations between feature curves. E.g. the image above shows two triangulations between the (red) feature curves. In contrast to the upper triangulation, the lower one introduces undesired high frequency detail.

This definition of a scale conforming FCN serves as a conceptual goal that the network should achieve, according to which we derive conflict detection and suppression schemes for feature edges in the following. These are required to find a maximal subset of feature curves, which are scale conforming.

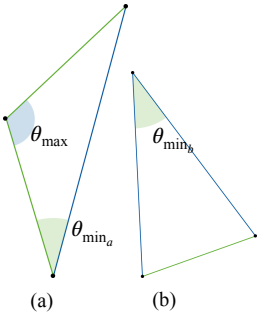
4.3.2.1 Quality Criteria

The extension $\boldsymbol{M}_{sc}$ of the scale conforming FCNs should be able to fulfill quality criteria that state of the art remeshing algorithms optimize for, in order to obtain high element quality as well as feature alignment. In the following we will give details on which aspects need to be taken into account for scale conformity that abides with high element quality.

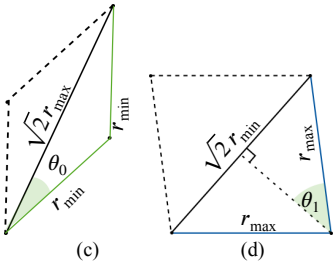
Mesh Quality The triangles that are adjacent to the scale conforming FCN should have sufficient quality. If we allow arbitrarily long edges, degenerate triangle configurations can occur. We can limit their aspect ratio by providing an

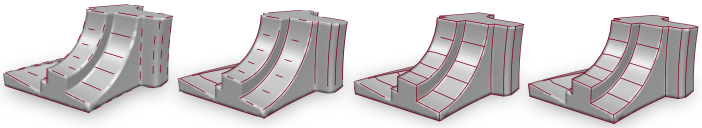
upper bound $r_{\max}$ on the edges of the scale conforming FCN, bounding aspect ratios to $\left[1, \frac{r_{\max}}{r_{\min}}\right]$. Thus, to ensure triangle quality, we define a scale interval $S = [r_{\min}, r_{\max}]$ so that all edges of the FCN have their length in S.

Feature Edge Angles The upper and lower bounds on the edge lengths also impose upper and lower bounds $[\theta_{\min}, \theta_{\max}]$ on the inner angles of each triangle (see image below). Consequently, each pair of adjacent edges in E has to enclose an angle from $[\theta_{\min}, \theta_{\max}] \cup [2\,\theta_{\min}, 2\,\theta_{\max}] \cup [3\,\theta_{\min}, 3\,\theta_{\max}]$. This set of feasible angles may consist of one, two, or three intervals, depending on whether $\theta_{\max} < 2\,\theta_{\min}$ or even $2\,\theta_{\max} < 3\,\theta_{\min}$. The smallest maximal angle in a triangle is $\frac{\pi}{3}$. Hence, $3 \cdot \theta_{\max} \geq \pi$ is the maximal possible angle between two edges, so we do not need to consider any further intervals. Depending on the scale interval S, the smallest representable angle $\theta_{\min}$ is either found in triangle type (a) with $\theta_{\min_a} = \arccos\left(r_{\max}/2r_{\min}\right)$ or in triangle type (b) where $\theta_{\min_b} = 2 \cdot \arcsin\left(r_{\min}/2r_{\max}\right)$. If $r_{\max}/r_{\min} < 1.618$ the angle in triangle (b) is smaller.



Feature Edge Angles for Quad Meshes We can further extend this definition to apply to quad meshes. In this case the diagonal should not be represented in the FCN. Therefore, the minimal and maximal angles have to be adapted in the following way: Since we want to constrain the diagonal to have a length in the interval $[\sqrt{2}r_{\min}, \sqrt{2}r_{\max}]$, we can derive the minimal and maximal angles from the two configurations shown to the right. Thus, the minimal angle is $\theta_{\min} = 2\min(\theta_0, \theta_1)$. The maximal angle can be derived analogously by switching $r_{\min}$ and $r_{\max}$ in configurations (c) and (d).





1.0 1.14 1.35 1.5

Figure 4.8: Effect of constraining FCNs to different aspect ratios for the fandisc at the same scale. The selected aspect ratios $\frac{r_{\max}}{r_{\min}}$ are given below. Aspect ratios close to 1 constrain angles of adjacent features to be close to multiples of 60 degrees. We found aspect ratios of 1.5 to give a good tradeoff between triangle shape and representable angles. For values greater than 1.5 no further changes would be visible in this example.

Aspect Ratio The range S should be chosen such that a sufficient quality of the output mesh M_{sc} is guaranteed while not being to rigidly confined to (close to) equilateral triangles. Figure 4.8 demonstrates the effect of constraining the upper bound of edge-lengths in abstractions of the fan-disc model. The first two FCNs with aspect ratios of $\frac{r_{\max}}{r_{\min}}$ close to 1 lead to gaps in the feature lines because the angles between the line-segments are not representable in the angle intervals. For ratios $\frac{r_{\max}}{r_{\min}} > 1.365$ the representable angle intervals merge (i.e. $\theta_{\max} < 2\theta_{\min}$), which allows to represent a broad range of angles. In practice, we found that $\frac{r_{\max}}{r_{\min}} = 1.5$ is a good choice since it includes 'triangulated quad meshes' in the set of acceptable output meshes (because the lengths of the diagonals in a triangulated quad mesh are about $\sqrt{2}$ times the lengths of the quad edges).

4.3.2.2 Achieving Scale Conformity

The definition of a FCN conforming to a given scale gives rise to a number of conditions that Γ has to satisfy, especially when fulfilling the quality criteria discussed above. First of all, Γ conforming to $r_{\min}$ trivially implies that all edges in E have to be longer than $r_{\min}$. Furthermore, constraining the aspect ratio of the triangles that are adjacent to E implies that all edges have their length in S.

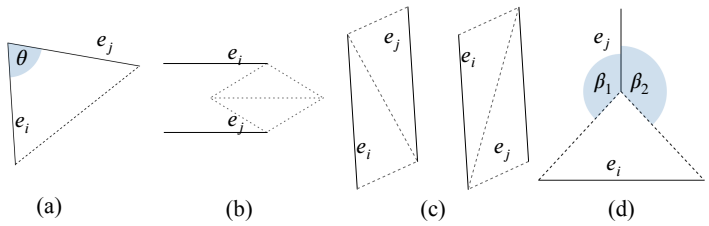


Figure 4.9: Conflict detection: (a) shows the case where two edges e_i and e_j share a common node. By testing the angle θ between them, it is possible to determine whether they are in conflict. (b), (c), and (d) show possible triangulations of non-adjacent edges e_i and e_j.

Different conditions need to apply for *adjacent* and *non-adjacent* edge pairs:

In order to adjust to the prescribed quality criteria, angles between *adjacent* edge-pairs need to be in the described bounds (cf. Figure 4.9 (a)). Also, edge-pairs that are connected by a short edge e_s ($\|\boldsymbol{e}_s\|_2 < r_{\min}$) are treated as if they were adjacent (since we remove short edges by collapsing them later, see Section 4.3.3.3). Hence, the same angle-criterion needs to hold for such edge-pairs.

For *non-adjacent* edges we need to ensure that no two arcs in $\boldsymbol{C}$ are closer than $r_{\min}$ to each other, since then the sampling rate induced by the scale parameter $r_{\min}$ could no longer resolve both feature curves properly. We classify a pair of feature edges e_1 and e_2 as *potentially conflicting* if their minimum distance (edge-to-edge) is below $r_{\min}$. The triangulation between two potentially conflicting edges can essentially be one out of four possible configurations, which are depicted in Figure 4.9 (b), (c), and (d). Configuration (b) can be excluded if $r_{\max}/r_{\min} < \sqrt{3}$, since then the minimum height of a triangle is larger than $r_{\min}/2$ and thus configuration (b) could be replaced by one of the configurations in (c). The required condition for non-adjacent edges is thus that at least one of the configurations (c) or (d) can be built with all edges having at least length $r_{\min}$, where in configuration (d)

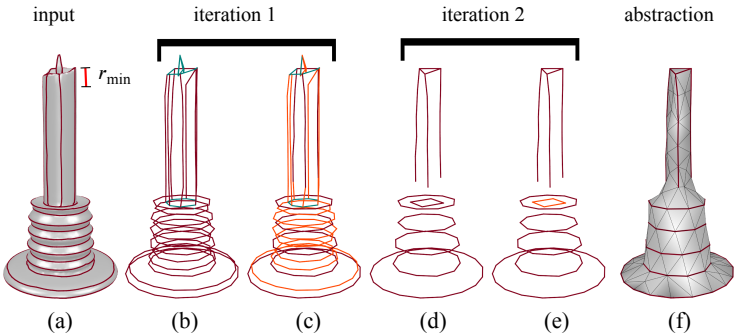


Figure 4.10: Intermediate steps of the FCN abstraction. (a) Input. (b) First resampling with red curves in the prescribed lengths, cyan curves are short curves, which will induce vertex conflicts. (c) After proximity based conflict detection and optimization. Orange curve segments are marked as 'delete' by the optimization due to conflicts with other curves. (d) Conflicts are removed and the arcs are resampled. (e) Resampled arc set that was formerly composed of short arcs introduces new conflicts, which are removed (orange). (f) Final abstraction.

we additionally have to make sure that the angles β_1 and β_2 are both $\geq 2\,\theta_{\min}$, allowing for two further triangles to be fitted inbetween. If we would only require $\beta_i \geq \theta_{\min}$, this would reduce to configuration (c).

4.3.3 Feature Network Adaptation

The idea of our scale adaptation scheme is that we start with a given feature network Γ and then iteratively convert it into a network Γ_{sc} that is conforming to a given scale (see example in Figure 4.10). In order to make Γ conform to $r_{\min}$, we iterate a four step procedure as shown in Figure 4.7.

During the *proximity based conflict detection* phase (Section 4.3.3.2), we first resample each arc of the network in order to satisfy the feature edge length condition. Secondly, we identify conflicts in the current FCN. In the *feature*

curve selection phase (Section 4.3.3.3), we first compute a weight coefficient for each feature edge to rate its relevance. In the third step, we solve a global labeling problem that finds the maximum set of non-conflicting edges in the network. Because these three steps cannot resolve complex constellations of multiple extraordinary feature vertices lying closer to another than $r_{\min}$, we have to perform a fourth step in which singularities are merged if required. Since this changes the FCN topology, we have to run the entire four-step procedure again, starting with the resampling of the arcs. We do this until no further changes occur. This procedure always converges, since at least one edge is removed in each iteration. Usually, it converges after 2-5 iterations.

4.3.3.1 Initial Feature Curve Network

We used three different feature detection methods to generate the initial FCNs Γ in our experiments: Variational Shape Approximation [CSAD04], the feature detection approach presented in [YBS05], and live-wire mesh segmentation [ZZCJ14]. The initial network has the same topology as the feature curves (e.g. segmentation boundaries as in Figure 4.10 (a) or crest lines) and with arcs geometrically following relevant feature curves on the input surface.

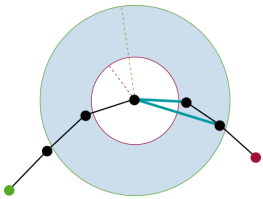
Note, that we preserve all features of the initial network that are not suppressed by a stronger feature. E.g. the VSA algorithm generates long streched patches in regions with constant curvature along one direction (e.g. inside of Rocker Arm, Figure 5.2). Since there are no stronger features nearby, they are preserved in the abstraction.

4.3.3.2 Proximity Based Conflict Detection

Taking a feature network as input we have to determine whether it is scale conforming (cf. Section 4.3.2). In order to do this efficiently, we have to resample the network. Non-conformity can then be represented as vertex and edge conflicts.

Feature Curve Resampling Since the edge lengths along the arcs of the initial feature network Γ might not lie in the prescribed scale interval S, we apply

a resampling procedure to each arc. Let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ be a set of dense samples along an arc in the initial network. Our goal is to find the best possible piecewise linear approximation $\overline{\boldsymbol{p}}_1, \ldots, \overline{\boldsymbol{p}}_k$ such that the length of each resulting feature edge $\left(\overline{\boldsymbol{p}}_i, \overline{\boldsymbol{p}}_{i+1}\right)$ lies in S. We formulate this optimization problem as a shortest path search.



We define a search graph as follows: $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ are the nodes of the graph (black points) and we define a directed edge (cyan) between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ $(j > i)$ if their Euclidean distance lies in the prescribed scale interval S (light blue). We assign a weight to this edge which is proportional to the geometric deviation (i.e. integral Euclidean distance) of the edge $\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right)$ from the (sub-)polygon $\boldsymbol{p}_i, \ldots, \boldsymbol{p}_j$. The optimal re-sampling is then found as the shortest path from $\boldsymbol{p}_1$ (green) to $\boldsymbol{p}_n$ (red) by using Dijkstra's algorithm. We label the nodes that are visited on this shortest path as $\overline{\boldsymbol{p}}_1, \ldots, \overline{\boldsymbol{p}}_k$. In case of loops where $\boldsymbol{p}_1 = \boldsymbol{p}_n$, we run Dijkstra's algorithm several times from different starting points to find the best $\boldsymbol{p}_1$.

In the following two situations we cannot resample the feature curve to have all lengths in S. An obvious problem occurs when the entire initial arc is shorter than $r_{\min}$. This is handled by assigning a special *short* edge e_s with $\left\|\boldsymbol{e}_s\right\|_2 < r_{\min}$ to this arc. Short edges will be removed in the post-processing step (Section 4.3.3.3) by collapsing them. Another, less obvious problem occurs if the arc length is larger than $r_{\max}$ but shorter than $2\,r_{\min}$, or more generally in an interval of the form $[k \cdot r_{\max}, (k+1) \cdot r_{\min}]$, which does not contain a solution in the given bounds. In this case we assign a special *long* edge e_l to the beginning or end of this arc with $||e_l|| > r_{\max}$.

Edge and Vertex Conflict Detection In the resampled FCN it is possible to identify edge- and vertex-pairs that violate the conditions which need to be fulfilled in a scale conforming network defined in Section 4.3.2.

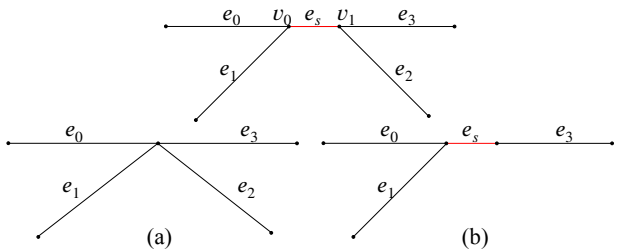


Figure 4.11: Resolution of vertex conflicts: Vertex conflicts occur if $||v_0 - v_1|| \leq r_{\text{min}}$ (top row). There are two possibilities to resolve such a conflict. (a) The vertices are collapsed into a common vertex. (b) The incident short edge is integrated into a neighboring arc. The choice between (a) and (b) is made by the global optimization (cf. Section 4.3.3.3). If the binary value of the edge e_s results in 0, (a) is applied. Otherwise, if it is 1, one of the adjacent vertices will have been downgraded to a regular vertex by the optimization.

Edge-Conflicts: Neighboring edges that do not fulfill the angle criteria are labeled as conflicting. Also, the unconnected potentially conflicting edges described in Section 4.3.2 are tested for the listed configurations. If none of these configurations apply, the edge-pair is labeled as conflicting.

Vertex-Conflicts: We define a pair of extraordinary vertices v_0 and v_1 as conflicting if and only if $||v_0 - v_1|| < r_{\text{min}}$. If the FCN contains a short edge e_s (top row in Figure 4.11), it is incident to two extraordinary vertices $v_0, v_1 \in V^*$. In case the valence of these vertices equals 1, the short edge is not representable in the network and it will be removed. Otherwise, either v_0 and v_1 are collapsed into a common vertex (Figure 4.11a), or e_s is fused into one of the neighboring arcs. In the latter case, one of the two extraordinary vertices needs to be downgraded to a vertex with valence 2 (Figure 4.11b). e_s is then removed in the resampling step of the next iteration.

4.3.3.3 Feature Curve Selection

To achieve conformity for the FCN, we have to resolve the detected conflicts found in Section 4.3.3.2. This is done by suppressing conflicting edges that are not as relevant as the edges they are in conflict with. Thus, we extract a globally optimal set of feature curves, not only respecting the amount of curves but also their quality. Finding such a globally optimal set outperforms possible greedy alternatives (e.g. farthest point/edge/curve sampling), since all conflicts and weights are regarded simultaneously, which is not possible for greedy solutions that inevitably lead to less appealing results.

Feature Edge Weights We account for both single-edge and combined-edge quality criteria by assigning weights to individual edges as well as to pairs of edges according to Section 4.1. We compute individual edge weights as a product of different weighting factors:

$$w(e) \;=\; w_{\kappa}(e) \cdot l(e) \cdot w_{\mathrm{Sym}}(e) \cdot w_{\mathrm{Loop}}(e).$$

We choose $\alpha_{\mathrm{sym}} = 2$ in our experiments. Symmetric feature curves are detected in a preprocessing step (cf. Chapter 3). Each edge that is labeled symmetric has a second (symmetric) representative, which only exists as long as its symmetric counterpart is preserved. We set $\alpha_{\mathrm{loop}} = 100$ in our experiments.

Pairs of edges are evaluated by their smoothness w_s, with a falloff parameter $p = 2$ in our experiments. Furthermore, orthogonal an parallel edge configurations are supported with the weight $w_{||,\perp}$.

Conflict Removal and Feature Edge Selection In order to find the maximum conflict-free subset of the feature edges in Γ, we formulate the problem-statement described in Sections 4.3.1 and 4.3.2 as a binary labeling problem for a set of optimization variables b_i^e, each indicating whether the corresponding feature edge $e_i \in E$ belongs to the conflict-free subset ($b_i^e = 1$) or not ($b_i^e = 0$). Since black-box numerical solvers can solve linear constrained problems more

efficiently than quadratically constrained problems, we formulate the optimization as a linear program (e.g. by introducing binary pseudo variables).

A conflict-free subset of E is *optimal* if it maximizes the objective function:

$$\begin{aligned} \mathcal{E}_{\text{single}} + \eta_0 \mathcal{E}_{\text{Sym}} + \eta_1 \mathcal{E}_{||,\perp} \\ \mathcal{E}_{\text{single}} &= \sum_i b_i^e \cdot w(e_i) \\ \mathcal{E}_{\text{Sym}} &= \sum_{i,j} b_{i,j}^e \cdot w_{\text{Sym}}(e_i, e_j) \\ \mathcal{E}_{||,\perp} &= \sum_{i,j} b_{i,j}^e \cdot w_{||,\perp}(e_i, e_j). \end{aligned} \tag{4.2}$$

The objective function is composed of the terms $\mathcal{E}_{\text{single}}$, $\mathcal{E}_{\text{Sym}}$, and $\mathcal{E}_{||,\perp}$ referring to the respective single-edge and combined-edge weights with weighting factors η_0 and η_1 set by the user. In all our experiments we choose $\eta_0 = 100$. For quad meshing related experiments we set $\eta_1 = 10$, otherwise $\eta_1 = 0$. For the combined objective, we introduce an additional set of binary pseudo variables $b_{i,j}^e$, which merely indicate if the combined term between e_i and e_j is active ($b_{i,j}^e = 1$) or not ($b_{i,j}^e = 0$). In case the combined term is not computed for the edges e_i and e_j (e.g. edges are not adjacent to a common vertex for the smoothness term), then $b_{i,j}^e = 0$. Otherwise, the boundary constraints:

$$b_{i,j}^e \leq b_i^e \quad \& \quad b_{i,j}^e \leq b_j^e \tag{4.3}$$

make sure that the combined term is active only if both involved edges are.

In the following we will further refine this mathematical model to generate scale-conforming networks. For this we translate conflicts described in Section 4.3.3.2 into constraints of the optimization.

If the feature edges e_i and e_j have a conflict (according to the definition in Section 4.3.2), then we add a constraint of the type

$$b_i^e + b_j^e \leq 1 \tag{4.4}$$

to the optimization problem, which makes sure that not both edges can belong to the optimal subset. Note that symmetric edge pairs e_i and e'_i share the same optimization variable b^e_i, since a symmetric edge exists if and only if its symmetric counterpart exists. In case of n-fold symmetries, n edges share the same optimization variable.

To handle vertex conflicts (cf. Section 4.3.3.2) where extraordinary feature vertices lie closer than $r_{\min}$ to each other, we introduce further binary variables b^v_i, which indicate for each extraordinary feature vertex $v_i \in V^*$ whether it should be kept ($b^v_i = 1$) or be removed ($b^v_i = 0$). First, for every pair of extraordinary feature vertices v_i and v_j whose distance is less than $r_{\min}$, we add a constraint

$$b^v_i + b^v_j \leq 1, \tag{4.5}$$

which is analog to the constraint for conflicting edges. Being labeled as "remove" does not necessarily mean that an extraordinary vertex is deleted. It can be sufficient to downgrade it to a regular feature vertex and then remove it in the feature arc resampling step of the next iteration, as described in Section 4.3.3.2. I.e. one of the conflicting vertices needs to become regular, which means its valence must become less or equal to 2. This condition for a vertex v_i can be formulated as another constraint:

$$(1 - b^v_i) \sum_{e_j \in \mathcal{N}_1(v_i)} b^e_j \leq 2 \tag{4.6}$$

A special case of this condition applies to extraordinary vertices v_i with valence 1 before conflict resolution. Here, the one adjacent edge e_j has to be removed as well, leading to the constraint:

$$(1 - b^v_i) \cdot b^e_j \leq 0. \tag{4.7}$$

These quadratic constraints can be linearized in the following way: For every edge e_i a pseudo variable b_i^p is added. An equivalent formulation to Constraint (4.6) (and analogously Constraint (4.7)) is:

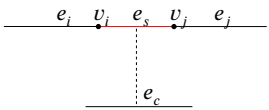
$$\begin{aligned} \sum_{e_j \in \mathcal{N}_1(v_i)} b_j^p \;&\leq\; 2 \\ b_j^p - b_j^e \leq b_i^v \quad &\& \quad b_j^e - b_j^p \leq b_i^v. \end{aligned} \tag{4.8}$$

With the above constraints, we guarantee that all conflicts are properly handled. We can further improve the quality of the resulting FCN by regarding two further types of conflicts. So far we might end up with isolated short edges, which cannot be resampled and are not representable at the given resolution. To avoid this, we introduce another constraint for all edges $e_s = (v_i, v_j)$ shorter than r_{min}:

$$\sum_{e_k \in \mathcal{N}_1(v_i) \setminus e_s} b_k^e + \sum_{e_k \in \mathcal{N}_1(v_j) \setminus e_s} b_k^e \;\geq\; b_{e_s}^e, \tag{4.9}$$

which implies that if e_s is removed ($b_{e_s}^e = 0$), then the constraint does not have any effect, but if the short edge is kept ($b_{e_s}^e = 1$), then at least one of the two end vertices must be connected to at least one more edge, leaving no isolated short edges in the abstraction.

As described above, we will remove a short edge e_s ($b_{e_s}^e = 0$) by collapsing its two end vertices. Then we must consider the following situation on the right: It can occur that there is a set E_c of other feature edges (e.g. e_c) that had a conflict with e_s. These conflicts (dashed line) are inherited after the collapse by the edges adjacent to e_s (e.g. e_i and e_j). Since these are foreseeable conflicts, we can take them into account preemptively by adding another set of constraints to our optimization problem. For every short edge $e_s = (v_i, v_j)$ and every edge e_c from its conflicting set E_c, we require



$$b_c^e + b_i^e + b_j^e \;\leq\; 2 \tag{4.10}$$

for all pairs of edges e_i and e_j that are adjacent to v_i and v_j respectively. This constraint can be interpreted as follows: if e_s wins against all its conflicting edges

(i.e. $b_s^e = 1$), then $b_c^e = 0$ (according to Constraint (4.4)) for all edges in E_c, and thus Constraint (4.10) is automatically satisfied. If e_s loses ($b_s^e = 0$) and an edge $e_c \in E_c$ is preserved by the optimization ($b_c^e = 1$), then in each pair e_i, e_j only one edge can be kept.

Overall, the task of finding the optimal non-conflicting subset of the given feature network Γ amounts to solving a linear program (4.2) with linear constraints (4.3) – (4.5), and (4.8) – (4.10). These types of optimization problems can be solved quite efficiently by off-the-shelf solvers such as the linearly constrained mixed integer solver by GUROBI [GO15].

The objective of this optimization is the preservation of as much feature information as possible. Since the edges are treated separately, it can occur that feature curves are fragmented. Applications such as segmentation layout generation might require the retainment of a patch structure. Incorporating higher order structure and treating an entire feature curve as a whole (and thus avoiding fragmentation and invalid segmentation layouts), can be implemented by assigning the same binary optimization variable to all edges belonging to a curve.

Short Edge Collapsing The result of the labeling problem is a set of binary variables that indicate, which edges belong to the maximum non-conflicting subset. Short edges e_s that are assigned $b_s^e = 0$ are now removed from the network by edge collapses, where we pull the vertex with lower sharpness value into the one with higher sharpness. This process together with the deletion of suppressed regular edges leads to topological changes in the feature network, since entire branches can disappear and formerly separate arcs can be joined. Hence, we iterate the process by feeding the result back into the resampling step (cf. Section 4.3.3.2).

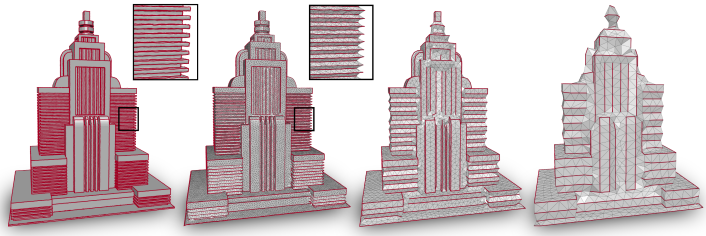


Figure 4.12: Feature curve networks (red edges) generated with our approach are adapted to different scales (from fine to coarse). Our goal is to preserve as much feature information as possible while avoiding features that are too close to each other for a given target edge length. This requires global optimization and cannot be solved by greedy approaches or combination of filtering and thresholding. Here, the feature curves are used as boundary constraints for isotropic remeshing at different target edge lengths. The input-mesh with the initial feature curves is shown on the left.

4.3.4 Results

Figures 4.12 and 5.2 show a variety of exemplary FCNs generated by our method. We will give more details on the application of these FCNs in Chapter 5. We observe that for each scale interval, continuous and prominent feature-lines are preserved, while less salient curves which are not representable are suppressed.

In Figure 4.13 we compare our results to conventional filtering techniques as well as smoothing and curvature thresholding. Some of the models are given to a feature preserving isotropic triangle remesher [BK04]. In these cases we threshold absolute maximal curvature values. For the candle model we additionally suppress features based on the cycledness integral presented in [YBS05]. Especially, for the Skyscraper we can observe that either all features are preserved or removed since they align along surface elements with a dihedral angle of 90 degrees. Hence, all curvature values have about the same magnitude. In the

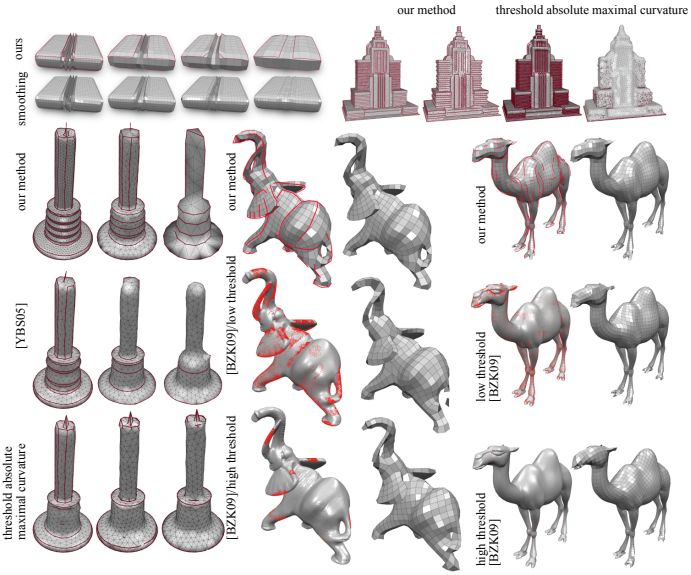


Figure 4.13: Comparisons of our method to conventional local filter approaches (thresholding/filtering/smoothing). In case of the triangle meshes we threshold absolute curvature values. For the candle we additionally use curvature thresholding via $\mathcal{T}$ (cf. Section 2.1.2, [YBS05]). For the filtering and thresholding of curvatures for the quadmeshes, we used [BZK09] with a filter-kernel radius of $r_{\min}/2$ for the elephant and the camel and mesh smoothing for the box (top). Note that in all cases, if we increase curvature thresholds such that all small-scale details are removed, also less prominent features are removed, which are important to convey the shape or to guarantee good element alignment. Especially, for the Skyscraper and the box model (top row) we can observe that the feature curves are subsampled with our method, while previous work gradually decreases the strength of all features simultaneously.

FCN of the Candel model, the flame is preserved until all other features are suppressed, due to high curvature values. With the method described in [YBS05],

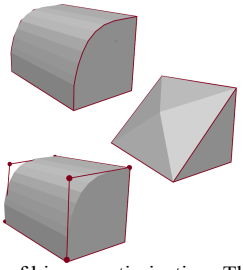
this is not the case, since they incorporate the segment length into their threshold. Nevertheless, they still cannot control the feature density. With our subsampling approach, all features that can be represented in the given resolution (i.e. target edge length) are preserved. E.g. the flame is suppressed, while larger features (e.g. top of the candle) are preserved.

Furthermore, we leverage a set of FCNs to the quadmeshing procedure presented in [ECBK14]. For the filtering and thresholding of curvatures (Camel and Elephant model), we used [BZK09] with a filter-kernel radius of $r_{\text{min}}/2$. In the respective top rows of Figure 4.13, our method is depicted. Below, we apply curvature thresholding with a threshold, where all important features are included. In the resulting quad meshes we can observe that this can lead to over-constrained parametrizations (e.g. the elephants tail degenerates). Also, the ears of the Elephant and the eyes of the Camel are regions with high feature density, which can lead to bad element quality if the respective feature directions do not align well (as can be observed in the respective models). Then, if we further increase the threshold to avoid this effect, all other features with lower curvature (e.g. on the body of the Elephant/Camel) are suppressed as well, leading to bad alignment of the elements. In contrast, our method avoids regions with high feature density, i.e. all less significant features that are closer than the minimum scale are suppressed by stronger features. At the same time, weaker feature curves that are not in conflict with any closer feature are preserved (such as the curves along the body of the Elephant/Camel).

In Figure 4.13 (top left), we demonstrate the effect of feature suppression by smoothing (bottom row) in contrast to our sub-sampling approach (top). For comparison, we compute quad meshes guided by curvature constraints, which were smoothed in advance to remove small-scale details. Note that the number of ribs remains constant and only their amplitude decreases, while the number of ribs is incrementally reduced with our sub-sampling approach and hence gives a representation of the feature set at different resolutions.

Limitations Since we strive to maintain a maximum amount of salient feature curves in the final network, we cannot guarantee that it will result in a connected B-Rep, as it is done in [MZL*09]. Also, the connectivity of the final feature-graph is strongly dependent on the initial network. E.g. the feature curves generated with [YBS05] are mostly isolated ridges or ravines without many junctions. The gaps between curves are also visible in our scale conforming FCNs, which should be acknowledged when choosing the method to generate the initial feature curves. Moreover, we do not close gaps between feature lines during the abstraction for two reasons. First of all, we do not want to ”invent” additional information to the initial feature set, since we would require a heuristic which selects two curves to close the gap. Secondly, we can guarantee that the algorithm terminates since we remove at least one feature edge in each iteration. Extending features could affect termination.

Furthermore, our arc resampling strategy restricts the possible abstractions. We only resample along the arc and do not take any points into account that deviate from the arc. E.g. the image to the right shows a rounded surface. Our resampling cuts this curve off (middle). A better abstraction (in the sense of a least squares error) might be to extend the curves as it is done in the image below. The benefit of our approach is that it has a comparatively small solution space, allowing an efficient computation. To extend features as discussed above would involve finding an appropriate formulation to solve this problem efficiently.



Another limitation derives from the NP-hardness of binary optimization. This can affect performance in cases with a large amount of inter-dependent conflicts. A greedy approximation strategy could be of value in such a setting. However, in our examples the duration of the optimization procedure ranged from 0.0023s (Octaflower) to 72.09s (Iphigenie) with an Intel Core i7-4770 CPU.

The globally scale conforming feature curves computed by our algorithm provide a new automatic alternative for feature based algorithms. The detection

and preservation of salient and representable features can be useful for various kinds of level-of-detail algorithms, which can range from geometry processing tasks as remeshing and smoothing to shape analysis tasks as feature persistence analysis.

4.4 Summary

In this Chapter we have discussed possible global and local measures to evaluate the relevance and meaningfulness of a feature curve. While local salience measures (e.g. sharpness, length) and operations (e.g. smoothing, filtering) provide reasonable heuristics to suppress feature curves and avoid effects of measurement noise, they ignore the global context given by the entire feature curve networks. Given global information, it is possible to make more reliable decisions on which feature curves are relevant. In this chapter we have discussed two possible global information sources: symmetry and scale. The resulting completed and abstracted feature curve networks can be used as input to downstream applications such as remeshing, smoothing, or completion. Also high-level structure information is provided by the algorithms, e.g. which features are preserved throughout a hierarchy of scales, or which templates reoccur in which manner. This high-level information can be exploited in inference tasks such as procedural modeling or segmentation.

5 Applications

In the previous chapters we have discussed how to obtain a meaningful set of feature curves from a dense, possibly noisy FCN computed on raw data. Additional structural information such as correspondence or template membership are computed in the process. The resulting FCNs are less fragmented with a significantly higher signal to noise ratio and can be adapted to a required density. This is especially beneficial for low-level geometry processing applications. We will show applications in isotropic and anisotropic remesheing as well as feature preserving smoothing in Section 5.1. High-level structural information is essential to shape analysis tasks. In Section 5.2 we present how the knowledge of corresponding feature curves can be leveraged to solve complex, far from isometric shape matching tasks.

Some of the material, results, and findings in this chaper have been published in [GLK16, GLK18, GBKS18].

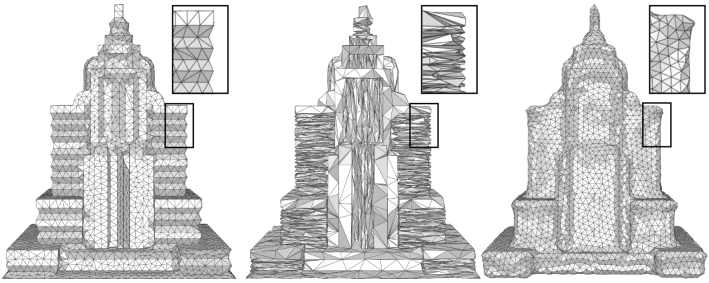
5.1 Low-Level Geometry Processing with Meaningful FCNs

Raw geometric data (e.g. scanned real-world data) usually requires further processing before it can be utilized in downstream applications. E.g. the data can be dense with millions of points and be influenced by different sources of noise (as discussed in Chapter 3). Typical surface enhancement tools are remeshing algorithms and smoothing. Remeshing techniques can change both the geometry and the topology of an input mesh, to improve the element quality and alter the target complexity. By smoothing the surface, its geometric representation is

changed. Surface smoothing acts like a low-pass filter on the mesh points. These are updated iteratively to improve the element quality and reduce noise. If the afore mentioned techniques do not take any feature information into account, sharp features are filtered such that strong features are less pronounced and weak features disappear entirely. Remeshing procedures typically reduce the target complexity. Hence, they take a scale parameter into account to adapt the input resolution to the required complexity. If we use feature curves computed on the raw data to guide the remeshing process, it will be constrained to too many small scale details and cannot adapt to the target resolution. When isolated feature curves are suppressed locally by filtering the curves based on a curvature threshold relevant information might be ignored. Hence, we use the globally scale adapted feature curves that we presented in Section 4.3 for this task. To smooth the data, we are not required to change the target complexity, however the surface points should not be constrained by misclassified feature curves. Furthermore, fragmentation of the feature curves can lead to unpleasant artifacts along a feature edge. Hence, we can exploit the global template information obtained in Section 4.2.3 to complete the FCN and reduce noise for this task.

5.1.1 Isotropic Remeshing

Advances in triangle remeshing focus on goals such as high quality, feature preservation, or fidelity [ADVDI03, AUGA08]. While features can be preserved, it is not possible to suppress them in a controlled manner for a given target edge length. For demonstration, we apply quadric decimation [GH97] and isotropic remeshing [BK04] on the Skyscraper model in Figure 5.1 (middle and right). We can observe that quadric decimation preserves features although they cannot be represented properly at the given resolution, which leads to many degenerated and pointy triangles. Isotropic remeshing on the other hand smoothes the features away. Both methods rely on local shape information. In contrast, our feature curve suppression method takes the global feature spacing into account and generates a subsampling of features which leads to well-shaped triangles, which align along feature curves (Figure 5.1(left)).



our method	Quadric Decimation	Isotropic Remeshing
0.0195	0.0194	0.0883

Figure 5.1: Comparison with Quadric decimation and Isotropic remeshing. The Quadric decimation was performed until the mesh-complexity (vertices) was equal, while isotropic remeshing was performed for the same target edge length. The numbers below give the Hausdorff distance (d_{H}) (with a bounding box diagonal of length 1.4). Although the d_{H} of the Quadric decimation is similar to that of our abstraction, ours has well-shaped triangles and suppresses features that cannot be represented, while the decimated object has many degenerate, pointy triangles aligning along the features. The isotropic remeshing has a much higher d_{H}, since features are smoothed away.

To achieve this, we leverage the FCN as input to the isotropic remeshing algorithm described in [BK04]. This is an iterative 4-step procedure which (1) splits edges that are too long and (2) collapses those that are too short, (3) flips edges to improve the overall valence (optimal is a valence of 6) and (4) applies tangential smoothing to improve the element quality. We add the following constraints to ensure that the scale adapted FCN Γ_{sc} is retained in the output mesh:

1. Collapses of edges involving a feature vertex v can only collapse into v.

2. Feature vertices of the scale adapted FCN Γ_{sc} are not changed during the smoothing process and they are not removed by collapse.

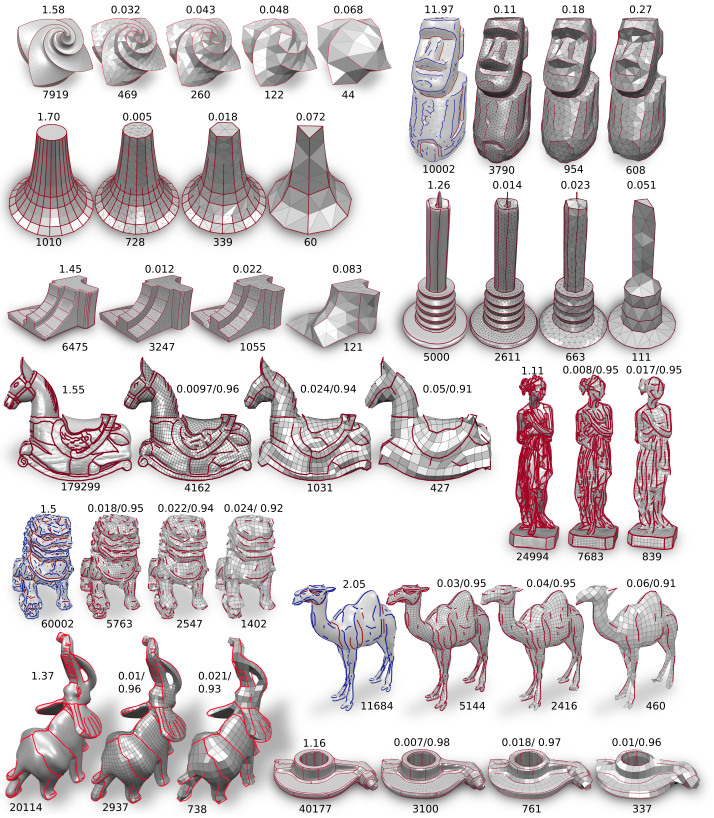


Figure 5.2: Scale conforming FCNs at different scales. Initial feature sets are depicted leftmost, with the length of the bounding box diagonal given above the respective object. These were generated with VSA (Trumpet, Octaflower, Fandisc, Iphigenie, Rockerarm), [YBS05] (Moai, Chiniese Lion, Camel), or Livewire (Candel, Isidore Horse, Elephant). Extracted FCNs at different scales were used as input to isotropic triangle remeshing [BK04], or level of detail quad meshing [ECBK14]. The number below the objects gives the mesh complexity (number of vertices). The one above gives the Hausdorff distance (d_{H}) for triangle meshes and d_{H}/average Scaled Jacobian ($a\boldsymbol{SJ}$) for quad meshes.

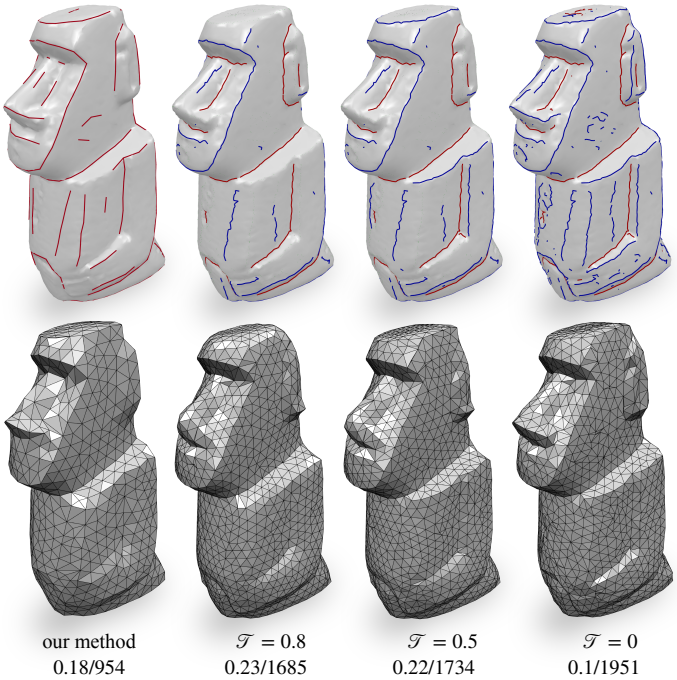


Figure 5.3: Comparison of our feature line abstraction method to curvature thresholding. The top row depicts the boundary constraints used for isotropic triangle remeshings (bottom row). The numbers below give Hausdorff distance (d_{H})/vertices. The left column shows results from our feature line abstraction. On the right, isotropic remeshing constrained by the ridges and ravines from [YBS05] is applied with decreasing curvature thresholding parameter $\mathcal{T}$. For $\mathcal{T} = 0$ the prescribed resolution cannot be guaranteed which yields a much higher mesh complexity, while for increasing $\mathcal{T}$ features are removed that could have been represented (e.g. tip of the nose, ear). It is not surprising that the d_{H} is smaller for $\mathcal{T} = 0$, since all features are preserved during remeshing.

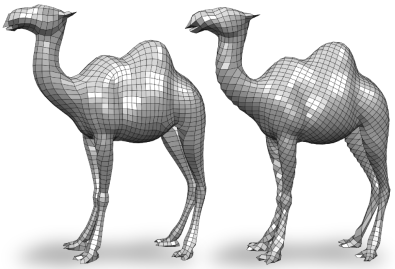
3. Feature edges are not flipped.

Figure 5.2 shows the scale adapted feature curves and the respective abstracted triangle meshes at different target scales S (fine-to-coarse). In contrast to previous feature suppression mechanisms, where curves are removed via filtering or smoothing, we can observe a subsampling of the feature-curves which is based on global scale parameter.

For demonstration we compare the curvature-filtering approach presented in [YBS05] to our feature subsampling in Figure 5.3. Examples are generated with the cycledness thresholding parameter set to $\mathcal{T} = 0$, $\mathcal{T} = 0.5$, and $\mathcal{T} = 0.8$. In the case where no thresholding is performed ($\mathcal{T} = 0$), we observe several small and dense features. Triangles that align along these features do not satisfy the target resolution. We increase $\mathcal{T}$ to remove small scale details. At the same time features are removed and smoothed away in the remeshing procedure, which could have been represented at the given scale (e.g. the tip of the nose or the ears in Figure 5.3). With our method, the feature curves are adapted to the target resolution, i.e., all features which can be represented at the given scale are preserved and resulting triangles are well shaped.

5.1.2 Anisotropic Remeshing: Guiding Constraints for Level-of-Detail Quad Meshing

State of the art quad meshing and quad layout methods are not able to suppress or select features based on their density. They create high quality surface representations only if they are provided with (or precompute) proper guiding constraints. These guiding constraints give a direction along which the quads should align. E.g. the image above



quad meshing with **our method** as guiding constraints

without guiding constraints

(two camels) depicts two quad meshes (computed with [EBCK13, ECBK14]). On the left, our feature abstractions were supplied as guiding constraints, while for the right mesh no constraints were given. Note that in contrast to the image on the right, the quads in the left image align along features and curvature directions, which demonstrates the need for such guiding constraints. So far, guiding constraints are usually either manually supplied or obtained from filtered curvature directions [KNP07, BZK09, NPPZ12, CBK12, BCE*13, ECBK14]. Although user-supplied constraints lead to high quality results, their acquisition is tedious. Guiding constraints obtained via filtering are sensitive to noise and ignorant of the feature density. Furthermore, certain thresholds need to be set manually by a user.

Figure 5.2 shows a variety of quad meshes at different levels of detail, which were computed using the extracted FCNs as soft guiding constraints. Our subsampling preserves the representable features and quads align along them with high element quality. Hence, using the scale conforming FCNs obtained from our method as soft guiding constraints introduces a high-quality, automatic alternative in this context. By combining our scale conforming approach with Level-of-Detail Quad Meshing from [ECBK14], we can suppress densely spaced features and noise, while preserving dominant large-scale features. The final quad meshes are extracted with QEx [EBCK13].

We compare quad meshes computed with curvature thresholded guiding constraints [BZK09] to those constrained by our scale conforming FCNs in Figure 5.4. In Figure 5.4 (a) we choose a high curvature threshold, to preserve only strong features. By increasing the threshold further (5.4 (b)), nearly all constraints are removed simultaneously, since all features of the same magnitude will either be preserved or removed. Also, less prominent features (e.g. the collar around the Chinese Lion's neck or the eyes' region in Figure 5.4 (a)) are not represented in the curvature thresholded feature set and thus smoothed away. They can only be included by thresholding if all other features of the same strength are incorporated. This can lead to overconstrained parametrizations.

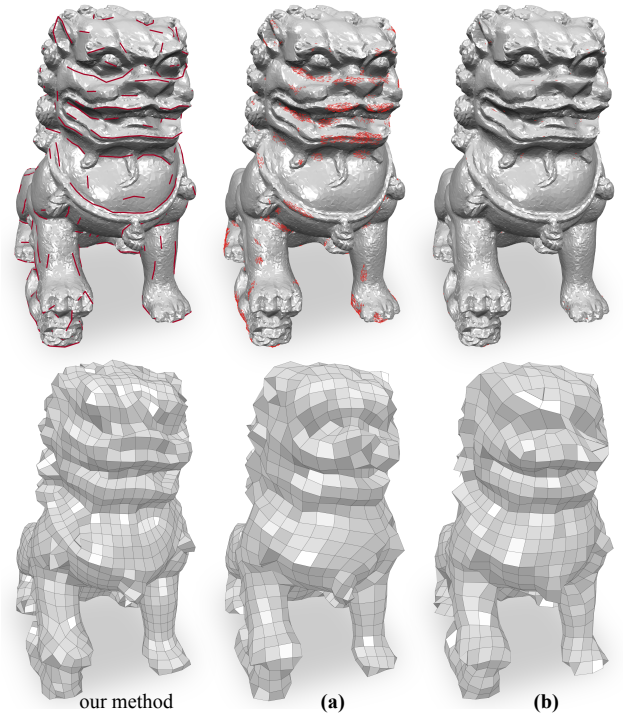


Figure 5.4: Comparison of our scale conforming FCN as guiding contraints for quadmeshing to contraints induced by setting curvature thresholds as in [BZK09]. The top row depicts the guiding constraints. We choose a high curvature threshold in (a) to preserve only strong features and avoid degeneracies in the parametrization. If this threshold is increased in (b), nearly all strong features are removed simultaneously. The numbers below give the d_{H}/average Scaled Jacobian ($a\boldsymbol{SJ}$). While in (a) the quads align along the features, the $a\boldsymbol{SJ}$ is much lower than in (b), which has a high $a\boldsymbol{SJ}$ due to the low number of constraints, but quads do not align well along features. With the scale adapted FCNs we can observe that the $a\boldsymbol{SJ}$ is similar to that of (b), while still aligning well to the features at the given scale.

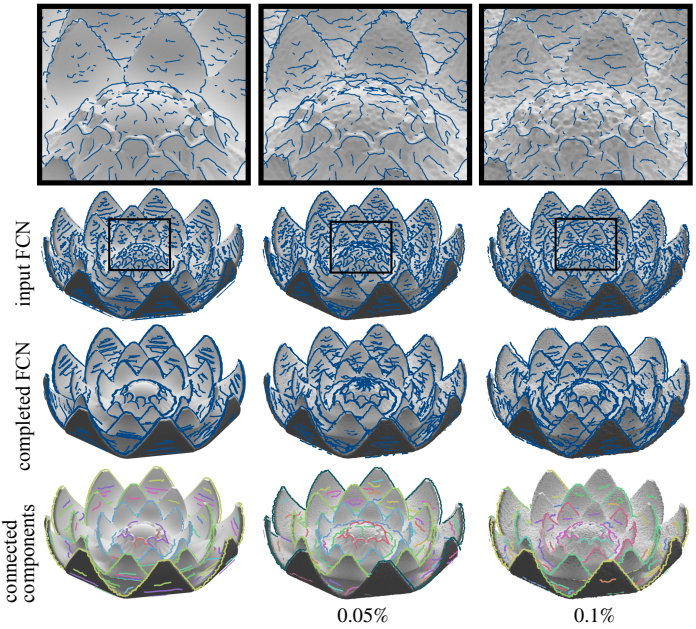


Figure 5.5: Noise test. Different levels of Gaussian noise with a standard deviation of 0.05% (middle) and 0.1% (right) of the bounding box diagonal were added to the synthetic lotus-flower mesh (left). The completed FCNs are depicted in the intermediate row. The bottom row shows the connected components. Note that most reoccurences are detected even with a high amount of noise.

5.1.3 Feature Preserving Smoothing

It is possible to reduce noise and enhance the element quality by Laplacian smoothing [DMSB99, Tau95]. Iteratively, each vertex is shifted into the direction of a weighted average of its one-ring neighborhood. This local adjacency information is given by the mesh Laplacian. Hence, it can be used to compute update vectors for the mesh points. Intuitively, this process can be understood as a low-pass

Noise	distance	co-occurring curves (ours)	all curves	no curves
0.05%	Hausdorff	$\mathbf{7.96 \cdot 10^{-3}}$	$1.83 \cdot 10^{-2}$	$1.64 \cdot 10^{-2}$
0.05%	average	$\mathbf{4.92 \cdot 10^{-4}}$	$7.79 \cdot 10^{-4}$	$8.31 \cdot 10^{-4}$
0.1%	Hausdorff	$\mathbf{1.49 \cdot 10^{-2}}$	$1.78 \cdot 10^{-2}$	$1.50 \cdot 10^{-2}$
0.1%	average	$\mathbf{5.47 \cdot 10^{-4}}$	$8.03 \cdot 10^{-4}$	$1.05 \cdot 10^{-3}$

Table 5.1: Hausdorff- and average distance values for smoothing experiment. We apply feature preserving C^1 smoothing (100 iterations) to the meshes in Figure 5.5, by retaining points on the curves detected by our method (3rd column) and the initial curves (4th column), and compare these to unconstrained smoothing (5th column). The bounding box diagonal has length 1.

filter which is applied to the mesh to remove high frequency noise. In this process sharp features are smoothed as well. To retain surface features, the smoothing process needs to be constrained. One possibility is to exclude edges lying on a reasonable set of feature curves from the update procedure and treat these as fixed boundary vertices.

We show quantitative results of this method in Table 5.1 for the input depicted in Figure 5.5, where we add Gaussian noise with a standard deviation of 0.05% (middle) and 0.1% (right) of the length of the bounding box diagonal to the clean lotus flower mesh. The smoothing process is constrained by feature curves that are enhanced by co-completion as discussed in Section 4.2.3. The initial feature curves (cf. Figure 5.5 top row) that are computed on the meshes with noise are more jagged and fragmented compared to the curves computed on the synthetic data. Nonetheless, we are able to detect most of the reoccurrences for these noise levels (bottom row) and complete the network based on these (middle row). However, if the noise level increases, the completed curve networks can also include jagged curves. To measure the effectiveness of the feature preserving smoothing with a co-completed FCN, we remove the artificial noise by applying 100 iterations of feature constrained C^1 smoothing by retaining points on the completed FCN. Hausdorff- and average distances to the orginal data are given

in Table 5.1. For comparison, we show the results using all input curves and unconstrained smoothing. Using all input curves can have two effects. First, the distance values are higher since more displacement noise is preserved. Second, in case of incomplete curve information (e.g. fragmentation of the curves) the geometry is smoothed, increasing the distance to the original surface.

5.2 Curve Constrained Functional Maps

Three-dimensional shape correspondence is essential to geometry processing, serving as a basic component of algorithms for statistical shape analysis, texture transfer, shape interpolation, and other tasks. Correspondence tools enable transfer of information from one shape to another, usually guided by geometric and semantic cues suggesting which features on one shape should be matched to features on another.

Solving the correspondence problem directly for a point-to-point map, however, is difficult to pose and often leads to computationally-intensive algorithms. Functional maps [OBCS*12] provide an efficient alternative structure to relate shapes. Rather than matching points directly, functional maps transport *functions*, represented in a low-dimensional basis on the two shapes. The operators transporting functions from one shape to another are linear, leading to fast algorithms for computing functional maps using linear algebra and convex optimization (cf. Section 2.6.3).

Most functional map algorithms rely strongly on the ability to provide pairs of corresponding descriptor functions, as we have presented in Section 2.6.1, over the two shapes. For this reason, they are based on the assumption that functions, e.g. local point signatures, which are computed independently on both shapes, are comparable. Reliance on high-quality descriptors becomes a key limitation in difficult correspondence settings: functional maps are effective for shapes for which it is straightforward to extract pairs of matching descriptors, e.g.

nearly-isometric surfaces, and otherwise can become unreliable.

When shapes vary significantly, the assumption that point signatures or segmentations can be matched breaks down. In this regime, automatically computed descriptors can be misleading for map computation and can give unsatisfying results. Finding a matching set of descriptors for a given pair of shapes becomes tedious and nearly as difficult as the correspondence problem itself. Algorithms designed to handle this scenario often iteratively recompute the map, adjust pointwise signatures, or add constraints that make the problem difficult to solve efficiently. Recent attempts at constructing descriptors with deep learning [LRR*17] show promising results but, require a training set of shapes with known correspondences, which is not always available.

We present a technique to compute functional maps between extremely non-isometric shapes by incorporating *curve constraints* in an interactive feedback loop. These constraints disambiguate poor or misleading geometric cues without incurring significant computational cost. Our technique is based on the observation that semantically similar shapes can be abstracted by feature curve networks, in which corresponding feature curves describe related parts. This representation is extremely flexible, since the shape of the curve along the geometry as well as its length suggest how the shapes relate without imposing hard isolated point constraints.

Our algorithm relies on corresponding curves. One possibility to obtain these is to detect corresponding feature curves as discussed in Chapter 3 and suggest these curve pairs to a user, who can verify these interactively (each curve correspondence can have a strong effect on the result). For more flexibility, we develop a graphical user interface, allowing to interactively and iteratively build and refine the correspondence between a pair of shapes. In our system, the user provides sparse input indicating start- and end-points of matching feature curves along two surfaces. This input is simple and can be provided by non-experts; furthermore, the intermediate connectivity of the curves can be constructed au-

tomatically. Behind the scenes, we incorporate feature curve constraints into the functional map optimization problem and are able to update the map using an efficient iterative algorithm to provide *interactive* feedback to the user. We demonstrate that high-quality functional maps can be created using only feature curve constraints (resulting from just a few user clicks), removing the reliance on computation of point signatures.

5.2.1 Problem Statement

Given a pair of shapes $\mathcal{S}_1$ and $\mathcal{S}_2$, correspondence algorithms seek a map $T : \mathcal{S}_1 \rightarrow \mathcal{S}_2$ that preserves some structure, e.g. a pointwise geometric quantity like curvature or a segmentation into semantically-meaningful parts. Additional properties can be imposed on T, e.g. bijectivity or smoothness. Two classes of relationships between $\mathcal{S}_1$ and $\mathcal{S}_2$ distinguish settings of the correspondence problem:

Geometric similarity: When $\mathcal{S}_1$ and $\mathcal{S}_2$ are nearly isometric, the desired correspondence preserves the metric as well as intrinsic structures like geodesic distances and Laplacian spectra. The isometry assumption is used widely in non-rigid correspondence, since it holds approximately for inelastic and articulated deformation (cf. Section 2.6.1). Correspondence quality in this setting can be quantified by measuring intrinsic distortion, giving a natural objective function for correspondence algorithms.

Semantic similarity: More generally, shapes are not necessarily isometric, but rather have semantically corresponding structures that should be preserved by the map T. For example, dogs and cats are quadrupeds; a meaningful correspondence would map legs to legs, the tail to tail, and so on. This notion of similarity is less precise and can be hard to quantify; correspondence is often guided by aesthetic considerations and may need manual user annotation.

The existing functional maps framework in [OBCS*12] applies mainly to the first species of similarity. When these strong geometric assumptions are removed,

we advocate leveraging sparse user guidance to provide semantic information that might be difficult to detect from differential operators and curvature; we will demonstrate cases in which a meaningful smooth map can be extracted between surfaces even if their intrinsic structure differs considerably.

To formulate a method for the interactive functional map design, several key components are required:

- *Input:* The user should be provided with a simple, fast, and effective method to input semantic information. For this purpose, we use ordered curve networks that are computed automatically from user-provided endpoints; effectively, this reduces the user input to a few mouse clicks.
- *Output:* The feedback to the user should be given in a way that he or she can assess the map quality and edit the set of feature curves accordingly.
- *Speed:* The functional map computation and display of the feedback to the user must be provided at interactive rates.

5.2.2 Input: Feature Curve Networks

Given two meshes M_1 and M_2 (discrete surface representations of $\mathcal{S}_1$ and $\mathcal{S}_2$), the input provided by the user is in the form of a set of q corresponding curves on M_1 and M_2, which we denote by $c_1^{M_1}, \dots, c_q^{M_1}$ and $c_1^{M_2}, \dots, c_q^{M_2}$, respectively. These two sets of curves form the FCNs Γ_1 and Γ_2. Since drawing curves on surfaces is challenging, the user is requested to provide only the endpoints of the curves, which are connected automatically. We use Dijkstra's shortest path algorithm along mesh edges to connect the endpoints, with two options for weights:

- *Geodesic distance:* The weight for an edge e is its length $\|e\| \in \mathbb{R}^3$, providing an approximation of a geodesic curve.
- *Anisotropic distance:* The weight for an edge e is biased to prefer ridge and valley curves by incorporating anisotropic geodesics as in [CHK13] (cf. Section 2.1.3).

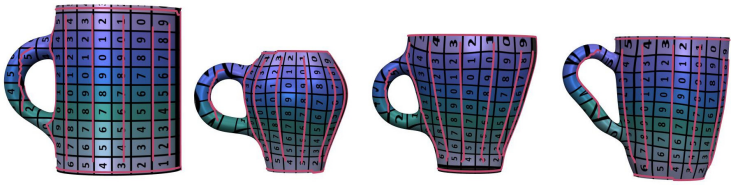


Figure 5.6: Corresponding feature curves (red) on a set of cups with texture coordinates mapped by the functional map. The left object is the source shape, while the other shapes are target meshes.

For simplicity, our segments coincide with triangle edges. Our method, however, extends easily to the general case, simply requiring a generalized definition of the restriction operators below; we found that the increased precision from this more complex computation was negligible.

5.2.3 Functional Maps with Curve Constraints

Let $\gamma^{\mathcal{S}_1} : [0,1] \to \mathcal{S}_1$ and $\gamma^{\mathcal{S}_2} : [0,1] \to \mathcal{S}_2$ be two corresponding curves on $\mathcal{S}_1$ and $\mathcal{S}_2$. $\boldsymbol{\mathcal{R}}^{\mathcal{S}}_{\gamma^{\mathcal{S}}}$ denotes the *restriction operator* taking a function g on $\mathcal{S}$ and outputting its values along the curve γ, i.e. $\boldsymbol{\mathcal{R}}^{\mathcal{S}}_{\gamma^{\mathcal{S}}} g = g \circ \gamma^{\mathcal{S}}$. Then, the fact that $\gamma^{\mathcal{S}_1}$ and $\gamma^{\mathcal{S}_2}$ correspond can be expressed as

$$\boldsymbol{\mathcal{R}}^{\mathcal{S}_1}_{\gamma^{\mathcal{S}_1}} g = \boldsymbol{\mathcal{R}}^{\mathcal{S}_2}_{\gamma^{\mathcal{S}_2}} T g, \tag{5.1}$$

for the pointwise map $T : \mathcal{S}_1 \to \mathcal{S}_2$ and a function $g : \mathcal{S}_1 \to \mathbb{R}$ defined on $\mathcal{S}_1$. Of course, T is not given and hard to compute as discussed above. Hence, in the following we replace T by the functional map T_F. In the discrete setting we represent $\mathcal{S}_1$ and $\mathcal{S}_2$ by meshes M_1 and M_2. The functions g are given in form of vectors $\boldsymbol{g} = \left(g\left(v_1\right), \ldots, g\left(v_{|V_1|}\right)\right) \in \mathbb{R}^{|V_1|}$, which contain per-vertex function values. Furthermore, we assume that the curves c^{M_1} on M_1 and c^{M_2} on M_2 are uniformly sampled at N_s samples.

Recall that it is necessary to represent functions $\boldsymbol{g}$ in a lower dimensional basis (e.g. the Laplace–Beltrami eigenbasis $\boldsymbol{\Phi}_M \in \mathbb{R}^{|V|\times k}$) in order to map them by a functional map $\tilde{\boldsymbol{F}}$ (cf. Section 2.6.3):

$$\tilde{\boldsymbol{g}} = \boldsymbol{\Phi}_M^\top \cdot \boldsymbol{D}_M \cdot \boldsymbol{g}, \tag{5.2}$$

where $\boldsymbol{D}_M$ denotes the lumped mass matrix. To reconstruct the functions from the spectral domain we apply (cf. Section 2.3):

$$\boldsymbol{g} \approx \boldsymbol{\Phi}_M \cdot \tilde{\boldsymbol{g}}, . \tag{5.3}$$

Hence, we can approximate $T \cdot \boldsymbol{g}$ by inserting equations (5.2) and (5.3) into equation (5.1):

$$T \cdot \boldsymbol{g} \approx \boldsymbol{\Phi}_{M_2} \cdot \tilde{\boldsymbol{F}} \cdot \tilde{\boldsymbol{g}}.$$

With this we get the (discrete) equation

$$\boldsymbol{\mathcal{R}}_{c^{M_1}}^{M_1} \boldsymbol{g} \approx \boldsymbol{\mathcal{R}}_{c^{M_2}}^{M_2} \boldsymbol{\Phi}_{M_2} \cdot \tilde{\boldsymbol{F}} \cdot \boldsymbol{\Phi}_{M_1}^\top \cdot \boldsymbol{D}_{M_1} \cdot \boldsymbol{g}.$$

Note that the same function $\boldsymbol{g}$ appears on both sides of the equation. Hence, in this formulation corresponding functions are not required and we can pick an arbitrary function which is mapped between the two shapes. If we transport the Laplace–Beltrami eigenfunctions, we obtain the following simplified equation, since $\boldsymbol{\Phi}_{M_1}^\top \cdot \boldsymbol{D}_{M_1} \cdot \boldsymbol{\Phi}_{M_1} = \boldsymbol{Id}$:

$$\boldsymbol{\mathcal{R}}_{c^{M_1}}^{M_1} \boldsymbol{\Phi}_{M_1} \approx \boldsymbol{\mathcal{R}}_{c^{M_2}}^{M_2} \boldsymbol{\Phi}_{M_2} \cdot \tilde{\boldsymbol{F}},$$

which can be transformed into a linear least squares optimization problem with the objective:

$$\mathcal{E}_{CC}\left(\tilde{\boldsymbol{F}}\right) = \left\| \boldsymbol{\mathcal{R}}_{c^{M_2}}^{M_2} \boldsymbol{\Phi}_{M_2} \cdot \tilde{\boldsymbol{F}} - \boldsymbol{\mathcal{R}}_{c^{M_1}}^{M_1} \boldsymbol{\Phi}_{M_1} \right\|_{\text{Fro}}^2 . \tag{5.4}$$

In the following we denote $\boldsymbol{\Phi}_{c^{M_1}} = \boldsymbol{\mathcal{R}}_{c^{M_1}}^{M_1} \Phi_{M_1}$ and $\boldsymbol{\Phi}_{c^{M_2}} = \boldsymbol{\mathcal{R}}_{c^{M_2}}^{M_2} \Phi_{M_2}$, which are $N_s \times k$ matrices representing the basis functions restricted to the respective curves.

Given a collection of q corresponding curves $c_1^{M_1}, \dots, c_q^{M_1}$ and $c_1^{M_2}, \dots, c_q^{M_2}$, we apply the constraint (5.4) in the form of a penalty together with the commutativity penalty (cf. Section 2.6.3), to find the functional map matrix $\tilde{\boldsymbol{F}}$ minimizing the energy

$$\mathcal{E}_{CFM}\left(\tilde{\boldsymbol{F}}\right) = \alpha \left\| \tilde{\boldsymbol{F}} \boldsymbol{\Lambda}_{M_1} - \boldsymbol{\Lambda}_{M_2} \tilde{\boldsymbol{F}} \right\|_{\text{Fro}}^2 + \sum_{\ell=1}^{q} \left\| \boldsymbol{\Phi}_{c_\ell^{M_2}} \tilde{\boldsymbol{F}} - \boldsymbol{\Phi}_{c_\ell^{M_1}} \right\|_{\text{Fro}}^2 . \tag{5.5}$$

5.2.4 Iterative Update

To give immediate feedback to the user after adding, deleting, or updating a curve, we present an efficient technique to update the functional map with a new objective term in (5.5). Each user input updates the second (data) term of the energy $\mathcal{E}_{CFM}\left(\tilde{\boldsymbol{F}}\right)$. Since this is a least-squares problem, implicitly we need to solve a linear system $\mathbf{H}\mathbf{x} = \mathbf{w}$ in each iteration (here $\mathbf{x}$ is the k^2-dimensional vectorized form of the $k \times k$ functional map matrix $\tilde{\boldsymbol{F}}$), which has a complexity of $\mathcal{O}(k^6)$ for a dense $\mathbf{H} \in \mathbb{R}^{k^2 \times k^2}$.

Since we can reuse previous map estimates as initial guesses, in our implementation iterative solvers converge quickly and lead to a performance improvement over direct solvers. Our linear system is guaranteed to be positive-definite, since it comes from a least-squares problem. Hence, we formulate an efficient optimization procedure adapting the *conjugate gradients* (CG) algorithm for positive definite systems.

The conjugate gradients algorithm finds minima of convex functions of the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} - \mathbf{x}^\top \mathbf{w} + w_0$ [HS52, Sol15]; the first-order optimality condition is the linear system $\nabla f(\mathbf{x}) = \mathbf{H}\mathbf{x} - \mathbf{w} = \mathbf{0}$. It requires multiplication by the positive-definite matrix $\mathbf{H}$ (the linear part of ∇f) as well as access to the elements of $\mathbf{w}$ (the constant part of ∇f); it does not require direct access to elements of $\mathbf{H}$.

In our case, computing the matrix derivative of $\mathcal{E}_{CFM}\left(\tilde{\boldsymbol{F}}\right)$ with respect to $\tilde{\boldsymbol{F}}$ yields the gradient in the canonical form

$$\nabla_{\tilde{\boldsymbol{F}}}\mathcal{E}_{CFM}\left(\tilde{\boldsymbol{F}}\right) = \underbrace{-2\sum_{\ell=1}^{q}\boldsymbol{\Phi}_{c_\ell^{M_2}}^{\top}\boldsymbol{\Phi}_{c_\ell^{M_1}}}_{\boldsymbol{W}_0}$$
$$\underbrace{+2\alpha\left(\boldsymbol{\Lambda}_{M_2}^2\tilde{\boldsymbol{F}}-\boldsymbol{\Lambda}_{M_1}\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}_{M_2}-\boldsymbol{\Lambda}_{M_2}\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}_{M_1}+\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}_{M_1}^2\right)+2\sum_{\ell=1}^{q}\boldsymbol{\Phi}_{c_\ell^{M_2}}^{\top}\boldsymbol{\Phi}_{c_\ell^{M_2}}\tilde{\boldsymbol{F}}}_{\mathbf{H}(\tilde{\boldsymbol{F}})}$$

where $\mathbf{H}(\tilde{\boldsymbol{F}})$ and $\boldsymbol{W}_0$ denote the linear and constant terms, respectively, and we used the fact that $\boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_2}$ are diagonal to simplify expressions.

A new curve provided by the user adds the $(q+1)$-st terms $\boldsymbol{\Phi}_{c_{q+1}^{M_2}}^{\top}\boldsymbol{\Phi}_{c_{q+1}^{M_2}}\tilde{\boldsymbol{F}}$ and $\boldsymbol{\Phi}_{c_{q+1}^{M_2}}^{\top}\boldsymbol{\Phi}_{c_{q+1}^{M_1}}$ to the summations in $\mathbf{H}(\tilde{\boldsymbol{F}})$ and $\boldsymbol{W}_0$, respectively. Furthermore, the q matrix products $\boldsymbol{\Phi}_{c_\ell^{M_2}}^{\top}\boldsymbol{\Phi}_{c_\ell^{M_2}}$ and $\boldsymbol{\Phi}_{c_\ell^{M_2}}^{\top}\boldsymbol{\Phi}_{c_\ell^{M_1}}$ of size $k \times k$ can be precomputed as they do not change in a typical workflow in which the user adds curves. With these observations, we can formulate the conjugate gradients algorithm to find the optimal functional map matrix $\tilde{\boldsymbol{F}}$. Algorithm 5.1 describes the interactive update procedure discussed above. The conjugate gradients algorithm is elaborated in Algorithm 5.2.

5.2.5 User Feedback

After updating the functional map, the user needs to be able to assess the quality of the resulting map. Unfortunately, conversion into a pointwise map of sufficient quality is not possible at interactive rates using the current available methods (in particular, the method of [ESBC19] that we use in our system). It is possible, however, to visualize the functional map by directly transporting texture mapping coordinates from the source shape $\boldsymbol{M}_1$ to the target shape $\boldsymbol{M}_2$.

function Interactive Functional Maps($\mathbf{\Lambda}_{M_1}$, $\mathbf{\Lambda}_{M_2}$, α)
  $\mathbf{T}_1 \leftarrow 2\alpha\mathbf{\Lambda}_{M_2}^2$
  $\mathbf{T}_2 \leftarrow 2\alpha\mathbf{\Lambda}_{M_1}^2$
  $\boldsymbol{W}_0 \leftarrow \mathbf{0}$
  initialize $\tilde{\boldsymbol{F}}$
  while user inputs a pair of curves $c_{\text{in}}^{M_1}$, $c_{\text{in}}^{M_2}$ **do**
    $\mathbf{\Phi}_{c_{\text{in}}^{M_1}} \leftarrow \boldsymbol{\mathcal{R}}_{c_{\text{in}}}^{M_1} \mathbf{\Phi}_{M_1}$
    $\mathbf{\Phi}_{c_{\text{in}}^{M_2}} \leftarrow \boldsymbol{\mathcal{R}}_{c_{\text{in}}}^{M_2} \mathbf{\Phi}_{M_2}$
    $\mathbf{T}_1 \leftarrow \mathbf{T}_1 + 2\mathbf{\Phi}_{c_{\text{in}}^{M_2}}^{\top} \mathbf{\Phi}_{c_{\text{in}}^{M_2}}$
    $\boldsymbol{W}_0 \leftarrow \boldsymbol{W}_0 + 2\mathbf{\Phi}_{c_{\text{in}}^{M_2}}^{\top} \mathbf{\Phi}_{c_{\text{in}}^{M_1}}$
    $\tilde{\boldsymbol{F}} \leftarrow \text{ConjGrad}(\tilde{\boldsymbol{F}}, \mathbf{\Lambda}_{M_1}, \mathbf{\Lambda}_{M_2}, \alpha, \mathbf{T}_1, \mathbf{T}_2, \boldsymbol{W}_0)$

Algorithm 5.1: Interactive Functional Maps: In an interactive update loop the user can add curve correspondences $\left(c_{\text{in}}^{M_1}, c_{\text{in}}^{M_2}\right)$. The objective is updated accordingly and handed to the CG procedure for optimization (cf. Algorithm 5.2).

We assume that the source shape M_1 has associated texture mapping coordinates represented as a vector-valued function $(u, v) : M_1 \to [0,1]^2$ on the manifold. In the discrete setting, texture mapping coordinates are represented as a $m \times 2$ matrix $\mathbf{U}$. Application of the functional map is performed by first computing the representation in the Laplace basis of the texture mapping coordinates on M_1, applying the matrix $\tilde{\boldsymbol{F}}$ to the obtained coefficients, and then reconstructing them on the manifold M_2:

$$\mathbf{TU} \approx \mathbf{\Phi}_{M_2} \tilde{\boldsymbol{F}} \mathbf{\Phi}_{M_1}^{\top} \boldsymbol{D}_{M_1} \mathbf{U}.$$

This operation is cheap and can be performed at interactive rates.

Visualizing the transported texture on the target mesh directly gives the user feedback about where the map needs to be improved. For example, in Figure 5.6 the user has added several curves, and the texture that is transported by the

function ConjGrad($\tilde{\boldsymbol{F}}_1, \boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_1}, \alpha, \mathbf{T}_1, \mathbf{T}_2, \mathbf{W}_0$)
  $\mathbf{R}_1 \leftarrow \mathbf{W}_0$ - H($\tilde{\boldsymbol{F}}_1, \mathbf{T}_1, \mathbf{T}_2, \boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_2}, \alpha$)
  $\mathbf{P}_1 \leftarrow \mathbf{R}_1$
  $\gamma_1 \leftarrow \sum_{ij}(\mathbf{R}_1 \odot \mathbf{R}_1)_{ij}$
  for $k \leftarrow 1 \ldots N_{\text{iter}}$ **do**
    $\alpha_k \leftarrow \frac{\gamma_k}{\sum_{ij}(\mathbf{P}_k \odot \mathrm{H}(\mathbf{P}_k, \mathbf{T}_1, \mathbf{T}_2, \boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_2}, \alpha))_{ij}}$
    $\tilde{\boldsymbol{F}}_{k+1} \leftarrow \tilde{\boldsymbol{F}}_k + \alpha_k \cdot \mathbf{P}_k$
    $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_k - \alpha_k \cdot \mathrm{H}(\mathbf{P}_k, \mathbf{T}_1, \mathbf{T}_2, \boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_2}, \alpha)$
    $\gamma_{k+1} \leftarrow \sum_{ij}(\mathbf{R}_{k+1} \odot \mathbf{R}_{k+1})_{ij}$
    if $\gamma_{k+1} \leq 10^{-6}$ **then return** $\tilde{\boldsymbol{F}}_{k+1}$
    else
      $\beta_k = \frac{\gamma_{k+1}}{\gamma_k}$
      $\mathbf{P}_{k+1} \leftarrow \mathbf{R}_{k+1} + \beta_k \cdot \mathbf{P}_k$
  return $\tilde{\boldsymbol{F}}_{N_{\text{iter}}}$
function H($\tilde{\boldsymbol{F}}, \mathbf{T}_1, \mathbf{T}_2, \boldsymbol{\Lambda}_{M_1}, \boldsymbol{\Lambda}_{M_2}, \alpha$)
  return $\mathbf{T}_1\tilde{\boldsymbol{F}} + \tilde{\boldsymbol{F}}\mathbf{T}_2 - \alpha(\boldsymbol{\Lambda}_{M_1}\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}_{M_2} + \boldsymbol{\Lambda}_{M_2}\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}_{M_2})$

Algorithm 5.2: Conjugate Gradients: Minimizes the objective (5.5) for a fixed set of curve correspondences.

resulting functional map, adequately maps semantically corresponding parts onto each other. If the user is satisfied with the quality of the map, the interface has an option allowing to run the slower higher-quality point-to-point conversion method of [ESBC19].

5.2.6 Applications and Results

In this section, we show how curve-constrained functional maps can be used to obtain high quality point-to-point correspondences (Section 5.2.6.1). Furthermore, we provide qualitative and quantitative evaluation of our method and compare to previous approaches (Section 5.2.6.2). In all the qualitative correspondence

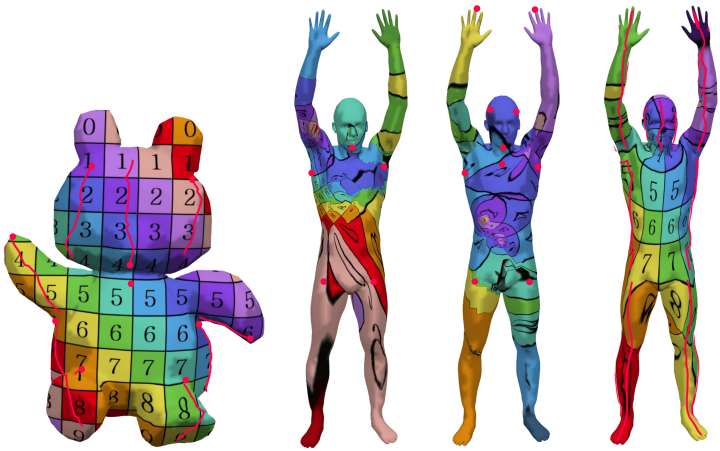


Figure 5.7: Qualitative comparison of our method (rightmost column) to [NO17] and [ESBC19] on non-isometric shapes, visualized with texture mapping. The reference shape is shown in the leftmost image. The magenta curves/dots indicate the user-input given to the respective method The correspondences recovered with our method are more accurate as it is able to deal with significant metric distortions.

results shown, unless indicated otherwise, the final pointwise correspondences are obtained from functional maps with the method of [ESBC19]. For our results we set $k = 120$ (number of eigenfunctions) and $\alpha = 0.5$.

5.2.6.1 Point-to-point correspondences

As with previous work on functional maps, our functional maps can be used as input to pointwise correspondence computation. Beyond the simple map recovery technique proposed in the original functional maps paper [OBCS*12], Ezuz et

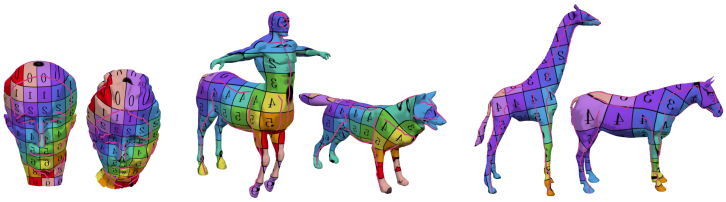


Figure 5.8: Examples of correspondence results produced by our method, visualized with texture mapping. Feature curves used as the input to our optimization procedure are shown in red.

al. [ESBC19] propose a method that, given pointwise correspondences from M_1 to M_2 and vice versa, refines them by minimizing a harmonic-style energy (cf. Section 2.6.3). We use their method as the final stage of our pipeline as follows. First, we compute the functional maps $\tilde{F}_{12}$ and $\tilde{F}_{21}$ in both directions between the shapes. Second, we extract an initial pointwise correspondence by nearest neighbor search as in [OBCS*12]. Finally, we refine the initial correspondences with the method of [ESBC19]. Results are shown in Figures 5.7, 5.8, and 5.9.

In Figure 5.7, we provide a qualitative comparison to [ESBC19] and [NO17]. These are automatic approaches, which are based on surface descriptors and require only sparse user input in form of point correspondences. We use the authors' implementations to compute the results. For the comparison to [ESBC19] (Figure 5.7, third column), we use their entire pipeline, employing 200-dimensional WKS and WKM functions (200 dimensions each) constructed on 21 landmark points picked by hand, leading to 4400 function-preservation constraints. For the method of [NO17] (Figure 5.7, second column), we used the authors' implementation based on 200 wave-kernel signature functions and 6 wave-kernel-map functions (with 200 dimensions each, also provided by hand), i.e. with 1400 function preservation constraints, to compute the functional maps in both directions, and then proceed as in our method (first compute the initial correspondence as in [OBCS*12] and then refine with [ESBC19]). The functional map computed

Figure 5.9: Correspondences between surfaces with *extremely distorting* constraints and varied geometry, visualized with texture mapping. Feature curves used as the input to our optimization procedure are shown in red. Note in the left example that we map the head of the wolf onto the tail of the cat and vice versa.

with our method gets as input 18 feature curve constraints shown in Figure 5.7. Methods for recovering pointwise correspondences from functional maps like [ESBC19] are usually computationally intensive. Since our method allows the user to control the quality of the functional map at runtime, the initialization of the pointwise correspondence recovery methods is reliable, implying that the user would typically have to run it only once at the end of the interactive session. Even though the corresponding parts are very non-isometric (see e.g. the legs of the bear and the human in Figure 5.7), the correspondence obtained with our curve-based approach is accurate. In contrast, automatic methods with little user input (usually in the form of point correspondences for the wave-kernel map) output inaccurate correspondences when shapes vary strongly in terms of conformal factor and other distortion measures.

Figure 5.9 shows maps between very challenging mesh pairs, including extremely non-isometric deformations and high-frequency details as well as a back to front map. While we do not expect to find low-distortion maps in these cases, our correspondences obey the curve constraints and are relatively meaningful.

5.2.6.2 Correspondence Accuracy

To evaluate the quality of the correspondence, we first recover the pointwise correspondence from the functional maps using nearest neighbors [OBCS*12],

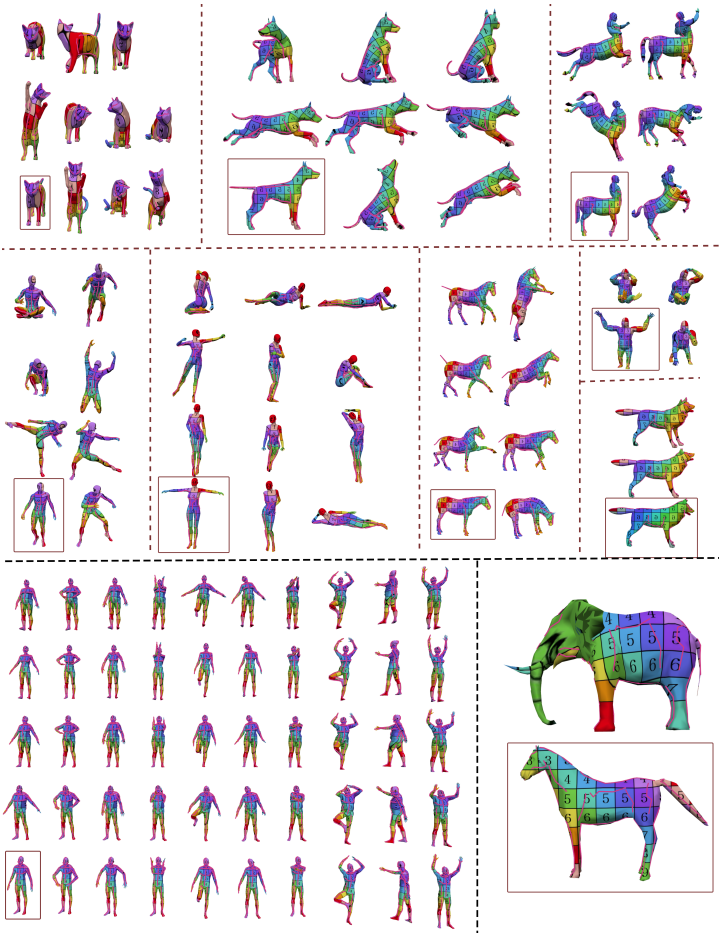


Figure 5.10: TOSCA, FAUST, and horse-elephant data set labeled with corresponding feature curves. Since these data sets have prelabled point-correspondences, matching feature curves can be defined by following the same vertices along the curve. The texture coordinates are mapped by the functional map that is computed based on the feature curve constraints. The source shape for each shapes' group is indicated by a box. We only show half of the FAUST shapes here.

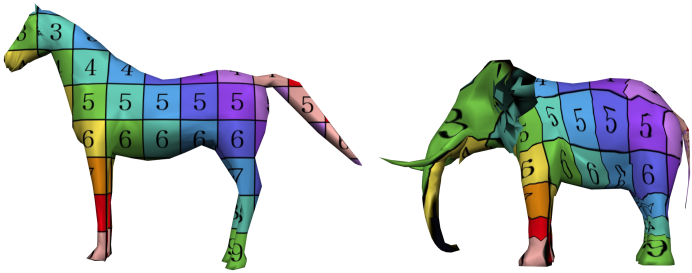


Figure 5.11: Ground-truth correspondences of the far from isometric elephant-horse data set.

and then plot the percentage of correspondences found within a certain geodesic radius (in % of the target shape geodesic diameter) using the Princeton benchmark [KLF11]. The results are depicted in Figure 5.12, where we show the average correspondence accuracy on the high-resolution *TOSCA* [BBK08] and *FAUST* [BRLB14] data sets, as well as a far-from-isometric data set containing a pair of horse and elephant shapes (cf. Figure 5.10). These data sets contain pre-labeled point-correspondences with which we are able to evaluate the quality of our computed maps. In Figure 5.10, we show the labeled feature curve correspondences. The texture is transported from the source shape (indicated by a box) to the target shapes with the functional map computed based on the feature curve constraints.

We compare our results to [NO17] and to an interactive alternative using just the descriptor-based linear solver in [OBCS*12] in Figure 5.12 (left). [NO17] is one of the recent state-of-the-art automatic functional map computation method. While it achieves high correspondence accuracy, this approach is computationally heavy and cannot run at interactive speeds. A possible interactive alternative to our approach can be implemented by solving the linear system $\tilde{\boldsymbol{F}}\tilde{\boldsymbol{G}}_1 = \tilde{\boldsymbol{G}}_2$, i.e. the descriptor preservation constraints presented in [OBCS*12]. For the functional map computation we provide the methods with 1400 WKS/WKM-function

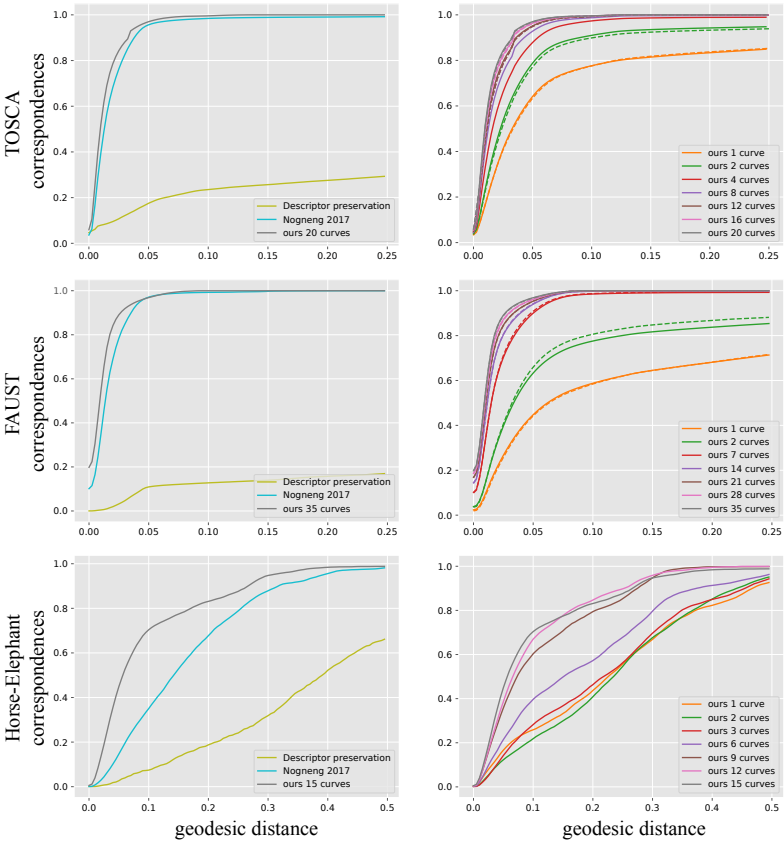


Figure 5.12: Correspondence quality of different methods on TOSCA (top), FAUST (center), and the Horse-Elephant pair (bottom). Point correspondences were obtained from the functional maps using nearest neighbors as in [OBCS*12]. Left we show the results of our method, descriptor preservation constraints presented in [OBCS*12], and [NO17]. The advantage of our method is especially pronounced on non-isometric shapes. On the right we used perfectly corresponding curves (solid) and shortest path curves computed between perfectly corresponding endpoints (dashed). Almost no degradation in performance is observed in the latter case, allowing us to conclude that our shortest path method approximates the feature curves sufficiently well.

preservation constraints as described above.

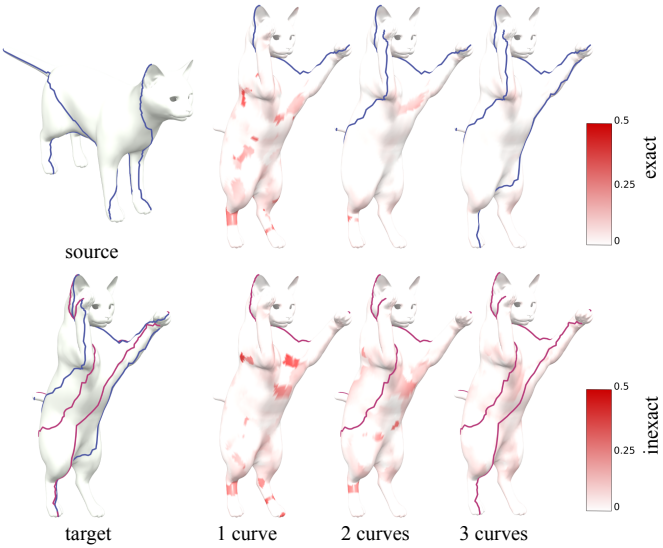
We vary the following test parameters in evaluating correspondence accuracy:

Increasing number of feature curves. We plot the correspondence accuracy for functional maps based on an increasing number of curves in Figure 5.12 (right). These pre-labeled corresponding curves follow features on the mesh. For a given number of curves we run the correspondence test multiple times by selecting a random subset from the prelabled feature curves. Examples for all data sets are given in Figure 5.10. By increasing the number of curves used as input, the correspondence accuracy is improved. We also observe that functional maps based on only one pair of corresponding curves already produces more accurate correspondences than that based on descriptor preservation solving $\tilde{\boldsymbol{F}}\tilde{\boldsymbol{G}}_1 = \tilde{\boldsymbol{G}}_2$.

By increasing the number of curves, we can even achieve a higher match rate than [NO17], even though the shapes in the TOSCA and FAUST data sets are nearly isometric deformations of one another. For these nearly-isometric pairs, state-of-the-art functional map methods are expected to perform well.

Imperfect correspondence information. The solid curves in Figure 5.12 (right) use perfectly corresponding curves (i.e. they follow the same path of point-correspondences). In contrast, the dashed plots are found by selecting the same start- and end-point and tracing the intermediate path with shortest path search. We observe that there is no significant difference in the results: The shortest path search provides a reliable method to select corresponding curves.

To evaluate the effect of imperfect correspondence information, we plot the geodesic error for correspondences found with [OBCS*12] based on functional maps with imperfect input curve correspondences. In Figure 5.13 the 3 blue curves are perfectly corresponding, while the magenta curves on the target are found by shortest path search on the target mesh between corresponding start- and end-points. Since start- and end-points lie far apart, the correspondences are inexact. We show the geodesic error in a range from 0 (white) to 0.5 (red) which



Curves	1 curve	2 curves	3 curves
exact	0.0387	0.0243	0.0043
inexact	0.0432	0.0368	0.0298

Figure 5.13: Geodesic error for imperfect correspondence information with increasing number of curves. We select 3 curves with distant start and end point. The blue curves indicate perfect correspondence, while the magenta curves show the shortest path curves found on the target mesh between start- and end-point. We visualize the geodesic error for an increasing number of corresponding curves in a range from 0 (white) to 0.5 (red). The geodesic error is higher at points where the curves deviate a lot (e.g. at the belly of the cat). The average geodesic error is given in the table above.

is obtained from the functional maps with exact (top row) and inexact (bottom row) feature curve correspondences for an increasing number of curves (left to right). In regions where the curves deviate a lot, the geodesic error is higher

(e.g. the belly of the cat). However, although locally the error is higher, the average geodesic distance is still quite low (0.029 with inexact correspondence information and 0.0043 with exact correspondences for 3 curves).

Far from isometric and baseline test. We plot the correspondence quality for the horse-elephant data set in Figure 5.12 (left, bottom row), where we show the match characteristics up to a total geodesic distance of 0.5. With the curve-constrained functional maps we can achieve much higher match accuracy compared to previous methods. This data set is especially challenging due to large distortion (see Figure 5.11 for a visualization of the ground-truth map).

In Figure 5.14 we show the match characteristics on the SHREC [VtH07] data set, which contains several highly non-isometric shapes with partial correspondence information. We use all meshes from the SHREC data set where a user can unambiguously label 10 corresponding curves without the need to see the ground truth (e.g. we did not use rotationally symmetric vases). As above we compare to [NO17] with 1400 function preservation constraints (WKS and WKM). Secondly, we perform a baseline comparison by providing the 10 feature curves as functions to [NO17]. The functions are set to -1 everywhere, but at the vertices along the feature curve. To each vertex along the curve we assign the value of the arc-length parametrization (normalized to a total length of 1). Correspondences computed with functional maps based on both the baseline test and the wave-kernel descriptors show higher geodesic distances to the ground truth than our method.

5.2.6.3 Timing

In Figure 5.15, we compare our timings for optimization and preprocessing to the results in [NO17] for a different number of curve constraints. In this comparison we provide the same user input to our method and to [NO17]. Results are computed on a commodity laptop on meshes with 27,560 (pig) and 33,638 (dog) vertices. For [NO17], we use 200 functions of the wave-kernel signature and the start and endpoints of the feature curves (user input) for the computation

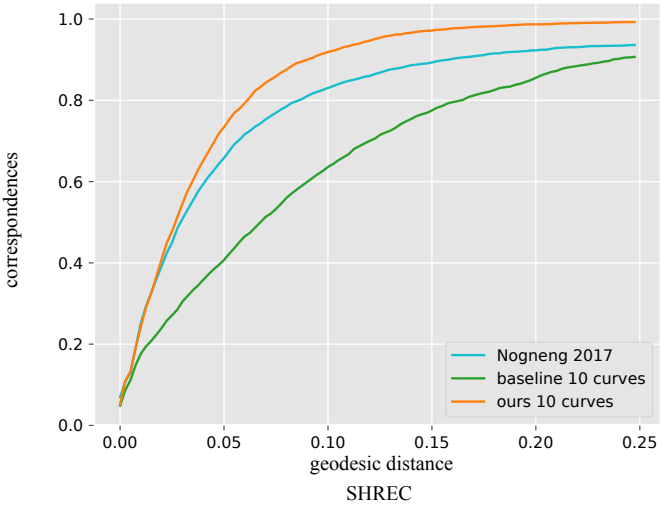
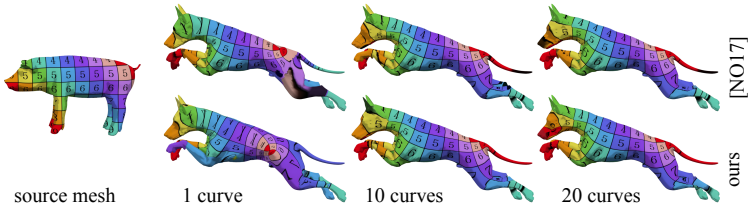


Figure 5.14: Correspondence quality of our method and [NO17] with descriptors (WKS and WKM) and curve functions (baseline test) on the SHREC data set [VtH07]. Point correspondences were obtained from the functional maps using nearest neighbors as in [OBCS*12]. Our method shows a higher correspondence accuracy than the compared approaches.

of the wave-kernel map with 200 functions each; for [NO17], the selection of these points at interactive speeds is not possible. The results with our method are obtained in under a second. Furthermore, we can observe that with our method we get slightly improved results (see e.g. the front foot and tail for 10 curves and the mouth and tail for 20 curves).

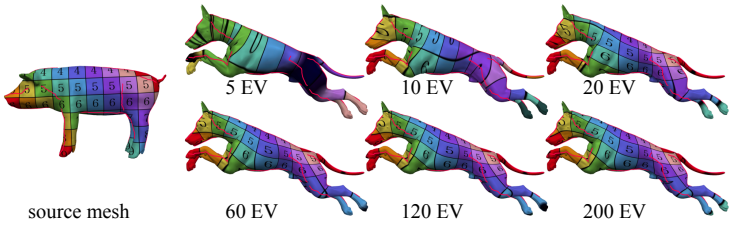
We compare the runtime of our conjugate gradients implementation with the quasi-Newton solver used by [NO17]. For this we optimize our curve based functional map objective (5.5) with both methods for a different number of curves. In



Method	Number of Curves	Preprocessing	Optimization
[NO17]	1 curve / 2 points	34.823s	24.223s
[NO17]	10 curve / 20 points	267.815s	99.592s
[NO17]	20 curve / 40 points	555.921s	147.252s
our method	1 curve	0.003s	0.145s
our method	10 curves	0.008s	0.624s
our method	20 curves	0.007s	0.642s

Figure 5.15: Timings and qualitative comparison to [NO17]. We provide a point-to-point mapping as input to the point-to-point reconstruction method [ESBC19] for our method and [NO17]. We use the same correspondences (start- and endpoints of the curves) for the WKM computation in [NO17]. The table below gives the timings for preprocessing (setting up descriptors and matrices) and the functional map optimization. Note that the selection of point-correspondences cannot be performed at interactive speeds for [NO17]. Furthermore, parts of the mesh show a higher correspondence accuracy with our method (see e.g. the front foot and tail for 10 curves and the mouth and tail for 20 curves).

contrast to [NO17], our conjugate gradients implementation provides feedback at interactive rates.



Number of EV (k)	5	10	20	60	120	200
Eigenvector computation	0.526s	0.610s	0.789s	1.618s	5.148s	8.993s
Preprocessing	0.005s	0.005s	0.005s	0.006s	0.007s	0.012s
Optimization	0.005s	0.006s	0.008s	0.059s	0.642s	2.788s

Figure 5.16: Timings and qualitative comparison for different numbers of Laplace eigenvectors (EV). The qualitative comparison shows the texture coordinates that are transported by the functional map computed with the respective number of eigenfunctions and 20 curve constraints. Although maps are slightly sharper with higher amount of eigenfunctions, the qualitative difference between results is not significant for $k > 20$.

Number of Curves	Quasi-Newton Solver	Conjugate Gradients
1 curve	3.955s	0.145s
10 curves	4.724s	0.624s
20 curves	5.077s	0.642s

Solving a dense linear system of the form $\mathbf{Hx} = \mathbf{w}$ with $\mathbf{H} \in \mathbb{R}^k$ has a complexity of $\mathcal{O}(k^6)$. Our iterative implementation converges quickly, leading to interactive response times.

We show the runtime of the initialization and optimization procedure in Figure 5.16 for increasing k. The images (top) show the texture coordinates transported with the respective functional map. Although the map is slightly sharpened for increasing k, for $k > 20$ no significant changes are visible in the qualitative com-

parison. For $k = 200$ the runtime increases beyond interactive rates. However, for the visualization of the functional map less eigenfunctions are sufficient.

Limitations One issue we observed is that the feedback of the interactive tool given by the texture coordinates mapped by the functional map can be inaccurate along (non-smooth) seams in the texture. Nonetheless, it provides an effective way to roughly evaluate the quality of the map during its design. Then in a followup step, the user can compute an exact pointwise map with the provided functional map.

Furthermore, our results show that several curve constraints are required to obtain a meaningful functional map (i.e. one curve-pair is usually not sufficient). However, the input the user has to provide is still quite sparse. We have shown that inexact curve constraints (based on automatically traced shortest paths between user-provided endpoints) are sufficient to obtain high-quality maps. Hence, curve tracing can be performed automatically given the start- and end-points of the curves.

We also observe that it is hard to provide meaningful curve-based constraints if one shape has parts that are absent in another one (e.g. elephant trunks) or very different from the corresponding parts (e.g., elephant ears). In such cases it is hard to obtain a reasonable bijective map. Previous work has dealt with the problem of partial functional maps [RCB*17]. An interesting direction for future research might be to combine these approaches with our method, especially due to the high efficiency and correspondence accuracy which can be obtained.

In contrast to related work that opts to find increasingly elaborate and computationally intensive approaches to functional maps, we present a method which is both simple (easy to implement) and efficient such that it runs at interactive speeds. In our tests we find that by adding feature curve constraints, we can outperform state-of-the art automatic approaches in terms of correspondence accuracy.

This new interactive direction for functional maps has two major benefits compared to previous work. First, it allows to compute smooth maps between

semantically similar objects, even if they vary geometrically. Secondly, results are obtained at interactive speeds and can be edited and evaluated by a user. For non-expert users, feature curve based functional maps are especially useful, since they avoid the requirement to engineer descriptors specific to the regarded shape family.

5.3 Summary

In this Chapter we have demonstrated the usefulness of meaningful FCNs in exemplary applications. Feature curves capture abstract information about the local and global shape of an object. Combined with correspondence or level-of-detail information, they provide an informative and sparse representation. Throughout the discussed applications we observe that feature curves add this abstract information to guide the respective application from a more global perspective. In each case we observed more accurate, enhanced results specific to the given objective. The fact that feature curves can be used to guide low-level geometry processing tasks as well as high-level shape analysis applications, shows that they hold both valuable geometric and semantic cues. The geometry of the feature curves is crucial to guide geometry processing applications, but also allows inference about the structure of the FCN (e.g. correspondences). The (inferred) topological information can be of value in high-level tasks, where global structural knowledge is required. Hence, the use of FCNs is extremely flexible such that various applications can benefit from the different levels of abstraction they provide.

6 Conclusion

This thesis was motivated by the fact that feature curves are a sparse and abstract but potentially informative high-level representation of geometry. While the use of feature curves was well established in geometry processing applications, there were only few related works based on feature curves in the field of shape analysis. Although meaningful feature curve networks yield more appealing results in geometry processing tasks, as we have shown in Section 5, using e.g. locally curvature thresholded feature curves in this context can be sufficient, depending on the expected quality. However, to make FCNs applicable to shape analysis applications, it is vital that the included feature curves are meaningful and do not contain any noise. Furthermore, high-level information such as reoccurrence or template membership are important structural cues. With this high-level information we are able to introduce FCNs to shape analysis applications.

Hence, in this thesis we have developed a framework which enables

1. the evaluation of feature curves based on local and global saliency measures,

2. the detection of structure based on reoccurrence of feature curves and feature curve templates,

3. the completion of FCNs from raw noisy input FCNs based on feature curve templates,

4. the creation of globally scale conforming FCNs based on local and global saliency measures,

5. feature curve constrained low-level geometry processing applications, and

6. inter-surface functional maps based on feature curve constraints.

The feature curve networks that can be obtained from this framework encode low-level geometric information, intermediate-level topological connectivity of the curves, as well as high-level structural information and saliency cues. FCNs which are augmented with these different levels of abstraction, contain valuable data which can be used to guide and simplify tasks from low-level geometry processing to high-level shape analysis. E.g. we have shown that the guidance with meaningful feature curves highly improves the resulting surface quality in tasks such as smoothing and remeshing, and on the other end of the scale how the knowledge of corresponding feature curves can have highly positive impact for complex tasks such as inter-surface mappings between shapes (cf. Section 5.2). The building blocks of the presented framework build upon each other in this thesis. However, since all techniques are based on arbitrary feature curve input, they are quite flexible and can be inserted into various pipelines which handle geometric curve data.

6.1 Outlook

Meaningful feature curve networks provide a sparse yet informative representation of the geometric data, which makes them particularly useful for shape analysis. In this thesis, we have mostly dealt with shape analysis based on co-occurrence and shape matching. However, there are various other tasks which benefit from the sparse and salient representation of the feature curves.

E.g. surface segmentations are typically based on patches that align with features. It might be valuable to investigate co-segmentations based on corresponding feature curve networks. Furthermore, the identification of class specific feature curves might be particularly interesting for shape retrieval in large databases, due to the sparse representation of the feature curves. By augmenting the curves with additional meta information, it might even be possible to compute statistics of

the reoccurrence of specific feature curves.

Another promising direction is to learn information about feature curves via neural networks. As we have demonstrated by our supervised feature curve correspondence learning, the representations learned by a network drastically outperform handcrafted models. Especially, due to the possibility of training sequence data with RNNs, it might be possible to learn statistics or saliency information on feature curves. Apart from supervised techniques, it can also be worth investigating unsupervised learning models in this direction. E.g. if we could evaluate the quality of the functional maps computed in Section 5.2 by a loss function, we would be able to compute the required feature curves fully automatically.

Overall, there are various tasks in the field of shape analysis where feature curves have the potential to introduce valuable information on the different levels of abstraction.

Bibliography

[ACK01] Amenta N., Choi S., Kolluri R. K.: The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications* (2001), ACM, pp. 249–266.

[ADVDI03] Alliez P., De Verdire E., Devillers O., Isenburg M.: Isotropic surface remeshing. In *Shape Modeling International* (2003), IEEE, pp. 49–58.

[AGM*90] Altschul S. F., Gish W., Miller W., Myers E. W., Lipman D. J.: Basic local alignment search tool. *Journal of molecular biology 215*, 3 (1990), 403–410.

[ASC11] Aubry M., Schlickewei U., Cremers D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops),* (2011), IEEE, pp. 1626–1633.

[AUGA08] Alliez P., Ucelli G., Gotsman C., Attene M.: Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring, Mathematics and Visualization* (2008), Springer.

[BBK06] Bronstein A. M., Bronstein M. M., Kimmel R.: Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences 103*, 5 (2006), 1168–1172.

[BBK08] Bronstein A., Bronstein M., Kimmel R.: *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 2008.

[BBW*09] Bokeloh M., Berner A., Wand M., Seidel H.-P., Schilling A.: Symmetry detection using feature lines. In *Computer Graphics Forum* (2009), vol. 28, pp. 697–706.

[BBX95] Bajaj C. L., Bernardini F., Xu G.: Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 109–118.

[BCE*13] Bommes D., Campen M., Ebke H.-C., Alliez P., Kobbelt L.: Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 98.

[Bis06] Bishop C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[BK04] Botsch M., Kobbelt L.: A remeshing approach to multiresolution modeling. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2004), ACM, pp. 185–192.

[BK07] Bommes D., Kobbelt L.: Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. *Proceedings of the International Conference on Vision, Modeling and Visualization 7* (2007), 151–160.

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B.: *Polygon mesh processing*. AK Peters/CRC Press, 2010.

[BL18] Bukenberger D. R., Lensch H. P. A.: Hierarchical Quad Meshing of 3D Scanned Surfaces. *Computer Graphics Forum 37*, 5 (2018), 131–141.

[BM92] Besl P. J., McKay N. D.: Method for registration of 3d shapes. In *Robotics-DL tentative* (1992), International Society for Optics and Photonics, pp. 586–606.

[BMP01] Belongie S., Malik J., Puzicha J.: Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems* (2001), pp. 831–837.

[BMRB16] Boscaini D., Masci J., Rodolà E., Bronstein M.: Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems* (2016), pp. 3189–3197.

[BPK98] Belyaev A. G., Pasko A. A., Kunii T. L.: Ridges and ravines on implicit surfaces. In *Computer Graphics International, 1998*. (1998), pp. 530–535.

[BRLB14] Bogo F., Romero J., Loper M., Black M. J.: FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), IEEE.

[BWM*11] Berner A., Wand M., Mitra N. J., Mewes D., Seidel H.-P.: Shape analysis with subspace symmetries. *Computer Graphics Forum 30*, 2 (2011).

[BZK09] Bommes D., Zimmer H., Kobbelt L.: Mixed-integer quadrangulation. In *ACM Transactions On Graphics (TOG)* (2009), vol. 28, ACM, p. 77.

[CBK12] Campen M., Bommes D., Kobbelt L.: Dual loops meshing: Quality quad layouts on manifolds. *ACM Transactions on Graphics (TOG) 31*, 4 (2012).

[CDGDS13] Crane K., De Goes F., Desbrun M., Schröder P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (2013), ACM, p. 7.

[CGT] Cgtrader. https://www.cgtrader.com/.

[CHK13] Campen M., Heistermann M., Kobbelt L.: Practical anisotropic geodesy. *Computer Graphics Forum 32*, 5 (2013), 63–71.

[CJL11] Chiang C.-H., Jong B.-S., Lin T.-W.: A robust feature-preserving semi-regular remeshing method for triangular meshes. *The Visual Computer 27*, 9 (2011), 811–825.

[COO15] Carrière M., Oudot S. Y., Ovsjanikov M.: Stable topological signatures for points on 3d shapes. *Computer Graphics Forum 34*, 5 (2015), 1–12.

[CP05] Cazals F., Pouget M.: *Topology driven algorithms for ridge extraction on meshes*. PhD thesis, INRIA, 2005.

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. In *ACM Transactions on Graphics (Proceedings SIGGRAPH 2004)* (2004), ACM, pp. 905–914.

[CSEH05] Cohen-Steiner D., Edelsbrunner H., Harer J.: Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational Geometry* (2005), ACM, pp. 263–271.

[CWW13] Crane K., Weischedel C., Wardetzky M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 152:1–152:11.

[CYW15] Cao Y., Yan D.-M., Wonka P.: Patch layout generation by detecting feature networks. *Computers & Graphics 46* (2015), 275–282.

[DBP05] Del Bimbo A., Pala P.: Retrieval by content similarity of 3d models using spin images. In *Annales des télécommunications* (2005), vol. 60, Springer, pp. 1360–1378.

[Dem09] Demarsin K.: *Extraction of Closed Feature Lines from Point Clouds Based on Graph Theory*. PhD thesis, Numerical Analysis and Applied Mathematics Section, Department of Computer Science, Faculty of Engineering Science, Jan. 2009.

[DGGDV11] De Goes F., Goldenstein S., Desbrun M., Velho L.: Exoskeleton: Curve network abstraction for 3d shapes. *Computers & Graphics 35*, 1 (2011), 112–121.

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[DSSC08] Daniels J., Silva C. T., Shepherd J., Cohen E.: Quadrilateral mesh simplification. *ACM Transactions on Graphics (TOG) 27*, 5 (2008), 148.

[EBC17] Ezuz D., Ben-Chen M.: Deblurring and denoising of maps between shapes. *Computer Graphics Forum 36*, 5 (2017), 165–174.

[EBCK13] Ebke H.-C., Bommes D., Campen M., Kobbelt L.: Qex: robust quad mesh extraction. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 168.

[EC02] Estivill-Castro V.: Why so many clustering algorithms: A position paper. *SIGKDD explorations 4*, 1 (2002), 65–75.

[ECBK14] Ebke H.-C., Campen M., Bommes D., Kobbelt L.: Level-of-detail quad meshing. *ACM Transactions on Graphics (TOG) 33*, 6 (Nov. 2014), 184:1–184:11.

[EHZ01] Edelsbrunner H., Harer J., Zomorodian A.: Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the 17th Annual Symposium on Computational Geometry* (2001), ACM, pp. 70–79.

[EKB*15] Eynard D., Kovnatsky A., Bronstein M. M., Glashoff K., Bronstein A. M.: Multimodal manifold analysis by simultaneous diagonalization of laplacians. *IEEE Transactions on Pattern Analysis and Machine Intelligence 37* (2015), 2505–2517.

[ELZ00] Edelsbrunner H., Letscher D., Zomorodian A.: Topological persistence and simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science* (2000), IEEE, pp. 454–463.

[ERGB16] Eynard D., Rodola E., Glashoff K., Bronstein M. M.: Coupled functional maps. In *4th International Conference on 3D Vision (3DV)* (2016).

[ESBC19] Ezuz D., Solomon J., Ben-Chen M.: Reversible harmonic maps between discrete surfaces. *ACM Transactions on Graphics (TOG)* (2019).

[ESCK16] Ebke H.-C., Schmidt P., Campen M., Kobbelt L.: Interactively controlled quad remeshing of high resolution 3d models. *ACM Transactions on Graphics (TOG) 35*, 6 (2016).

[FB81] Fischler M. A., Bolles R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24* (1981), 381–395.

[FCODS08] Fu H., Cohen-Or D., Dror G., Sheffer A.: Upright orientation of man-made objects. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 42.

[FH75] Fukunaga K., Hostetler L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 1 (Sept. 1975), 32–40.

[For98] Forman R.: A discrete morse theory for cell complexes. *Advances in Mathematics 134* (1998), 90–145.

[GBC16] Goodfellow I., Bengio Y., Courville A.: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GBK16] Gehre A., Bommes D., Kobbelt L.: Geodesic iso-curve signature. In *Proceedings of the International Conference on Vision, Modeling and Visualization* (2016), Eurographics Association, pp. 17–28.

[GBKS18] Gehre A., Bronstein M., Kobbelt L., Solomon J.: Interactive curve constrained functional maps. *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing 37*, 5 (2018).

[GG04] Gelfand N., Guibas L. J.: Shape segmentation using local slippage analysis. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2004), ACM, pp. 214–223.

[GH97] Garland M., Heckbert P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[GI04] Goldfeather J., Interrante V.: A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics (TOG) 23*, 1 (2004), 45–63.

[GLK16] Gehre A., Lim I., Kobbelt L.: Adapting feature curve networks to a prescribed scale. In *Computer Graphics Forum, Proceedings Eurographics* (2016), vol. 35, pp. 319–330.

[GLK18] Gehre A., Lim I., Kobbelt L.: Feature curve co-completion in noisy data. *Computer Graphics Forum, Proceedings Eurographics 37*, 2 (2018).

[GO15] Gurobi Optimization I.: Gurobi optimizer reference manual, 2015.

[GSV*17] Gori G., Sheffer A., Vining N., Rosales E., Carr N., Ju T.: Flowrep: descriptive curve networks for free-form design shapes. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 59.

[GWM01] Gumhold S., Wang X., MacLeod R. S.: Feature extraction from point clouds. In *IMR* (2001).

[HK06] Hornung A., Kobbelt L.: Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2006), pp. 41–50.

[HLR*18] Halimi O., Litany O., Rodolà E., Bronstein A., Kimmel R.: Self-supervised learning of dense shape correspondence. *arXiv preprint arXiv:1812.02415* (2018).

[Hor91] Hornik K.: Approximation capabilities of multilayer feedforward networks. *Neural networks 4*, 2 (1991), 251–257.

[HPW05] Hildebrandt K., Polthier K., Wardetzky M.: Smooth feature lines on surface meshes. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2005), pp. 85–90.

[HS52] Hestenes M. R., Stiefel E.: *Methods of conjugate gradients for solving linear systems*, vol. 49. NBS Washington, DC, 1952.

[HS97] Hochreiter S., Schmidhuber J.: Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[HWG14] Huang Q., Wang F., Guibas L.: Functional map networks for analyzing and exploring large shape collections. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 36.

[Jef98] Jeffreys H.: *The theory of probability*. OUP Oxford, 1998.

[JH99] Johnson A. E., Hebert M.: Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 5 (1999), 433–449.

[KBB*13] Kovnatsky A., Bronstein M. M., Bronstein A. M., Glashoff K., Kimmel R.: Coupled quasi-harmonic bases. *Computer Graphics Forum, Proceedings Eurographics 32* (2013), 439–448.

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2006), Eurographics Association, pp. 61–70.

[KBLB12] Kokkinos I., Bronstein M. M., Litman R., Bronstein A. M.: Intrinsic shape context descriptors for deformable shapes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), IEEE, pp. 159–166.

[KLF11] Kim V., Lipman Y., Funkhouser T.: Blended intrinsic maps. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 30*, 4 (July 2011).

[KNP07] Kälberer F., Nieser M., Polthier K.: Quadcover-surface parameterization using branched coverings. In *Computer Graphics Forum, Proceedings Eurographics* (2007), vol. 26, Wiley Online Library, pp. 375–384.

[Knu00] Knupp P. M.: Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities.

International Journal for numerical methods in engineering 48, 8 (2000), 1165–1185.

[KPNK03] Körtgen M., Park G.-J., Novotni M., Klein R.: 3d shape matching with 3d shape contexts. In *The 7th central European seminar on computer graphics* (2003), vol. 3, pp. 5–17.

[KR93] Kass R. E., Raftery A. E.: *Bayes Factors and Model Uncertainty*. Tech. Rep. 571, Carnegie Mellon University Dept of Statistics, Pittsburgh, PA 15213, 1993.

[KR95] Kass R. E., Raftery A. E.: Bayes factors. *Journal of the american statistical association 90*, 430 (1995), 773–795.

[KST08] Kolomenkin M., Shimshoni I., Tal A.: Demarcating curves for shape illustration. *ACM Transactions on Graphics (TOG) 27*, 5 (2008).

[KST09] Kolomenkin M., Shimshoni I., Tal A.: On edge detection on surfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), IEEE, pp. 2767–2774.

[LB14] Litman R., Bronstein A. M.: Learning spectral descriptors for deformable shape correspondence. *IEEE transactions on pattern analysis and machine intelligence 36*, 1 (2014), 171–180.

[LDCK18] Lim I., Dielen A., Campen M., Kobbelt L.: A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *European Conference on Computer Vision (ECCV) 2018 Workshops* (2018), Springer International Publishing.

[Lie03] Liepa P.: Filling holes in meshes. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2003), Eurographics Association, pp. 200–205.

[LL02] Lee Y., Lee S.: Geometric snakes for triangular meshes. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 229–238.

[Low99] Lowe D. G.: Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV)* (1999), vol. 2, IEEE, pp. 1150–1157.

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. In *ACM Transactions on Graphics (TOG)* (2002), vol. 21, ACM, pp. 362–371.

[LRR*17] Litany O., Remez T., Rodola E., Bronstein A. M., Bronstein M. M.: Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017), vol. 2, p. 8.

[LZL16] Liu Z., Zhang J., Liu L.: Upright orientation of 3d shapes with convolutional networks. *Graphical Models 85* (2016).

[MBBV15] Masci J., Boscaini D., Bronstein M., Vandergheynst P.: Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2015), pp. 37–45.

[MBLS18] Mena G., Belanger D., Linderman S., Snoek J.: Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665* (2018).

[MBM*17] Monti F., Boscaini D., Masci J., Rodola E., Svoboda J., Bronstein M. M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), vol. 1, p. 3.

[MGP06] Mitra N. J., Guibas L. J., Pauly M.: Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 560–568.

[Mod] Modelnet. http://modelnet.cs.princeton.edu/.

[MOR*18] Melzi S., Ovsjanikov M., Roffo G., Cristani M., Castellani U.: Discrete time evolution process descriptor for shape analysis and matching. *ACM Transactions on Graphics (TOG) 37*, 1 (2018), 4.

[MPWC13] Mitra N. J., Pauly M., Wand M., Ceylan D.: Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum 32*, 6 (2013), 1–23.

[MZL*09] Mehra R., Zhou Q., Long J., Sheffer A., Gooch A., Mitra N. J.: Abstraction of man-made shapes. *ACM Transactions on Graphics (TOG) 28*, 5 (2009).

[NIH*11] Newcombe R. A., Izadi S., Hilliges O., Molyneaux D., Kim D., Davison A. J., Kohi P., Shotton J., Hodges S., Fitzgibbon A.: Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on* (2011), IEEE, pp. 127–136.

[NLNZ16] Ni H., Lin X., Ning X., Zhang J.: Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods. *Remote Sensing 8*, 9 (2016), 710.

[NMR*18] Nogneng D., Melzi S., Rodolà E., Castellani U., Bronstein M., Ovsjanikov M.: Improved functional mappings via product preservation. *Computer Graphics Forum 37* (2018).

[NO17] Nogneng D., Ovsjanikov M.: Informative descriptor preservation via commutativity for shape matching. *Computer Graphics Forum 36*, 2 (2017), 259–267.

[NPPZ12] Nieser M., Palacios J., Polthier K., Zhang E.: Hexagonal global parameterization of arbitrary surfaces. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (2012), 865–878.

[NSP10] Nieser M., Schulz C., Polthier K.: Patch layout from feature graphs. *Computer Aided Design 42*, 3 (2010), 213–220.

[OBCS*12] Ovsjanikov M., Ben-Chen M., Solomon J., Butscher A., Guibas L.: Functional maps: A flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–11.

[OBS03] Ohtake Y., Belyaev A., Seidel H.-P.: A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *Shape Modeling International, 2003* (2003), IEEE, pp. 153–161.

[OBS04] Ohtake Y., Belyaev A., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics (TOG) 23*, 3 (Aug. 2004), 609–612.

[OMMG10] Ovsjanikov M., Mérigot Q., Mémoli F., Guibas L.: One point isometric matching with the heat kernel. *Computer Graphics Forum 29*, 5 (2010), 1555–1564.

[PBB*13] Pokrass J., Bronstein A. M., Bronstein M. M., Sprechmann P., Sapiro G.: Sparse modeling of intrinsic correspondences. *Computer Graphics Forum 32* (2013), 459–468.

[PKG03] Pauly M., Keiser R., Gross M.: Multi-scale feature extraction on point-sampled surfaces. In *Computer Graphics Forum* (2003), vol. 22, Wiley Online Library, pp. 281–289.

[PMW*08] Pauly M., Mitra N. J., Wallner J., Pottmann H., Guibas L. J.: Discovering structural regularity in 3d geometry. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 43.

[RBG*09] Reuter M., Biasotti S., Giorgi D., Patanè G., Spagnuolo M.: Discrete laplace–beltrami operators for shape analysis and segmentation. *Computers & Graphics 33*, 3 (2009), 381–390.

[RCB*17] Rodolà E., Cosmo L., Bronstein M. M., Torsello A., Cremers D.: Partial functional correspondence. *Computer Graphics Forum 36*, 1 (2017), 222–236.

[Ris83] Rissanen J.: A universal prior for integers and estimation by minimum description length. *Annual Statistics 11*, 2 (1983), 416–431.

[RM05] Rokach L., Maimon O.: Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[RMC17] Rodolà E., Möller M., Cremers D.: Regularized pointwise map recovery from functional correspondence. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 700–711.

[ROA*13] Rustamov R. M., Ovsjanikov M., Azencot O., Ben-Chen M., Chazal F., Guibas L.: Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 72.

[RPC10] Rabin J., Peyré G., Cohen L. D.: Geodesic shape retrieval via optimal mass transport. In *European Conference on Computer Vision (ECCV) 2010* (2010), Springer Berlin Heidelberg, pp. 771–784.

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission* (Sept. 2004).

[SAD*16] Shi Z., Alliez P., Desbrun M., Bao H., Huang J.: Symmetry and orbit detection via lie-algebra voting. In *Computer Graphics Forum* (2016), vol. 35, pp. 217–227.

[Sha] Shapenet. https://www.shapenet.org/.

[SJW*11] Sunkel M., Jansen S., Wand M., Eisemann E., Seidel H.-P.: Learning line features in 3d geometry. In *Computer Graphics Forum* (2011), vol. 30, pp. 267–276.

[SOG09] Sun J., Ovsjanikov M., Guibas L.: A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2009), Eurographics Association, pp. 1383–1392.

[Sol15] Solomon J.: *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. CRC Press, 2015.

[SRGB14] Solomon J., Rustamov R., Guibas L., Butscher A.: Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG) 33*, 4 (July 2014).

[SSCO08] Shapira L., Shamir A., Cohen-Or D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249.

[SSK*05] Surazhsky V., Surazhsky T., Kirsanov D., Gortler S. J., Hoppe H.: Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics (TOG) 24*, 3 (July 2005), 553–560.

[STJ*17] Schertler N., Tarini M., Jakob W., Kazhdan M., Gumhold S., Panozzo D.: Field-aligned online surface reconstruction. *ACM Transactions on Graphics (Proceedings SIGGRAPH) 36*, 4 (2017).

[SvKK*11] Sidi O., van Kaick O., Kleiman Y., Zhang H., Cohen-Or D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia) 30*, 6 (2011), 126:1–126:10.

[SWPL08] Sahner J., Weber B., Prohaska S., Lamecker H.: Extraction of feature lines on surface meshes based on discrete morse theory. In

Computer Graphics Forum (2008), vol. 27, Wiley Online Library, pp. 735–742.

[Tau95] Taubin G.: A signal processing approach to fair surface design. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), ACM, pp. 351–358.

[TBW*11] Tevs A., Berner A., Wand M., Ihrke I., Seidel H.-P.: Intrinsic shape matching by planned landmark sampling. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 543–552.

[Thia] Thingi10k. https://ten-thousand-models.appspot.com/.

[Thib] Thingiverse. https://www.thingiverse.com/.

[Tri] Trimble 3d warehouse. https://3dwarehouse.sketchup.com/.

[TSDS10] Tombari F., Salti S., Di Stefano L.: Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)* (2010), Springer, pp. 356–369.

[Tur] Turbosquid. https://www.turbosquid.com/.

[VL08] Vallet B., Lévy B.: Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 251–260.

[VSHJ01] Vergeest J. S., Spanjaard S., Horváth I., Jelier J. J.: Fitting freeform shape patterns to scanned 3d objects. *Journal of Computing and Information Science in Engineering 1*, 3 (2001), 218–224.

[VtH07] Veltkamp R., ter Haar F.: Shrec 2007 3d shape retrieval contest.

[WB01] Watanabe K., Belyaev A. G.: Detection of salient curvature features on polygonal surfaces. In *Computer Graphics Forum* (2001), vol. 20, pp. 385–392.

[WG09] Weinkauf T., Günther D.: Separatrix persistence: Extraction of salient edges on surfaces using topological methods. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2009), Eurographics Association, pp. 1519–1528.

[WGBS18] Wang L., Gehre A., Bronstein M. M., Solomon J.: Kernel functional maps. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 27–36.

[WK05] Wu J., Kobbelt L.: Structure recovery via hybrid variational surface approximation. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 277–284.

[WMKG07] Wardetzky M., Mathur S., Kaelberer F., Grinspun E.: Discrete Laplace operators: No free lunch. In *Computer Graphics Forum, Proceedings Eurographics Symposium on Geometry Processing* (2007), The Eurographics Association.

[WVR*14] Windheuser T., Vestner M., Rodolà E., Triebel R., Cremers D.: Optimal intrinsic descriptors for non-rigid shape analysis. In *Proceedings of the British Machine Vision Conference* (2014).

[XKH*16] Xu K., Kim V. G., Huang Q., Mitra N., Kalogerakis E.: Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses* (2016), ACM, p. 4.

[YBS05] Yoshizawa S., Belyaev A., Seidel H.-P.: Fast and robust detection of crest lines on meshes. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling* (2005), ACM, pp. 227–232.

[YNBT01] Yacoub M., Niang N., Badran F., Thiria Ë.: A new hierarchical clustering method using topological map. *Proceedings 10th International Symposium on Applied Stochastic Models and Data Analysis* (2001).

[ZHX*11] Zhang L., He Y., Xia J., Xie X., Chen W.: Real-time shape illustration using laplacian lines. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 993–1006.

[ZZCJ14] Zhuang Y., Zou M., Carr N., Ju T.: Anisotropic geodesics for live-wire mesh segmentation. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 111–120.

3D Model Sources

The 3D models used in this thesis to demonstrate our techniques and illustrate our findings are courtesy of, the Shape COSEG data set [SvKK*11], the ShapeNet database at at Princeton University and Stanford University, the Max-Planck-Gesellschaft zur Förderung der Wissenschaften, the Institut für Kartographie und Geoinformatik (Leibniz University Hannover), the AIM@SHAPE repository, the Stanford 3D Scanning Repository, the Image-based 3D Models Archive, Télécom Paris, the TOSCA [BBK08] and SHREC [VtH07] data sets, and the Lincoln 3D Scans Repository.

Publications

Anne Gehre, Isaak Lim, Leif Kobbelt: Feature Curve Co-Completion in Noisy Data. In *Computer Graphics Forum, Proc. Eurographics*, 2018

Anne Gehre, Michael Bronstein, Leif Kobbelt, Justin Solomon: Interactive Curve Constrained Functional Maps. In *Computer Graphics Forum, Proc. Eurographics Symposium on Geometry Processing*, 2018

Larry Wang, Anne Gehre, Michael Bronstein, Justin Solomon: Kernel Functional Maps. In *Computer Graphics Forum, Proc. Eurographics*, 2018

Anne Gehre, Isaak Lim, Leif Kobbelt: Adapting Feature Curve Networks to a Prescribed Scale. In *Computer Graphics Forum, Proc. Eurographics*, 2016

Isaak Lim, Anne Gehre, Leif Kobbelt: Identifying Style of 3D Shapes using Deep Metric Learning. In *Computer Graphics Forum, Proc. Eurographics Symposium on Geometry Processing*, 2016

Anne Gehre, David Bommes, Leif Kobbelt: Geodesic Iso-Curve Signature. In *Proc. 21st International Symposium on Vision, Modeling and Visualization*, 2016

Statement of Originality

Many of the ideas that lead to the realization of the algorithms and results shown in this thesis were influenced by the inspiring discussions with Prof. Dr. Leif Kobbelt as well as members of the *Visual Computing Institute (RWTH Aachen)*. Also, the discussions with our collaborators Prof. Dr. David Bommes, Prof. Dr. Justin Solomon and Prof. Dr. Michael Bronstein as well as the members of the *Geometric Data Processing Group (Massachusetts Institute of Technology)* had impact on the developed methods. Below, my contributions to the articles relevant to this thesis are listed.

[GLK16] As main author I conceived and wrote most parts of this paper . I implemented the software modules (with small contributions from Isaak Lim), which lead to the results of this article. Furthermore, I conducted all experiments and the evaluation listed in the paper and the supplemental material.

[GBK16] As main author I conceived and wrote most parts of this paper. I implemented the software modules (with contributions from David Bommes), which lead to the results of this article. Furthermore, I conducted all experiments and the evaluation listed in the paper.

[GLK18] As main author I conceived and wrote most parts of this paper . I implemented the software modules (with small contributions from Isaak Lim), which lead to the results of this article. Furthermore, I conducted all experiments and the evaluation listed in the paper.

[GBKS18] As main author I conceived and wrote most parts of this paper. I implemented the software modules, which lead to the results of this article. Furthermore, I conducted all experiments and the evaluation listed in the paper.