RWTH AACHEN UNIVERSITY

The present work was submitted to the

Visual Computing Institute
Faculty of Mathematics, Computer Science and Natural Sciences
RWTH Aachen University

Retrieval and Re-Embedding of Macro Constraints for Parametrization Based Quad Meshing

Master's Thesis

presented by

Patrick Schmidt
Student ID Number 308691

March 2017

First Examiner: Prof. Dr. Leif Kobbelt
Second Examiner: Prof. Dr. David Bommes

Abstract

Recent developments in parametrization based quad meshing allow for interactive constraint editing sessions. Automatic suggestion of constraint templates for local surface regions is expected to further assist a high-level design process.

This thesis sketches a system that recommends such constraint sets, called *macro constraints*, based on a template library. In particular, it focuses on two sub-problems: retrieving a suitable template from the database and then re-embedding it on the target surface.

While the first task can be solved by a standard approach, the second one involves the difficult problem of partial inter-surface mappings. For the latter, a descriptor guided approach using non-linear optimization is proposed.

Hiermit versichere ich, diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht zu haben.

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Aachen, March 30, 2017

Patrick Schmidt

Contents

Chapter 1

Introduction

Parametrization based quad meshing methods recently became robust and fast enough for interactive workflows, cf. [CBK15], [ESCK16]. Users are now able to explore the design space by iteratively adding and manipulating constraints such as prescribed directions, sizing and singularity configurations.

A major motivation for interactive approaches is that a user's design intent often deviates strongly from formalized quality criteria like uniformity, regularity, isotropy and alignment to principal curvature directions. Instead, it may incorporate artistic expression as well as expert knowledge for which explicit formulations are not available.

However, typically chosen constraint sets reveal common patterns within a class of shapes, and interactive quad meshing sessions still contain highly repetitive actions. Often, information about possible design choices is already contained in previous sessions on different shapes of the same class.

To further assist the design process on a higher, more creative level, it is conceivable to use data-driven approaches which infer constraint sets for parts of 3D models based on previous sessions on similar parts. The resulting constraint sets could potentially be superior to those obtained by traditional methods as they are based on information going beyond surface properties of a single isolated shape and incorporate knowledge about prevalent design intents.

In this thesis, a potential system for this task is considered and two subproblems are examined in detail. This system is centered around a database containing triangle mesh patches, cut out from larger meshes. Each patch is equipped with a set of constraints, which was created by a user in a preceding session. We call such a set of constraints, defined on a local surface patch, a *macro constraint*. In an ongoing session on a new triangle mesh, unknown to the system, the user can specify a region of interest and query the database for constraint suggestions. After that, a patch which is similar in shape to the target region will be identified by the system and mapped to the target surface. Constraints attached to the queried

patch are then transferred to the target where they are used as an input to existing quad meshing methods.

1.1 Quad Meshes in Character Animation

Discrete surface meshes are the representation of choice for three-dimensional objects in a majority of applications. Despite the various advantages of triangle meshes, some domains prefer meshes consisting entirely of quadrilateral elements. For example, quad meshes are commonly used as control meshes for parametric spline surfaces like NURBS, see [Bom12]. Moreover, they are particularly well suited for subdivision surfaces, as they maintain a consistent edge flow under regular refinement, cf. [BLP+13]. Especially for use in character animation, this property is of great importance as it influences the appearance of the resulting surface as well as its behavior under deformation.

Quality Criteria

Typical quality criteria aimed for in the literature include, but are not limited to, the following: (1) *Uniformity* aims at equally sized quads or equal edge lengths in the entire mesh. (2) *Isotropy* describes quads extending equally along both directions and thus being close to a square. (3) *Regularity* is present in a region of the mesh if it contains no singularities, i.e. no vertices with valence other than four. (4) *Alignment* to the underlying geometry is, for example, achieved if the edge flow of a quad mesh follows the principal curvature directions of the object. (5) *Preservation* of sharp features, e.g. edge paths of high dihedral angle, is also an important quad mesh property. Common quality criteria are discussed more thoroughly in e.g. [BZK09] and [BLP+13].

However, these criteria are not independent from each other. Most prominently, regularity and alignment to given directions can be conflicting concepts. While, depending on the genus of the object, pure regularity is topologically impossible, a larger number of singularities can give more flexibility in the alignment of the edge flow. Thus, regularity and alignment often present a trade-off, cf. [MZ13]. Furthermore, snapping edge paths of the quad mesh to given feature lines typically comes at the cost of decreased isotropy, which is gladly accepted in e.g. [BZK09].

Quad meshes created manually by artists however reveal a different set of properties. For example, quad meshes used in character animation do not only have to provide a precise surface representation under subdivision, but also have to maintain good quality under deformation. If the type of deformation is not

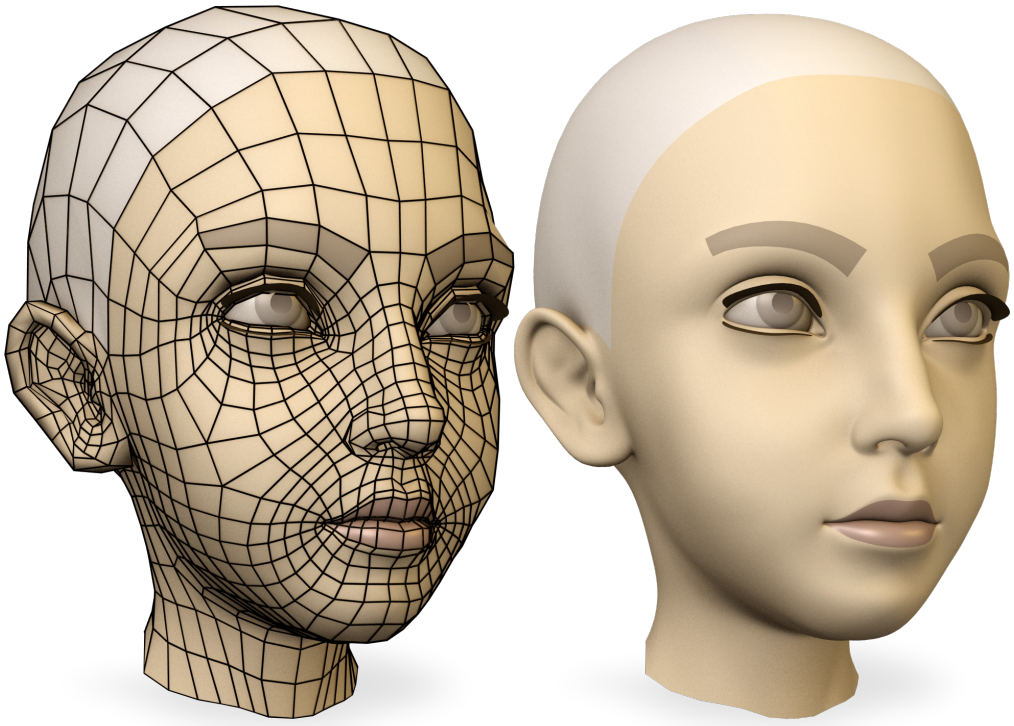


Figure 1.1: (left) The quad mesh was created manually for the film “Sintel” as part of the Blender open movie project. (right) The same mesh is shown under two steps of Catmull-Clark subdivision. © Copyright Blender Foundation — `www.sintel.org`.

known a priori, the necessary quality criteria are difficult to formalize. Therefore in practice, high-quality quad meshes are preferably created by skilled artists.

As a representative of the resulting class of meshes, Figure 1.1 gives a number of useful insights. First of all, large differences in element size can be observed, when e.g. comparing quads around the mouth to quads on the forehead. This can be easily explained by the amount of expected deformation in the respective areas, since a high mesh resolution provides more flexibility in handling various kinds of distortion. Additionally, in a subdivision setting, finely tessellated regions allow modeling sharper details, while coarse regions are heavily smoothed.

Moreover, the isotropy criterion is clearly violated in this example. As can be seen at the ear of the model, this allows accurate surface modeling while still maintaining a relatively low number of quads. In Addition, anisotropy is used to locally increase the resolution without adding additional singularities, as observed in the area around the eyes.

Accordingly, regularity still seems to be of high importance, since the mesh shows only few irregular vertices. In contrast to e.g. [BZK09], these are not primarily placed in regions of high Gaussian curvature, but far away from areas of high anticipated deformation.

Furthermore, artists strive to align the edge flow of a mesh to muscular structures representing the direction of expect deformation. Often, these coincide with principal curvature directions, however not always in a strict sense.

1.2 Interactive Constraint Editing

The above observations motivate closing the gap between purely manual and entirely automatic quad remeshing methods. A promising approach in this directions is presented by interactive constraint editing sessions, built on top of parametrization based quad meshing algorithms. In these sessions, users are initially provided with an automatically generated quad mesh. In the following, they can iteratively add or manipulate various types of constraints, while the resulting quad mesh is updated after each interaction, see [ESCK16].

Parametrization Based Quad Meshing

The family of parametrization based methods is particularly well suited for this purpose, due to numerous possible ways to constrain them. Such methods, e.g. [BZK09], take a given triangle mesh as an input and start by cutting it open until it is topologically equivalent to a disk. Then, the mesh is mapped to the plane, i.e. parametrized, by assigning a pair of (u, v) coordinates to each vertex. In the resulting 2D embedding, the flattened mesh is covered by a regular grid. Finally, mapping this grid back to the original 3D mesh, provides a quad pattern on the surface. To assure consistency of this pattern across cuts, certain conditions have to be imposed upon the parametrization. In [BCE+13], these conditions are formalized and the term *integer-grid maps* is coined for the class of parametrization functions adhering to them.

In the following, we list a selection of constraints which can be used within the framework of parametrization based methods. In particular, this list focuses on manual constraints that can be directly controlled by user input.

Directional Constraints

As an alternative to directly aligning the edge flow of a quad mesh to the principal curvature directions of a shape, guiding constraints can be manually employed by a user. In their simplest form, these constraints are defined by polygonal lines,

drawn on the input surface. The direction of these lines locally translates into guiding directions for the resulting edge flow. An important aspect is that edges of the eventual quad mesh do not have to be located on an input line, but merely follow its direction. In [ESCK16] the effect of such input lines is extended to entire regions around the line. The size of these regions provides the user with another control parameter for the effect of a constraint.

Feature Constraints

A similar concept is presented by feature constraints, as these are also defined by lines embedded in the input surface. In addition to prescribing directions, this constraint type also forces a sequence of connected vertices in the resulting mesh to lie exactly on the input line. While directional guides are often referred to as soft constraints, feature lines are also called hard constraints. On the one hand these can be used to preserve sharp features of an object. On the other hand, hard constraints provide an effective way for users to very directly manipulate the resulting quad mesh. In the extreme case, these constraints can be used to completely prescribe a fixed quad pattern in a region of the surface.

Singularity Constraints

Although automatic methods strive to place irregular vertices in geometrically meaningful locations, these might not reflect the user's design intent, as seen in the example above. Parametrization methods therefore allow manual placement of singularities as long as certain topological requirements are fulfilled, cf. [Bom12] and [PZKW11]. Consequently, a user is allowed to re-locate existing singularities, to add pairs of new singularities or to merge existing ones. Moreover, it is possible to either entirely hand over the control of the singularity structure to the user, or to just prescribe some singularities and let the algorithm choose the others.

Connection Constraints

As another way to gain control over the topology of the quad mesh, connection constraints can be utilized. Such constraints are employed between pairs of singularities or other feature vertices and ensure that these are directly connected by an edge path in the resulting quad mesh. Similarly, loop constraints can be used around cylindrical surface parts, assuring that the edge flow forms closed loops around them. This is an especially important tool to prevent helix structures in cylindrical regions.

Sizing & Anisotropy Constraints

The example in Figure 1.1 demonstrates that adaptive or non-uniform sizing is often a desired property. Desired sizing information can be expressed by a scalar sizing field or even an anisotropy field defined on the input surface. Alternatively, some approaches allow sketching single quads of certain size and anisotropy in different surface regions and automatically infer a corresponding field. While prominent parametrization methods can be extended to take such a field as an input, a limiting factor is presented by their impact on the singularity structure. As effective transitions between areas of large quads and areas of small quads require special singularity configurations, realizing sizing constraints is an intricate problem on its own. Methods successfully taking such constraints as an input are e.g. [PPTS14] and [JFH+15].

1.3 Macro Constraints

The system sketched in this thesis is based on the notion of *macro constraints*. We define a macro constraint to be a set of constraints defined on a local surface patch. This set may include all constraint types defined in the previous section. In fact, we design our method to be agnostic to the exact nature of these constraints.

A large number of macro constraints, collected in a database, builds the basis of the recommendation system. In its final version, this database should be filled with numerous macro constraints, which are obtained from interactive quad meshing sessions performed by skilled artists. Within the scope of this thesis, a small manually filled test database suffices, as we primarily focus on two technical sub problems of the system.

Further, we employ a restriction on the nature of the supporting surface patch. To simplify the setting, all patches are assumed to be simply connected, i.e. topologically equivalent to a disk, having a single boundary. Moreover, we choose the patches in our test database to be geodesic disks. This means that starting from a central point on the patch, it exhibits equal geodesic distance to all boundary points.

In a first version of the system, we picture the following user interaction: during an interactive constraint editing session, the user starts by also selecting such a geodesic disc on the target object. This can be done by choosing a point on the surface as well as a geodesic radius. In the following retrieval step, the database is searched for similar surface patches. For example, a fixed number of suitable matches can be shown to the user. After choosing one of the suggested patches, a map from this source patch to the selected part of the target surface is computed. Finally, all constraints are transferred form the source patch to the target. We also

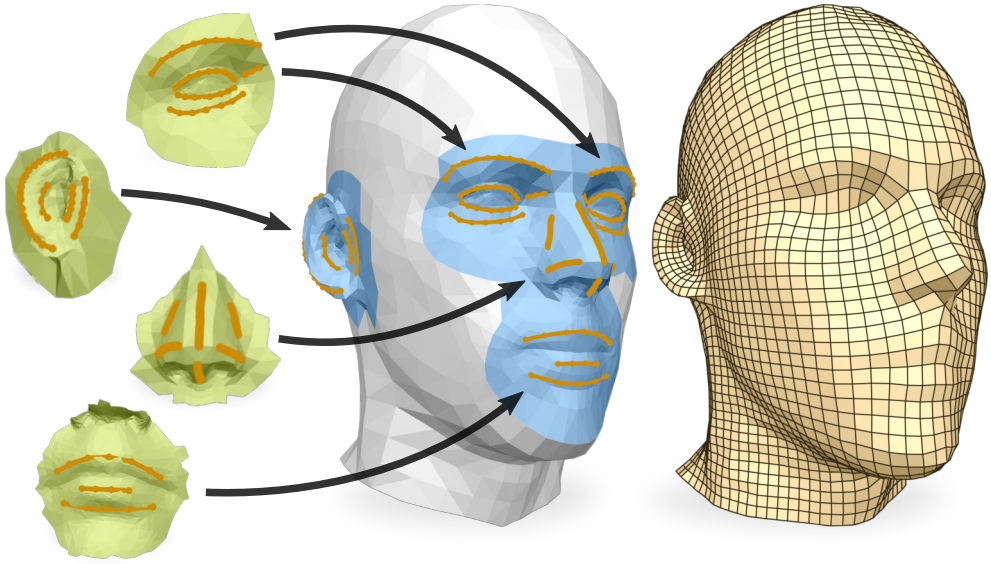


Figure 1.2: The envisioned constraint recommendation system is illustrated. On the target mesh (center), five regions are subsequently selected by a user (blue). Each one is equipped with direction guides by re-embedding a macro constraint (left). The macro constraint used for the eye is employed twice, giving an approximately symmetric result. The resulting quad mesh is locally aligned to the transferred direction constraints (right).

say they are *re-embedded* on the target surface. There, they are used as an input to a standard quad meshing algorithm.

To achieve independence from the actual types of constraints, both the retrieval and the re-embedding phase do not consider the constraints attached to the source patch. In particular, both steps are based entirely on the geometry of the supporting patch.

Figure 1.2 depicts a use case of the system. Here, multiple regions on the target mesh are selected and each one is equipped with a macro constraint from the database. Based on the re-embedded directional constraints, an aligned quad mesh is generated.

This thesis is structured as follows: we start by a review of related work in Chapter 2. In Chapter 3, a bag-of-features approach is tailored to the retrieval task, followed by an evaluation of the results on a small dataset. The main contribution of this thesis is presented in Chapter 4, where a method to compute partial inter-surface mappings between two patches is proposed. In the following evaluation, successful examples are shown and the remaining issues of the approach are discussed in detail. Finally, Chapter 5 gives an outlook to possible future work.

Chapter 2

Related Work

This chapter gives an overview of existing publications in the fields of parametrization based quad meshing as well as shape descriptors, shape retrieval and shape correspondences.

Parametrization Based Quad Meshing

The general approach employed in modern parametrization based quad meshing algorithms is introduced in both [KNP07] and [BZK09]. The former takes a guiding cross field as an input and solves the mixed-integer problem, arising in the parametrization step, by directly rounding variables of a continuous solution to close-by integer values. The latter computes both, a smooth cross field and a global parametrization using an iterative greedy rounding scheme, alternating between solving a continuous problem and rounding a subset of integer variables. In [BCE+13] a robust method is proposed. First, sufficient conditions for parametrizations to imply a quad mesh are formally introduced using the term integer-grid maps. After that, it is shown how to compute an exact solution of a reduced problem using a general purpose solver. However, the linearization of local injectivity constraints excludes valid parts of the solution space. Finally, [CBK15] proposes a new greedy strategy which both guarantees valid solutions and is significantly faster than its predecessor. A robust method to extract valid quad topology from a parametrization, despite some violations of the integer-grid map conditions, is introduced in [EBCK13].

Optimizing for both, high quality parametrizations and a suitable placement of singularities at the same time proved to be a challenging problem. Therefore, the process is commonly split up into two steps. First, a guiding frame field or cross field is generated which already defines the position and degree of singularities. In the following step, these singularities stay fixed, and a parametrization is computed such that its iso-lines follow the guiding field. Extending the approach of

[BZK09], [ECBK14] generates noise-robust cross fields at a specified level of detail by a controlled smoothing of normal fields. The authors of [PPTS14] produce anisotropic and non-orthogonal frame fields by computing a cross field on a deformed input surface. Similarly, [JFH+15] compute such fields via cross fields in a customized metric. In [DVPS14], even more general n-vector fields are computed and the commonly used mixed-integer formulation is avoided by encoding the variables in complex polynomials. On top of this formulation, [DVPS15] makes an attempt to create integrable frame fields, which constitutes a step towards reuniting both the field generation and the parametrization step. A comprehensive survey on state-of-the-art direction field computation is given in [VCD+16].

Interactive Quadrangulation

Some publications target the automatic generation of particular constraint types: [CIE+16] finds connected regions of homogeneous curvature directions, while [GLK16] detects feature curves at a specified scale. In adition to automatically generated constraints, the methods listed in the previous section can be constrained by user input to various degrees.

Other publications explicitly define user metaphors: e.g. in the context of quad layouts, [CK14] proposes an input method method in which cyclic strips are used to completely define a layout.

To achieve run times fast enough for interactive workflows, the authors of [ESCK16] introduce a hierarchical approach to speed up an entire class of parametrization based methods.

A different path is taken in [TPSS13] and [TPS14]. Here, a sketch based user interface is created, assisting manual workflows by completing various actions. In [MTP+15], a database of quad topologies is used to automatically fill user-defined patches. This data-driven approach follows a similar motivation to the system described in this thesis. However, (1) the data employed in [MTP+15] is of purely topological nature and (2) the resulting quad meshes have no means of being globally optimal. In contrast, our approach allows encoding various combinations of geometrical and topological constraints in the database. Furthermore, by using constraints instead of actual quad patterns as a currency, we can rely on external methods to compute or approximate a globally optimal solution.

Shape Descriptors

The concept of shape descriptors finds its roots in computer vision, where image descriptors are successfully applied in various tasks. While in computer vision the SIFT descriptor [Low04] emerged as a de facto standard, a well-performing general purpose shape descriptor is yet to be found.

The survey [XKHK17] lists numerous shape descriptors and divides them into the categories of global (representing an entire shape) and local (representing a single point) descriptors. In the following, descriptors considered in the context of this work are described.

Following the concept of the histogram of oriented gradients descriptor in computer vision, [LWWS15] proposes the histogram of oriented curvatures. For a single point, this descriptor employs a tangential grid. In a neighborhood around the point, the maximum curvature directions are projected into the grid and an orientation histogram is created for each cell. Due to the discrete cell assignment as well as discrete binning, the distance fields emerging from this descriptor are far from smooth and thus not suitable for our method in Chapter 4. Furthermore, the local grid has to be aligned to the maximum curvature direction at each point, which can be unstable or unavailable in flat surface regions.

Two prominent representatives from the family of spectral descriptors are the heat kernel signature [SOG09] and the wave kernel signature [ASC11]. Based on the eigenfunctions of the Laplace-Beltrami operator, these descriptors consider shapes in a frequency spectrum. Using this representation, they are intrinsic, i.e. invariant to isometric deformations. Both methods borrow different concepts from physics, resulting in different behavior with respect to high-frequency details. In [BBC+10] the heat kernel signature is extended to a scale invariant descriptor. The author of [Bro11] shows how to obtain a spectral descriptor, customized to a given database of shapes, by learning techniques. In the setting of local surface patches with boundaries, spectral descriptors lose a large part of their discriminative power due to the lack of global information.

The authors of [KBLB12] propose a meta descriptor, which takes an existing descriptor field as an input and enhances it by including neighborhood information in each point. Specifically, descriptor values are softly assigned to the cells of a polar grid around each point. To achieve rotation invariance, the arbitrary rotational offset, present in each instance of the grid, is removed using a Fourier transform.

Shape Retrieval

Chapter 3 is mainly based on the bag-of-features approach proposed in [OBBG09]. For evaluation, the authors use the popular TOSCA dataset [BBK08] providing shapes categorized in different classes. To test the invariance with respect to certain transformations, the set includes isometric deformations, topological differences and missing parts in each class.

In the context of the 3D Shape Retrieval Contest (SHREC), an overview of recent developments in shape retrieval and matching is published each year. In particular, the presented methods are thoroughly evaluated using unified bench-

marks. Notable examples include [LGB+11], [LZC+15] and [RCL+17]. The former two evaluate methods with respect to non-rigid deformations: [LGB+11] introduces and [LZC+15] extends a test database with models assigned to different classes. Each class exhibits variance in form of pose variations, artificially created by deforming a skeleton representation. While these two benchmarks are based on watertight meshes, [RCL+17] considers meshes with missing parts and topological noise. Both cases, i.e. retrieving complete meshes using a partial query and finding partial results matching a complete query, are evaluated. This situation is similar to our setting, as we consider parts of objects with varying degrees of common geometry.

Moreover, [SSSC08] partitions objects and matches corresponding parts both within a single shape and between shapes. Also relevant within our context is [LWWS15], where similar parts of a single shape are detected and transformations between them are computed.

Shape Correspondences and Mappings

The problem of defining meaningful correspondences between points of two (or more) surfaces is subject to ongoing research and has been approached from multiple perspectives. As no general and effective solution to the problem is available, numerous methods have been published, following different solution paradigms. Methods differ in the exact formulation of the problem, in the types of input provided, and in the types of invariance they support. The survey [KZHC11] gives an extensive overview over existing literature.

One particular family of methods emerged as an extension of registration methods like the iterative closest point (icp) algorithm [RL01]. Such non-rigid registration algorithms alternate between deforming one of the shapes and updating its correspondences to the other. A demonstrative example of this class is the soft-icp algorithm presented in [SBSC06]. This method continuously interpolates between a rigid icp-like alignment and a soft deformation. In [HAWG08] a robust algorithm following a similar approach is proposed, which can handle missing parts as well as isometric deformations and preserves geodesic distances.

Another popular class of methods is based on pointwise shape descriptors. Some publications, presenting novel descriptors, also propose simple matching algorithms, such as [ASC11]. Here, a greedy strategy is described, which first generates a large number of candidate correspondences and then picks those that best preserve geodesic distances between both shapes. The authors of [OMMG10] use the heat kernel signature to recover isometries between shapes by taking a single point-to-point correspondence as an input. [DLL+10] also uses the heat kernel signature to partition each object and then establishes a part-to-part matching, specifically targeting incomplete models.

Further, probabilistic models also yield effective solution methods. Models such as Markov random fields allow formulating a discrete assignment problem by defining single potentials (favoring geometric similarity) and pairwise potentials (favoring isometry). An illustrative example of such a method is described in [ASP+04]. [TB11] follows a similar approach, but operates on a subset of vertex pairs. Both solve the resulting problem using loopy belief propagation.

The authors of [SNB+12] abandon the concept of discrete point-to-point correspondences and represent soft maps using probability martrices. Their descriptor based method can cope with non-isometric deformations and some issues related to the inherent ambiguity of such mappings are avoided. In [OBS+12], the authors also generalize the concept of explicit point-to-point correspondences by establishing a map between the eigenfunctions of the Laplace-Beltrami operator. Despite the indirect formulation, these maps can still be used to transfer attributes between isometric objects. [RCB+15] analyses the behavior of these eigenfunctions when removing parts of a shape and proposes an extended method. [LRBB17] extends the concept even further, working entirely within a spectral domain. Except for an initialization step, this method is independent of the mesh complexity.

A series of publications addresses the problem of maps between surfaces in a parametrization setting. However, all of them use a sparse set of user-defined correspondences as an input. In [APL14], both meshes are cut to topological disks and are then parametrized minimizing isometric distortion. It is shown how to establish a mapping from these parametrizations using the property of local but not global injectivity. In addition, [APL15] proposes alternative parametrizations, rendering the method invariant to the nature of the cut graphs. Finally, both [AL15] and [AL16] introduce a novel extension of Tutte embeddings, also enabling maps between multiple shapes, given a set of initial correspondences.

The term inter-surface maps, as used in this thesis, is coined in [SAPH04]. There, a hierarchical approach is used to compute distortion minimizing mappings, however without explicitly using an intermediate parametrization domain.

Parts of our deformation method in Chapter 4 show parallels to [SS15]. The authors also start with a Tutte embedding of a surface chart and then deform it by non-linear optimization. They minimize a distortion measure based on the singular values of the Jacobian and use barrier functions to ensure local as well as global injectivity of the parametrization. However, their setting is somewhat simpler than ours, because the authors only consider a single parametrization instead of an inter-surface map.

Chapter 3

Patch Retrieval

This chapter describes how to find an appropriate retrieval method for the system sketched above. Such a method should be able to efficiently query an existing database for a suitable macro constraint, based on its supporting geometry.

This database is filled with surface patches, each one equipped with a set of constraints. For the scope of this thesis, each patch is restricted to be a geodesic disc, manually cut out of a larger model. We normalize patches such that their geodesic radius is 1. All patches and models, also called shapes, are considered to be triangle meshes.

To simplify the setting, we require queries to the database to be of the same type as its data, i.e. geodesic surface patches. This means that a user, who wishes to add constraints to a certain part of a model, has to select a target region on that model. Since this region also has to be a geodesic disc, the selection can be performed by choosing a point and a radius on the target mesh. The resulting region is then cut out of the model and referred to as the target patch. This patch can now be compared against all items in the database.

To permit quick response times, the comparison has to be efficient. For example it is not feasible to try and compute a mapping from each source patch to the target patch and pick the best one. Similarly, methods computing a deformation energy between patches cannot be used. More specifically, we are looking for a method in which the cost of a query does not depend on the complexity of the patches but works on a reduced representation. This representation is supposed to capture the characteristics of a patch and can be computed in an offline process for the entire database. In contrast to the patches in the database, the target patch is not known in advance, so that its representation has to be computed on the fly. In this version of the system, we choose to base the comparison entirely on the geometric properties of both patches. Existing constraints on a source patch are not taken into account at this point.

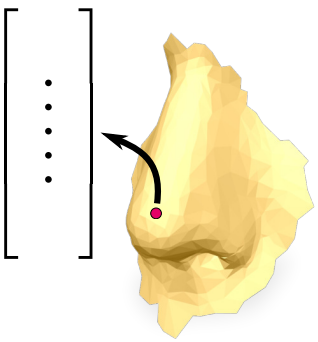
We use dense shape descriptors to characterize each point of a surface patch. Based on this, a bag-of-features approach is employed, generating a histogram-like representation for each patch. Computing a distance measure between these representations then is a very inexpensive operation. Therefore, databases of moderate size can be linearly searched for either a single best match or a set of best matches.

In the following, Section 3.1 briefly introduces a selection of relevant descriptors. Section 3.2 describes how a bag-of-features approach can be employed in our setting and Section 3.3 gives an evaluation.

3.1 Shape Descriptors

Local shape descriptors are a successful concept in computer vision. There, points of an image are described by an intermediate representation, taking into account a small neighborhood around each point. On the one hand, this representation should capture all relevant and discriminative information about this point. On the other hand, it should be widely invariant to a class of transformations, most importantly to rotations. A standard example is the extremely popular SIFT descriptor [Low04].

While image descriptors yield strong results in a variety of applications, transferring the same concept to shapes has proven to be slightly less successful. Somewhat surprisingly, local geometry does not provide as much discriminative information as a piece of a colored image, cf. [KBLB12]. Furthermore, a surface region lacks a natural global coordinate system, which is trivially given in the case of images. For these reasons, numerous shape descriptors have been proposed. However, no standard choice, providing good results in a majority of applications, has emerged. Consequently, many applications come with their own custom-tailored descriptor.



This thesis relies on local shape descriptors in both this chapter and the re-embedding technique introduced in Chapter 4. Due to the variety of existing descriptors, we formulate our algorithms as invariant as possible to a concrete choice. In particular, we treat a shape descriptor as a function mapping each point of a given surface to a vector in a fixed-dimensional space (see inset). For a surface $\mathbf{S}$, we denote this function by $\mathbf{I} : \mathbf{S} \to \mathbb{R}^d$. In a discrete setting we assume it to be defined on vertices of a mesh. Furthermore, each descriptor comes with a distance function $\boldsymbol{d}(p, q) \geq 0$, comparing elements of the descriptor space observed at point p and q. This dis-

tance may be an L1, L2 or any other kind of distance measure. While the approach described in Chapter 4 is entirely invariant to the nature of the distance function, in this chapter we make use of an embedding in Euclidean space.

While some methods evaluate descriptors only sparsely, i.e. at a number of pre-selected feature points, this work relies on dense descriptor fields, available at each point of a shape.

As a thorough evaluation of different descriptors exceeds the scope of this thesis, we pick two representatives for our experiments. For more possible descriptor choices, the reader is referred to the literature listed in Chapter 2.

Curvature Based Descriptors

A local property which is both well studied and invariant to rigid transformations is the curvature of a surface. At a point on a two-dimensional manifold, embedded in 3D, walking into a tangential direction changes the unit normal. The amount of change with respect to an infinitesimal step is called the normal curvature. The two directions in which the normal curvature takes its extreme values are called the principal directions, which are always orthogonal to each other. The magnitude of change along these directions is called the minimum and maximum curvature respectively.

Some descriptors directly adapt concepts from computer vision and regard the distribution of curvature directions in the neighborhood of a point, e.g. [LWWS15]. These however can only be measured with respect to a reference direction, e.g. the maximum curvature direction at the point itself. Unfortunately, this direction is not uniquely defined at flat or umbilical points and often unstable in their vicinity. Thus, such a descriptor is not robustly available at each point of the surface.

Instead we can simply consider the scalar curvature magnitudes which are meaningful without a reference direction. In particular, we pick three quantities computed from them: their mean, their product (called Gaussian curvature) and the absolute value of the maximum curvature. While the absolute maximum curvature only describes how much a surface is curved locally, the Gaussian curvature can distinguish between elliptic and hyperbolic regions, and the mean curvature is able to describe parabolic regions.

The required principal curvature magnitudes can be computed as the eigenvalues of the shape operator at each point. In a discrete setting, the shape operator has to be integrated over a certain neighborhood, e.g. using a Gaussian kernel. The scaling parameter σ_{curv} of this Gaussian determines how much neighborhood information is included in the value at a certain point. In other words, choosing larger values of σ_{curv} smoothes the resulting descriptor field.

Spectral Descriptors

A very popular class of state-of-the-art descriptors is based on the spectral—or frequency—analysis of a shape. Similarly to the Fourier transform, a shape is considered in its frequency spectrum, using the eigenfunctions of the Laplace-Beltrami operator as a basis. The approach is particularly attractive as this eigenbasis is invariant to isometric deformations of the shape.

Two prominent descriptors using this setting are the heat kernel signature [SOG09] and the wave kernel signature [ASC11]. The former solves the heat equation to describe diffusion properties of the shape. Intuitively, for each point, it answers the following question: if a fixed amount of heat is applied this point, how much of this heat remains after a certain time t? The extrema of the resulting field behave similarly to the Gaussian curvature. In elliptical regions heat diffuses slower, yielding high values, while in hyperbolic regions neighborhoods of larger area allow for faster diffusion, and thus result in low values. This measure is taken at different values of t and each result is used as a component of the resulting feature vector. The distance between two such vectors is built around the Euclidean distance, but also performs a normalization in each dimension and accommodates for logarithmically spaced time steps.

The second example, the wave kernel signature, is also based on the eigenvalues and eigenfunctions of the Laplace-Beltrami operator, but exploits a different physical concept. It measures the average probability over time to observe a quantum particle of a certain energy e at the surface point in question. As the heat kernel signature is parametrized over time, the wave kernel signature is parametrized over the energy e. Here, large values of e describe high frequencies of the surface, while small values correspond to low frequencies. The authors of [ASC11] propose to sample 100 discrete values of e, i.e. the wave kernel signature is a 100-dimensional descriptor. Two descriptor vectors are compared by the sum of relative differences between corresponding vector entries. As the sampling of e is chosen depending on the smallest eigenvalue, care has to be taken to choose the same sampling when comparing multiple shapes.

Practical differences between the heat kernel and the wave kernel signature can be observed considering its sensitivity to small details. While the heat kernel signature acts as a sequence of low-pass filters, and thus describes rather global properties, the wave kernel signature also incorporates the description of high-frequency detail.

Within the setting of patch retrieval, we have to make the additional choice of whether to compute a descriptor on isolated patches with boundaries, or to compute it on entire shapes and then cut out the patch. In the case of spectral descriptors the difference is significant, as this decides if global information beyond patch

boundaries is taken into account. As we prefer to regard each patch individually and independent from the model it was taken from, we aim for the first option. While in the context of shape retrieval this seems to be a reasonable choice, it is not possible to use the wave kernel signature in our mapping technique (Chapter 4). One of the reasons for this is that the boundary of a patch has a too significant impact on points close to it. Thus, it performs poorly when distinguishing between points lying on the same radius of a disk-shaped patch.

In this thesis, we aim for simple descriptor choices whenever possible. Later in this chapter, we will use a descriptor assembled from the Gaussian, mean and maximum curvature at three different scales and conduct a small experiment with the wave kernel signature. In Chapter 4 solely the mean curvature at a fixed scale will be used as a scalar descriptor to guide our inter-surface mappings.

3.2 A Bag-of-Features Approach

The bag-of-words model is a popular classification method originating in document retrieval, cf. [MRS08]. In this model, text documents are solely represented by the unordered set of words they contain, including the frequency of these words. Deliberately neglecting the structure of these words within the document is key to the simplicity of the method and coins the term *bag*-of-words for this representation. In addition, one considers a global list of discriminative words for the entire document collection. This list is called the *vocabulary*. A single document can now be classified by measuring how frequent each term of the vocabulary appears in it. This measure can be seen as an activation histogram. Consequently, documents with similar histograms are considered to belong to the same class.

This concept has been adapted to image retrieval and is commonly used in computer vision (see [PCI+07] for an example and [OD11] for a survey). Here, the terms bag-of-visual-words or *bag-of-features* are often used interchangeably. Instead of plain text, the representation of an image contains the values of some local feature descriptor evaluated at multiple locations of the image. The vocabulary now is a list of especially characteristic descriptor values. This list is usually obtained in an offline process by a quantization of the descriptor space.

The same idea has been carried over to shape retrieval and classification. In this section, we make use of the approach presented in [OBBG09]. This method densely evaluates a shape descriptor at each point of a surface. All descriptor values observed in the entire database are gathered in the descriptor space, which is then divided into a fixed number of clusters. The cluster centers, i.e. especially representative descriptor values, are now called geometric words and form

the vocabulary. When computing the activation histogram of a single shape, its descriptor values over the entire surface are matched to words of the vocabulary using a soft assignment. The resulting activation histogram of a shape is then called its bag-of-features.

Consider, for example, a point on the tip of a human nose. This point is characterized by relatively high curvature, which might be reflected in the descriptor value at that point. If the database contains a large number of human noses, their points will form a cluster in the descriptor space and its center appears as a geometric word in the vocabulary. When computing the bag-of-features, i.e. activation histogram, for each shape, those containing a nose will exhibit a high activation with respect to this word. The bag-of-features for e.g. a human eye looks differently, as a different subset of geometric features has a high importance for this shape. Thus, a distance between shapes can be computed by comparing their histograms.

Vocabulary Generation

The goal of this step is to choose a vocabulary $\mathcal{V}$ of fixed size $k = |\mathcal{V}|$, representing the existing database as well as possible. We evaluate the descriptor of our choice at all vertices of all shapes in the database and collect the resulting points in the d-dimensional descriptor space. Thereafter, we perform a vector quantization of this space using k-means clustering. The resulting cluster centers $\mathbf{I}_0, \dots \mathbf{I}_{k-1} \in \mathbb{R}^d$ form the vocabulary $\mathcal{V}$. Each of these words is similar to a large number of observed descriptor values and at the same time preserves a certain distance to all other words of the vocabulary. Thus, we expect it to have the power to both identify similar shapes as well as to discriminate between different shapes.

Since the classic k-means algorithm is inherently based on the Euclidean distance, we expect best results from descriptors having a meaningful embedding in Euclidean space. In the special case of a one-dimensional descriptor, say the discrete Gaussian curvature at a certain scale, the descriptor space is the real line. A shape's bag-of-features is then simply a histogram of its Gaussian curvature distribution. However, it provides the additional convenience, that the location and size of its—unequally spaced—bins is chosen automatically, according to the existing data.

Bag-of-Features Computation

To generate the bag-of-features representation of a shape, we first consider each vertex individually and compute its activation, also called *feature distribution*, with respect to the vocabulary. In a second step, we integrate this distribution over the surface.

The feature distribution $\boldsymbol{\theta}(v) \in \mathbb{R}^k$ at a vertex v is the vector in which each entry measures how similar the respective word is to this vertex. Since we cannot expect a geometric word to match exactly, a soft assignment is used. In [OBBG09], the similarity between the descriptor value $\mathbf{I}(v)$ at a vertex and a single word $\mathbf{I}_i$ is defined as

$$\theta_i(v) \;=\; \exp\left(-\frac{\|\mathbf{I}(v)-\mathbf{I}_i\|_2^2}{2\sigma_{\text{BoF}}^2}\right) \;>\; 0. \tag{3.1}$$

Here, a Gaussian kernel is used to turn the Euclidean distance between both descriptor values into a similarity. This similarity measure takes the value of 1 for a perfect match and approaches 0 as the distance increases. We will use the same conversion in another context in Equation 4.6 where it is explained in more detail.

To obtain the feature distribution, these similarities are assembled into a vector which is then normalized with respect to the L1 norm:

$$\boldsymbol{\theta}(v) \;=\; \frac{1}{\sum_i \theta_i} \;\cdot\; (\theta_0,\; \ldots\; \theta_{k-1})^T\,.$$

The global parameter σ_{BoF} controls the softness of the assignment. Larger values of σ_{BoF} gradually increase the similarity of more distant descriptor values and, due to the normalization of $\boldsymbol{\theta}$, lead to a softer assignment. In the limit $\sigma_{\text{BoF}} \to 0$, the entry of $\boldsymbol{\theta}$ corresponding to the closest geometric word will approach 1, while all others approach 0. The authors propose to choose σ_{BoF} as twice the median distance between the cluster centers.

Finally, the feature distribution $\boldsymbol{\theta}$ can be integrated over the surface to obtain the bag-of-features $\boldsymbol{\Theta}(\mathbf{P})$ for an entire patch $\mathbf{P}$. Hence, we evaluate the sum

$$\boldsymbol{\Theta}(\mathbf{P}) \;=\; \sum_{v\in\mathbf{P}} \;\; \boldsymbol{\theta}(v) \;\cdot\; A_v,$$

where A_v is the surface area corresponding to vertex v. Two bags-of-features, e.g. of a source patch $\mathbf{S}$ and a target patch $\mathbf{T}$ are then compared by their L1 distance:

$$\text{dist}_{\text{BoF}}\,(\mathbf{S},\mathbf{T}) \;=\; \|\boldsymbol{\Theta}(\mathbf{S}) - \boldsymbol{\Theta}(\mathbf{T})\|_1\,.$$

This means that the activations of $\mathbf{S}$ and $\mathbf{T}$ with respect to each geometric word are compared individually and their absolute differences are then summed up.

Spatial Sensitivity

The main reason for the simplicity of this method stems from the disregard of any spatial relation between surface features. While being very convenient, the loss of spatial relations at the same time presents the major disadvantage of these

methods. Regarding again the example of a Gaussian curvature based descriptor, it is possible to e.g. confuse a human nose with an entire human face, as both might exhibit the same distribution of curvature values in their bags-of-features. However, the *spatial* distributions of these values over the surfaces are different.

Therefore, the authors of [OBBG09] propose the following extension: Instead of regarding single geometric words, all pairs of words are considered in relation to their geodesic distance on the surface. Two words being important and spatially close to each other lead to a strong activation of this pair. With growing distance, the activation decreases. Thus, proximity is measured in both the descriptor space and on the surface. Unfortunately, this approach leads to a quadratic run time in the complexity of each surface patch. Furthermore, the authors only demonstrate a mild improvement of the result quality. With this dissatisfactory trade-off in mind, we do not employ this extension for our experiments, and leave improvements of spatial sensitivity for future work (see Section 5.1.2).

Descriptor Space Normalization

Some descriptors used in our examples come with large differences in magnitude among their dimensions. Furthermore, these dimensions are often independent of each other. For example, this is the case if the descriptor is built out of discrete curvature values observed at different integration radii. Due to increasing smoothness, curvature magnitudes decrease with larger integration area. Since the k-means clustering as well as the descriptor similarity defined above inherently depend on Euclidean distances, some dimensions receive a higher weight than others. We accommodate for this issue by normalizing the descriptor space such that the standard deviation over the entire database equals 1 in each dimension.

3.3 Evaluation

After a bag-of-features method was applied to our setting in the previous section, this section provides an evaluation on a small, manually created dataset.

Dataset

Our exemplary dataset contains 83 surface patches, manually extracted from laser scans of humans. These scans are collected from two sources: The FAUST dataset [BRLB14] and an as yet unpublished set of facial 3D scans. Our dataset contains 8 shape classes in the following quantities: eye (18), nose (9), ear (10), foot (10), face (9), arm (8), hand (10) and mouth (9). Examples of each class are

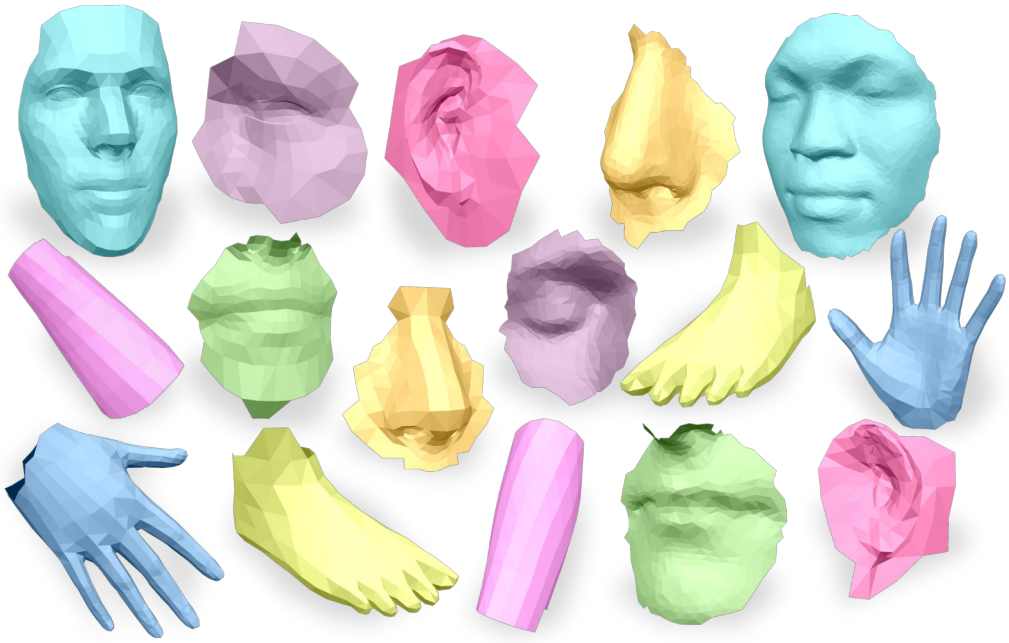


Figure 3.1: The test dataset used for evaluation consists of 83 patches divided into 8 classes: eye, nose, ear, foot, face, arm, hand and mouth.

shown in Figure 3.1. Both right-hand and left-hand instances of the classes eye, ear, foot, arm and hand are contained. However, no exact symmetries exist due to slight asymmetries of the models, unsymmetrical triangulation and different choices of patch boundaries. Patches from the FAUST dataset are used in the authors' "remeshed" triangulation, whereas meshes from our second data source are decimated to between 200 and 2000 vertices. The patch boundaries are chosen by manually selecting a center vertex and a geodesic radius. All boundaries follow edge paths of the original mesh, i.e. no triangles are split. All patches are normalized to have a geodesic radius of 1.

Descriptors

We perform experiments with the wave kernel signature (WKS) as well as a simple curvature based descriptor: Gaussian curvature, mean curvature and absolute maximum curvature are computed at three scales ($\sigma_1 = 0.025, \sigma_2 = 0.05, \sigma_3 = 0.1$) and then concatenated to a nine-dimensional descriptor. We refer to this descriptor as CURV9. In addition, a two-dimensional descriptor is used solely for illustration purposes: CURV2 consists of the mean curvature computed at σ_2 and σ_3.

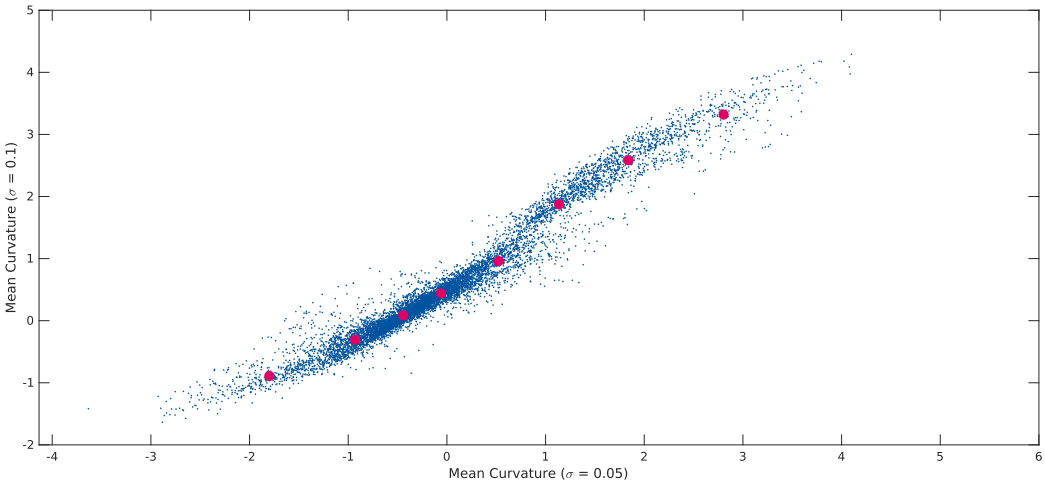


Figure 3.2: A random subsampling of 10 000 vertices within the dataset is plotted in the two-dimensional descriptor space of CURV2 (blue colored dots). A quantization of the space is performed using the k-means algorithm, 8 representatives, i.e. geometric words, are identified (magenta colored dots).

Vocabulary Generation

Figure 3.2 visualizes the distribution of descriptor values in the 2D descriptor space of CURV2. This descriptor has a meaningful embedding in Euclidean space. Thus, k-means clustering yields a vocabulary that can properly represent a majority of descriptor values.

Note that the roughly linear distribution in this example stems from the fact that both dimensions describe relatively similar surface properties, which only differ by the slightly altered kernel size. However, this is not the case in general. For example using CURV9, points with increasing maximum curvature could exhibit increasing positive or negative Gaussian curvature as well as increasing positive or negative mean curvature. Consequently, the distribution in the nine-dimensional descriptor space is far from linear in general.

Using WKS, the distribution becomes even more complex in a 100-dimensional space. As a result, a larger vocabulary size is necessary to adequately represent the distribution. Moreover, the authors of the wave kernel signature [ASC11] propose a relative difference measure to compare its values. Thus, it is not well embedded in Euclidean space, which is necessary for the classical k-means algorithm. The resulting inferior choice of the vocabulary might be one reason for the bad performance of WKS as demonstrated later in this section.

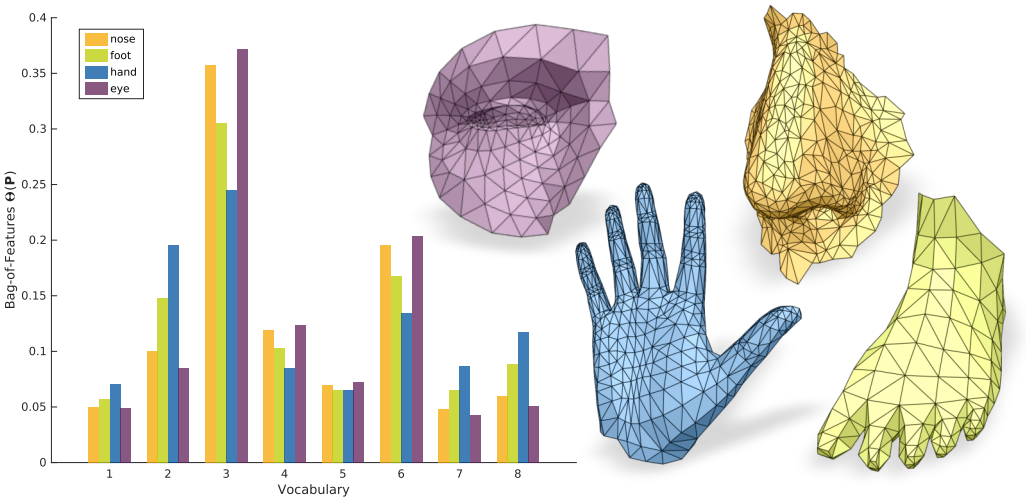


Figure 3.3: Each of the four histograms on the left visualizes the bag-of-features $\mathbf{\Theta}$ for the shape of the same color. The vocabulary ($k = 8$) is the one chosen in Figure 3.2 for the CURV2 descriptor, and an assignment softness of $\sigma_{\text{BoF}} = 1$ is used here.

Bag-of-Features Representation

Sticking to the above example with a vocabulary size of 8 and the descriptor CURV2, we computed the bag-of-features $\mathbf{\Theta}$ for the four shapes obtained in Figure 3.3. The numbers 1 to 8 on the horizontal axis correspond to the geometric words shown in Figure 3.2. Each bar shows how relevant a word is for the respective shape. It can be seen that the bags-of-features follow a general distribution over a variety of shapes. In other words, we can see that some words, e.g. 3 tend to be of higher importance in general than others, e.g. 1. This can be explained by the example of relatively flat regions, which cover large parts of a majority of patches in our database. The high general relevance of low curvature regions is reflected by the increased density of points around the origin in Figure 3.2. Still, the differences between the bags-of-features of the depicted shapes are large enough to distinguish between them.

Increasing the assignment softness parameter σ_{BoF} in Equation 3.1 smoothes out both the general distribution of words over the entire database as well as the differences between distributions of individual shapes. Yet, choosing an assignment that is too hard promotes the following problem: roughly similar points can be hard-assigned to different words and the information about their proximity is lost. Finally, the L1 norm will yield a large difference between both histograms, since all bins are compared individually. This problem is of particular relevance if the number of words k is large.

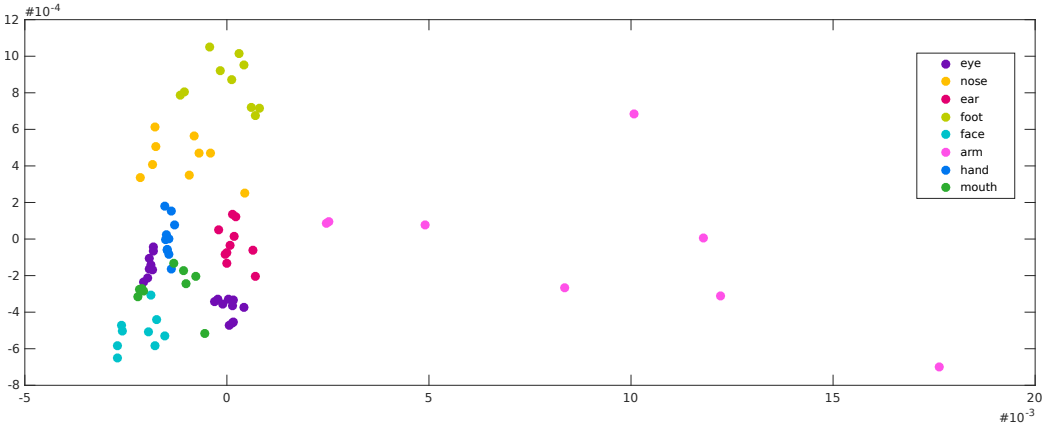


Figure 3.4: Multidimensional scaling of the bag-of-features distances $\text{dist}_{\text{BoF}}(\mathbf{P_i}, \mathbf{P_j})$ in our dataset, using the WKS descriptor, $k = 128$, and $\sigma_{\text{BoF}} = 11$.

The authors of [OBBG09] choose σ_{BoF} relative to the distance of cluster centers in the descriptor space. In contrast to that, we can choose σ_{BoF} as a constant since our descriptor spaces are normalized.

Discussion of Results

In a first experiment, the distance matrix between all patches in the database is computed using the wave kernel signature. All parameters are tuned to achieve the best results with this descriptor. Consequently, we chose a relatively large vocabulary of size $k = 128$ and a rather soft assignment using $\sigma_{\text{BoF}} = 11$. The resulting bags-of-features are points in a 128-dimensional space and we expect patches of the same class to form clusters of low L1 distance in this space. At the same time, the distance between multiple classes should be high enough to discriminate them. Figure 3.4 visualizes this 128-dimensional space using multi-dimensional scaling (MDS), i.e. distances between points are preserved as well as possible in a 2D embedding.

Although, the figure shows shapes of the same class in roughly similar positions, they do not form easily distinguishable clusters. In addition, the arm class (see inset for a representative patch) is scattered with unreasonably high distances. One possible explanation for the bad performance of the wave kernel signature was already given above, namely the potentially suboptimal choice of geometric words due to the non-Euclidean metric.

Moreover, the wave kernel signature relies on representing the entire frequency spectrum of a surface. While high frequencies are only affected by a small neighborhood, low frequencies are influenced by large surface regions. This com-

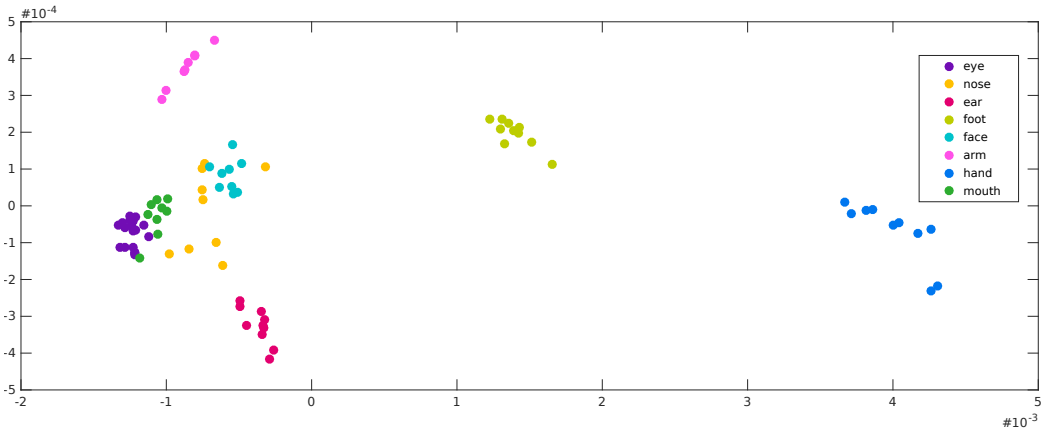
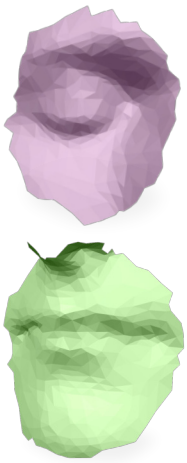


Figure 3.5: Multidimensional scaling of the bag-of-features distances $\text{dist}_{\text{BoF}}(\mathbf{P_i}, \mathbf{P_j})$ in our dataset using the CURV9 descriptor, $k = 128$, and $\sigma_{\text{BoF}} = 11$.

bination of local and increasingly global information is a key property, rendering the wave kernel signature a strong descriptor. In our setting however, global information is not present and the region of influence for a single point quickly exceeds the patch boundaries. Therefore, the wave kernel signature—as a representative for the entire family of spectral descriptors—cannot play its strength when restricted to small surface patches with boundary. In particular, this explains the unstable behavior with respect to the arm class (see inset) in Figure 3.4, as its meshes contain up to no high-frequency detail and the discriminative power of the WKS dissolves almost entirely.

As a consequence, we stick to simple descriptors based entirely on local curvature. Now, the supporting region for a single surface point is given in direct relation to the kernel size σ_{curv}. We repeat the experiment with the same settings but replace the WKS by the CURV9 descriptor and obtain pleasant results.

Figure 3.5 shows the results using CURV9. Most shape classes form clear clusters and are well separated from each other, apart from two exceptions. First, the nose class is scattered into the face class. This demonstrates the typical weakness of bag-of-features approaches, namely their disregard of spatial relations. Both, faces and some noses seem to exhibit a similar distribution of curvature values. If the position of these values is not taken into account, it is plausible that a nose can be confused for a head.



Second, the eye and mouth clusters are too close to each other to clearly distinguish between them. This however is not surprising as their surfaces show very similar details and little discriminating geometric information exists in the first place (see inset).

We keep the parameter choice from Figure 3.5 (CURV9, $k = 128$, $\sigma_{\text{BoF}} = 11$), and further analyze the results. To do so, we in turn treat each patch as the query patch and examine its nearest neighbors. When picking the single nearest neighbor of each patch, the result belongs to the same class in 79 out of 83 cases. In the four failure cases, the closest patch of the correct class is the 2nd, 3rd, 10th and 13th nearest neighbor, respectively.

Again, we regard each patch as the query once and pick its n nearest neighbors as the result set. For evaluation, we count the items of the confusion matrix: true positives (retrieved patches of the correct class), false positives (retrieved patches of a wrong class), false negatives (patches of the correct class that were not retrieved) and true negatives (patches of a wrong class that were not retrieved). From these numbers, we can compute precision and recall. Precision is the ratio of correct patches within the n retrieved patches and recall is the ratio of how many patches of the correct class are actually retrieved. In Figure 3.6 both measures are plotted with respect to an increasing size n of the result set. For example, it can be seen that retrieving four patches gives 90% results of the same class on average and roughly 30% of the correct class are actually within the retrieval set.

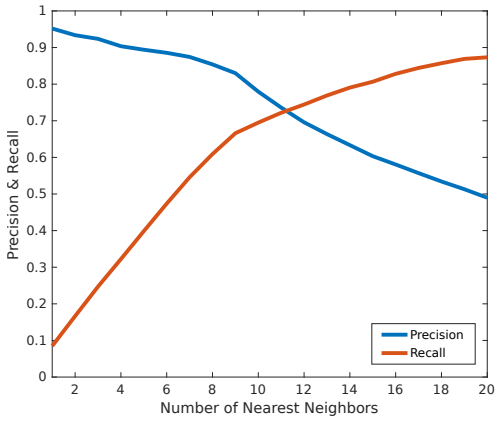


Figure 3.6: Precision and recall of our bag-of-features implementation per number of retrieved patches n.

Timings

In our unoptimized MATLAB implementation, initializing the evaluation dataset takes approximately 12 seconds on commodity hardware, while the lookup time for a query is negligible.

In this chapter, one of many possible shape retrieval methods was adapted to our setting. An evaluation shows that the bag-of-features approach using a simple descriptor, built of a few curvature measures at different scales, yields satisfying results. In future work, the method could be further improved by addressing its major shortcoming, namely the indifference towards spatial relations (see Section 5.1.2).

Chapter 4

Patch Re-Embedding

This chapter presents the main contribution of this thesis. It addresses the intricate problem of re-embedding a successfully retrieved macro constraint on the target surface. As such a macro constraint is defined on a surface patch which has similar but not identical geometry to the target surface, this re-embedding step is not straightforward. Naïve solutions, such as rigidly aligning both surfaces in 3D and employing a simple projection, provide insufficient results. This is because geometric differences are not accommodated for in a controlled fashion and an arbitrary number of artifacts is possible. Imagine, for example, two human faces being rigidly aligned to each other, both exhibiting subtle differences in the relative position and size of geometric features. Projecting constraints from one surface to the other using e.g. normal piercing or nearest neighbor correspondences, yields critical disadvantages: (1) geometric features might not exactly correspond to each other; (2) in highly curved regions, such as ears, the mapping is likely to become discontinuous and non-injective; (3) the mapping lacks a general notion of smoothness; and (4) it is essentially restricted to patches which are rigid transformations of each other.

Inter-Surface Mappings

For a constraint transfer that is free from the above limitations, we strive to compute a continuous map between the source and the target surface. This map, which can also be interpreted as a field of dense point-to-point correspondences, can then be used to transfer a broad range of information between the surfaces. In fact, it is widely agnostic to the type of constraints being mapped. Especially, it allows to transfer smooth fields, for example containing sizing constraints.

To keep the setting relatively simple, we maintain the restrictions already employed in the previous chapters. In particular, we compute a mapping between a source and a target patch, both being geodesic discs with a single boundary. Thus,

we choose to ignore existing target geometry beyond the boundary of the user-selected target patch. This assumption could be relaxed in future work. Moreover, we cannot expect the boundaries of both patches to exactly match each other—a presumption made in a variety of existing methods. Hence, we only aim to compute a mapping between the "common parts" of both patches.

In order to address the alignment of geometric features, i.e. problem (1), we choose a descriptor guided approach. This means that a surface descriptor is computed on both patches, and the desired mapping is sought to align surface regions with similar descriptor values.

The continuity of the resulting inter-surface map will be given by the design of our method. Further, injectivity constraints as well as distortion measures can be explicitly formulated in our setting. Choosing a smoothness measure involves the following question: which kinds of distortion do we expect in our mappings and which ones should be penalized? A common path taken in the literature is to optimize for low isometric distortion. However, our method should eventually be able to transfer constraints between patches of moderate geometric differences beyond isometries. Consequently, we cannot expect this goal to be reached. To allow more degrees of freedom in our map, we aim for the slightly weaker notion of conformality. A conformal, i.e. angle preserving, map allows for an isotropic scaling factor at each point. This gives a map more flexibility to adjust to scale differences of local features.

Although good descriptor matches and low distortion often coincide, they are different concepts and have to be balanced against each other. For example, situations exist in which permitting a slight increase in distortion allows for a significant improvement of descriptor distances, and vice versa.

Optimization Setting

We observe that the degrees of freedom in the problem at hand are inherently two-dimensional: consider a point on the source patch which is mapped to a specific position on the target patch. Now, altering the inter-surface map means moving this point to a different location on the target surface. Despite the 3D embedding of this surface, the point can only be moved in two dimensions. Due to this fact, we choose to approach the problem in a 2D parametrization space. In this space, point-to-point correspondences are trivially given by points lying at the same 2D location if the parametrizations are injective. As an additional benefit, this setting allows to find the "common part" of both patches in a natural way, without formulating an additional optimization problem.

Alternative Approaches

A different class of methods aims to solve the problem directly in its three-dimensional embedding space. Here, a non-rigid alignment is computed by deforming the source surface until it matches the target. Correspondences are then found by points located at the same 3D position. Although this approach initially seems attractive, formulating both distortion and injectivity measures in three dimensions is substantially less straightforward than in 2D. Furthermore, the non-rigid registration process itself is intricate: similar to rigid registration it carries a cyclic dependency between improving an intermediate alignment and updating a set of tentative correspondences.

Another family of methods approaches the problem using a purely discrete optimization. For example, correspondences between two sets of points can be found by solving a labeling problem. This however bears the challenge of first determining appropriate point sets on both surfaces, as unequal numbers of elements prevent a one-to-one matching. Smoothness of the resulting correspondence field is often encouraged by rewarding neighboring points of the source set for choosing neighboring correspondences on the target surface. This kind of problem is often expressed in a probabilistic model, e.g. by a Markov random field. Again, explicit control over the distortion of the induced map is not easily given.

Outline

In this chapter, we start by finding a suitable problem formulation in a continuous setting (Section 4.1). For some parts of the problem, different alternatives are discussed. Afterwards, we discretize all relevant expressions and examine their first derivatives (Section 4.2). Finally, a first approach towards optimizing the resulting non-linear, non-convex objective function is presented (Section 4.3) and the involved challenges are discussed in detail. Finally, an experimental evaluation is given (Section 4.4).

4.1 Problem Formulation

We are given a source patch $\mathbf{S}$ and a target patch $\mathbf{T}$. Both are two-manifold, simply connected surfaces with single boundary $\partial\mathbf{S}$ and $\partial\mathbf{T}$, respectively. We are searching for an inter-surface mapping $\mathbf{\Phi}$ (see Figure 4.1).

Both patches are assumed to contain similar geometry, but their boundaries are not expected to exactly match each other. Thus, we only consider a partial map between subsets $\mathbf{S}' \subseteq \mathbf{S}$ and $\mathbf{T}' \subseteq \mathbf{T}$. Points in either $\mathbf{S}'$ or $\mathbf{T}'$ are said to lie in the overlapping region. Now the desired mapping is defined as $\mathbf{\Phi} : \mathbf{S}' \to \mathbf{T}'$, where in addition to the map $\mathbf{\Phi}$ itself, both $\mathbf{S}'$ and $\mathbf{T}'$ are unknown.

The primary objective of our optimization is to map points to geometrically similar locations. A measure for this similarity is provided by pointwise surface descriptors on both patches and a descriptor distance function $\boldsymbol{d}(p, q) \geq 0$ for $p \in \mathbf{S}$, $q \in \mathbf{T}$. Hence, we want $\boldsymbol{\Phi}$ to minimize this distance for all corresponding points in the overlapping region:

$$\mathrm{obj}_{\mathrm{descr}} \quad = \quad \int_{\mathbf{S}'} \boldsymbol{d}\left(p, \boldsymbol{\Phi}(p)\right)\, dA \quad \rightarrow \min . \tag{4.1}$$

We design our optimization to be invariant to the choice of a descriptor, as long as $\boldsymbol{d}(p, q)$ is densely available, that is for all pairs of surface points.

The objective 4.1, with its variables encoded in $\boldsymbol{\Phi}$, already is a non-linear, non-convex function, potentially having numerous local minima. To see this, consider a point $p \in \mathbf{S}'$ and its correspondence $q = \boldsymbol{\Phi}(p) \in \mathbf{T}'$. Following a line through the solution space of the problem, q describes a path on the target surface along which the descriptor distance to p can increase and decrease arbitrarily.

4.1.1 Parametrization Setting

The degrees of freedom in $\boldsymbol{\Phi}$ are continuous and two-dimensional: namely for each point p, its correspondence q can be moved over the target surface. However, the fact that the domain $\mathbf{S}'$ is unknown poses an additional challenge. Having to choose whether a correspondence for p exists or not, i.e $p \in \mathbf{S}'$ or $p \notin \mathbf{S}'$, adds binary degrees of freedom to the problem.

Fortunately, our restriction on $\mathbf{S}$ and $\mathbf{T}$ to have disk topology, allows for an intuitive formulation using parametrization and deformation, that covers all degrees of freedom in a single, two-dimensional domain. In a nutshell, our setting is the following: (1) we parametrize both patches $\mathbf{S}$ and $\mathbf{T}$ to the unit disk $\boldsymbol{\Omega}$ in $\mathbb{R}^2$, (2) while the disk of the target patch stays fixed, we compute a continuous 2D deformation of the source disk. After this deformation, the overlapping part between both 2D shapes defines $\mathbf{S}'$, and correspondences are trivially given by points lying at the same 2D location. See Figure 4.1 for an illustration.

More formally, the parametrizations $\boldsymbol{f} : \mathbf{S} \rightarrow \boldsymbol{\Omega}$ and $\boldsymbol{g} : \mathbf{T} \rightarrow \boldsymbol{\Omega}$ map each point from the respective surface to the unit disk $\boldsymbol{\Omega} \subset \mathbb{R}^2$, i.e. a subset of the plane. Many techniques for this step are available. For the sake of simplicity, we start by using harmonic parametrizations and constrain the boundary of the respective patch to the unit circle. Both $\boldsymbol{f}$ and $\boldsymbol{g}$ can be computed such that they are guaranteed to be bijective, cf. [Tut63] and [Flo03].

The continuous 2D deformation applied to the source parametrization is called $\boldsymbol{\phi} : \boldsymbol{\Omega} \rightarrow \mathbb{R}^2$. All degrees of freedom for our optimization are now encapsulated within the choice of $\boldsymbol{\phi}$.

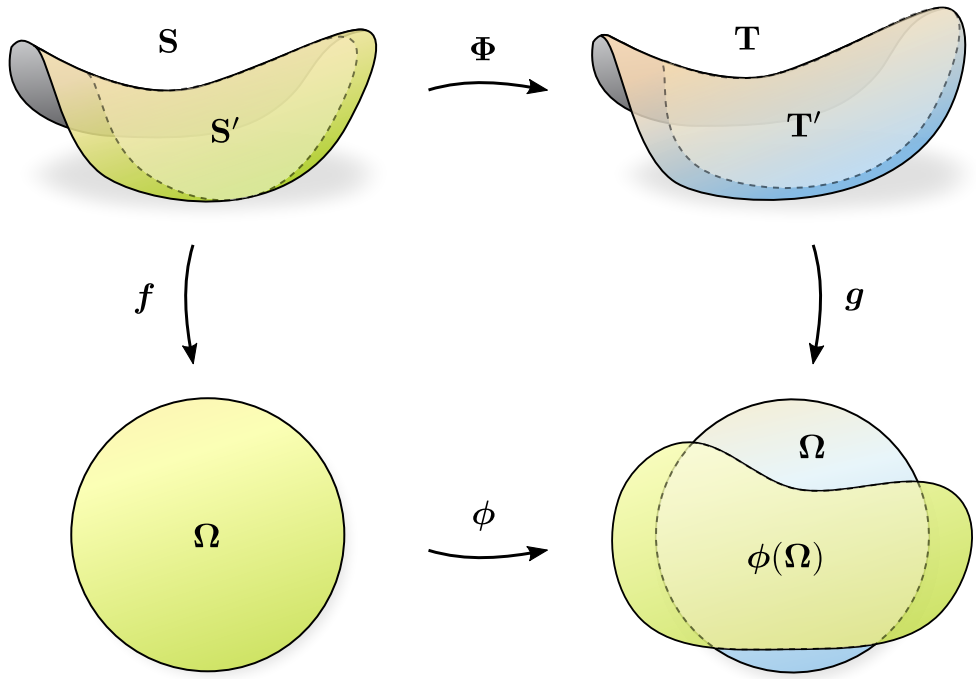


Figure 4.1: Correspondences between subsets of the surfaces $\mathbf{S}$ and $\mathbf{T}$ are computed via an intermediate parametrization domain. First, $\mathbf{S}$ and $\mathbf{T}$ are both mapped to the unit disk $\boldsymbol{\Omega}$ by $\boldsymbol{f}$ and $\boldsymbol{g}$. After that, the disk belonging to $\mathbf{S}$ is deformed by a map $\boldsymbol{\phi}$ in 2D. Finally, the overlap of both shapes defines the partial inter-surface map $\boldsymbol{\Phi}$.

Finally, we can define the desired inter-surface map as the composition

$$\boldsymbol{\Phi} = \boldsymbol{g}^{-1} \circ \boldsymbol{\phi} \circ \boldsymbol{f}.$$

Thus, a single point from the source patch is first mapped to the plane by $\boldsymbol{f}$, then re-located within the plane by $\boldsymbol{\phi}$ and eventually lifted to the target surface by $\boldsymbol{g}^{-1}$. If a point lies outside the unit circle after the deformation was applied, it cannot be lifted to the target and is thus not contained in $\mathbf{S}'$. The target region $\mathbf{T}'$ is defined to be the subset of $\mathbf{T}$ for which an overlap with the deformed 2D shape exists, i.e. $\mathbf{T}' = \boldsymbol{g}^{-1}\left(\boldsymbol{\phi}(\boldsymbol{\Omega}) \cap \boldsymbol{\Omega}\right)$.

4.1.2 Map Distortion

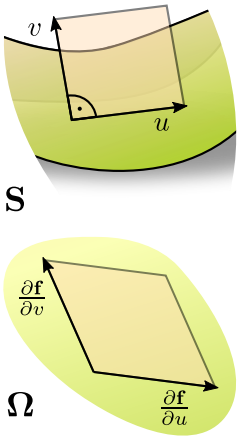
Each of the three maps introduces its own distortion. Especially harmonic parametrizations as used for $\boldsymbol{f}$ and $\boldsymbol{g}$ are known to exhibit high area distortion. Typically, areas close to the boundary are stretched when mapped to the parameter domain,

while areas towards the center or with high Gaussian curvature can exhibit extreme shrinkage. Consequently, it is important to regard the metric induced by both $\boldsymbol{f}$ and $\boldsymbol{g}$ during the optimization. For example, moving a correspondence by a small distance in the parameter domain can have very different effects. Close to the boundary, the change in the overall result might be negligible, while it might be extreme if the same movement is performed close to the center of the disc. Similar differences can be observed for angular distortion, which tends to become high in the proximity of the boundary.

While $\boldsymbol{f}$ and $\boldsymbol{g}$ stay fixed, $\boldsymbol{\phi}$ changes during the optimization. When $\boldsymbol{\phi}$ is initialized as the identity, the optimization gradually deforms the planar unit disk $\boldsymbol{\Omega}$ and thus builds up additional distortion. Although the distortions of $\boldsymbol{f}$ and $\boldsymbol{g}$ show similar characteristics, they are not the same for two reasons: (1) the corresponding 3D surfaces are not the same and have to undergo some geometric deformation to be morphed into each other; and (2) we don't require the boundaries of both patches to match. For example, a shared geometric feature might lie at the center of one disk but slightly off-center in the other disc. Hence, both features are likely to be distorted differently.

For these reasons, we are not only interested in the distortion introduced by $\boldsymbol{\phi}$, but in the distortion of the entire composition $\boldsymbol{\Phi} = \boldsymbol{g}^{-1} \circ \boldsymbol{\phi} \circ \boldsymbol{f}$. In fact, increased distortion in $\boldsymbol{\phi}$ can accommodate for differences between $\boldsymbol{f}$ and $\boldsymbol{g}$, and thus reduce the overall distortion in $\boldsymbol{\Phi}$.

Singular Values of the Jacobian



While the functions $\boldsymbol{f}$, $\boldsymbol{\phi}$ and $\boldsymbol{g}$ map points from one domain to another, their differentials do the same with tangent vectors. Imagine an orthonormal tangent frame with axes u and v at a point on $\mathbf{S}$. When mapped to the plane, this tangent frame gets distorted by a linear transformation, i.e. scaled, rotated and sheared (see inset). This transformation is represented by the the Jacobian matrix

$$\mathbf{J}_{\boldsymbol{f}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial u} & \frac{\partial \boldsymbol{f}}{\partial v} \end{bmatrix},$$

which exists at each point on $\mathbf{S}$. The same definition applies to $\mathbf{J}_{\boldsymbol{\phi}}$ and $\mathbf{J}_{\boldsymbol{g}}$. As the domain of $\mathbf{J}_{\boldsymbol{\phi}}$ is two-dimensional, the choice of a tangent frame is particularly simple, e.g. using the Cartesian coordinate axes. Technically, $\mathbf{J}_{\boldsymbol{f}}$ and $\mathbf{J}_{\boldsymbol{g}}$ map between 2D and 3D vectors, giving us Jacobian matrices in $\mathbb{R}^{2\times3}$. However, since we only consider 3D vectors living in the tangent space of a surface, we can always express them locally in a tangential 2D coordinate system. Thus, we write $\mathbf{J}_{\boldsymbol{f}}, \mathbf{J}_{\boldsymbol{\phi}}, \mathbf{J}_{\boldsymbol{g}} \in \mathbb{R}^{2\times2}$.

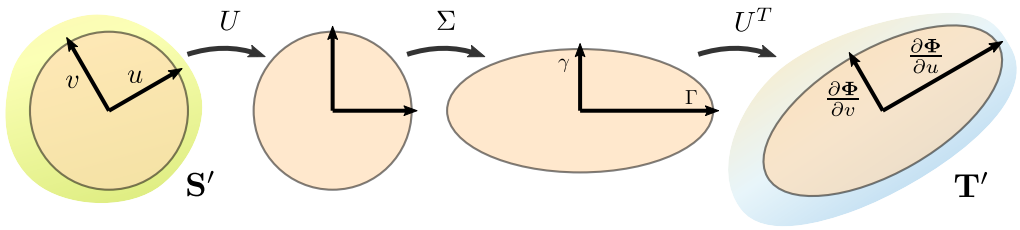


Figure 4.2: The singular value decomposition separates the Jacobian $\mathbf{J}_{\mathbf{\Phi}}$ into a rotation U, an anisotropic scaling Σ and another rotation V^T.

Since we are interested in the distortion of the entire map $\mathbf{\Phi}$, we now regard the composition $\mathbf{J}_{\mathbf{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \circ \mathbf{J}_{\boldsymbol{\phi}} \circ \mathbf{J}_{\boldsymbol{f}}$. Due to the linearity of the Jacobian, this is simply the matrix multiplication

$$\mathbf{J}_{\mathbf{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}.$$

The linear distortion of our map is completely characterized by this 2×2 matrix at each point. Yet, it carries some additional information we are not interested in: namely its rotational part, as the mere rotation of a local frame does not add any distortion. Moreover, having to choose a tangential coordinate system for each point actually adds an arbitrary rotation to its Jacobian.

Numerous distortion measures based on the Jacobian have been introduced in the literature. The most popular family of measures examines its singular value decomposition $\mathbf{J}_{\mathbf{\Phi}} = U\Sigma V^T$ (e.g. [DMK03], [HG00], [SCGL02], [APL14]). Here, the Jacobian is split into two 2D rotations U and V^T and an anisotropic scaling by the singular values

$$\Sigma = \begin{bmatrix} \Gamma & 0 \\ 0 & \gamma \end{bmatrix}$$

with $\Gamma \geq \gamma$, see Figure 4.2. Now these singular values can be combined into a single distortion measure. If $\Gamma = \gamma = 1$ at every point, the map is *isometric*, as no length distortion exists. If Γ and γ take the same value α, not necessarily 1, the map is *conformal*. This means that both axes are scaled uniformly and thus angles cannot change. Still, length distortion is possible, since the local scaling factor α can vary over the surface.

In a majority of cases, we will not be able to find an isometric mapping, because we do not expect our surfaces $\mathbf{S}$ and $\mathbf{T}$ to be isometric deformations of each other. A conformal mapping however is always possible as long as we are working in a continuous setting. The uniformization theorem states that a simply connected surface with boundary can always be conformally mapped to the unit disk [Kee13]. Applying this statement to $\boldsymbol{f}$ and $\boldsymbol{g}$, and setting $\boldsymbol{\phi}$ to the identity,

shows that a conformal map $\mathbf{\Phi}$ exists. Yet in practice, we have to work with discretized surfaces for which the theorem does not hold. Therefore, we can just aim to minimize conformal distortion.

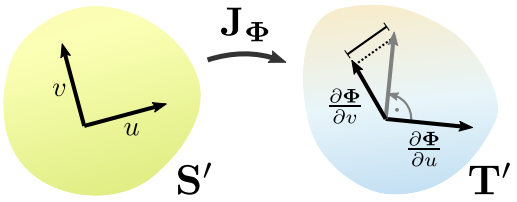
Among others, [DMK03] suggest to use the measure $\frac{\Gamma}{\gamma}$ for conformal distortion. If the mapping is perfectly conformal in a point, this ratio is 1 and it grows as the relative difference between both singular values increases. Although it is easy to compute this measure, it is harder to directly optimize for it. The central problem is that this objective depends on the singular values of a 2×2 matrix. Closed form expressions for these values exist (e.g. [Bli96], [SCGL02], [SS15], [Lip12]) but they are non-convex. Since our method cannot guarantee to give an initial solution close enough to the optimum, there is a risk of getting stuck in local minima. [Lip12] propose a convex formulation to strictly bound $\frac{\Gamma}{\gamma}$ at the cost of cutting away valid parts of the solution space. In our setting however, it is more beneficial to minimize the overall distortion than to adhere to explicit bounds.

Several other measures based on the singular values have been proposed, e.g. $\frac{\Gamma}{\gamma} + \frac{\gamma}{\Gamma}$ [HG00], $\max(\gamma, \frac{1}{\Gamma})$ [SCGL02] or $\sqrt{\gamma^2 + \Gamma^{-2}}$ [APL14]. All these measures describe different combinations of isometry and conformality and are of varying difficulty to optimize for.

To avoid additional issues arising from these more involved distortion measures, we use a well-known approximation to conformality, which is both easy to derive and simple to optimize for.

Least-Squares Conformal Maps

A very popular approximation to conformality, are least-squares conformal maps, introduced in [LPRM02]. While the approach is originally used to map a disc-like surface to the plane, we can apply the same concept to the inter-surface map $\mathbf{\Phi}$.



Regard again an orthonormal tangent frame at a point on $\mathbf{S}'$. When mapped to $\mathbf{T}'$, we would like both vectors to still be orthogonal and of equal length, as this implies conformality. Choosing a tangential coordinate system (u, v) on $\mathbf{S}'$, its coordinate axes get mapped to $\frac{\partial \mathbf{\Phi}}{\partial u}$ and $\frac{\partial \mathbf{\Phi}}{\partial v} \in \mathbb{R}^2$ on $\mathbf{T}'$, also expressed in a local coordinate system. Rotating $\frac{\partial \mathbf{\Phi}}{\partial v}$ by $90°$ counter-clockwise gives us $\frac{\partial \mathbf{\Phi}}{\partial u}$ if the map is conformal at this point. If this is not exactly the case, the difference between both vectors measures how far the map is from being conformal (see inset). This measure is then minimized in a least-squares sense. Thus, we measure the conformal distortion of $\mathbf{\Phi}$

at a point p as

$$\mathrm{lscm}_{\mathbf{\Phi}}(p) \;=\; \left\| \frac{\partial \mathbf{\Phi}}{\partial v} - \mathrm{Rot}_{90} \frac{\partial \mathbf{\Phi}}{\partial u} \right\|^2$$

and seek to minimize

$$\int_{\mathbf{S}'} \mathrm{lscm}_{\mathbf{\Phi}}(p)\; dA.$$

When used in the original parametrization setting (surface to plane), the least-squares conformal maps term is convex and quadratic. Furthermore, its minimum is invariant to similarity transformations, i.e. scaling, rotation and translation, in parameter space. For this reason, one usually fixes two points of the map to obtain a unique minimum. This minimum can then be found by solving a single linear system.

In our setting however, the conformality measure is not based on the Jacobian of a single map, but on the composition $\mathbf{J}_{\mathbf{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}$. Consider a single point v on $\mathbf{S}'$. For this point, the Jacobian of the first map $\mathbf{J}_{\boldsymbol{f}}$ is always constant. The Jacobian of the second map $\mathbf{J}_{\boldsymbol{\phi}}$ is not constant, as it contains the variables of our problem. Moving around the variable point $\boldsymbol{\phi}(v)$ in the parameter domain clearly changes $\mathbf{J}_{\boldsymbol{\phi}}$. So far, the least-squares conformal maps term still exhibits its original properties, since its variables are just multiplied by a constant matrix. Finally, we also take the last Jacobian $\mathbf{J}_{\boldsymbol{g}}^{-1}$ into account. This Jacobian is picked at the point on the target patch corresponding to $\boldsymbol{\phi}(v)$. Since moving $\boldsymbol{\phi}(v)$ around changes this point, the Jacobian $\mathbf{J}_{\boldsymbol{g}}^{-1}$ is picked at varying locations. Unfortunately, $\mathbf{J}_{\boldsymbol{g}}^{-1}$ can take arbitrary values at different locations. Thus, the variables of the least-squares conformal maps term are multiplied by arbitrarily changing values. For this reason, the least-squares conformal maps term is no longer quadratic nor convex in the variables of our problem. This is why we cannot directly solve for its minimum but have to fall back on non-linear optimization techniques.

Moreover, there is no need to fix two correspondences, as the other parts of the objective serve as a regularization.

Note that the above integral only covers $\mathbf{S}'$, as $\mathbf{\Phi}$ is solely defined on this domain. In order to rate a single instance of $\mathbf{\Phi}$ this is sufficient, since $\mathbf{\Phi}$ does not depend on the remaining part of $\mathbf{S}$. During the optimization however, the entire parametrization of the source patch is deformed and points can freely move in and out of the overlapping region. Therefore, it is important that points in $\mathbf{S} \setminus \mathbf{S}'$ are also moved reasonably during the optimization process. Because $\boldsymbol{g}^{-1}$ is not available for these points, we cannot use the same distortion measure. Instead, we ignore the constant distortion of $\boldsymbol{f}$ as well as the non-existing $\boldsymbol{g}^{-1}$, and simply apply our measure to the 2D deformation $\boldsymbol{\phi}$:

$$\mathrm{lscm}_{\boldsymbol{\phi}}(p) \;=\; \left\| \frac{\partial \boldsymbol{\phi}}{\partial v} - \mathrm{Rot}_{90} \frac{\partial \boldsymbol{\phi}}{\partial u} \right\|^2 .$$

In total, we seek to minimize the following conformal distortion objective:

$$\text{obj}_{\text{lscm}} \quad = \quad \int_{\mathbf{S}'} \text{lscm}_{\mathbf{\Phi}}(p)\, dA \quad + \quad \int_{\mathbf{S}\setminus\mathbf{S}'} \text{lscm}_{\boldsymbol{\phi}}(p)\, dA. \tag{4.2}$$

Both terms are integrated using the area element dA of the original 3D embedding of the surface $\mathbf{S}$.

4.1.3 Injectivity

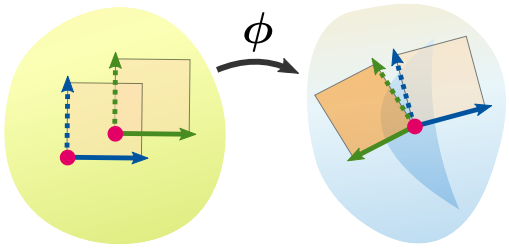
To be able to consistently map information from the source to the target surface, the mapping $\mathbf{\Phi}$ should be injective. Otherwise, information from multiple source points will be mapped to the same target location, which might be problematic.

Recall that both surface maps $\boldsymbol{f}$ and $\boldsymbol{g}$ to the unit disk are guaranteed to be bijective. If we can compute a deformation $\boldsymbol{\phi}$ that is injective, it follows that the entire mapping $\mathbf{\Phi}$ is also injective. Since we set the co-domain of $\mathbf{\Phi}$ to its actual range $\mathbf{T}'$, it is also surjective by definition. Hence in total, injectivity of $\boldsymbol{\phi}$ will lead to a bijective map $\mathbf{\Phi}$.

In the following, we will distinguish between local and global injectivity. Local injectivity means that, in an infinitesimal neighborhood around each point, the map is free of fold-overs. Whereas global injectivity is given if the boundary of the source patch does not intersect itself.

Local Injectivity

Similarly to the discussion of distortion in Section 4.1.2, local injectivity can be characterized by the Jacobian of a map. Now however, we can restrict our observations to the deformation $\boldsymbol{\phi}$.



The Jacobian $\mathbf{J}_{\boldsymbol{\phi}}$ shows how vectors in the parameter domain $\mathbf{\Omega}$ are affected by the deformation. Local injectivity is given when these vectors preserve their order in each point. This condition can be checked by picking two vectors spanning a positive area in $\mathbf{\Omega}$ and making sure that this area stays positive under $\boldsymbol{\phi}$ (see inset). If we pick the normalized coordinate axes $\mathbf{u} = (0,1)^T$ and $\mathbf{v} = (1,0)^T$ as our vectors, their images are the columns of the Jacobian $\mathbf{J}_{\boldsymbol{\phi}} = \left[\begin{array}{cc} \frac{\partial \boldsymbol{\phi}}{\partial u} & \frac{\partial \boldsymbol{\phi}}{\partial v} \end{array} \right]$. The area of $\mathbf{u}$ and $\mathbf{v}$ is 1, whereas $\boldsymbol{\phi}(\mathbf{u})$ and $\boldsymbol{\phi}(\mathbf{v})$ span a parallelogram of signed area $\det \mathbf{J}_{\boldsymbol{\phi}}$. Thus, we would like to enforce the constraint

$$\det \mathbf{J}_{\boldsymbol{\phi}} > 0$$

at all points in $\mathbf{S}$. To stay within the realm of unconstrained minimization, we employ the barrier function

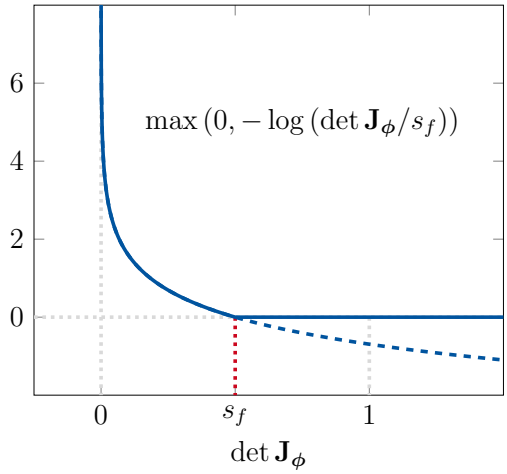
$$-\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}\right)$$

instead. If the area $\det \mathbf{J}_{\boldsymbol{\phi}}$ is sufficiently positive, the term yields a result reasonably close to 0. However, as the area approaches 0, the term explodes and gives extremely high numbers, quickly approaching ∞ in the limit $\det \mathbf{J}_{\boldsymbol{\phi}} \to 0$.

Although, the logarithm increases very slowly for arguments far enough from 0, the term still pursues infinite growth of $\boldsymbol{\phi}(\boldsymbol{\Omega})$. Moreover, it yields negative values for $\det \mathbf{J}_{\boldsymbol{\phi}} > 1$. We therefore clip negative values to 0 and thus obtain a function with local support:

$$\max\left(0, -\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}\right)\right).$$

The support of this modified barrier starts at $\det \mathbf{J}_{\boldsymbol{\phi}} = 1$, i.e. in the area preserving case. Now, area growth is neither encouraged nor punished, while slight area shrinkage is still penalized. To work around this issue, we move the starting point of the support from 1 to a different location closer to 0. This is done by introducing a scaling parameter $s_f \in (0, 1]$. The parameter s_f now defines the value of $\det \mathbf{J}_{\boldsymbol{\phi}}$ at which the barrier function starts having an effect (see inset):



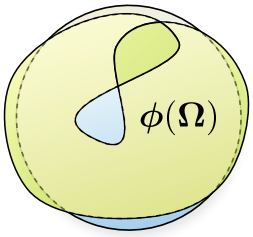
$$\max\left(0, -\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}/s_f\right)\right).$$

Finally, given a feasible initial solution, we can guarantee local injectivity by adding the term

$$\text{obj}_{\text{flip}} \quad = \quad \int_{\mathbf{S}} \max\left(0, -\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}/s_f\right)\right) \, dA \tag{4.3}$$

to our objective function. Notice that by choosing $\mathbf{S}$ as the integration domain, we also enforce this condition in the non-overlapping part of the source surface. This allows $\mathbf{S}'$ to grow freely during the optimization without hitting any non-injective elements from $\mathbf{S} \setminus \mathbf{S}'$.

Global Injectivity



To achieve global injectivity of ϕ, and thus of Φ, the boundary has to stay free of self intersections (see inset). Depending on the purpose of the mapping, this property might not strictly be necessary. In fact, we do not include it in our final objective. Still, we discuss two possible options to address the issue here.

The challenge in guaranteeing an intersection-free boundary arises from the global nature of this constraint, as it prescribes relations between all pairs of boundary points. In [SS15], the following barrier function is set up for all boundary points p and edges e:

$$\max\left(0, \frac{\varepsilon}{\operatorname{dist}(p,e)} - 1\right)^2.$$

If p is within distance of ε of the boundary edge e, the fraction will become larger than 1, resulting in a positive function value. As the point approaches the edge, the barrier approaches ∞. Outside of the region defined by ε, the function is clipped to 0.

Despite the number of barrier functions being quadratic in the size of the boundary, they can still be evaluated quite efficiently due to their local support. Using a 2D acceleration structure, the small subset of pairs (p, e) potentially having an effect on each other can be found quickly.

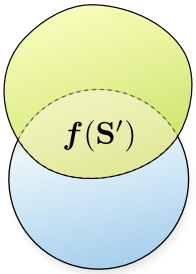
In our setting, another very simple option to guarantee global injectivity is conceivable. Since both surfaces are geodesic discs containing similar geometry, we do not expect the deformation ϕ to dramatically change the shape of the unit circle. In particular, we can restrict the space of deformations to those that maintain its convexity. Fortunately, this constraint can be formulated locally, by restricting the signed curvature κ of the boundary curve of $\phi(\Omega)$ to positive values. We use the same concept of a log-barrier function as we did for local injectivity:

$$\text{obj}_{\text{convex}} \quad = \quad \int_{\partial \mathbf{S}} \max(0, -\log\left(\kappa / s_c\right))\, dl. \tag{4.4}$$

Again, the parameter s_c defines the value of κ at which the function starts to have an effect. The curvature of the unit circle is $1/r = 1$ at each point, thus s_c should be set to a small value in $(0, 1]$. We integrate the barrier function over the boundary curve using the length element dl of $\partial\mathbf{S}$.

Although being effective, this constraint is a very conservative measure for global injectivity and cuts off large parts of the solution space. As we do not observe problems with respect to global injectivity in our examples, we do not add this term to the objective function used in our experiments.

4.1.4 Sufficient Overlap



So far, combining Equations 4.1, 4.2 and 4.3, i.e. $\mathrm{obj}_{\mathrm{descr}}$, $\mathrm{obj}_{\mathrm{lscm}}$ and $\mathrm{obj}_{\mathrm{flip}}$, to a single objective describes almost all of our desired properties. However, there is still one caveat. The global optimum is a solution with no overlap between the two patches at all, i.e. $\mathbf{S}' = \mathbf{T}' = \emptyset$. This is because $\mathrm{obj}_{\mathrm{descr}}$ and $\mathrm{obj}_{\mathrm{lscm}}$ penalize errors integrated over $\mathbf{S}'$ and there is no reward for $\mathbf{S}'$ maintaining a certain size.

Several approaches to introduce this reward into the optimization target are imaginable. A measure for the amount of overlap is simply given by $\frac{|\mathbf{S}'|}{|\mathbf{S}|} \in [0, 1]$. Here $|\mathbf{S}|$ and $|\mathbf{S}'|$ denote the original surface areas in 3D. The inset shows the image of $\mathbf{S}'$ in the parameter domain.

Penalty Factor

A first idea is to multiply parts of the objective function by the reciprocal of this fraction, i.e.

$$\mathrm{obj} \quad = \quad \frac{|\mathbf{S}|}{|\mathbf{S}'|} \cdot (\mathrm{obj}_{\mathrm{descr}} + \mathrm{obj}_{\mathrm{lscm}}) + \mathrm{obj}_{\mathrm{flip}},$$

and choosing $\mathrm{obj} = \infty$ if $|\mathbf{S}'| = 0$. In the case of maximum overlap, $|\mathbf{S}'| = |\mathbf{S}|$, the total objective value stays unaltered. As $|\mathbf{S}'|$ decreases, $\mathrm{obj}_{\mathrm{descr}}$ and $\mathrm{obj}_{\mathrm{lscm}}$ are multiplied with increasingly larger factors. This favors solutions with higher integrated penalty energies to become the global minimum if the overlap is sufficiently large. It can already be seen, that there is a natural trade-off between low penalty energies and large overlap. However, finding a universal balance between both sides is not very intuitive in the above formulation.

Overlap Barrier

Provided with the insight that at least one parameter has to be involved to control this balance, we can come up with a different formulation. We now give this parameter a very simple meaning. Namely, the minimum amount of overlap we want to achieve. Thus, we further restrict the solution space to mappings providing an overlap larger than a prescribed ratio: $\frac{|\mathbf{S}'|}{|\mathbf{S}|} > r_o$. Again, we can employ a log-barrier function:

$$\mathrm{obj}_{\mathrm{overlap}} \quad = \quad \max\left(0, -\log\left(\frac{\frac{|\mathbf{S}'|}{|\mathbf{S}|} - r_o}{s_o - r_o}\right)\right). \tag{4.5}$$

As the difference $\frac{|\mathbf{S}'|}{|\mathbf{S}|} - r_o$ approaches 0, the barrier function grows infinitely. Negative values are again clipped to 0. The denominator $s_o - r_A$ defines the characteristic of the curve. Like in all our barriers function, $s_o \in (r_A, 1]$ moves the point at which the support of the function begins. If s_o is chosen as 1, the barrier is always active, pushing the solution towards larger overlap. If s_o is close to r_o, the barrier prevents further shrinkage of the overlap only in the last moment.

In this short section, we mentioned two possible ways, out of many, to relocate the global optimum. For both ones, failure cases can be easily imagined. On the one hand, there can be cases in which the global optimum still exhibits less overlap than the expected solution, because it avoids slightly higher descriptor or distortion penalties. On the other hand, giving too much weight to large overlap might prevent solutions in which slightly less overlap is in fact appreciated.

Therefore, an alternative approach is proposed in the next section. This approach does not try to cut off the undesired global optimum, but instead chooses an initialization that is close enough to a more desirable local optimum.

4.1.5 An Alternative Descriptor Objective

A general approach in non-linear optimization is to find a good initial guess of a solution and then descend into a close-by local minimum. We strive to find such an initialization by first solving a simpler problem. More precisely, we restrict ϕ to rigid transformations in 2D. This means that the source disk $\boldsymbol{f}(\mathbf{S})$ can be translated and rotated in $\mathbb{R}^2$ until a good alignment with the target disk $\boldsymbol{g}(\mathbf{T})$ is found. From this initialization, the non-linear deformation process is started.

Here, an alternative optimization target for this rigid 2D alignment is presented, which avoids the issues described in the last section.

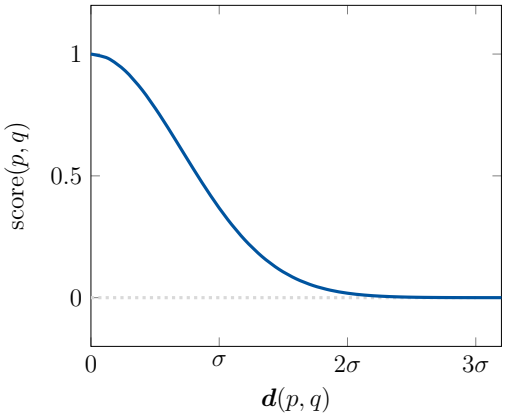
In this restricted setting, the formulation of an objective becomes significantly simpler. Fist of all, injectivity is not issue, as it cannot be violated by rigid transformations. In addition, we choose not to care about the distortion of the induced map $\mathbf{\Phi}$ yet. Therefore, the only remaining aspects are descriptor distances and the size of the overlap. We can now focus on merging these two concepts into a single objective without having to perform any balancing.

The core of the problem at hand is that descriptor distances for existing correspondences are penalized, but non-existing correspondences are not taken into account. In other words, they are penalized with distance 0, which clearly encourages fewer correspondences. Choosing a value other than 0 for non-existing correspondences brings us back to the above balancing problem.

Instead, we would like to invert our measure, and turn descriptor distances into a descriptor matching score. This score should be 0 if there is no correspon-

dence, close to zero if there is a bad correspondence, and approach 1 for a perfect correspondence. In this formulation, maximizing the matching score solves both problems at once: (1) points with similar descriptor values are aligned, because they increase the total score; and (2) the overlap does not become too small, because the total score can only increase in the overlapping region.

Unfortunately, the types of descriptor distances we want to allow are not bounded from above. This means, that we have to map the unbounded interval $[0, \infty)$ to the bounded interval $(0, 1]$. However, the problem of turning distance measures into similarity measures is well studied (see e.g. [VTS04]). One way to achieve such a conversion is to use a Gaussian kernel, as shown on the right:



$$\text{score}(p, q) = \exp\left(-\frac{\boldsymbol{d}(p, q)^2}{2\sigma^2}\right). \tag{4.6}$$

A descriptor distance $\boldsymbol{d}(p, q) = 0$, i.e. a perfect match, gives the maximum score of 1. If the distance increases, the similarity $\text{score}(p, q)$ decreases and approaches 0 in the limit $\boldsymbol{d}(p, q) \to \infty$. How quickly the function approaches 0 is controlled by the parameter σ. We choose σ to be the standard deviation of all points from both surfaces in the descriptor space, with respect to the distance function $\boldsymbol{d}$. The case of non-existing correspondences is added by defining:

$$\text{score}_{\boldsymbol{\Phi}}(p) = \begin{cases} \text{score}\left(p, \boldsymbol{\Phi}(p)\right) & p \in \mathbf{S}' \\ 0 & p \notin \mathbf{S}'. \end{cases}$$

Finally, we end up with the following very simple objective for the rigid initialization phase:

$$\text{obj}_{\text{init}} \quad = \quad \int_{\mathbf{S}} \text{score}_{\boldsymbol{\Phi}}(p)\, dA \quad \to \max. \tag{4.7}$$

Having found a good rigid transformation ϕ maximizing this objective, we can proceed by minimizing our original objective composed of $\text{obj}_{\text{descr}}$, obj_{lscm} and obj_{flip} using a non-linear deformation.

4.2 Discretization and Derivatives

So far, all expressions were derived in a continuous setting to allow for a higher level of abstraction. This section discusses the details of how to actually evaluate our objective function in a discrete setting. In Section 4.3, the objective will be optimized using a gradient descent based method. For this reason, first derivatives of the relevant terms ($\mathrm{obj}_{\mathrm{descr}}$, $\mathrm{obj}_{\mathrm{lscm}}$ and $\mathrm{obj}_{\mathrm{flip}}$) will also be discussed here.

Surface Meshes

Both surfaces are represented by triangle meshes $\mathbf{S} = (\mathbf{V_S}, \mathbf{E_S}, \mathbf{F_S})$ and $\mathbf{T} = (\mathbf{V_T}, \mathbf{E_T}, \mathbf{F_T})$, with sets of vertices $\mathbf{V}$, edges $\mathbf{E}$ and faces $\mathbf{F}$, respectively. The words triangles and faces are used interchangeably. Each mesh comes with an embedding into the three-dimensional Euclidean space, defined per vertex $v \in \mathbf{V}$ and linearly interpolated over edges $e \in \mathbf{E}$ and triangles $t \in \mathbf{F}$. For convenience, mesh elements are also addressed by v, e or $t \in \mathbf{S}$ or $\mathbf{T}$. Each triangle has a local orthonormal coordinate system with basis vectors $\mathbf{e}_u$ and $\mathbf{e}_v$. The positions of triangle corners a, b, $c \in \mathbf{V}$, expressed in a local coordinate system or in the plane, are denoted $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c} \in \mathbb{R}^2$. A surface point $\mathbf{p}$ lying within a triangle is expressed via its barycentric coordinates $\mathbf{p} = \alpha\mathbf{a}+\beta\mathbf{b}+\gamma\mathbf{c}$, where $\gamma = (1-\alpha-\beta)$. We enumerate the one-ring neighborhood of a vertex $a \in \mathbf{V}$ using the notation $(b, c) \in N(a)$, where a, b, c form the triangles incident to a in counterclockwise orientation. Alternatively, we address incident triangles directly using $t \in N(a)$.

The area of a triangle in its original 3D embedding is denoted A_t or A_{abc}. Furthermore, we define the vertex area A_v to be $1/3$ of the triangle area sum in the one-ring of v.

Map Representation

The mappings $\boldsymbol{f}$, $\boldsymbol{g}$ and $\boldsymbol{\phi}$ can be represented by simply assigning a pair of scalar coordinates (u, v) to each vertex of the respective mesh, i.e. we have $\boldsymbol{f} : \mathbf{V_S} \to \mathbb{R}^2$, $\boldsymbol{g} : \mathbf{V_T} \to \mathbb{R}^2$ and $\boldsymbol{\phi} : \mathbf{V_S} \to \mathbb{R}^2$. All these maps are piecewise linear, which means that a point within a triangle can be mapped using its barycentric coordinates, e.g. $\boldsymbol{f}(\mathbf{p}) = \alpha\boldsymbol{f}(a) + \beta\boldsymbol{f}(b) + \gamma\boldsymbol{f}(c)$.

Consequently, the inter-surface map $\boldsymbol{\Phi}$ can map barycentric coordinates in a triangle of $\mathbf{S}'$ to barycentric coordinates in a triangle of $\mathbf{T}'$. It is constructed by the composition of $\boldsymbol{f}$, $\boldsymbol{\phi}$ and $\boldsymbol{g}^{-1}$. More precisely, mapping a point $\mathbf{p}$ from $\mathbf{S}'$ to $\mathbf{T}'$ works as follows: (1) $\mathbf{p}$ is expressed via barycentric coordinates within its triangle (a, b, c) in $\mathbf{S}'$. (2) a, b, c are first mapped to the plane by $\boldsymbol{f}$. (3) The images of a, b, c and then relocated using $\boldsymbol{\phi}$. By evaluating the same barycentric coordinates here, the image of $\mathbf{p}$ is found as a point in $\mathbb{R}^2$. (4) A triangle lookup with respect

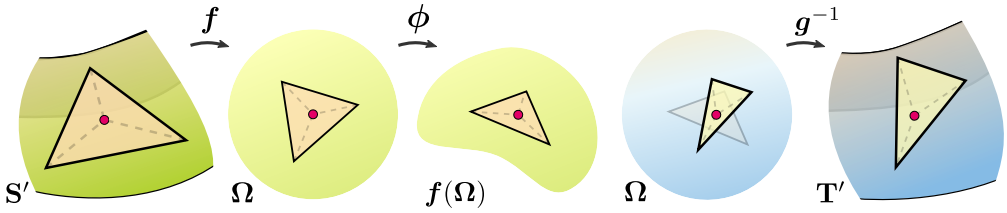


Figure 4.3: The piecewise linear map $\mathbf{\Phi}$ transfers points from the source to the target mesh via barycentric coordinates. (Green) When the vertices of a triangle are mapped to the parameter domain by $\boldsymbol{f}$ and relocated by ϕ, a point within the triangle keeps its barycentric coordinates. (Blue) To lift the point to the target surface, it first has to be re-expressed by its barycentric coordinates within a target triangle.

to the target mesh is performed in parameter space and the point is re-expressed by a different set of barycentric coordinates within this target triangle. (5) The newly acquired barycentric coordinates now also define the final 3D location of the mapped point $\mathbf{\Phi}(\mathbf{p})$ on $\mathbf{T}'$. The process is illustrated in Figure 4.3.

Finally, the map $\mathbf{\Phi}$ constructed this way is also piecewise linear, however not in the triangulation of $\mathbf{S}'$ but with respect to the mesh constructed by intersecting $\mathbf{S}'$ and $\mathbf{T}'$ in the parameter domain.

Acceleration Structure

The triangle lookup in step (4) is sped up using a 2D acceleration structure. Since the parametrization of the target mesh is bounded within $[-1, 1] \times [-1, 1]$, we can simply employ a uniform grid. Each grid cell of this grid contains a list of its intersecting triangles. When a query is made for a point $\mathbf{p}$, its corresponding grid cell is looked up and then linearly searched for the triangle $\mathbf{p}$ actually lies in.

Note that this structure has to be initialized just once, as it only depends on the constant map $\boldsymbol{g}$. We choose the number of grid cells proportional to the number of vertices in $\mathbf{T}$. Assuming a uniform distribution of vertices in parameter space, this enables constant-time lookups.

Variables

The discrete variables of our optimization problem are the pairs of (u, v) coordinates defining the map ϕ. Our variable vector $\mathbf{x}$ is simply the concatenation of all these coordinates, i.e.

$$\mathbf{x} = [u_0,\ v_0,\ \ldots\ u_{n-1},\ v_{n-1}]^T \in \mathbb{R}^{2n},$$

with $n = |\mathbf{V_S}|$.

During the optimization, we will have to evaluate $\mathrm{obj}(\mathbf{x})$ as well as its gradient with respect to $\mathbf{x}$. For a fixed $\mathbf{x}$, this gradient is a vector in the solution space $\mathbb{R}^{2n}$ and we refer to it by

$$\nabla \mathrm{obj} = \left[\frac{\partial}{\partial u_0}\mathrm{obj},\ \frac{\partial}{\partial v_0}\mathrm{obj},\ \ldots\ \frac{\partial}{\partial u_{n-1}}\mathrm{obj},\ \frac{\partial}{\partial v_{n-1}}\mathrm{obj}\right]^T \in \mathbb{R}^{2n}.$$

An intuitive interpretation is possible by regarding individual pairs of entries $[\frac{\partial}{\partial u}\mathrm{obj},\ \frac{\partial}{\partial v}\mathrm{obj}]^T \in \mathbb{R}^2$. Each pair represents a 2D vector per variable vertex in the parameter domain. The direction of this vector shows how to move the vertex in order to maximally increase the objective value.

4.2.1 Descriptor Distances

Before starting the optimization, a descriptor type and its parameters have to be chosen. Then, the descriptor values $\mathbf{I}(v)$ are precomputed for each vertex in both $\mathbf{S}$ and $\mathbf{T}$. Moreover, we are provided with a distance function $\boldsymbol{d}$, that can be defined in different ways, depending on the descriptor type (cf. Section 3.1).

This distance function can easily be evaluated between two vertices from $\mathbf{S}$ and $\mathbf{T}$ respectively. However, during the optimization, distances between a vertex from $\mathbf{S}$ and an arbitrary surface point on $\mathbf{T}$ have to be computed. Thus, the question is how to compute the distance $\boldsymbol{d}(v, \boldsymbol{\Phi}(v))$ where $\boldsymbol{\Phi}(v)$ is defined by barycentric coordinates.

One possible option is to linearly interpolate the descriptor values $\mathbf{I}(a)$, $\mathbf{I}(b)$ and $\mathbf{I}(c)$ within each triangle of $\mathbf{T}$. This way, $\mathbf{I}$ is densely available and distances can be computed for each point within a triangle. Finding a gradient of this distance within the triangle however is more difficult as it involves derivatives of the distance function. Since we wish to treat this function as a black box, we choose a different approach.

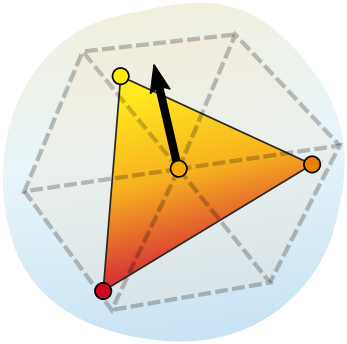


Figure 4.4: The descriptor distance between a source vertex (center) and three target vertices is linearly interpolated within a target triangle. The gradient with respect to the position of the source vertex is shown as a black arrow.

Considering a single vertex v of the source mesh, we can compute the distance to all vertices on the target mesh. These distances now form a scalar field on $\mathbf{T}$. Although for each vertex $v \in \mathbf{S}$ such a scalar field over the entire mesh $\mathbf{T}$ exists, we only have to consider it locally. We are solely interested in its values at the triangle corners corresponding to $\boldsymbol{\Phi}(v)$, i.e. $\boldsymbol{d}(v, a)$, $\boldsymbol{d}(v, b)$ and $\boldsymbol{d}(v, c)$, with $a, b, c \in \mathbf{V_T}$. These distances can now be linearly interpolated within the triangle (see Figure 4.4). The result is just a

different—however not equivalent—approximation than first interpolating the descriptor values and then computing distances. Fortunately, this new approximation comes with a very simple gradient. Because the distance field from v to a target triangle is linear, this gradient is constant within the triangle. Therefore, we define

$$\boldsymbol{d}(v, \boldsymbol{\Phi}(v)) \;=\; \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{d}(v,a) \\ \boldsymbol{d}(v,b) \\ \boldsymbol{d}(v,c) \end{bmatrix},$$

with α, β and γ being the barycentric coordinates of $\boldsymbol{\Phi}(v)$ in the target mesh. The gradient with respect to the unknown $\boldsymbol{\phi}(v)$ within the target triangle is then

$$\nabla \boldsymbol{d}\,(v, \boldsymbol{\Phi}(v)) \;=\; \frac{1}{2A_t} \begin{bmatrix} \mathbf{e}_a^{\perp} & \mathbf{e}_b^{\perp} & \mathbf{e}_c^{\perp} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}(v,a) \\ \boldsymbol{d}(v,b) \\ \boldsymbol{d}(v,c) \end{bmatrix} \in \mathbb{R}^2,$$

where $\mathbf{e}_a^{\perp}$, $\mathbf{e}_b^{\perp}$ and $\mathbf{e}_c^{\perp}$ are the edge vectors opposite to vertices a, b and c rotated by $90°$ counterclockwise (see e.g. [LPRM02]). Now our discrete equivalent of the descriptor objective in Equation 4.1 is

$$\mathrm{obj}_{\mathrm{descr}} \quad = \quad \sum_{v \in \mathbf{V}_{\mathbf{S}'}} \boldsymbol{d}\,(v, \boldsymbol{\Phi}(v)) \cdot A_v, \tag{4.8}$$

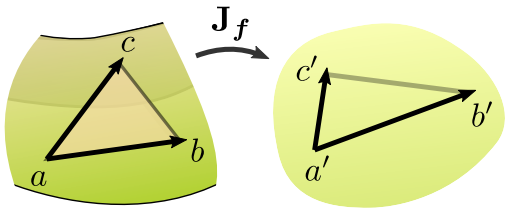
and its gradient

$$\nabla\, \mathrm{obj}_{\mathrm{descr}} \;=\; \begin{bmatrix} \nabla \boldsymbol{d}\,(v_0, \boldsymbol{\Phi}(v_0)) \cdot A_{v_0} \\ \vdots \\ \nabla \boldsymbol{d}\,(v_{n-1}, \boldsymbol{\Phi}(v_{n-1})) \cdot A_{v_{n-1}} \end{bmatrix} \in \mathbb{R}^{2n}. \tag{4.9}$$

The objective $\mathrm{obj}_{\mathrm{descr}}$ now is a piecewise linear, i.e. C^0 continuous, function in the variables of $\mathbf{x}$. This means that its gradient is constant per face, but discontinuous across the edges of $\mathbf{T}$. Working with a discontinuous gradient requires special handling in the gradient descent algorithm, which is explained in Section 4.3.2.

4.2.2 Map Distortion

Recall that the distortion of the map $\boldsymbol{\Phi}$ is measured in terms of its Jacobian $\mathbf{J}_{\boldsymbol{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}$. The individual Jacobians $\mathbf{J}_{\boldsymbol{f}}$, $\mathbf{J}_{\boldsymbol{g}}$ and $\mathbf{J}_{\boldsymbol{\phi}}$ can be directly computed per face: consider vertex positions $\mathbf{a} = (a_0, a_1)^T$, $\mathbf{b} = (b_0, b_1)^T$ and $\mathbf{c} = (c_0, c_1)^T$ being mapped to $\mathbf{a}' = (a_0', a_1')^T$, $\mathbf{b}' = (b_0', b_1')^T$, $\mathbf{c}' = (c_0', c_1')^T$ by either $\boldsymbol{f}$, $\boldsymbol{g}$ or $\boldsymbol{\phi}$. In the case of $\boldsymbol{\phi}$, all points are expressed in a global 2D coordinate system. For $\boldsymbol{f}$ and $\boldsymbol{g}$, the points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are defined in a local coordinate system per face of the respective mesh. We pick the point $\mathbf{a}$ as its origin and choose the basis vectors $\mathbf{e}_u = \frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}$ and $\mathbf{e}_v = \mathbf{n} \times \mathbf{e}_u$.



For each of the maps, the Jacobian $\mathbf{J}$ is constant per face and is represented by the 2×2 matrix mapping vectors to vectors. Due to its linearity, it is uniquely defined if two vectors and their images are given. For this, we choose the edge vectors $\mathbf{b}-\mathbf{a}$ and $\mathbf{c}-\mathbf{a}$ which are mapped to $\mathbf{b}'-\mathbf{a}'$ and $\mathbf{c}'-\mathbf{a}'$ (see inset for the case of $\mathbf{J}_f$). Per triangle we can now compute the Jacobian as:

$$\mathbf{J} \underbrace{\begin{bmatrix} b_0 - a_0 & c_0 - a_0 \\ b_1 - a_1 & c_1 - a_1 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} b'_0 - a'_0 & c'_0 - a'_0 \\ b'_1 - a'_1 & c'_1 - a'_1 \end{bmatrix}}_{M'}$$

$$\mathbf{J} = M' \cdot M^{-1}$$

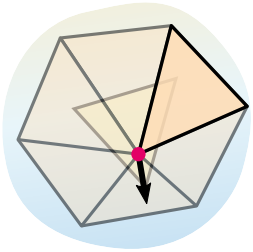
When composing the eventual Jacobian $\mathbf{J}_\Phi$, the matrices $\mathbf{J}_f$ and $\mathbf{J}_\phi$ can simply be multiplied. $\mathbf{J}_g^{-1}$ however poses an additional problem. While $\mathbf{J}_f$ and $\mathbf{J}_\phi$ are defined per triangle in $\mathbf{S}$, the Jacobian $\mathbf{J}_g^{-1}$ is defined per triangle in $\mathbf{T}$. Due to this incompatibility, we cannot evaluate $\mathbf{J}_\Phi$ per face in $\mathbf{S}'$. Computing it for a single point in $\mathbf{S}'$ however is always possible: $\mathbf{J}_f$ and $\mathbf{J}_\phi$ are defined by the face the point lies in and a triangle lookup with respect to the target mesh tells us which $\mathbf{J}_g^{-1}$ to use. This way, we can evaluate $\mathbf{J}_\Phi = \mathbf{J}_g^{-1} \cdot \mathbf{J}_\phi \cdot \mathbf{J}_f$ per point in $\mathbf{S}'$. Note that it is still piecewise constant, however not in the triangulation of $\mathbf{S}'$ or $\mathbf{T}'$ but in the intersection of both meshes in parameter space.

We refer to the entries of $\mathbf{J}_\Phi$ as

$$\mathbf{J}_\Phi = \begin{bmatrix} \frac{\partial \mathbf{\Phi}}{\partial u} & \frac{\partial \mathbf{\Phi}}{\partial v} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{00} & \mathbf{J}_{01} \\ \mathbf{J}_{10} & \mathbf{J}_{11} \end{bmatrix}.$$

Now, for a single triangle $t = (abc)$ in $\mathbf{S}'$, we express the least-squares conformal maps term with respect to its corner vertex a as:

$$\begin{aligned} \mathrm{lscm}_\Phi(abc) &= \left\| \frac{\partial \mathbf{\Phi}}{\partial v} - \mathrm{Rot}_{90} \frac{\partial \mathbf{\Phi}}{\partial u} \right\|^2 = \left\| \begin{bmatrix} \mathbf{J}_{01} + \mathbf{J}_{10} \\ \mathbf{J}_{11} - \mathbf{J}_{00} \end{bmatrix} \right\|^2 \\ &= (\mathbf{J}_{01} + \mathbf{J}_{10})^2 + (\mathbf{J}_{11} - \mathbf{J}_{00})^2. \end{aligned}$$



Then, to form its gradient, we derive $\mathrm{lscm}_{\mathbf{\Phi}}(abc)$ with respect to the location of the triangle corner $\mathbf{a}'$ in parameter space. In other words, we answer the question: how do we have to move the point $\mathbf{a}' = \boldsymbol{\phi}(\boldsymbol{f}(a))$ in order to maximally increase the distortion. The result is a vector $\frac{\partial}{\partial \mathbf{a}'} \mathrm{lscm}_{\mathbf{\Phi}} \in \mathbb{R}^2$ for each triangle corner in $\mathbf{S}'$ (see inset).

Furthermore, we answer the same question for all corners in $\mathbf{S} \setminus \mathbf{S}'$, yielding vectors $\frac{\partial}{\partial \mathbf{a}'} \mathrm{lscm}_{\boldsymbol{\phi}} \in \mathbb{R}^2$. For simplicity, we refer to either of them by $\nabla \mathrm{lscm}$. The derivation of $\nabla \mathrm{lscm}$ can be found in Appendix A.

Finally, we take a look at the total discrete distortion energy expressed as a sum over all triangle corners a:

$$\mathrm{obj}_{\mathrm{lscm}} \quad = \quad \frac{1}{3} \sum_{a \in \mathbf{S}'} \sum_{(b,c) \in N(a)} \mathrm{lscm}(abc) \cdot A_{abc}. \tag{4.10}$$

The term is divided by 3 because each triangle is enumerated three times. In this notation, it can be seen that each vertex is influenced by the distortion of its incident triangles. This makes it straightforward to derive the gradient of $\mathrm{obj}_{\mathrm{lscm}}$:

$$\nabla \mathrm{obj}_{\mathrm{lscm}} \; = \; \frac{1}{3} \begin{bmatrix} \sum_{(b,c) \in N(a_0)} \nabla \mathrm{lscm}(a_0 bc) \cdot A_{a_0 bc} \\ \vdots \\ \sum_{(b,c) \in N(a_{n-1})} \nabla \mathrm{lscm}(a_{n-1} bc) \cdot A_{a_{n-1} bc} \end{bmatrix} \in \mathbb{R}^{2n}. \tag{4.11}$$

Unfortunately the discrete distortion energy exhibits C^0 discontinuities in the variables of $\mathbf{x}$. This is due to the piecewise constant discretization of $\mathbf{J}_{\boldsymbol{g}}^{-1}$ and is discussed in more detail in Section 4.3.2.

4.2.3 Local Injectivity

Deriving the barrier function in Equation 4.3 is somewhat simpler, as it is independent of the maps $\boldsymbol{f}$ and $\boldsymbol{g}$ and thus avoids discretization artifacts. The discrete version of $\mathrm{obj}_{\mathrm{flip}}$ is:

$$\mathrm{obj}_{\mathrm{flip}} \quad = \quad \sum_{v \in \mathbf{S}} \max\left(0, \underbrace{-\log\left(\det \mathbf{J}_{\boldsymbol{\phi}} / s_f\right)}_{=:B}\right) \cdot A_v \tag{4.12}$$

To find its gradient we again consider a variable triangle corner a and fixed corners b and c. As before, the 2D gradient vector $\nabla \mathrm{flip}(abc) = \frac{\partial}{\partial \mathbf{a}'} \mathrm{obj}_{\mathrm{flip}}$ tells us how to move the variable vertex $\mathbf{a}'$ to increase the barrier term as quickly as possible.

Using $\nabla \operatorname{flip}(abc)$, which is derived in Appendix B, the gradient of the entire function $\mathrm{obj}_{\text{flip}}$ with respect to the variable vector $\mathbf{x}$ is then:

$$\nabla \mathrm{obj}_{\text{flip}} = \frac{1}{3} \begin{bmatrix} \sum_{(b,c)\in N(a_0)} \nabla \operatorname{flip}(a_0bc) \cdot A_{a_0bc} \\ \vdots \\ \sum_{(b,c)\in N(a_{n-1})} \nabla \operatorname{flip}(a_{n-1}bc) \cdot A_{a_{n-1}bc} \end{bmatrix} \in \mathbb{R}^{2n}. \tag{4.13}$$

The objective $\mathrm{obj}_{\text{flip}}$ is C^0 continuous within the feasible region. Its gradient has discontinuities at points where the support of a barrier starts, i.e. wherever $\det \mathbf{J}_{\boldsymbol{\phi}}/s_f = 1$ in a triangle. Otherwise, both the function and its gradient are smooth.

We skip the discretization of the convexity barrier 4.4, as well as the overlap barrier 4.5, since they only serve experimental purposes and are not part of the final optimization target.

Initialization Objective

The alternative descriptor objective $\mathrm{obj}_{\text{init}}$ is discretized analogously to $\mathrm{obj}_{\text{descr}}$: For each vertex $v \in \mathbf{S}$, the matching score with respect to the vertices of the intersecting target triangle is computed and then linearly interpolated. Hence, the discrete version of Equation 4.7 is:

$$\mathrm{obj}_{\text{init}} = \sum_{v\in\mathbf{S}} \operatorname{score}_{\boldsymbol{\Phi}}(v) \cdot A_v \tag{4.14}$$

Since our initialization method will not require derivatives, we do not have to evaluate its gradient.

Finally, we are equipped with all individual functions and derivatives to solve the optimization problem. These can now be composed into the eventual objective function:

$$\mathrm{obj} = \alpha_d \cdot \mathrm{obj}_{\text{descr}} + \alpha_l \cdot \mathrm{obj}_{\text{lscm}} + \alpha_f \cdot \mathrm{obj}_{\text{flip}}\,. \tag{4.15}$$

Analogously, its gradient is:

$$\nabla \mathrm{obj} = \alpha_d \cdot \nabla \mathrm{obj}_{\text{descr}} + \alpha_l \cdot \nabla \mathrm{obj}_{\text{lscm}} + \alpha_f \cdot \nabla \mathrm{obj}_{\text{flip}}\,. \tag{4.16}$$

The weights for each term are discussed during the evaluation in Section 4.4.

4.3 Optimization

The variable vector $\mathbf{x} \in \mathbb{R}^{2n}$ in our optimization problem is a discrete sampling of the continuous map $\boldsymbol{\phi}$. It is the concatenation of all vertex positions $(u, v)^T$ of $\mathbf{S}$ in the plane after the deformation. Finding a good assignment to $\mathbf{x}$ is split into two steps: First, an initial rigid transformation is found using a randomized approach. This phase seeks to maximize the alternative descriptor objective $\mathrm{obj}_{\mathrm{init}}$. After that, a gradient descent based method is employed to find a close-by local minimum of the non-linear, non-convex function obj.

4.3.1 Initialization

As stated above, a rigid 2D transformation $\boldsymbol{\phi}$ will serve as initialization. This transformation has three continuous degrees of freedom, namely a rotation θ and a 2D translation $\mathbf{t} = (t_0, t_1)^T$. We restrict t_0 and t_1 to the range $[-2, 2]$ to include all possible translations with non-empty overlap.

Furthermore, it is beneficial for our application if the inter-surface map supports reflections. This makes it possible to, for example, map a left human eye to a right one. Since there is no mechanism allowing reflections during the following optimization, this additional binary degree of freedom has to be completely resolved during the initialization phase.

Due to the low-dimensional and bounded solution space $\{0, 1\} \times [0, 2\pi) \times [-2, 2] \times [-2, 2]$, a very simple brute force approach is practical.

Random Sampling

We draw k random samples (r, θ, t_0, t_1) from this space, evaluate $\mathrm{obj}_{\mathrm{init}}$ for each one, and keep the solution with the highest matching score. Probabilities for each of the four dimensions are distributed as follows:

One half of the samples receive a reflection across an arbitrarily rotated axis. Note, that an implementation using oriented meshes (e.g. a half-edge data structure) requires flipping the orientation of elements. The rotation θ is sampled with uniform probability. As we are more interested in solutions with high overlap, the translation $\mathbf{t}$ follows a normal distribution centered around the origin. Its parameter σ is chosen as 1, such that a sufficiently high sampling probability ($\approx 5\%$) is still available at the boundary of the range.

Unless stated otherwise, $k = 10\,000$ is chosen for all our experiments. Furthermore, $\mathrm{obj}_{\mathrm{init}}$ is evaluated in parallel and a constant seed is used for reproducibility.

The initializations obtained this way appear to be sufficiently good starting points for a gradient descent. More thought-out initialization methods could be subject to future work.

4.3.2 Non-Linear Optimization

The objective we seek to minimize is a non-linear, non-convex function. Given an initial solution, such problems are commonly optimized by Newton methods. These use a quadratic approximation around the current point $\mathbf{x}$ to determine a descent direction in each iteration. Known for their fast convergence, Newton methods require the Hessian matrix of the objective in each step.

For now, we avoid the effort of computing the Hessian by retreating to a gradient descent method, trading the second-order approximation of obj for a linear one. This ease usually comes at the cost of slow convergence. Nonetheless, for our experiments, we choose simplicity over speed and leave performance improvements for future work (see Section 5.1.3). In addition, only parts of our objective function would benefit from a second-order method, as $\mathrm{obj}_{\mathrm{descr}}$ is a piecewise linear function, i.e. its Hessian is 0. Moreover, the nature of obj will pose additional challenges, which can be sufficiently demonstrated without the added complexity of second derivatives.

Gradient Descent

Given a continuously differentiable (C^1) function f and an initial solution $\mathbf{x}_0$, the *gradient descent* method iteratively converges to a close-by local minimum $\mathbf{x}^*$. In each iteration the method takes a step into the direction of the negative gradient, as this is the direction minimizing a first-order approximation of f (cf. [BV04]):

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \lambda^k \cdot \nabla f(\mathbf{x}^k).$$

The parameter $\lambda^k > 0$ controls the step size and is independently chosen in each iteration after computing the gradient. Given a fixed direction, ∇f, the problem of finding the best step size is a one-dimensional problem in λ. Instead of exactly determining the optimal λ one usually settles with a step size providing some decrease in the objective value. One can argue that the effort for finding the optimal λ in each step can instead be invested into updating the search direction more often.

A very popular and simple approach to find a step size with sufficient improvement is a *backtracking line search*. Starting from an initial large value, the step size is decreased exponentially until a criterion is fulfilled. The commonly used Armijo-Goldstein condition checks whether the actual improvement

is within a factor of the expected improvement considering the first-order approximation of f:

$$\underbrace{f(\mathbf{x}^k) - f(\mathbf{x}^{k+1})}_{\text{actual improvment}} \quad \geq \quad \alpha \cdot \underbrace{\lambda^k \cdot \left\| \nabla f(\mathbf{x}^k) \right\|^2}_{\text{expected improvement}}.$$

If this criterion is fulfilled, λ is accepted. Otherwise it is decreased by a factor β, i.e. $\lambda := \beta \cdot \lambda$. Due to the continuity of ∇f, this condition can always be satisfied for a sufficiently small λ. The parameters are usually chosen within $\alpha \in (0, 0.5), \beta \in (0, 1)$.

We use an initial λ in each iteration that has an interpretation in our optimization domain $\mathbb{R}^2$. Namely, λ is chosen such that the longest 2D update vector of a point $(u, v)^T$ is scaled down to the mean edge length of $\boldsymbol{g}(T)$. If the longest update vector is already shorter, $\lambda = 1$ is used.

Challenges

There are several challenges arising from our current formulation of obj, which prevent this method from being immediately successful. First of all, the precondition for the objective function, i.e. being continuously differentiable, is violated: (1) since the descriptor distance fields are piecewise linear, $\nabla \mathrm{obj}_{\mathrm{descr}}$ is piecewise constant, i.e. C^1 discontinuous. (2) The least-squares conformal map term is even C^0 discontinuous in our setting. (3) Beyond that, there are cases in which the backtracking line search gets stuck in the proximity of barriers.

In the following, those three issues will be discussed in detail. It is shown how to circumvent (1) by handling two simple special cases. In contrast to that, (2) poses more severe problems which are worked around by a crude heuristic, succeeding in some cases. Finally, (3) can be tempered by parameter adjustments, and proper handling is left for future work.

First-Order Discontinuities

In this section $\mathrm{obj}_{\mathrm{descr}}$ is analyzed and its problems are handled in isolation, i.e. all other terms of the objective are temporarily disabled. Recall that for a single vertex $v \in \mathbf{S}$, the objective describes the scalar field of descriptor distances to potential correspondences in $\mathbf{T}$. This can be intuitively visualized in the parameter domain. When moving around the image of v, i.e. $\boldsymbol{\phi}(v)$, in $\mathbb{R}^2$, the descriptor distance to the underlying point in the parametrization of $\mathbf{T}$ changes. These distances can be imagined as a height field over $\boldsymbol{g}(\mathbf{T})$. A perfect correspondence for v has height 0 and regions with bad matches form high "mountains". While the height field stays fixed, v can move freely and we expect it to to descent into a close-by "valley" (as long as all other energies are disabled).

In our discretization however, this height field is piecewise linear (see Figure 4.5). This means that the vertices of $\boldsymbol{g}(\mathbf{T})$ receive a certain height, which is linearly interpolated over its planar triangles. The first consequence of this is that local minima are always obtained at vertices of $\boldsymbol{g}(\mathbf{T})$. Excluding the case where two adjacent vertices have the same height and a local minimum extends along their common edge, we thus expect all source vertices to exactly move to a target vertex.

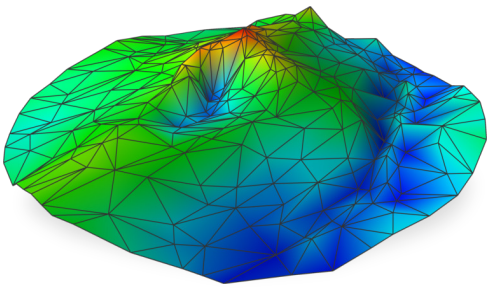
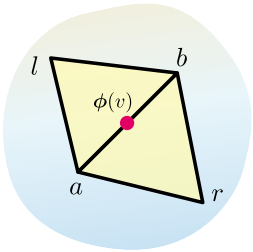


Figure 4.5: For a single vertex in $\mathbf{S}$, its descriptor distance to the target disk can be visualized as a piecewise linear height field.

Another consequence of the piecewise linear height field is that its gradient is constant per triangle. In particular, it does not approach the zero vector at a local minimum. Imagine the source vertex v lying exactly over a minimizing target vertex w. Now v is arbitrarily assigned to one of the incident target triangles and receives the gradient of this triangle, pointing towards w. Since v is already in its optimal location and any movement would increase the objective, the backtracking line search will shrink the global step size λ infinitely. Consequently, the optimization gets stuck at this point although other source vertices might not be in their optimal location yet. Note that this deadlock might not immediately occur if only a few vertices are converged: due to the global nature of the objective, it is still possible that a large improvement at some other vertices compensates the setback of an already converged vertex. If this is the case, the supposedly converged vertex v often jitters around its optimal location. When however a majority of vertices converged, the optimization gets stuck.

A similar problem occurs if v is located at an edge of $\boldsymbol{g}(\mathbf{T})$ with gradients of the incident triangles pointing towards the edge. Depending on the slope of the edge, this leads to extreme oscillation of v or another deadlock.

Both problems however can be solved by handling them as special cases and adjusting the gradient accordingly: If $\boldsymbol{\phi}(v)$ is within distance ε of a vertex w in $\boldsymbol{g}(\mathbf{T})$ the *vertex case* is used. If this does not apply and $\boldsymbol{\phi}(v)$ is within distance ε to an edge, the *edge case* is employed. Otherwise, the gradient for v is computed as before, which we refer to as the *triangle case*.



- **Edge Case:** To compute the gradient of source vertex v lying on a target edge (a, b), we also consider the opposite target vertices l and r (see inset). If the minimum height among a, b, l, r is obtained at either a or b, the prob-

lematic situation can occur. However, it is clear that the maximum decrease in $\mathrm{obj}_{\mathrm{descr}}$ is obtained by moving $\boldsymbol{\phi}(v)$ along the edge (a, b). Thus we compute the descriptor gradient with respect to the edge instead of a triangle:

$$\nabla \boldsymbol{d}(v, \boldsymbol{\Phi}(v)) \;=\; \underbrace{\frac{\mathbf{e}}{\|\mathbf{e}\|}}_{\text{direction}} \cdot \underbrace{\frac{\boldsymbol{d}(v,b) - \boldsymbol{d}(v,a)}{\|\mathbf{e}\|}}_{\text{magnitude}},$$

where $\mathbf{e} = \boldsymbol{g}(b) - \boldsymbol{g}(a)$ is the edge vector in $\mathbb{R}^2$. This prevents oscillation if the edge exhibits some slope in the height field. If $\boldsymbol{d}(v, a) = \boldsymbol{d}(v, b)$, i.e. no slope, the gradient is the zero vector and thus does not provoke a deadlock.

In contrast, if the minimum height among a, b, l, r is obtained at l or r, the vertex should not move parallel to the edge and we apply the triangle case with respect the triangle of l or r, depending on which value is lower.

- **Vertex Case:** When the source vertex v is located very close to a target vertex w, and w is a local minimum, we simply set $\nabla \boldsymbol{d}(v, w) = 0$. This fixes the vertex sufficiently close to its desired position and prevents the deadlock, as it does not impact the line search anymore. If any neighbor of w has a lower height, we assign v to the corresponding edge and apply the edge case, as this gives the maximal decrease in $\mathrm{obj}_{\mathrm{descr}}$.

Using these two special cases, optimizing solely for $\mathrm{obj}_{\mathrm{descr}}$ using the gradient descent method explained above works as expected. Source vertices approach their minimizing target vertices within an error of ε and only stay inside a triangle or on an edge if the local minimum is not unique. Of course the so-obtained solution exhibits extreme distortion an numerous triangle flips. Therefore in the next paragraph, $\mathrm{obj}_{\mathrm{lscm}}$ is investigated more closely.

Zero-Order Discontinuities

While we can now handle piecewise linear objective functions, the conformal distortion term $\mathrm{obj}_{\mathrm{lscm}}$ adds more difficult problems. In contrast to $\mathrm{obj}_{\mathrm{descr}}$ it is piecewise quadratic and not only C^1 discontinuous but also C^0 discontinuous across the edges of $\boldsymbol{g}(\mathbf{T})$.

At first, this might be surprising, since the original least-squares conformal maps energy, used for surface parametrization, is a convex quadratic function. In our setting however, this energy is based on the composition of Jacobians $\mathbf{J}_{\boldsymbol{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}$. For a single vertex v, the matrix $\mathbf{J}_{\boldsymbol{f}}$ is constant because it solely depends on $\boldsymbol{f}$, which does not change. $\mathbf{J}_{\boldsymbol{\phi}}$ changes continuously as $\boldsymbol{\phi}(v)$ is moved around in the plane. Thus, just considering these two terms, $\mathrm{obj}_{\mathrm{lscm}}$ is a quadratic function as expected. The Jacobian $\boldsymbol{g}^{-1}$ however is constant per triangle in $\boldsymbol{g}(\mathbf{T})$

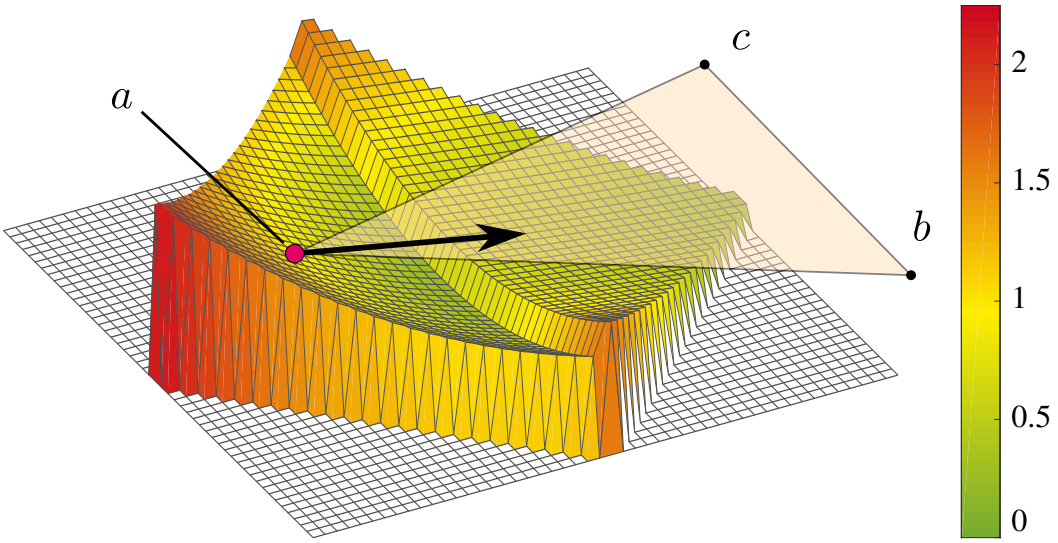


Figure 4.6: Two neighboring triangles in $\boldsymbol{g}(\mathbf{T})$ have different Jacobians $\mathbf{J}_{\boldsymbol{g}} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 3 \end{smallmatrix}\right)$ (left) and $\mathbf{J}_{\boldsymbol{g}} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ (right). The height field shows the conformality term $\mathrm{lscm}(abc)$ with respect to the position of a source triangle corner a. The field is discontinuous across the edge and exhibits an undesired local minimum. The negative conformality gradient with respect to a is indicated by a bold arrow.

and jumps when $\boldsymbol{\phi}(v)$ crosses an edge of the target mesh. As a result, $\mathrm{obj}_{\mathrm{lscm}}$ is a convex quadratic function in each target triangle but discontinuous across edges.

Unfortunately, these discontinuities introduce numerous undesired local minima and force the line search to fail. In the example of Figure 4.6, a line search along the depicted gradient direction is likely to back-off from the "cliff" in the height field. In our experiments, the optimization always stops within the first few gradient descent iterations.

Encouraged by the observation that the gradient $\nabla \mathrm{obj}_{\mathrm{lscm}}$ often roughly points into the desired direction, we employ the following crude modification of the algorithm: Instead of choosing a step size using backtracking line search, we simply perform steps of a small but fixed length into the negative gradient direction. This way, discontinuities in the objective function are simply jumped over. Only if a step would cross a barrier, its length is shrunk, using the same exponential back-off as before, until the resulting point $\mathbf{x}^{k+1}$ is back inside the feasible region.

The scaling parameter λ is chosen similarly to the initial value for the backtracking line search above. Namely such that the longest update vector of a vertex in $\mathbb{R}^2$ is scaled down to a factor $\bar{\lambda}$ of the mean edge length of $\boldsymbol{g}(\mathbf{T})$. If the longest update vector is already shorter that the desired length, it is not scaled up.

Depending on the nature of $\boldsymbol{g}$, this blunt approach works more or less well. Another indication that the heuristic might be reasonable is the following: If $\boldsymbol{g}$ itself is conformal, the effect of $\mathbf{J}_{\boldsymbol{g}}^{-1}$ on the gradient vector as depicted in Figure 4.6 is just a scaling (see Equations A.1, A.2 and A.3 in Appendix A). If this is the case, the gradient vector will only jump discontinuously in its length but not in its direction when crossing the edge. As a result, walking small steps into this direction should move the solution towards the desired local minimum. The less conformal $\boldsymbol{g}$ is, the less this assumption holds.

Another problem, that is worked around by this heuristic, constitutes in vertices entering or leaving the overlapping region. Both $\mathrm{obj}_{\mathrm{descr}}$ and $\mathrm{obj}_{\mathrm{lscm}}$ behave discontinuously in this case. For example, $\mathrm{obj}_{\mathrm{descr}}$ jumps from zero to a positive value when a vertex enters the overlap. In our approach, these discontinuities are simply walked over, giving a (temporary) increase in the objective value.

Evidently, our heuristic has several obvious disadvantages. First of all, the already slow convergence rate of the gradient descent is further decreased, as we have to choose an extreamly conservative step size. Secondly, the approach is no longer a descent method, since the fixed step size might overshoot the desired location and instead increase the total objective. In fact, we observe extreme oscillation in both the objective value and the gradient direction (demonstrated later in Figure 4.8). Moreover, the gradient does not approach zero length upon "convergence". Instead, the solution tends to oscillate around a local minimum. For this reason we experiment with an alternative stopping criterion: We keep track of the accumulated update of the last m iterations and stop if this smoothed-out vector falls below a length threshold. Finally, all guarantees known for gradient descent methods are lost. Whether the stopping criterion is reached or whether the obtained result is anywhere close to a local minimum is purely heuristic.

Nonetheless, we obtain good results in a set of examples. Despite being far from ideal, this approach allows us to investigate the overall behavior of our problem formulation and sufficiently evaluate other aspects. Finding a more solid optimization method for this kind of objective function or, more desirable, a reformulation without the discussed problems could be an important aspect of future work.

Optimization Trapped at Barriers

Another issue emerges from the barrier functions with which we restrict the feasible region of the problem. Recall that we use a negative logarithm that rapidly increases if a triangle degenerates and is about to flip. In our experiments, the optimization sometimes stops in an almost degenerated configuration although the desired local minimum is clearly not reached.

Imagine a situation in which the current optimization variable is a point very close to the boundary of the feasible region. This means that a triangle threatens to flip, in which case a barrier function would approach infinity. Further, imagine a strong gradient vector, of e.g. the descriptor objective, crossing the barrier. Since this update is not allowed, an exponential back-off is performed until the updated point lies within the feasible region. If the current point is very close to the boundary, the step size might have to be shrunk extremely. To escape this situation, the gradient vector of the barrier function is supposed to push the point away from the boundary. However, due to the nature of the logarithm, this gradient is rather weak until the point is extremely close to the barrier. In other words, there is only a very tiny range near the boundary in which the barrier gradient is strong enough to push a point away from it. Because the backtracking algorithm performs discrete steps, it is possible to miss this range. As a result, it happens that the point stays in its approximate location and the exponential back-off shrinks the step size to a value close to zero in each of the following iterations. Depending on the tessellation of the source mesh, this situation occurs quite often in our experiments and usually the optimization does not recover from it within a reasonable number of iterations.

Such issues in the presence of barrier functions frequently appear in the literature. For example, [MW94] mention that "a premature move too close to the boundary can lead to a long sequence of iterates 'trapped' near the singularity". Consequently, they develop more elaborate line search strategies. Among many others, the authors of [CMI09] also state that simple approaches to back-off from barriers do not guarantee convergence, and they propose another line search method.

How likely this issue is to occur in our setting, is influenced by the weight α_f. This parameter scales the values of the barrier function and thus also scales its gradient. Therefore, increasing α_f can ease the problem, as it strengthens the force pushing points away from the barrier.

A popular family of algorithms to robustly optimize problems with barrier functions are *interior point methods* (cf. [Ter13]). These solve a sequence of problems: Starting with a high factor α_f a solution is found using a standard (e.g. Newton) method. After convergence, α_f is iteratively decreased and the solution is updated using the same method in each step. Considering barrier functions that are not truncated as in our case, but have infinite support, the solution of the first problem is expected to be far away from all barriers, i.e. in the *interior* of the feasible region. In each of the following problems, the weight of the barrier terms decreases and the solution can move closer to the boundary. The sequence of solutions obtained this way is called the *central path*.

Unfortunately, an interior point method is not directly applicable to our formulation of the problem. For example, the simple approach explained above is

designed for convex optimization and assumes that a unique minimum exists for each value of α_f [BV04]. This however is not given in our case. Moreover, using a very high α_f and barrier functions with non-local support, the source disk will grow infinitely in $\mathbb{R}^2$. This is why we use truncated functions in the first place.

Within the scope of this thesis, successful experiments are conducted by manually adjusting the factor α_f and the truncation point s_f. Overcoming these issues however should be an important aspect of future work (see Section 5.1.3).

In conclusion, this section shows a first approach towards optimizing obj. For simplicity, a gradient method without seconds derivatives is used. Three challenges are understood an discussed in detail. Although our approach cannot robustly solve all problem instances yet, it provides data for the following evaluation.

4.4 Evaluation

This final section of the chapter gives an experimental evaluation of the approach introduced so far. It will, on the one hand, show successful applications of the method and, on the other hand, discuss its current shortcomings.

We applied our mapping technique to pairs of patches within the same class of the datset described in Section 3.3. The resulting inter-surface maps are visualized by transferring a set of polygonal lines from the source to the target patch. These lines represent prescribed directions or feature lines of a macro constraint.

Figure 4.7 shows snapshots of the optimization process in both the surface and the parameter domain. Eventually, the lines defined on the source patch are successfully mapped to geometrically similar locations on the target patch under low distortion.

Parameter Choice

The parameters of our method are tuned experimentally to achieve the best results. In fact, the method is sensitive to these parameters and can fail if they are chosen carelessly.

First, the initial transformation is found by sampling $k = 10\,000$ random solutions, unless stated otherwise. In the following, we fix the weight of the descriptor objective to $\alpha_d = 1$. Thereon, in most cases a good balance is achieved by weighting the conformalitiy term with $\alpha_c = 10$. In some experiments this value is adjusted to avoid certain local minima. Choosing the factor α_f for the anti-flip barrier term is a delicate issue as explained in Section 4.3.2. While in Figure 4.7 a value of $\alpha_f = 10$ suffices, the optimization gets stuck and does not

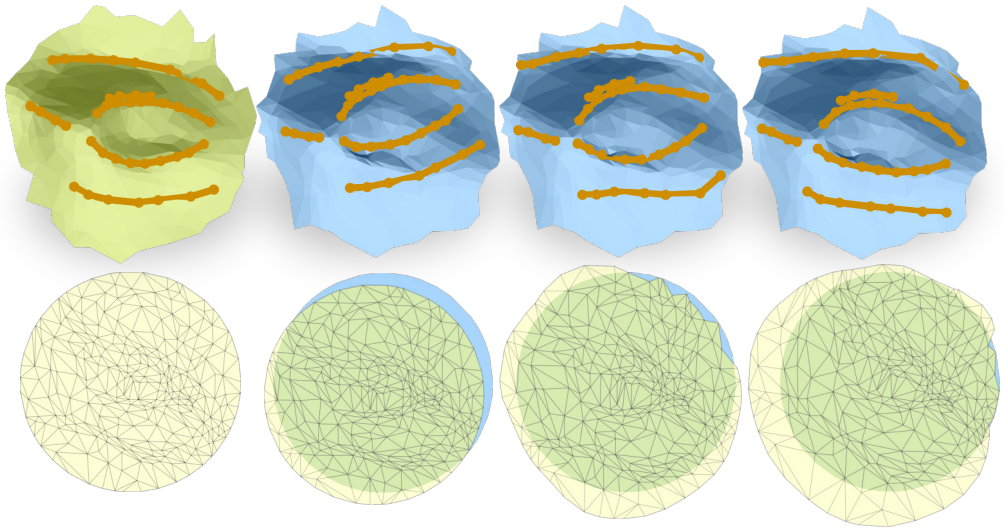


Figure 4.7: A set of polygonal lines is transferred from the source patch (green, far left) to the target patch (blue). A rigid alignment in parameter space is found by sampling 500 random transformations (mid left). The non-linear deformation of the source disk, together with the resulting constraint mapping, is shown after 2 000 (mid right) and 15 000 iterations (far right).

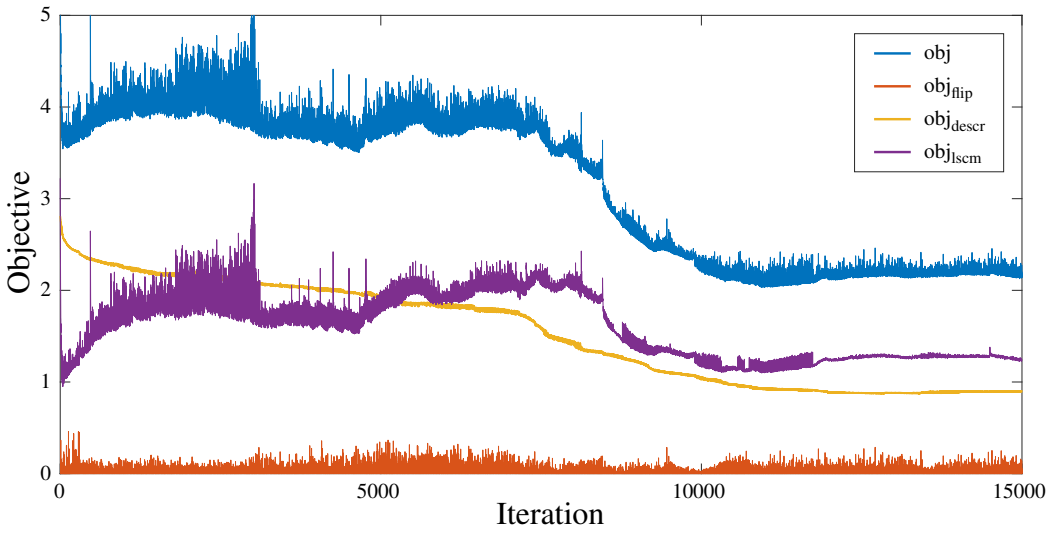


Figure 4.8: Due to our optimization heuristic, the objective value is subject to strong oscillation. This plot shows the individual parts of the objective function over the course of the example in Figure 4.7. While the descriptor objective decreases close to monotonically, the conformal distortion behaves less stable and, in addition, passes multiple local minima. The local injectivity term poses only a mild influence on the total objective and is not witnessed to take extreme values.

escape the situation in other examples. The problem can be diminished by choosing $\alpha_f = 10\,000$. The starting point of the barrier's support is set to $s_f = 0.5$ in all experiments.

Furthermore, the step size of the gradient method is chosen by $\bar{\lambda} = 0.05$. This means that the distance by which a vertex can move in $\mathbb{R}^2$ is limited to 0.05 times the mean edge length. Although leading to slow convergence, this small value is necessary to suppress too extreme oscillation of the objective value as well as a large number of back-off iterations from barriers. Due to the still remarkably strong oscillation of the objective value, we fail to detect convergence by both traditional measures and the stopping criterion proposed in Section 4.3.2. Instead, we perform a fixed number of up to $15\,000$ iterations each example.

Descriptor Choice

We obtained the best results by simply choosing the mean curvature at $\sigma = 0.1$ as our descriptor, and keep this choice for all examples presented here. Maybe surprisingly, results did neither improve by choosing more complicated combinations of different curvature types, nor by using the wave kernel signature.

An important property of a descriptor field for the success of our method is its smoothness. If the descriptor reflects too much high-frequency detail, the optimization quickly gets stuck in a local minimum. The mean curvature at a relatively large integration kernel of $\sigma = 0.1$ shows to fulfill this smoothness requirement, while smaller values do not. A disadvantage of this descriptor is its poor discrimination due to being almost strictly local. E.g. each fingertip of a human hand exhibits similar descriptor values and the optimization depends on a high-quality initialization (see Figure 4.9, top). In contrast, the wave kernel signature is able to distinguish between the individual fingertips (see see Figure 4.9, bottom). However its gradient field lacks sufficient smoothness to properly guide the optimization.

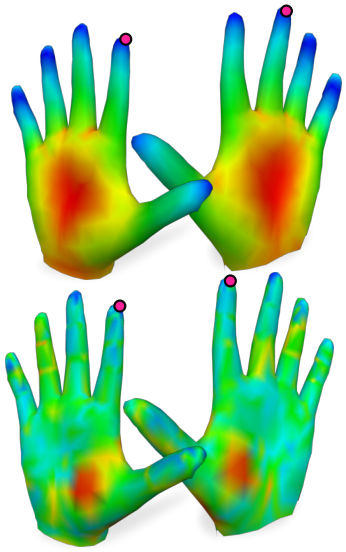


Figure 4.9: The same point on the tip of the index finger is picked in two examples. Using the mean curvature descriptor (top row) and the wave kernel signature (bottom row), descriptor distances are shown as a heat map. While the nearest neighbor with respect to mean curvature is highly ambiguous and located on the wrong finger, the wave kernel signatures yields the correct correspondence.

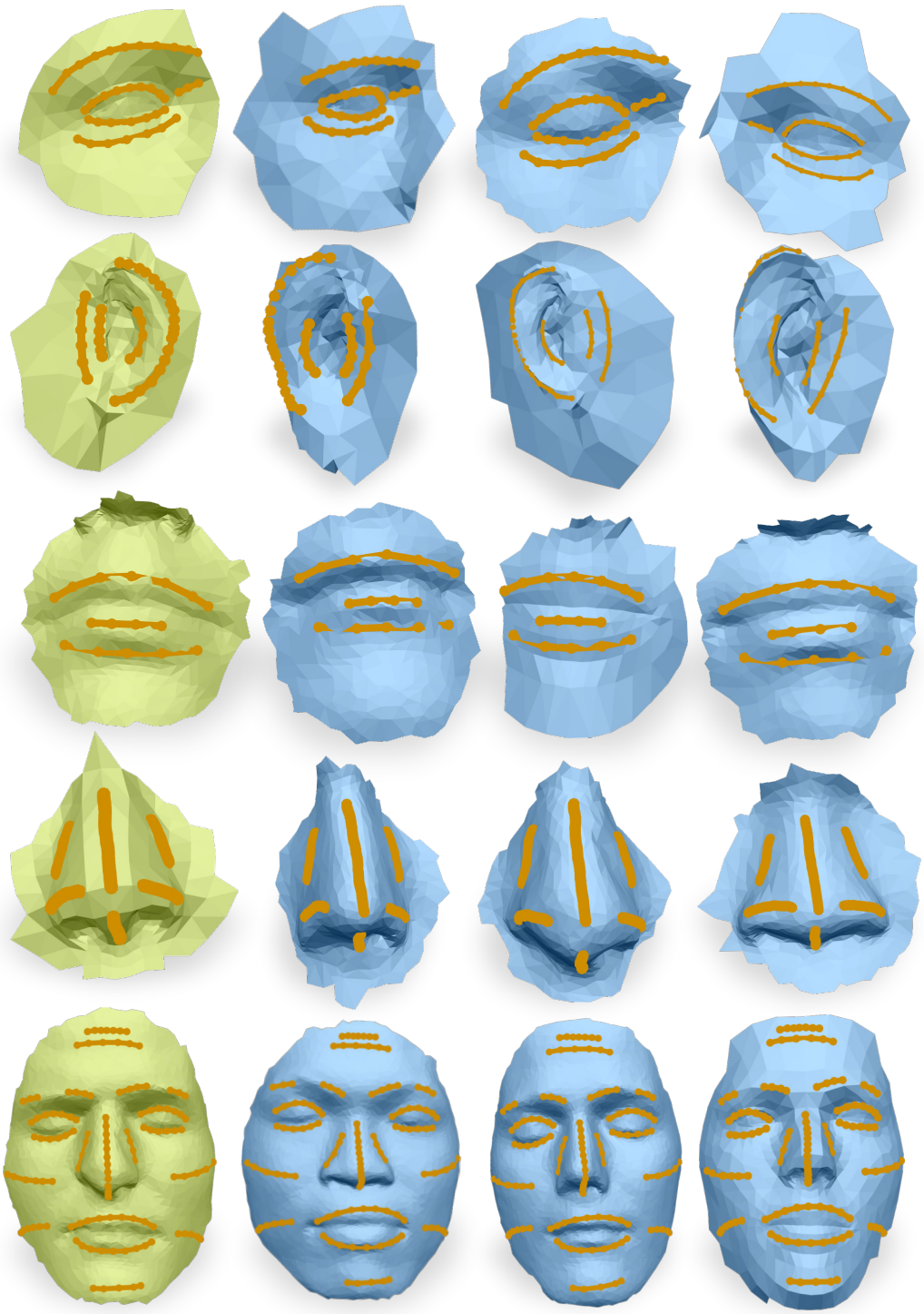


Figure 4.10: Each row shows a source patch (green), equipped with a set of polygonal lines. Using our method, the lines are mapped to three different target patches each (blue). The resulting mappings contain reflections (row 1 and 2), minor scale difference (e.g. row 1, column 2) and slightly misaligned patch centers (row 3, column 3).

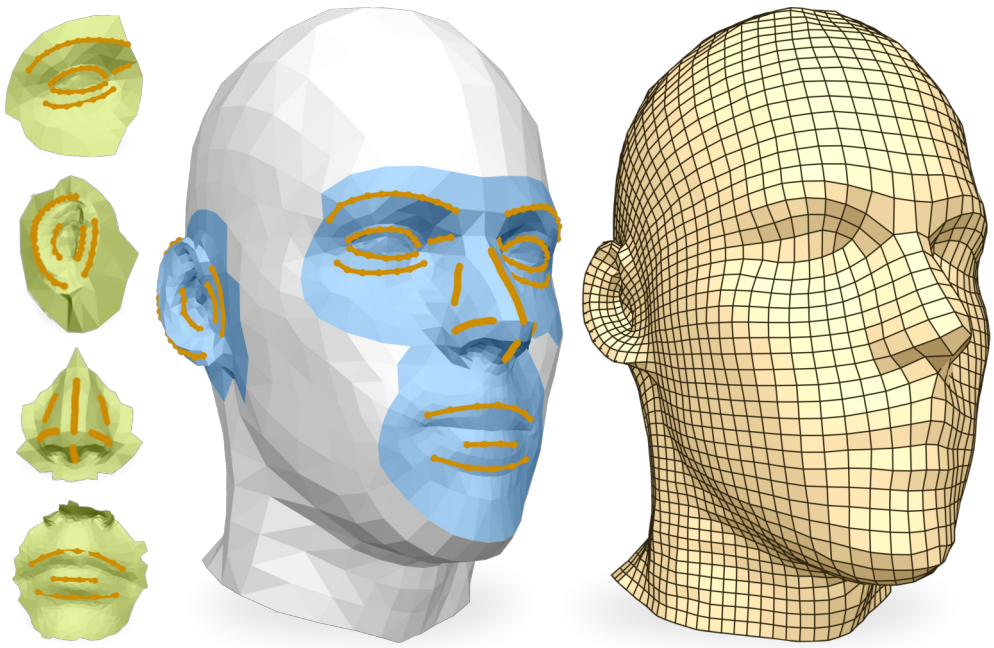


Figure 4.11: The model of a human head is equipped with four distinct macro constraints. At six positions, a point and a geodesic radius is picked, defining a target patch (blue). Our re-embedding method is then employed to transfer polygonal lines from one of the source patches (green). For both eyes and ears, the same source patch is applied twice, yielding symmetric constraint sets. Finally, the mapped lines are used as guiding constraints for [CBK15], producing the quad mesh on the right.

Using the settings and descriptor choice explained so far, the constraint re-embedding results in Figure 4.10 are obtained. Figure 4.11 shows a quad mesh guided by direction constraints transferred from multiple source patches.

Continuity and Injectivity

Using our technique to transfer lines between surfaces demonstrates its power only partially. In contrast to many point matching methods, we compute a map that is (1) densely available at each point of the overlapping region, (2) C^0 continuous, and (3) guaranteed to be locally injective everywhere. It thus can be used to transfer any kind of field between two surface regions, as long as this field can be linearly interpolated. For the particular use case of macro constraints, this allows to also transfer singularity configurations as well as local sizing information in form of a scalar or metric field. In Figure 4.12 this property is demonstrated by mapping texture coordinates from the source to the target mesh.

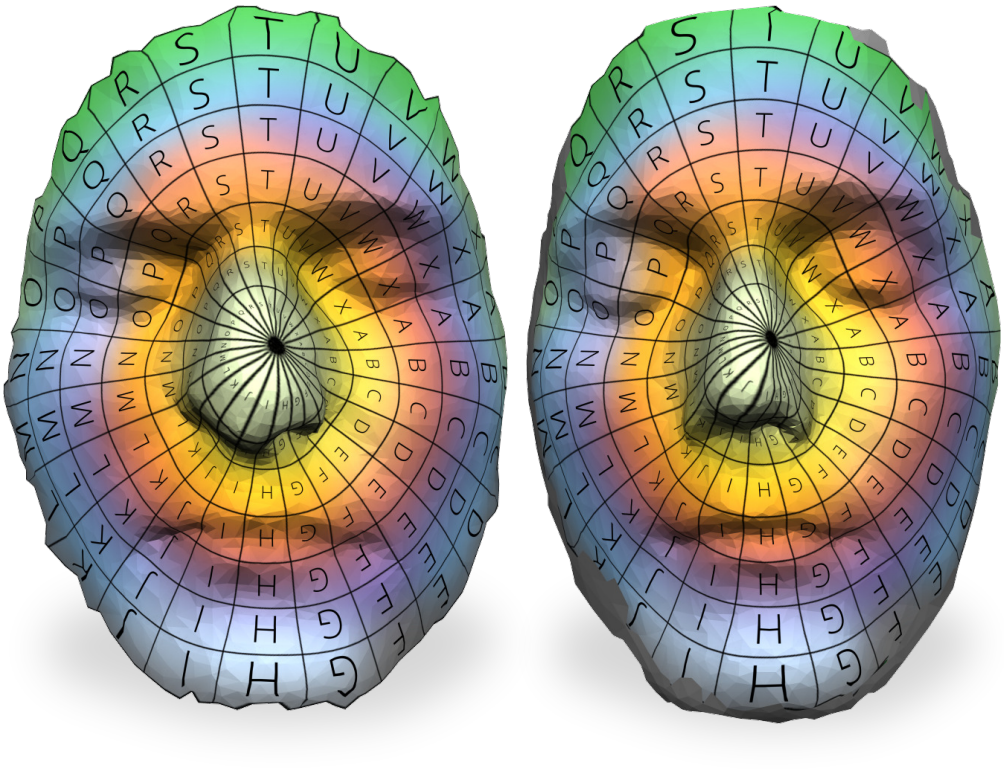


Figure 4.12: Our method computes a continuous mapping establishing a dense field of high-quality point-to-point correspondences. It can be used to injectively map various kinds of fields between the surface regions. Here, the source patch (left) is textured using its initial disk parametrization and the texture coordinates are transferred to the target (right) using our inter-surface map. Note how details in both meshes are textured correspondingly despite geometric differences. Regions in which the map is not available are colored in gray on the target mesh.

Isotropic and Anisotropic Scaling

Since conformality was chosen as our distortion measure, certain types of maps are favored by our method. A conformal map allows for an isotropic scaling per point, however not for an anisotropic one. The practical effects of this are discussed at two examples.

In the first example, the recovery from a synthetically stretched initialization is observed (see Figure 4.13). Here, the initial source parametrization is scaled by a factor of 2 along a single axis. It can be seen, that in a first phase (iteration 0 to $1\,000$) the conformality term is dominant and deforms the anisotropic map back into an isotropic one. In a second phase (iteration $1\,000$ to $10\,000$) the source mesh is scaled down uniformly to achieve better descriptor distances.

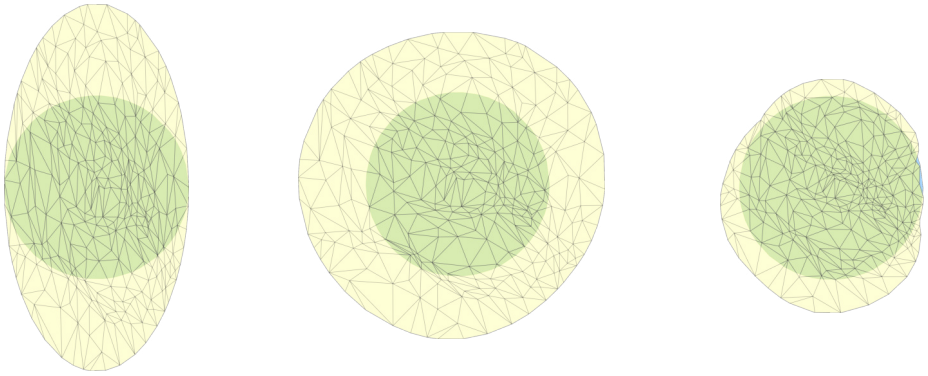


Figure 4.13: From an artificially stretched initialization, we observe that our method first restores isotropy of the map and then applies a uniform down-scaling to obtain better descriptor matches. The deformation is shown at iterations 0, 1 000 and 10 000.

The second example (Figure 4.14) shows two patches, both containing an ear but including differently large mesh regions around it. Since both patches are normalized to a geodesic radius of 1, the ear in the source patch is slightly smaller than its target counterpart. As an effect, the transferred constraints only cover the upper part of the ear in the initial solution. During the following iterations, this difference in scale is accommodated for in a roughly uniform manner. Since however the target ear is also higher then the source, an anisotropc scaling would be required for a the desired result. This however is only provided by our method up to a certain extend.

Furthermore, the least-squares conformal maps energy, which we use as an approximation to conformality, is commonly known to favor shrinkage in the target domain. This is because the deviation from conformality is measured by a 2D vector which shrinks if the map is a down-scaling. In our experiments we do not observe this behavior as the descriptor objective provides a strong regularization. In contrast, slight growth can be noticed close to the boundary of the overlapping region. This is provoked by two factors: (1) the descriptor objective can only push vertices out of the overlap but cannot pull them back inside, and (2) conformality as an objective tends to "blow up" boundaries of a convex shape.

Runtime Complexity

The execution time of a single iteration is primarily dominated by the 2D triangle lookup for each vertex of the source mesh. Assuming a uniform distribution of target vertices in the parameter domain, the uniform gird acceleration structure (see Section 4.2) allows for constant-time lookups. If this is not given, falling

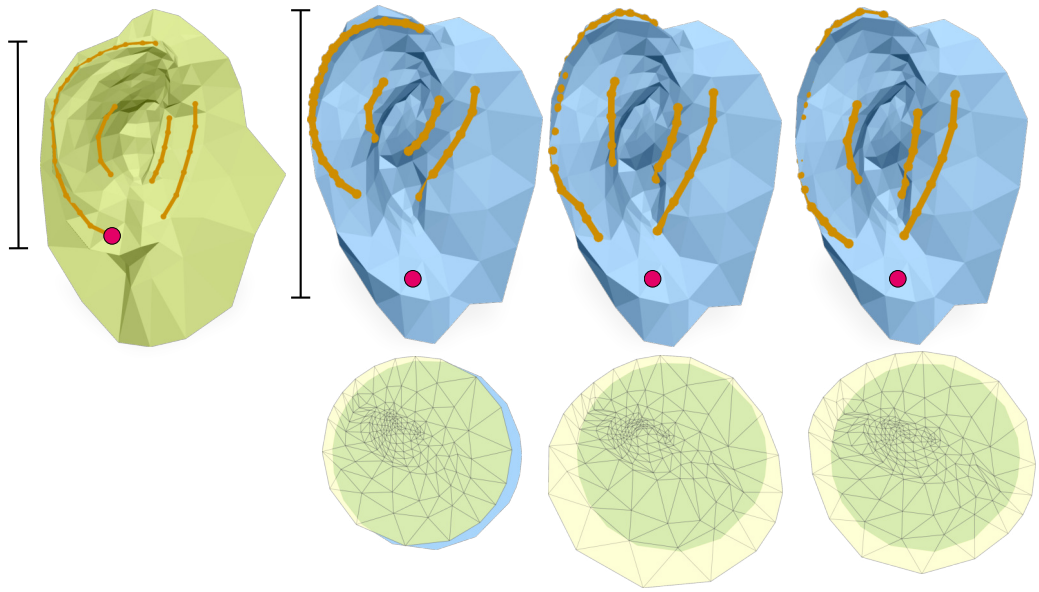


Figure 4.14: Both ears show different relative sizes within their patches. While isotropic scaling is handled gracefully by our method, anisotropic scaling is only possible to a certain extend. The magenta colored dot symbolizes a desired correspondence which is not fully reached. The results at iterations 0, 2 000 and 10 000 ("converged") are shown.

back to a quad tree or BSP tree gives a logarithmic lookup time. Given the target triangle for each source vertex, the computation of both the objective function as well as its gradient are relatively inexpensive. Note that these are purely local, i.e. each source triangle corner can be processed by just considering its other two vertices and its corresponding target triangle. No extended neighborhoods have to be taken into account. In total, each fixed-step iteration has linear or log-linear runtime in the number of source vertices. If an exponential back-off or a line search is performed, the costly lookup has to be repeated in each step.

Due to our crude optimization heuristic, an extremely large number of iterations has to be performed. Frequent back-offs from barriers slow down the optimization even further. Consequently, the performance is far from being interactive. In our unoptimized implementation, the examples shown here took up to three minutes to compute. Possible improvements are addressed in Section 5.1.3.

Descriptor Magnitude Mismatch

An inherent problem of descriptor distance minimization as an optimization target can be observed in Figure 4.15. Here, our one-dimensional curvature descriptor is visualized as a heat map. The eyebrow in both patches can easily be identified by its high curvature. However, the curvature takes slightly higher values in the

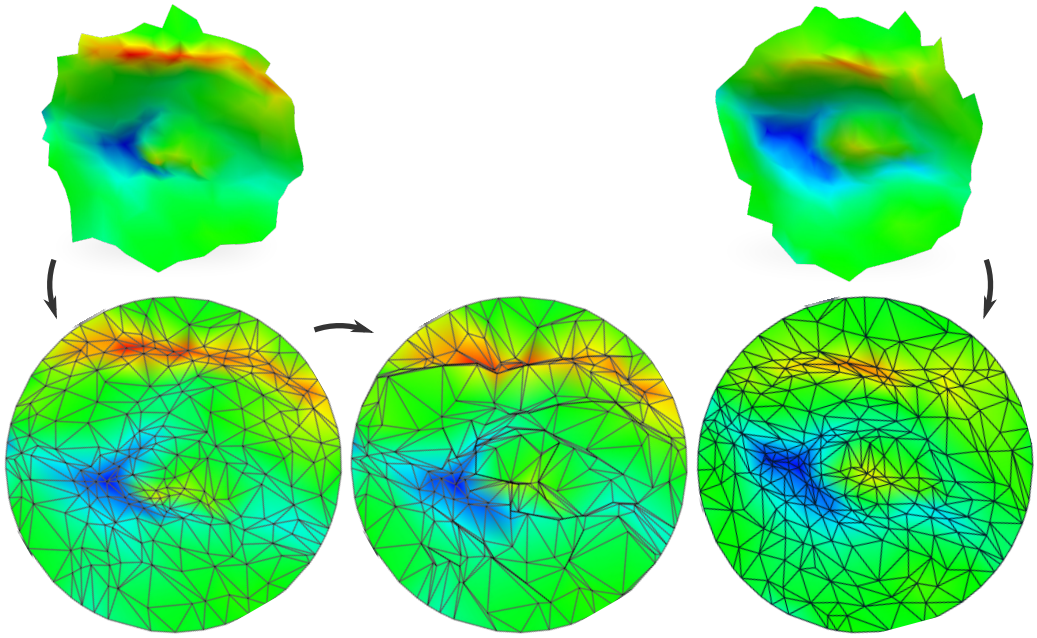


Figure 4.15: The scalar mean curvature descriptor is shown as a heat map. Note the difference in magnitude at the eyebrow of the source (top left) and the target (top right) patch. Disabling our distortion measure, i.e. setting $\alpha_c = 0$, reveals the effects of this mismatch problem. In the deformed source disk (bottom center), an entire area long the eyebrow degenerates to a line.

source patch than in the target patch. Clearly, we expect our algorithm to be robust with respect to these small geometric changes as such situations present its main use case.

When only considering the descriptor objective and switching off the distortion measure, the following issue is revealed: since each variable vertex of the source mesh tries to minimize its descriptor distance, entire regions in the source mesh find their best match at peaks of the target descriptor. Thus, parts of the mesh try to degenerate to a point or a line.

Consequently, in these cases, the descriptor term is in ongoing conflict with the distortion term, which prevents the degeneration. This is a reason why the distortion measure has to receive a high weight in most examples and represents more than just a slight regularization.

Patch Tessellation

Experiments show that our current implementation is sensitive to the mesh tessellation. All positive results presented here are computed on relatively coarse

meshes of up to 1600 vertices (face class) and 700 vertices (other classes). Choosing a finer resolution has the following effects: (1) small triangles in the source patch threaten to flip more easily, which increases the number back-off steps and provokes the issue of getting stuck at barriers. (2) Short edge lengths in the target mesh enforce a smaller gradient step length and thus require more iterations. (3) More triangles in the target mesh also increase the number of discontinuities in the objective function and emphasize the problems of the current formulation. In fact, we could not produce comparable results with significantly finer tessellations. The issue is further addressed in the future work Chapter 5.1.

High Curvature Patches

If the geometry of a patch exhibits extreme Gaussian curvature, a highly distorted disk parametrization is inevitable. Although our approach is intended to be as agnostic as possible to the metric induced by the initial maps $\boldsymbol{f}$ and $\boldsymbol{g}$, some problems remain. Depending on the type of distortion in $\boldsymbol{f}$ and $\boldsymbol{g}$, one of the following situations occurs:

First, high area distortion significantly decreases the quality of an initialization via rigid alignment. Due to shrinkage in parameter space, small distances in $\mathbb{R}^2$ correspond to large geodesic distances in the surface domain, magnifying slight misalignments. Furthermore, in the presence of high distortion, we observe considerable differences in the the initial parametrizations of both patches which cannot be compensated by a rigid transformation. As a result of a bad initialization, it becomes likely that the desired local optimum is not found.

Second, it is possible to reduce the area distortion of e.g. an harmonic parametrization at the cost of higher conformal distortion. However, assuming a uniform tessellation of both patches in the surface domain, this enforces extremely thin triangles in the parameter domain. As a consequence, the entire optimization process takes place in close proximity to the boundary of the feasible region, because thin triangles threaten to flip easily. This provokes extreme numbers of back-off iterations and accentuates the problem of getting stuck at barriers.

Indeed, these problems do not permit successful results on both the hand and foot classes of our dataset. Overcoming these issues is an important aspect of future work (see Section 5.1.3).

In this main part of the thesis, a descriptor guided approach towards computing partial inter-surface mappings between two topological disks was presented. Challenges in finding a suitable problem formulation were discussed in detail. Despite remaining issues of the current formulation, promising results can already be achieved in a range of examples.

Chapter 5

Conclusion

This work tackled two sub problems of an envisioned constraint recommendation system for interactive quad meshing sessions. First, the task of patch retrieval in a given database was approached. As various techniques for the general problem of shape retrieval exist, a bag-of-features approach could be successfully adapted to our specialized setting. An evaluation on a small dataset revealed satisfactory results given the simplicity of the method.

After that, the intricate problem of both detecting matching regions of two surface patches and computing a mapping between them was approached. Formulating the problem in a 2D parametrization setting and minimizing descriptor distances as well as conformal distortion results in a non-convex optimization problem. Our proposed attempt to solve the problem yields detailed insight into its core challenges. A thorough understanding of these challenges gives rise to future work.

Implementation

As part of this thesis, the bag-of-features retrieval (Chapter 3) was implemented in MATLAB. The proposed mapping algorithm (Chapter 4) was implemented in C++ using the OpenMesh data structure [BSBK02] and the OpenFlipper framework [MK12]. Existing implementations were used for principal curvature computation as well as for quad mesh generation [BZK09], [CBK15]. The wave kernel signature was also computed using the original MATLAB implementation published by the authors [ASC11].

5.1 Future Work

Potential topics for future work arising from this thesis can be divided into three fields: (1) working towards an interactive constraint suggestion system from an application perspective, (2) evaluating more elaborate retrieval methods in our context, and (3) further pursuing descriptor-guided mappings between surfaces.

5.1.1 An Interactive Constraint Suggestion System

At the heart of the system sketched here, there has to be a powerful database of surface patches equipped with artist-chosen constraints. The data acquisition for such a project carries an entire field of challenges on its own. First, a sufficiently large number of meshes needs to be constrained by skilled artists. The search for suitable demonstration meshes itself already proved to be of little success. Commonly used datasets to showcase quad meshing algorithms (e.g. [MPZ14]) either contain too exotic meshes, providing no additional value, or show little variety within a class of models. Shape collections, as used in global shape retrieval and mapping tasks (e.g. [BBC+10]), usually exhibit a low level of detail and thus often lose their value when restricted to small regions. In general, it is still unclear, how much inter-class variation should be permitted to create an effective recommendation system.

Second, supporting patches around macro constraints have to be extracted. Even when performing this task manually, it is often unclear how to choose a suitable region around a constraint. One the one hand, such a region has to carry enough discriminating geometry to assure that no unrelated or undesired constraint sets are proposed. On the other hand, a region has to be local enough to still allow for flexible re-use. In addition, an artist's constraint, e.g. a polygonal line, might cover large parts of a mesh, rendering it inherently non-local. In this context, our restriction of patches being geodesic discs without handles is clearly a limiting factor. Automatic extraction of such patches is expected to be difficult, as the expected result is often unclear.

Both aspects, collecting an adequate amount of data and extracting useful patches, seem to require new ideas before the proposed system can be turned into a practically useful tool.

Another aspect, omitted in this work, is to define meaningful user interactions. A system actually assisting artists has to be simple to use and, maybe even more importantly, its proposed results have to be easy to modify. Thus, three important questions for the design of a user interface should be: (1) How to efficiently select query regions on the target mesh? (2) How to present a preview of one or multiple

retrieved macro constraints? (3) How to allow manual post processing of the re-embedded constraint set?

As seen in this thesis, implementing such a system comes with enormous challenges in both content and algorithmic aspects. Moreover, it is not yet studied to which extent the system would be valued by users. In its current draft, it has the potential to save small parts of manual work or provide inspiration, but bears the danger of suggesting suboptimal or undesired solutions. Thus, as a precondition to further pursuing this direction, the general effectiveness of the sketched system should be evaluated thoroughly.

5.1.2 Patch Retrieval

A well known drawback of a bag-of-features approach, as employed here, is its lack of spatial sensitivity. In our situation this shortcoming is particularly emphasized: spectral descriptors, usually carrying a certain amount on non-local information, cannot play on their strength when restricted to local surface patches.

Thus, a possible direction for future work could be to experiment with some of the many existing spatial extensions of the bag-of-features model. In parallel to this, other families of retrieval methods could be tailored to our setting and evaluated as well.

More generally, various descriptors could be analyzed with respect to their performance in locally restricted regions. In particular, an interesting question is how descriptor values computed solely on a surface patch compare to those computed on the entire shape. While this analysis is straightforward for simple curvature based descriptors, it is more involved in the case of spectral descriptors. E.g. [RCB+15] already provides helpful insights by showing how cutting away parts of a mesh changes the eigenfunctions of its Laplacian.

5.1.3 Partial Inter-Surface Mappings Between Disks

The most challenging aspect of this work is finding a high-quality partial mapping between two patches. Although the general approach presented here seems promising, it could not be turned into a robust algorithm within the scope of this thesis. In the following, multiple ways to finalize the method are proposed.

Reformulation of the Objective Function

An inherent problem of our current formulation is that optimizing for low descriptor distances and optimizing for large overlap are conflicting concepts. The situation was partially solved by introducing an alternative objective function. This

turns the descriptor distance measure into a similarity measure and thus naturally combines both concepts. Yet, it is only used for initialization as it has no means of penalizing distortion. Adding a distortion penalty brings back the original conflict, i.e. fewer correspondences imply less distortion. A possible idea is to also invert the distortion measure such that non-existing overlap is ignored, high-distortion overlap is tolerated, and low-distortion overlap is rewarded. This inversion however is expected to impose a different characteristic on the behavior of distortion, which is worth investigation. In case of positive results, three problems might be solved: (1) initialization and non-linear optimization work on the same objective function, (2) balancing between large overlap and other objectives is abandoned entirely, and (3) discontinuities with respect to vertices entering and leaving the overlap disappear.

An even more severe problem of the current objective is the way distortion measures are discretized with respect to the composition of three maps. Recall that the Jacobian of the map from the parameter domain to the target surface is constant per face. This is due to the piecewise linearity of the mapping and eventually causes C^0 discontinuities in the objective, preventing well-defined optimization techniques. Since the problematic third map does not change, it could be slightly modified to smooth out discontinuities. For example, it is conceivable to average the Jacobian in each one-ring at its center vertex and from there on linearly interpolate it back over the triangles. This modification does not cause any changes when considered in the limit of infinitesimally fine tessellation and yields a C^0 continuous distortion measure.

Still, the resulting distortion objective is not expected to be a convex function, due to the metric induced by the disk parametrization of each patch. It would be interesting to better understand the non-convexity of this function with respect to both disk parametrizations. This could give further insights in how to effectively optimize for it while avoiding local minima.

Finally, the distortion measure employed here (least-squares conformal maps), was mainly chosen due to its simplicity. Since our optimization is inevitably non-linear anyway, a variety of other distortion measures is available. How the choice of distortion measure influences the non-convexities or the scale inversion, mentioned above, is still unclear.

Efficient Optimization

Considering the interactive application purpose for our mapping technique, real-time performance is a requirement. Thus, more efficient ways to solve the non-linear problem have to be investigated.

Given a C^0 continuous objective function, as described above, we can abandon our optimization heuristic and pick from a number of general purpose optimization methods. Gradient descent with backtracking line search, as initially pursued, is expected to yield a significantly lower number of iterations as well as proper convergence.

The cost of a line search is dominated by the 2D triangle lookup. Considering a piecewise linear objective, an alternative to this lookup is tracing through the target mesh: a given descent direction consists of a 2D ray for each variable vertex. Instead of picking different positions along these rays by backtracking, they could be explored in a forward fashion, i.e. each time a ray intersects an edge of the target mesh, the objective function is updated.

Since C^1 discontinuities are still inherently present in both the descriptor and the distortion objective, Newton methods cannot easily be applied. Still, the entire family of quasi-Newton methods is available and can be evaluated for performance improvements. These algorithms do not require an explicit Hessian, but successively compute an approximation by considering a sequence of the latest gradient vectors. Alternatively, turning the objective into a least-squares energy allows to apply the Levenberg–Marquardt algorithm. This method can be seen as a smooth interpolation between Gauss-Newton and gradient descent steps.

Hierarchical Approaches

Our method calls for an hierarchical approach in two ways. First, the 2D deformation could be computed in a coarse-to-fine manner to improve run time performance. A possible technique would be to start by decimating the source mesh and then letting the optimization converge. After that, the decimation can successively be reverted, while the optimization continues after each step. In addition, the decimation could be performed sensitively to the attached descriptor field. This means that vertices are preferably removed in areas of nearly constant descriptor value. More generally, re-meshing both patches, such that they match specified quality criteria and then transferring the resulting mapping from the re-meshed to the original mesh, can also be considered.

Second, high-frequency descriptor fields are likely to cause plenty of undesired local minima in the objective function. An idea worth pursuing is to smooth the descriptor fields and consequently the objective function. In this setting, a coarse-to-fine approach could start the optimization on very smooth fields, allowing the source parametrization to roughly slide into the right location. In the following, the smoothing can be reverted successively, updating the solution after each step.

Last, in case both concepts prove to be practical on their own, a combination of them is desirable.

Improved Initial Parametrization

For a majority of our experiments, harmonic parametrizations with mean value weights were used as initialization. However, these cause issues due to their high area distortion and better solutions are likely to be found.

Moreover, properties of these parametrizations are not exploited by our method yet. In particular, it is of interest if an interleaved optimization schemes exist, which in turn optimizes one of the three maps involved. Whether such a setting could avoid having to optimize the composition of all three maps at once is important for the choice of future research directions.

Finally, the randomized approach for finding the initial rigid transformation could be replaced by more reliable methods.

Extension to Non-Disk Topologies

If a robust method for the current setting can be found, it is interesting to see if our method can be extended to compute mappings between shapes of higher genus. This involves cutting the mesh and optimizing in the presence of charts and transition functions. A series of papers operating in such a setting exists ([APL14], [APL15], [AL15]), yet it has to be examined how their insights can be applied to our setting.

5.2 Summary

In this thesis, an envisioned constraint suggestion system for parametrization based quad meshing was sketched. By specifying a region of interest on a target mesh, a potential user can query a database for a similar region, on which a set of constraints is already defined. These constraints can then be transferred to the target mesh, where they are used as an input to existing quad meshing methods. The particular contribution of this work is two-fold:

First, a bag-of-features model was adapted to this specific situation. While an evaluation already provides good results, future work might target improved spatial sensitivity as well as a more thorough understanding of the behavior of spectral descriptors on isolated surface patches.

Second, a descriptor-guided method for the computation of partial inter-surface maps between disks was presented. The inherently two-dimensional nature of the task is exploited by formulating the problem in a parametrization setting. Several aspects of the problem formulation as well as optimization strategies were discussed in detail. In the evaluation, first positive results could be demonstrated. Finally, a thorough understanding of the remaining challenges was used to identify specific tasks for future research.

Acknowledgments

I wish to thank my advisors Prof. Dr. Leif Kobbelt and Prof. Dr. David Bommes for making this thesis possible.

Spacial thanks go to my supervisor Hans-Christian Ebke for his continuous help and for being a mentor over the past three years.

Further, I would like to thank Professor David Bommes, Anne Gehre, Isaak Lim, Javor Kalojanov and Manish Mandad for their advice on optimization, descriptors and mappings.

In addition, I would like to thank Martha Wingen for her ongoing support, Hannes Hergeth for numerous insightful discussions as well as for reviewing this thesis, and Daniel Töws for his valuable feedback.

Finally, I thank Jan Möbius for maintaining the OpenFlipper framework and David Bommes for providing the QGP pipeline.

Appendix A

Least-Squares Conformal Maps Gradient

In this appendix, we derive the gradient of the least-squares conformal maps energy of the concatenated map $\mathbf{\Phi} = \boldsymbol{g}^{-1} \circ \boldsymbol{\phi} \circ \boldsymbol{f}$. In particular, the gradient is formed for each source triangle (abc) with respect to the location of a single triangle corner $\mathbf{a}' = \boldsymbol{\phi}(\boldsymbol{f}(a))$ in parameter space.

Recall that the Jacobian of the map $\mathbf{J}_{\boldsymbol{\phi}}$ is computed as:

$$\mathbf{J}_{\boldsymbol{\phi}} \underbrace{\begin{bmatrix} b_0 - a_0 & c_0 - a_0 \\ b_1 - a_1 & c_1 - a_1 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} b'_0 - a'_0 & c'_0 - a'_0 \\ b'_1 - a'_1 & c'_1 - a'_1 \end{bmatrix}}_{M'}$$

$$\mathbf{J}_{\boldsymbol{\phi}} = M' \cdot M^{-1}.$$

Here, $\mathbf{a} = (a_0, a_1)^T$, $\mathbf{b} = (b_0, b_1)^T$ and $\mathbf{c} = (c_0, c_1)^T$ denote the positions of triangle corners before $\boldsymbol{\phi}$ is applied, whereas $\mathbf{a}'$, $\mathbf{b}'$, $\mathbf{c}'$ are their respective images under $\boldsymbol{\phi}$. Further, note that the energy we want to derive depends on the composition of Jacobians $\mathbf{J}_{\mathbf{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}$. We refer to the entries of $\mathbf{J}_{\mathbf{\Phi}}$ as

$$\mathbf{J}_{\mathbf{\Phi}} = \begin{bmatrix} \frac{\partial \mathbf{\Phi}}{\partial u} & \frac{\partial \mathbf{\Phi}}{\partial v} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{00} & \mathbf{J}_{01} \\ \mathbf{J}_{10} & \mathbf{J}_{11} \end{bmatrix}.$$

Now, for a single triangle $t = (abc)$ in $\mathbf{S}'$, we express the least-squares conformal maps term as:

$$\begin{aligned} \mathrm{lscm}_{\mathbf{\Phi}}(abc) &= \left\| \frac{\partial \mathbf{\Phi}}{\partial v} - \mathrm{Rot}_{90} \frac{\partial \mathbf{\Phi}}{\partial u} \right\|^2 = \left\| \begin{bmatrix} \mathbf{J}_{01} + \mathbf{J}_{10} \\ \mathbf{J}_{11} - \mathbf{J}_{00} \end{bmatrix} \right\|^2 \\ &= (\mathbf{J}_{01} + \mathbf{J}_{10})^2 + (\mathbf{J}_{11} - \mathbf{J}_{00})^2. \end{aligned}$$

To form its gradient, we derive $\mathrm{lscm}_{\mathbf{\Phi}}(abc)$ with respect to the location of the triangle corner a in parameter space. This location $\mathbf{a}' = \boldsymbol{\phi}(\boldsymbol{f}(a)) \in \mathbb{R}^2$ corresponds to two variables in the unknown vector $\mathbf{x}$. We start the derivation top-down, by applying the chain rule to $\mathrm{lscm}_{\mathbf{\Phi}}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{a}'} \mathrm{lscm}_{\mathbf{\Phi}} \;=\; & 2 \cdot (\mathbf{J}_{01} + \mathbf{J}_{10}) \cdot (\frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{01} + \frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{10}) \\ & + \; 2 \cdot (\mathbf{J}_{11} - \mathbf{J}_{00}) \cdot (\frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{11} - \frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{00}) \;\in \mathbb{R}^2.\end{aligned} \tag{A.1}$$

To find derivatives for the individual entries $\mathbf{J}_{\bullet\bullet}$, we use the decomposition $\mathbf{J}_{\mathbf{\Phi}} = \mathbf{J}_{\boldsymbol{g}}^{-1} \cdot \mathbf{J}_{\boldsymbol{\phi}} \cdot \mathbf{J}_{\boldsymbol{f}}$. Assuming $\mathbf{J}_{\boldsymbol{f}}$ and $\mathbf{J}_{\boldsymbol{g}}^{-1}$ constant, we have:

$$\frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{\mathbf{\Phi}} \;=\; \mathbf{J}_{\boldsymbol{g}}^{-1} \;\cdot\; \frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{\boldsymbol{\phi}} \;\cdot\; \mathbf{J}_{\boldsymbol{f}}.$$

Furthermore using $\mathbf{J}_{\boldsymbol{\phi}} = M' \cdot M^{-1}$, where M^{-1} is constant, gives:

$$\frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{\boldsymbol{\phi}} \;=\; \frac{\partial}{\partial \mathbf{a}'}M' \;\cdot\; M^{-1}.$$

Recalling that

$$M' \;=\; \begin{bmatrix} b_0' - a_0' & c_0' - a_0' \\ b_1' - a_1' & c_1' - a_1' \end{bmatrix} \quad \text{and} \quad M^{-1} \;=\; \frac{1}{\det M}\begin{bmatrix} c_1 - a_1 & a_0 - c_0 \\ a_1 - b_1 & b_0 - a_0 \end{bmatrix},$$

we have

$$\frac{\partial}{\partial a_0'}M' \;=\; \begin{bmatrix} -1 & -1 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \frac{\partial}{\partial a_1'}M' \;=\; \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix},$$

which finally leads to

$$\frac{\partial}{\partial a_0'}\mathbf{J}_{\mathbf{\Phi}} \;=\; \frac{1}{\det M} \;\cdot\; \mathbf{J}_{\boldsymbol{g}}^{-1} \;\cdot\; \begin{bmatrix} b_1 - c_1 & c_0 - b_0 \\ 0 & 0 \end{bmatrix} \;\cdot\; \mathbf{J}_{\boldsymbol{f}} \;\in \mathbb{R}^{2\times 2} \tag{A.2}$$

and

$$\frac{\partial}{\partial a_1'}\mathbf{J}_{\mathbf{\Phi}} \;=\; \frac{1}{\det M} \;\cdot\; \mathbf{J}_{\boldsymbol{g}}^{-1} \;\cdot\; \begin{bmatrix} 0 & 0 \\ c_0 - b_0 & b_1 - c_1 \end{bmatrix} \;\cdot\; \mathbf{J}_{\boldsymbol{f}} \;\in \mathbb{R}^{2\times 2}. \tag{A.3}$$

Now let $\frac{\partial}{\partial \mathbf{a}'}\mathbf{J}_{\bullet\bullet}$ be the $\mathbb{R}^2$ vector having the entries $\left(\frac{\partial}{\partial a_0'}\mathbf{J}_{\mathbf{\Phi}}\right)_{\bullet\bullet}$ and $\left(\frac{\partial}{\partial a_1'}\mathbf{J}_{\mathbf{\Phi}}\right)_{\bullet\bullet}$ as its components. These vectors can be plugged into Equation A.1, yielding a gradient vector $\frac{\partial}{\partial \mathbf{a}'}\mathrm{lscm}_{\mathbf{\Phi}}$ in $\mathbb{R}^2$. This vector tells us in which direction $\mathbf{a}'$ has to be moved to maximally increase the distortion within a single triangle.

Forming the gradient of $\mathrm{lscm}_{\boldsymbol{\phi}}$ in $\mathbf{S} \setminus \mathbf{S}'$ works in exactly the same way except for $\mathbf{J}_{\boldsymbol{f}}$ and $\mathbf{J}_{\boldsymbol{g}}^{-1}$ being the identity.

We will write $\nabla \mathrm{lscm}(abc)$ for either $\mathrm{lscm}_{\mathbf{\Phi}}(abc)$ or $\mathrm{lscm}_{\boldsymbol{\phi}}(abc)$ depending on whether the triangle lies within the overlapping region or not.

Appendix B

Injectivity Barrier Gradient

Deriving the barrier function in Equation 4.3 is somewhat simpler, as it is independent of the maps $\boldsymbol{f}$ and $\boldsymbol{g}$ and thus avoids many discretization artifacts. The discrete version of $\mathrm{obj}_{\text{flip}}$ is:

$$\mathrm{obj}_{\text{flip}} \quad = \quad \sum_{v \in \mathbf{S}} \max\left(0, \underbrace{-\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}/s_f\right)}_{=:B}\right) \cdot A_v \tag{B.1}$$

To find its gradient, we again consider a variable triangle corner a and fixed corners b and c. As before, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ denote the original 2D positions of vertices from $\mathbf{S}$ and $\mathbf{a}'$, $\mathbf{b}'$, $\mathbf{c}'$ their images under $\boldsymbol{\phi}$. The 2D gradient vector $\nabla \operatorname{flip}(abc) = \frac{\partial}{\partial \mathbf{a}'} \mathrm{obj}_{\text{flip}}$ tells us how to move the variable vertex $\mathbf{a}'$ to increase the barrier term as quickly as possible. Using the definitions of M and M' from above, the barrier can be rewritten as:

$$\begin{aligned} B(\mathbf{a}') &= -\log\left(\det \mathbf{J}_{\boldsymbol{\phi}}/s_f\right) \\ &= -\log\left(\det(M' \cdot M^{-1})/s_f\right) \\ &= -\log\left(\frac{\det M'}{s_f \cdot \det M^{-1}}\right) \\ &= -\log\left(\det M'\right) + \underbrace{\log\left(s_f \cdot \det M^{-1}\right)}_{\text{const}}. \end{aligned}$$

Its derivative with respect to the position $\mathbf{a}'$ is then

$$\frac{\partial}{\partial \mathbf{a}'} B \;=\; -\frac{1}{\det M'} \cdot \frac{\partial}{\partial \mathbf{a}'} \;\det\; M' \;\in \mathbb{R}^2,$$

with

$$\begin{aligned}\frac{\partial}{\partial \mathbf{a}'} \det M' &= \frac{\partial}{\partial \mathbf{a}'}\big((b'_0 - a'_0)(c'_1 - a'_1) - (b'_1 - a'_1)(c'_0 - a'_0)\big) \\ &= \frac{\partial}{\partial \mathbf{a}'}\big(b'_0c'_1 - b'_0a'_1 - a'_0c'_1 - b'_1c'_0 + b'_1a'_0 + a'_1c'_0\big) \\ &= \begin{bmatrix} b'_1 - c'_1 \\ c'_0 - b'_0 \end{bmatrix}.\end{aligned}$$

Considering, that we clip negative function values to 0, the gradient for a single triangle corner is:

$$\nabla \operatorname{flip} = \begin{cases} -\frac{1}{\det M'} \cdot \begin{bmatrix} b'_1 - c'_1 \\ c'_0 - b'_0 \end{bmatrix} & \det \mathbf{J}_{\boldsymbol{\phi}}/s_f \;\leq\; 1 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \det \mathbf{J}_{\boldsymbol{\phi}}/s_f \;>\; 1. \end{cases}$$

In total, the gradient of $\mathrm{obj}_{\mathrm{flip}}$ with respect to the variable vector $\mathbf{x}$ is then:

$$\nabla \mathrm{obj}_{\mathrm{flip}} = \frac{1}{3} \begin{bmatrix} \sum\limits_{(b,c)\in N(a_0)} \nabla \operatorname{flip}(a_0bc) \cdot A_{a_0bc} \\ \vdots \\ \sum\limits_{(b,c)\in N(a_{n-1})} \nabla \operatorname{flip}(a_{n-1}bc) \cdot A_{a_{n-1}bc} \end{bmatrix} \in \mathbb{R}^{2n}. \tag{B.2}$$

References

[AL15] Noam Aigerman and Yaron Lipman. “Orbifold Tutte Embeddings”. In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 190:1–190:12 (cited on pages 13, 74).

[AL16] Noam Aigerman and Yaron Lipman. “Hyperbolic Orbifold Tutte Embeddings”. In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 217:1–217:14 (cited on page 13).

[APL14] Noam Aigerman, Roi Poranne, and Yaron Lipman. “Lifted Bijections for Low Distortion Surface Mappings”. In: *ACM Trans. Graph.* 33.4 (July 2014), 69:1–69:12 (cited on pages 13, 35, 36, 74).

[APL15] Noam Aigerman, Roi Poranne, and Yaron Lipman. “Seamless Surface Mappings”. In: *ACM Trans. Graph.* 34.4 (July 2015), 72:1–72:13 (cited on pages 13, 74).

[ASP+04] Dragomir Anguelov, Praveen Srinivasan, Hoi-Cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. “The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces”. In: *Proceedings of the 17th International Conference on Neural Information Processing Systems*. NIPS’04. Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 33–40 (cited on page 13).

[ASC11] M. Aubry, U. Schlickewei, and D. Cremers. “The wave kernel signature: A quantum mechanical approach to shape analysis”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. 2011, pp. 1626–1633 (cited on pages 11, 12, 18, 24, 69).

[Bli96] J. Blinn. “Consider the lowly 2 x 2 matrix”. In: *IEEE Computer Graphics and Applications* 16.2 (Mar. 1996), pp. 82–88 (cited on page 36).

[BRLB14] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. “FAUST: Dataset and evaluation for 3D mesh registration”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Piscataway, NJ, USA: IEEE, June 2014 (cited on page 22).

[Bom12] David Bommes. “Quadrilateral Surface Mesh Generation for Animation and Simulation”. Dissertation. RWTH Aachen, 2012 (cited on pages 2, 5).

[BCE+13] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. “Integer-grid Maps for Reliable Quad Meshing”. In: *ACM Trans. Graph.* 32.4 (July 2013), 98:1–98:12 (cited on pages 4, 9).

[BLP+13] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. “Quad-Mesh Generation and Processing: A Survey”. In: *Computer Graphics Forum* 32.6 (2013), pp. 51–76 (cited on page 2).

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. “Mixed-integer Quadrangulation”. In: *ACM Trans. Graph.* 28.3 (July 2009), 77:1–77:10 (cited on pages 2, 4, 9, 10, 69).

[BSBK02] Mario Botsch, Stefan Steinberg, Stefan Bischoff, and Leif Kobbelt. “OpenMesh: A Generic and Efficient Polygon Mesh Data Structure”. In: *OpenSG Symposium 2002*. 2002 (cited on page 69).

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004 (cited on pages 52, 59).

[BBC+10] A. M. Bronstein, M. M. Bronstein, U. Castellani, A. Dubrovina, L. J. Guibas, R. P. Horaud, R. Kimmel, D. Knossow, E. von Lavante, D. Mateus, M. Ovsjanikov, and A. Sharma. “SHREC 2010: robust correspondence benchmark”. In: 2010 (cited on pages 11, 70).

[Bro11] Alexander M. Bronstein. “Spectral descriptors for deformable shapes”. In: *CoRR* abs/1110.5015 (2011) (cited on page 11).

[BBK08] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008 (cited on page 11).

[CBK15] Marcel Campen, David Bommes, and Leif Kobbelt. “Quantized Global Parametrization”. In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 192:1–192:12 (cited on pages 1, 9, 63, 69).

[CIE+16] Marcel Campen, Moritz Ibing, Hans-Christian Ebke, Denis Zorin, and Leif Kobbelt. “Scale-Invariant Directional Alignment of Surface Parametrizations”. In: *Comput. Graph. Forum* 35.5 (Aug. 2016), pp. 1–10 (cited on page 10).

[CK14] Marcel Campen and Leif Kobbelt. “Dual Strip Weaving: Interactive Design of Quad Layouts Using Elastica Strips”. In: *ACM Trans. Graph.* 33.6 (Nov. 2014), 183:1–183:10 (cited on page 10).

[CMI09] E. Chouzenoux, S. Moussaoui, and J. Idier. “A majorize-minimize line search algorithm for barrier function optimization”. In: *2009 17th European Signal Processing Conference*. Aug. 2009, pp. 1379–1383 (cited on page 58).

[DMK03] P. Degener, J. Meseth, and R. Klein. “An Adaptable Surface Parameterization Method”. In: *In Proceedings of the 12th International Meshing Roundtable*. 2003, pp. 201–213 (cited on pages 35, 36).

[DLL+10] T.K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang. “Persistent Heat Signature for Pose-oblivious Matching of Incomplete Models”. In: *Computer Graphics Forum* 29.5 (2010), pp. 1545–1554 (cited on page 12).

[DVPS14] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. “Designing N-PolyVector Fields with Complex Polynomials”. In: *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing)* 33.5 (2014), pp. 1–11 (cited on page 10).

[DVPS15] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. “Integrable PolyVector Fields”. In: *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 34.4 (2015), 38:1–38:12 (cited on page 10).

[EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. “QEx: Robust Quad Mesh Extraction”. In: *ACM Trans. Graph.* 32.6 (Nov. 2013), 168:1–168:10 (cited on page 9).

[ECBK14] Hans-Christian Ebke, Marcel Campen, David Bommes, and Leif Kobbelt. “Level-of-detail Quad Meshing”. In: *ACM Trans. Graph.* 33.6 (Nov. 2014), 184:1–184:11 (cited on page 10).

[ESCK16] Hans-Christian Ebke, Patrick Schmidt, Marcel Campen, and Leif Kobbelt. “Interactively Controlled Quad Remeshing of High Resolution 3D Models”. In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 218:1–218:13 (cited on pages 1, 4, 5, 10).

[Flo03] Michael S. Floater. “Mean Value Coordinates”. In: *Comput. Aided Geom. Des.* 20.1 (Mar. 2003), pp. 19–27 (cited on page 32).

[GLK16] Anne Gehre, Isaak Lim, and Leif Kobbelt. “Adapting Feature Curve Networks to a Prescribed Scale”. In: *Computer Graphics Forum* (2016) (cited on page 10).

[HG00] Kai Hormann and Günther Greiner. *MIPS: An efficient global parametrization method*. Tech. rep. DTIC Document, 2000 (cited on pages 35, 36).

[HAWG08] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. “Non-rigid Registration Under Isometric Deformations”. In: *Proceedings of the Symposium on Geometry Processing*. SGP ’08. Copenhagen, Denmark: Eurographics Association, 2008, pp. 1449–1457 (cited on page 12).

[JFH+15] Tengfei Jiang, Xianzhong Fang, Jin Huang, Hujun Bao, Yiying Tong, and Mathieu Desbrun. “Frame Field Generation Through Metric Customization”. In: *ACM Trans. Graph.* 34.4 (July 2015), 40:1–40:11 (cited on pages 6, 10).

[KZHC11] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-or. *A Survey on Shape Correspondence*. 2011 (cited on page 12).

[KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. *QuadCover – Surface Parameterization using Branched Coverings.* 2007 (cited on page 9).

[Kee13] Mathieu Desbrun Keenan Crane Fernando de Goes. “Digital Geometry Processing with Discrete Exterior Calculus”. In: *ACM SIGGRAPH 2013 courses*. SIGGRAPH ’13. Anaheim, California: ACM, 2013 (cited on page 35).

[KBLB12] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein. “Intrinsic shape context descriptors for deformable shapes”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 159–166 (cited on pages 11, 16).

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. “Least Squares Conformal Maps for Automatic Texture Atlas Generation”. In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 362–371 (cited on pages 36, 47).

[LWWS15] C. Li, M. Wand, X. Wu, and H. P. Seidel. “Approximate 3D Partial Symmetry Detection Using Co-occurrence Analysis”. In: *2015 International Conference on 3D Vision*. Oct. 2015, pp. 425–433 (cited on pages 11, 12, 17).

[LGB+11] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. “SHREC’11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes”. In: *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval*. 3DOR ’11. Llandudno, UK: Eurographics Association, 2011, pp. 79–88 (cited on page 12).

[LZC+15] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. “Non-rigid 3D Shape Retrieval”. In: *Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval*. 3DOR. Zurich, Switzerland: Eurographics Association, 2015, pp. 107–120 (cited on page 12).

[Lip12] Yaron Lipman. “Bounded Distortion Mapping Spaces for Triangular Meshes”. In: *ACM Trans. Graph.* 31.4 (July 2012), 108:1–108:13 (cited on page 36).

[LRBB17] Or Litany, Emanuele Rodolà, Alex M. Bronstein, and Michael M. Bronstein. “Fully Spectral Partial Shape Matching”. In: Eurographics Association, 2017 (cited on page 13).

[Low04] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110 (cited on pages 10, 16).

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008 (cited on page 19).

[MTP+15] Giorgio Marcias, Kenshi Takayama, Nico Pietroni, Daniele Panozzo, Olga Sorkine-Hornung, Enrico Puppo, and Paolo Cignoni. “Data-driven Interactive Quadrangulation”. In: *ACM Trans. Graph.* 34.4 (July 2015), 65:1–65:10 (cited on page 10).

[MK12] Jan Möbius and Leif Kobbelt. “OpenFlipper: An Open Source Geometry Processing and Rendering Framework”. In: *Proceedings of the 7th International Conference on Curves and Surfaces*. Avignon, France: Springer-Verlag, 2012, pp. 488–500 (cited on page 69).

[MW94] Walter Murray and Margaret H. Wright. “Line Search Procedures for the Logarithmic Barrier Function”. In: *SIAM Journal on Optimization* 4.2 (1994), pp. 229–246 (cited on page 58).

[MPZ14] Ashish Myles, Nico Pietroni, and Denis Zorin. “Robust Field-aligned Global Parametrization”. In: *ACM Trans. on Graphics - Siggraph 2014* 33.4 (2014), Article No. 135 (cited on page 70).

[MZ13] Ashish Myles and Denis Zorin. “Controlled-distortion constrained global parametrization”. In: *ACM Trans. Graph.* 32.4 (July 2013), 105:1–105:14 (cited on page 2).

[OD11] Stephen O’Hara and Bruce A. Draper. “Introduction to the Bag of Features Paradigm for Image Classification and Retrieval”. In: *CoRR* abs/1101.3354 (2011) (cited on page 19).

[OBBG09] M. Ovsjanikov, A. M. Bronstein, M. M. Bronstein, and L. J. Guibas. “Shape Google: a computer vision approach to isometry invariant shape retrieval”. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. Sept. 2009, pp. 320–327 (cited on pages 11, 19, 21, 22, 26).

[OBS+12] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. “Functional Maps: A Flexible Representation of Maps Between Shapes”. In: *ACM Trans. Graph.* 31.4 (July 2012), 30:1–30:11 (cited on page 13).

[OMMG10] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas Guibas. “One Point Isometric Matching with the Heat Kernel”. In: *Computer Graphics Forum* 29.5 (2010), pp. 1555–1564 (cited on page 12).

[PPTS14] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. “Frame Fields: Anisotropic and Non-orthogonal Cross Fields”. In: *ACM Trans. Graph.* 33.4 (July 2014), 134:1–134:11 (cited on pages 6, 10).

[PZKW11] Chi-Han Peng, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. “Connectivity Editing for Quadrilateral Meshes”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. SA ’11. Hong Kong, China: ACM, 2011, 141:1–141:12 (cited on page 5).

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. “Object Retrieval with Large Vocabularies and Fast Spatial Matching”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2007 (cited on page 19).

[RCB+15] Emanuele Rodolà, Luca Cosmo, Michael M. Bronstein, Andrea Torsello, and Daniel Cremers. “Partial Functional Correspondence”. In: *CoRR* abs/1506.05274 (2015) (cited on pages 13, 71).

[RCL+17] Emanuele Rodolà, Luca Cosmo, Or Litany, Michael M. Bronstein, and Alex M. Bronstein. “SHREC’17: Deformable Shape Retrieval with Missing Parts”. In: *EUROGRAPHICS Workshop on 3D Object Retrieval*. Eurographics Association, 2017 (cited on page 12).

[RL01] S. Rusinkiewicz and M. Levoy. “Efficient variants of the ICP algorithm”. In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152 (cited on page 12).

[SAPH04] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. “Inter-surface Mapping”. In: *ACM SIGGRAPH 2004 Papers*. SIGGRAPH ’04. Los Angeles, California: ACM, 2004, pp. 870–877 (cited on page 13).

[SSSC08] S. Shalom, L. Shapira, A. Shamir, and D. Cohen-Or. “Part Analogies in Sets of Objects”. In: *Proceedings of the 1st Eurographics Conference on 3D Object Retrieval*. 3DOR ’08. Crete, Greece: Eurographics Association, 2008, pp. 33–40 (cited on page 12).

[SBSC06] Andrei Sharf, Marina Blumenkrants, Ariel Shamir, and Daniel Cohen-Or. “SnapPaste: an interactive technique for easy mesh composition”. In: *The Visual Computer* 22.9 (2006), pp. 835–844 (cited on page 12).

[SS15] Jason Smith and Scott Schaefer. “Bijective parameterization with free boundaries”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 70 (cited on pages 13, 36, 40).

[SNB+12] Justin Solomon, Andy Nguyen, Adrian Butscher, Mirela Ben-Chen, and Leonidas Guibas. “Soft Maps Between Surfaces”. In: *Comput. Graph. Forum* 31.5 (Aug. 2012), pp. 1617–1626 (cited on page 13).

[SCGL02] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. “Bounded-distortion Piecewise Mesh Parameterization”. In: *Proceedings of the Conference on Visualization ’02*. VIS ’02. Boston, Massachusetts: IEEE Computer Society, 2002, pp. 355–362 (cited on pages 35, 36).

[SOG09] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. “A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion”. In: *Proceedings of the Symposium on Geometry Processing*. SGP ’09. Berlin, Germany: Eurographics Association, 2009, pp. 1383–1392 (cited on pages 11, 18).

[TPSS13] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. “Sketch-based generation and editing of quad meshes.” In: *ACM Trans. Graph.* 32.4 (2013), pp. 97–1 (cited on page 10).

[TPS14] Kenshi Takayama, Daniele Panozzo, and Olga Sorkine-Hornung. “Pattern-Based Quadrangulation for N-Sided Patches”. In: *Computer Graphics Forum*. Vol. 33. 5. 2014, pp. 177–184 (cited on page 10).

[Ter13] Tamás Terlaky. *Interior point methods of mathematical programming*. Vol. 5. Springer Science & Business Media, 2013 (cited on page 58).

[TB11] Aggeliki Tsoli and Michael J. Black. “Shape- and Pose-Invariant Correspondences Using Probabilistic Geodesic Surface Embedding”. In: *Pattern Recognition: 33rd DAGM Symposium, Frankfurt/Main, Germany, August 31 – September 2, 2011. Proceedings*. Ed. by Rudolf Mester and Michael Felsberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 256–265 (cited on page 13).

[Tut63] W. T. Tutte. “How to Draw a Graph”. In: *Proceedings of the London Mathematical Society* s3-13.1 (1963), pp. 743–767 (cited on page 32).

[VCD+16] Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen, and Daniele Panozzo. “Directional Field Synthesis, Design, and Processing”. In: *SIGGRAPH ASIA 2016 Courses*. SA ’16. Macau: ACM, 2016, 15:1–15:30 (cited on page 10).

[VTS04] JP. Vert, K. Tsuda, and B. Schölkopf. “A Primer on Kernel Methods”. In: *Kernel Methods in Computational Biology*. Cambridge, MA, USA: MIT Press, 2004, pp. 35–70 (cited on page 43).

[XKHK17] Kai Xu, Vladimir G. Kim, Qixing Huang, and Evangelos Kalogerakis. “Data-Driven Shape Analysis and Processing”. In: *Computer Graphics Forum* 36.1 (2017), pp. 101–132. ISSN: 1467-8659 (cited on page 11).