Mesh Parameterization for Feature-based Mesh Editing Applications

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Μηχανικών Η/Υ Εξεταστική Επιτροπή

από τον

Θεόδωρο Αθανασιάδη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

Οκτώβριος 2013

Figure 1: This research has been co-financed by the European Union (European Social Fund-ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund

Dedication

To my parents, Sotiri and Fotini,
and my sister, Amalia.

Acknowledgements

I would like to thank my supervisor Professor Ioannis Fudos for his valuable help, his advice and the patience he has shown during the elaboration of this work.

I would also like to thank Professor Christophoros Nikou for his help, suggestions and advice.

Furthermore, I would like to thank my friends Georgios Zioupos for his help in the mathematical analysis of planar parameterization and Iraklis Goudas for his help and valuable advice.

Finally, I would like to thank NVIDIA and the PUMPS 2011 evaluation commitee for awarding us a Tesla c2070 that we used extensively for the experiments in this thesis.

Table of Contents

List of Figures

List of Tables

Glossary

$A \cdot B$,$A \times B$	dot product of vectors, cross product of vectors
$\|\|X\|\|$	magnitude of vector X
sin^{-1},cos^{-1},tan^{-1}	inverse sine, inverse cosine and inverse tangent functions
X^T	vector transpose
$\frac{\partial P}{\partial u}$	first-order partial derivative of P with respect to u component

Abstract

Athanasiadis, Theodoros.
PhD, Department of Computer Science and Engineering, October 2013
Mesh parameterization for feature-based mesh editing applications
Supervisor: Ioannis Fudos
Mesh morphing is a technique for computing a smooth transition between two (or more) objects. Animation using deforming objects is frequently used in computer graphics for entertainment. Furthermore, smooth animation of nonrigid objects (e.g. articulated objects) can be accomplished by mesh morphing on a set of object snapshots.

In recent years, mesh morphing along with cut-and-paste techniques have started to find their way in the product design and optimization process. Nevertheless, new problems arise from the use of these techniques on these new fields. Unlike computer graphics were the primary objective is only the rendering of the results, these new applications require the intermediate results of the deformed models to be robust and respect form-features. Therefore, in this thesis we devise robust and fast techniques for the deformation of meshes which are based on mesh parameterization and respect form features. In a nutshell, we treat the problems of feature-based morphing, mesh segmentation and cut-and-paste design. Our final goal is to incorporate these techniques in real-time applications, therefore great emphasis is given on exploiting the power of modern graphic processing units and massively parallel systems.

θεόδωρος Αθανασιάδης του Σωτηρίου και της Φωτεινής.
Phd, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Οκτώβριος 2013.
3Δ Μετασχηματισμοί Μορφών Βασισμένοι σε Χαρακτηριστικά με Εφαρμογές στη Σχεδίαση με Υπολογιστή και στη Μηχανολογία
Επιβλέποντας: Ιωάννης Φούντος.

Ο 3Δ μετασχηματισμός μορφών (Morphing) είναι μία τεχνική δημιουργίας μιας ομαλής μετάβασης μεταξύ δύο ή περισσοτέρων αντικειμένων. Το animation με μετασχηματιζόμενα αντικείμενα χρησιμοποιείται συχνά στα γραφικά. Επιπλέον, ο 3Δ μετασχηματισμός μορφών μπορεί να χρησιμοποιηθεί και για την παραγωγή animation μετασχηματίζοντας διάφορα στιγμιότυπα του ίδιου αντικειμένου. Τα τελευταία χρόνια ο μετασχηματισμός μορφών μαζί με τεχνικές αποκοπής και επικόλλησης χαρακτηριστικών έχουν αρχίσει να εφαρμόζονται με επιτυχία στην σχεδίαση με χαρακτηριστικά και γενικότερα στις φάσεις σχεδίασης και βελτιστοποίησης προϊόντων. Ωστόσο στα νέα πεδία εφαρμογών αυτές οι μεθοδολογίες έρχονται αντιμέτωπες με μια σειρά από νέα προβλήματα. Σε αντίθεση με τις εφαρμογές στα γραφικά, ο στόχος πλέον δεν είναι μόνο το καλό οπτικό αποτέλεσμα, αλλά η ακρίβεια, η ευρωστία και το κατά πόσο είναι αξιοποιήσιμα τα ενδιάμεσα στάδια.

Με στόχο την εφαρμογή του μετασχηματισμού μορφών σε εφαρμογές μηχανολογίας και σχεδίασης με χαρακτηριστικά στα πλαίσια της εργασίας αυτής αναπτύσουμε γρήγορες και αποδοτικές μεθοδολογίες για μεταφορά παραμορφώσεων, αναγνώριση χαρακτηριστικών και 3Δ μετασχηματισμό μορφών βασισμένο σε χαρακτηριστικά. Μεγάλη έμφαση δίνεται στην αξιοποίηση των δυνατοτήτων των σύγχρονων καρτών γραφικών και των παράλληλων συστημάτων για την αποδοτική υλοποίηση των μεθοδολογιών καθώς τελικός στόχος είναι οι μεθοδολογίες αυτές να χρησιμοποιηθούν σε διαδραστικές εφαρμογές πραγματικού χρόνου.

Chapter 1

Introduction

1.1 Product design challenges

In product design, 3D models are often created in the early stage of product development, because such models are very effective for preliminary design evaluation by the development team. The evaluation of design concepts in the early stage helps products meet requirements for manufacturing, cost, safety, quality, maintenance, and so on. These geometric models are used as an intermediate object shared between the different groups involved in the design process. It is therefore crucial to use for the design, tools that enable fast and intuitive definitions and modifications of the product geometry.

To this end in recent years, feature-based editing approaches have become popular for the design of products in CAD systems. In Feature-based design the designer does not manipulate directly the surface but the features themselves. Form features with geometric meaning, such as holes, ribs, and slots can be manipulated and parameterized by numerical parameters. Thus, the product definition can rapidly change according to the modifications performed. In addition, form features from existing models can be used to assemble new engineering parts. However, form features do not support all the needs in terms of shape definition during the design process since analytic surfaces are unable to represent free form shapes that are widely used in engineering designs.

Another major challenge in product design is the optimization of design variables in order to meet specific requirements. The optimization is usually performed by specialized optimization software that usually takes as an input a *Finite Element Method* (FEM) mesh model derived from the original CAD design [122]. Nevertheless, every required mesh modification even the slightest one produces a cycle in the design process since it requires a return to the CAD system. This procedure is not only tedious, but sometimes not even possible since the various stages of the design process may be handled by different groups or even by different companies. Therefore, automated shape variation procedures are important to avoid cycles in the product design. One way to achieve slight mesh modifications is through morphing techniques. Consequently, morphing as a tool for small modifications in the optimization process has started to find its way in in commercial software [23].

It is therefore essential to devise new powerful methodologies for the design of models beyond traditional CAD editing of mechanical parts that will provide robust and efficient 3D mesh deformation and feature pasting that respects design constraints. To this end, our goal is to devise techniques with applications on feature-based morphing and cut-and-paste design that can be used to deform certain parts of a free-form model with respect to design constraints imposed by the product specifications.

Mesh morphing and cut-and-paste operations can enable the creation of different variants of mechanical components or assemblies and the ability to rapidly obtain an improved mesh without returning to the CAD system for slight changes. Common operations that need to be supported include local modifications on mesh parts, moving features along the contour of a surface, and snapping features to new target geometry. The optimization task can then be performed based on design variables, while following certain design constraints to achieve the final design objective. This way the model can be invoked in an optimization cycle where a multitude of valid models can be created for optimization. The optimization algorithm calculates new values for the design variables and the process is repeated until the optimal solution is found. The constraints to be satisfied may be surfaces that are required to precisely preserve their type and general shape or design parameters such as relative distances between surface areas. To this end two powerful techniques are necessary: mesh morphing and cut-and-paste editing.

Unfortunately, existing methods for morphing and cut-and-paste on free-form surfaces lack the ability to preserve features and constraints, which are required in engineering applications and are more oriented towards visual pleasing results. These techniques will be reviewed in detail in the following sections.

1.2 Preliminaries and related work

1.2.1 Features in product design

Features in a product model encapsulate the engineering meaning or significance of the geometry of portions of the product. By *features* we mean the generic shapes of a product which engineers can associate certain attributes and knowledge useful for reasoning about that product. These high-level modeling entities can be used to link the design rationale with the model. They can also be used to associate geometric and other constraints with the model in terms of high-level characteristics of the part modeled, and to organize constraint propagation after a design change. Hence *features* can be thought of as building blocks for the product definition and make the design process more efficient. It is therefore essential for engineering applications to preserve these *features* after every operation or deformation performed on the product model during the design process. There are various types of features that are used in product design, some examples are the following:

- *Form features.* They describe portions of a part's nominal geometry.
- *Tolerance features.* They describe geometry variation from the nominal form.
- *Assembly features.* They describe relationships between parts in a mechanical assembly.
- *Functional features.* Non geometric parameters related to function, performance etc.
- *Material features.* Material composition, treatment, condition etc.

In general, the design features can be classified according to the application in which they are used. Form features, tolerance features, and assembly features are closely related to the geometry parts and are called *geometric features.* Current features-based CAD systems mainly address geometric features, in particular form features and some kinds of assembly features and hence our work is mainly focused on form features.

A form feature is defined as a partial shape that has an engineering meaning, such as a round hole. A form feature contains both shape and parametric information. Shape information that describes the general form of the feature can be a set of curves and parametric surfaces. These curves and surfaces are often required to keep their original types, such as circles and cylinders for manufacturing and assembly reasons. Therefore, it is essential to maintain the curvature and the general shape of the form feature by defining a set of appropriate constraints to be satisfied during any deformation. Since in a mesh model a form feature consists of a subset of vertices of the original model, these constraints refer to constraints over the vertices during a deformation. In general, these constraints can be categorized into two groups: *soft constraints* that are approximately satisfied in the least squares sense and *hard constraints* that are precisely satisfied.

Some possible constraints can be:

- positional constraints, where vertices must be fixed at a certain constant position.
- curvature constraints, where the curvature normal is constrained.
- rigidity constraints, where the relative positions of pairs of vertices is constrained.

One important consideration when defining the form feature deformation behavior is how to avoid conflicting or redundant constraints. In this case, the system formed from the constraints can be very hard or even impossible to solve.

1.2.2 Cut-and-paste editing

Cutting and pasting are among the most common operations implemented by image drawing and manipulation software. These operations are a natural way to build complex images from individual components from various sources. In the context of mesh editing, Cut-and-paste editing extracts a characteristic feature form a source model and copies it to a target model. The user usually selects a surface region which is separated into the base surface and the detail surface, only the detail surface is used as a feature to be pasted. The detail surface can be stored either as a height-field or a parametric volume map [41]. The drawback of the height-field representation is that usually general features may be thick or have overhangs and can not be properly represented. To paste the detail surface to a target model, the corresponding vertices of the target model are moved based on the detail map. For 3D models, we should take into account that the smooth attachment of boundaries between a pasted model and its base is sometimes necessary. One possible way to resolve this issue is to perform a union operation between the two models and then apply a blending function along the boundaries of the features [86]. However blending functions for arbitrary meshes is a difficult problem to tackle.

Another approach [120],[108] is the modification of differential coordinates instead of directly changing spatial coordinates. The mesh geometry is then implicitly modified after reconstructing the surface from the differential coordinates. This method has the advantage of reducing deformation artifacts that may appear after the feature pasting.

Existing Cut-and-paste editing methods can be roughly categorized into two broad groups. The first approach uses mesh fusion to blend the source surface and target surface directly [58],[86]. The second one first extracts a base surface as a medium between the source surface and the target surface, and then transfers the details to the target surface via the base surface [41],[16]. The former pays more attention to the smoothness of the boundaries at the joint of the source and target surfaces. The latter focuses on the global deformation of the source surface according to the target surface. Base surface extraction is the most important step of the latter algorithms. On the one hand, it determines the geometry to be pasted, because the details to be pasted are defined as the difference between the feature surface and the base surface. On the other hand, it decides, to some extent, the type of surfaces that can be handled by the algorithm. This is an important factor since one fundamental problem with most of the aforementioned approaches is their inability to deal with non-zero genus features.

1.2.3 Morphing

Shape morphing is a technique that aims to generate a smooth sequence that transforms a source shape into a target shape. Although we have some quite efficient and effective methods for 2D morphing, the 3D case remains an open problem both in terms of feasibility and efficiency.

Existing methods for 3D morphing can be categorized into two broad classes: *volume based* or *voxel based* [74] and *mesh based* or *structural* [62] approaches. The volume-based approach represents a 3D object as a set of voxels usually leading in intensive computations. The mesh-based approach exhibits better results in terms of boundary smoothness and rendering since the intermediate morphs are represented as volumes and techniques such as marching cube [82] are employed to acquire the final polygonal representation used for rendering. Furthermore, most applications in graphics use mesh-based representations, thus making mesh-based modeling more broadly applicable. However, volume-based methods surpass the mesh based ones in that they can handle the morphing of very different topologies more easily, since volume to volume morphing is a lot similar to image morphing by means of treating voxels instead of pixels.

Although mesh morphing is more efficient as compared to volume-based morphing, it requires a considerable preprocessing of the two considered objects. Mesh morphing involves two steps. The first step establishes a mapping between the source and the target object (correspondence problem), which requires that both models are meshed isomorphically with a one-to-one correspondence. The second step involves finding suitable paths for each vertex connecting the initial position to the final position in the merged mesh (interpolation problem). For performing structural morphing, we can use *boundary representation (Brep)* or *surface representation* in which we represent each object by its surface description, or *volumetric or solid meshes*, for instance tetrahedral representations. In volumetric mesh morphing, it is much easier to maintain robustness and avoid the folding phenomenon. However, volumetric mesh morphing is computationally expensive as compared to surface mesh morphing since the number of elements in the former case is much larger in comparison to the latter case.

Most surface-based mesh morphing techniques employ a merging strategy to obtain the correspondence between the vertices of the input model. The merging strategy may be either automatic or user specified. Kent et al. [62] proposed an algorithm for the morphing of two objects topologically equivalent to the sphere. The algorithm works in two steps, first the two objects are mapped to a sphere and then the two projected topologies are merged. A common topology suitable for interpolation is created. The mapping presented is accomplished by a mere projection to the sphere and thus is applicable solely to star shaped objects.

The main problem with 3D parameterization techniques like [62] is how to find an appropriate mapping over the unit sphere for each of the morphed objects. Several techniques have been proposed to overcome this limitation inspired by physics. In [57] the authors use a spring system to model the mesh and gradually force the mesh to expand or

shrink on the unit sphere by applying a force field. Methods using springs do not always produce acceptable mappings especially when handling complex non convex objects.

[4, 5, 124] use a spring-like relaxation process. The relaxation solution may collapse to a point, or experience foldovers, depending on the initial state. Several heuristics achieving convergence to a valid solution are used.

[105, 94, 37] describe methods to generate a provably bijective parameterization of a closed genus-0 mesh to the unit sphere. The projection involves the solution of a large system of non-linear equations. A set of constraints on the spherical angles is maintained to achieve a valid spherical triangulation.

[103] uses a polyhedron realization algorithm that can transform any general polyhedron into a convex one which is isomorphic to the original. The realization consists of two phases, simplification and re-attachment. During the simplification phase, low valence vertices are detached from the vertex-neighborhood graph of the polyhedron one by one, and the corresponding graph is re-triangulated. This step is repeated until a 4-clique results. The second phase starts by first creating a tetrahedron and then the vertices are re-attached to the polyhedron, in the reverse order of their detachment, while maintaining the polyhedrons convexity.

[92] presents a similar method that first simplifies the surface mesh to a tetrahedron while creating a progressive mesh favoring triangles with good aspect ratio and then in similar way reattaches the vertices and simultaneously optimizes positions of the embedded vertices. The positions of the vertices are optimized to minimize a stretch metric.

[100] presents a method that directly create and optimize a continuous map between the meshes instead of using a simpler intermediate domain to compose parametrizations. Progressive refinement is used to robustly create and optimize the inter-surface map. The refinement minimizes a distortion metric on both meshes.

[68] also presents a method that relies on mesh refinement to establish a mapping between the models. First a mapping between patches over base mesh domains is computed and then mesh refinement is used to find a bijective parameterization. One advantage of this approach is that it naturally supports feature correspondence, since feature vertices are required as user input for the initial patch mapping.

[71] uses reeb-graphs and boolean operations to extend spherical parameterization for handling models of arbitrary genus. Each genus-n model is represented as a genus-0 positive mesh and n genus-0 negative meshes. Therefore $n+1$ spheres are required to parameterize these $n+1$ meshes independently, and thus to accomplish the spherical parameterization of genus-n models. Once a consistent embedding is computed for each model the positive meshes and the negative sets are paired. In the case where the number of negative meshes is not equal in the two models, extra pseudo negative meshes are generated to have an equal number of paired negative meshes. For each pair of meshes the morphing sequence is computed independently. Finally, boolean difference operation is applied to subtract each intermediate negative object from an intermediate positive object to obtain the morphing sequence. Existing methods for producing valid spherical

embeddings of genus-0 models can be integrated into their framework.

Another method that uses reeb-graphs for morphing topologically different objects of arbitrary genus is [59]. The method specifies the correspondence between the input models by using graph isomorphism theory. The super Reeb graph, which has the equivalent topological information to the Reeb graphs of the two input objects, is constructed and used to conduct the morphing sequence.

[70, 79] provide efficient techniques for morphing 3D polyhedral objects of genus-0. The emphasis of the method is on efficiency and requires definition of feature patches to perform 2D mapping and subsequent merging. Their method does not avoid self intersection and requires embedding merging and user intervention for mapping.

An interesting work for volume morphing is based on wavelets and presented in [47]. This is a promising approach whose principle could be applied to surface based morphing. This volume morphing technique yields rather slow algorithms which have time complexity $\Omega(n^3)$ where n is the size of the size of the volume representation.

1.2.4 Mesh fusion

Kanai et al. [58] present an approach to merge two meshes through 3D mesh-based morphing. The basic procedure of their approach is divided into two steps. In the first step, face correspondences are established between the two meshes by which each point on the face of source mesh is mapped to a point on the face of the target mesh. This step is called the *correspondence problem*. The second step, called the *interpolation problem* generates a smooth transition by interpolating corresponding points from the source to the target positions using those correspondences. To address the correspondence problem both the source feature and the target region must be homeomorphic to a disc to compute a harmonic map for each. The parameterizations are then mapped to each other and used to compute a merged topology with the combined topologies. Their main contribution is the development of several methods for resolving the interpolation problem. Further, they propose an algorithm based on three geometrical operations, *rigid transformation*, *scaling* and *deformation*, for adjusting the shape of the two feature boundaries to establish a smooth attachment. The final topology of the pasted feature is computed as a blend between the target geometry and the source geometry.

Museth et al [86] use *Level-set* models, which are deformable implicit surfaces that have a volumetric representation, to present a framework for editing operators. Their framework supports cut-and-paste by giving the ability to the user to copy,remove and merge models (using CSG operations) and automatically blend the intersecting regions. Since level set models are volumetric, the constructive solid geometry operations can be applied to them in a straightforward manner. Moreover, blending can be applied near the intersection by defining the region of influence based on the distance to the *intersection curve* shared by both input surfaces.

1.2.5 Implicit representations

Yu et al. [120] present a method for mesh merging,deformation and smoothing. Their approach is based on editing a Poisson-based gradient field. The distinctive feature of their approach is that the mesh geometry is modified implicitly through the gradient field manipulation. More specifically, the technique has tree components, a mesh solver based on the Poisson equation, a gradient field manipulation scheme using local transforms and a generalized boundary condition representation based on local frames. The theoretical foundation is that the Poisson equation is able to reconstruct a scalar function from a guidance vector field and a boundary condition. Thus, with these characteristics editing a function can be achieved by modifying its gradient field and boundary condition and a succeeding reconstruction with the Poisson equation. The surface is reconstructed by solving the least-squares system resulting from discretizing the Poisson equation with Dirichlet boundary conditions. The approach is able to perform merging of two meshes if a correspondence can be established between their open boundaries which serve as Poisson boundary conditions. The correspondence is not automatically established and requires user intervention. One advantage of their approach is that artifacts that can be introduced during deformation can be removed during reconstruction because least-squares minimization tends to distribute errors uniformly across the function.

Sorkine et al. [108] deformation approach is also based on the modification of differential coordinates instead of directly changing spatial coordinates. Their approach use the term *coating* to refer to the mixing of geometric details between two surfaces, and transplanting of a partial surface mesh onto another surface. The *coating* is defined as the difference between the original surface and a low-frequency band of the surface. Thus, coating transfer is the process of peeling the coating of a *source* surface and transferring it onto a *target* surface. Their surface representation is based on the Laplacian of the mesh, by encoding each vertex relative to its neighborhood. To apply their approach to meshes with different topologies a cross mapping is established by parameterizing the meshes over a common domain. To this end, they use the mean-value coordinate parameterization [33] for the mapping of patches homeomorphic to the disk over a unit square. To achieve the cross mapping a registration of the two parts in world coordinates is required. They do not describe how the feature in the source and target surfaces are defined, how they align the features and how they find the required correspondence of the feature boundaries required for the mapping. The vertices of the target surface are obtained by the interpolated Laplacians and the final mesh reconstruction requires solving a linear least-squares system.

1.2.6 Detail surfaces

Biermann et al. [16] present an approach to copy and paste relief features on multiresolution surfaces. In their approach they use semiregular multi resolution subdivision surfaces as their underlying pasting representation. Each surface is separated into two parts: the

base surface and the *detail* surface. The goal is to replace the detail part of the second surface with the detail part of the first. The separation is user-guided, a base surface is selected by the user along a single flatness parameter. Subsequently, a Least-Squares fitting procedure is employed to perform the separation. The relief feature details are encoded as a scalar displacement along the normal and a tangential displacement and a subdivision defines a smooth surface recursively as the limit of a sequence of meshes. Each finer mesh is obtained from a coarse mesh by using a set of fixed refinement rules such as Catmull-Clark subdivision rules. Multiresolution surfaces extend subdivision surfaces by introduction *details* at each level. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarse mesh. To achieve the feature transfer a parameterization is computed of the source and the target feature over the plane. The idea is to map each surface onto the plane as isometrically as possible and then align the two planar parameterizations, using a linear transformation to compensate for the distortion. Consequently, their approach is difficult to generalize to pasting regions with topology different from that of a subset of a plane. In addition it may result in higher distortion than a direct mapping from one surface to the other. Their approach supports real-time transfer of relier features that are homeomorphic to a disc.

Masuda et al. [41] present a cut-and-paste method based on constrained B-spline volume fitting. Their method is a volume-pasting approach, which paste a parametric volume instead of a height-field. The volume approach allows to deform and paste a feature in the volume even if it contains overhangs or handles. They first compute the pure *feature region* and the surrounding *context region* on the source surface based on user input. Then, the base surface is calculated by approximating the *context region*. For the separation process a global search segmentation method is employed, more specifically a maximum flow minimum cut problem is solved [60]. The base surface can be used to support a cut operation by replacing the *feature region*. To perform the feature transfer an initial parametric volume is defined so that it involves the entire feature region and the control points of the volume are optimized so that the bottom surface corresponds to the base surface. The parametric volume is used to parameterize the *feature region*. To paste a feature the base volume of the feature is deformed to fit the target model. Their method supports copy and paste of features with overhangs and non-zero genus and is able to avoid self-intersections of thick features.

Fu et al [39] also present a method that allows to cut and paste features of non-zero genus onto the target surface. The user first identifies the region of interest by selecting a set of points. A flood fill algorithm is then used in the closed polygon curve implicitly defined by the points selected to get the complete region. The source surface is automatically constructed according to the boundary information of the feature. To achieve this, the user selected boundary is triangulated and then mesh optimization techniques are used to remove degenerate triangles and adjust the density of the base surface. The triangulation is solved by using *dynamic programming* techniques with an $O(n^3)$ running time, with n the number of boundary vertices. However, the base surface is not guaran-

teed to be self-intersection free and assume that the selected boundary is triangulable. After extracting the base surface, an intrinsic parameterization [27] is computed to map the source feature onto the base surface. Then, the source surface is attached to the target surface, replacing the target region and the feature is reconstructed.

1.3 Thesis motivation and overview

The goal of this research is to devise robust and efficient mesh editing techniques that identify and respect form features. To achieve this, our first objective will be to tackle the problem of identifying form-features. After the features have been identified and matched, appropriate techniques will be used to perform cut-and-paste and morphing operations with respect to hard and soft constraints.

A key observation from the quick overview of related work in the previous section is that a lot of the methods rely directly or indirectly on some form of parameterization of the features or the meshes. Therefore, fast and robust parameterization methods are crucial for any mesh editing algorithm. To this end, our next goal is to improve and extend existing methods of mesh parameterization in order to incorporate constraints that are derived from the features of the meshes and satisfy our needs.

Finally, another goal of this work is to integrate the techniques devised in an interactive editing framework and to evaluate them in terms of efficiency and robustness. This is a challenging task and to achieve it we will use technologies and features provided by modern Graphic Processing Units (GPUs). Modern GPUs offer an impressive processing power in terms of floating point operations. Nevertheless, most GPUs are designed specifically for graphics and therefore are very restrictive in their programming model. Due to this, algorithms that need to be efficient in the GPUs must be redesigned by taking in consideration the way these units work. This poses a new set of challenges in designing and implementing efficient and robust algorithms.

1.4 Thesis contributions and highlights

As explained in the previous section the core of this thesis is mainly fast and robust parameterization methods with soft and hard constraints. The applicability of the mesh parameterization methods devised will be shown in a range of applications such as: mesh segmentation and feature detection, feature-based morphing, and cut-and-paste operations with constraints in real-time. To summarize, we can categorize the contributions in the following domains:

- Planar parameterization methods
- Spherical parameterization methods
- Feature based morphing

In the domain of planar parameterization this work makes the following technical and theoretical contributions:

- Establishes the relation between mesh smoothing and parameterization techniques and derives a simplified formulation for the *isometric* parameterization problem (section 2.2).
- Presents an efficient parallel implementation of a non-linear solver along with a number of heuristics that speed up substantially the parallel realization on modern hardware (section 2.3.2).
- Presents an iterative topological untangling process that solves efficiently the constrained parameterization problem (section 2.3).
- Demonstrates the applicability of the parallel solver on realizing the feature cut-and-paste design paradigm (chapter 5).

In the domain of spherical parameterization, this work makes the following contributions:

- Introduces a novel iterative quadratic solver for spherical mesh parameterization (section 3.3).
- Presents an efficient parallel implementation along with a number of heuristics that speed up significantly the parallel realization on modern hardware (section 3.4).
- Demonstrates the usefulness of the parallel mesh parameterization algorithm in several applications that exploit mesh morphology analysis (section 3.5).

Finally, in the domain of feature based morphing this thesis:

- Presents a feature preserving spherical parameterization process based on geometrically constrained optimization (section 4.3).
- Introduces an algorithm that captures high level geometric structure by building a feature region adjacency graph (section 4.5).
- Describes a novel feature matching technique that automatically aligns two solid objects and identifies a set of feature correspondence points (section 4.5).
- Introduces a feature guided optimized parameterization that is used to achieve smooth visual results in morphing between objects with structural similarities (sections 4.5, 4.6).

1.5 Structure of this thesis

The rest of this thesis is organized as follows. Chapter 2 offers theoretical background for mesh smoothing and establishes how it is related to planar parameterization. In this chapter we also describe the core of our constrained parallel solver for isometric parameterizations.

Chapter 3 offers some background material on planar parameterizations and spherical parameterizations that use reduction to the planar case. In addition our iterative quadratic solver for spherical mesh parameterization is presented. Furthermore, in this chapter our parallel implementation along with an experimental study and several heuristics that speedup the parallel realization on modern architectures is described. Finally, it presents applications of our technique on automated feature selection, mesh decomposition and similarity-based object retrieval.

Chapter 4 presents related work on 3D morphing and presents the spherical parameterization step of our approach. In addition, it briefly describes the efficient computation of the intersections among the polygons on the sphere and the calculation of the interpolation trajectory. Finally, it presents an experimental evaluation of our method and some visual morphing results and offers conclusions.

Chapter 5 presents an application of cut-and-paste design. In this application the solver for constrained planar parameterization described in chapter 2 is combined with modern GPU technologies such as the OpenGL tessellator unit to enable cut-and-paste operations on meshes in real-time.

Finally, chapter 6 offers conclusions.

Chapter 2

Planar parameterization

2.1 Introduction

The purpose of mesh parameterization is to obtain a piecewise linear map, associating each face of the mesh with a surface patch on the parameterization domain. The parameterization domain is the surface that the mesh is parameterized on. Since the geometric shape of the parameterization surface will typically be different than the shape of the original mesh, angle and area distortion is introduced. Maps that minimize the angular distortion are called *conformal*, maps that minimize area distortion are called *authalic*, and maps that minimize distance distortion are called *isometric*. In this work, we deal with constrained *isometric* planar parameterizations. These maps are central to a broad spectrum of applications such as texture mapping, mesh completion, morphing and deformation transfer.

An important goal of parameterization is to obtain bijective (invertible) maps. The bijectivity of the map guarantees that there is no triangle flipping or overlapping. This is an important guarantee for certain applications, especially in the presence of user defined constraints on the vertices. On a planar parameterization domain a map may exhibit local or global bijectivity. Local bijectivity is achieved when there are no local triangle flips in the local neighborhoods of the mesh, whereas global bijectivity is achieved when there is no global mesh overlapping. Generally, global bijectivity is harder to achieve. Nevertheless, for most applications local bijectivity is sufficient.

The existing planar parameterization methods can be classified into two categories (for an extensive survey see [52]) : (i) methods that solve only linear systems, for example [75],[27],[81] and (ii) methods that use some kind of non-linear optimization. Typical methods of the former category, especially the earlier ones, have no guarantee for local or global bijectivity and usually offer inferior results as compared to the latter. Nevertheless, they are usually very fast and can be useful even as an initial solution for non-linear approaches. For example, in [51] although the energy minimized is non linear, a linear system is solved to obtain an initial parameterization of the mesh on the plane.

Amongst the latter category, several methods use some form of constrained or unconstrained non-linear optimization. These methods either reformulate the problem (resulting in non linearity) [106],[63] or directly minimize a non linear energy term [51],[96]. An indicative example is the work of [106] where the parametrization problem is reformulated in terms of angles subject to a set of constraints that ensure planarity and triangle validity of the final parameterization. Another example is the work of [63] where the authors use a set of vertices of the mesh called *cone singularities* to absorb the Gaussian curvature so as to compute conformal parameterizations of meshes. This idea was further extended in [13] where the authors first determine automatically the location and the target curvatures of the singularities. They then proceed by solving a discrete Poisson equation on the mesh vertices to compute edge lengths and compute the final embedding using a linear least squares methodology based on the computed edge lengths. A related work is [109] based also on *cone singularities* where a non linear solver is used to minimize the corresponding metric and compute the final parameterization. Ma and Lin [83] proposed a flattening technique based on optimizing a non linear objective function that compares the edge lengths and the areas between the triangles on the parameterization space and the mesh. Their work was extended in [11] where the authors modified the energy function and improved the initial solution. Finally, another interesting work is [84] where the authors compute isometric parameterizations by first deriving a non linear deformation metric based on the linear theory of elasticity and then proceed by minimizing that metric using standard non linear conjugate gradient methods. A comparative study of the results of some of the above methods can be found in [10].

For practical applications there is usually an additional requirement to accommodate user defined or automatically imposed constraints on the vertices of the parameterization. Generally, these constraints can be categorized into two groups: *soft constraints* that are approximately satisfied in the least squares sense and *hard constraints* that are precisely satisfied. Methods based on energy minimization can support soft constraints by adding a quadratic term to the energy function that measures the distance between the vertices and the desired location. Nevertheless, for linear approaches the additional term usually breaks the guarantees for bijectivity even for parameterizations on convex domains [52]. Hard constraints are even more difficult to support. Some methods can be extended to enforce hard constraints by the use of *Lagrange* multipliers [27]. However, such methods do not guarantee parameterization bijectivity.

In a nutshell, many previous approaches employ non linear solvers for constrained or unconstrained non linear optimization targeted to conformal parameterizations ([106, 63, 109, 13]). Others use fast linear solvers (e.g. [81]) to obtain isometric parameterizations but fail to support constraints and local bijectivity. In this work, we deal with the problem of computing an isometric bijective planar parameterization of a mesh, subject to hard constraints. Additionally, soft constraints can be trivially supported due to the formulation of the problem.

2.2 Isometric parameterization

2.2.1 Mesh smoothing preliminaries

Before explaining the connection between the parameterization and the smoothing problem, we must define some element types. The definition of the term element here depends on the type of the mesh and is either an arbitrary polygon for surface meshes or a polyhedron for volume meshes. The three element types we define are: (i) the *physical* element which is obtained through a mapping, possibly with area and angle distortion, of a element of the original mesh on the parameterization space, (ii) the *reference* element which is constructed by placing one node at the origin and the other nodes at unit lengths along the cartesian axes, and (iii) the *ideal* element which depends on the desired properties of the final mesh (see [65], [66]).

Furthermore, we define two affine mappings. The first mapping from the *reference* element x_r to the *ideal* element x_i is defined as :

$$x_i = \mathbf{W}\, x_r \tag{2.1}$$

where matrix $\mathbf{W}$ is the edge matrix (Jacobian) of the *ideal* element. The second mapping from the *reference* element x_r to the *physical* element x is defined as :

$$x = \mathbf{A}\, x_r + x_0 \tag{2.2}$$

where matrix $\mathbf{A}$ is the edge matrix of the *physical* element and x_0 is the vector with the coordinates of the first vertex. The matrix $\mathbf{A}$ holds information about the volume (for polyhedra), the area, and the orientation of the *physical* element while x_0 controls its translation.

Based on the above definitions the shape matrix from the *ideal* to the *physical* element was defined in [65] as:

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-1} \tag{2.3}$$

and the associated barrier *shape* quality metric $(\eta_{shape}) : \mathbb{R}^{n\times n} \rightarrow \mathbb{R}$ as :

$$\eta_{shape} = \frac{\|\mathbf{S}\|_{\mathbf{F}}^2}{n \det(\mathbf{S})^{2/n}} \tag{2.4}$$

where for surface and volume meshes n is 2 and 3 respectively. The above metric can be used in an optimization process as an objective function to minimize over the vertices to obtain an optimal mesh. This quality metric assumes that each element has positive and non-zero determinants and consequently non-zero local area or volume. Furthermore, the barrier form is used to enforce positive Jacobian determinants to prevent folding. The mappings of a surface mesh triangle are depicted in Figure 2.1.

2.2.2 Shape matrix construction for conformal parameterization

As noted in the previous section the definition of the *ideal* element depends on the desired properties of the final mesh. Therefore to preserve the angles of a triangle of the original

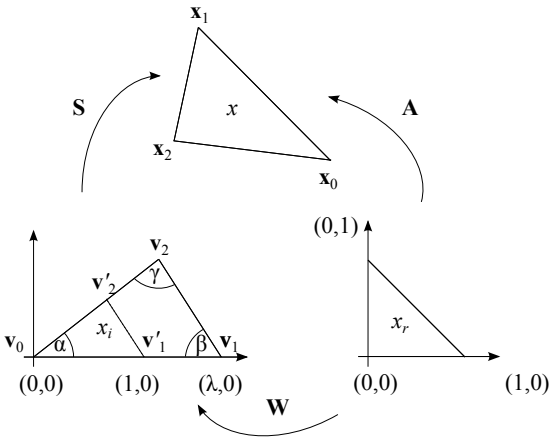


Figure 2.1: Ideal triangle $\triangle v_0 v_1 v_2$ and its similar triangle $\triangle v_0' v_1' v_2'$ on $\mathbb{R}^2$ along with the corresponding mappings.

mesh, we define on the parameterization space an *ideal* triangle $\triangle v_0 v_1 v_2$ with the same angles. Moreover for reasons that will become apparent, we define its similar triangle $\triangle v_0' v_1' v_2'$ with base $||v_0' v_1'|| = 1$ and $v_0' = v_0$ (see Figure 2.1) where:

$$\frac{||v_0' v_1'||}{||v_0 v_1||} = \lambda, \quad \lambda > 0 \tag{2.5}$$

with the use of basic trigonometry on $\triangle v_0' v_1' v_2'$ we may further define the coordinates of the point v_2' as :

$$v_{2x}' = \frac{\cot \hat{\alpha}}{\cot \hat{\alpha} + \cot \hat{\beta}} \quad , \quad v_{2y}' = \frac{1}{\cot \hat{\alpha} + \cot \hat{\beta}} \tag{2.6}$$

Therefore, from (2.5) and the definition of $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} v_{1x} & v_{2x} \\ v_{1y} & v_{2y} \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} 1 & v_{2x}' \\ 0 & v_{2y}' \end{bmatrix} = \frac{1}{\lambda} \mathbf{W}' \tag{2.7}$$

and from equations (2.6) and (2.7) :

$$\mathbf{W}^{-1} = \lambda \mathbf{W'}^{-1} \tag{2.8}$$

$$= \lambda \begin{bmatrix} 1 & -\frac{v'_{2x}}{v'_{2y}} \\ 0 & \frac{1}{v'_{2y}} \end{bmatrix} = \lambda \begin{bmatrix} 1 & -\cot\hat{\alpha} \\ 0 & \cot\hat{\alpha} + \cot\hat{\beta} \end{bmatrix} \tag{2.9}$$

it follows that :

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-1} = \mathbf{A}(\lambda \mathbf{W'}^{-1}) = \lambda \mathbf{A}\mathbf{W'}^{-1} = \lambda \mathbf{S'} \tag{2.10}$$

Matrices W' and S' are defined on the triangle $\triangle v'_0 v'_1 v'_2$. Moreover, using (2.10) the barrier *shape* metric (2.4) is defined as :

$$\begin{aligned} \eta_{shape} &= \frac{\|\mathbf{S}\|_{\mathbf{F}}^2}{\det(\mathbf{S})} \overset{(2.10)}{=} \frac{\|\lambda \mathbf{S'}\|_{\mathbf{F}}^2}{\det(\lambda \mathbf{S'})} \\ &= \frac{\lambda^2 \|\mathbf{S'}\|_{\mathbf{F}}^2}{\lambda^2 \det(\mathbf{S'})} = \frac{\|\mathbf{S'}\|_{\mathbf{F}}^2}{\det(\mathbf{S'})} = \eta'_{shape} \end{aligned} \tag{2.11}$$

showing that the barrier *shape* quality metric is scale invariant. Therefore, to obtain angle preserving parameterizations, only the angles of the triangles are required to compute the above matrices.

2.2.3 Connection with MIPS energy

An early method that supported free boundaries and aimed at computing a parameterization that minimized the *Dirichlet energy per parameter-space area* was Hormann and Greiner's MIPS method [51]. Since this energy is minimal for conformal mappings this gives parameterizations that are "as conformal as possible". To show how it relates to the *shape* quality metric that targets angle preserving mappings we start with the definition of $\mathbf{A}$:

$$\mathbf{A} = [\vec{v_1} - \vec{v_0}, \vec{v_2} - \vec{v_0}] = \begin{bmatrix} v_{1x} - v_{0x} & v_{2x} - v_{0x} \\ v_{1y} - v_{0y} & v_{2y} - v_{0y} \end{bmatrix} \tag{2.12}$$

and the shape matrix $\mathbf{S'}$:

$$\begin{aligned} \mathbf{S'} = \mathbf{A}\mathbf{W'}^{-1} &\overset{(2.12)}{=} [\vec{v_1} - \vec{v_0}, \vec{v_2} - \vec{v_0}]\mathbf{W'}^{-1} \\ &\overset{(2.9)}{=} [\vec{v_1} - \vec{v_0}, \vec{v_2} - \vec{v_0}] \begin{bmatrix} 1 & -\cot\hat{\alpha} \\ 0 & \cot\hat{\alpha} + \cot\hat{\beta} \end{bmatrix} \\ &= [\vec{v_1} - \vec{v_0}, (\vec{v_2} - \vec{v_1})\cot\hat{\alpha} + (\vec{v_2} - \vec{v_0})\cot\hat{\beta}] \end{aligned} \tag{2.13}$$

Furthermore, we have for $\det(\mathbf{S'})$:

$$\begin{aligned} \det(\mathbf{S'}) &\overset{(2.8)}{=} \det(\mathbf{A}\mathbf{W'}^{-1}) \\ &= \det(\mathbf{A})\det(\mathbf{W'}^{-1}) \\ &\overset{(2.9)}{=} (\cot\hat{\alpha} + \cot\hat{\beta})\det(\mathbf{A}) \end{aligned} \tag{2.14}$$

Finally, it can be shown (see Appendix) that :

$$\|\mathbf{S}'\|_{\mathbf{F}}^2 = (\cot\hat{\alpha} + \cot\hat{\beta})\big(c^2\cot\hat{\gamma} + a^2\cot\hat{\alpha} + b^2\cot\hat{\beta}\big) \tag{2.15}$$

Using (2.11),(2.14) and (2.15) we get :

$$\eta'_{shape} = \frac{1}{2}\frac{a^2\cot\hat{\alpha} + b^2\cot\hat{\beta} + c^2\cot\hat{\gamma}}{\det(\mathbf{A})} \tag{2.16}$$

From (2.16), we derive that the barrier shape quality metric is equal to the half of the MIPS energy [51].

2.2.4 Isometric parameterization

To obtain an area-preserving parameterization the area of each triangle on the parameterization space should tend to match its original area (E) on the mesh : $\det(A) \rightarrow 2E$. To measure this deviation a usual metric is [26]:

$$\frac{2E}{\det(\mathbf{A})} + \frac{\det(\mathbf{A})}{2E} \tag{2.17}$$

Therefore, a metric for the area preservation can be defined as:

$$\begin{aligned}\eta_{area} &\overset{(2.17,2.14)}{=} \frac{2E(\cot\hat{\alpha} + \cot\hat{\beta})}{\det(\mathbf{S}')} + \frac{\det(\mathbf{S}')}{2E(\cot\hat{\alpha} + \cot\hat{\beta})} \\ &\overset{(2.10)}{=} \det(\mathbf{S}) + \frac{1}{\det\mathbf{S}}\end{aligned} \tag{2.18}$$

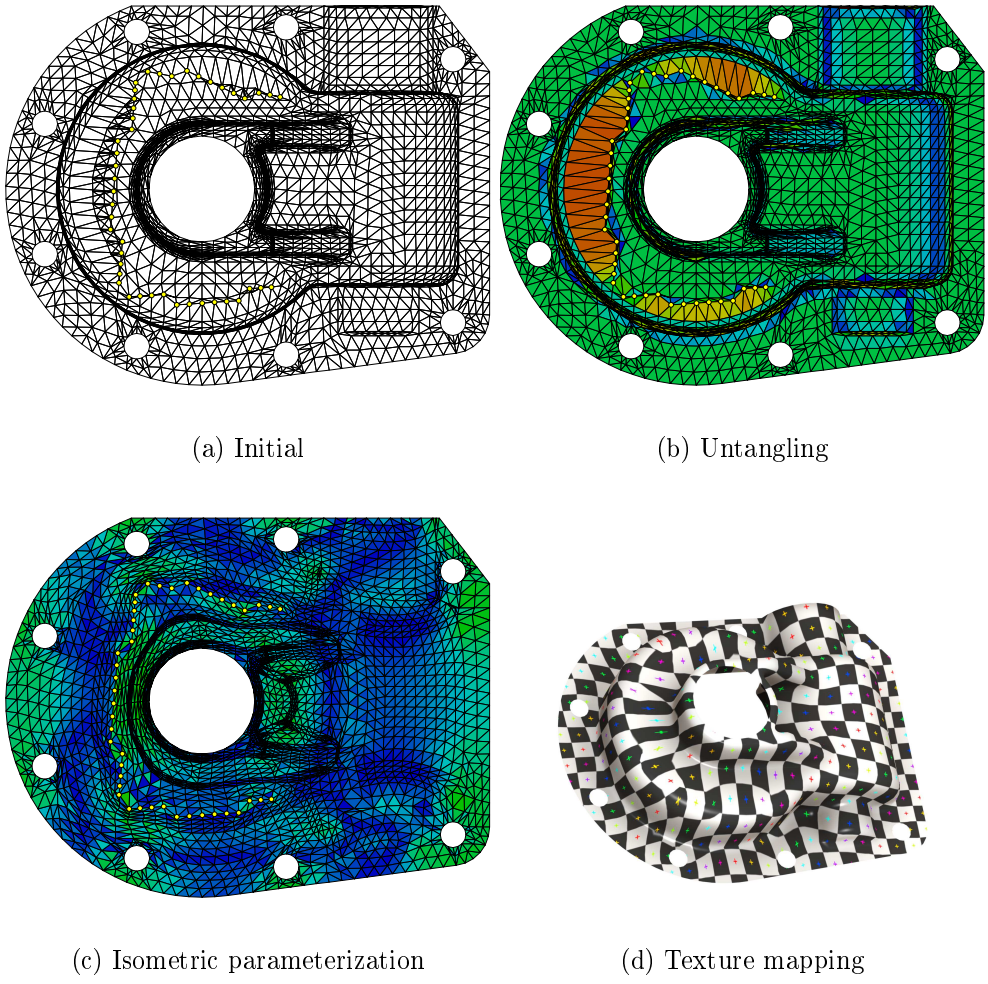
where we define for each *ideal* triangle $\frac{1}{\lambda} = \sqrt{2E(\cot\hat{\alpha} + \cot\hat{\beta})}$. Unlike the shape metric, η_{area} is not scale invariant. More specifically, the scale factor of each *ideal* triangle defines the desired area on the final parameterization. By combining the two metrics η_{shape} for shape preservation and η_{area} for area preservation, we get the simplified combined metric that targets isometric parameterizations :

$$\eta_{isometric} = \eta_{shape} \cdot \eta_{area} = ||\mathbf{S}|| + \frac{||\mathbf{S}||}{\det(\mathbf{S})^2} \tag{2.19}$$

Therefore, to define the $\mathbf{S}$ for each triangle i we need three scaling factors $[\lambda_i, \lambda_i\cot\hat{\alpha}_i, \lambda_i\cot\hat{\beta}_i]$ computed from the original mesh.

2.3 Constrained isometric parameterizations

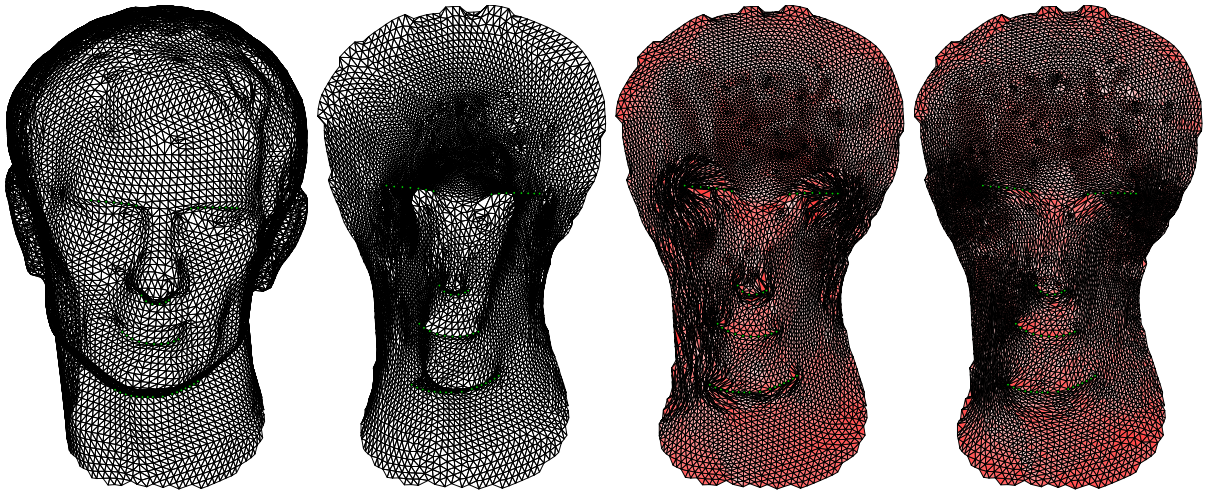
There are several alternatives in minimizing the non linear metric for isometric parameterizations. For example, in the works of [51] and [66] a non-linear solver is used in which each node is individually optimized based on the objective function. However, it is not always feasible to efficienlty parallelize such an approach due to the arising data



(a) Initial (b) Untangling

(c) Isometric parameterization (d) Texture mapping

Figure 2.2: Constrained parameterization of the Casting model [1]. Area deformation is also depicted (blue and red colors correspond to high and low distorted areas respectively).



(a) Initial (b) ABF++ (c) Untangled (d) Isometric parameterization

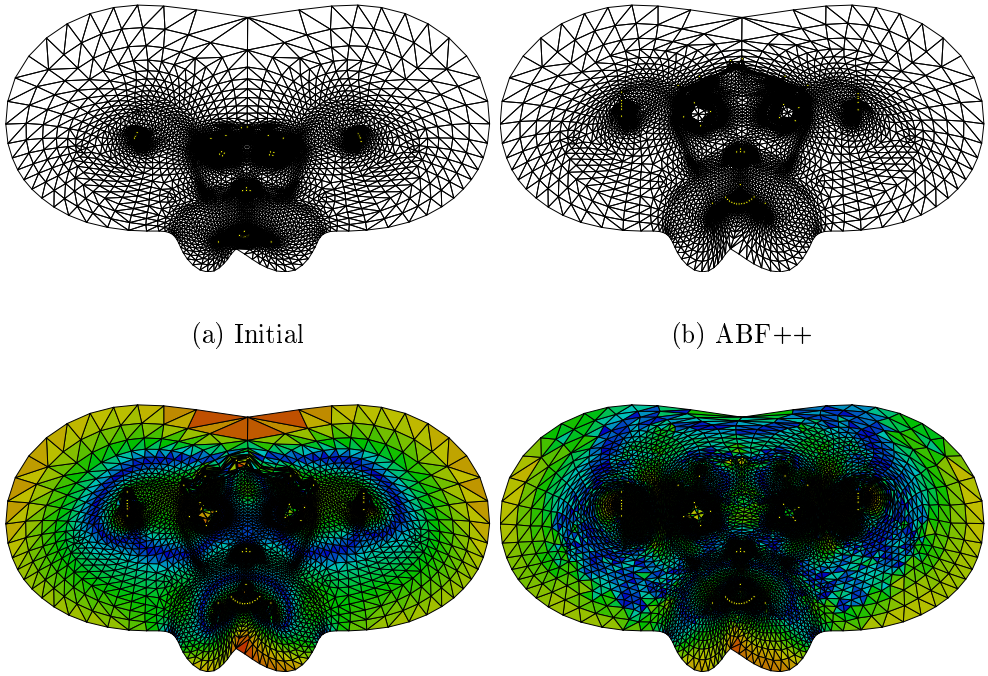
Figure 2.3: Constrained parameterization comparison. The constraints are : (i) the outer boundary and (ii) a set of internal nodes around the eyes and the nose area.

dependencies. For this reason, we have opted to use a preconditioned conjugate gradient approach and optimize all nodes simultaneously.

Conjugate gradient methods comprise a class of algorithms for unconstrained optimization. These methods have very low memory requirements and have linear convergence for most problems. An advantage of this class of algorithms is that only the objective function value and the gradient of the objective function are used during the optimization phase. Therefore, they do not require knowledge of the sparsity structure of the Hessian and are suitable for large scale optimization. Another important advantage of these algorithms is that it is possible to implement them with only BLAS-1 operations. There operations can be implemented very efficiently on modern hardware [89].

To obtain an initial solution to the parameterization problem we may use one of the established linear parameterization approaches. This can be easily done by using the standard parameterization techniques based on barycentric coordinates [112],[34]. The boundary vertices are mapped to the boundary vertices of a convex polygon with the same number of vertices and in the same order. Then, the interior vertices are placed in such a way that each vertex is the centroid of its neighboring vertices. Following this approach, there are two issues to take into account : (1) the shape of the boundary polygon and (2) how to map the boundary vertices to the polygon. For the boundary polygon usual choices are the unit circle and the unit quad whereas for the mapping usual approaches are parameterization methods such as the *chord length* or *centripetal* parameterization.

In this process, an important problem is that for most practical examples the bound-



(a) Initial (b) ABF++

(c) Untangled (d) Isometric parameterization

Figure 2.4: Constrained parameterization of the Suzanne model [18]. The constrained nodes consist of the outer boundary (86 nodes) and 45 internal nodes (Figure a) that are translated to new positions (Figure b).

ary vertices do not form a convex polygon and therefore the obtained parameterizations may not be bijective. Another issue is that if we use this parameterization as an initial solution for conjugate gradient this can be very far from the optimal solution resulting in slow convergence. A better alternative for computing an initial solution in our approach, is to use a parameterization technique such as [75],[27], or even [106] that minimize angular distortion. Naturally, in the presence of additional internal constraints and non convex boundaries the resulting mapping is not expected to be bijective. For example, as demonstrated in Figure 2.3(b), constraining a set of internal vertices and using the ABF++ method [106] results in a final parameterization that is not bijective.

2.3.1 Preconditioning

Non linear CG can be preconditioned by choosing an appropriate positive definite preconditioner matrix. Any matrix that approximates $\nabla^2 f(x^*)^{-1}$ is a good preconditioner for nonlinear functions. Therefore, a reasonable choice for such a matrix is the inverse of the diagonal of the Hessian matrix. Nevertheless, if x is far from a local minimum, the diagonal of the Hessian may not be positive-definite. Another possible CG preconditioning strategy is to compute an approximation to $\nabla^2 f(x^*)^{-1}$ generated by a quasi-Newton

Figure 2.5: Comparison of parameterization results for the Gargoyle model. The ARAP parameterization is non bijective in the highlighted area.

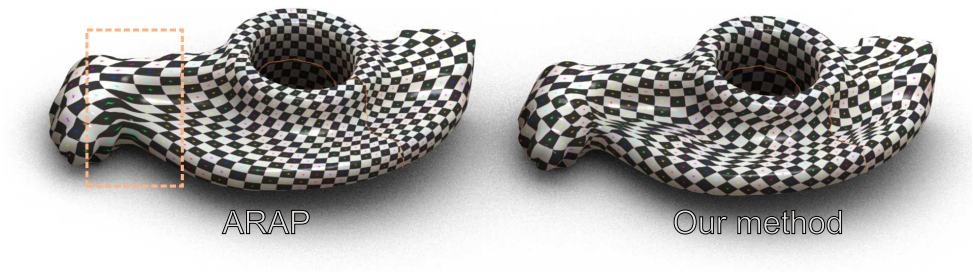


Figure 2.6: Comparison between the ARAP parameterization and our solver result. The highlighted region is not locally bijective.

update formula of the Broyden family [87], [80]:

$$H_{k+1} = v_k^T H_k v_k + \rho_k s_k s_k^T \tag{2.20}$$

where $\rho_k = 1/y_k^T s_k$, and

$$v_k = I - \rho_k y_k s_k^T \tag{2.21}$$

here, $s_k = x_{k+1} - x_k$, $g_k = \nabla f(x_k)^T$, and $y_k = g_{k+1} - g_k$. The above mentioned approach is known as Limited-memory BFGS (L-BFGS) and is useful for solving large problems since this method maintains a simple and compact approximation of the Hessian matrices. Therefore, it is suitable when the Hessian is dense or the second derivatives are costly to compute. A modified version of H_k is stored implicitly, by storing a certain number (m) of the vector pairs (s_i, y_i) that are used in (2.20) and (2.21). Figure 2.7 demonstrates the convergence of the BFGS preconditioned nonlinear CG, with Hestenes and Stiefel [48] update formula and different choices of m. The basic Hessian matrix H_k^0 plays an important role in the performance and the robustness of the algorithm. A popular choice

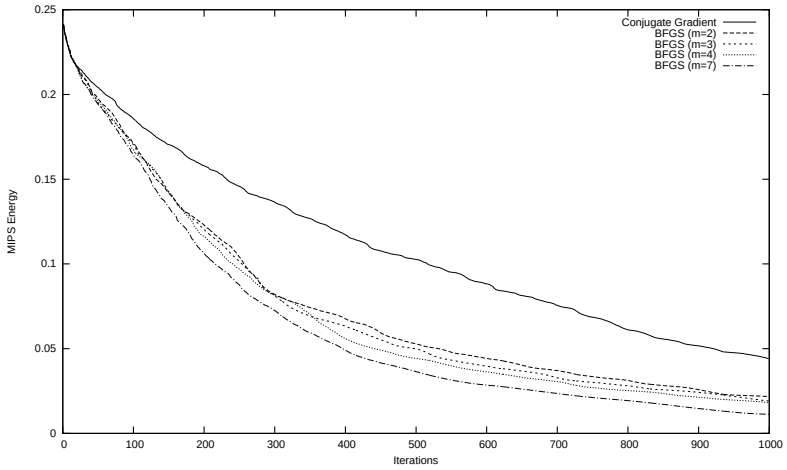


Figure 2.7: Comparison of the Conjugate Gradient convergence with and without preconditioning.

is to set $H_k^0 = \gamma_k I$ where γ_k is a scaling factor. The scale factor generally is used as an estimate of the size of the true Hessian matrix to ensure that the search direction is well scaled, and as a result the full step length can be accepted (usually if it satisfies the *Wolfe* conditions [117]). Moreover, the scaling prohibits the eigenvalues of the approximate Hessian from becoming large. There are several approaches in the literature for computing γ_k, such as those presented in [102],[80] and [3]. In practice, we found the scaling factor proposed by [3] to perform the best.

2.3.2 Parallel implementation and results

As an API for our implementation, we have used OpenCL 1.1. The core of our solver is the L-BFGS method described in [87] and the software is available at http://www.cs.uoi.gr/ fudos/smi2013.html. Algorithm 1 summarizes the basic steps of the solver. To maximize

1: $d_0 = -H_0 g_0$
2: $\beta_k = \frac{y_{k-1}^T g_k}{y_{k-1}^T d_{k-1}}$
3: $d_k = -H_k g_k + \beta_k d_{k-1}$
4: $x_{k+1} = x_k + a_k d_k$

Algorithm 1: Preconditioned Conjugate Gradient

the performance of our implementation, we have considered a number of factors. Two important considerations in modern GPUs are: (i) the efficient memory usage so as to achieve maximum memory bandwidth and (ii) tuning the instruction usage so as to achieve the maximum instruction throughput [88]. Therefore, we have used a number of heuristics to optimize the parallel performance of our non linear solver such as :

- Reduction of the divergence of the parallel kernels.

- Coalesced memory operations.
- Exact line searches.

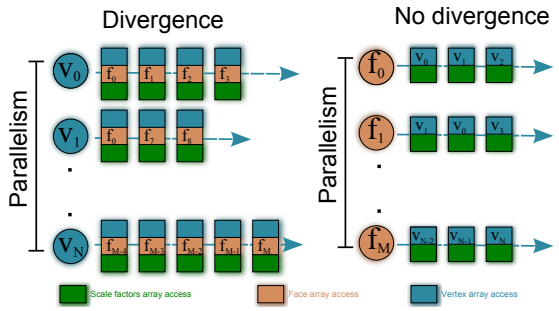


Figure 2.8: The vector ∇f can be computed in parallel for each vertex with indirect memory access and execution divergence (left) or for each face with more coalesced memory transactions and without divergence (right).

The costlier step in the solver is the computation of the gradient ∇f. The gradient can be computed per vertex, where each computation should access the adjacency list of the vertex and the corresponding scaling factors of the adjacent faces. If we use this straightforward approach to parallelize the computation, two problems arise on massively parallel platforms: random memory access patterns and divergence due to the different adjacency lists. In our implementation, we follow a two step approach to tackle these problems. First, we use an auxiliary buffer where we store the contribution of each face to its adjacent vertex gradients and then we add up the corresponding partial contributions to obtain the final gradient vector. This process is an efficient way to parallelize the computation on the faces (see Figure 2.8). Each face accesses only its three scaling factors and its vertices; consequently there are fewer levels of indirection while most of the read accesses are coalesced. The summation step still requires to randomly access the auxiliary buffer to compute the final gradient, but the most computationally intensive first part of the gradient computation is accelerated substantially.

The other major performance issue in the performance of conjugate gradient methods is the line search, which requires sufficient accuracy to ensure that the search direction yields a descent. The line search is typically performed in two stages: a bracketing phase where we constrain the desirable step length, and an interpolation phase that computes the step length within this bracketed interval. Ideally, we would like to find the global minimizer for f. However, this approach usually requires many evaluations of the objective function f and possibly the gradient ∇f. Therefore, there is a trade-off between the accuracy of the line search and the computational cost. A common approach is to perform inexact line searches until some accuracy criteria are met. Common criteria are the *Wolfe* conditions [117].

The usual "strong" *Wolfe* condition requires an extra evaluation of the gradient of the function. This approach performs well for applications running on platforms with only

a few threads like CPUs. On the contrary, the evaluation of the gradient on massively parallel platforms is relatively expensive. Even though we improved the speed of this computation with the previously described approach, due to the complexity of the gradients the evaluation of ∇f on such platforms is still an order of magnitude slower than the evaluation of f. Thus, in many cases it is more efficient to carry out more function evaluations instead of using the "strong" *Wolfe* condition to stop the line search process prematurely. A related issue is the scaling of the Hessian matrix, the scaling ensures that the search direction is well-scaled and that the full step ($a_k = 1$) is accepted in most iterations.

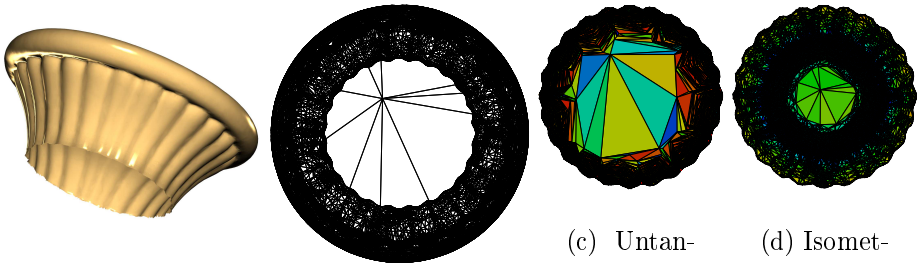
Having made the above observations, we have opted for a hybrid approach. First we scale the Hessian and check the strong *Wolfe* condition with the full step. If the step is rejected, we perform an exact line search using the derivative free *Brent* method [21] having as a limit the square of the hardware double accuracy. This is also the accuracy limit for the specific line search method. Since the *Brent* method needs a bracketing triplet (x_a, x_b, x_c), satisfying $f(x_b) < f(x_a)$ and $f(x_b) < f(x_c)$, in the case of the isometric metric, we take into account the discontinuity of the function when an element becomes inverted. This discontinuity occurs along the search direction d_k using a negative step. To avoid this case, and assuming that the line search begins with a bijective mapping, we find a low bound ϵ where the function is defined. To compute ϵ, we start from a small value and follow a procedure similar to the backtracking line search approach. This approach worked reliably in our experiments; the initial bracketing phase typically costs three or four function evaluations whereas the *Brent* line search method usually costs less than ten function evaluations.

Figures 2.5 and 2.6 illustrate the parameterization of two meshes from [1] with our solver and show a comparison with the ARAP method that also targets isometric parameterizations [81]. In both the cases the ARAP method failed to produce bijective parameterizations.

Furthermore, we have obtained performance results minimizing the isometric metric and using an NVIDIA Tesla c2070 and an Intel i7-2600k processor. The Tesla c2070 contains 448 CUDA cores delivering 515 Gigaflops of double-precision peak performance whereas the i7 processor contains 4 cores and delivers less than 70 Gigaflops of double precision peak performance. Table 2.1 illustrates the scalability of the solver while running a fixed number of iterations with different level of detail whereas Table 2.2 and Tables 3.3,2.4 provide a comparison with the publicly available parameterization software of [81] in terms of running times and parameterization quality.

2.3.3 Parameterization with hard constraints

Computing a parameterization with positional constraints is a difficult research problem. Traditionally, planar parameterization methods support such constraints. For example Levy et al [75] incorporates soft and hard constraints in the linear system formed while Desbrun et al [27] use Lagrange multipliers to add positional constraints. However in the



(a) Feature

(b) Planar projection

(c) Untangling

(d) Isometric

Figure 2.9: Constrained boundary parameterization (a) Original feature (b) Planar projection (not bijective) (c) Untangled mesh (with area deformation) (d) Further optimized isometric parameterization (also showing area deformation).

presence of a lot of constraints, these methods can fail to compute a bijective parameterization even if such a parameterization is known to exist. The inherent problem is that the bijectivity of the resulting map is based on certain properties of the energy minimized. Thus, adding linear terms in the system formed for soft constraints or removing variables for hard constraints can break theoretical guarantees for bijectivity. More recent methods methods such as [106] also support positional constraints but bijective parameterizations are not guaranteed (Figures 2.3,2.4). Other methods proceed by adding Steiner vertices to ensure the bijectivity [29] or compute a parameterization by partitioning the mesh into patches that are then parameterized while maintaining the smoothness and continuity between them [69].

When the boundary vertices are fixed, the constrained parameterization problem is equivalent to mesh untangling. Existing methods for untangling employ geometric or optimization-based approaches. For example in [36] the authors seek to maximize the minimum element area and formulate the mesh untangling problem as a series of local linear programming problems. Knupp et al [64] optimize a global function that measures the difference between the absolute and signed element area. The latter requires a custom solver and a modified line search approach since the gradient of that function is not continuous.

Another approach to handle folded meshes is to modify the shape quality metric η_{shape} so as to incorporate an untangling process as suggested by [31]. Having established the connection between the parameterization and the shape metric based smoothing, we may use this method with the appropriate isometric equations of section 2 and solve the resulting convex problem. In our experiments this approach worked reasonably well for small to medium sized meshes. Nevertheless, it further requires the solution of a series of non linear problems. More importantly, this objective function can take extreme values on large meshes and consequently it is very difficult to optimize it effectively. For the above reasons we follow a different two step approach. First, we consider the

boundary of the parameterization along the constraints imposed fixed and we treat the constrained parameterization problem as an untangling problem which is solved with simple topological operations. Afterwards, we proceed by optimizing the untangled map with the non linear solver in order to improve the quality of the parameterization. For such an approach to work in the case of free boundaries a good initial solution should be available. In this case of free boundaries we use the method of [106] to obtain this solution.

To derive the topological operator per vertex we formulate the untangling problem as a minimization problem on the inverted elements:

$$min f(x) = -\sum_{i}^{n} \det(\mathbf{A}) \tag{2.22}$$

For the above function if we compute the gradients of a triangle with vertices v_0,v_1,v_2 and edges e_0,e_1,e_2 with respect to vertex v_0 we obtain:

$$\begin{aligned} \det A &= (e_{0y} \cdot e_{2x} - e_{0x} \cdot e_{2y}) \Rightarrow \\ -\frac{\partial \det(A)}{\partial v_{0x}} &= -(v_{2y} - v_{1y}) = -e_{1y} \qquad (2.23) \\ -\frac{\partial \det(A)}{\partial v_{0y}} &= (v_{2x} - v_{1x}) = e_{1x} \qquad (2.24) \end{aligned}$$

This means that the gradient of the negative triangle area with respect to one of its vertices equals the opposite triangle edge rotated by $\frac{\pi}{2}$ clockwise about the triangle normal. If we sum all the gradients of the adjacent triangles this gives a descent direction and therefore we can move all the vertices along the corresponding lines:

$$v_i^{k+1} = v_i^k + \delta \sum_{j \in N_i} CW_{\frac{\pi}{2}}(e^{i,j}_{opposite}) \tag{2.25}$$

where N_i is the set of adjacent inverted triangles around vertex v_i, and $e^{i,j}_{opposite}$ is the edge opposite to v_i in triangle j. CW stands for the clock wise rotation transformation. Unfortunately, if we only apply the above operator there are numerical instability issues. The problem is directly related to the invariance of the area deformation under shears [26]. A shear does not change the area of a triangle and therefore extremely sheared triangles may occur during the minimization process. The problem becomes severe when the total area of the triangles around a vertex is very small and the corresponding valley has almost collapsed. In that case the edge lengths may become arbitrarily large especially if we use a bad initial solution. For this reason, we added a correction term that also reduces the shearing of a triangle when the area of that triangle drops below a certain threshold. This correction term is a vector perpendicular to the gradient of the area and the orientation depends on the angles of the triangle as illustrated in Figure 2.10. The shear direction is perpendicular to the gradient and the addition of this term preserves the descent property of the total operator. The addition of this term topologically "bends"

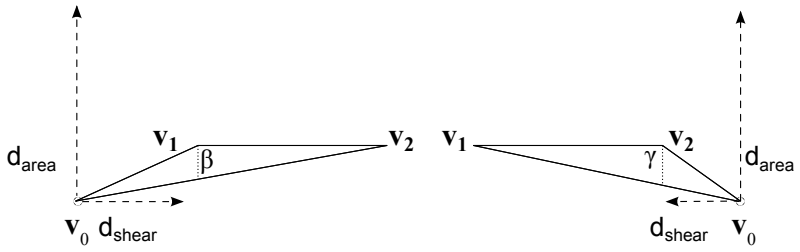


Figure 2.10: Shear operator for two different triangle configurations. The shear operator is parallel to the opposite edge and the direction depends on which of the two opposite angles (β,γ) is greater than $\frac{\pi}{2}$. If both angles are acute then no shear operator is applied.

the descent direction of the vertex towards the centroid of the valley resulting in reduced edge lengths.

Algorithm 2 summarizes the basic steps of the untangling process. where the factors

$\delta \leftarrow \frac{1}{2}$
for $k = 0$ to convergence or $\delta < \epsilon$ **do**
  for $i = 0$ to N **do**
    $d_{area} = \sum_{j \in N_i} CW_{\frac{\pi}{2}}(e^{i,j}_{opposite})$
    $d_{shear} = \sum_{j \in N_i} factor_j \cdot shear(e_{opposite})$
    $v_i^{k+1} = v_i^k + \delta(d_{area} + d_{shear})$
  end for
  if $\sum |\det(\mathbf{A}^{k+1})| \geq \sum |\det(\mathbf{A}^k)|$ **then**
    $\delta \leftarrow \frac{\delta}{2}$
  end if
  $k = k + 1$
end for

Algorithm 2: Untangling Process

are defined based on a user defined minimum area β as :

$$factor_j = \begin{cases} 1 - \frac{|\det(A_j)|}{\beta} & , -\beta \leq \det(A_j) \leq \beta \\ 0 & , otherwise \end{cases} \tag{2.26}$$

This way we move all the vertices along the descent direction according to a δ value without performing an exact line search. We have avoided exact line searches for two reasons. First, the gradient computations are affordable, unlike the non linear functions of the previous section, and therefore it is better to perform more gradient than function evaluations. Second, in this way we only need local topological operations that are more robust arithmetically. The latter is important especially for modern GPUs since it allows us to use only single precision operations that reduce the memory bandwidth requirements and are much faster than the corresponding double precision operations. Figures 2.2,2.3,2.9,2.4 and 2.11 illustrate some untangling results that are used as an intermediate

step for computing isometric parameterizations. The running time of the untangling process depends on the number of constraint vertices and the initial solution. For example, for the mesh of Figure 2.3 with 13095 faces and 241 constrained vertices the time for the untangling process was 100ms on a Tesla c2070. In all the experiments performed, the running time is primarily affected by the non linear optimization step that follows the untangling process. For typical meshes of up to 50k triangles, with hundreds of constraints, the average time for untangling was less than a second.

Table 2.1: Numerical results for different levels of detail while running a fixed number of iterations (=1000) on the Blech mesh ($Area(rms) = 0.490457$ and $Angular = 0.101195$).

Level	# vertices	# faces	i7 (ms)	c2070 (ms)	Speedup	$Area(rms)_{1000}$	$Angular_{1000}$
Lod1	1815	3456	999	1072	0.93	0.114456	0.0058238
Lod2	7085	13824	1704	1196	1.42	0.114835	0.0052326
Lod3	27993	55296	4708	1744	2.69	0.142656	0.0069087
Lod4	111281	221184	18156	3322	5.46	0.319943	0.0464538
Lod5	443745	884736	73828	9795	7.53	0.452182	0.0741808

Table 2.2: Time comparison between our solver and ARAP

method	model	# vertices	# faces	bijectivity	iters	# evals f	time(s)
ARAP	Blech	27993	55296	yes	7	-	1.95
Solver(1000)	Blech	27993	55296	yes	1000	2633	1.65
Solver(1500)	Blech	27993	55296	yes	1500	3658	2.38
ARAP	gargoyle	24406	48672	no	4	-	1.08
solver(5000)	gargoyle	24406	48672	yes	5000	15570	10.16
solver(10000)	gargoyle	24406	48672	yes	10000	30551	19.88
ARAP	julius	209083	416286	yes	28	-	59.02
solver(2500)	julius	209083	416286	yes	2500	5583	14.71
solver(4000)	julius	209083	416286	yes	4000	8943	23.92

2.4 Summary

We presented an efficient parallel scheme to compute *isometric* parameterizations subject to soft and hard constraints. Our approach is based on establishing a theoretical connection between the well studied non linear mesh smoothing and the *isometric* parameterization. Using this scheme, we have successfully carried out a large number of experiments to validate the solver on parameterizing meshes up to one million triangles in a few seconds on a modern GPU. Finally, as an application of our solver, we have parameterized and stored free form features on floating point textures and then by exploiting the capabilities of the tessellation unit on modern GPUs

Table 2.3: Comparison of quality between our solver (number of iterations in parenthesis) and ARAP

Method	Model	L2(min)	L2(max)	L2(rms)
ARAP	Blech	0.719205	1.35083	0.081242
Solver(1000)	Blech	0.721571	1.39323	0.062888
Solver(1500K)	Blech	0.708662	1.48147	0.057315
ARAP	Gargoyle	0.408068	7465.28	74.4895
Solver(5000)	Gargoyle	0.18212	25.9687	3.81626
Solver(10000)	Gargoyle	0.226868	16.4789	2.36491
ARAP	Julius	0.63778	56.1057	0.203055
Solver(2500)	Julius	0.575291	2.03919	0.155464
Solver(4000)	Julius	0.618229	2.00369	0.130395

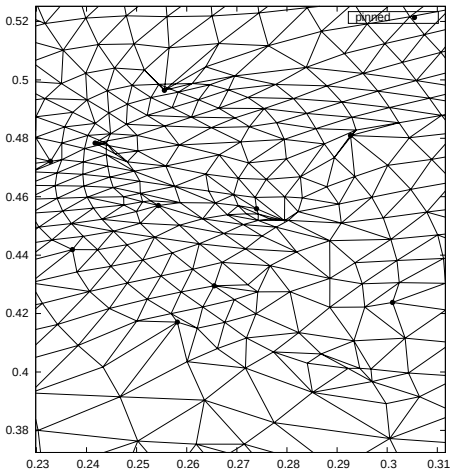
Table 2.4: Comparison of quality between our solver (number of iterations in parenthesis) and ARAP

Method	Model	Area(min)	Area(max)	Area(rms)	Angular
ARAP	Blech	0.565892	2.0186	0.15528	0.005267
Solver(1000)	Blech	0.595902	4.4074	0.13110	0.008731
Solver(1500)	Blech	0.574745	4.4745	0.12180	0.005351
ARAP	Gargoyle	0.000164	8.2565	1.03866	0.421756
Solver(5000)	Gargoyle	0.005277	31.9465	1.78979	0.108743
Solver(10000)	Gargoyle	0.010048	22.0854	1.63601	0.121859
ARAP	Julius	0.004867	4.269	0.34954	0.043286
Solver(2500)	Julius	0.299203	8.559	0.32586	0.041716
Solver(4000)	Julius	0.315212	6.409	0.28318	0.045691

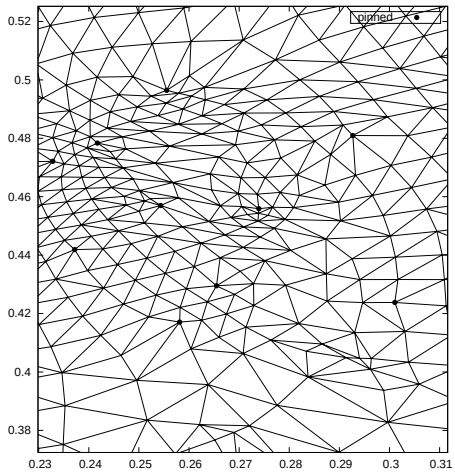
we have supported cut and paste operations in real time. This procedure is described in details in chapter 5.

A possible extension of our work would be the usage of a hierarchical decimation scheme similar to [51] or [106] to accelerate the convergence of the non-linear solver. Going a step further, we could exploit the connection between the parameterization and the smoothing problem to compute the *isometric* parameterization of volume meshes on 3D domains. By doing so, more properties of the original mesh could be preserved such as the volume preserving morphing and blending operations.

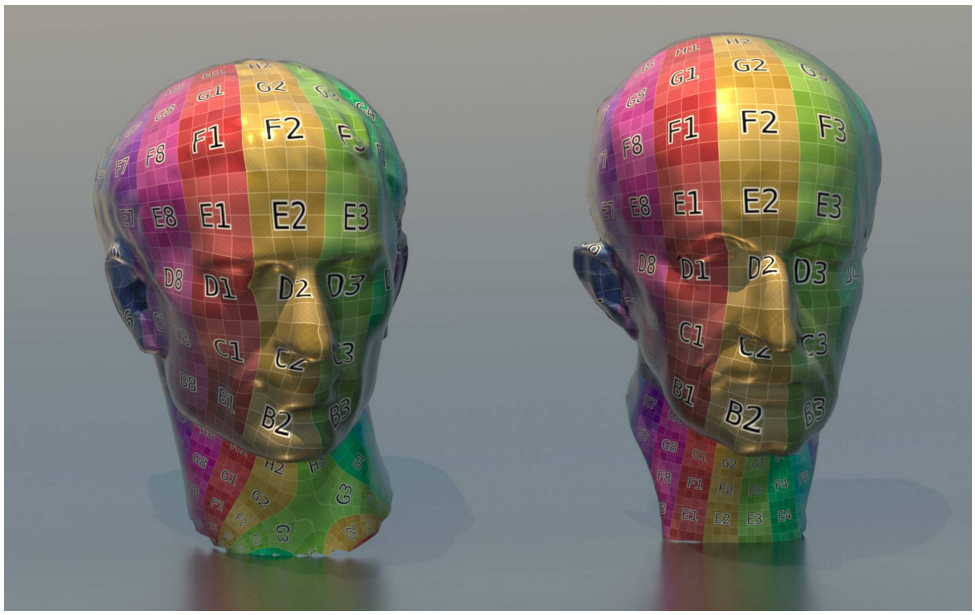
Another direction for future research is to provide a formal proof regarding regarding the convergence of the untangling process. This could be reached from connecting our process to standard gradient descent approaches and analysing the numerical instabilities that occur near the solution.



(a) ABF++ Detail



(b) Optimized Detail



(c) Final Mapping

Figure 2.11: Mapping the Julius model to the plank model. We first parameterize the plank model (right) with ABF++ and we pin the vertices around the ears and eyes of the Julius model (left) at the corresponding positions of the plank parameterization. (a) The ABF++ parameterization if folded. (b) The final unfolded parameterization is locally bijective (c) Detail of the final parameterization.

Chapter 3

Spherical parameterization

3.1 Introduction

Fast mesh parameterization is central to many applications such as remeshing, filtering, texture mapping, compression, mesh completion and morphing. The surface that the mesh is parameterized on is typically referred to as the parameter domain. The purpose of mesh parameterization is to obtain a piecewise linear map, associating each triangle of the original mesh with a surface patch of the domain. An important goal of parameterization is to obtain *bijective (invertible)* maps, where each point on the domain corresponds to exactly one point of the mesh. The bijectivity of the map guarantees that there is no triangle flipping or overlapping.

Since the geometric shape of the domain surface patches will typically be different than the shape of the original triangles, angle and area distortion is introduced. The distortion of the parameterization is an important factor, therefore applications typically try to minimize the distortion for the whole mesh. aps that minimize the angular distortion are called *conformal*, maps that minimize area distortion are called *authalic*, and maps that minimize distance distortion are called *isometric*.

In this section, we deal with the problem of computing a bijective mapping between a closed genus-0 mesh and a spherical domain, such that distortion is globally minimized. It is important to note that on such a domain, an arbitrary mesh mapping can be *authalic*, or *conformal* but not *isometric*, as it would have to be both authalic and conformal and in general this is not feasible.

Although mainly due to recent fundamental theoretical work [20, 4, 5, 91, 105, 44, 93, 12], there is a good understanding of the mathematical aspects underlying spherical mesh parameterizations, the problem of computing such parameterizations efficiently remains open.

The existing spherical mesh parameterization methods can roughly be classified into two categories : (i) methods that attempt to extend planar methods and (ii) methods that use some kind of non-linear optimization. Typical methods of the former category generalize planar parameterization methods of barycentric coordinates [112] to the spherical domain [4, 54, 46]. For example in the work of [46] the non-linear spherical problem is transfered to the disk and then the stereographic projection is used to obtain the spherical mapping. Other methods in this category proceed by splitting the mesh in two half-meshes and mapping each half individually onto a hemisphere [54].

Amongst the latter category, several methods use non-linear optimization [20, 91, 44] and usually have a higher computational cost. An indicative example of this category is the work by [91] that combines a hierarchical method with the optimization of a stretch metric to obtain geometric images of closed meshes. Another method following a hierarchical approach to obtain an approximate solution was introduced by [17]. Firstly, the original mesh is simplified by using an edge-collapse technique until a tetrahedron is obtained. Afterwards, the simplification process is reversed by reinserting the vertices.

Another method which does not fall directly in either category, is presented by [45] where the parameterization is based on the properties of the complex conformal gradient field. This method has also the advantage of being able to handle higher genus meshes.

The extension of planar methods to the spherical domain is attractive, since planar parameterizations require the solution of a simple linear system. However, they are usually required to introduce some cuts in the mesh. Such cuts induce distortion that may be undesirable for most applications.

Therefore, the generalization of planar barycentric mapping to the spherical domain is important. The weights assigned to the vertices offer some degree of control over the final parameterization and it is guaranteed that the final parameterization will be fold free provided that the weights are positive. Nevertheless, earlier fast methods by [4] that attempt to converge to valid barycentric parameterizations by employing simple projected *Gauss-Seidel* techniques are bound to fail in certain cases [93]. ore recently, approaches that combine various techniques from the above-mentioned parameterization methods have appeared (e.g. the work by [93]). Still, they need several minutes to calculate parameterizations for a typical, by today's standards, model. In addition, prior methods do not consider the issues arising from a parallel implementation.

The rest of the chapter is organized as follows. Section 3.2 offers some background material on planar parameterizations and spherical parameterizations that use reduction to the planar case. Section 3.3 presents our iterative quadratic solver for spherical mesh parameterization. Section 3.4 describes our parallel implementation along with an experimental study and several heuristics that speedup the parallel realization on modern architectures. Section 3.5 presents applications of our technique on automated feature selection, mesh decomposition and similarity-based object retrieval. Finally, Section 3.6 offers conclusions.

3.2 Preliminaries

3.2.1 Planar parameterizations

A *planar triangulation* is a simple triangulated plane graph the edges of which are represented by straight lines. The triangulation is called *valid* when the only intersections between its edges are at the common endpoints. It is known by [32] that every planar graph G has a valid straight line representation. Therefore, for any planar graph there exist a set of points p such that the induced triangulated graph $T(G,p)$ is valid. A way to construct such a graph is described by [112]. The boundary vertices of G are mapped to a convex polygon with the same number of vertices and in the same order. Then, the interior vertices are placed such that each vertex is the centroid of its neighboring vertices. This was extended by [34] who has proven that the vertices can be any convex combination of its neighboring vertices.

$$\begin{aligned} v_i &= \sum_{j \in N_i} w_{ij} v_j \\ \sum_{j \in N_i} w_{ij} &= 1 \\ w_{ij} &> 0 \end{aligned} \tag{3.1}$$

Consequently, for finding a one-to-one bijective mapping for a mesh with an open boundary to a convex parametric domain (unit disk, unit square), a sufficient condition is to find a set of positive weights that satisfy (3.1) and solve the corresponding linear system for those weights. The resulting system has always a unique solution provided that the boundary vertices are fixed. A straightforward choice is to choose equal weights so that each vertex represents the centroid of its neighbors. This is also referred to as barycentric mapping.

Though through this process a valid triangulation can always be created with no folded triangles, it is generally desirable that the resulting mapping minimizes a distortion metric that is related with some distinct shape characteristics of the original mesh.

Two possible sets of weights are described by [27] that produce *authalic* and *conformal* parameterizations. Nevertheless, these weights might be negative when the polygon is not convex [33]. Thus, although the final parameterization is *conformal* or *authalic* it may contain folded triangles. In addition, since the weights are not positive, the corresponding linear system may be singular. To overcome this limitation for conformal weights we can clamp the angles between 0^o and 90^o degrees and therefore have strictly positive weights. Another possible set of weights are the *mean value* coordinates introduced by [33]. These weights are not symmetric ($w_{ij} \neq w_{ji}$).

3.2.2 Spherical parameterization by reduction to the planar case

The methods for planar parameterizations [119] can be directly extended to a spherical domain by reducing the spherical parameterization problem to the planar case. A first approach to reducing the problem is to select two vertices as the poles (north and south) of the parameterization. Subsequently, a geodesic path must be established between the poles over the mesh surface. The path connecting the two poles defines the boundaries of the parameterization and thus the spherical surface can be converted to a unit disk. If equal weights are chosen and the poles are selected based on the largest distance along the z direction in object space, the resulting system

is the linear system proposed by [20]. This approach yields a valid spherical parameterization for every mesh. Nevertheless, the choices for the poles and the path directly affect the quality of the parameterization.

Moreover, it turns out that selecting a good path is a difficult problem on its own and usually there is severe distortion in the final parameterization. The underlying issue is that the mapping of a set of boundary vertices to a fixed convex polygon is far from trivial. This is due to the fact that in most cases the boundary vertices do not form a convex polygon. Therefore, the obtained parameterizations exhibit high deformation. To tackle these difficulties [75] construct parameterizations with free boundaries. Nevertheless, the seams introduced by the cuts in the mesh may be undesirable for certain applications.

A second approach to reduce the problem is to cut out a triangle from the mesh, leaving an open boundary, and to make the mesh homeomorphic to the unit disk. This approach, also referred to in the literature as stereo mapping, usually results in heavily distorted parameterizations since using the corresponding unit triangle as a boundary tends to cluster the remaining vertices in the center of the triangle.

3.3 Spherical parameterizations

The main drawback of extending the planar methodologies to the spherical domain is the unnecessary distortion introduced in the parameterization. Therefore, it is advantageous to directly parameterize the meshes on the spherical domain to allow seamless continuous parameterizations of genus-0 meshes. Unfortunately, generalizing the barycentric coordinates and the planar parameterization theory to a spherical domain is not straightforward. Since the domain is non-planar, expressing a vertex on the sphere as a convex combination of its neighbors is in general not feasible. This would imply for example that if the neighbors of a vertex are co-planar, then the vertex should also lie on the same plane. Nevertheless, the mathematical aspects of the parameterization on the spherical domain have received increasing attention in the last few years. One important observation according to [44] is the following:

Theorem 3.1. *If each vertex position is expressed as some convex combination of the positions of its neighbors projected on the sphere (3.2), then the formed spherical triangulation is valid.*

$$\begin{aligned} v_i &= \frac{\sum_{j \in N_i} \lambda_{ij} v_j}{|| \sum_{j \in N_i} \lambda_{ij} v_j ||} \\ \sum_{j \in N_i} \lambda_{ij} &= 1 \\ \lambda_{ij} &= \lambda_{ji} \\ \lambda_{ij} &> 0 \end{aligned} \tag{3.2}$$

The spherical triangulation may be controlled by choosing a proper set of symmetric weights, similarly to the planar case. The system of equations (3.2) can also be expressed as a set of non-linear equations for the nodes $i = 1, ..., n$ of a mesh, seeking solution for the positions of the vertices (x_i, y_i, z_i) and the n auxiliary variables a_i,

$$
\begin{aligned}
a_i x_i - \sum_{j \in N_i} \lambda_{ij} x_j &= 0 \\
a_i y_i - \sum_{j \in N_i} \lambda_{ij} y_j &= 0 \\
a_i z_i - \sum_{j \in N_i} \lambda_{ij} z_j &= 0 \\
x_i^2 + y_i^2 + z_i^2 &= 1
\end{aligned}
\tag{3.3}
$$

The physical interpretation of the equations (3.3), assuming that the weights λ_{ij} correspond to spring constants, is the minimization of the sum of the squared weighted lengths (spring energy) subject to the vertices being on the sphere. Therefore, the energy that is minimized is:

$$
E(v_1, v_2, ..., v_n) = \frac{1}{2} \sum_{(i,j) \in E} \lambda_{ij} ||v_i - v_j||^2 \tag{3.4}
$$

Generally, a solution of this system is not unique. Without restricting some degrees of freedom, there may be infinite solutions due to the possible rotations over the sphere. ore importantly, there are degenerate solutions that satisfy (3.3). The most obvious one is observed when $a_i = 0$ where all the vertices of the parameterization collapse to one point on the sphere. Another possible degenerate solution can occur when the mesh contains a Hamiltonian cycle and the vertices are mapped to the equator of the sphere. Other degenerate solutions have been presented (for example by [44]).

Moreover, even a robust and stable non-linear solver may calculate degenerate solutions for the system of equations (3.3). A key observation that sheds light on this situation, is that as the solver iterations proceed, some triangles start growing and eventually pass through the equator of the sphere. The fundamental problem is that the spherical energy minimum occurs at a collapsed configuration, since the area of a planar triangle is always smaller than the area of the corresponding spherical triangle. Such cases are problematic, because there is an estimation error introduced in the calculation of the distortion metric over the surface. This error increases disproportionately with the size of the triangles. Therefore, the non-linear optimizer may minimize the corresponding distortion metric (energy function) over the sphere surface by increasing the size of the triangles with the largest error.

One way to avoid these degenerate solutions is to constrain three or more vertices, thus constraining the solver. However, in practice there are two problems: (i) the extra constraints introduce additional distortion in the parameterization, making the determination of a proper set of constrained vertices difficult, (ii) in addition, without paying special attention to the set of the constrained vertices, the non-linear problem may become infeasible.

3.3.1 An energy decreasing algorithm

Summarizing the above observations, if we try directly to solve (3.3), the following problems occur,

- *Non convexity.* The constraints $x_i^2 + y_i^2 + z_i^2 = 1$ are not convex. Therefore classical convex minimization cannot be used directly.

- *Non uniqueness.* The energy does not have a unique minimum and degenerate solutions always exist.

- *High computation cost.* Due to the above reasons, the usual approach of non-linear optimization has a high computation cost.

An approach to tackle these difficulties was proposed by [37]. Here a penalty term d_{min}^{-2}, where d_{min} is the minimum distance of each triangle from the sphere center, was added in the corresponding planar quadratic energy and therefore there is no need to constrain any vertices or reproject the solution to the sphere. The motivation of this approach is to provide an upper bound of the spherical energy by scaling the corresponding planar energies of the triangles. Therefore, the corresponding problem (3.3) is transformed to an unconstrained one, that can be solved with standard methods.

Another possible approach to overcome the high computation cost is to use iterative procedures that attempt to converge to a valid parameterization by applying local improvement (relaxation) [4]. The idea is to reduce the spring energy of the points with Laplacian smoothing ignoring the sphere constraint and renormalise the solution to obtain valid spherical points. However, these algorithms are heuristic based and there is no guarantee that they will terminate. In practice, the iterative process can collapse and may require a restart. Furthermore, the termination criteria for such an algorithm are difficult to define. For example, a similar method is used by [93] to calculate an initial guess for the solution. However, the residual reduction criterion proposed to terminate the method is usually too conservative and the initial guess is far away from the solution. Thus, there is the need to complement it with a non-linear optimization step.

Moreover, techniques that rely upon non-linear software need to devise new parallel solutions and strategies to conform to new parallel architectures. For this reason, the effectiveness of all the techniques relying upon non-linear optimization is limited on inherently parallel architectures like modern GPUs.

To efficiently employ iterative procedures, a central issue is the renormalization step. The iterative procedure to solve the problem with an energy decreasing step and the renormalization of the solution can be described through the following steps:

1. Let vertices $v_1^0, ..., v_n^0$ be an initial guess for the solution

2. For $j = 0 \ldots$ until convergence

   (a) Find $v_1^{j+1}, ..., v_n^{j+1}$ such that $E(v_1^{j+1}, ..., v_n^{j+1}) \leq E(v_1^j, ..., v_n^j)$
   where v_i^{j+1} may not belong to the sphere

   (b) Set $v_i^{j+1} = \frac{v_i^{j+1}}{||v_i^{j+1}||}$ for $i = 1, ..., n$

The question that arises is whether the energy is still decreasing after the renormalization step and whether the algorithm converges to a solution of (3.3). The problem is the unknown behavior of the energy after the normalization. In other words, the gain obtained by the energy decreasing step can be lost during renormalization.

It is therefore evident that the extension of iterative schemes in the spherical domain is not straightforward. A useful observation for the energy is the following,

Proposition 3.1. *If $v_i \in \mathbb{R}^3$ and $||v_i|| \geq 1$ for the nodes $i = 1, ..., n$ of the mesh, then $\frac{v_i}{||v_i||}$ is on the sphere surface and moreover for the energy (3.4) with barycentric or conformal weights :*

$$E(\frac{v_1}{||v_1||}, ..., \frac{v_n}{||v_n||}) \leq E(v_1, .., v_n) \tag{3.5}$$

Proof. First we observe that $\forall (i, j)$:

$$\begin{aligned} ||\frac{v_i}{||v_i||} - \frac{v_j}{||v_j||}||^2 \leq ||v_i - v_j||^2 \\ ||v_i|| \geq 1 \\ ||v_j|| \geq 1 \end{aligned} \tag{3.6}$$

∎

Therefore, by the definition of the energy (3.4), and because the barycentric and the (clamped) conformal weights [27] are positive, moving each vertex to the sphere cannot increase any term of the summation.
The above observation motivates the following iterative procedure,

1. Let vertices $v_1^0, ..., v_n^0$ be an initial guess for the solution
2. For $j = 0 \ldots$ until convergence
   (a) Find $v_1^{j+1}, ..., v_n^{j+1}$ such that $E(v_1^{j+1}, ..., v_n^{j+1}) \leq E(v_1^j, ..., v_n^j)$
   subject to $v_i^{j+1} \cdot v_i^j = 1$
   (b) Set $v_i^{j+1} = \frac{v_i^{j+1}}{||v_i^{j+1}||}$ for $i = 1, ..., n$

At each iteration, we seek a solution that minimizes the energy function subject to the constraint that the new vertices should be coplanar with the vertices in the previous iteration. Thus, the nonlinear constraints are converted to linear ones. Furthermore, the energy is decreasing after the renormalization step since $||v_i^{j+1}|| \geq 1$. For the cases of barycentric and conformal quadratic energy functions, the energy minimizing problem in step 2(a) is a saddle point problem.

Saddle point problem solution

We first present an iterative process for solving the generic saddle point problem and then we reduce our problem to this process. Consider the block 2x2 linear system of the form,

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} r \\ q \end{pmatrix} \tag{3.7}$$
$$A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, n \geq m$$

It is known that the solution of the linear system is equivalent to minimizing a function f subject to a set of m linear constraints [14],

$$\begin{aligned} \min_u f(u) = \frac{1}{2} u^T A u - u^T r \\ \text{s.t. } B^T u = q \end{aligned} \tag{3.8}$$

When A is a symmetric positive semidefinite matrix, this equality-constrained quadratic problem describes a (generalized) saddle point problem. In this case the variable v represents the vector of Lagrange multipliers. Any solution (u_*, v_*) of (3.7) is a saddle point for the Lagrangian

$$\mathcal{L}(u, v) = \frac{1}{2}u^T Au - r^T u + (Bx - q)^T v \tag{3.9}$$

where a saddle point $(u_*, v_*) \in \mathbb{R}^{n+m}$ satisfies

$$\mathcal{L}(x_*, y) \leq \mathcal{L}(u_*, v_*) \leq \mathcal{L}(u, v_*), u \in \mathbb{R}^n \text{ and } v \in \mathbb{R}^m \tag{3.10}$$

Under the following conditions there is a solution to the system (3.7) and it is unique,

Theorem 3.2. *Let,*

1. *A be a real symmetric positive semi-definite $n \times n$ matrix*
2. *B be a real $n \times m$ matrix with full column rank*
3. *A and B^T have no nontrivial null vectors in common*

Then (3.7) has a unique solution for u,v.

Proof. See Theorem 2 by [14]. ∎

Given a non zero vector u^0 and assuming a splitting of the matrix $A = M - N$, an iterative scheme to solve (3.7) is the following,

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} u^{k+1} \\ v^{k+1} \end{pmatrix} = \begin{pmatrix} N & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u^k \\ v^k \end{pmatrix} + \begin{pmatrix} r \\ q \end{pmatrix} \tag{3.11}$$

Therefore to solve (3.7) a procedure is,

1. Solve $M\tilde{u}^{k+1} = Nu^k + r$
2. Solve $(B^T M^{-1} B)v^{k+1} = B^T \tilde{u}^{k+1} - q$
3. Solve $M(u^{k+1} - \tilde{u}^{k+1}) = -Bv_{k+1}$

If $A = D - L - L^T$, where D is a nonsingular diagonal matrix and L is a strictly lower triangular matrix, then the iterative scheme is convergent for the following choices of M and N [28],

$$M = \frac{1}{\omega}D, N = \frac{1-\omega}{\omega}D + L + L^T \tag{3.12}$$

with $\omega > 0$ so small that $\frac{2}{\omega}D - A$ is a positive definite matrix.

$$M = \frac{1}{\omega}D - L, N = \frac{1-\omega}{\omega}D + L^T \tag{3.13}$$

with $0 < \omega < 2$.

(3.12) and (3.13) are the usual *Jacobi* and *Gauss-Seidel* iterations with over relaxation parameter ω.

To map our problem to (3.11), we first establish some necessary notation. Without loss of generality, we assume that some vertices are fixed on the sphere and those are numbered last. Therefore, let n and b represent the number of free and fixed vertices respectively. Next, we define the weighted $m \times m$ block matrix Λ for the vertices of the mesh,

$$\begin{pmatrix} \sum_j^m w_{1j} I_3 & -w_{12} I_3 & & \cdots & -w_{1m} I_3 \\ -w_{21} I_3 & \sum_{j=1}^m w_{2j} I_3 & & & -w_{2m} I_3 \\ \vdots & \vdots & \ddots & & \vdots \\ & & & \sum_{j=1}^m w_{m-1j} I_3 & -w_{m-1m} I_3 \\ -w_{m1} I_3 & -w_{m2} I_3 & \cdots & -w_{mm-1} I_3 & \sum_{j=1}^m w_{mj} I_3 \end{pmatrix}$$

where $m = n + b$, w_{ij} is the weight (conformal or barycentric) of the edge connecting i and j vertices, and I_3 is the 3×3 identity matrix. Moreover, $w_{ij} = 0$ iff $(i, j) \notin E$. In addition, E,V denote the edges and the vertices of the mesh, and (x_i, y_i, z_i) are the coordinates of the ith vertex of the mesh. We further denote by $\tilde{\Lambda} = [A_U, A_C]$ the matrix that is derived from Λ by deleting its last $3b$ rows. A_U contains all of the weights corresponding to the free vertices and is an $3n \times 3n$ matrix, A_C contains all of the weights corresponding to the fixed vertices and is an $3n \times 3b$ matrix.

We use the following matrices to minimize the quadratic energies for barycentric, and conformal mesh mappings,

$$A := A_U, A \in \mathbb{R}^{3n \times 3n} \tag{3.14}$$

$$B := \begin{pmatrix} x_1 & & \\ y_1 & & 0 \\ z_1 & & \\ & \ddots & \\ & \ddots & \\ & \ddots & \\ & & x_n \\ 0 & & y_n \\ & & z_n \end{pmatrix}, B \in \mathbb{R}^{3n \times n} \tag{3.15}$$

Furthermore, we have the vectors,

$$u^k = [x_1^k, y_1^k, z_1^k, ..., x_n^k, y_n^k, z_n^k]^T, u^k \in \mathbb{R}^{3n} \tag{3.16}$$

$$r = -A_C [x_{n+1}, y_{n+1}, z_{n+1}, ..., x_{n+b}, y_{n+b}, z_{n+b}]^T, r \in \mathbb{R}^{3n} \tag{3.17}$$

$$q = [1, ..., 1]^T, q \in \mathbb{R}^n \tag{3.18}$$

using the splitting of A, $M = \frac{1}{\omega} D$ and $N = \frac{1-\omega}{\omega} D + L + L^T$ (Jacobi iteration) we observe that $B^T M^{-1} B = \omega D$ (since the vertices are on the sphere). Furthermore, assuming that we use the normalized weights at each vertex, the iterative procedure becomes,

1: **for** k=0 until convergence **do**
2: **for** i=1 until n **do**
3: $[\tilde{x}_i, \tilde{y}_i, \tilde{z}_i] = (1-\omega)[x_i^k, y_i^k, z_i^k] + \omega(\sum_{j=1,j\neq i}^{n} w_{ij}[x_j^k, y_j^k, z_j^k] + \sum_{j=n+1}^{n+b} w_{ij}[x_j, y_j, z_j])$
4: $\lambda_i = [x_i, y_i, z_i]^T[\tilde{x}_i, \tilde{y}_i, \tilde{z}_i] - 1$
5: $[x_i^{k+1}, y_i^{k+1}, z_i^{k+1}] = [\tilde{x}_i, \tilde{y}_i, \tilde{z}_i] - \lambda_i[x_i, y_i, z_i]$
6: **end for**
7: **end for**

Algorithm 3: Iterative Saddle point solution

Proposition 3.2. *The iterative algorithm 1 converges to the unique solution of the saddle point problem (3.7) for symmetric weights.*

Proof. We first prove the conditions for Theorem 2. Because the mesh is connected and a positive weight is associated with each edge, A is irreducible. In addition, $|a_{ii}| \geq \sum_{j=1,j\neq i}^{3n} |a_{ij}|$ for each row. Equality is true if the corresponding vertex is connected only with free nodes. Consequently, if the weights are symmetric and at least one node is fixed, the matrix A is a hermitian irreducibly diagonally dominant matrix, and therefore positive-semidefinite with $ker(A) = 0$. In addition, it is obvious that the matrix B has full column rank since at least one of the components of each vertex will be non-zero in the spherical domain. Therefore, the conditions of Theorem 2 are satisfied. Furthermore, $\frac{2}{\omega}D - A$ is positive definite for $0 < \omega < 1$ since it is a strictly diagonally dominant and irreducible matrix with positive diagonal elements. Thus, the iterative procedure converges to the unique solution. ∎

3.4 Parallel parameterization

We have developed a parallel implementation of the algorithm presented in Section 3.3. The software is available at `http://www.cs.uoi.gr/~fudos/smi2011.html`. The inherent parallelism of the specific method enables us to map the problem to the hardware as efficiently as possible. As an API for our implementation, we have used OpenCL 1.1 to achieve almost direct portability of our core source to both GPU and CPU based architectures.

To maximize the performance of our parallel implementation we have considered a number of key factors. A characteristic that affects parallel algorithm effectiveness on all architectures is the number of sequential steps of the algorithm. To maximize the parallel execution we have employed the Jacobi iteration.

We present heuristics that optimize the parallel performance of the proposed algorithm. We have investigated the employment of three optimization principles and we have evaluated their effect on the performance for both GPU and multicore architectures:

- Optimize memory usage to maximize instruction throughput.
- Test for convergence only every n iterations have been carried out, to reduce the data synchronization overhead.
- Increase the processing unit cache hit ratio.

One important consideration in modern GPUs are the effective memory usage to achieve the maximum memory bandwidth and the optimization of the instruction usage to achieve the maximum instruction throughput [88]. To optimize the memory usage we have further minimized the data transfer between the host and the GPU device by reducing the number of residual tests needed for the convergence test for the solution of the saddle point problem. Since the convergence of the Jacobi is guaranteed, it is not necessary to test the convergence of the linear system at every iteration. ore specifically, we have opted to perform convergence tests only every a certain number of iterations. This is beneficial to modern GPUs since it is better to increase the OpenCL kernel invocations on the GPU and reduce the synchronization between the host and the device.

We have conducted experiments with equal and conformal weights. Since conformal weights can be negative in certain cases, all input angles can be clamped between 5^o and 85^o degrees as suggested by [37]. For all the examples, the algorithm termination δ for the residual reduction was set to 10^{-7}.

To reduce the data synchronization overhead, the residual of the saddle point problem was tested for convergence every 1000 iterations. The sparse residual check policy has a large impact on GPUs and especially on GPUs with slower buses (PCIe 1.0).This has a small positive effect on multicores as well. Table 3.1 summarizes the results of the parameterization on different commonly used models [1, 18] using an NVIDIA GTX 480. We have also obtained performance results by carrying out our algorithm on two multi core processors, an Intel Core Duo E6600 and an Intel Core i7-870. Figure 3.1 compares the performance of these architectures. Table 3.3 illustrates the difference in running times on the GPU and on the CPU, while Table 3.4 presents a direct comparison with the running times of the publicly available parameterization software by [93]. For the results of Table 3.4 we have applied our approach to the models accompanying the software, using though a much smaller residual target than the one used by [93].

Table 3.1: Numerical results for finding a spherical parameterization on the GPU with different models. In this context, the number of iterations is the number of saddle point problems solved.

model	map	# vertices	# faces	# iterations	L_2 res ($\times 10^{-8}$)	time (secs)
Suzanne	Barycentric	7573	15142	4	5	0.575
Suzanne	Conformal	7573	15142	3	5	0.589
Gargoyle	Barycentric	24990	49976	4	2	1.706
Gargoyle	Conformal	24990	49976	3	6	2.326
Igea	Barycentric	25586	51168	3	4	0.908
Igea	Conformal	25586	51168	2	3	0.936
Lion Vase	Barycentric	38952	77900	3	3	1.567
Lion Vase	Conformal	38952	77900	3	3	2.053
Homer	Barycentric	78850	157696	3	1	4.923
Homer	Conformal	78850	157696	3	4	10.920
Buste	Barycentric	183580	367156	3	1	13.759
Buste	Conformal	183580	367156	2	1	22.667

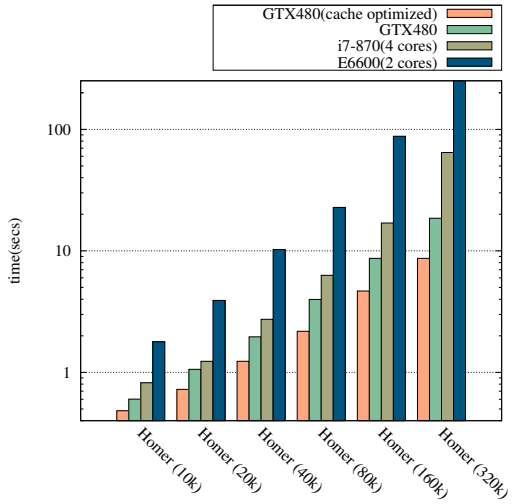


Figure 3.1: Performance difference between GPU and CPU OpenCL implementation in logarithmic scale. Results with vertex cache optimization are also included.

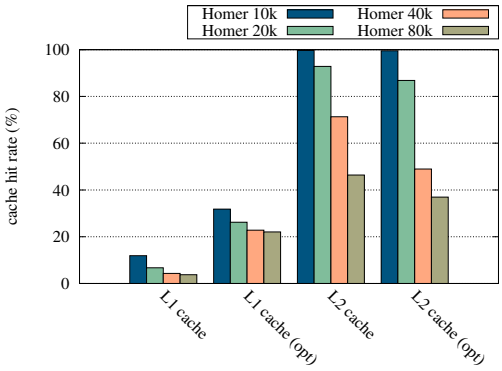


Figure 3.2: Cache hit rate statistics on the GPU. Results with a vertex locality optimization step from [95] are also shown for comparison.

Finally, many cache misses can be avoided by combining the method with preprocessing techniques that further improve the cache locality of the mesh indices. Figure 3.2 presents experimental results regarding the cache hit ratio for various models. The cache efficiency is decreased as the size of the model increases. As a fast preprocessing step, we have used the vertex locality optimization proposed by [95]. The optimized results are also included in Table 3.2 for comparison. We observe that with the optimized meshes the computation time is reduced up to 50% on the GPU. On multicore architectures this has no effect since the CPU cache is usually large enough to fit the entire mesh index.

Table 3.2: Numerical results for barycentric mapping on the GPU with different levels of detail

model	# vertices	# faces	iters	L_2 res ($\times 10^{-8}$)	time (secs)	opt time (secs)
Homer (Lod1)	5002	10000	4	10	0.577	0.483
Homer (Lod2)	10002	20000	4	2	1.059	0.725
Homer (Lod3)	20002	40000	4	2	1.961	1.224
Homer (Lod4)	40002	80000	3	2	3.986	2.178
Homer (Original)	78850	157696	3	1	4.923	3.636

Table 3.3: Comparison of running times (in secs) between GPU and CPU with different core configurations.

model	map	# vertices	# faces	iters	GTX 480	i7-870(4)	i7-870(2)	i7-870(1)
Gargoyle	Barycentric	10002	20000	4	0.946	1.186	1.950	3.135
Gargoyle	Conformal	10002	20000	4	0.949	1.045	1.685	2.714
Torso	Barycentric	11362	22720	4	0.718	1.107	1.731	2.808
Torso	Conformal	11362	22720	3	0.870	1.123	1.747	2.745
Skull	Barycentric	20002	40000	3	0.649	1.076	1.719	2.904
Skull	Conformal	20002	40000	2	0.643	0.920	1.373	2.230
Bunny	Barycentric	67038	134074	3	1.217	3.616	6.635	12.038
Bunny	Conformal	67038	134074	2	2.158	3.778	7.737	14.118

3.5 Applications

3.5.1 Mesh segmentation

Different measures have been used to detect structural features on meshes such as curvature based computations [121], average error from fitting with surface patches, planes, cylinders or spheres [73, 9], dihedral angles of adjacent triangles [123], electrical charge density distribution [118], region flatness or smoothness [53], geodesic distances [60] and convexity [90]. Such measures have been used in conjunction with region growing [97], watershed functions [67], reeb graphs [6], skeletons [76], clustering [60] and hierarchical clustering [42] and segmentation [9], boundary extraction [72], Morse theory [113] or probabilistic fields [90]. For an extended survey of mesh decomposition techniques the reader is referred to [2] and [8].

[77] and [78] present a shape decomposition and skeletonization method for polyhedrons that is based on approximate convex decomposition. This principle was used by [110] to decompose point clouds into components that represent features by using the concavity intensity to detect saddle points and the discrete curvature to detect edges. A more general principle that can handle even complex (non star-shaped) objects is to derive a minimal length 3D path (curve segment) to connect the point to the convex hull without crossing the mesh. Since the convex hull has a straightforward spherical parameterization, finding a path that connects v to the convex hull corresponds to blowing the interior of the object until it expands to the convex hull. This is a

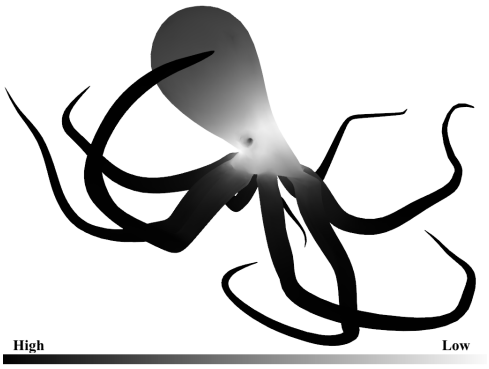


Figure 3.3: Visualization of the area stretch factor.

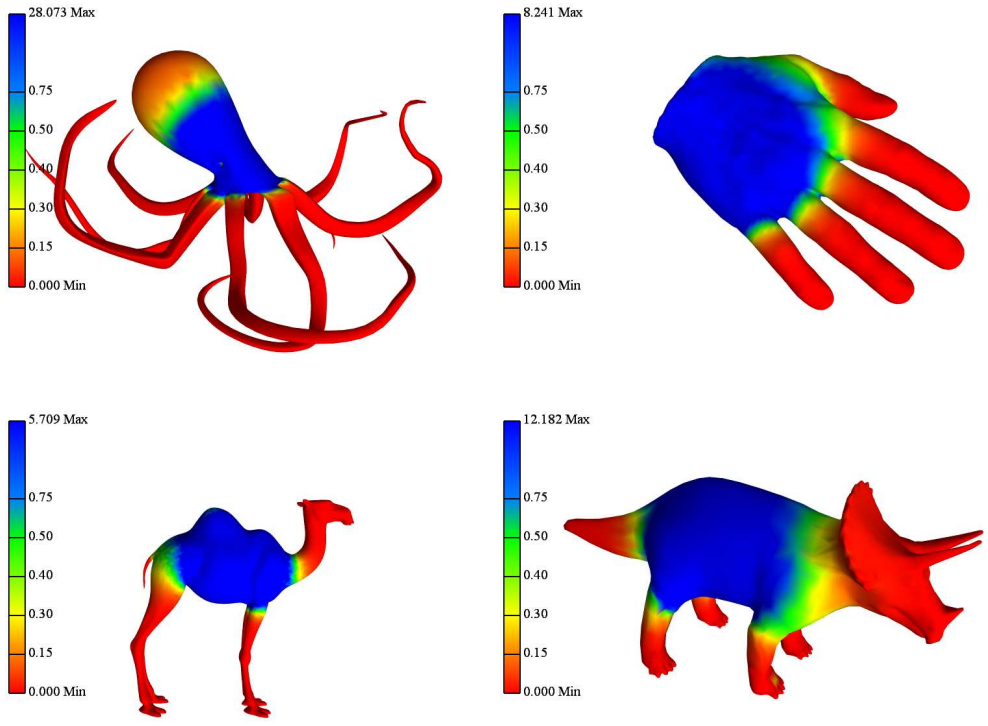


Figure 3.4: Visualization of the ratio between the mapped area and the original surface. Blue and red colors correspond to high and low distorted areas respectively.

Table 3.4: Comparison of running times of our method vs the one by [93] on the same CPU (E6600).

model	map	# vertices	# faces	[93] method (secs)	our method (secs)
Gargoyle	Barycentric	10002	20000	23.58	3.422
Gargoyle	Conformal	10002	20000	62.86	2.734
Torso	Barycentric	11362	22720	26.50	2.704
Torso	Conformal	11362	22720	84.49	2.719
Skull	Barycentric	20002	40000	67.73	2.828
Skull	Conformal	20002	40000	87.26	2.469

CPU-intensive process that can be simulated with a spring system [78, 85]. Our efficient parallel spherical mesh parameterization can derive a measure that is somehow related to the minimum path by computing the deformation that has been applied to the adjacent triangles of a vertex. This yields a more robust measure that can be computed exactly very efficiently. As compared to very sophisticated techniques such as the one presented by [61] that uses mesh coarsening, DS transforms and refinement, our work yields results of comparable quality much faster. Post processing can always be used for application specific mesh-segment refinement.

Our approach is based on the key idea that the spherical embedding represents a pose invariant representation of the mesh for quasi articulated objects. Any spherical embedding is expected to create some dense concentrations of faces on the sphere due to the prominent extremities of the mesh. The extruding parts of the meshes, for example the limbs, are expected to be mapped to relatively small regions on the sphere. Therefore, the *area stretching factor* (ratio of area in the surface and in the mapping) in those parts is expected to be much higher than in the rest of the mesh. Moreover, in the case of the conformal map the angles are generally preserved in the mapping. Therefore, the area distortion is affected more by the geometry of the model and less by the distortion introduced in the mapped angles. Consequently, our thesis is that the spherical embedding of a mesh contains a substantial amount of information about its geometric shape. To illustrate, consider Figure 3.3 where the area stretch factor of the conformal parameterization for the octopus model is depicted. Figure 3.4 visualizes the distortion of the parameterization in four typical models with limbs.

Definition 3.1. The *Area stretch factor* of a vertex v_0 of a mesh denoted by $A(v_0)$ is the average of the area stretch deformation of its adjacent faces.

Furthermore, we have carried out a number of experiments with a region growing approach that takes advantage of the above observation. The method starts from an initial vertex (the seed) and expands while a threshold in the variation of the area stretch factor is satisfied. Figures 3.5 and 3.6 show the segmentation of four typical models and a comparison with the mesh segmentation method presented by [60]. [60] quoted running times of a few minutes for segmenting moderately sized meshes by relying on mesh simplification to reduce the computation cost. To compare the efficiency of our method to theirs, it is important to note that our approach does not require any pre-process or simplification of the meshes. In all the experiments performed, the running time for our segmentation method is dominated by the parameterization step. For

meshes up to 100K triangles, the overall time was less than 5 seconds.

Definition 3.2. A vertex v_0 is called a *seed candidate* if and only if $A(v_0)$ exhibits a local minimum or maximum at v_0.

We first derive the candidate seeds for our model. Subsequently, the seeds are sorted in descending order according to the area stretch factor and our region growing approach is instantiated from these seeds. When a candidate seed is included in a new region it is removed from the set. This results in a number of regions that represent object extremities.

3.5.2 Texture mapping

With our method angle-preserving parameterizations can be efficiently obtained and are often suitable for texture mapping. Figure 3.10 shows the differences between the parameterizations using equal and conformal weights. Furthermore, in Figures 3.9, 3.11, 3.13 and 3.15 we have applied a checker texture to the meshes, by using the spherical coordinates of the parameterization as uv coordinates, to visualize the deformation differences between the two types of parameterization. By acquiring this correspondence one may apply deformations affecting the area of texture features of the original texture, prior to mapping it to our original mesh.

3.5.3 Shape search

One important goal of shape searching algorithms is to represent the mesh vertices with a pose invariant representation [12, 55, 56, 101]. Initial tests showed that the spherical parameterization can be used to find similar poses of meshes. The key idea is to compare the signatures derived from the conformal mappings of the meshes. To derive the signature of a mesh we use the histogram of the area stretch factor. Since a conformal mapping is independent of the resolution of the mesh and preserves the consistency of the orientation, we can further make the signature invariant to the tessellation of the mesh. This can be achieved with uniform or random sampling of the meshes. Figures 3.7, 3.8 show the histograms obtained from various poses of the same meshes. A thorough comparison of the proposed shape search measure to other approaches, in terms of hits and misses, remains as future work.

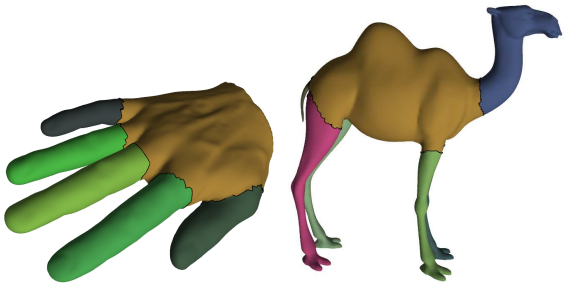
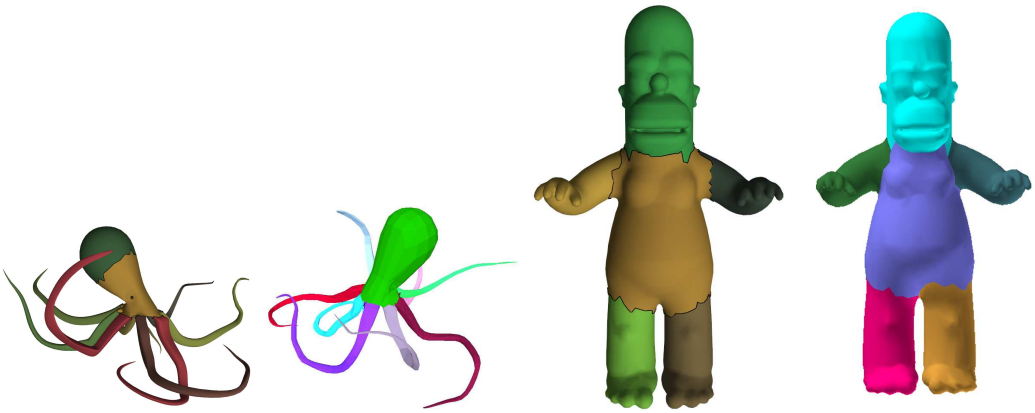
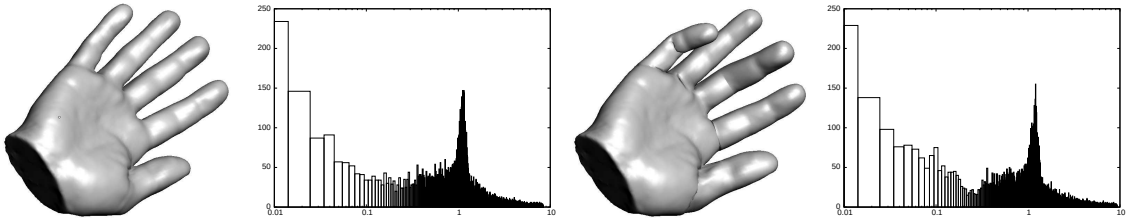


Figure 3.5: Automatic mesh segmentation.



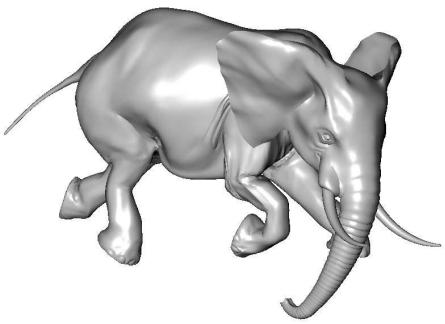
(a) Octopus with our method (b) [60] (c) Homer with our method (d) [60]

Figure 3.6: Comparison of mesh segmentation results.

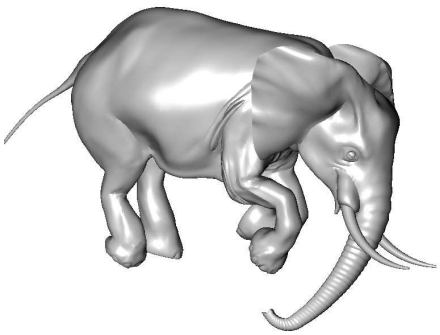


(a) Pose 1 (b) Histogram (c) Pose 2 (d) Histogram

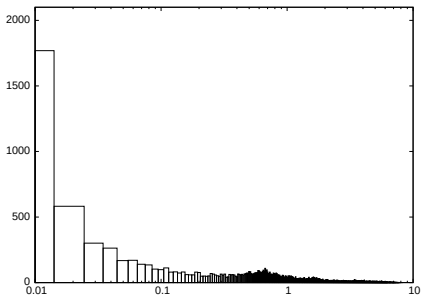
Figure 3.7: Hand mesh.



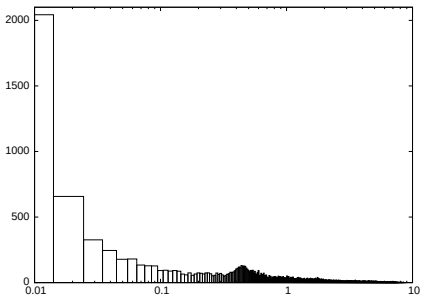
(a) Pose 1



(b) Pose 2



(c) Histogram of pose 1



(d) Histogram of pose 2

Figure 3.8: Elephant mesh.

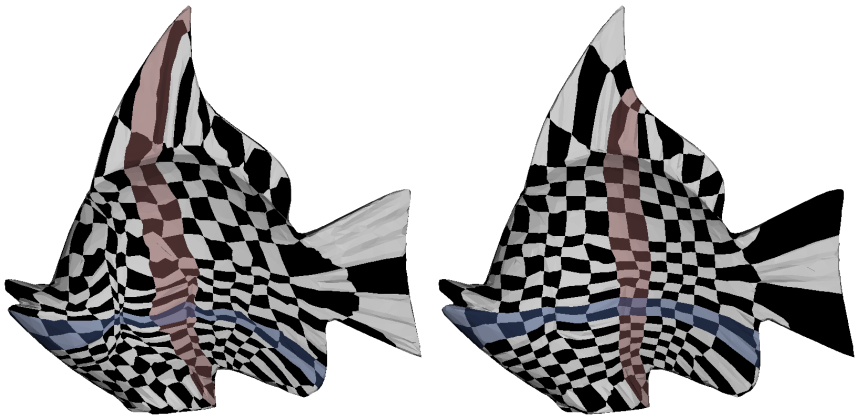
3.6 Summary

We have presented a simple and efficient parallel numerical scheme to approximate a spherical parameterization of a genus-0 mesh. We have successfully used our scheme to parameterize meshes of up to 400K triangles in less than 25 secs.

We have carried out a large number of experiments to validate that our iterative method converges to the actual bijective mapping. Using a number of standard graphical models, we have confirmed that in each case the L_2 residual is decreased below a small tolerance value (10^{-7}).

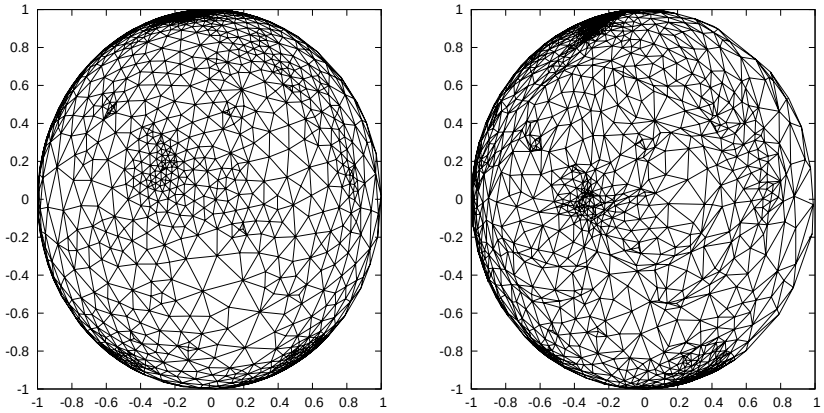
A possible extension of our work would be a theoretical result of the convergence behavior. This could be reached from the fact that the algorithm is energy-decreasing so that the iterative solution follows a path close to the solution of the non-linear equations (3.3).

Finally, exploring other implementation options might result in improved computational efficiency. In particular, a more sophisticated iterative method for solving the saddle point problem can be used to reduce the number of iterations required for convergence. Nevertheless, our profiling tests show that our implementation is dominated by the memory access latency and the data synchronization delay between the host and the GPU. Therefore, whether such an improvement would be beneficial remains to be determined.



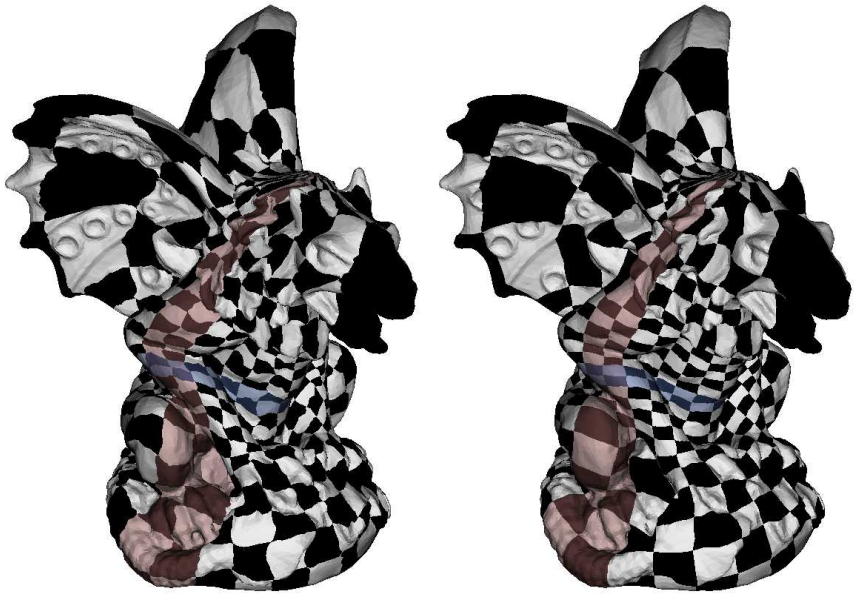
(a) Barycentric (b) Conformal

Figure 3.9: Comparison of texture mapping results for the Fish model [1].



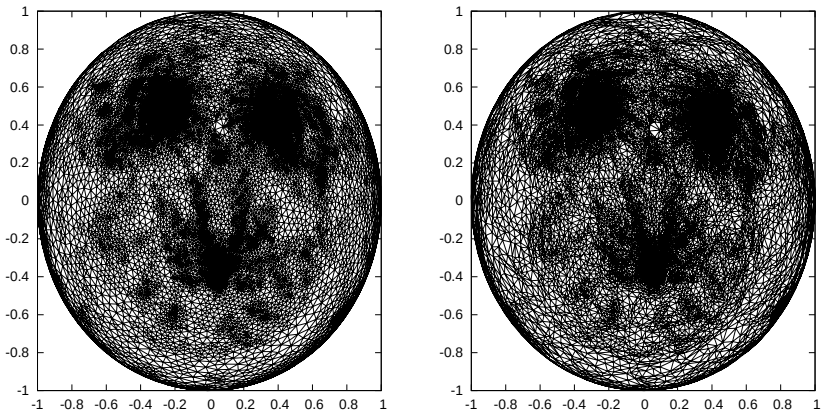
(a) Barycentric (b) Conformal

Figure 3.10: Comparison of sphere mapping results for the Fish model.



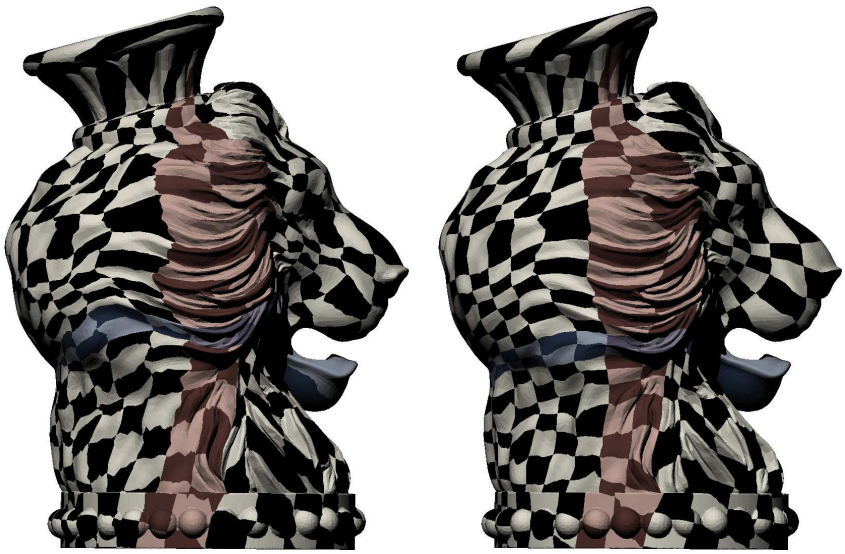
(a) Barycentric (b) Conformal

Figure 3.11: Comparison of texture mapping results for the gargoyle mesh [1].



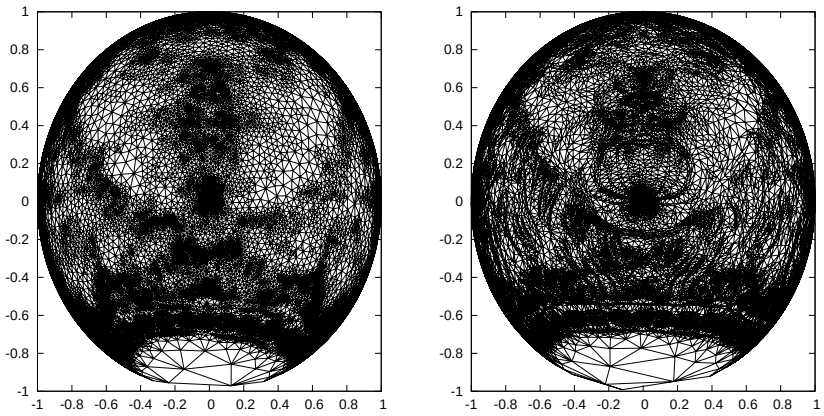
(a) Barycentric (b) Conformal

Figure 3.12: Comparison of projection results for the gargoyle mesh.



(a) Barycentric (b) Conformal

Figure 3.13: Comparison of texture mapping results for the Lion Vase model [1].



(a) Barycentric (b) Conformal

Figure 3.14: Comparison of sphere projection results for the Lion Vase model.

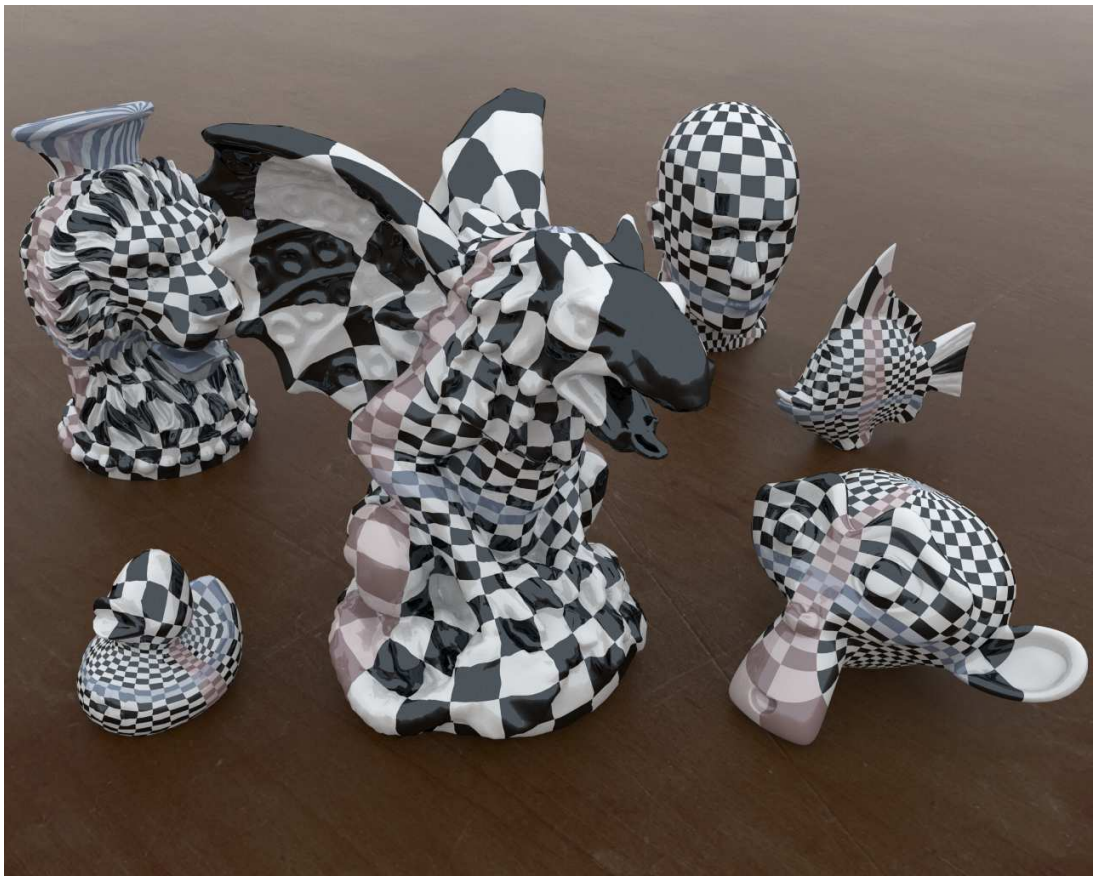


Figure 3.15: Texture mapping results for different models from [1],[18].

Chapter 4

Feature based morphing: an application of spherical parameterization

4.1 Introduction

Feature-based computer-aided design has enabled efficient and robust editing of complex CAD models through effectively capturing designer intent [50]. There is an increasing trend to make the CAD design process accessible to users with no previous CAD/CAM software experience. To this end, researchers and manufacturing companies have proposed to mimic the way an artist shapes a sculpture: start from a volume or object that is close to the intended target and iteratively shape (morph) its parts to finally render what the artist had in mind.

Our ultimate goal is to offer a novel editing paradigm for CAD models that goes beyond traditional CAD editing of mechanical parts. Towards this goal, we present an accurate and robust feature-based morphing technique that can be applied between any pair of genus-0 objects.

Although there are quite versatile and accurate methods for 2D image morphing, the 3D case remains an open problem both in terms of feasibility and accuracy.

Existing methods for 3D morphing can be categorized into two broad classes: *volume-based* or *voxel-based* [74] and *mesh-based* or *structural* [62] approaches. In this work, we follow a

mesh-based approach. The volume-based approach represents a 3D object as a set of voxels usually leading to computationally intensive computations. The volume-based approach exhibits better results in terms of boundary smoothness and rendering, since the intermediate morphs are represented as volumes. Techniques such as marching cubes [82] are employed to acquire the final polygonal representation used for rendering. Furthermore, most applications in graphics use mesh-based representations, making mesh-based modeling more broadly applicable.

Although mesh morphing is more efficient as compared to volume-based morphing, it requires a considerable preprocessing of both the source and the target object. Mesh morphing involves two steps. The first step establishes a mapping between the source and the target object (correspondence problem), which requires that both models are meshed isomorphically with a one-to-one correspondence. The second step involves finding suitable paths for each vertex connecting the initial position to the final position in the merged mesh topology (interpolation problem). due to the increased space complexity of the representation.

In this work, we introduce a sound and complete approach to morphing between any two genus-0 objects. Recall that genus-0 objects are by definition homeomorphic to the sphere. Our mapping works in two phases. In the first phase, we calculate an initial bijective mapping. In the second phase, we optimize the mapping to achieve a better placement under specific geometric criteria and topological constraints. We also present an improvement of this approach that takes into consideration 3D features and derives a feature correspondence set to improve the final visual effect. This is a very important characteristic for similar objects, as in the case of morphing between two articulated human representations. Object alignment, feature detection and feature point matching is performed automatically without user intervention.

The rest of this chapter is structured as follows. Section 4.2 presents related work on 3D morphing. Section 4.3 presents the spherical parameterization step of our approach. Section 4.4 briefly describes the efficient computation of the intersections among the polygons on the sphere and the calculation of the interpolation trajectory. Section 4.5 presents an alternative mapping method that can be applied to one of the morphed objects based on the mapping of the other object and a feature point correspondence list of the two meshes. Section 4.6 presents an experimental evaluation of our method and some visual morphing results. Finally, Section 4.7 offers conclusions.

4.2 Related Work

Most surface-based mesh morphing techniques employ a merging strategy to obtain the correspondence between the vertices of the input model. The merging strategy may be either automatic or user specified. [62] proposes an algorithm for the morphing of two objects topologically equivalent to the sphere. The mapping presented is accomplished by a mere projection to the sphere and thus is applicable solely to star shaped objects.

[57] use a spring system to model the mesh and gradually force it to expand or shrink on the unit sphere by applying a force field. Methods using springs do not always produce acceptable mappings especially when handling complex non convex objects. We overcome this problem successfully in our approach.

In [4, 124], a spring-like relaxation process is used. The relaxation solution may collapse

to a point, or experience foldovers, depending on the initial state. Several heuristics achieving convergence to a valid solution are used.

[105, 94, 38] describe methods to generate a provable bijective parameterization of a closed genus-0 mesh to the unit sphere. The projection involves the solution of a large system of non-linear equations. A set of constraints on the spherical angles is maintained to achieve a valid spherical triangulation. We have adapted some of these ideas in our work.

[100] present a method that directly creates and optimizes a continuous map between the meshes instead of using a simpler intermediate domain to compose parameterizations. Progressive refinement is used to robustly create and optimize the inter-surface map. The refinement minimizes a distortion metric on both meshes. [68] present a method that relies on mesh refinement to establish a mapping between the models. First a mapping between patches over base mesh domains is computed and then mesh refinement is used to find a bijective parameterization. An advantage of this approach is that it naturally supports feature correspondence, since feature vertices are required as user input for the initial patch mapping. However, it requires user supervision and interaction whereas our method is fully automated.

In [71], reeb-graphs and boolean operations are used to extend spherical parameterization for handling models of arbitrary genus. Existing methods for producing valid spherical embeddings of genus-0 models can be integrated into their framework. In that respect, this work is orthogonal to our approach. Another method that uses reeb-graphs for morphing topologically different objects of arbitrary genus is [59]. The method specifies the correspondence between the input models by using graph isomorphic theory. The super Reeb graph, which has the equivalent topological information to the Reeb graphs of the two input objects, is constructed and used to conduct the morphing sequence.This method is very interesting from a theoretical point of view, but in practice the resulting matching may be unintuitive. Our method obtains intuitive matching results for similar objects and produces visually smooth morphing sequences.

Finally, [79] provide efficient techniques for morphing 3D polyhedral objects of genus-0. The emphasis of the method is on efficiency and requires the definition of feature patches to perform 2D mapping and subsequent merging. Their method does not avoid self intersection and requires embedding merging and user intervention for mapping. Our method overcomes these shortcomings in the expense of considerable increase in preprocessing time for mapping.

The method presented in this work overcomes the limitations of prior methods and allows for a totally automated and appropriate for morphing mapping of an object of genus-0 surface into a 2D space with spherical topology. An initial mapping over the unit sphere is computed and used as initial state and is then improved by employing nonlinear optimization. For smoother morphing that exploits the high level geometric structure we have introduced a feature-based approach. Feature correspondence is performed automatically without any user intervention.

4.3 Topology preserving spherical parameterization

4.3.1 Preliminaries

A *planar triangulation* is a simple triangulated plane graph whose edges are represented by straight lines. The triangulation is called *valid* when the only intersections between its edges are at the common endpoints. It has been shown by Fary et al [32] that every planar graph G has a

valid straight line representation. Therefore, for any planar graph G there exists a set of points p such that the induced triangulated graph $T(G,p)$ is valid. A way to construct such a valid triangulated graph is described in [112]. The boundary vertices of G are mapped to a convex polygon with the same number of vertices and in the same order. Then, the interior vertices are placed such that each vertex is the centroid of its neighboring vertices. This was extended by Floater et al [34] who has proven that each vertex v_i=(x_i,y_i) can be any convex combination of its N_i neighboring vertices (4.1).

Consequently, for finding a one-to-one bijective mapping for a mesh with an open boundary B to a convex parametric domain $P \in R^2$ (e.g. a unit disk), a sufficient condition is to find a set of positive weights that satisfy (4.1) and solve the corresponding linear system for those weights.

$$\begin{aligned} v_i &= \sum_{v_j \in N_i} w_{ij} v_j \\ \sum_{v_j \in N_i} w_{ij} &= 1 \\ w_{ij} &> 0 \end{aligned} \tag{4.1}$$

Theorem 4.1. *The linear system (4.1), which expresses the position of each node as a convex combination of its neighbors, has a unique solution if at least one of its nodes is fixed.*

Proof. See Appendix. ∎

Thus, the resulting system has always a unique solution provided that the boundary vertices are fixed. A straightforward choice is to choose equal weights resulting in each vertex representing the centroid of its neighbors. This is also called barycentric mapping. For a mesh $M(V,E)$, barycentric mapping minimizes the sum of the squares of edges lengths, with respect to a fixed boundary. This is due to:

$$\begin{aligned} f(v_1, v_2, ..., v_n) &= \sum_{(v_i,v_j) \in E} ||v_i - v_j||^2 \\ ||v_i - v_j||^2 &= (x_i - x_j)^2 + (y_i - y_j)^2 \end{aligned} \tag{4.2}$$

Since f is convex, it has a global minimum when $\partial f/\partial x_i = \partial f/\partial y_i = 0$ for $i = 1, ..., n$:

$$\frac{\partial f}{\partial x_i} = 2\sum_{v_j \in N_i}(x_i - x_j), \quad \frac{\partial f}{\partial y_i} = 2\sum_{v_j \in N_i}(y_i - y_j) \tag{4.3}$$

This is equivalent to solving the linear system (4.1) with equal weights:

$$\begin{aligned} \sum_{v_j \in N_i} w_i x_j &= x_i \\ \sum_{v_j \in N_i} w_i y_j &= y_i \\ w_i &= \frac{1}{|N_i|} \end{aligned} \tag{4.4}$$

The theory for planar parameterizations can be directly extended to a spherical domain by reducing the problem to the planar case. The parameterization is then computed in polar coordinates. One approach to alleviate the problem is to select two vertices as the poles (north and south) of the parameterization. Subsequently, a geodesic path must be established between the poles over the mesh surface. The path connecting the two poles defines the boundaries of the parameterization and thus the spherical surface can be converted to a unit disk. Therefore, we can directly use the previous analysis to compute one-to-one bijective mapping. If equal weights are chosen and the poles are selected based on the largest distance along the z direction in object space, the resulting system is the linear system proposed in [20]. This way a valid spherical parameterization can always be produced for every mesh. The quality of parameterization depends on the choices for the poles and the connecting path. It turns out that selecting a good path is a difficult problem that affects the distortion in the final parameterization.

Another approach is to cut out a triangle from the mesh, leaving an open boundary, and make the mesh homeomorphic to the unit disk. This approach, also referred to in literature as stereo mapping, usually results in very distorted parameterizations since using the corresponding unit triangle as a boundary tends to cluster the remaining vertices in the center of the triangle.

As explained, the main drawback of the previously described techniques is the unnecessary distortion introduced by the parameterization. Unfortunately, generalizing the barycentric coordinates and the planar parameterization theory to a spherical domain is not straightforward. Since the domain is non-planar, expressing a vertex on the sphere as a convex combination of its neighbors is not feasible in general. This would imply for example that if the neighbors of a vertex are co-planar, then the vertex should also lie on the same plane. Nevertheless, it turns out that the following holds:

Theorem 4.2. *If each vertex position is expressed as some convex combination of the positions of its neighbors projected on the sphere (4.5), then the formed spherical triangulation is valid.*

$$\begin{aligned} v_i &= \frac{\sum_{v_j \in N_i} w_{ij} v_j}{||\sum_{v_j \in N_i} w_{ij} v_j||} \\ \sum_{v_j \in N_i} w_{ij} &= 1 \\ w_{ij} &= w_{ji} \\ w_{ij} &> 0 \end{aligned} \tag{4.5}$$

Proof. This is an immediate result of Theorem 2 in [44]. ∎

Problem (4.5) can be expressed as a set of $3n-3$ non-linear equations for the nodes $i = 1, ..., n-1$ of the mesh, where $n = |V|$ is the number or vertices. The equation for the last vertex is redundant in the case of a connected triangular mesh. We should also introduce n equations that constrain the vertices to lie on the unit sphere. We then seek the positions of the vertices $v_i(x_i, y_i, z_i)$ and the n auxiliary variables a_i:

$$
\begin{aligned}
a_i x_i - \sum_{v_j \in N_i} w_{ij} x_j = 0, i = 1, \ldots, n-1 \\
a_i y_i - \sum_{v_j \in N_i} w_{ij} y_j = 0, i = 1, \ldots, n-1 \\
a_i z_i - \sum_{v_j \in N_i} w_{ij} z_j = 0, i = 1, \ldots, n-1 \\
x_i^2 + y_i^2 + z_i^2 = 1, i = 1, \ldots, n
\end{aligned}
\tag{4.6}
$$

In general, a solution to this system is not unique. Without removing some degrees of freedom there are infinite solutions due to the possible rotations over the sphere (three degrees of freedom). More importantly, there are degenerate solutions that satisfy (4.6). The most obvious one is observed when $a_i = 0$ where all vertices of the parameterization collapse to one point on the sphere. Another possible degenerate solution can occur when the mesh contains a Hamiltonian cycle and the vertices are mapped to the equator of the sphere. Other degenerate solutions also exist, see e.g. [44]. We can eliminate the infinite solutions if we fix three degrees of freedom (for example vertex v_n and an angle that will determine a unit circle on which a second vertex lies). However, even if we manage to avoid degenerate solutions we may still have a finite but exponentially large number of solutions (see e.g. [40, 107]) that we may have to eliminate by overconstraining or by introducing inequalities.

Even a robust and stable non-linear solver may converge to degenerate solutions for the system of equations (4.6). A key observation is that, as the solver iterations proceed, some triangles start growing and eventually pass through the equator of the sphere. The fundamental problem is that the spherical energy minimum occurs at a collapsed configuration. This situation occurs because the continuous spherical energy is approximated by a quadratic energy function calculated over the mesh triangles. Therefore, since the area of a planar triangle is always smaller than the area of the corresponding spherical triangle, an estimation error is introduced in the calculation of the energy over the surface. This error increases disproportionately with the size of the triangles. Therefore, the non-linear minimizer may minimize the corresponding distortion metric (energy function) over the sphere surface by increasing the size of the triangles with the largest error.

One common fix to avoid these degenerate solutions is to fix three or more vertices, thus constraining the solver. In practice however there are two problems, the extra constraints introduce additional distortion in the parameterization, and it is difficult in general to determine a proper set of fixed vertices. In addition, without paying special attention to the set of the constrained vertices, the non-linear problem may become infeasible.

Another important issue with the system of equations in (4.6) is that there is no guarantee that the solution is bijective in the case of non-symmetric or negative weights. Therefore, adapting the weights for morphing is difficult because weights for conformal mappings can be negative and weights for authalic mappings are non-symmetric [27].

Summarizing the above observations, if we try directly to solve (4.6), the following problems occur:

- *Non convexity.* The constraints $x_i^2 + y_i^2 + z_i^2 = 1$ are not convex. Therefore classical convex minimization cannot be used directly.

- *Non regularity.* Due to non-convex constraints, uniqueness and higher regularity of solutions cannot be expected [49].
- *Non uniqueness.* The energy does not have a unique minimum and degenerate solutions always exist.

An approach to tackle these difficulties was proposed in [38]. Here a penalty term d_{min}^{-2}, where d_{min} is the minimum distance of each triangle from the sphere center, was added in the corresponding planar quadratic energy and the constraints were removed. The motivation of this approach is to provide an upper bound of the spherical energy by scaling the corresponding planar energies of the triangles. Therefore, the corresponding problem (4.6) is transformed to an unconstrained one, that can be solved with standard methods. However, this approach is possible to restrict the minimize process and the convergence properties are unclear since no theoretical guarantee is provided.

In the method presented in this work, we overcome the above difficulties by employing a two step approach. An initial bijective parameterization over the unit sphere is computed and is used as an initial guess for a nonlinear optimization process. The optimized parameterization is guaranteed to be bijective by enforcing a proper set of constraints.

4.3.2 Initial spherical parameterization

A possible technique that one can use to obtain an initial parameterization is an iterative process that attempts to converge to a valid parameterization by applying local improvement (relaxation) [4]. The principle for this improvement is to reduce the spring energy of the points with Laplacian smoothing ignoring the sphere constraint and renormalise the solution to obtain valid spherical points. In practice however, the iterative process may converge to a degenerate solution and will then require a restart. Since Laplacian smoothing does not perform any triangle area balancing, certain elements may collapse leading to a degenerate solution. The following key observation motivates our approach,

Observation 4.1. *Iterative projected Laplacian smoothing collapses after one or more elements overgrow.*

The above observation motivates an area balancing procedure where the new position of the vertices is determined based on an area weighted sum. More specifically, the new position of each vertex is determined by the weighted sum of the centroids of the surrounding triangles, where the weights are determined by the area of each neighboring triangle. This approach yields a smoother mesh with more balanced element area since larger polygons tend to attract vertices, while smaller polygons tend to repulse them. Since the set of weights is positive, each individual folding is not stable and is forced to unfold according to the area weighted centroid attraction rule.

The above procedure is expressed concisely by the following steps:

1. Let $(v_1, v_2, ..., v_n)^0$ be an initial guess for the solution
2. For $j = 0$; until no folded elements exist; j++
   (a) Set $v_i^{j+1} = \sum_{v_k \in N_i} a_k c_k$ for $i = 1, ..., n$

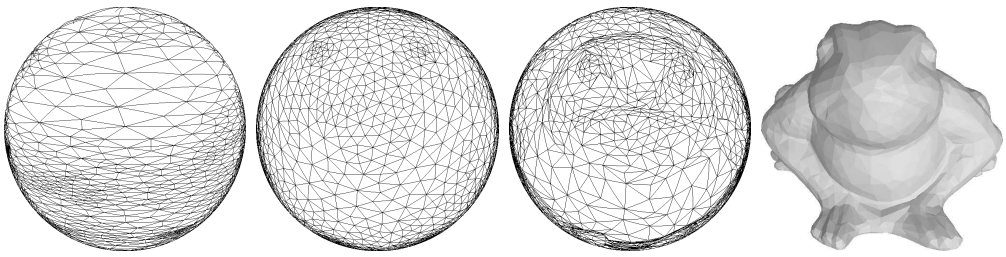
(b) Set $v_i^{j+1} = \frac{v_i^{j+1}}{||v_i^{j+1}||}$ for $i = 1, ..., n$

where a_k is the area of the corresponding k-th triangle that is adjacent to vertex v_i and c_k is the centroid of this triangle. The initial solution is obtained by normalizing the original vertex coordinates (assuming the object is centered at the origin),

$$v_i^0 = \frac{v_i}{||v_i||} \text{for } i = 1, ..., n \tag{4.7}$$

The normalizing denominator maintains vertices on the unit sphere.

We have used two alternative methods for obtaining an initial mapping: barycentric mapping and area weighted Laplacian smoothing. In barycentic mapping, two polar coordinates are determined for all vertices in two steps. Two vertices are selected as the poles (north and south) for this process. The poles must not be too close as this will result in a poor initial parameterization. Therefore, we have implemented this initial mapping by selecting as poles the vertex pair with the largest distance between them (diameter of the solid). In Laplacian smoothing, we use the area weighted variation. Figure 4.1 shows the results of the initial mapping when applying the planar barycentric mapping method and Laplacian smoothing on the frog from [1]. In general, Laplacian smoothing is faster and provides a robust unfolded initial mapping while preserving similarities with the initial mesh.



(a) (b) (c) (d)

Figure 4.1: (a) The result of the initial mapping using the planar barycentric method, (b) the Laplacian smoothing technique, (c) the result after optimization and (d) the original frog model.

Objective Function: We use as the objective function f to be minimized the sum of all dot products of every mapped vertex v_i^s with their corresponding initial position v_i^0 on the mesh.

$$f(v_1, v_2, ..., v_n) = \sum_{v_i \in V} v_i^0 \cdot v_i^s \tag{4.8}$$

4.3.3 Optimized parameterization for morphing

For the optimized mapping we use the following objective function and set of constraints that are appropriate for morphing:
Geometric Constraints: For each vertex v_i we use the spherical constraints from (4.6) to keep the vertices on the unit sphere surface.

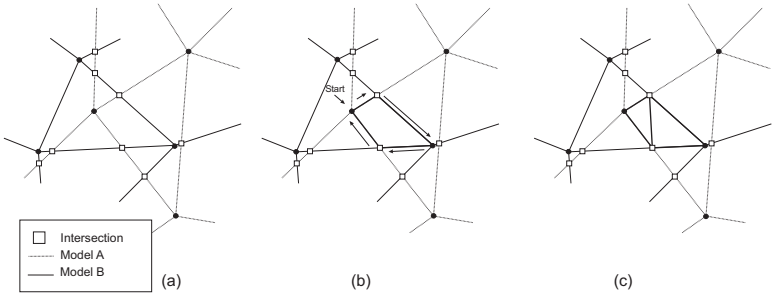


Figure 4.2: (a) Finding intersections in merged topology, (b) curve faces visited in clockwise manner and (c) triangulation

$$x_i^2 + y_i^2 + z_i^2 = 1 \tag{4.9}$$

Topological Constraints: For each face f_i of the mesh with vertices v_{i0}, v_{i1}, v_{i2} and for each vertex of this face, each vertex should stay on the same side of the plane defined by the other two vertices and the center of the sphere:

$$\begin{aligned} (v_{i1} \times v_{i2}) \cdot v_{i0} > 0 \\ (v_{i2} \times v_{i0}) \cdot v_{i1} > 0 \\ (v_{i0} \times v_{i1}) \cdot v_{i2} > 0 \end{aligned} \tag{4.10}$$

The goal of our optimization approach is to find a valid spherical parameterization suitable for morphing purposes. Since the minimization of the above objective function does not guarantee the validity of the final spherical parameterization, we must enforce this requirement with additional topological constraints (4.10). These constraints guarantee that the final parameterization is valid provided that we initiate the solver with a valid solution. In addition, by using a suitable tolerance $\epsilon > 0$ instead of 0, these constraints offer control over the area of each face in the final parameterization and thus degenerated elements are avoided.

The choice of the objective function is motivated by the observation that a mapping suitable for morphing should introduce more distortion in the concave areas. Therefore, the structurally important vertices of the mesh (for example those over the convex hull) are kept in their original projected positions on the sphere, whereas the distortion is concentrated on the less significant concave areas. Figure 4.1 illustrates the optimized mapping for the frog, while Figure 4.14(a) illustrates the final optimized mapping on the sphere for the Blender Suzanne model [18] and the head model [1]. The preservation of the initial characteristics is apparent.

To solve the problem (4.8) under the constraints (4.9) and (4.10) for large meshes it is important to have a stable and efficient numerical procedure. We use the Ipopt software [114], an implementation of the primal-dual interior point approach for nonlinear programming. This approach provides an efficient method for handling problems with large numbers of inequality constraints. Furthermore, interior-point methods allow convergence from poor starting points that may appear when computing an initial solution from methods such as the barycentric mapping (see Figure 4.1(a)). Under mild assumptions, it can be shown that such a procedure converges to a local solution of the original problem [35]. Moreover, since this is a convex optimization problem if a local minimun exists, then it is a global minimum. For all the examples the optimization process successfully terminated, satisfying the convergence tolerance error we

have used (10^{-6}). As far as efficiency is concerned the experimental evaluation of Section 4.6 indicates an $O(|V|^2)$ behavior for the NL optimization step.

4.4 Surface correspondence and interpolation

Following the successful mapping of two meshes M_A and M_B on the sphere, a merging process of the two topologies is performed. The purpose of this step is to create a final merged topology that is suitable for navigating back and forth to the original models.

This process requires each projected edge of one model to be intersected with each projected edge of the other. The algorithm to compute this step efficiently is based on the observation that starting from an intersection over an edge we can traverse all the remaining intersections by exploiting the topological information contained in the models. The complexity of this step is $O(E_A + K)$ where K is the total number of intersections.

From the intersections found, along with the vertices of the two models, a set of spherical regions bounded by circular arcs is determined. These regions are always convex, therefore it is straightforward to triangulate them. First for each edge, the list of intersections that belong to that edge is sorted by the distance from each vertex of the edge. Additionally, for each vertex, a list of the edges incident to it in clockwise order is calculated. Based on the aforementioned geometrical data we traverse each closed bounded region in a clockwise order and compute the triangulated merged topology in $O(K \log K)$ time complexity. Figure 4.2 illustrates this process.

The final step of the algorithm involves the projection of the merged topology back to the original models. For each model A the vertices of model B along with the intersection points are mapped back to A.

Following the successful establishment of a correspondence between the source and target vertices, the vertex positions are interpolated to acquire the final morphing sequence. To this end, we use simple linear interpolation. The advantage of linear interpolation, besides its simplicity, is that it can be efficiently realized on GPUs using a simple morphing shader for interpolating vertices and attributes (lighting, textures) in real-time. Nevertheless, linear interpolation may not always be desirable, especially in very complex meshes where self-penetrations may appear during the morphing sequence of the models. More advanced interpolation techniques are applied in such cases. Some of them are also implemented in shaders but their performance may vary depending on the limits set by the GPU.

4.5 Feature-based morphing

The overall process of feature based morphing is presented in Figure 4.3. First, the optimized spherical parameterizations are computed for both models (this step can be carried out as pre-processing and the mapping can be stored along with the mesh representation). Then feature regions are detected on both models using region growing and matched between the two models. Subsequently, feature point pairs are extracted and an optimized spherical parameterization is computed for the second model with respect to the feature point pairs. Finally, the actual morphing is carried out on the GPU.

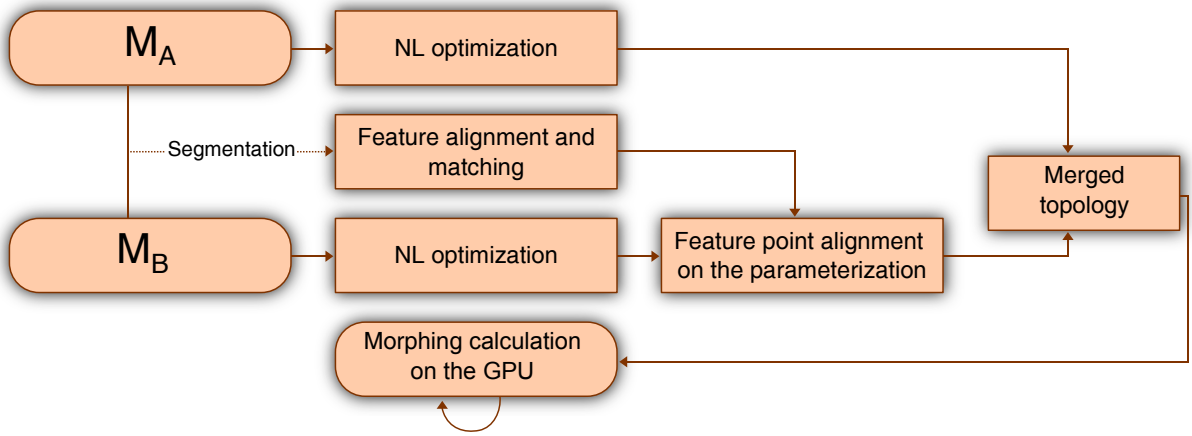


Figure 4.3: Overview of the feature-based algorithm.

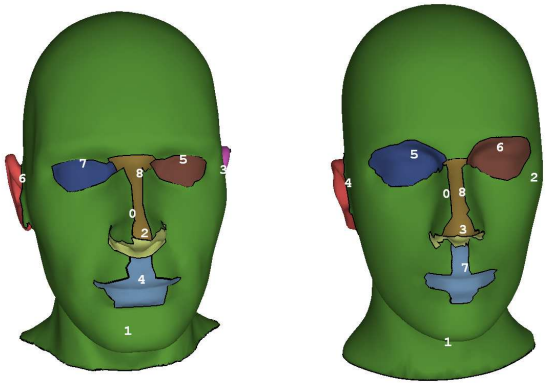
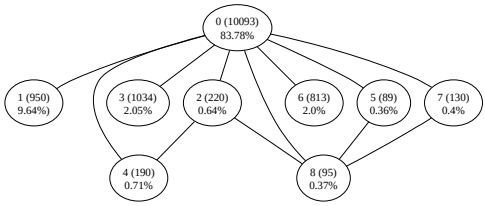
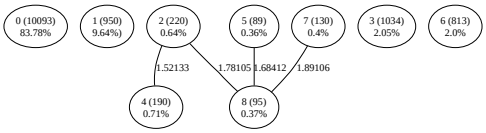


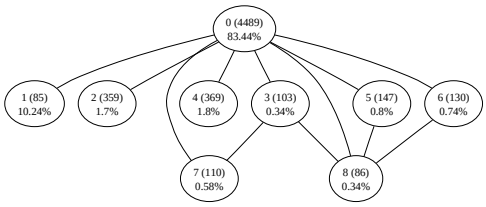
Figure 4.4: Detecting feature regions in two head meshes: (left) mesh M_A and (right) mesh M_B. Numbers correspond to identifiers for feature regions.



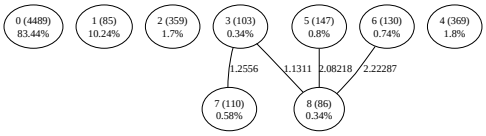
(a) Original adjacency graph of M_A showing the region number (see Figure 4.4), the number of nodes and the area covered in the original model.



(b) Reduced graph of M_A, all edges with large geodesic distances are eliminated.



(c) Original graph of M_B



(d) Reduced graph of M_B

Figure 4.5: Graph reduction of the head meshes.

To detect feature regions on meshes, we built on a method developed earlier in [110] for reverse engineering based on discovering features on the point cloud by detecting local changes in the structure of the point cloud. This method works even better on meshes, since in meshes vertex adjacency information is provided a priori.

We use region growing, detection of rapid variations of the surface normal and the concavity intensity and saddle points of the concavity intensity (the concavity intensity is the distance from the convex hull). This results in a number of regions that represent object feature regions (Figure 4.4). In the context of this work, we employ this method to detect features in models for the purposes of matching and alignment of the two meshes that are to be morphed.

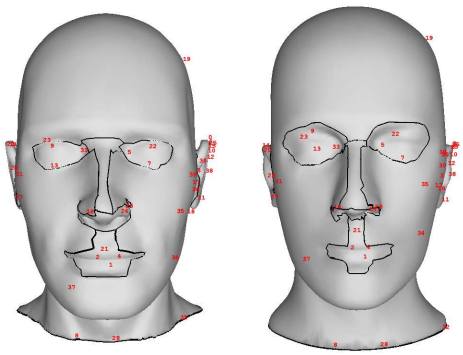


Figure 4.6: Detecting feature points inside feature regions.

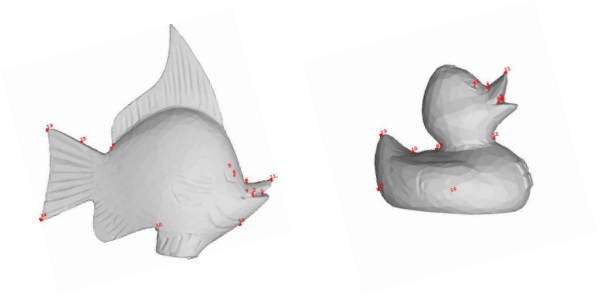


Figure 4.7: Feature point matching for the fish and the duck model.

More specifically several softer features on the meshes are detected using a characteristic called *concavity intensity* of a point which represents the smallest distance of a point from its convex hull.

Definition 4.1. The *concavity intensity* of a vertex v_i of a mesh denoted by $I(v_i)$ is the distance of v_i from the convex hull of the mesh.

This characteristic is used to detect soft convex or concave features on the mesh. The surface normal and the concavity intensity are used in conjunction with a region growing method that results in detecting sets of faces, called *feature regions* or simply *features* that correspond to areas with distinctive characteristics.(Figures 4.4 and 4.14(b)). Finally, we merge adjacent feature regions with similarity criteria, starting from features of small area. For a mesh with a set of vertices V, this process takes time $O(|V|)$ due to the planarity of the original graph and the almost linear behavior of the smaller-regions-merge-first practice.

After obtaining the features of the object, we create a connectivity graph that captures adjacency information as illustrated in Figure 4.5. For each edge, we calculate the geodesic distances between the centroids of the corresponding feature regions. The graphs are then simplified by reducing the edges that correspond to large geodesic distances to facilitate region matching (Figure 4.5). In addition, small regions that can introduce noise and are insignificant are merged to form larger regions and when this is not feasible they are eliminated. Elimination of large distances is motivated by the observation that usually meshes do not exhibit a general structural similarity but rather a local feature one. This process takes at most time $O(|R|^2)$ where R is the original set of feature regions.

In simple cases (Figure 4.4), where meshes have an almost identical structure, matching of the corresponding graphs is trivial. For more complex cases (see Figure 4.14), meshes possess only local structural feature similarities. Therefore, by eliminating the edges with large geodesic distances we match only local neighborhoods in the graph. These local neighborhoods still capture higher level information about the structure of the features, for example they detect eyes, nose and mouth similarities between completely different character models. Alternatively, for larger graphs subgraph matching algorithms were tested for detecting similar subgraphs in the two reduced adjacency graphs using randomized algorithms (see e.g. [15]). It remains to be determined whether such techniques are meritorious in terms of efficiency and scope. Finally, if one of the two graphs is of small size and it can be considered as a fixed pattern then a deterministic subgraph isomorphism algorithm may be used [30] which derives an $O(|R|)$ algorithm.

The reduced adjacency graphs are used to perform a 3D alignment of the two models and establish a correspondence between the regions. This is achieved by first matching the two highest degree nodes in the two graphs and then performing a 3D alignment of the two models. The remaining regions are paired according to their degree and the distance between them. Furthermore, we also take into consideration the area covered by each region by favoring the matching of regions covering similar areas. We have used the following heuristic similarity measure for matching,

$$s_{ij} = ||c_i - c_j|| \frac{\max\{a_i, a_j\}}{\min\{a_i, a_j\}} \frac{\max\{d_i, d_j\}}{\min\{d_i, d_j\}} \tag{4.11}$$

where c_i and c_j are the centroids of regions i and j, a_i and a_j are the corresponding areas and d_i and d_j are the degrees of the nodes in the reduced region adjacency graphs. This entire step for feature region alignment and matching takes time $O(|R_1||R_2|)$, where R_1, R_2 are the sets of nodes of the reduced graphs.

Moreover, for each feature region we detect points with certain properties that capture specific structural characteristics of the meshes. The resulting point set, called a *feature point set*, provides a summary of concave and convex regions of the object.

Definition 4.2. A vertex v_i is called a *feature point*, if and only if, $I(v_i)$ exhibits a local extremum at v_i.

Following the establishment of a correspondence between the region patches of the two models, the feature points of the corresponding patches are associated according to their distance. Since the patches may be in different locations in each model, the two regions are translated so that their corresponding centroids coincide. This step has worst case time complexity $O(|V|+|FP|^2)$, where FP is the set of feature point pairs. Figures 4.6, 4.7, and 4.14(c) illustrate the final feature point matching for different models.

So the overall time complexity for the feature matching is $O(|V| + |R|^2 + |FP|^2)$ and the space complexity is $O(|V| + |R| + |FP|)$.

For the feature based mapping of the second model we use the following objective function and set of constraints to obtain a more appropriate mapping based on the feature point correspondence of the models:

Objective Function: We use as the objective function to be minimized the sum of all dot products of every pair $p_i = (v_{i1}, v_{i2})$ of feature vertices $v_{i1} \in M_A$, $v_{i2} \in M_B$. Let FP be the set of pairs of feature vertices p_i.

$$\sum_{p_i \in FP} v_{i1} \cdot v_{i2} \tag{4.12}$$

Geometric Constraints: For each vertex v_i we use constraint (4.9).

Topological Constraints: In addition to equation (4.9), the length of each edge e_i (circular arc over the sphere) that connects the vertices v_{i1},v_{i2} must remain the same during optimization:

$$v_{i1} \cdot v_{i2} = v_{i1}^s \cdot v_{i2}^s \tag{4.13}$$

recall that v_i^s is the position of vertex v_i after the sphere optimization process. By doing so, we preserve the topology of the second object during the optimization process. In addition, this avoids very long stretches of triangles to satisfy a certain feature point pair matching. The same algorithm as in Section 4.3.3 is used for the optimization. This final nonlinear optimization step

Input: Input: Two triangular meshes M_A and M_B
for *each vertex* v_i *of* M_A **do**
    calculate $I(v_i)$
end
for *each vertex* v_j *of* M_B **do**
    calculate $I(v_j)$
end
for M_A *and* M_B **do**
    compute the corresponding feature region sets F_A and F_B
end
for F_A *and* F_B **do**
    compute the corresponding connectivity graphs and perform graph reduction on them
end
Establish a correspondence of the two nodes with the highest degree in the two graphs and perform a 3D alignment of F_1 and F_2 up to rotation based on that correspondence;
for *each feature region in* F_B **do**
    find a feature region in F_A using the similarity measure (4.11) and match the corresponding feature point sets
end
Calculate the spherical parameterization for M_A and M_B;
Optimize the spherical parameterization of M_B in order to match the paired feature points of the parameterizations;

Figure 4.8: The algorithm for feature based morphing.

is applied on the optimized spherical mapping of the second model. This is the only optimization step that cannot be performed as preprocessing and thus cannot be stored along with the model representation. Fortunately, however the convergence speed of this optimization step depends on the number of feature points and the number of faces of the second model, as it is indicated by the experiments in Section 4.6.

The method described above offers a way of automatically detecting a set of feature point pairs in structurally similar meshes to guide the morphing sequence. Generally, it is a difficult task to automatically find common features in every pair of shapes. The fundamental issue is that common features are usually defined through a semantic description and not through a geometric one. In addition, meshes may exhibit a global structural similarity instead of a local feature similarity.

Our underlying assumption for the graph matching step is that the relative placement of the features we want to detect, bears a resemblance between the two models as far as local features are concerned. Thus, our approach is sensitive in the level of detail that is used for the segmentation step. Although we try to exploit the high level representations of the objects in conjunction with low level geometric metrics (area and distances), in cases of global similarity we must tune the segmentation parameters to limit the number of the regions detected. Moreover,

we assume that the segmentation step results in a set of convex or almost convex regions and therefore the distances of the centroids are a reasonable approximation. Another problem may occur when there exists inherent symmetry in each mesh and as a result of this symmetry the matching process yields erroneous results.

To tackle these deficiencies a more general subgraph matching algorithm was examined, such as the randomized subgraph search algorithm [15], on a more detailed region graph. Nevertheless, our extensive experiments indicate that usually a rough matching of the feature regions gives satisfactory results. Therefore, whether such an extension would be beneficial in terms of accuracy and performance it remains to be determined. To summarize our current approach to feature-based morphing has the following limitations :

- Sensitivity to the segmentation step, since the high level representation may affect the matching of similar features.
- Intra object symmetry may yield non optimal feature point pairs.
- Semantically different features with similar connectivity and geometry can be matched.
- Very soft features that are not clearly identified by the segmentation algorithm may be left out.

The algorithm for feature-based morphing is presented in Figure 4.8. Figures 4.15 and 4.16 illustrate the visual improvement offered by this method.

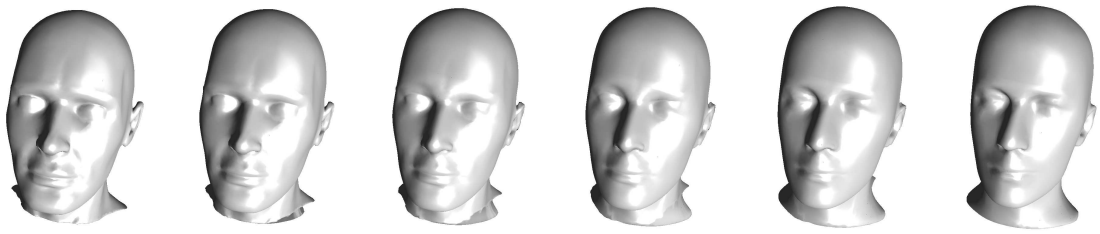


Figure 4.9: Morphing with alignment and feature point matching. Morphing is visually smooth through the entire sequence.

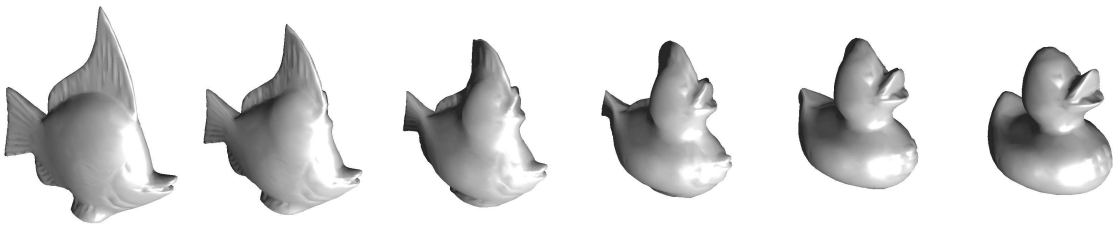


Figure 4.10: Morphing with alignment but no feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 28526 faces.

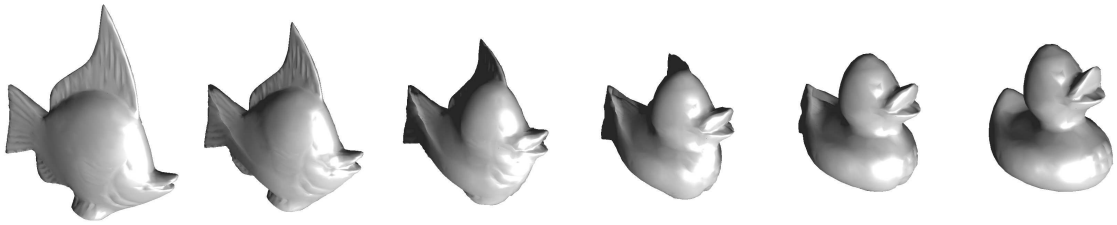


Figure 4.11: Morphing with alignment and feature point matching: fish (4994 faces) to duck (1926 faces), merged topology has 33038 faces.

Figure 4.12: Morphing with alignment but no feature point matching of the Charioteer model (11098 faces) to a Cycladic idol model (16798 faces), merged topology has 142422 faces.

Figure 4.13: Morphing with alignment and feature point matching, merged topology has 142512 faces

4.6 Experiments and performance evaluation

We have developed software for implementing mapping, merging and interpolation as described in the previous sections. The software is available at `http://www.cs.uoi.gr/~fudos/cagd2011.html`. The platform used for development consists of a Windows XP Professional based system running on a Intel Pentium Q6600 Core 2 at 2.4GHz, 2GByte of RAM, with NVIDIA GeForce 8600GT. We have developed the system on Visual Studio 2005, using OpenGL 2.0 (Shader Model 3.0) and GLUT.

Table 4.1 summarizes the results of some of our experiments on mapping for different models using both the barycentric method and the Laplacian smoothing initialization. $|V|$ is the number of vertices of the mesh, $|F|$ is the number of faces, and $|C|$ is the number of constraints for the optimization procedure. The number of iterations refers to the optimization phase, while time refers to the total time for both deriving the initial mapping and for performing optimization.

We observe that the Laplacian smoothing initialization yields a much faster convergence in the optimization phase (half the number of iterations and 50% faster). Our extensive experiments indicate that the number of iterations increases quadratically over the number of vertices of the polyhedral representation for triangular models. This is a considerable overhead but it can be calculated offline during a preprocessing phase and stored along with the polyhedral representation. Table 4.2 shows the results for the same set of experiments for the same model with different LODs ranging from 854 faces up to 5610 for the Suzanne model. This set of experiments confirms the above observations.

Finally, Table 4.3 shows the results of the feature guided optimization step. This step is significantly faster compared to the original parameterization and it depends on the number of feature point $|FP|$ and the number of faces of the second model. Another factor that affects the speed of convergence is the similarity of the two models. For example in Table 4.3 we observe a rapid convergence, in terms of iterations, for the case of the two similar head meshes (Figure 4.6).

Table 4.1: Experimental results of mapping with different models of various level of detail

model	method	$\|V\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Suzanne	Laplace	429	854	2991	36	10.9
Suzanne	Barycentric	429	854	2991	78	22.6
Bunny(Lod1)	Laplace	440	876	3068	94	24.3
Bunny(Lod1)	Barycentric	440	876	3068	165	52.0
Frog(Lod1)	Laplace	1964	3924	13736	70	422.2
Frog(Lod1)	Barycentric	1964	3924	13736	152	895.8

As mentioned in Section 4.4, merging takes in average $O(K \log K)$ time, where K is the number of intersections. For all cases in Tables 4.1 and 4.2, this step took less than 2.5 sec. Finally, the interpolation step is implemented in GPU so it is very fast and can accommodate almost unlimited number of frames.

Figures 4.9, 4.10 and 4.11 illustrate 3 different cases of morphing whereas Figure 4.14 shows the different steps to obtain the final morphing sequence. We have performed the experiments

Table 4.2: Experimental results with the same model with different levels of detail

model	method	$\|V\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Suzanne(Lod1)	Laplace	429	854	2991	36	10.9
Suzanne(Lod1)	Barycentric	429	854	2991	78	22.6
Suzanne(Lod2)	Laplace	703	1402	4909	26	21.3
Suzanne(Lod2)	Barycentric	703	1402	4909	70	54.8
Suzanne(Lod3)	Laplace	1404	2804	9816	49	151.3
Suzanne(Lod3)	Barycentric	1404	2804	9816	91	271.3
Suzanne(Lod4)	Laplace	2807	5610	19637	79	934.0
Suzanne(Lod4)	Barycentric	2807	5610	19637	136	1578.6

Table 4.3: Feature alignment optimization

model1	model2	$\|FP\|$	$\|F\|$	$\|C\|$	iterations	time (secs)
Fish	Duck	23	1926	9632	123	14.3
Head1	Head2	26	11040	55202	34	39.9
Head2	Suzanne	36	5610	25245	171	78.7
Head2	Caesar	60	13530	67652	182	321.1

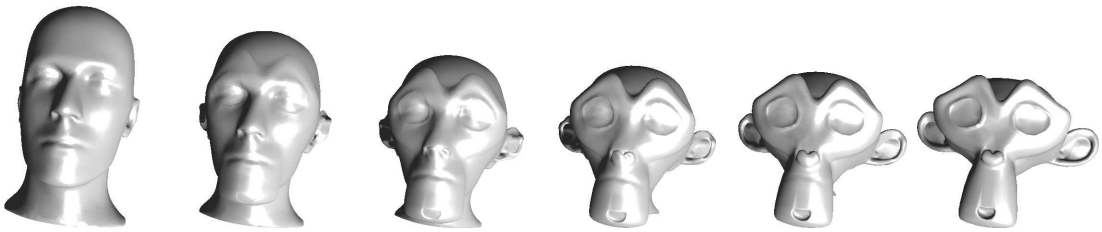
on well-known models such as the Stanford bunny [111], the Blender Suzanne [18] and the Aim@shape frog [1]. Finally, we have applied the algorithm to generate the morph sequence for two models obtained from a 3D scanner The results are illustrated in figures 4.12, 4.13 and 4.16. In addition to the geometry, the textures of the model were interpolated to produce the final morphing sequence.

4.7 Summary

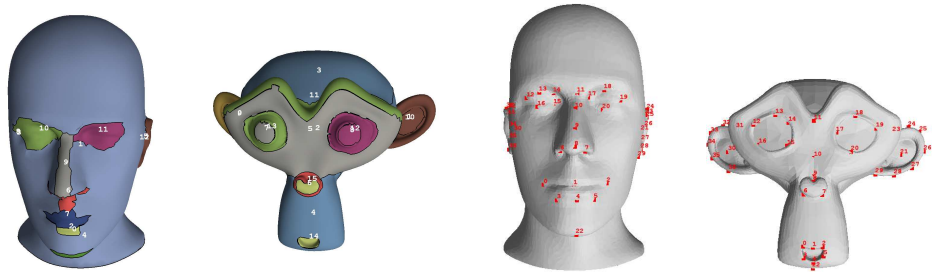
We have presented a method that performs morphing between arbitrary genus-0 objects without any user intervention. The sphere mapping can be considered as a preprocessing step and stored along with the representation of the solid. The merging is very fast in the average case, and the interpolation is implemented with GPU GLSL shaders. Finally, we have presented a fully automated technique for feature matching and alignment that greatly improves the visual effect and allows for applying controlled morphing to CAD model editing. We have used our method successfully on object pairs of similar topology (for examples busts) and of a quite different one (fish and duck). We are currently exploring the feasibility of parallelization through GPUs of the optimization phase and the use of user defined constraints for feature matching and morphing-based editing.

Furthermore, recent results on parallel computation of spherical parameterizations for mesh analysis [7] may adapted for fast accurate feature region detection. Finally, the benefits of using randomized subgraph matching algorithms for detecting common feature patterns [15] in objects with large number of features should be investigated further.

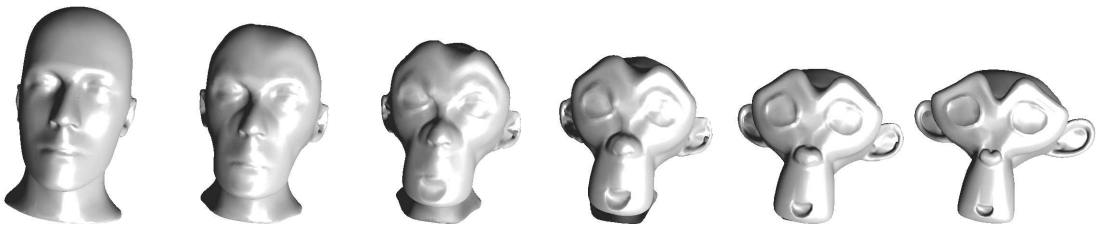
(a)



(b)

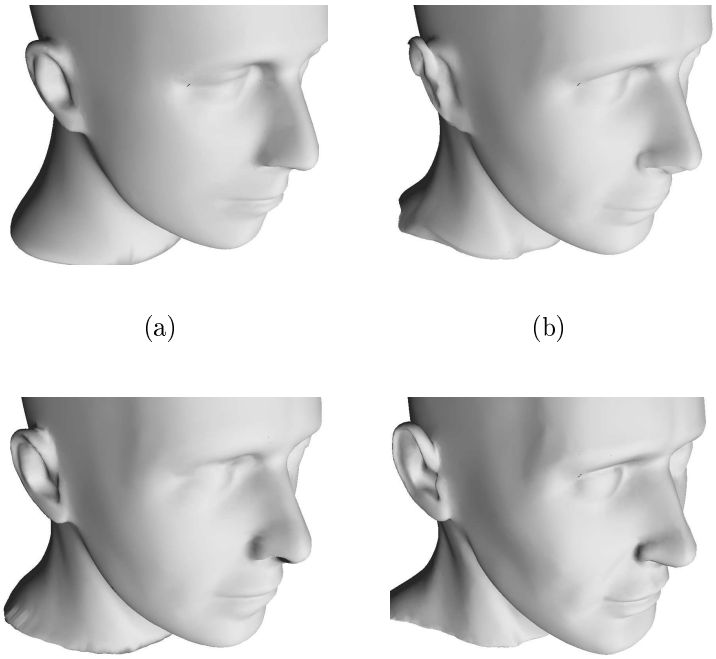


(c) (d)



(e)

Figure 4.14: An example for the entire feature-based morphing process: (a) (from left to right): The head model (11042 faces) and the optimized spherical parameterization, Suzanne (5600 faces) and the corresponding optimized spherical parameterization, and finally the optimized spherical parameterization of Suzanne with respect to the feature point pairs detected. (b) Morphing without feature point matching (the rightmost spherical parameterization of (a) is not used). (c) The feature regions are detected and paired. (d) Establishing feature point correspondence between the two meshes. (e) Finally, after optimizing one of the spherical parameterizations with the use of the feature point pairs, a visually smooth morphing sequence is obtained by linearly interpolating the vertices.



(a) (b)

(c) (d)

Figure 4.15: Close-up of the morphing sequence: (a) Model M_1, (b) 50% morph without feature point matching, (c) 50% morph with feature point matching and (d) target model M_2. The improvement around the ear area with feature point matching in (c) as compared to morphing without feature point matching in (b) is evident.

(a) (b) (c) (d)

Figure 4.16: Comparison of morphing results: (a) Model M_1, (b) 50% morph without feature point matching, (c) 50% morph with feature point matching and (d) target model M_2. The improvement around the ear area and the outline of the model with feature point matching in (c) as compared to morphing without feature point matching in (b) is apparent.

Chapter 5

Feature cut-and-paste: an application of planar constrained parameterization

5.1 Introduction

In the context of mesh editing, the cut-and-paste paradigm extracts a characteristic feature from a source surface and copies it on a target surface. The user usually selects a surface region which has two parts: the base surface and the detail surface. The base surface is a connected subset of the original surface and the detail surface is used as a feature to be copied/pasted. The goal is to replace the detail part of the target surface with the detail part of the source surface. The key question is how to transfer correctly the details from the source surface to the target. The target base surface may be altered as well to achieve smooth blending.

The detail surface can be stored either as a height-field or a parametric volume map [41]. The drawback of the height-field representation is that usually general features may be thick or contain overhangs and they can not be properly represented. To paste the detail surface to a target model, the corresponding vertices of the target model are moved based on the detail map. In 3D, a smooth attachment of boundaries between the pasted feature and its base is sometimes required. One possible way to resolve this issue is to perform a union operation between the two models and then apply a blending function along the boundaries of the features [86]. However, blending functions for arbitrary meshes is a difficult problem to solve efficiently and robustly. Snappaste [104] suggests an iterative algorithm for aligning the feature and the base surface, by positioning and deforming both surfaces. However they do not avoid the need for remeshing.

Another approach [120],[108],[19] is the modification of differential coordinates instead of directly changing spatial coordinates. The mesh geometry is then implicitly modified after reconstructing the surface from the differential coordinates. This method has the advantage of

reducing deformation artifacts that may appear after feature pasting. Nevertheless, sharp features are difficult to support. A tool for interactive cut-and-paste operations that uses the above approach is Mesh Mixer [99]. A distance preserving local parameterization is computed around the pasted area using approximate geodesic distances [98] and both the base surface and the feature are deformed using variational surface deformation techniques [19]. This method is very fast but suffers from some robustness issues as mentioned by the authors [98].

Existing cut-and-paste editing methods can be roughly categorized into two broad groups. The first group, uses mesh fusion to blend the source surface and target surface directly [58],[86]. The second group first extracts a base surface as a medium between the source surface and the target surface, and then transfers the details to the target surface via the base surface [41],[16],[39]. The former pays more attention to the smoothness of the boundaries at the joint of the source and target surfaces. The latter focuses on the global deformation of the source surface according to the target surface.

5.2 Our approach

In our approach, we have adapted tools and techniques from the differential coordinate and the mesh fusion scheme. We use an adaptive tessellation scheme that is built on top of the GPU tessellation unit. The tessellation is adaptive in the sense that only areas of interest are tessellated while the rest of the mesh remains untessellated. The feature area is parameterized with our non-linear solver and is subsequently stored in a 2D floating point texture. This texture is used in the tessellation evaluation shader to offset the base surface so as to create the feature in real time. The two main problems we address in this approach are: (i) how to support arbitrary features for pasting operations and (ii) how to smoothly blend the pasted feature and the base surface. Our approach is fast enough to support interactive operations.

The parameterization of the pasted feature should satisfy the following requirements to support arbitrary features:

- The boundary of the parameterization should be arbitrary. Convex boundary parameterizations are useful only for simple features and usually exhibit high distortion.
- The parameterization should be bijective and isometric. This allows us to store the feature in a 2D texture. The area and angular distortion of the parameterization should be minimal to avoid under sampling during the storage phase.
- Hard and soft constraints on the vertices should be handled robustly. This implies that the parameterization is bijective regardless of the constraints on the vertices (if such a parameterization exists).

All the above requirements can be satisfied by our non-linear solver described in chapter 2. Therefore, provided that we have parameterized the base area using one of the established methods, we proceed by computing a constrained parameterization of the feature. More specifically, we fix the boundary along with other user defined internal points on corresponding positions of the base surface parameterization. To find the specific correspondences we may follow either an automatic or a user-driven approach. To automatically compute the constraints we project the

feature boundary on the base surface and use the corresponding (s,t) parameterization coordinates of the base surface to constrain the feature parameterization. A limited user intervention may be necessary in cases of very complex boundaries. Alternatively, other methods can be used such as the snapping algorithm of [104].

Regarding the smooth attachment problem, a way to paste the feature without distorting its form is to treat the base surface as an elastic object and smoothly deform it to produce an appropriate area for pasting. The above observation led us to use a method that works very well in practice and produces intuitive results without deforming the feature surface. Our approach is based on the use of Radial basis functions (RBFs) [115],[22]. RBFs are a tool for interpolating data and are used to derive the displacement in any location in the space. RBF applications include mesh warping, medical imaging, and surface reconstruction [24],[25].

An RBF, s, is a function of the form:

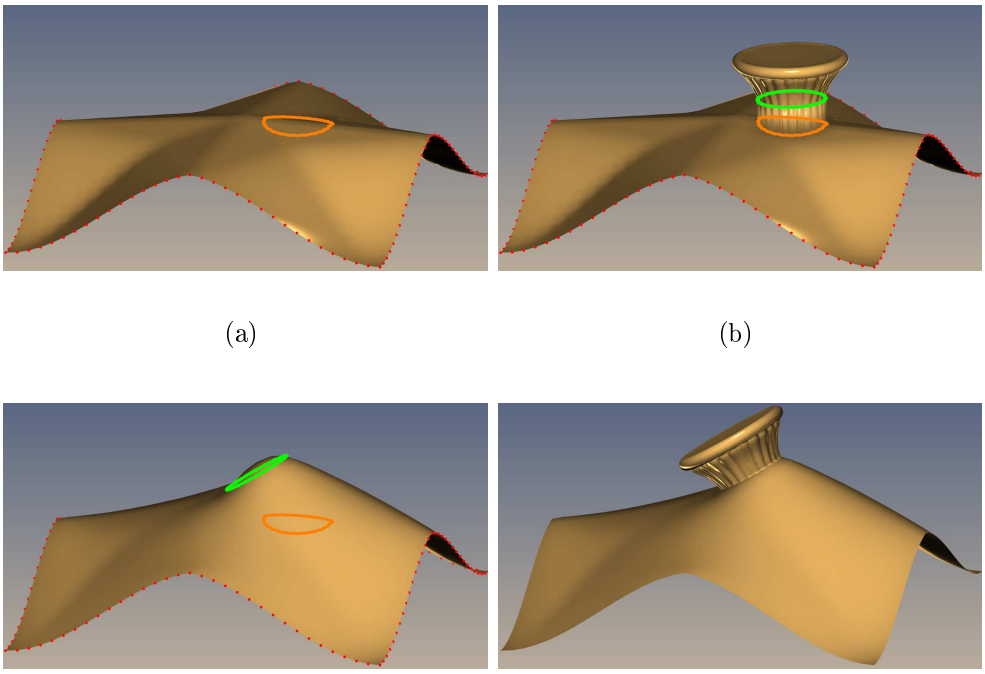
$$s(y) = \sum_{i=1}^{N} w_i \phi(||y - x_i||) + p_m(y) \tag{5.1}$$

where ϕ is called the basic function, w_i is a scalar coefficient, $x_1, ..., x_N$ are the pairwise distinct control points of the RBF, and $p(x)$ is an m degree polynomial. Popular choices of basic function include the thin-plane and the polyharmonic splines. To compute the coefficients we need to solve a linear system of order equal to the number of RBF control points (5.2).

$$\begin{bmatrix} \Phi & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w \\ \alpha \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{5.2}$$

where f are the known values at the control points, Φ is filled with the values of the basis function between the control points, and P depends on the polynomial used. Once the coefficients have been calculated, any arbitrary point can be expressed through the function s (5.1). In our experiments we have observed that the polyharmonic and the Wendland [116] basic functions produce the most natural looking results. Furthermore, we use two sets of control points. The first set of stationary control points is positioned at the boundaries of the base region and fixes the boundaries of the domain. The coordinates of each control point are used to set the right part. The second set of moving control points is positioned at the boundaries of the pasted feature. More specifically, we project the boundary points of the pasted feature on the base surface and use the projected points as control points. To set the right part for this set of control points we use the corresponding coordinates from the boundary of the feature. Having established an initial correspondence between the boundary of the feature and the base surface, we may apply an additional transformation on the feature to better place it on top of the surface. This additional transformation only affects the right part of the linear system. Therefore, for reasons of efficiency we compute and store the LU factorization only once and perform a back substitution when this transformation matrix changes. This way, the user can interactively move and rotate the feature on the base surface.

After transforming the base surface with RBFs the feature is pasted with the use of the tessellation unit. More specifically, in the tessellation control shader we sample the feature texture to decide if the base surface needs tessellation. If the base surface is tessellated, we offset the new vertices using the feature texture and apply the additional transformation matrix (if there is one).



(a) (b)

(c) (d)

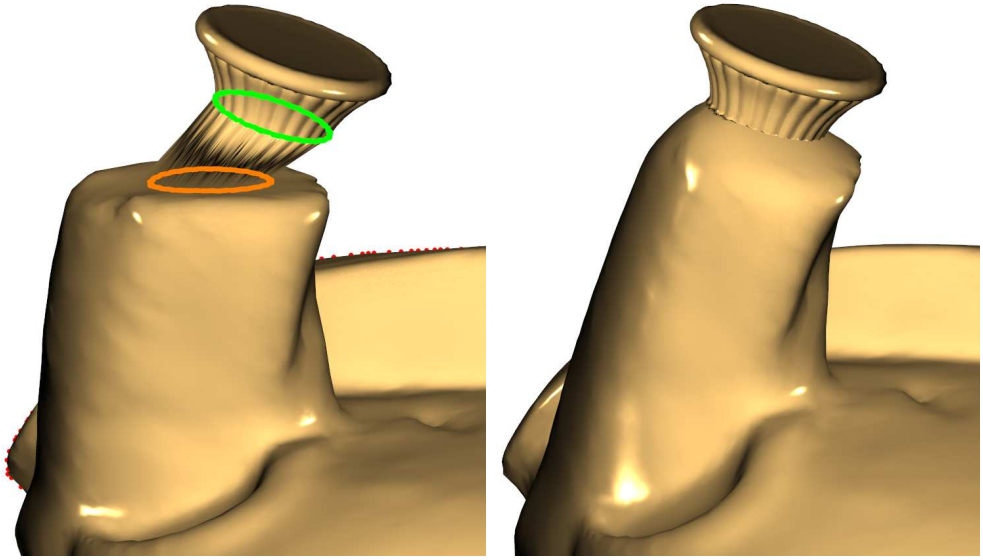
Figure 5.1: Overview of the pasting process. (a) Base surface, fixed and movable control points are colored red and orange respectively. (b) Pasted feature, the boundary of the feature is green. (c) We may apply an additional transformation on the feature and deform the base surface with RBFs. (d) Final pasting result.

The process of RBF interpolation and pasting is illustrated in Figure 5.1 whereas Figures 5.4 and 5.5 illustrate the deformation result for different base surfaces. The reader is also referred to the supplementary material that accompanies this work and is also available at `http://www.cs.uoi.gr/~fudos/smi2013.html`.

Having established an initial correspondence between the boundary of the feature and the base surface, we may apply an additional transformation on the feature to better place it on top of the surface. This additional transformation only affects the right part of the linear system. Therefore, for reasons of efficiency we compute and store the LU factorization only once and perform a back substitution when this transformation matrix changes. This way, the user can interactively move and rotate the feature on the base surface.

After transforming the base surface with RBFs the feature is pasted with the use of the tessellation unit. More specifically, in the tessellation control shader we sample the feature texture to decide if the base surface needs tessellation. If the base surface is tessellated, we offset the new vertices using the feature texture and apply the additional transformation matrix (if there is one).

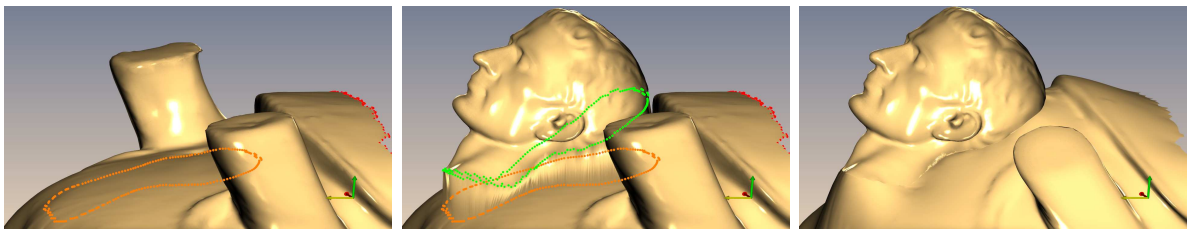
The process of RBF interpolation and pasting is illustrated in Figure 5.1 whereas Figures 5.4,5.5 and 5.6 illustrate the deformation result for different base surfaces. The reader is also referred to the supplementary material that accompanies this work and is also available at `http:`



(a) Undeformed (b) RBF Interpolation

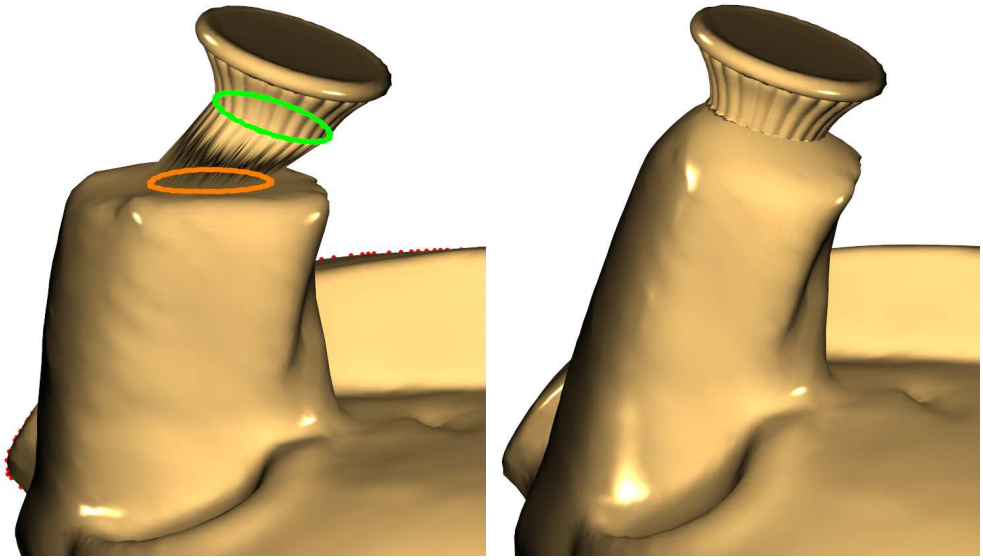
Figure 5.2: Base surface deformation.

`//www.cs.uoi.gr/~fudos/smi2013.html`.



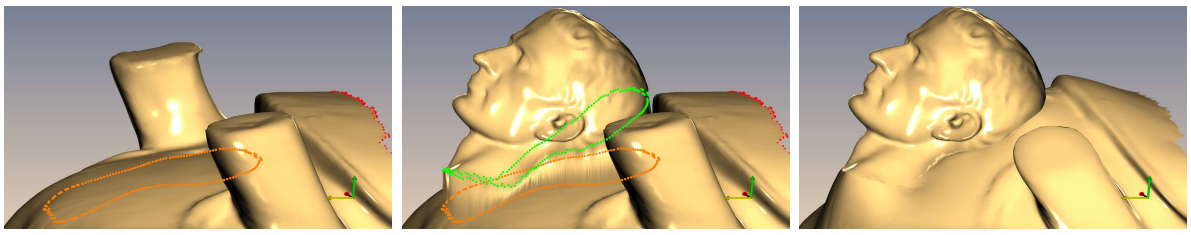
(a) Base surface (b) Cut-and-paste without RBF (c) RBF Interpolation

Figure 5.3: Cut-and-paste example.



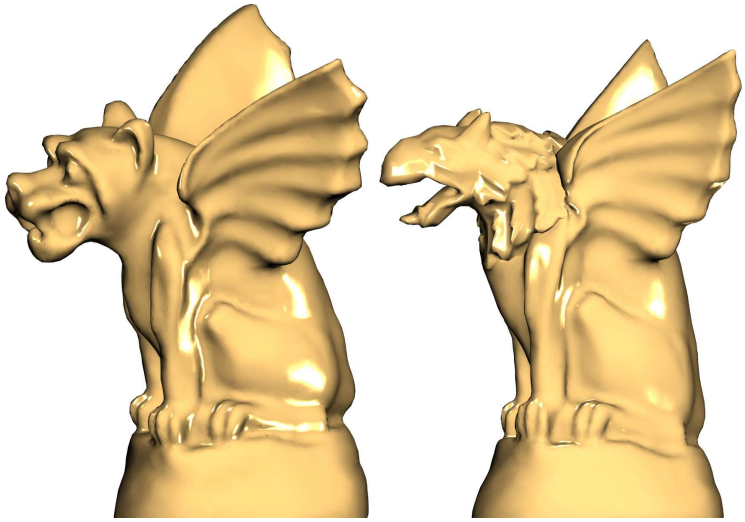
(a) Undeformed (b) RBF Interpolation

Figure 5.4: Base surface deformation.



(a) Base surface (b) Cut-and-paste without RBF (c) RBF Interpolation

Figure 5.5: Cut-and-paste example.



(a) (b)

Figure 5.6: Example of replacing the the head of the gargoyle mesh.

Chapter 6

Conclusions

6.1 Conclusions

Fast and robust constrained parameterization methods are essential for many applications. In this thesis, we have presented robust methods for efficiently computing planar and spherical parameterizations subject to soft and hard constraints.

In the domain of planar parameterization, our approach is based on establishing a theoretical connection between well studied mesh smoothing methods and the *isometric* parameterization problem. Based on our theoretical analysis we have implemented a novel non-linear solver that exploits the capabilities of modern GPUs for computing constrained isometric parameterizations. A possible direction for future research in that domain is to provide a formal proof regarding the convergence of the untangling process that is used for computing constrained parameterizations. This could be reached from connecting our process to standard gradient descent approaches and analysing the numerical instabilities that occur near the solution.

In the domain of spherical parameterization, we tackle the difficult non linear problem with non linear constraints by transforming the problem to a series of easier to solve saddle point problems. Based on this novel idea we have presented a simple and efficient numerical scheme to compute spherical parameterizations with the use of modern GPUs. Our method is at least an order of magnitude faster than state of the art spherical parameterization methods. A possible extension of our work would be a theoretical result of the convergence behavior. This could be reached from the fact that the algorithm is energy-decreasing so that the iterative solution follows a path close to the solution of the non-linear equations solved. In addition a thorough comparison of the proposed shape search using spherical parameterizations to other approaches, in terms of hits and misses, remains as future work.

Finally, the aforementioned methods have been used to perform morphing between arbitrary genus-0 objects without any user intervention and cut-and-paste operations of form-features. Moreover, we have presented a fully automated technique for feature matching and alignment

that greatly improves the visual effect and allows for applying controlled morphing to CAD model editing. As a future direction in that domain we intend to explore more complex cut-and-paste operations such as sliding features along surfaces. This would require the reparameterization of the feature with a set of dynamic constraints during the movement on the surface which is challenging.

Bibliography

[1] Aim@shape shape repository.

[2] E. Agathos, Ioannis Pratikakis, Stavros Perantonis, Nikolaos Sapidis, and Philip Azariadis. 3d mesh segmentation methodologies for cad applications. *Computer-Aided Design and Applications*, 4:827–842, 2007.

[3] Mehiddin Al-Baali. Improved hessian approximations for the limited memory bfgs method. *Numerical Algorithms*, 22:99–112, 1999.

[4] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[5] M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–197, 2002.

[6] G. Antini, S. Berretti, A. Del Bimbo, and P. Pala. 3d mesh partitioning for retrieval by parts applications. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1210 –1213, July 2005.

[7] Theodoros Athanasiadis and Ioannis Fudos. Parallel computation of spherical parameterizations for mesh analysis. *Computers & Graphics*, 35(3):569 – 579, 2011.

[8] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - a comparative study. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference*, pages 7 –7, June 2006.

[9] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181–193, 2006.

[10] P. N. Azariadis and N. S. Sapidis. Planar development of free-form surfaces: Quality evaluation and visual inspection. *Computing*, 72(1-2):13–27, 2004.

[11] Phillip N Azariadis and Nikos A Aspragathos. On using planar developments to perform texture mapping on arbitrarily curved surfaces. *Computers & Graphics*, 24(4):539–554, 2000.

[12] Mirela Ben-chen and Craig Gotsman. Characterizing shape using conformal factors. In *Eurographics Workshop on 3D Object Retrieval*, 2008.

[13] Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum*, 27(2):449–458, 2008.

[14] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, pages 1–137, 2005.

[15] Alexander Berner, Martin Bokeloh, Michael Wand, Andreas Schilling, and Hans-Peter Seidel. A graph-based approach to symmetry detection. In *Symposium on Volume and Point-Based Graphics*, pages 1–8, Los Angeles, CA, 2008. Eurographics Association.

[16] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph.*, 21(3):312–321, 2002.

[17] Hermann Birkholz. Shape-preserving parametrization of genus 0 surfaces, 2004.

[18] Blender. Blender Suite, Open Source Suite, http://www.blender.org, Blender Foundation.

[19] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, January 2008.

[20] Ch. Brechbuhler, G. Gierig, and O. Kubler. Parametrization of closed surfaces for 3d shape description. *Computer Vision and Image Understanding*, 61(2):154–170, August 1995.

[21] R.P. Brent. *Algorithms for Minimization Without Derivatives*. Dover books on mathematics. Dover Publications, 2002.

[22] M.D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[23] Beta CAE. ANSA, Pre-processing software, http://www.beta-cae.gr/ansa.htm.

[24] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 119–ff, New York, NY, USA, 2003. ACM.

[25] J.C. Carr, W.R. Fright, and R.K. Beatson. Surface interpolation with radial basis functions for medical imaging. *Medical Imaging, IEEE Transactions on*, 16(1):96 –107, feb. 1997.

[26] P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *In Proceedings of the 12th International Meshing Roundtable*, pages 201–213, 2003.

[27] Mathieu Desbrum, Mark Meyer, and Pierre Alliez. Intristic parameterization of surface meshes. In *Eurographics Proceedings*, pages 209–218, 2002.

[28] "N. Dyn and Jr." W. E. Freguson. The numerical solution of equality-constrained quadratic programming problems. *Mathematics of Computation*, 41(163):165–170, July 1983.

[29] Ilya Eckstein, Vitaly Surazhsky, and Craig Gotsman. Texture mapping with hard constraints. *Computer Graphics Forum*, 20(3):95–104, 2001.

[30] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3(3), 1999.

[31] J.M. Escobar, E. RodrΔ˝Mḛguez, R. Montenegro, G. Montero, and J.M. GonzΓɣlez-Yuste. Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2775 – 2787, 2003.

[32] I. Fary. On straight line representation of planar graphs. *Acta Univ. Szeged Sect. Sci. Math.*, 11:229–233, 1948.

[33] M. S. Floater. Mean value coordinates. In *CAGD*, pages 19–27, 2003.

[34] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[35] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44:525–597, 2002.

[36] Lori A. Freitag and Paul Plassmann. Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49(1-2):109–125, 2000.

[37] I. Friedel, Peter Schröder, and Mathieu Desbrun. Unconstrained spherical parameterization. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 134, New York, NY, USA, 2005. ACM.

[38] Ilja Friedel, Peter Schröder, and Mathieu Desbrun. Unconstrained spherical parameterization. *Journal of Graphics, GPU, and Game Tools*, 12(1):17–26, 2007.

[39] Hongbo Fu, Chiew lan Tai, and Hongxin Zhang. Topology-free cutand-paste editing over meshes. In *In Geometric Modeling and Processing 2004*, pages 173–182, 2004.

[40] Ioannis Fudos and Christoph M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Trans. Graph.*, 16(2):179–216, 1997.

[41] Yoshiyuki Furukawa, Hiroshi Masuda, Kenjiro T. Miura, and Hiroyuki Yamato. Cut-and-paste editing based on constrained b-spline volume fitting. *Computer Graphics International Conference*, 0:222, 2003.

[42] M. Garland, A. Willmott, and P.S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 49–58, New York, NY, USA, 2001. ACM.

[43] G. H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, 1996.

[44] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. In *ACM Transactions on Graphics 22*, pages 358–363, July 2003.

[45] X. Gu and S. T. Yau. Global conformal surface parameterization. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 127–137, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[46] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6:181–189, April 2000.

[47] T. He, S. Wang, and A. Kaufman. Wavelet-based volume morphing. In *Proceedings of Vizualization 1994*, pages 85–92, Washington D.C., October 1994. ieee.

[48] Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, December 1952.

[49] S. Hildebrandt, H. Kaul, and K. O. Widman. Dirichlet's boundary value problem for harmonic mappings of riemannian manifolds. *Mathematische Zeitschrift*, 147:225–236, 1976.

[50] C.M. Hoffmann and R. Joan-Arinyo. On user-defined features. *Computer Aided Design*, 30(5):321–332, 1998.

[51] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, Innovations in Applied Mathematics, pages 153–162. Vanderbilt University Press, Nashville, TN, 2000.

[52] Kai Hormann, Bruno Lévy, and Alla Sheffer. Mesh parameterization: theory and practice. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[53] Keisuke Inoue, Takayuki Itoh, Atsushi Yamada, Tomotake Furuhata, and Kenji Shimada. Face clustering of a large-scale cad model for surface mesh generation. *Computer-Aided Design*, 33(3):251 – 261, 2001.

[54] Martin Isenburg, Stefan Gumhold, and Craig Gotsman. Connectivity shapes. In *In IEEE Visualization 2001 Conference Proceedings (2001*, pages 135–142, 2001.

[55] Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509 – 530, 2005.

[56] Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Shape-based clustering for 3d {CAD} objects: A comparative study of effectiveness. *Computer-Aided Design*, 41(12):999 – 1007, 2009.

[57] T. Kanai, H. Suzuki, and F. Kimura. 3d geometric metamorhosis based on harmonic map. In *Proceedings of the 5th Pacific Computer Graphics and Applications*. IEEE, 1997.

[58] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive mesh fusion based on local 3d metamorphosis. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 148–156, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[59] P. Kanonchayos, T. Nishita, S. Yoshihisa, and T. L. Kunii. Topological morphing using reeb graphs. In *CW '02: Proceedings of the First International Symposium on Cyber Worlds (CW'02)*, page 0465, Washington, DC, USA, 2002. IEEE Computer Society.

[60] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22:954–961, July 2003.

[61] Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[62] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *Proceedings of SIGGRAPH 1992*, volume 26(2), pages 47–54. New York, Published as Computer Graphics, July 1992.

[63] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.*, 25(2):412–438, April 2006.

[64] Patrick M. Knupp. Matrix norms and the condition number: A general framework to improve mesh quality via node-movement. In *Eighth International Meshing Roundtable (Lake Tahoe, California*, pages 13–22, 1999.

[65] Patrick M. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, January 2001.

[66] Patrick M. Knupp. Introducing the target-matrix paradigm for mesh optimization via node-movement. In Suzanne Shontz, editor, *Proceedings of the 19th International Meshing Roundtable*, pages 67–83. Springer Berlin Heidelberg, 2010.

[67] A. F. Koschan. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *in Proceedings of the International Conference on Computer Vision and Pattern Recognition, II*, pages 27–32, 2003.

[68] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, 23(3):861–869, 2004.

[69] Vladislav Kraevoy, Alla Sheffer, and Craig Gotsman. Matchmaker: constructing constrained texture maps. *ACM Trans. Graph.*, 22(3):326–333, July 2003.

[70] T. Y. Lee and P. H. Huang. Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme. *IEEE Transactions of Visualization and Computer Graphics*, 9(1):85–98, 2003.

[71] Tong-Yee Lee, Chih-Yuan Yao, Hung-Kuo Chu, Ming-Jen Tai, and Cheng-Chieh Chen. Generating genus-n-to-m mesh morphing using spherical parameterization: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):433–443, 2006.

[72] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.P. Seidel. Intelligent mesh scissoring using 3d snakes. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, pages 279–287, Washington, DC, USA, 2004. IEEE Computer Society.

[73] A. Leonardis, A. Jaklič, and F. Solina. Superquadrics for segmenting and modeling range data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:1289–1295, November 1997.

[74] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorhosis. In *Proceedings of SIGGRAPH 1995*, pages 449–456. ACM SIGGRAPH, 1995.

[75] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, July 2002.

[76] X. Li, T.W. Woon, T.S. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 35–42, New York, NY, USA, 2001. ACM.

[77] J. Lien and Nancy M. Amato. Approximate convex decomposition of polyhedra. Technical report, In Proc. of ACM Symposium on Solid and Physical Modeling, 2005.

[78] J. M. Lien, J. Keyser, and N. M. Amato. Simultaneous shape decomposition and skeletonization. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling*, SPM '06, pages 219–228, New York, NY, USA, 2006. ACM.

[79] C. H. Lin and T. Y. Lee. Metamorphosis of 3d polyhedral models using progressive connectivity transformations. *IEEE Transactions of Visualization and Computer Graphics*, 11(1):2–12, 2005.

[80] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[81] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1495–1504, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[82] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH 87, published as Computer Graphics*, pages 163–169. ACM SIGGRAPH, 1987.

[83] S. D. Ma and H. Lin. Optimal texture mapping. In *Proceedings of Eurographics 88*, pages 421–428, 1988.

[84] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 27–34, New York, NY, USA, 1993. ACM.

[85] M. Mortara, G. Patane, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38:227–248, October 2003.

[86] Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. Level set surface editing operators. *ACM Trans. Graph.*, 21(3):330–338, 2002.

[87] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[88] NVIDIA. Opencl best practices guide, July 2009.

[89] NVIDIA. Cublas, 2012.

[90] A. Pichler and R.B. Fisher amd M. Vincze. Decomposition of range images using markov random fields. *ICIP*, pages 1205–1208, 2004.

[91] Emil Praun and Hugues Hoppe. Spherical parameterization and remeshing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 340–349, New York, NY, USA, 2003. ACM.

[92] Emil Praun and Hugues Hoppe. Spherical parametrization and remeshing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 340–349, New York, NY, USA, 2003. ACM.

[93] Shadi Saba, Irad Yavneh, Craig Gotsman, and Alla Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 258–267, 2005.

[94] Shadi Saba, Irad Yavneh, Craig Gotsman, and Alla Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 258–267, 2005.

[95] P. V. Sander, D. Nehab, and J. Barczak. Fast triangle reordering for vertex locality and reduced overdraw. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[96] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 409–416, New York, NY, USA, 2001. ACM.

[97] N.S. Sapidis and P.J. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Trans. Graph.*, 14:171–200, April 1995.

[98] Ryan Schmidt, Cindy Grimm, and Brian Wyvill. Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph*, 25:605–613, 2006.

[99] Ryan Schmidt and Karan Singh. meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pages 6:1–6:1, New York, NY, USA, 2010. ACM.

[100] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877, 2004.

[101] Subramani Sellamani, Ramanathan Muthuganapathy, Yagnanarayanan Kalyanaraman, Sundar Murugappan, Manish Goyal, Karthik Ramani, and Christoph M. Hoffman. Pcs: Prominent cross-sections for mesh models. *Computer Aided Design and Applications*, 7:601–620, 2010.

[102] D.F. Shanno and Kang-Hoh Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14:149–160, 1978.

[103] Avner Shapiro and Ayellet Tal. Polyhedron realization for shape transformation. *The Visual Computer*, 14(8/9):429–444, 1998.

[104] Andrei Sharf, Marina Blumenkrants, Ariel Shamir, and Daniel Cohen-Or. Snappaste: an interactive technique for easy mesh composition. *Vis. Comput.*, 22(9):835–844, September 2006.

[105] A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parameterization of triangular meshes. *Computing*, 72(1-2):185–193, 2004.

[106] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, April 2005.

[107] Meera Sitharam, Adam Arbree, Yong Zhou, and Naganandhini Kohareswaran. Solution space navigation for geometric constraint systems. *ACM Trans. Graph.*, 25(2):194–213, 2006.

[108] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[109] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. *ACM Trans. Graph.*, 27(3):77:1–77:11, August 2008.

[110] V. Stamati and I. Fudos. A feature-based approach to re-engineering objects of freeform design by exploiting point cloud morphology. In *Proc. SPM 2007*. ACM, Beijing, China 2007.

[111] Stanford. The Stanford 3D Scanning Repository, Stanford University, http://graphics.stanford.edu/data/3Dscanrep, Stanford Computer Graphics Laboratory.

[112] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc*, 13:743–768, 1963.

[113] T. Várady, M.A. Facello, and Z. Terék. Automatic extraction of surface structures in digital shape reconstruction. *Comput. Aided Des.*, 39:379–388, May 2007.

[114] Andreas Wachter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[115] H. Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[116] Holger Wendland and Angewandte Mathematik. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995.

[117] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):pp. 226–235, 1969.

[118] K. Wu and M.D. Levine. 3d part segmentation using simulated electrical charge distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(11):1223 –1235, November 1997.

[119] Shin Yoshizawa, Alexander Belyaev, and Hans peter Seidel. A moving mesh approach to stretch-minimizing mesh parameterization. *International Journal of Shape Modeling*, 11:25–42, 2005.

[120] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 644–651, New York, NY, USA, 2004. ACM.

[121] Y. Zhang, J. Paik, A. Koschan, and M. A. Abidi. A simple and efficient algorithm for part decomposition of 3d triangulated models based on curvature analysis. In *in Proceedings of the International Conference on Image Processing, III*, pages 273–276, 2002.

[122] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method*, volume 1-3. Butterworth Heinemann, 2005.

[123] Emanoil Zuckerberger, Ayellet Tal, and Shymon Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733 – 743, 2002.

[124] M. Zwicker and C. Gotsman. Meshing point clouds using spherical parameterization. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, Zurich, June 2004.

Appendix

A. Equivalence of shape metric and MIPS

$$\begin{aligned}
\|\mathbf{S}'\|_{\mathbf{F}}^2 = trace(\mathbf{S}'^T\mathbf{S}') &\overset{(2.13)}{=} (\vec{v_1}-\vec{v_0})\cdot(\vec{v_1}-\vec{v_0}) + ((\vec{v_2}-\vec{v_1})\cot\hat{\alpha}+ \\
&(\vec{v_2}-\vec{v_0})\cot\hat{\beta})\cdot((\vec{v_2}-\vec{v_1})\cot\hat{\alpha}+(\vec{v_2}-\vec{v_0})\cot\hat{\beta}) \\
&= \|\vec{v_1}-\vec{v_0}\|_2^2 + \|\vec{v_2}-\vec{v_1}\|_2^2\cot\hat{\alpha}^2 + \|\vec{v_2}-\vec{v_0}\|_2^2\cot\hat{\beta}^2+ \\
&2(\vec{v_2}-\vec{v_1})\cdot(\vec{v_2}-\vec{v_0})\cot\hat{\alpha}\cot\hat{\beta} \\
&\overset{(CL)}{=} \|\vec{v_1}-\vec{v_0}\|_2^2 + \|\vec{v_2}-\vec{v_1}\|_2^2\cot\hat{\alpha}^2 + \|\vec{v_2}-\vec{v_0}\|_2^2\cot\hat{\beta}^2+ \\
&(\|\vec{v_2}-\vec{v_1}\|_2^2 + \|\vec{v_2}-\vec{v_0}\|_2^2 - \|\vec{v_1}-\vec{v_0}\|_2^2)\cot\hat{\alpha}\cot\hat{\beta}
\end{aligned} \tag{6.1}$$

where we use the *cosine law* (CL) on the last equation, if we further define the triangle edge lengths as :

$$a = \|\vec{v_2}-\vec{v_1}\|_2, b = \|\vec{v_2}-\vec{v_0}\|_2, c = \|\vec{v_1}-\vec{v_0}\|_2 \tag{6.2}$$

it follows that :

$$\begin{aligned}
\|\mathbf{S}'\|_{\mathbf{F}}^2 &\overset{(6.1),(6.2)}{=} (1-\cot\hat{\alpha}\cot\hat{\beta})c^2 + (\cot\hat{\alpha}+\cot\hat{\beta})(a^2\cot\hat{\alpha}+b^2\cot\hat{\beta}) \\
&= c^2\frac{\sin\hat{\alpha}\sin\hat{\beta}-\cos\hat{\alpha}\cos\hat{\beta}}{\sin\hat{\alpha}\sin\hat{\beta}} + \\
&(a^2\cot\hat{\alpha}+b^2\cot\hat{\beta})\frac{\cos\hat{\alpha}\sin\hat{\beta}+\sin\hat{\alpha}\cos\hat{\beta}}{\sin\hat{\alpha}\sin\hat{\beta}} \\
&= c^2\frac{-\cos(\hat{\alpha}+\hat{\beta})}{\sin\hat{\alpha}\sin\hat{\beta}} + (a^2\cot\hat{\alpha}+b^2\cot\hat{\beta})\frac{\sin(\hat{\alpha}+\hat{\beta})}{\sin\hat{\alpha}\sin\hat{\beta}} \\
&= c^2\frac{-\cos(\pi-\hat{\gamma})}{\sin\hat{\alpha}\sin\hat{\beta}} + (a^2\cot\hat{\alpha}+b^2\cot\hat{\beta})\frac{\sin(\pi-\hat{\gamma})}{\sin\hat{\alpha}\sin\hat{\beta}} \\
&= c^2\frac{\cos\hat{\gamma}}{\sin\hat{\alpha}\sin\hat{\beta}} + (a^2\cot\hat{\alpha}+b^2\cot\hat{\beta})\frac{\sin\hat{\gamma}}{\sin\hat{\alpha}\sin\hat{\beta}} \\
&= \frac{\sin\hat{\gamma}}{\sin\hat{\alpha}\sin\hat{\beta}}\big(c^2\cot\hat{\gamma}+a^2\cot\hat{\alpha}+b^2\cot\hat{\beta}\big) \\
&= (\cot\hat{\alpha}+\cot\hat{\beta})\big(c^2\cot\hat{\gamma}+a^2\cot\hat{\alpha}+b^2\cot\hat{\beta}\big)
\end{aligned} \tag{6.3}$$

B. Solving Linear Systems with Laplacian Smoothing

Let (x_i, y_i) denote the coordinates of the ith node of a mesh. In addition, let the coordinates of its adjacent vertices be $(x_j, y_j) : v_j \in N_i$, where N_i denotes the set of neighbors of node v_i.

If we assign a set of positive weights w_{ij}, where w_{ij} is the weight of a neighbor node j when determining node i then,

We will prove *Theorem 1*, i.e. that the linear system (4.1), which expresses the position of each node as a convex combination of its neighbors, has a unique solution if at least one node is fixed.

Proof. Let b and m represent the numbers of boundary (or fixed) and interior (or free) nodes, respectively. Next, define x_B and y_B to be vectors of length b that contain the initial x and y coordinates of the boundary nodes. Similarly, define x_I and y_I to be the vectors of length m that contain the initial x and y coordinates of the interior nodes. Thus, $[x_B|y_B]$ and $[x_I|y_I]$ contain the original positions of the boundary and interior nodes respectively. The weighted matrix L, for the graph $G(V;E;w)$ is:

$$L(i,j) = \begin{cases} -w_{ij}, & i \neq j \\ \sum_{k \in V} w_{ik}, & i = j \end{cases} \qquad w_{ij} = 0, \quad (i,j) \notin E \tag{6.4}$$

where the boundary nodes are placed in last b rows and columns $m+1, \ldots, m+b$, i.e. after the interior nodes which are placed in the first m rows and columns $1, \ldots, m$. Let $A = [A^I|A^B]$ be the matrix that is derived from the weighted matrix L by deleting its last b rows. Then, the linear system (4.1) is expressed as:

$$A^I[x_I|y_I] = -A^B[x_B|y_B] \tag{6.5}$$

where A^I is an $m \times m$ matrix that contains all the weights corresponding to the interior neighbors. In addition, A^B is an $m \times b$ matrix contains all of the weights corresponding to the boundary neighbors. Because the mesh is connected and a positive weight is associated with each edge, A^I is irreducible. In addition, $|a^I_{ii}| \geq \sum_{j=1, j\neq i}^{m} |a^I_{ij}|$ for each row because the diagonal elements are 1, and the off-diagonal elements are negative summing to a value in $[-1, 0]$. Equality is true if the corresponding vertex is connected only with interior nodes. Therefore, if $b > 0$, there exist i such that $|a^I_{ii}| > \sum_{j=1, j\neq i}^{m} |a^I_{ij}|$, and A^I is weakly dominant. Thus, A^I is invertible and has a unique solution [43], if there is at least one boundary node.

Moreover, it can be shown that the *Jacobi* iteration for each row i represents a step of the *simultaneous version* of Laplacian smoothing, where all the positions are modified simultaneously,

$$[x_I|y_I]_i^{k+1} = \sum_{\substack{j=1 \\ j\neq i}}^{m} w_{ij}[x_I|y_I]_j^k + \sum_{j=m+1}^{m+b} w_{ij}[x_B|y_B]_j \tag{6.6}$$

Similarly, it can be shown that the iterations produced by the *Gauss-Seidel* method are the same as the *sequential version* of Laplacian smoothing, where the positions are modified sequentially and depend on the order in which the vertices are considered. Since A^I is irreducible and weakly dominant and thus invertible, both Jacobi and Gauss-Seidel methods converge [43], and thus Laplacian smoothing converges to the same point which is the solution of the linear system (6.5). ∎

Index

Author's Publications

Journals

1. D. Vartziotis, T. Athanasiadis, I. Goudas, and J. Wipper. Mesh smoothing using the geometric element transformation method. *Computer Methods in Applied Mechanics and Engineering*, 197:3760–3767, 2008.

2. Theodoros Athanasiadis and Ioannis Fudos. Parallel computation of spherical parameterizations for mesh analysis. *Computers & Graphics*, 35(3):569 – 579, 2011.

3. Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou, and Vasiliki Stamati. Feature-based 3d morphing based on geometrically constrained spherical parameterization. *Computer Aided Geometric Design*, 29(1):2 – 17, 2012.

4. Theodoros Athanasiadis, Georgios Zioupos, and Ioannis Fudos. Efficient computation of constrained parameterizations on parallel platforms. *Computers & Graphics*, 37(6):596 – 607, 2013.

Conferences

1. Theodoros Athanasiadis, Ioannis Fudos, Christophoros Nikou, and Vasiliki Stamati. Feature-based 3d morphing based on geometrically constrained sphere mapping optimization. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1258–1265, New York, NY, USA, 2010. ACM.

2. Theodoros Athanasiadis and Ioannis Fudos. Parallel computation of spherical parameterizations for mesh analysis. In *Shape Modeling International 2011 (SMI), Herzliya, Israel*, June 2011.

3. Theodoros Athanasiadis and Ioannis Fudos. Parallel computation of spherical parameterizations for mesh analysis. Presented as poster during *Barcelona Computing Week 2011, "Programming and Tuning Massively Parallel Systems", Barcelona, Spain*, July 2011. (NVIDIA Best Poster award)

4. Theodoros Athanasiadis, Georgios Zioupos, and Ioannis Fudos. Efficient computation of constrained parameterizations on parallel platforms. In *Shape Modeling International 2013 (SMI), Bournemouth, UK (Shape Modeling Best Paper award)*. July 2013.

Short Vita

Theodoros Athanasiadis was born on May 14, 1983 in Thessaloniki. He is currently pursuing his PhD at the Computer Science Department of University of Ioannina. He received his BSc and MSc degrees from the same institution in 2006 and 2009 respectively. His PhD studies are supported by a Heracleitus II grant. In addition, he received two awards during his PhD studies. More specifically, he received the NVIDIA Best Poster award during the Barcelona Computing Week 2011 for his work "Parallel computation of Spherical Parameterizations for Mesh Analysis" and the Computer & Graphics Best Paper award for his work "Efficient Computation of Constrained Parameterizations on Parallel Platforms" during Shape Modeling International 2013.

His research interests include parallel computing on GPUs, computational and differential geometry, and real-time graphics.