2011

3D mesh metamorphosis from spherical parameterization for conceptual design

Ruqin Zhang
Iowa State University

3D mesh metamorphosis from spherical parameterization for conceptual design

by

Ruqin Zhang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Human Computer Interaction; Mechanical Engineering

Program of Study Committee:
James Oliver, Co-major Professor
Eliot Winer, Co-major Professor
Judy Vance
Song Zhang
Chris Harding

Iowa State University

Ames, Iowa

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Engineering product design is an information intensive decision-making process that consists of several phases including design specification definition, design concepts generation, detailed design and analysis, and manufacturing. Usually, generating geometry models for visualization is a big challenge for early stage conceptual design. Complexity of existing computer aided design packages constrains participation of people with various backgrounds in the design process. In addition, many design processes do not take advantage of the rich amount of legacy information available for new concepts creation.

The research presented here explores the use of advanced graphical techniques to quickly and efficiently merge legacy information with new design concepts to rapidly create new conceptual product designs. 3D mesh metamorphosis framework "3DMeshMorpher" was created to construct new models by navigating in a shape-space of registered design models. This efficient software framework enables designers to create numerous geometric concepts in real time with a simple graphical user interface.

The framework is composed of: 1) a fast spherical parameterization method to map a geometric model (genus-0) onto a unit sphere; 2) a geometric feature identification and picking technique based on 3D skeleton extraction; and 3) a LOD controllable 3D remeshing scheme with spherical mesh subdivision based on our spherical parameterization.

Our spherical parameterization is focused on closed genus-zero meshes. The method is based upon barycentric coordinates with convex boundary. Unlike most existing similar

approaches which deal with each vertex in the mesh equally, the method developed in this research focuses primarily on resolving overlapping areas, which helps speed the parameterization process. The algorithm starts by normalizing the source mesh onto a unit sphere and followed by some initial relaxation via Gauss-Seidel iterations. Due to its emphasis on solving only challenging overlapping regions, this parameterization process is much faster than existing spherical mapping methods.

To ensure the correspondence of features from different models, we introduce a skeleton based feature identification and picking method for features alignment. Unlike traditional methods that align single point for each feature, this method can provide alignments for complete feature areas. This could help users to create more reasonable intermediate morphing results with preserved topological features. This skeleton featuring framework could potentially be extended to automatic features alignment for geometries with similar topologies. The skeleton extracted could also be applied for other applications such as skeleton-based animations.

The 3D remeshing algorithm with spherical mesh subdivision is developed to generate a common connectivity for different mesh models. This method is derived from the concept of spherical mesh subdivision. The local recursive subdivision can be set to match the desired LOD (level of details) for source spherical mesh. Such LOD is controllable and this allows various outputs with different resolutions. Such recursive subdivision then follows by a triangular correction process which ensures valid triangulations for the remeshing. And the

final mesh merging and reconstruction process produces the remeshing model with desired LOD specified from user. Usually the final merged model contains all the geometric details from each model with reasonable amount of vertices, unlike other existing methods that result in big amount of vertices in the merged model. Such multi-resolution outputs with controllable LOD could also be applied in various other computer graphics applications.

CHAPTER 1

INTRODUCTION

1.1 Engineering Design Process

Product design is an information intensive engineering process of decision-making. It is estimated that as much as 75% of the cost of a product is spent during the product design phase including manufacturing and maintenance [1]. Companies are increasingly using digital prototypes from CAD [2] rather than manufacturing expensive physical models earlier in the product development process. Product design usually starts with the definition of a design problem, followed by a sequence of approaches to find an optimal solution and ends with a detailed description of the product. A design process can be divided into several phases. The first is collecting and defining design specifications about the product such as performance, quality, and safety. The second is concept generation where rough design concepts are proposed to meet the design specifications. Next is detailed design, where all design specifics such as part dimensions, material specification, and assembly arrangement, are finalized. These 3D product models form the basis for detailed performance analysis, manufacturing planning, and all other product life-cycle activities such as production and maintenance. Many computer tools have been developed to assist design and analysis at the detailed stage of design [3, 4, 5], whereas concept generation and selection are still mostly dependent upon experience of engineers and use of tools not built to handle the requirements of concept generation.

1.2 Motivation and Contribution

Usually, new product development is a process which not only is concerned with product design, but also includes prototyping, manufacturing, distribution and service. Such processes always involves people with different backgrounds (engineer, designer, worker, consumer and etc.). This kind of collaborative context enables interactions among different criteria from different disciplines. With this kind of collaborative product context, there is a need of visualization for product shapes to truly keep stakeholders on the same page.

A description of a conceptual design can be decomposed into various aspects including function, behavior, and structure [10]. To generate and select the feasible solutions, it is necessary to determine the correlations and interactions among these aspects. Computers have been used extensively in areas of simulation, modeling, and optimization, but there are relatively few applications at the conceptual design stage [11] due to the lack of knowledge of design specifications and constraints. This lack of knowledge causes two inherent difficulties in conceptual design activities: a) modeling interactions between components and b) reasoning to generate and select feasible solutions.

Generating geometry models for visualization is a big challenge for early stage conceptual design. Existing tools for this include traditional computer aided design (CAD) packages, sketch-based 3D geometry creation tools (e.g. Google SketchUp), and geometry manipulation and deformation techniques (e.g. free-form deformation, direct manipulation).

These solutions provide designers different ways to generate new design concepts. But they all have one common problem: the difficulty to bring people from different backgrounds into the design process. These tools mostly aim to those design engineers and unfriendly to other people who are also willing to contribute in conceptual design since the learning curve is usually too steep.

Another issue for many conceptual design tools is the inability to integrate legacy geometric models. While some designs can be started simply from a sketch or geometric primitives larger, more complex designs require more to be created quickly. Legacy information can be a critical component to rapidly creating conceptual geometry that meets the needs of a diverse group of stakeholders.

In this work, there are two ways of the usage for the legacy data. The advanced systems design suite (ASDS) makes use of the legacy geometries as well as some meta data. The legacy geometries and some primitive models work together and are employed for new geometry creation. And the meta data will assist some physical evaluations for design in early stage. The second way of using legacy models is implemented in our software "3DMeshMorpher". The idea is by developing a 3D mesh metamorphosis framework, the users can navigate in a shape-space of registered design models and construct numerous design concepts in real-time. This framework provides a non-traditional user interface that can take in inputs (e.g. weight, cost etc.) from non-tech users (e.g. consumer), which enables collaborative and interactive design process for people with different disciplines.

The main contributions for this 3D mesh metamorphosis framework can be categorized and summarized as following:

(1) A fast spherical parameterization framework. This framework could benefit various parameterization related applications by performing in a much faster manner than existing spherical parameterization methods.

(2) An innovative feature alignment method based on geometry skeleton generated by applying reported skeleton contraction and extraction algorithm [155]. The method allows users to identify and pick features from input models on an extracted model "skeleton".

(3) A 3D remeshing scheme with spherical mesh subdivision based on our spherical parameterization. This method generates remeshing representations to match the level of details (LOD) for the source meshes and delivers various remeshing outputs with controllable multi-resolution.

(4) A 3D mesh metamorphosis software "3DMeshMorpher" to aid new geometry generation from existing legacy models. This software integrates the spherical parameterization method, skeleton-based feature alignment method and spherical subdivision remeshing algorithm.

1.3 Dissertation Organization

The remainder of this thesis is organized as following: Chapter 2 presents a literature

survey of conceptual design methods and tools. Chapter 3 provides an in-depth review of surface parameterization applications and classified mapping domains and related methods. In Chapter 4, the fast overlapping-solving spherical parameterization approach is introduced. Chapter 5 presents the feature alignment method based on 3D mesh skeleton extraction. And this is followed by the remeshing scheme with spherical subdivision in Chapter 6. Then, the 3D mesh metamorphosis framework is described in Chapter 7, which includes the implementation of the mesh morphing software "3DMeshMorpher" and its user interface design. The results of the spherical parameterization, remeshing with spherical subdivision, and mesh morphing are presented in Chapter 8. Finally, the research presented in this dissertation is discussed and concluded with some future work proposed in Chapter 9.

CHAPTER 2

ADVANCED SYSTEMS DESIGN SUITE

2.1 Literature for Engineering Conceptual Design

As mechanical systems and products continue to be developed and become increasingly more complex, the early stages of a design process become more critical to the success of the resulting product. Given well-defined design requirements, it is challenging to generate and select a concept that effectively satisfies all of the requirements. Conceptual design can have significant impacts on the downstream design and manufacturing process [6]. Early design stages typically include engineers identifying the requirements of a particular project and producing a concept pool using various creative methods such as brainstorming [7]. Engineers produce as many different concepts as possible in order to have a wide variety of ideas to evaluate at the next level of design. Depending on the project, concept generation could produce anywhere from tens to hundreds of possible concepts.

Once the pool of concepts has been established, engineers must reduce the list to a manageable number to proceed to detailed design. Currently, there are limited tools to aid in this process. The most prevalent method is to model concepts using detailed design tools such as CAD software. However, due to the specificity needed to create a solid model, considerable time and resources are consumed producing these 3D concepts simply to assess the rough measure of feasibility needed for evaluation of a concept. Due to the complexity and information needed by detailed design tools, an adequate evaluation of every conceptual

configuration cannot be performed. Such evaluations would be too time consuming and costly to the company. In order to address this problem, some CAD software companies have released “lightened” versions of their products to attempt to release a product less complex and easier to use. Two examples of such products are Pro/CONCEPT [8] and CATIA V6 PLM [9] which can be seen in Figure 1. However, these interfaces are still very complex offering many options and features. Thus, without extensive training and a large learning curve, these lightened applications still do not meet the real-time creation and analysis requirements of digital prototyping at the conceptual design phase.

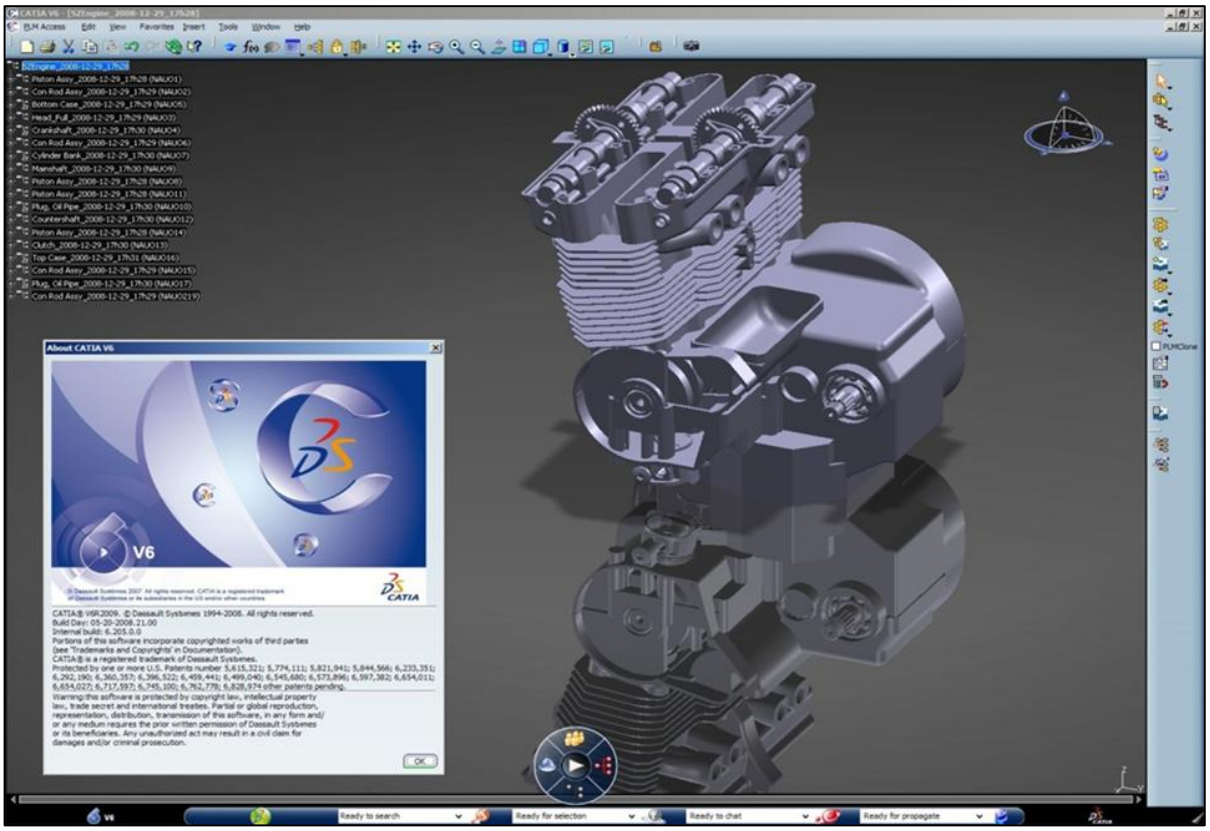


Figure 1: Screenshot of interface for CATIA V6 PLM

To overcome the modeling and reasoning problems, some design related techniques and

methodologies have been developed. Sahin et al. [12] developed a graphical modeling tool to visualize the modeling method to address the challenges of product design decisions. Chang et al. [13] extended this work to support the graphical modeling tool with an ontology-based approach to promote the systematic capture of design knowledge. Cao et al. [14] proposed a port-based ontology to map the concept connections and interactions to compute semantic similarities. Christophe et al. [15] combined the use of Function-Behavior-Structure, System Modeling Language, and artificial intelligence to create a dynamic mapping of ontology layers.

Research has also been done to try to provide more high-fidelity feedback to conceptual designers. Taskahashi et al. [16] integrated a detailed flight control systems synthesis tool into a vehicle configuration development MDO environment to better simulate aerodynamic efficiency, stability, and controllability in air vehicle configurations. Noon and Winer [17] used metamodeling techniques to capture high-fidelity analysis trends from legacy geometry datasets to provide real-time feedback of conceptual design models for large-vehicle designs.

Significant research has also been performed on overall design processes such as axiomatic design [18], decision-based design [19], and specific stages of a design process such as quality function deployment (QFD) [20]. Based on Keeney's Value Focused Thinking [21, 22], Jin et al. took a value-based design (VBD) [23, 24] approach to conceptual design by specifying designer's intent with design variable values. The design value is defined as a group of structured design objectives and a design objective driven approach is proposed to

assist design concept generation. Hoyle and Chen [25] created a design tool called product attribute function deployment (PAFD) which extended the qualitative matrix principles of QFD with utilizing the quantitative decision-making processes of decision-based design (DBD).

Concept selection methods exist to help engineers rank a population of concepts. Examples of these methods are estimating technical difficulty, Pugh concept selection charts, and numerical concept scoring [10]. These methods have proven effective but are simply a ranking system of engineers' opinions on each concept's ability to meet defined criteria of the design proposal. In-depth modeling and analysis (factual hands-on information) does not play a role in these elimination sessions. In order to use these methods more effectively, more information needs to be provided to the engineers before implementing these methods to make concept selections and decisions.

All three conceptual design methods - CAD packages, lightened CAD packages, and concept selection methods - have their advantages and disadvantages. All methods have numerous capabilities but, in today's digital age, still do not define a clear set of tools to be easily integrated into conceptual design. What is truly needed in conceptual design is a means to quickly create and analyze lower fidelity digital models in real-time in order to access every conceptual idea in order to make accurate and informed decisions as early as possible in the design process.

2.2 Development of Advanced Systems Design Suite

Currently, new product concepts are often evaluated by developing detailed virtual part and assembly models with traditional Computer Aided Design (CAD) tools followed by appropriate analyses (e.g., finite element analysis, computational fluid dynamics, etc.). The creation of these models and analyses are tremendously time consuming. If a number of different conceptual configurations have been determined, it may not be possible to model and analyze each of them due to the complexity of these evaluation processes. Thus, promising concepts might be eliminated based solely on insufficient time and resources for assessment. In addition, the virtual models and analyses performed are usually of much higher detail and accuracy than what is needed for such early assessment. By eliminating the time-consuming complexity of a CAD environment and incorporating qualitative assessment tools, engineers could spend more time evaluating concepts that may have been previously abandoned due to time constraints. To address these issues, a software framework, the Advanced Systems Design Suite (ASDS), was created. The ASDS incorporates a PC user interface with an immersive virtual reality (VR) environment to ease the creation and assessment of conceptual design prototypes individually or collaboratively in an immersive VR environment. Assessment tools incorporate metamodeling approximations and immersive visualization to evaluate the validity of each concept. In this paper, the ASDS framework and interface along with specifically designed immersive VR assessment tools such as state saving and dynamic viewpoint creation are presented alongside a test case example of

redesigning an airplane in the conceptual design phase.

2.3.1 Virtual Reality Technology

Virtual reality (VR) development can be traced back as early as the 1960s [26]. Once computing and projection power advanced over the next two decades, academic institutions and industrial centers began investing in VR research and development. Before projection-based systems were feasible, head-mounted displays were a common form of a VR display system. Then, in the early 1990s, VR systems ranging from single-wall projection screens to four-walled CAVE displays [27] were developed across the world and have since become a very popular research area from both a hardware and software perspective. VR hardware has undergone many technological advances since its debut in the early 1990's including projectors, tracking systems, interactive devices, and auditory interfaces.

The first projectors used for VR were capable of producing 1024 x 768 pixel images on 10' x 10' display screens with a pixel resolution of approximately 54 pixels per square inch [28]. High-end projectors on the market today can produce up to 4096 x 2160 pixel images totaling over eight million pixels [29]. When these projectors push images onto a 10' x 5' screen, a pixel resolution of approximately 1228 pixels per square inch is created. This 2260% increase in pixel resolution gives a much clearer and more detailed display of the virtual environment than with the previous generations of VR systems.

More powerful projection systems led to the development of interactive immersive environments. To interact with these immersive environments, tracking systems and

interaction devices were developed. Early tracking systems used electromagnetic fields to perform location tracking [30]. Since these electromagnetic tracking systems could only achieve high accuracy in small environments, companies began researching new technological possibilities for tracking systems. For example, InterSense [31] developed a tracking system combining ultrasound and inertial technology to track multiple devices simultaneously in a large-scale environment with high accuracy called the IS-900.

Nowadays, VR technology is gaining increasing utility for a variety of applications in product development [32]. With real-time interactive graphics, stereoscopic display, and user tracking, VR can be particularly useful for applications in which one-to-one scale is important or when the assessment of complex geometric relationships is required. Haptic interfaces have also been employed for assisting conceptual design [33]. Fischer and Vance [34] also used haptic devices inside a six-sided virtual reality environment for installing an aircraft rudder pedal assembly. Duncan and Vance [35] later developed an immersive virtual reality environment to help engineers better understand complex fluid behaviors in the mixing process. Finally, Abdul-Jalil and Bloebaum [36] created a collaborative virtual environment (VRoom) that allowed designers from multidisciplinary backgrounds to view and manipulate 3D models in an immersive environment simultaneously. With all these technologies available, engineering within immersive virtual reality can provide a collaborative design environment with additional features which cannot be matched with a 2D desktop environment.

2.3.2 ASDS Methodology and Implementation

The software framework is named the Advanced System Design Suite (ASDS) [37, 38]. The ASDS was created to enable an engineer to quickly build a 3D model of a proposed design, assess a concept with real-time simulation analysis, and visualize the results on both desktop and immersive VR systems. The environment enables fast geometry creation by simplifying or eliminating the inputs and interfaces that CAD systems typically require, but are unnecessary at the conceptual design phase.

With the VR-based ASDS system, a group of engineers can create and assess multiple concept ideas in real-time. For example, the process could start by selecting from several base component geometries (e.g., chassis designs). Then, through a unique, intuitive 3D modeling system, features can be added or taken away to produce a new design concept. Typical modifications range from relatively small parameter changes, such as increasing the length of the frame by 10%, to large-scale changes such as adding a third axle to a two-axle vehicle.

Following the model creation some basic properties will be computed such as vehicle weight and center of gravity. Additional output will provide information on other vehicle performance measures including wheel load distribution and static tipping angle. The engineers then have information from which to base further decisions. These decisions might include whether to proceed with this concept to a more detailed analysis or to investigate other conceptual configurations. The 3D model and assessment output will also foster new

ideas to the current concept, ideas that would have been previously overlooked. Multiple iterations of conceptual designs can help design teams develop a specific list of requirements and ultimately a final direction for the team to pursue into the next phase of design.

Figure 2 shows the underlying architecture of the system. User interaction is done through the desktop interface on a tablet or laptop. The desktop application incorporates its own interactive 3D viewing window that controls all of the manipulation—rotation, scaling, panning, and translating—of the model. As shown in the figure, all design changes done on the desktop are transmitted over a network connection and performed in the immersive viewer simultaneously. The immersive viewer uses models from the same data source as the desktop. Navigation in the immersive application is controlled by a gamepad controller. This allows the desktop user to focus on design instead of also having to worry about the changing the immersive application view. By decoupling the immersive navigation from the desktop application, the immersive application environment became much more user friendly and allowed the development of immersive only tools to be developed.

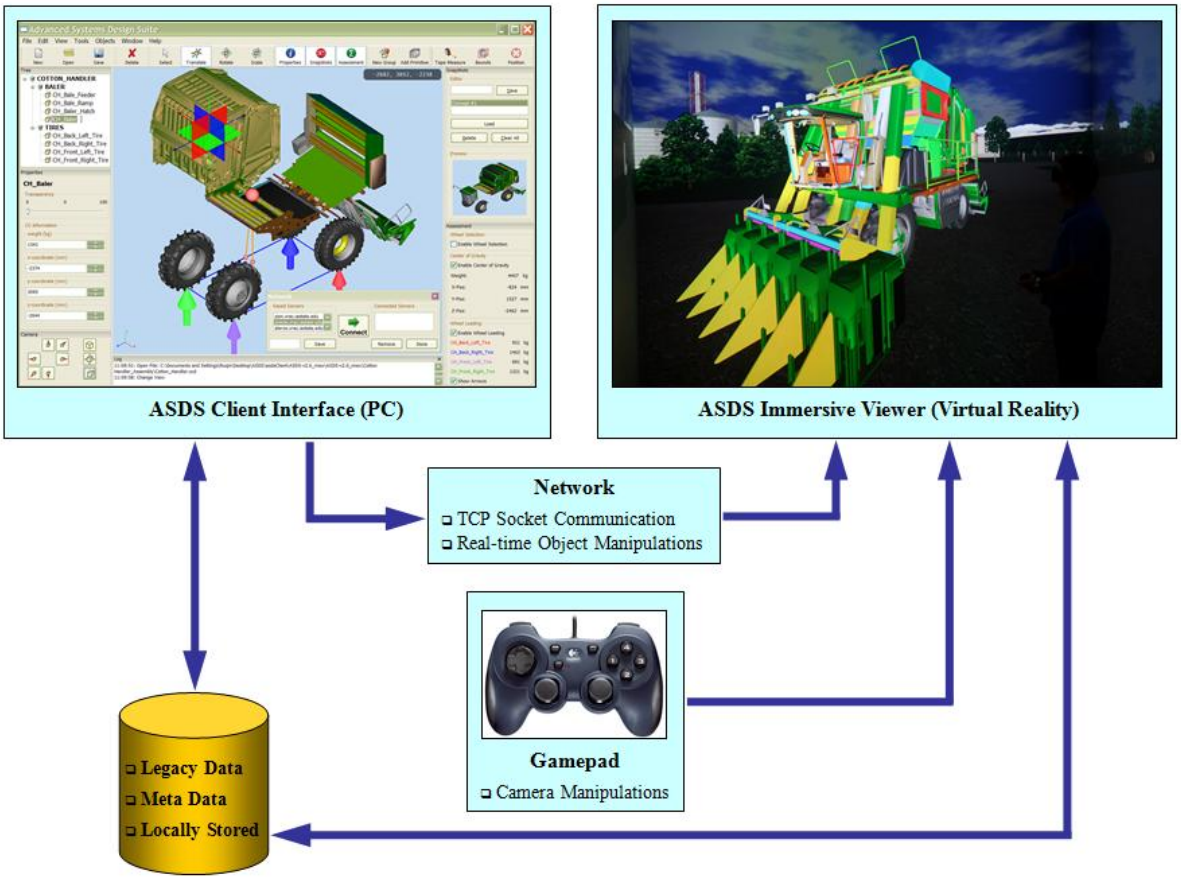


Figure 2: Schematic of the ASDS system architecture

The desktop application as shown in Figure 3 runs under several Operating Systems including Windows XP, Windows Vista, and Windows 7 on both 32 and 64-bit platforms as well as Mac OSX, and the immersive application operates on both 32 and 64-bit Linux Operating Systems. The desktop interface is built upon two open-source packages called OpenSceneGraph (OSG) [39] and wxWidgets [40]. OSG is a scene modeling and manipulation software built on-top of OpenGL to aid developers in scene graph rendering. A sample scene graph tree structure consisting of one group and four sub-groups is shown in Figure 4. The wxWidgets application programming interface (API) is used to develop the

desktop user interface. Since both OSG and wxWidgets are cross-platforms APIs, the ASDS desktop application is able to run on multiple operating systems without having to build separate applications for each Operating System.

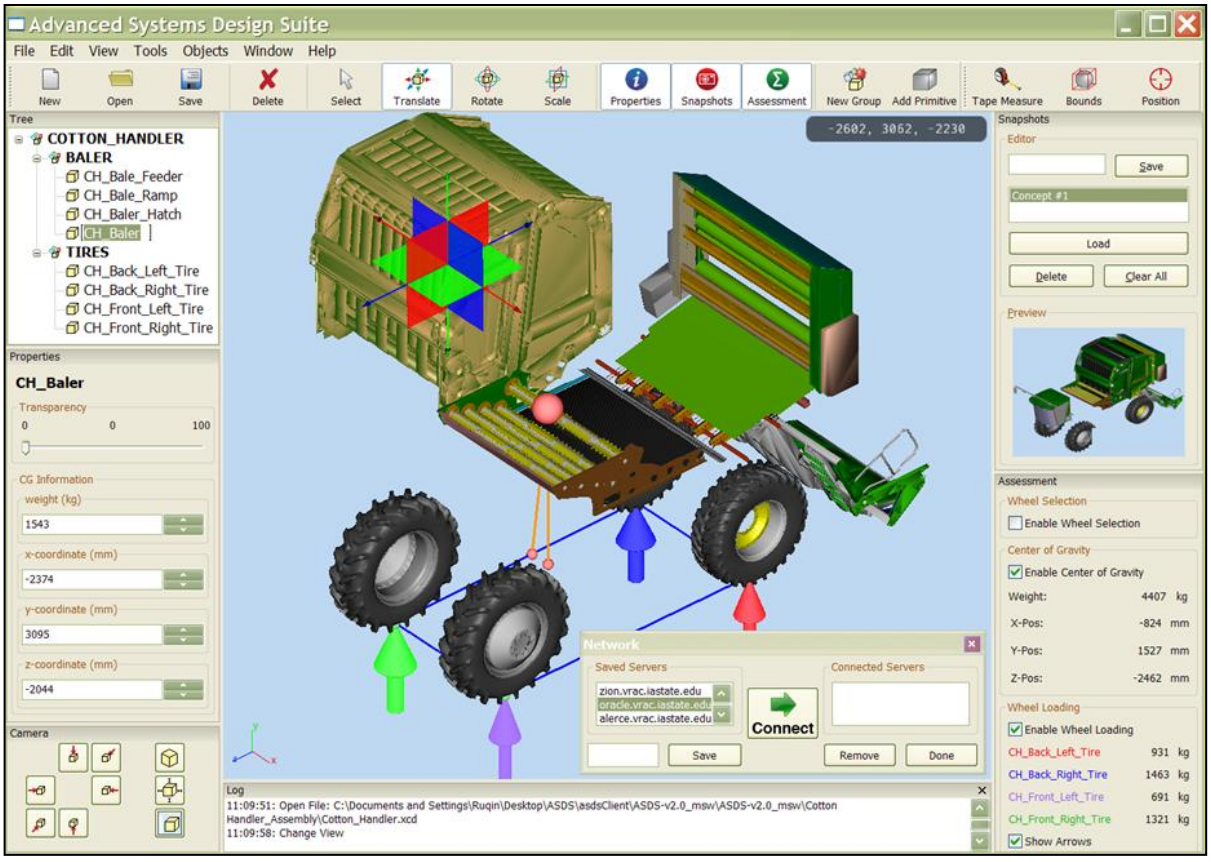


Figure 3: Snapshot of the ASDS desktop user interface

Network communications between the desktop and immersive applications are transported using a Transmission Control Protocol/Internet Protocol (TCP/IP) socket program. TCP/IP was chosen over other types of communication protocols such as User Datagram Protocol UDP due message verification. The TCP/IP protocol ensures the client message is received by the connected server socket before continuing to send additional

messages. This data transmission takes longer than other protocols, but ensuring each message is received and rendered appropriately on the immersive side is a must for these two applications. Message verification also ensures both the desktop and immersive scene graphs stay in sync with each other.

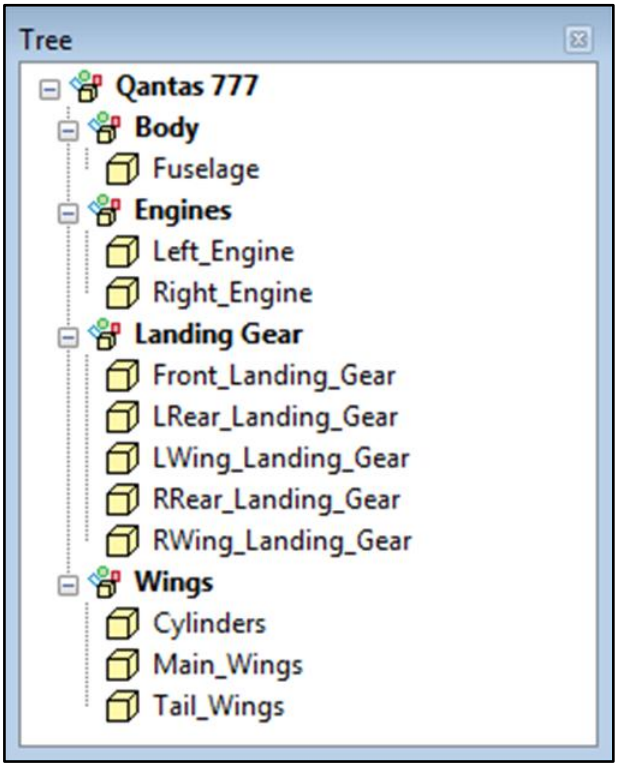


Figure 4: Sample model tree structure of the scene graph

The ASDS immersive visualization software was developed using OSG to handle all the geometric scenegraph rendering while VR Juggler [41] was integrated to abstract display and device interface communication as well as stereoscopic viewing from the development requirements. In order to facilitate the all the different types of network and device communication between all computers on a large-scale cluster, several software

improvements have had to be made. First off, all device and network communications are running in separate threads from the main rendering thread. By processing all the communications in separate threads, the main rendering thread is only required to pick up the processed message when it deems necessary instead of having to process all the communications before being able to render the next frame. The next improvement involved integration VR Juggler cluster synchronization in the main thread once the processed communication was picked up by the main thread. The cluster synchronization is now done through VR Juggler's built in data serialization and synchronization methods. VR Juggler receives and processes the input on a single computer, and then is responsible for passing the message off to the other render nodes on the cluster once it is picked up by the main rendering thread. These cluster-based improvements in the immersive application help ensure a much more stable and reliable collaborative environment with steady frame rates when the immersive application is in use.

CHAPTER 3

SURFACE PARAMETERIZATION

As a tool developed in computational geometry, 3D mesh parameterization is a powerful technique assisting geometric modeling and geometry processing in numerous applications of computer graphics. Sheffer et al. [42] reviewed and addressed practical considerations for various parameterization techniques and their applications. Considering any two surfaces with similar topology, usually there exists a one-to-one (bijective) mapping between them. In general, if one of the surfaces is a triangle-based mesh (either represented or approximated by triangles), the mapping process is referred as mesh parameterization [43; 44]. Typically, the destination surface that the mesh is mapped to is called the parameter domain with the other one named the source mesh. The objective of mesh parameterization is to generate a map between the source mesh and a triangulation of the domain. An essential goal of parameterization is to get a bijective map, where for each vertex in the source mesh there is only one correspondent vertex in the target parameterization domain. Usually, such parameterizations are piecewise linear, associating each triangle of the original source mesh with exactly one triangle in the parameterization domain.

Mesh parameterization was originally employed in computer graphics as a method for texture mapping for 3D surfaces [43; 45], which was the main driving force used in computer graphics to enhance the visual quality of 3D surface models. Later, it became a necessity due to the development of 3D scanning technologies and the resulting demand for efficient

compression techniques for increasingly complex triangulations. It is also influenced by other applications like surface approximation, and remeshing. Other fields that benefit from parameterization include detail mapping, detail synthesis, detail transfer, mesh fixing, mesh editing, object database creation, mesh compression, surface fitting, modeling from material sheets, medical visualization, filtering, texture mapping, remeshing, and morphing.

Aside from the topological similarity, there is typically quite a large geometric difference between the source mesh and the parameterization domain, which almost always introduces distortions existing in either angles or areas. Very few cases admit isometric parameterization (zero distortion). The goal of a good parameterization algorithm is to minimize these distortions for the entire mesh. Based on the type of distortion minimization methodology, most parameterizations can be classified as two groups of mappings. Maps that attempt to minimize the angle distortions are named “conformal” and maps that minimize the area distortion are referred as “authalic”. Research has been performed to measure the conformality of a mesh in several ways [46, 47, 48, 49] by applying different functions to be optimized. For example, Hormann and Greiner [46] consider the minimal and maximal eigenvalues of the first fundamental form of the mapping. Sheffer and de Sturler [47] directly calculate the difference between the corresponding angles in the source mesh and output parameterized mesh. Floater and Hormann [44] report that authalic parameterizations are not very useful in practice as they allow extreme angular and linear distortion. Due to that, researchers that attempt to preserve area [49, 50] also consider angular minimization for

balance.

The traditional surface parameterization problem derived from texture mapping considers the case where the target domain is a planar region [44]. The mapping from the source mesh to parameterization domain is represented by the parametric locations of vertices within the plane. Various optimizations are applied to freely relocate the vertices within the domain as long as the mesh is maintained bijectively. However, with increasingly different applications of computer graphics and geometric modeling, the 2D planar parameterization domain does not meet this requirement in many emerging applications.. High dimensional parameterization domains are actively being researched and many different algorithms and methodologies have been developed. Based on the type of mapping domain, mesh parameterization can be categorized as planar parameterization, spherical parameterization, simplicial parameterization and inter-surface mapping.

In this section, a literature survey will first be presented about the applications that are closely related to and benefit from mesh parameterization. Then different parameterization methodologies and techniques are discussed, classified by parameter domain used: planar, spherical, simplicial and inter-surface parameterization. Lastly, some methods of introducing constraints into a parameterization is also reviewed and discussed.

3.1 Applications for Parameterization

As mentioned above, 3D mesh parameterization was originally investigated and

introduced into the field of computer graphics and computational geometry as a technique for texture mapping. With the fast growth in the field of mesh and geometry processing, more and more related research integrated and benefited from mesh parameterization.

3.1.1 Texture and Detail Mapping

Over the last decade, texture mapping has been recognized as one of the most successful techniques for high quality image synthesis in computer graphics. Although application variety is diverse, the techniques of texture mapping are basically the same for different cases. Here we can categorize surface attributes as color, surface normal, specularity, transparency, illumination, and surface displacement. Early synthesized raster images of surfaces emphasized the smoothness of surfaces, without attempting to represent fine details like scratches or dirt, and generally lacked realistic effects and complexity. This, texture mapping was developed as a relatively efficient means to create complicated and realistic surface renderings.

The main goal of texture mapping is to realistically represent the complex appearance of 3D geometry. In the early years, research in texture mapping was more focused on parameters like color and surface normal. Some of the parameters that have been texture mapped include, surface color (the most common use) [51], specular reflection [52], normal vector perturbation (bump mapping) [53], specularity (the glossiness coefficient) [54], transparency [55], diffuse reflection [56], shadows, surface displacement, and mixing coefficients [57], and local coordinate system (frame mapping) [58].

For computer graphics display, surfaces are represented by a tessellated geometric model (typically a triangular mesh, polygonal mesh or subdivision surface) with texture and other information stored separately. For rendering 3D digital geometry, textures can enhance visual appearance with simple static pictures. However, with a continuous and smooth texture, the rendering for pixel neighborhoods is very similar, and under varying lighting conditions the object can look flat and unrealistic in animations. Bump mapping [53] addresses this issue by storing small deviations of the point-wise surface normal and applies the perturbed version during the shading process. Sheffer et al. [59] develops a similar method called normal mapping which replaces the normals instead of storing perturbations. The shading variations generate small pits and dimples in the surface to simulate the shadows when there is a direction change of the light source. Bump mapping and normal mapping address the issue of shading variation. But since the geometry of the model is never deformed or edited, the appearance of the object always looks either smooth or polygonal. This problem is handled by displacement mapping with small local deformations of the surface record stored in the texture. Recentl techniques [60, 61, 62] employ so called volumetric textures, instead of conventional 2D textures, to model a thick region of space in the neighborhood of the surface. These techniques deal with the situations of complicated topology or details which are not easy to estimate with a local height field.

Unlike texture mapping that attempts to represent the complex appearance of 3D objects, newly developed detail synthesis techniques are designed to create rich local details by

applying surface parameterization. One such procedure implements flat sample patches with detailed textures. Pederson et al. [63] present a set of interactive tools for subdividing an implicit surface into convenient patches with an efficient and reliable algorithm for deriving parameterizations for such patches. Lapped texture [64] is a means for creating texture over an arbitrary surface mesh using an example 2D texture by identifying interesting regions (texture patches) in the 2D example and repeatedly pasting them onto the surface until it is completely covered. Wei and Levoy [65] provide a solution to synthesize a general texture over arbitrary manifold surfaces by extending their original texture synthesis method with a generalization of their definition of search neighborhoods. They realize this by establishing a local parameterization surrounding each mesh vertex and using the parameterization to create a small rectangular neighborhood with the vertex at its center, and then search a sample texture for similar neighborhoods. Turk et al. [66] believe that the best way to create a surface pattern is to synthesize a texture directly on the surface of the model. Using texture synthesis methods that use image pyramids (and a mesh hierarchy to serve in place of such pyramids), they create a similar texture over an irregular mesh hierarchy on a give surface with a texture sample in the form of an image. By considering a very general type of texture that including color, transparency and displacement, Ying et al. [67] present a novel method for texture synthesis on a surface that synthesizes the texture directly on the surface rather than synthesizing a texture image and then mapping it to the surface. Finally, the multi-scale algorithm [68] maps a texture defined by an input image onto an arbitrary surface. This

method progressively covers the surface by texture patches of different sizes and shapes from a single input image.

An alternative approach for detail synthesis is to directly process user input and editing (e.g., painting). Igarashi et al. [69] implement a method to dynamically generate an efficient texture bitmap and its associated UV-mapping in an interactive texture painting system for 3D models. To eliminate the distortion of brush strokes, they develop an adaptive unwrapping mechanism where the system dynamically generates a tailored UV-mapping for newly painted polygons during the interactive painting process. The final texture bitmap resulting from this process is more compact since texture space is only allocated for the painted polygons. Carr and Hart [70] present a GPU based texture atlas algorithm which distributes initial texture samples evenly according to the surface area and texture frequency, and maintain the distribution as the texture signal changes during the surface painting process. They make the redistribution of samples transparent to the user which results in a surface painting system of theoretically unlimited resolution.

3.1.2 Mesh Manipulations

Mesh manipulation refers to those applications that involve topology or connectivity changes of the source mesh, such as moving vertex position, adding more vertices, removing vertices, and switching from a triangular to polygonal mesh. Many different mesh parameterization and manipulation operations are described in this section including: mesh fixing, mesh editing, and remeshing.

Many 3D complex geometric models are generated from high quality 3D scanning. During such a process, the resulting models are usually not perfect due to holes and multiple components. Mesh fixing is the process of repairing such imperfections to produce a complete model that contains the original scanned model. Lévy [71] provides an approach to compute a natural boundary to triangulate around holes with global planar parameterization. This work makes it feasible to extrapolate the geometry beyond the existing boundary rather than just smooth an existing geometric model. For some scanned models, there may be prior knowledge about their overall appearance and such knowledge can facilitate the mesh fixing process. Allen et al. [72] develop a novel technique to fit high-resolution template meshes to detailed human body range scans with sparse 3D markers. To achieve this, they formulate an optimization problem such that the degrees of freedom are affine transformations for each template vertex and solve the problem with a non-linear optimizer running at two resolutions to assist convergence. Also with human shape, Anguelov et al. [73] introduce a data-driven method (SCAPE, Shape Completion and Animation for PEope) to build a human shape model. The method involves variations in both human body shape and pose, which is set up with a representation that incorporates articulated and non-rigid deformations.

Kraevoy et al. [74] present a more generic and robust template-based approach for mesh completion for arbitrary 3D scans. This approach employs a mapping between the incomplete mesh and a template model, which is calculated with a novel framework for bijective parameterization of meshes with gaps and holes. This mapping can correctly glue together

the components of the input mesh and to close the holes, as well as fill the topological and geometric information missing in the input.

Local/Global mesh parameterization techniques are also applied to facilitate mesh editing related operations in many applications. Based on local parameterization, Biermann et al. [75] generate a set of algorithms for multiresolution subdivision surfaces which perform at interactive rates and enable intuitive cut-and-paste operations. The local parameterizations for areas of interest for two different models are placed overlapping each other and applied to transfer local shape properties and details from one model to another. Sorkine et al. [76] treat geometric detail as an intrinsic property of a surface and point out that surface editing is best performed by operating over an intrinsic surface representation. They provide a Laplacian representation for the mesh, which is enhanced to be invariant to locally linearized transformation and scaling. From this representation, mesh editing operations are developed, including interactive free form deformation (FFD) in a local region, geometric detail transfer and mixing between surface meshes and transplantation of a partial mesh onto another mesh. Lévy [71] use a similar local parameterization for mesh composition by calculating an overlapping planar parameterization of the region near the composition boundary on the source models and utilize it to extract and blend shape information from the source models smoothly.

Remeshing is another important mesh operation that is dependent on mesh parameterization techniques. Resampling raw surface meshes has become one of the most

fundamental operations used by nearly all digital geometry processing systems. With a similar level of detail, there are a lot of different triangular representations for a selected shape. Usually, the selection of such mesh representation is determined by the requirements of their application. The most straightforward method of replacing one triangulation by another is to parameterize the source mesh into a domain, map a desired well-defined mesh into the same domain, and finally, map the desired mesh back to the source mesh based on the overlapping information from the domain. Gu et al. [77] present a novel remeshing technique called “geometry image” which captures geometry as a 2D array of quantized points and stores normals and colors as surface signals in a similar 2D array by applying the same implicit surface parameterization. Cutting the mesh and mapping the resulting chart onto a square creates the geometry image. A promising perspective of this work is that the geometry image can be encoded with image compression algorithms (e.g. wavelet-based encoders). Inspired by differential geometry, Guskov et al. [78] introduce a new fundamental surface description called the “normal mesh”, which is a mesh with multiresolution representation. Each level of resolution is written as a normal offset from a coarser version of the mesh and is stored with a single float per vertex. They also provide an algorithm to estimate any arbitrary surface closely with a normal semi-regular mesh. Lee et al. [79] introduce another surface representation referred to as the “displaced subdivision surface” to represent a detailed surface geometry as a scalar-valued displacement over a smooth surface domain. Both the domain surface and the displacement function are defined with a unified

subdivision framework which provides a way to evaluate the analytic surface properties simply and efficiently. Khodakovsky et al. [80] present a parameterization framework which takes in surface meshes with arbitrary topology and generates a globally smooth parameterization with small distortion. They demonstrate the performance of this algorithm with numerical evaluation of distortion measurement and distortion performance of semi-regular remeshes produced. Desbrun et al. [49] present a new theoretical and practical parameterization of triangulated surface patches called “intrinsic parameterizations,” which minimize the distortion of different intrinsic measures of the original mesh within a simple, sparse linear system. With planar Delaunay triangulation, they are able to generate high quality remeshing of the surface and propose it to faciliate the rapid design of parameterizations.

Instead of global parameterization, Surazhky and Gotsman [81] introduce a robust local parameterization based remeshing scheme that enhances mesh quality with many local modifications of geometry and connectivity. To achieve this, they describe a family of local modification techniques with an area-based smoothing method that allows the control of both the triangle quality and vertex sampling over the mesh. Dynamic patch-wise parameterization is performed to local modifications of meshes with arbitrary genus and a novel algorithm is implemented to improve the regularity of the mesh connectivity by creating an unstructured mesh with a very small number of irregular vertices. Ray et al. [82] present a globally smooth parameterization technique for triangulated surfaces with arbitrary topology. Their method

calculates two piecewise linear periodic functions by minimizing an objective function, which can construct both quasi-conformal (angle preserving) and quasi-isometric (angle and area preserving) parameterization. This work claims to be particularly suitable for surface fitting and remeshing due to the alignment of parameterization with the principle curvature directions.

Unlike most work that has focused on triangular remeshing, Dong et al. [83] focus on quadrilateral meshes that are more suitable for many surface PDE problems. In their work, they describe an innovative algorithm to quadrangulate manifold polygonal meshes by applying Laplacian eigenfunctions. With algorithms and heuristics to efficiently and effectively choose the harmonic most suitable for the intended application, they are able to build a well-shaped quadrilateral mesh with very few extraordinary vertices.

3.1.3 Other Applications

Data compression is the process of encoding information by using fewer bits (or other information storage units) than an unencoded representation would within specific encoding schemes. With the growth of the computational geometry field, researchers have borrowed ideas from data compression to develop “mesh compression” which is utilized to compactly store information for geometric models [84]. The efficiency of a compression method is usually evaluated by compression rate. To facilitate higher compression rate for geometric models, the mesh should have either all the vertices of the same degree, or in other words, the triangles should be similar to each other in terms of shape and size with vertices around the

geometric centroid of their neighbors,.

Mesh compression algorithms can be classified by two major techniques depending on whether the model is decoded during or only after the transmission: single-rate and progressive compressions. For single-rate compression, the objective is to delete the redundancy present in the original description of the data. And for progressive, the aim is to get the best rate-distortion tradeoff, which is the tradeoff between data size and approximation accuracy. A regular mesh is generated first to facilitate high-rate compression. The source mesh is usually parameterized to a domain and then remeshed with regular sampling patterns [77]. Similarly, Hoppe and Praun [85] develop an mesh compression scheme with the introduction of their spherical parameterization and remeshing algorithm which maps a genus-0 surface mesh onto a 2D grid, a spherical geometry image. Their compression and decompression algorithms work on 2D arrays and are claimed to be ideally suite for hardware acceleration. Their two approaches for shape compression are wavelet-based and use spherical geometry images.

Within a common domain, parameterizations of a large amount of meshes can facilitate the creation of object databases. Analyses based on such parameterizations can be performed to determine the common characteristics among objects and their distinguishing traits. Allen et al. [72] present a novel technique to fit high-resolution template meshes to detailed human body range scans with sparse 3D markers. In their work, they set up a database of human shapes with the possible distinguishing traits of gender, height, weight, etc. Blanz and Vetter

[86] develop a new technique for modeling textured 3D faces based on a face database with facial expressions. With their face database, transforming the shape and texture of the examples into a vector space representation is used to derive a morphable face model. Based on a linear combination of the prototype models, new faces or expressions can be generated with a morphing interface. Marschner et al. [87] build a face model system for modeling, animation and rendering the human face using measured information for geometry, as well as motion and reflectance that regenerates a particular human's facial appearance and facial expressions. This system creates structured face model database with correspondences across different faces, providing a foundation for various facial animation operations. Blanz et al. [88] introduce an image based animation technique for human faces, which necessitates no exemplar data of mouth movements and no restricted poses or illumination requirements. Their system transforms mouth movements and expressions among examples based upon a common representation of various faces and face expressions in a vector space of 3D shapes and textures, which is calculated from 3D scans of neutral faces and face expressions. Using the same database of human faces, Blanz et al. [89] present a system that can substitute faces with big differences in viewpoint and illumination, unlike the traditional photo retouching and image processing tools with fixed viewpoint and illumination. To achieve this, they implement an algorithm to estimate 3D shape and texture along with all relevant parameters, and a user interface for clicking a set of feature points and marking the hairline in the target image. This technique is claimed to be helpful for image processing, virtual hairstyle try-on

and face recognition.

Mesh parameterization methods are also applied in medical visualization (e.g., volume rendering). Instead of mapping vertex positions and mesh connectivity, usually features such as surface normal-map, color and other properties are parameterized to a simpler, canonical domain for visualization and further analysis. In particular, such mapping techniques are useful for studying human brain. Hurdal et al. [90] provide a new approach to create parameterization for flattening maps of the human brain. Based on Riemann Mapping Theorem, their algorithm performs a conformal parameterization for angular preservation. This parameterization can deal with a mapping domain of not only traditional Euclidean plane but also the hyperbolic plane and the sphere, without cuts to be introduced in the source surface. Haker et al. [91] treat the brain as a genus-zero surface and visualize it through spherical parameterization with topologically equivalence. To do that, they introduce an explicit method to map any simply connected surface on a sphere which relies on some conformal mapping from differential geometry. A finite element method is also merged into their work for a triangulated surface description. Based on the structure of the co-homology group of holomorphic one-forms for surfaces, Gu et al. [92] develop a general method of global conformal mapping for genus-zero meshes. They apply such method in parameterizing the human brain. By using a mesh structure to represent magnetic resonance imaging (MRI) data, their algorithm is robust in handling such conformal parameterization stably and has good extensibility.

Most of the techniques developed for computer graphics and computational geometry are focused only on digital geometric models and rarely consider real-world engineering applications. Some work, however, has dealt with such issues by applying 2D planar mesh parameterization as a tool to model 3D objects from sheets of material. The relevant applications range from garment modeling to metal forming or forging. Mitani and Suzuki [93] propose a novel method of making paper craft toys with triangulated meshes by means of a strip-based approximation. The approach approximates the model mesh with a set of triangular strips so that the unfold pattern can be generated using only mesh operations and a simple unfolding algorithm. The crafted model maintains smooth features of the original model meshes by bending the paper without breaking edges. Julius et al. [94] introduce a new quasi-developable mesh segmentation framework “D-Charts” based on a new metric of developability for surface meshes and a technique for automatic pattern design. They practically apply this method in making fabric and paper copies of some popular computer graphics models.

3.2 Parameterization Domains and Methods

Traditional surface mesh parameterization techniques for computer graphics applications are focused primarily on mapping meshes with disk-like topology to a planar region. As such planar parameterization methods are only applicable to surfaces with disk topology and cannot be directly applied to closed surfaces. Since most practical 3D surface meshes are

closed surfaces or contain closed surface features, more and more research focuses on these parameterization problems. Technically speaking, the challenge is to solve the topological inequivalence between source mesh and target domain. Techniques such as cutting and chart generation aim to change the source mesh to match the topology for the target domain. Alternatively, spherical or simplicial parameterization aims to switch the planar open domain to closed 3D surface domain, which is equivalent to the source mesh topologically. In this section, parameterization techniques based on different target mapping domains are reviewed and discussed.

3.2.1 Planar parameterization

Early planar parameterization aimed to address the issue of texture mapping for surfaces with disk-like topology. With the development of computer graphics, recent applications involve parameterization in other surface properties (e.g., normals) and geometry processing operations (e.g., remeshing, and mesh fixing). Mostly, parameterization from 3D surface mesh to 2D planar domain unavoidably produces distortions except some rare cases. Based on a well-known theorem [95] from differential geometry, an isometric parameterization which preserves distances does not exist for planar parameterization. Many parameterization methods work with distortion minimization in either angular, stretch or area. Besides distortion, there are other important considerations for the planar parameterization (also applicable to spherical and simplicial cases): i) boundary conditions - either free or fixed boundary; ii) validity and robustness - bijective mapping globally or locally; iii) efficiency -

practical numerical complexity, linear or non-linear system solution.

The uniform parameterization from Tutte's [96] graph embedding method is recognized as one of the earliest methods in mesh parameterization. In this method, vertices on the boundary of 3D meshes are mapped onto the boundary of a 2D planar domain. The boundary for this mapping domain needs to be a convex region. Instead of defining a uniform weight for each edge of the mesh in Tutte's method, Floater [97] calculates the weight for each edge based on the information from their neighborhood. Such non-uniform weight has proven to be shape-preserving and lead to visually smooth surface approximation. For the validity and robustness for this planar parameterization method, usually if the weights are positive and symmetric, and the boundary is convex, the parameterization obtained can be guaranteed to be bijective.

Angle preserving parameterization is one of the most investigated methods in the field of mesh parameterization. Angle preservation is required by some graphics applications such as remeshing and some engineering applications like numerical simulations. In these applications, angular distortions especially small angles will either affect the numerical results (i.e., generate numerical singularity) or visual quality (e.g., produce unsmooth appearance). Eck et al. [98] introduce harmonic, or cotangent weights, parameterization. These weights are generated from a finite element method based representation for harmonic energy and lead to minimizing angular distortion. To ensure bijectivity, Kharevych et al. [99] introduce intrinsic Delaunay triangulation of the surface mesh as an input to harmonic

parameterization. This method constructs discrete conformal mappings based on circle patterns. It supports different boundary conditions ranging from natural boundaries to controlled boundary shape. The anisotropic mesh parameterization scheme [100] brings in an anisotropic modification to Floater's shape preserving parameterization method [97]. The implementation introduces an additional stretching term to the original discrete energy minimization scheme, which allows flattening of parametric mapping along a given discretionary field. Similar to the harmonic weights, weights can also be generated from mean-value coordinates [101]. This is a generalization of barycentric coordinates to allow a vertex to be represented by a convex combination of its neighbor vertices, which is based on the mean value theorem for harmonic functions. Although the resulting weight matrix is non-symmetric, it has been proven that mean-value parameterization is guaranteed to be bijective.

These mapping techniques are weight-based and can be implemented by solving a linear system. Theoretically, parameterization distortion depends on the difference between the actual boundary shape of the source mesh and the boundary shape for the 2D domain. Lee et al. [102] create a fixed virtual boundary to make the real domain boundary free and thus the real boundary can better reflect the shape of the source 3D boundary. With such freely moving boundary, the parameterization is able to introduce less distortion than methods with fixed boundaries. Similarly, Zhang et al. [103] present an automatic planar parameterization method for mesh segmentation and flattening. This feature-based parameterization method performs patch creation with genus reduction and feature identification, and applies scaffold

triangles in the virtual boundary for minimizing distortion. LSCM (Least Squares Conformal Maps) [48] and DCP (Discrete Conformal Parameterization) [49] are two explicit formulations for linear parameterization with free boundary. They both aim to minimize angular distortions, but are independently proposed with different formulations of harmonic energy.

Unlike most planar parameterization methods mathematically defined with vertex positions, the ABF and ABF++ (Angle-Based Flattening) [104, 59] introduce a novel method with a definition in term of angles. The algorithm runs iterations to search for angles that are as close as possible to the angles in the original 3D source mesh. These angles are converted to coordinates for all vertices after this minimization process. Zayer et al. [105, 106] extend ABF with additional methods borrowed from traditional parameterization process in terms of vertex coordinates. In their work, either convex boundary condition is forced to the parameterization domain to guarantee global bijectivity, or an iterative free-boundary conformal method is applied to minimize distortions.

Distance preserving parameterization is another category for 3D mesh parameterization. Since only some developable surfaces can be parameterized with distance preservation, existing methods aim to minimize such distortions instead of eliminate them. Lévy and Mallet [107] introduce a technique for non-distorted texture mapping on complex triangular meshes by using an iterative optimization. Unlike other global optimization techniques, they allow local distortions minimization in order of preference from user’s input. While, it is

reported that minimization formulations for the distance are numerically complex and hard to solve. Sander et al. [108] present a technique for constructing a progressive mesh to make all mesh sequences share a common texture parameterization. They introduce two metrics of parameterization stretch, which are widely used for linear distortion comparison between different mapping methods. Iso-charts [109] merge two apparently incompatible techniques together to create texture atlas for arbitrary meshes. These two techniques include stretch-minimizing parameterization from the surface integral of the trace of the local metric tensor and the multi-dimensional scaling (MDS) parameterization from an eigen-analysis of the matrix of squared geodesic distance between two vertices. In a later work, Sander et al. [110] extend the method with signal specialized parameterization, which allows the user to affect the distribution of distortions along the mesh surface. Tewari et al. [111] report a more accurate signal with significant savings in texture area than the signal specialized parameterization method by Sander et al. [110]. They make use of a metric for the surface parameterization specialized signal to generate a more efficient high-quality texture mapping.

Area preserving parameterization, referred as authalic, deals with area preservation for mesh triangles by typically introducing additional optimization terms or constraints. Desbrun et al. [49] derive a similar method from their discrete conformal map (DCP) algorithm and implement a linear formulation for local triangular area preservation. Their formulation supports a tradeoff between angular and area distortions. Degener et al. [50] directly target global area deformation for mesh parameterization. A non-linear formulation is developed

with an energy functional which measures angular and area distortions simultaneously with a tradeoff parameter controlled by user. The method does not require a fixed boundary condition and the non-linear energy implementation could assist in preventing triangle flips. A hierarchical optimization framework based method is employed to minimize the energy and guarantee the convergence of the algorithm.

2D planar parameterization techniques are only applicable to 3D meshes with the same topology. For closed meshes or high genus (greater than 0) meshes, theoretically it is impossible to directly map them onto a 2D domain. Thus, techniques have been developed to cut the meshes before parameterization. The cuttings will decrease the parameterization distortions while increasing cross-cut discontinuities. The tradeoff between these distortions and discontinuities must be considered and balanced during parameterization. The first category of mesh cutting techniques is to cut the surface to an atlas of charts. Maillot et al. [45] introduce an algorithm to automatically produce an atlas from any type of mesh for texture mapping. The distortion is lowered by a general optimization function with an energy minimization process. Multi-chart geometry images [112] refer to a representation for arbitrary geometric surfaces to map the surface piecewise onto charts of arbitrary shape by using an atlas construction. They create a watertight surface with the implementation of a novel zippering algorithm to eliminate unacceptable surface cracks for shapes with long extremities, high genus or disconnected components. Gu and Yau [113] present a global parameterization algorithm that preserves conformality everywhere (except for a few points)

and introduces no boundary discontinuities by constructing a basis of the underlying linear solution space. It is claimed that the mapping result is independent of connectivity and insensitive to resolution. Tarini et al. [114] introduce a technique called “PolyCube-Maps” which maps 3D meshes on a set of square charts. The texture is stored as a collection of small image pieces on these square charts. Vertices from the source mesh are parameterized onto the base domain formed by a collection of assembled cubes. Each segment of the 3D surface mesh is projected onto a nearby cube face, and pixel information is read and assigned to every associated vertex based on the texture for the face.

Instead of segmenting the mesh into multiple separate patches, another widely used technique of mesh cutting is to cut the mesh to a single chart. Compared to mesh segmentation, this typically leads to shorter cutting paths and yet still reduces parameterization distortion. Sheffer et al. [115] show that areas of high surface curvature yield more distortion during parameterization and cutting the surface in these areas can reduce distortion. They introduce a fast technique to lead a texture map seam through such high curvature areas and restrict the seam to regions with low visibility. Their results indicate less distortion and are less visually distracting. Sorkine et al. [116] provide the first method to parameterize and partition the mesh simultaneously and automatically. With strictly bounded distortion, their method generates low distortion and guarantees avoiding global and local self-intersections by minimizing the total length of the cutting seams. Lazarus et al. [117] present two optimal algorithms for the problem related to cutting the surface with high genus

and map it into a topological disk from canonical polygonal schema. A handle cutting method by Erickson et al. [118] aims to converting a polyhedral manifold surface into a single topological disk by minimizing either the total number of cutting edges or total cutting length. This method is reported to be complicated to implement. Ni et al. [119] perform small number of cuttings for genus reduction by solving a relaxed form of Laplace's equation to find a fair Morse function [120] with a user-controlled number and configuration of critical points.

3.2.2 Simplicial Parameterization

Planar parameterization is widely applied to map meshes with disk topology onto a 2D planar domain. For surfaces with different topology (e.g., high genus or closed surface), mesh segmentation or seam cutting will be performed before or during parameterization. Most of these techniques are developed to minimize distortions of either angle or area. With mesh segmentation and seam cutting, there will always be inevitable discontinuities generated from parameterization. However, some graphic applications are very sensitive to, or even cannot tolerate such discontinuities. In these cases, researchers try to employ non-planar base domains to avoid unwanted segmentation or cutting. In this section, parameterization base on the domain of a simplicial complex is reviewed.

It is reported that a simplicial complex has been the most popular non-planar domain for parameterization [123]. Usually, the simplicial parameterization process consists of two steps. The first one is to define a coarse simpilicial complex. One method applied for this is

to simplify the original source mesh. Once this domain complex is generated, each vertex from the original mesh is parameterized to the base complex domain by computing its barycentric coordinates. The challenge in parameterization with a simplicial complex domain is the difficulty for global parameterization optimization. Due to the sharp edges and vertices existing in the simplicial domain, most algorithms employ linear relaxation of local neighborhood instead of global parameterization. Such parameterization processes usually produce distortions.

Eck et al. [98] introduce a method to convert completely arbitrary meshes to multiresolution form by overcoming the subdivision connectivity restriction. They claim that the essential component of this algorithm is the construction of a parameterization over a simplicial complex domain. They perform a local iterative relaxation on a pair of adjacent faces and parameterize the surface neighborhood over the resulting quadrilateral. Hybrid mesh [121] is another multi-resolution surface representation with both regular and irregular refinements. The regular operations are enabled by an efficient tree based data structure and processing algorithms but lack of flexibility in resolving shapes with high genus or features at many scales. However the irregular operations can adjust mesh topology throughout the hierarchy and estimate detailed features at multiple scales. Consistent mesh parameterizations introduced by Praun et al. [122] provide an algorithm that establishes consistent parameterizations for a group of models by sharing the same base domain and respective features. They implement remeshing based on the same connectivity, which forms

a wide range of application algorithms including principal component analysis (PCA), wavelet transforms, as well as detail and texture transfer between models.

Rather than iteratively optimizing local neighborhoods as in most simplicial parameterization methods, Khodakovsky et al. [80] set up a global system in which the adjacent domain faces are regarded as they are locally opened up into a plane. This global system converges in a fast manner. Schreiner et al. [123] extend some of their previous work to construct the simplicial complex domain automatically in parallel to the patch formation. They make use of a set of correspondences between feature vertices from the input meshes as the vertices to form the base domain. In a similar way, cross-parameterization [124] also preserves the feature vertex correspondences from user input and the shape correlation between the models. It seems this remeshing algorithm can generate an output mesh with fewer elements but still approximate the input geometry accurately. Boier-Martin et al. [125] develop a method for parameterizing irregular triangular meshes over polyhedral domains with quadrilateral faces. They construct a coarse mesh with normal-based clustering of faces and spatial-based clustering of the initial generated charts. The coarse polygonal mesh is defined by region boundaries, which is cleaned up and quadrangulated to generate the base domain over which the input mesh is parameterized.

3.2.3 Spherical Parameterization

As we discussed above, planar parameterization for high genus meshes usually introduce mesh segmentation or seam cutting, which can generate discontinuities and distortions. For

the simplicial complex domain, it is hard to optimize the parameterization globally in most cases. Topologically, closed manifold, genus-zero meshes, are equivalent to a sphere. So a spherical base surface is the natural parameterization domain for these meshes. Compared to the planar and simplicial domains, the advantage of spherical parameterization is that it allows smooth, seamless and continuous parameterization of genus zero models. Thus, much research attention has been devoted to the spherical domain in the past few years.

One approach [126] is to reduce the 3D spherical parameterization to the 2D planar case. First, closed mesh is cut into two pieces with topological equivalence to a disk. Then, each of these two pieces is parameterized on a 2D planar domain with the same fixed boundary. Finally, each disk is mapped to a hemisphere and the two are combined into a full sphere. Instead of cutting the whole mesh to get the boundary, another approach [91] picks one triangle as a boundary and computes a planar parameterization of the remaining open mesh over the triangle by applying a planar parameterization method, and finally calculate the stereographic projection to obtain the spherical parameterization. However, some report that this method may introduce severe distortion and does not guarantee a valid spherical triangulation. Zayer et al. [127] cut the mesh along a date line defined by some poles from user input and apply planar parameterization over a rectangular domain by solving a Laplace equation in curvilinear coordinates. They reduce the mapping distortions by using a variant of quasi-harmonic maps and performing tangential Laplacian smoothing.

Directly parameterizing a mesh over a spherical domain is more natural than applying a

2D plane as a temporary medium domain between the two. Some researchers have applied the mesh simplification and multi-resolution methods to facilitate such spherical parameterization processes. Similar to the work by Das and Goodrich [128], Shapiro and Tal [129] apply mesh simplification for vertex removal until a tetrahedron remains. They map the tetrahedron onto the closed domain and then add removed vertices back, one-by-one. The interpolation of the corresponding vertices is based on the spatial relations among neighboring vertices. Birkholz [130] provides another parameterization method with mesh simplification. He utilizes the edge collapse method with a collapse order based on edge length to obtain the tetrahedron. He also develops an optimization process for shape-preserving with spherical angles from barycentric maps. However, it is reported from other work that this method may not be able to guarantee a valid parameterization (i.e., prevent triangle foldover).

Barycentric based convex boundary methods have been well developed in planar parameterization and extended to spherical base domain with Gauss-Seidel iterations. Kobbelt et al. [131] borrow the shrink wrapping process by adapting the deformable surface technique from image processing and apply it in parameterization. Alexa [132, 133] performs heuristic iterative procedures with uniform weights to achieve spherical parameterization for genus-zero polyhedra. Based on the parameterization, he implements feature correspondence and mesh merging algorithms for 3D mesh morphing. The mapping method is fairly easy to understand and implement, but the algorithm is slow to converge and is not verified to

always yield valid mapping results. From spectral graph theory and its extension, Gotsman et al. [134] generalize the method of barycentric coordinates for planar parameterization to solve the spherical mapping problem. They prove its correctness theoretically and provide a quadratic system of equations which is the spherical equivalent to the barycentric formulation. They do not provide an effective solution to this quadratic system, which limits its applications. To efficiently solve this large system of non-linear equations, Saba et al. [135] show the failure of solving the equations with simple iterative methods and introduce a successful numerical approach by using optimization methods associated with an algebraic multi-grid technique. Their method is claimed to guarantee a bijective spherical parameterization of closed manifold genus-zero meshes in a fast manner (parameterization for hundreds of thousands of vertices in minutes).

Researchers have also contributed other approaches to spherical parameterization. Praun and Hoppe [136] develop a scheme for sampling the spherical domain using uniformly subdivided polyhedral domains. With this scheme, a practical parameterization is implemented with the minimization of a stretch-based measure to reduce scale-distortion. Sheffer et al. [137] extend the idea of ABF (Angle-based Flattening) from 2D planar parameterization to the spherical case. Instead of dealing with positions for vertices, they formulate and solve an optimization procedure in terms of angles on the sphere. However, the ABF method applied to spherical parameterization appears to be less stable than the planar case and is impractical for large meshes.

CHAPTER 4

FAST SPHERICAL PARAMETERIZATION

In this work, parameterization is focused on closed genus-zero meshes. The method is based upon barycentric coordinates with convex boundary. Unlike most existing similar approaches which deal with each vertex in the mesh equally, the method developed in this research focuses primarily on resolving "spikes," or overlapping areas, that occur in the spherical domain during parameterization, which helps speed the parameterization process. The algorithm starts by normalizing the source mesh onto a unit sphere and followed by some initial relaxation via Gauss-Seidel iterations, as suggested by Alexa [132, 133]. Alexa's approach suggests continuing these relaxation steps until all the overlapping regions disappear. However, this can be very computationally demanding, and it is reported that such a process is not guaranteed to converge. Here a different approach is introduced. After the initial relaxation steps, most overlapping vertices are resolved. Then a novel solution for the remaining overlapping regions (as shown in Figure 6) is applied which are typically very hard to resolve. Finally, a minimization process is also implemented to reduce distortion. Due to its emphasis on solving only challenging overlapping regions, this parameterization process is much faster than existing spherical mapping methods.

4.1 Approach Overview

As reported in [135], there are two main challenges for most of existing spherical

parameterization methods. The first is the difficulty of making the procedures stable and convergent. Sometimes, these processes cause endless iterations or residual overlapping areas. No triangle overlapping is the most crucial requirement for a valid parameterization since a bijective one-to-one mapping is a must for many applications based on parameterization. The second challenge is the time-wise efficiency. Due to the computational complexity of some algorithms, the processes involve intense calculations so that the time spent could be up to hours or even more. These inefficient procedures will hinder many applications that require speed. The algorithm developed in this research is designed to be robust, yet faster than existing spherical parameterization methods. The flowchart of the fast spherical parameterization framework is shown in Figure 5.

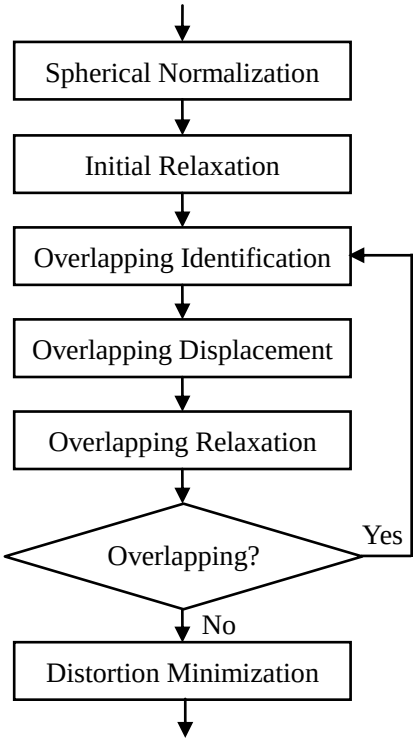


Figure 5: Flowchart of the fast spherical parameterization process

This algorithm targets the challenge of solving overlapping for most spherical parameterizations. Like some of the existing methods, barycentric maps are employed with uniform weight for parametric embedding but the method can be easily extended to other weight formats if necessary. The method consists of several procedures as shown in the diagram. These procedures include: i) Normalize and project each vertex from the source mesh onto a unit sphere; ii) Apply Gauss-Seidel iterations for the initial relaxation to solve most overlapping; iii) Identify and find remaining overlapping vertices; iv) Stretch each overlapping area and form a convex boundary for it; v) Fix these convex boundaries and map the overlapping vertices over related areas until no more overlapping exists; vi) Relax the whole spherical mesh with displacement constraint to minimize distortions.

4.2 Initial Mesh Relaxation

After a triangular mesh is loaded, it is normalized and projected onto a unit sphere directly, which keeps the distance between each vertex and the coordinate origin (0,0,0) to be 1. At this point, such spherical mesh contains massive irregular overlapping. To make it easier for further processing, we perform some initial relaxation from Gauss-Seidel procedure based on barycentric embedding.

$$p_i = \sum_{j \in N(i)} w_{ij} p_j \qquad i = 1,2,3, \dots, n$$

$$w_{ij} = \begin{cases} 1/d_i, & j \in N(i) \\ 0, & j \notin N(i) \end{cases}$$

$$\|p_i\| = 1, \quad i = 1,2,3, \dots, n$$

Where, $N(i)$ represents the indices for the neighboring vertices of the *i-th* vertex and d_i is the number of elements in $N(i)$.

Here a uniform mapping method is used with identical weight for each neighboring vertex. Alexa's [132, 133] parameterization employs such a method throughout the entire mapping process until all the overlapping regions are eliminated. Tests conducted with Alexa's approach indicate that in some cases, most (over 90%) of the vertices are displaced without overlapping each other within 100-200 iterations. However, to solve the remaining (~10%) overlapping vertices typically costs over 10,000 iterations. Considering the number of vertices involved, these later iteration steps are very expensive and computationally inefficient. Further, in some cases these iterations can be endless and the overlapping cannot be removed completely. So, a small number of relaxation iterations can be applied to solve most of the overlapping regions and the following solution is developed to target the remaining overlapping areas.

4.3 Overlapping Solution

Instead of treating each vertex equally the method developed in this research focuses on parameterization of overlapping areas that remain after initial Gauss-Seidel relaxation. As shown in Figure 6, after the initial relaxation process, most vertices of the mesh are well located except for some overlapping spikes in areas with high populations of vertices. The

relaxation is akin to the process of dragging and expanding these areas vertex-by-vertex.

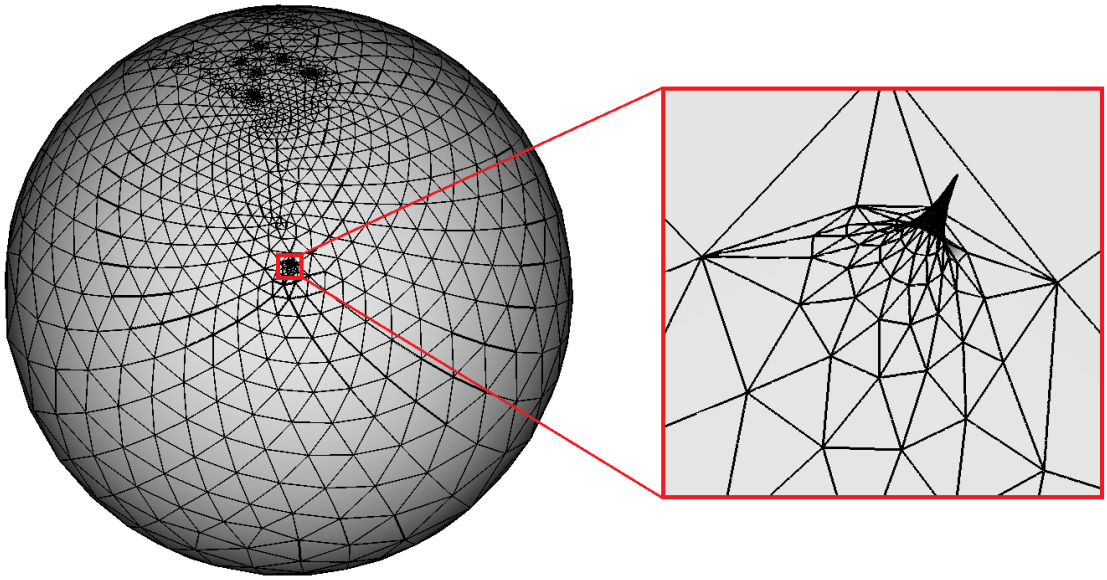


Figure 6: Remaining overlapping spike after initial relaxation

With the distribution of highly dense vertices in these areas, however, the relaxation can be extremely slow and unreliable. The basic idea of the method developed here is to stretch the boundary of these overlapping areas so that more space will be generated to accelerate the mapping process. The overlapping solution begins with finding overlapping area, followed by stretching such areas. Finally vertices within these areas will be relaxed with fixed boundaries.

4.3.1 Overlapping Identification

This step determines the overlapping regions and identifies the vertices involved. A vertex is identified as overlapping if the line segment between it and the sphere center (referred to as the cardinal line segment) intersects any other triangle on the sphere. The steps to check overlapping are: i) intersect the cardinal line with the plane defined by the three

vertices from the subject triangle; ii) if so, check whether the intersection point lies within the cardinal line segment; iii) and if so, check if the intersection point lies within the subject triangle.

Nominally, every vertex must be checked against every triangle in the mesh, which can quite time-consuming. For example, a test using this naive approach on a typical spherical triangular mesh with about 2,500 vertices takes about 9 seconds to find all the overlapping vertices. To address this inefficiency a neighborhood searching algorithm is employed to accelerate this task. First, the spherical surface normal of all the triangles is computed, based on their connectivity, to identify the "folded" triangles, i.e., those spherical normal direction toward the inside of the sphere. Then, based on the vertices of the "folded" triangles, their neighboring vertices are determined and checked for overlap with any of the "folded" triangles. During this process, if none of the neighbors for a selected vertex overlap with any of the "folded" triangles, searching around that vertex stops and the other ones are similarly processed until each folded triangle has no overlapping neighbors. This identifies a local boundary around the folded triangle. This searching algorithm is very accurate and efficient. With the same test mesh mentioned above, with about 2,500 vertices, this searching algorithm completes the operation in a time scale of milliseconds.

4.3.2 Overlapping Displacement

Overlapping vertices are identified and marked from the previous step. This is followed by displacing overlapping vertices. Before any manipulation is applied, the overlapping

vertices are sorted into a nested structure to facilitate efficient processing. There will usually be multiple overlapping areas as shown, for example, in Figure 6. The sorting will first separate the overlapping vertices into groups based on their connectivity with each other, which will put all connected overlapping vertices into a group. After that, each group of vertices will then be sorted from the shortest distance to the boundary (non-overlapping) vertices. This process forms several nested groups of overlapping vertices for each spherical triangulation.

The idea of mapping only the overlapping areas is essentially an attempt to simplify the 3D parameterization problem into several 2D ones. To assist local parameterization for each overlapping area, a stretching process is employed to generate locally convex spaces, as shown in Figure 7(a). This is a modified Gauss-Seidel procedure which sets the weight for each overlapping vertex to zero when it is involved as a neighbor vertex in calculation. And the identified vertices in the overlapping area will keeep their current positions during this step. In this process, the non-overlapping outer vertices essentially pull the boundary of the overlapping area away from the overlapping center. A convex or close to a convex shaped boundary will be created after this process. It is followed by relocating overlapping boundary vertices, which will ensure the boundary to be convex or close to convex. As shown in Figure 7(b), for each overlapping region, the overlapping vertices are all placed together on the centroid of the convex boundary. Such convex boundaries will ensure the local parameterization (simplified into a 2D problem) to be converged.

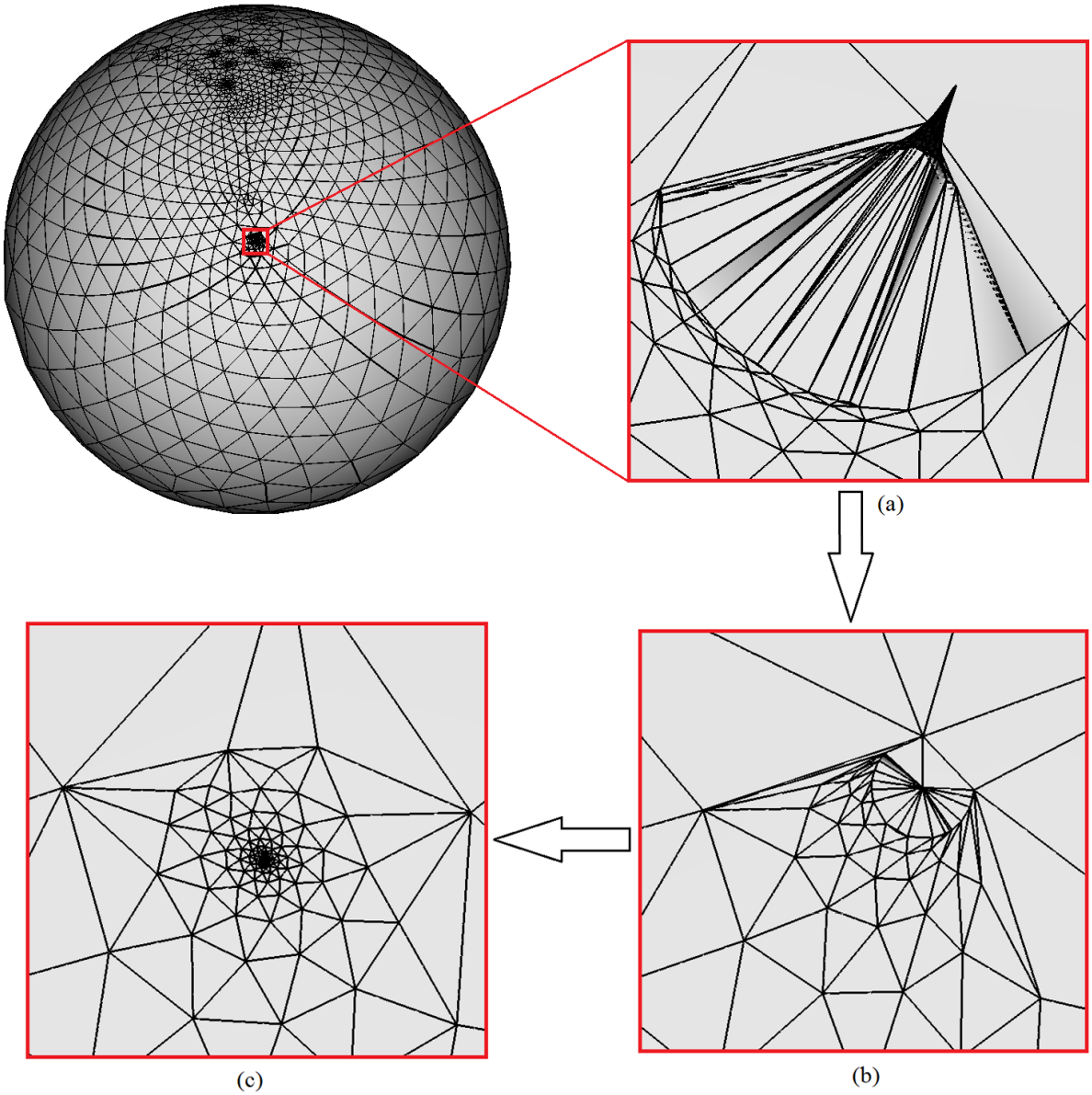


Figure 7: Solution for overlapping: (a) stretched overlapping area; (b) overlapping vertices placed on centroid; (c) overlapping and boundary vertices relaxed

4.3.3 Overlapping Relaxation

For 2D planar parameterization, a regular convex fixed boundary would theoretically guarantee the existence of a bijective mapping, which is the motivation of overlapping displacement from previous step. In this relaxation step, all the non-overlapping vertices and

all the vertices on the overlapping boundaries are fixed. Vertices in the overlapping areas are released with Gauss-Seidel iterations until all the overlapping areas disappear, as shown in Figure 7(c). The positions for boundary vertices are also recalculated and updated after all the overlapping is eliminated. These operations are repeated until all the overlapping areas are resolved.

4.4 Parameterization Distortion Minimization

This method focuses on solving overlapping with some imposed stretching, which could lead to dislocation for vertices that are moved away from their parametric locations. To minimize distortions created from these dislocations, vertices on the mesh need to be relocated as close to their parametric positions as possible. Testing indicates that directly relaxing the entire mesh at this moment will generally re-introduce overlapping regions.

To minimize these distortions, a minimization method is introduced that guarantees bijectivity and avoids new overlapping region generation during the final global relaxation process. In this method, all vertices are classified into two categories: regular and sensitive. The sensitive ones refer to the overlapping vertices recognized from the previous overlapping identification step. As shown in Figure 8, the current position for vertex P is V and the new position V' is calculated from their neighboring vertices (V_1, V_2, V_3 and V_4) positions. Clearly the transition from V to V' takes vertex P outside the region of $V_1V_2V_3V_4$, which causes overlapping. Such movement makes the mapping non-bijective and must be prohibited.

However, if the movement of vertex P is skipped at this step, the non-overlapping property for this local area may be lost, which can slow down or even prevent local distortion minimization. A solution is to keep the same movement with the same path, but use a smaller scalar magnitude. As shown in Figure 8, the scale is defined by VV'', where V'' is the updated new position for vertex P which can be expressed as:

$$V'' = V + (1 - \varepsilon) \cdot \overrightarrow{VP_{int}}$$

Here, P_{int} is the overlapping critical position for vertex P which can be calculated from the intersection between VV' and neighbor edge (V_1V_2, V_2V_3, V_3V_4 or V_4V_1).

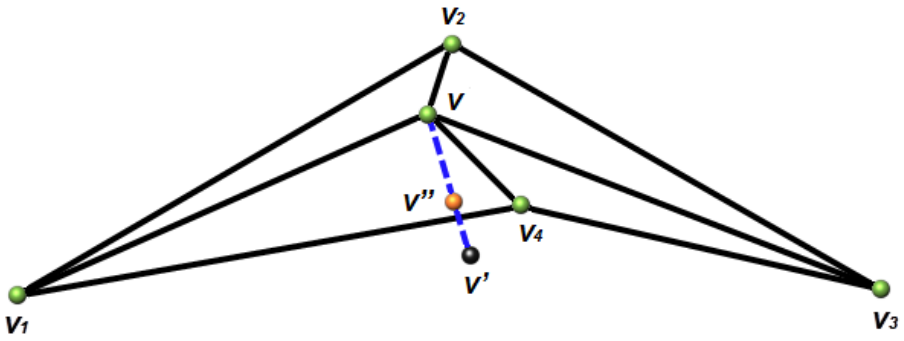


Figure 8: Vertex relocation without creating new overlapping

CHAPTER 5

FEATURES ALIGNMENT WITH EXTRACTED SKELETON

As mentioned in Chapter 3, mesh morphing involves much more than simply interpolating among source geometries. The intermediate mesh blends should preserve featuring information from source geometries. For example, a model generated from morphing between a horse model and a pig model should probably have four legs rather than eight resulting from morphing without features alignment. Moreover, the smooth evolution of featuring components from one model to another can facilitate users' decision-making. For engineering conceptual design, such feature evolution enables designers to generate new design concepts from a pool of existing designs.

Many existing mesh morphing methods use a single point (usually geometric centroid or an approximation of it) to represent a geometric feature component. By aligning corresponding feature points from different models, some reasonable outputs may be obtained. However, features are generally comprised of more than one vertex (more typically they are collections of triangle) so mesh morphing with point-based feature alignment can lead to parts that are not fully aligned. The solution developed in this research aims to map not only representative feature points (e.g., toe or finger tip points of an animal human model) but also correlates feature areas (e.g., an entire front-left leg from an animal model) completely.

5.1 Approach Overview

Rather than aligning simple feature vertices, this feature alignment method focuses on aligning the entire feature areas from each model. This feature alignment process follows after spherical parameterization or each subject model and is performed before spherical subdivision based remeshing. The process does not directly modify original input geometries but rather establishes correspondences by adjusting their spherical maps. The algorithm is summarized in Figure 9.

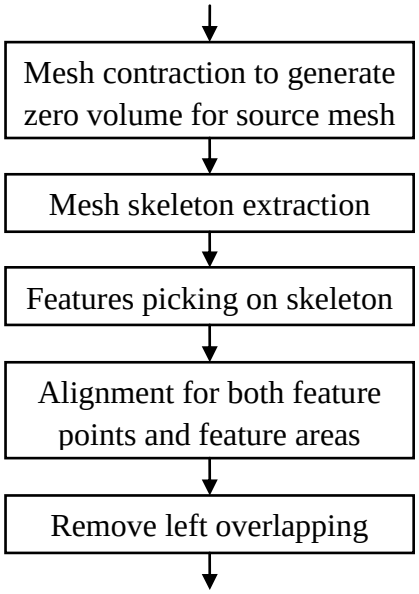


Figure 9: Flowchart of the feature alignment process

First extract the skeletons for all the source geometries is generated based on the algorithm developed by Au et al [155]. Each vertex from the skeleton has a group of vertices on the original mesh associated with it. Therefore, the skeleton wires can be used to assist users in features picking. Based on the picked vertices in skeleton, feature areas and feature

points for each model are be calculated and identified. Then, as described in detail below, the feature points are used to develop a set of constraints that describe the desired feature relationships and spatial transforms are generated with least-squares singular value decomposition. After solving this system, the resulting spatial transforms are applied to the models to obtain roughly similar orientations. Each feature point is then relocated to the centroid of all feature points that belong to the same feature group. The surrounding feature areas for these feature points are moved correspondingly. Following this, overlapping generated from the process is relaxed and the size of each feature area is adjusted to match with corresponding feature areas from other models, as much as possible. After all these steps, a spherical map for each model will be aligned with other ones in terms of feature points and feature area. In the following sections, this procedure is presented and discussed.

5.2 Mesh Skeleton Extraction

Before aligning geometric features, a method must be developed to define them from the input meshes. The most straightforward way would be to directly pick feature vertices/triangles with a graphical user interface. However, there are several shortcomings of this method. First, in some scenarios, it is difficult to recognize or pick features for some geometric models. Given their complexity, the picking process could take a very long time, and it would require a fairly complex user interface with substantial view manipulation and selection interaction.

A skeleton can be is a useful abstraction for interacting with complex shapes. For example, from human skeleton, one can easily identify the different regions for human body such as arms, legs, etc. In the field of computational geometry, algorithms to compute a wireframe skeleton from a triangular mesh have been proposed to generate a simplified natural representation of the geometry and topology of 3D object. In this work, a skeleton extraction algorithm presented by [155] is employed for skeleton generation. By interacting with representative skeleton curves, users can clearly identify and select features. The selected feature areas and points are then used to compute global spatial transforms that are applied to each model to affect the desired feature alignment. In addition to simplified interaction, another reason for choosing the skeleton for feature alignment is that this method has potential to be extended for automatic feature alignment for 3D objects with similar feature distributions. Feature recognition algorithms could be developed to identify all features based on the abstracted skeletons and align them automatically. This could be further explored in the future work.

5.2.1 Geometry Contraction by Laplacian Smoothing

The method [155] starts with a mesh contraction process which contracts the mesh geometry into a zero-volume skeleton shape by applying implicit Laplacian smoothing with global positional constraints. The mesh connectivity among vertices is not be changed and the key features of original mesh are preserved. Usually, a big challenge of mesh contraction is to control the contraction process such that it leads to a collapsed shape that can

approximate the input model instead of some random shape or perhaps shrinking into point. Au's method presents an iterative contraction process to solve a sequence of constrained Laplace equations with weaker positional constraints, which removes details and noise from mesh surface by moving the vertices along their curvature normal directions. An implicit updating scheme controlled by anchor points provides positional constraints to avoid converging into a single point that can result from smoothing with an unconstrained normal flow.

For Laplacian smoothing, the vertex positions are contracted along their normal directions. The whole smoothing process is governed by the discrete Laplace equation. The curvature-flow Laplace operator *L* is defined as,

$$LV' = 0$$

$$L_{ij} = \begin{cases} \omega_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}) & if\ (i,j) \in E \\ \sum_{(i,k)\in E}^{k} -\omega_{ik} & if\ i = j \\ 0 & otherwise \end{cases}$$

Where, *V* represents the group of vertex positions and *E* is the edges for all the connected vertices. α_{ij} and β_{ij} are the opposite angles corresponding to edge (i,j). By solving this discrete Laplace equation, the normal components would be removed and the mesh would be contracted under the "force" provided by contraction constraints.

To eliminate the singularity of the Laplace operator, extra constraints are put into the system. These constraints, called attraction constraints, are also defined to preserve the

original shape of the mesh with soft constraints to keep their current positions. Therefore, contraction constraints and attraction constraints are included in the system as,

$$\begin{bmatrix} W_L \cdot L \\ W_H \end{bmatrix} V' = \begin{bmatrix} 0 \\ W_H \cdot V \end{bmatrix}$$

Where, W_L is the diagonal contraction weighting matrix and W_H is the diagonal attraction weighting matrix. This system is over determined and needs to solved with least squares by minimizing the quadratic energy as,

$$\left\| W_L \cdot L \cdot V' \right\|^2 + \sum_i (W_{H,i}^2 \cdot \left\| v_i' - v_i \right\|^2)$$

For each iteration step, the contraction weights and attraction weights are updated based on the current state. The vertices with smaller contracted one-ring area (area formed by all directly neighboring triangles for a vertex) are controlled to be attracted more strongly to their current position and contract less in the next iteration. Each step, the new vertex positions can be obtained from last step (*t* is step),

$$\begin{bmatrix} W_L^t \cdot L^t \\ W_H^t \end{bmatrix} V^{t+1} = \begin{bmatrix} 0 \\ W_H^t \cdot V^t \end{bmatrix}$$

The contraction weights and attraction weights for each iteration step are defined as,

$$W_L^{t+1} = S_L \cdot W_L^t \qquad \text{with } W_L^0 = 10^{-3}\sqrt{A}$$

$$W_{H,i}^{t+1} = W_{H,i}^0 \sqrt{A_i^0 / A_i^t} \qquad \text{with } W_H^0 = 1.0$$

Figure 10 shows some results from geometry contraction based on the algorithm described above.

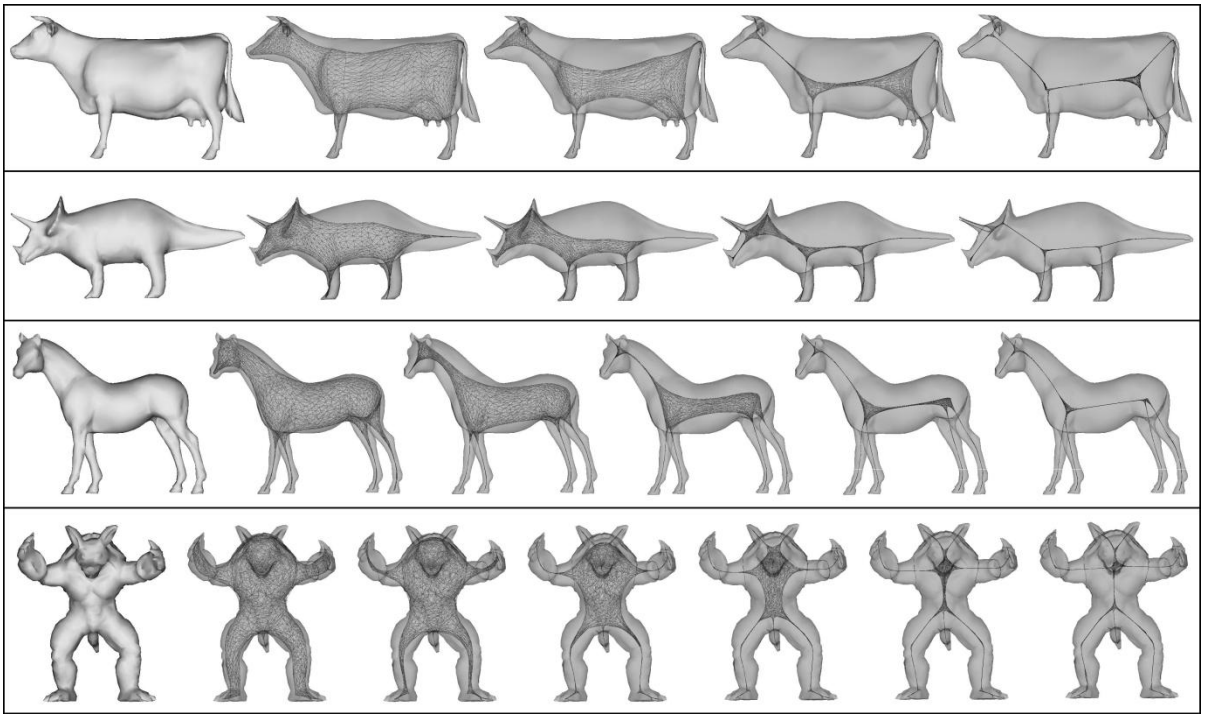


Figure 10: Results of 3D mesh contraction from left to right

5.2.2 Skeleton Extraction from Edge-collapses

After geometry contraction, the mesh shrinks to almost zero volume but still keeps the same connectivity. To generate a skeleton wire, a skeleton extraction is performed with connectivity surgery from edge-collapses [155]. The shape of the contracted mesh needs to be preserved with sufficient skeleton nodes in the extracted skeleton. The orders or edge collapse are decided by a defined cost function consisting of a shape cost and a sampling cost. For each iteration step, the edge with minimum cost will be collapsed. After collapse, all faces surrounding the edge will also be removed. The topology of the original mesh will be preserved and connected components will not be disconnected.

According to Au et al [155], the shape cost represents the potential distortion caused by

an edge collapse which is calculated with an error metric at each vertex. Since the volume of the contracted mesh is near zero, the triangles also have near zero area. Instead of computing the sum of squared distance between a vertex and its neighboring triangles, the error metric is measured by the sum of squared distances from a vertex to all related adjacent edges. For each potential edge collapse from vertex *i* to *j*, the shape cost can be defined as,

$$F_a(i,j) = f_i(v_j) + f_j(v_j)$$

Where, the initial error metric of vertex *i* is defined as the sum of all the squared distances to its neighboring edges,

$$f_i(v) = v^T \sum_{(i,j)\in E} (K_{ij}^T K_{ij})v = v^T Q_i v$$

The Matrix *K* is defined as,

$$K_{ij} = \begin{bmatrix} 0 & -a_z & a_y & -b_x \\ a_z & 0 & -a_x & -b_y \\ -a_y & a_x & 0 & -b_z \end{bmatrix}$$

Where, *a* is the normalized edge vector and *b* is the cross product defined as,

$$b = a \times v_i$$

This shape cost controls the order of the edge collapse sequence but could leads to over-collapse which will lose some nodes on the skeleton and make the final skeleton too coarse. The sampling cost is defined to penalize long edge generation and prevent such a problem. It calculates the total travel distance of related edges during collapse,

$$F_b(i,j) = \|v_i - v_j\| \cdot \sum_{(i,k)\in E} \|v_i - v_k\|$$

The total cost for edge collapse from vertex i to j is defined with weight parameter as,

$$F(i,j) = w_a \cdot F_a(i,j) + w_b \cdot F_b(i,j)$$

Figure 11 presents some results of skeleton extraction from contracted mesh.

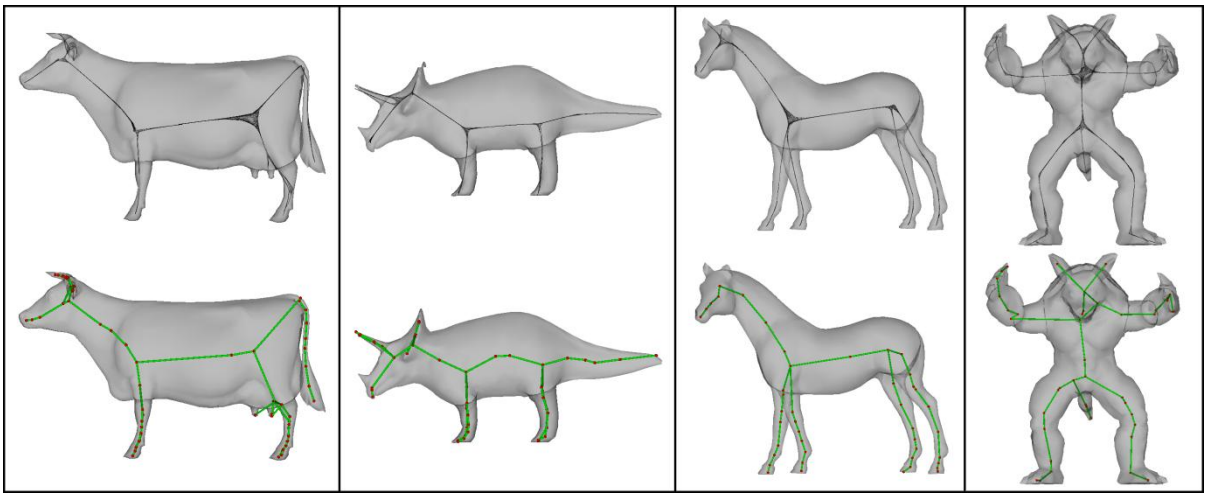


Figure 11: Skeleton extractions (lower) from contracted mesh (upper)

5.3 Features Alignment

For each input mesh model, a skeleton is generated as an abstract representation. After this, it is used to facilitate the user's identification and selection features from different models. As the user selects a feature with a graphical user interface, both feature points and corresponding areas are highlighted. The feature points on the spherical map for each model are grouped and an initial alignment for them is performed based on a least-square singular value decomposition (SVD) that minimizes the sum of the distances between all corresponding vertices. Then the feature points and feature areas are scaled and relocated to match each other. Finally, all overlapping generated from this process is eliminated so that

the spherical maps for each mesh remain bijective with respect to the original input geometries.

5.3.1 Features Picking with Skeleton

During the skeleton extraction process, most skeleton edges are collapsed and the related pairs of vertices are merged together. Based on the order of the collapses ruled by the shape and sampling costs, each node on the resulting skeleton actually includes a group of adjacent vertices from the original mesh. Usually, these groups of vertices are geometrically close to the merged node. Thus, by picking the nodes on the skeleton, users can actually select the corresponding vertices on the original model.

In some cases, mesh vertices may not correspond to the closest node on the skeleton. This is solved by sorting all the mesh vertices again to regenerate the proper spatial correspondences. To accelerate the sorting process, a *kd*-tree data structure is applied and works well. The first picked skeleton node is noted as the starting node and its corresponding mesh vertices are employed to calculate the feature point from their area centroid. The second skeleton node picked is noted as ending node and its corresponding mesh vertices define the boundary of the feature areas. After the ending node is picked, a node traversal is performed from the starting node to the ending node. All the nodes passed are noted as feature nodes and the feature areas are derived from the corresponding mesh vertices. This process is illustrated in Figure 12.

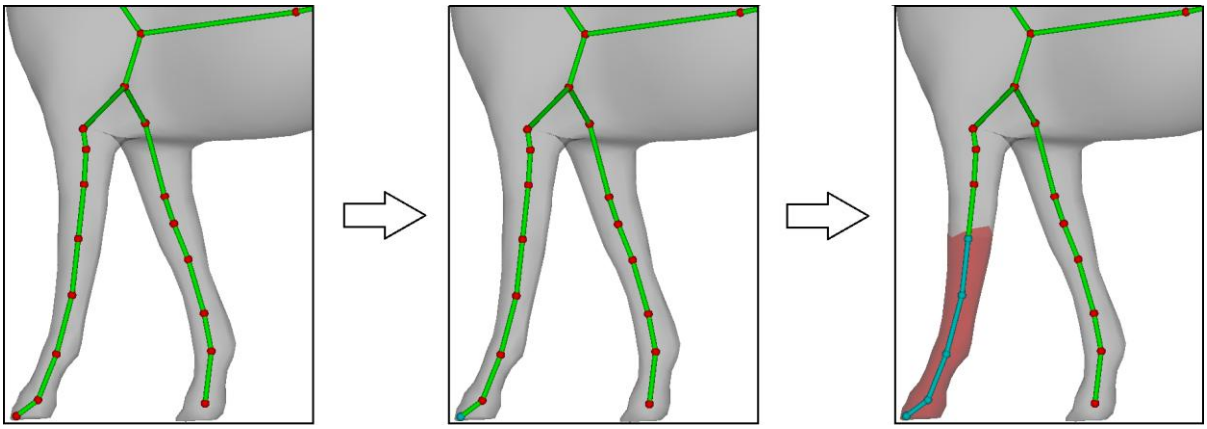


Figure 12: Feature selection by picking nodes (in blue) on the skeleton: before first feature node selection (left); first feature node selected (middle); second feature node selected (right)

5.3.2 Initial Alignment with Singular Value Decomposition

Each feature is described with a feature point and a feature area. Since the feature point is computed as the centroid of the corresponding mesh points, is located within the feature area. Nominally, the original input models will have different orientations in terms of features, so one model is arbitrarily chosen to represent the base orientation, and spatial transformations for each of the other mesh models are computed to rotate them so that the defined features in each are in rough alignment. The feature points are utilized for this process, and a least-squares singular value decomposition is formulated and solved to find the desired rotational matrices for each model.

The first model loaded is considered as the reference and is not be rotated during this initial alignment. The other models are transformed such that their feature points will match the ones in the reference model. If there are k input models and each model has n features defined, the transformation can be expressed as the following optimization problem,

$$\begin{cases} \min \sum_{j=1}^{n} dist\{P_i(j) - P_1(j)\}, & i = 1,2,3,\ldots,k \\ dist\{P_i'(j) - P_i'(l)\} = dist\{P_i(j) - P_i(l)\}, & \forall\, j,l = 1,2,3,\ldots,n \end{cases}$$

Where, P_i represents the matrix containing coordinates for all feature points on model i and P_i' represents the coordinates after transformation,

$$P_i = \begin{bmatrix} x_{i1} & y_{i1} & z_{i1} \\ x_{i2} & y_{i2} & z_{i2} \\ x_{i3} & y_{i3} & z_{i3} \\ \vdots & \vdots & \vdots \\ x_{in} & y_{in} & z_{in} \end{bmatrix}$$

The problem is to find the rotational matrix for each of other models such that the sum of distance between corresponding feature points is minimized. This can be solved by applying singular value decomposition (SVD). For each set of P_i, we can define matrix C_i as,

$$C_i = P_1^T \cdot P_i$$

$$C_i = U_i \cdot \Sigma_i \cdot V_i^T$$

Where, Σ_i is a diagonal matrix with nonnegative real numbers on the diagonal, The diagonal entries $\Sigma_i(a,b)$ of Σ_i are known as the singular values of C_i. The columns of U_i and the columns of V_i are called the left singular vectors and right singular vectors of C_i, respectively. By applying SVD, the corresponding matrices U_i, Σ_i and V_i^T can be obtained,

$$[U_i,\ \Sigma_i,\ V_i^T] = svd(C_i)$$

Then, the rotational matrix can be expressed as,

$$R_i = U_i \cdot\ V_i^T$$

And the new position for all the feature points in model *i* are calculated from,

$$P_i^{'} = R_i \cdot P_i$$

5.3.3 Features Relocation and Alignment

The initial feature alignment process reduces the distances between features without modifying the local vertex positions on the spherical maps. This eliminates most of the positional differences. As discussed in previous section, since the constraint matrix is over-determined, such a global rotation will not accommodate all the relative differences between corresponding features and align corresponding feature points without changing the relative locations of feature points within the same spherical map.

To achieve full alignment, a local transformation must be applied to the feature points and areas. The steps to compute these local transformations are: i) calculate the centroid of corresponding feature points on each model's spherical map; ii) relocate all the feature points onto the centroid calculated from last step; iii) compute the rotation matrix (based on origin) for each feature point's relocation; iv) apply this rotation transformation to all vertices in the feature area related to the feature point; v) eliminate all the overlapping generated from these local transformations with a similiar relaxation method as in Chapter 4; vi) and finally, resize each of the corresponding feature areas to identical size and make sure no overlapping created.

An example of feature alignment is shown in Figure 13. The selected features (four legs) are marked with red circles as shown in the figure.

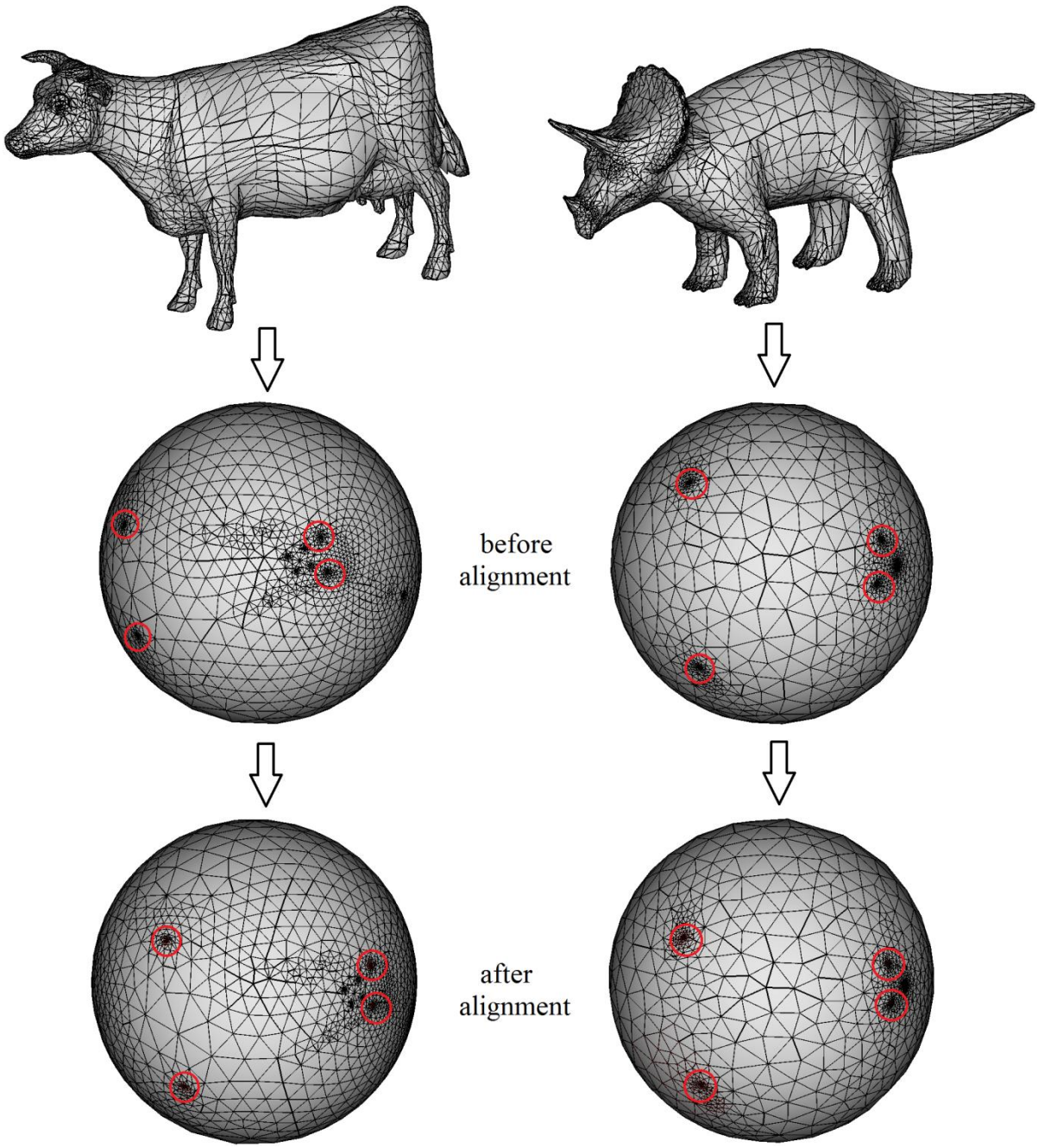


Figure 13: Features selection and alignment with spherical maps

CHAPTER 6

REMESHING WITH MESH SUBDIVISION

Given multiple feature-aligned meshes with corresponding bijective spherical parameterizations, a 3D remeshing method is developed to facilitate robust morphing between the input meshes. This method is derived from the concept of spherical mesh subdivision and leverages the spherical parameterization techniques described earlier. The local recursive subdivision can be set to correspond to the level of detail (LOD) of the source spherical meshes. Alternatively, the LOD can be controlled to allow output with variable resolutions. This multi-resolution subdivision scheme employs a triangular validation process that assures a valid triangulation for the resulting morphed mesh. The final mesh merging and reconstruction process produces the morphed mesh model with the desired LOD specified from user.

6.1 Approach Overview

After parameterizing surface meshes onto the spherical domain, and conducting feature alignment as described above, a spherical remeshing method is introduced to facilitate morphing. This method is proposed in order to generate a common connectivity for different mesh models in our mesh morphing framework. The concept of spherical mesh subdivision is introduced and extended to develop the remeshing algorithm. Spherical mesh subdivision refers to adding detail into the spherical triangular mesh by breaking related triangles into

smaller ones. Unlike most existing work, this algorithm will subdivide only those areas that contain detailed geometric information rather than subdividing every triangle evenly. This process makes use of few vertices while still covering every geometry detail. Usually, to remesh one single model, the number of vertices for remeshing is approximately the same number as the input mesh. And for multiple input models, the final merged representation has fewer vertices than the sum of all input meshes after aligning the feature areas which typically involve large vertex crowds. For most existing morphing methods, the number of vertices employed in the merged representation is much more than the total vertices from source meshes. The flowchart for our remeshing with spherical subdivision is shown in Figure 14.

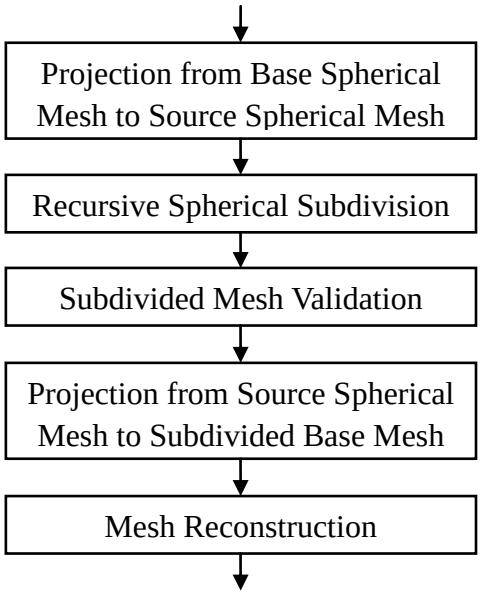


Figure 14: Flowchart of remeshing with spherical subdivision

6.2 Base Spherical Triangulation

The subdivision process is to break existing triangles in the base spherical mesh into smaller ones in the region with details. Such base mesh is employed to generate merged remeshing model with common connectivity for each input model before morphing. As shown in Figure 15, a triangle is subdivided by removing the existing triangular connectivity ($v_1v_2v_3$) and adding new connectivities ($v_1v_{12}v_{31}$, $v_2v_{23}v_{12}$, $v_3v_{31}v_{23}$ and $v_{12}v_{23}v_{31}$) resulting in the generation of four new triangles. Here v_{12}, v_{23} and v_{31} represent the midpoint for original edges v_1v_2, v_2v_3 and v_3v_1 respectively.

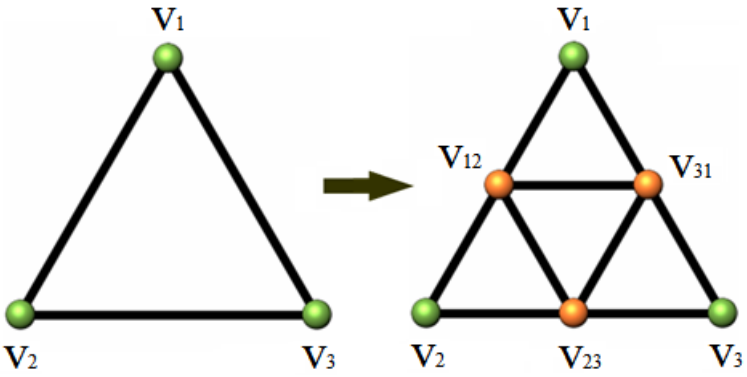
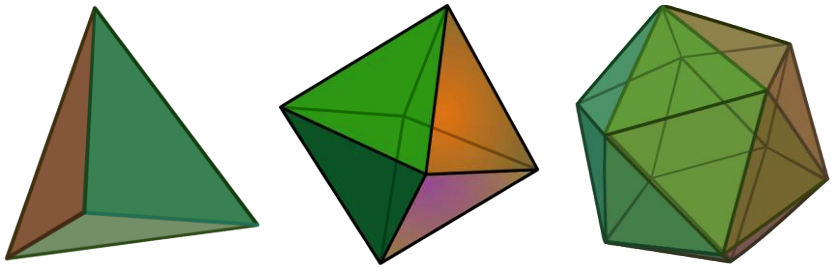


Figure 15: Subdivision by breaking one triangle into four

Since bijective spherical parameterization is applied for the closed genus-zero input meshes, the base mesh is equivalent to a sphere topologically. Thus, platonic solids with regular polygons for each face and closed surfaces are ideal choices as initial base meshes. As shown in Figure 16, there are three platonic solids with triangular faces: tetrahedron, octahedron and icosahedron. All three are made available in the remeshing algorithm presented here, and further investigation and comparison regarding the remeshing qualities

from these different types of initial base meshes will be explored in the future.



(a) (b) (c)

Figure 16: Triangular platonic solids: (a) tetrahedron; (b) octahedron; (c) icosahedron

6.3 Recursive Spherical Subdivision

To remesh from a source mesh, the algorithm needs to match the level of detail (LOD) of the source mesh. This requires that the detailed areas (those that contain dense collections of vertices) be well represented in the remeshed representation. A simple approach would be to use a single uniform subdivision (sufficient to capture detailed areas) over the entire mesh, but this would generate a large number of unnecessary vertices. So a local spherical subdivision method is developed and implemented which subdivides the triangles in the base mesh to match the corresponding local LOD in the source mesh. In this way, no excess vertex distribution will be expended in the areas that contain little geometric detail.

This method employs a recursive subdivision process which can be divided into several steps. First, all the vertices of the source spherical mesh are projected onto the base spherical mesh. Then, classify and label each vertex of the source mesh with respect which triangle on the base mesh contains it. Finally, check the number source member vertices for each triangle

on the base mesh and perform subdivision. If a base mesh triangle contains more than a preset number (e.g., 1) of source mesh vertices, is is subdivided into four smaller ones as in the previous section. Each of the resulting four triangles is similarly checked and subdivided recursively until every triangle in the base mesh has no more than the preset number of source mesh vertices. Figure 17 shows the results of this recursive subdivision algorithm from the spherical representation of an input model.

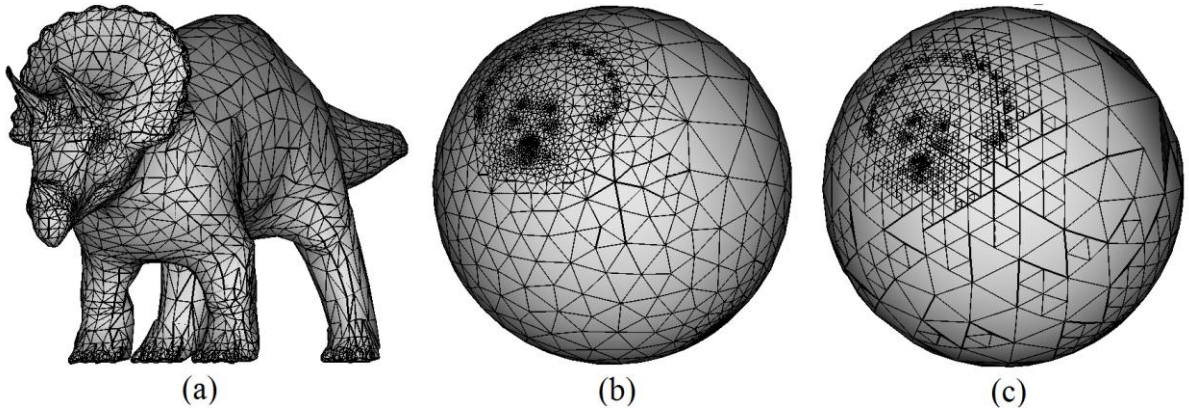


Figure 17: Spherical subdivision: (a) original source mesh; (b) parameterized spherical mesh; (c) remeshed subdivision (LOD = 1)

6.4 Validation for Subdivided Triangulation

As the figure shows, the resulting subdivided base mesh reflects the geometric characteristics of the source mesh. However, the resulting spherical base mesh is not a valid spherical triangulation since not all the faces are 3-connected, as shown in Figure 17(c). To address this, a triangulation validation process is introduced which restores 3-connected connectivity without inserting or removing additional vertices.

Invalid triangles occur in the triangles whose neighboring triangles have been

subdivided. Middle-vertices for associated edges generated from neighboring subdivision cause the problem. Fortunately, there are only three conditions for refined tessellation of a triangle depending on how many of its adjacent neighboring triangles have been subdivided. Figure 18 shows the three cases and the subsequent tessellation required to restore 3-connectivity. Note that the tessellation does not introduce additional vertices, only triangles.

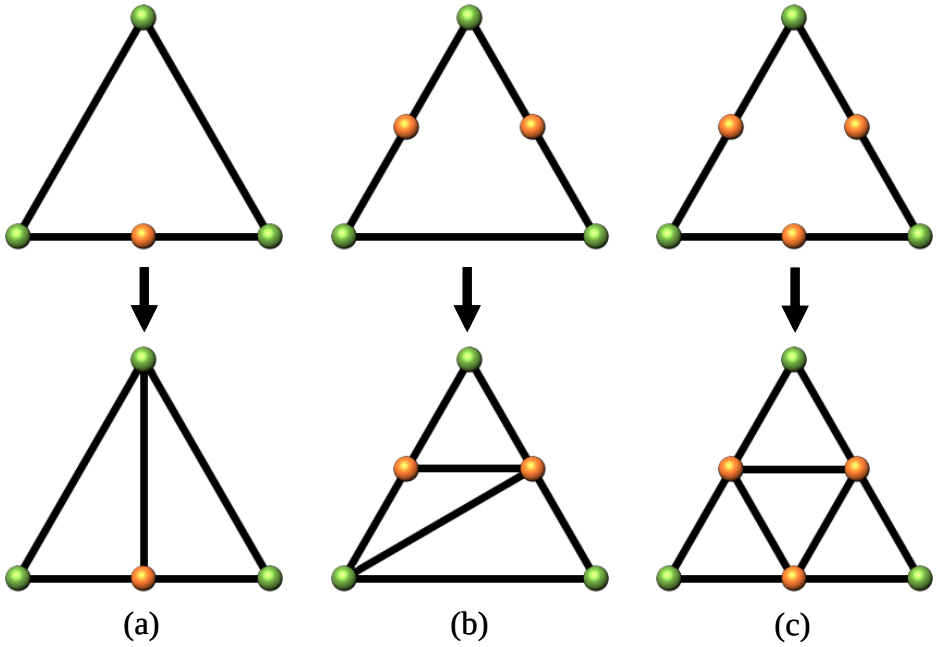


Figure 18: Three cases of correction for a triangle with subdivided neighbor(s)

Figure 19 presents a result of a remeshed spherical triangulation from an example mesh. The representation matches the details from subdivision and maintains a valid spherical triangulation.

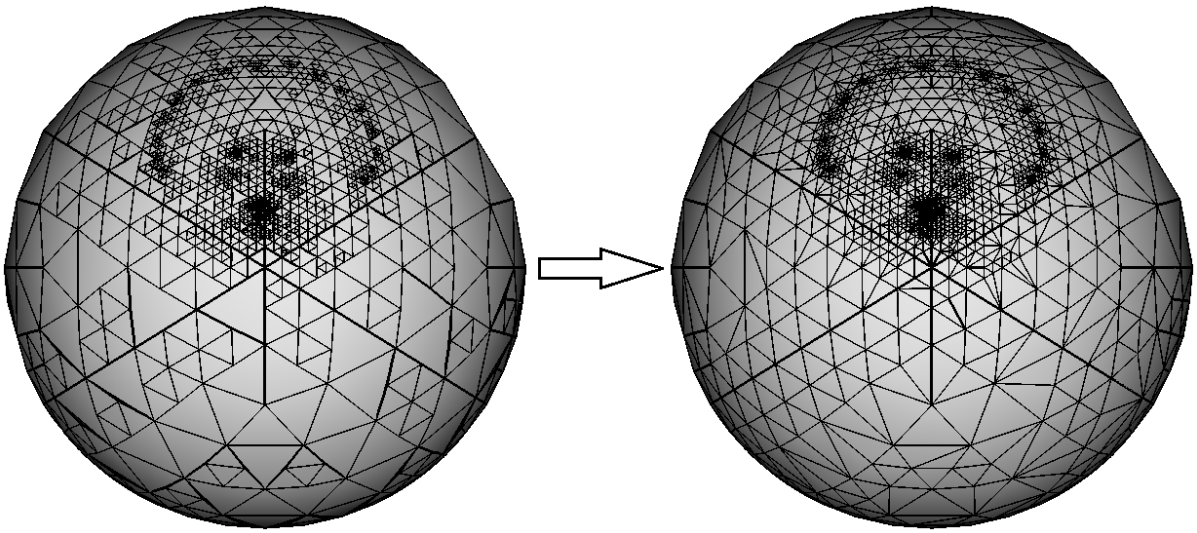


Figure 19: Triangulation validation for a spherical subdivision

6.5 3D Mesh Reconstruction

Give the reconstructed mesh in its parametric spherical representation an equivalent 3D spatial representation can be generated from it by essentially reversing the subdivision process. First, the reconstructed spherical parameterization of the mesh generated via subdivision is projected onto the spherical parametric mesh of the original model. Intersections between the vertices from the reconstructed mesh and triangles from the original spherical mesh (number of hits for each triangle could be more than 1) are calculated. Then the barycentric coordinates of the intersection point are calculated in the associated triangle from the original spherical mesh. These coordinates represent blending weights for three vertices in the original triangle that will generate the intersection point (i.e., a vertex from the reconstructed mesh).

With the barycentric coordinates for each vertex of the reconstructed mesh embedded into original spherical mesh, a remeshed 3D model can be generated from the original source model by blending corresponding vertices. To apply this remeshing method into our mesh morphing framework, a simple platonic base model is subdivided by projection with the first input model and then the subdivided model will be used as base model for projection with the next input model, and so on. In this way, a merge geometry representative model is generated containing all geometry information from all input models. Some results of reconstructed meshes will be presented in Chapter 8.

CHAPTER 7

3D MESH METAMORPHOSIS

7.1 Previous Work in Mesh Metamorphosis

Morphing, or metamorphosis, aims to produce a smooth and continuous sequence that transforms from a source object into a target object, or perhaps even performs transformation among more than two objects. The idea of morphing was originally initiated and developed with 2D image applications. Wolberg [138] describes techniques where pair of 2D images is used to map features from one image to the other. This is done by having an artist choose a point in the first image, and decide where on second image would be the most interesting position for the mapping. After all these pairs are determined, the method processes the neighbors surrounding the points of interest, and then generates a sequence of metamorphosis over time for the selected images.

With the increase in 3D applications in computer graphics, these 2D image techniques have been extended into 3D geometry models. Usually, methods of 3D morphing can be categorized into two major approaches. The first one is volume-based method [139, 140, 141, 142] which blends volumes where the source and target shapes are embedded. This method is able to support topological changes throughout the morphing process. However, it is computationally expensive, and in many cases, geometry boundaries generated from volume-based method are not smooth. The second general approach is surface-based methods which blend mesh representations for input objects. As with most recent 3D morphing techniques,

the method developed in this research falls into the surface-based category. More related work is reviewed and discussed in the following section.

Most 3D surface-based mesh metamorphosis techniques involve two steps. The first is to find the mapping from the source to the target meshes and establish the feature correspondence between them. This step usually employs mesh parameterization techniques and related work that has been reviewed in previous chapters. The parameterization process requires that the mapping for the meshes must be bijective, i.e. generating a one-to-one correspondence between the source and the parameterized meshes. The second step is to choose a continuous path for each vertex and produce a smooth sequence of intermediate geometries by interpolation of corresponding vertices. A linear procedure is performed for this interpolation in most cases.

Some researchers consider generating a common connectivity for pairs of shapes of genus-zero by applying topological merging methods. Given two input meshes with convex and star shapes, Kent [143] apply an algorithm to calculate a merging topology that can represent the geometric information for both the source and the target meshes. Then, the morphing process linearly interpolates the vertex positions over time, which transforms the source mesh into the target. Alexa [132] develop a procedure to embed genus-zero meshes onto a unit spherical domain. This is followed by aligning corresponding features for the source and target meshes with a simple iterative scheme inspired by radial basis functions. The merging process generates a common connectivity for both the source and target meshes

by solving a spherical map overlay problem. After reconstructing the source and target meshes from the spherical representation with the common connectivity, the final morphing step is accomplished with linear interpolation between corresponding vertices in the reconstructed meshes. Instead of mapping the meshes onto a spherical parametric domain, Shapiro and Tal [129] map them into a convex polyhedra by employing a parameterization process consists of two phases: simplification and reattachment. They also apply a mesh merging process to create isomorphic representations for the input mesh pairs.

Kanai et al. [144] present a method which employs harmonic mapping to embed the source and target meshes onto a 2D unit circle and establishes vertex correspondences by the intermediate objects generated from overlapping the two embeddings. The interpolation is controlled through an assigned boundary loop and a vertex on that boundary. In extended work by the same authors [145], an efficient morphing method for two arbitrary meshes with the same topology is introduced with feature correspondence controlled by user. With the assistance of a reference mesh specified by the user, the source and target meshes can be partitioned to establish a common connectivity for vertex-to-vertex correspondences. Lee et al. [146] also focus on the establishment of a correspondence map between the source and target meshes. They employ a multi-resolution parameterization algorithm to generate simplified coarse models, apply the MAPS algorithm to map both meshes over this simple base domain, and then use an additional harmonic map for vertex correspondences. To attain a better control of feature correspondences, Lee and Huang [147] develop a two level of

correspondence technique where the higher level partitions the models into corresponding patches with specified scattered features, and the lower level allows better correspondence control through extra input for features on each patch. They also introduce a technique named Structures of Minimal Contour Coverage (SMCC) to merge corresponding embeddings from the source and target meshes.

Bao and Peng [148] propose a general method for setting up vertex correspondence for polyhedral objects with the same genus. They develop an interactive partitioning algorithm to generate polygonal patches and their correspondence. A cluster scheme is followed to create feature polyhedrons for the input polyhedrons. These feature polyhedrons are utilized as the bridges in the final morphing transformations. Gregory et al. [149] decompose the boundary of the input meshes based on feature pairs specified by the user. The correspondence between sub-meshes of two objects is then established using a greedy area-preserving mapping. The corresponding patch pairs are merged together to create a common topological polyhedron, which is used to define the final morphing trajectories. Yu and Chuang [150] perform a geometric-stretch optimization to generate the consistent parameterizations for multiple input models. Then a foldover-free algorithm is applied to aid in features alignment. Both spatial and wavelet domains from a simple common dissection and remeshing are integrated in the morphing applications. They also claim that their parameterization framework can be applied with other geometric and graphical applications. Another consistent parameterization method from Praun et al. [122] provides a tracing method to dissect a set of meshes automatically.

This method employs the same base domain and respects features of the input shapes. They remesh these models to set up a common connectivity for all, which also form a basis for a large class of other applications. Instead of manually dissecting meshes or setting up feature pairs, Shlafman et al. [151] propose an automatic decomposition with a clustering method. They also introduce a projection framework to deal with polyhedral surfaces with cylinder-like topology. However, their approach cannot guarantee suitable and meaningful corresponding patch creation.

Most morphing related work, as discussed above, has focused on generating vertex correspondences for the input meshes. This requires an equivalent topology for both the source and target meshes. To address the problem of morphing between objects with different topologies, Dinh et al. [157] define the morphing process between two implicit surfaces as a 4D implicit function. They calculate a mapping between two surfaces by solving two PDEs over a tetrahedralized hypersurface. The first PDE depicts a vector field that governs how vertices on one mesh flow to the other. And the second PDE indicates the position labels along this vector field so that the second surface is associated with a position on the first surface. This method can produce correspondence between surfaces with different topologies. Bennett et al. [152] present a robust approach by developing an initial alignment scheme to identify topological holes. From this, they automatically derive a continuous deformation using a variational implicit method. Lee et al. [153] extend the spherical parameterization to handle non genus-zero meshes. The parameterization consists of a single positive spherical

parameterization and several negative spherical parameterizations depending on genus value for the mesh. A Boolean difference operation is applied to calculate the source mesh by subtracting the negative meshes from the positive one. From this approach, they can generate a mesh morphing sequence for meshes with different topologies. Unlike most other approaches, Liu and Wang [154] present a method for shape blending based on their intrinsic definitions, rather than interpolating the vertex locations explicitly. They generalize the algorithm for shape morphing between triangular meshes with arbitrary topologies, and between free-form curves or surfaces.

7.2 Methodology and Implementation

In last section, recent advances in 3D mesh metamorphosis were reviewed. Clearly, most of these methods focus on creating a common connectivity for pairs of shapes with the same genus, and most focus only on genus-zero meshes. These methods usually employ some surface parameterization techniques to assist in finding a bijective mapping between meshes. The generation for correspondence is completed either by user input (in most cases) or some sort of “intelligent” matching and alignment algorithm. Some of the more recent work has attempted to address the problem of morphing models with different topology. These methods attempt to solve the mesh morphing problem in a more general way and are reported to work well.

However, for most of these methods, there are still issues that need to be addressed. The

challenges associated with existing 3D mesh morphing techniques can be categorized as:

For methods that generate common connectivity, i) most of their mesh parameterization algorithms are relatively slow; ii) the final merging representation usually contains many more vertices than the input source meshes. These limitations can lead to expensive calculation and slow down the entire process, especially, if the applications requires blending among more than two input models and needs to perform in real time. The geometric conceptual design application addressed in this research imposes exactly these constraints.

For the methods that address with topological change, the morphing process not only vertex position calculation, but also continuously computes the connectivity between vertices in the mesh. The morphing is no longer a simple blending of vertex positions and it is not feasible for real-time applications.

7.2.1 Morphing Framework

The focus of this research is to enable real time 3D mesh morphing to support engineering geometric conceptual design that makes use of multiple (more than two) existing legacy geometries and generates new design concepts with an interactive blending interface. To achieve real-time metamorphosis, this approach creates a common connectivity among all of the input mesh models. As discussed above, the spherical parameterization algorithm develop here works faster than existing spherical methods. The spherical subdivision remeshing method allows generation of a common connectivity for multiple input models with fewer vertices than most mesh merging methods, while still representing all the

geometric details for each model.

The mesh morphing framework is shown schematically in Figure 20. It includes several different components: spherical mesh parameterization, feature alignment from extracted skeletons, mesh reconstruction from remeshing with spherical subdivision and final mesh morphing. The spherical mesh parameterization, skeleton-based features alignment and mesh reconstruction from remeshing with spherical subdivision have already been introduced in previous chapters. In this chapter, the interactive morphing interface with barycentric coordinates is introduced as well as the software implementation of 3D mesh morphing framework.

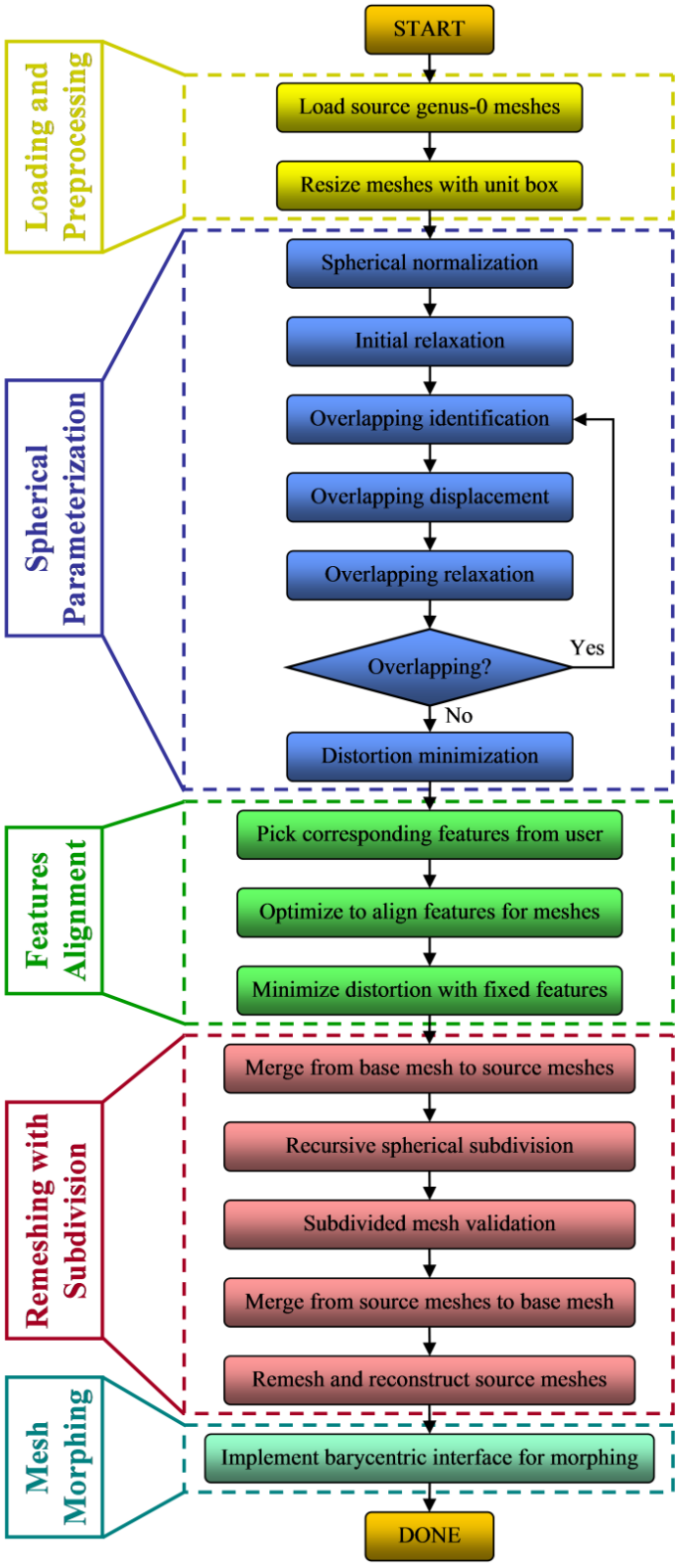


Figure 20: Flowchart of 3D mesh metamorphosis framework

7.2.2 User Interface for Navigation

The idea of mesh morphing in geometric conceptual design is associated with a simple-to-use user interface available to non-engineer users, which eliminates steep learning curve from the complex input with traditional CAD system. To implement such user interface for morphing, the concept of barycentric coordinates is employed which defines the coordinates for any point in a polyhedron in terms of a linear combination of its vertices. Here the formulation from Wachspress' [156] is adopted.

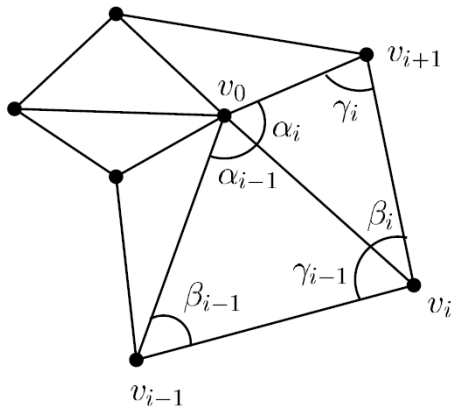


Figure 21: Star-shaped polygon for barycentric coordinates

In the special case that the polygon $(v_1 v_2 \ldots v_k)$ is convex, the coordinates of v_0 can be expressed in terms of rational polynomial, according to Wachspress' formulation,

$$\lambda_i = \frac{w_i}{\sum_{j=1}^{k} w_j}$$

$$w_i = \frac{S(v_{i-1}, v_i, v_{i+1})}{S(v_{i-1}, v_i, v_0) \cdot S(v_i, v_{i+1}, v_0)} = \frac{cot(\gamma_{i-1}) + cot(\beta_i)}{\|v_i - v_0\|^2}$$

Where $S(a, b, c)$is the signed area of triangle abc, γ_i and β_i are the angles as in Figure 21.

7.2.3 Software Development of 3DMeshMorpher

The software is developed with open source GUI API wxWidgets (www.wxwidgets.org) and open source real-time graphics toolkit OpenSceneGraph (www.openscenegraph.org). OpenSceneGraph is employed to organize the scene hierarchy and its rendering pipeline is rewritten in this software to enable more flexible rendering modes and better re-rendering capability for updated geometric information (e.g. new vertex positions). Figure 22 is a snapshot of the 3D mesh metamorphosis framework implementation referred to as "3DMeshMorpher".

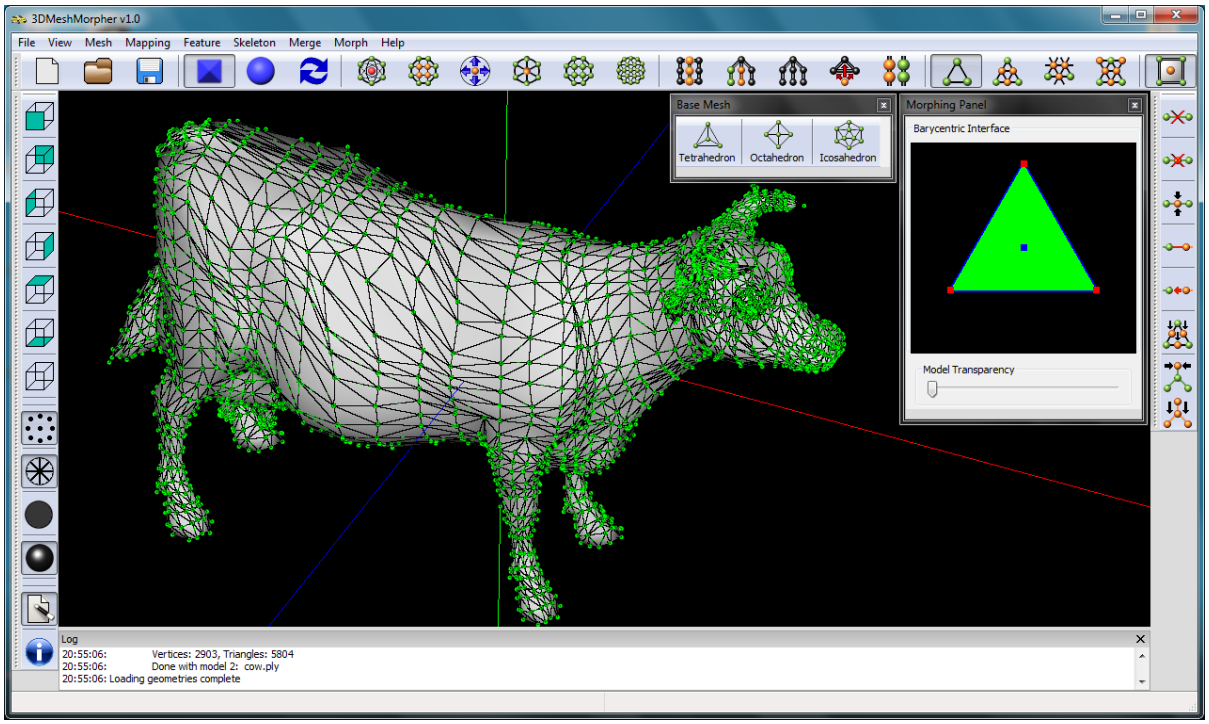


Figure 22: Snapshot of 3D mesh morphing software 3DMeshMorpher

CHAPTER 8

RESULTS AND DISCUSSION

A spherical parameterization method and a spherical subdivision remeshing framework based on it have been described previously. Both algorithms are implemented in C++ and integrated into the "3DMeshMorpher" software, which is designed to assist engineering conceptual design by enabling new geometry generation via morphing among existing 3D models. To test the robustness and effectiveness, several experiments were conducted using various input 3D models. In these experiments, were conducted on a commodity laptop PC with a 2.26 GHz CPU and 1 GB memory. Experimental observations for both the spherical parameterization and subdivision remeshing methods are presented in the following sessions. Finally, results from the developed mesh metamorphosis framework are presented as well.

8.1 Results for Spherical Parameterization

Several recent investigations [132, 130, 92] have employed the Gauss-Seidel iterations in their spherical parameterization methods. However, it is reported that their procedures are shown to be unstable and do not guarantee bijectivity. Alexa [132] applies heuristic iterative procedures with uniform weights and quotes a parameterization time for a mesh with 4,169 triangles at 45.9 seconds. Praun and Hoppe [136] utilize uniformly subdivided polyhedral domains in spherical parameterization and their method requires 7-25 minutes' processing time for 25,000-200,000 faces. Gu et al. [92] report time of around 530 seconds for mapping

30,000 faces with a successful case, and Birkholz's [130] hierarchical method needs about 600 seconds to parameterize roughly 100,000 faces. Saba et al. [135] developed a fast numerical solution which could efficiently solve the non-linear equations for spherical mapping from Gotsman et al. [134]. They report a minimum total solution time of 8.15 seconds for a model with 5,660 triangles.

Performance results from tests of the spherical parameterization method developed in this research, applied to several representative models, is presented in Table 1. Clearly, the algorithm can handle thousands of faces in only a couple of seconds.

Model	Vertices	Triangles	Weights	**Time (sec)**
horse	1929	3854	uniform	**1.4**
armadillo	2164	4324	uniform	**1.7**
triceratops	2832	5660	uniform	**2.4**
cow	2904	5804	uniform	**2.9**

Table 1: Statistics of spherical parameterization efficiencies

8.2 Results for Remeshing with Subdivision

Given spherical parameterizations for the input meshes from the fast spherical parameterization method, the following test demonstrates the performance of the remeshing algorithm with spherical subdivision. Figure 23 shows an original source mesh together with the remeshing results from 3 different base meshes: tetrahedron, octahedron and icosahedron. All of the remeshed representations capture the geometric details from the source mesh fairly well with about same amount of vertices.

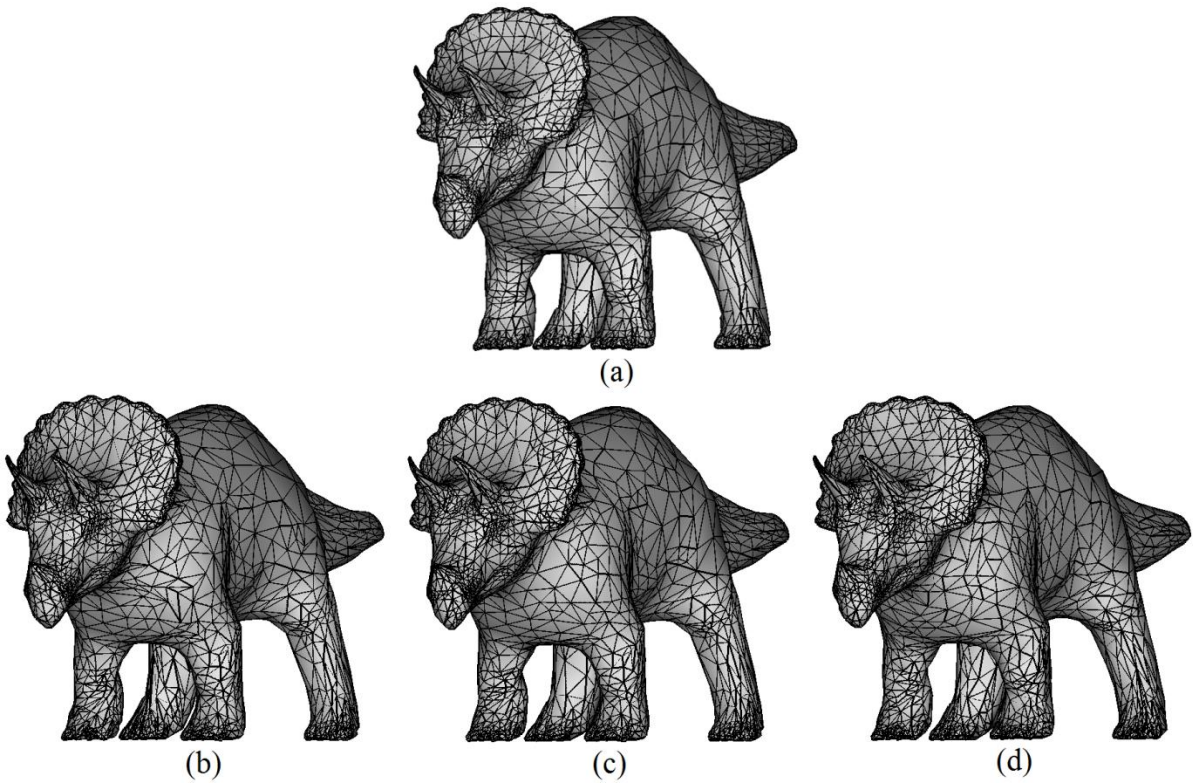


Figure 23: Remeshing from spherical subdivision in seconds: (a) source mesh; (b) remeshing with tetrahedron; (c) remeshing with octahedron; (d) remeshing with icosahedron

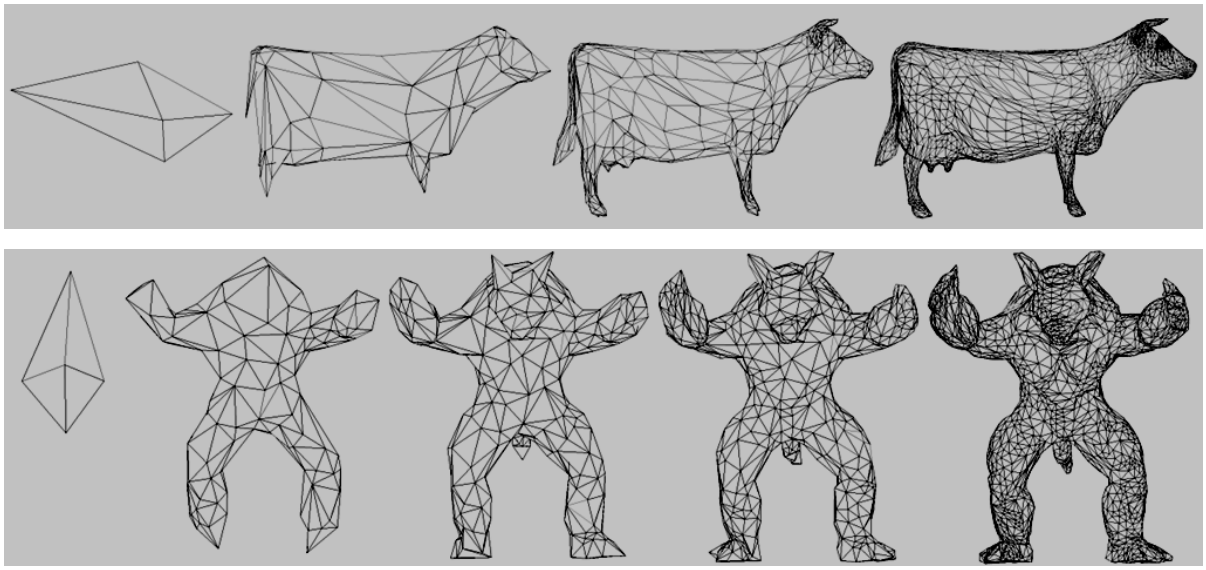


Figure 24: Multi-resolution remeshing outputs from coarse to fine

With the guaranteed bijectivities of the subdivided spherical map and original spherical map, the projection from vertices on the subdivided spherical map to the faces on the original spherical map will be one-on-one. This ensures the validity for this remeshing process. As discussed at the recursive spherical subdivision section, the base mesh is subdivided until it matches some defined LOD parameter. By modifying the LOD setting within the algorithm, a variety of remeshed representations can be computed at different resolutions. This capability of multi-resolution can be applied to aid mesh decimation and mesh refinement related applications. Figure 24 shows two examples, with LOD settings from coarse to fine (the same resolution as the input model), generated with this algorithm.

8.3 Results for 3D Mesh Morphing

In the "3DMeshMorpher" software, the fast spherical parameterization algorithm described in Chapter 4, is integrated with the skeleton-based feature identification and alignment method described in Chapter 5, and the remeshing technique from spherical mesh subdivision described in Chapter 6. An openGL-based navigation interface is introduced to control the barycentric weights for each of the input models, which is defined by Wachspress' formulation for barycentric coordinates. The common connectivity generated for all input models enables the system to generate a new morphed model with accurate geometric information (e.g., normals for every face and every vertex). As users press and drag the mouse button within the blending panel, a continuous sequence of transformed shapes is

created and rendered in real time. Figure 25 and Figure 26 present groups of morphing outputs from 3 input models (horse, triceratops and cow) without and with feature alignment respectively, generated from "3DMeshMorpher". In Figure 26, since we have aligned related features, any intermediate shape shares the same topology as the input models (e.g., four legs instead of eight legs if features had not been aligned or aligned incorrectly as in Figure 25).

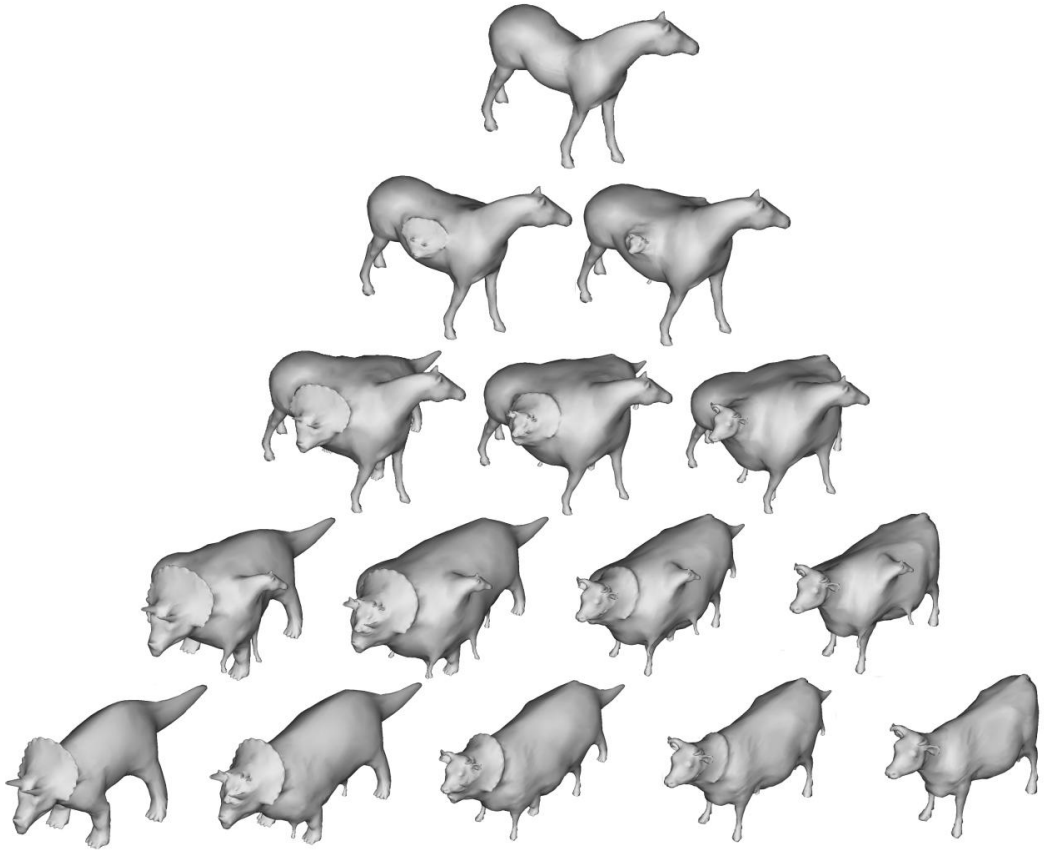


Figure 25: Morphing outputs from three sample models (horse, triceratops and cow) without feature alignment (simply based on their original geometry orientations)

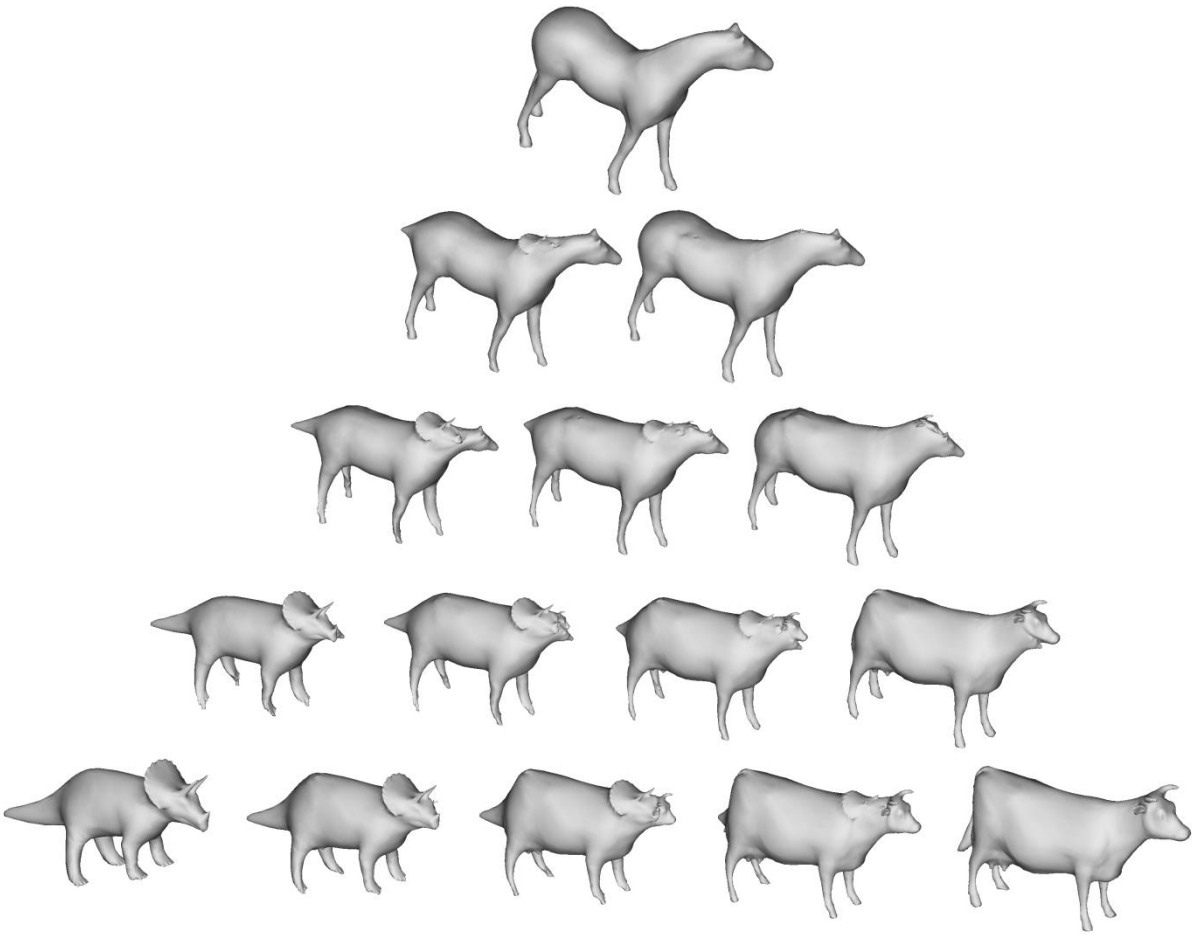


Figure 26: Morphing outputs from three sample models (horse, triceratops and cow) with feature alignment for four legs and head

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

Geometry creation and visualization during the conceptual design phase of product development is a challenging task. This work focuses on the development of tools that can assist an engineer in the generation of geometric concept models by rapidly and continuously morphing multiple legacy models.

To quickly construct new geometric model from existing ones, a 3D mesh metamorphosis framework "3DMeshMorpher" is implemented. To support the software framework, the following unique algorithms were developed: a fast spherical parameterization method to map a (genus-zero) geometric model onto a unit sphere; a geometric feature identification and alignment technique based on 3D skeleton extraction; and a LOD controllable 3D remeshing scheme with spherical mesh subdivision (based on the same spherical parameterization algorithm). This software allows designers to continuously navigate through the shape-space of design models bounded by existing models to produce numerous design concepts in real-time.

9.2 Future Work

A 3D mesh metamorphosis software framework has been presented to aid engineering conceptual design. Several related techniques and methods were developed and integrated in

this framework to facilitate real-time mesh morphing. The validity and effectiveness of these methods have been verified by the testing results obtained from sample models. However, the following possible investigation and development could be conducted to improve the efficiency and usability of this framework, as future work,

1) Improve the ‘overlapping relaxation’ in our spherical parameterization method. Current iterative relaxation is not effective enough for big geometric models with dense overlapping area. This could be improve by formulating and solving discrete Laplace equation systems. There exists some efficient matrix solving algorithm that could be employed to fulfill this purpose.
2) Investigate feasibility of automatic feature alignment with skeleton extraction. This could be beneficial when the input models are similar to each other topologically. A pre-designed template will help, but a more promising solution is to develop/implement a generic feature recognition algorithm.
3) Implement different user morphing interface. Instead of a barycentric graphical interface for engineer, navigation through metadata could bring people from non-engineering background to the design phase. For example, a slider bar of weight, gas mileage or sportiness for designing a new vehicle from existing automobiles with this mesh morphing framework.
4) Realize mesh morphing in virtual reality by integrating morphing framework into ASDS. An immersive virtual reality environment could influence designers'

perception and inspire their ideas for new concepts creation. And the collaborative context of ASDS will also enable group design and the interactions among people will definitely improve the efficiency and effectiveness of the design process.

REFERENCES

[1] Lotter, B., Manufacturing Assembly Handbook, Butterworths, Boston, MA, 1986.

[2] Ulrich, K. T. and Eppinger, S. D., Product design and development, 3rd edition, McGraw Hill, 2004.

[3] SIMULIA Abaqus, http://www.simulia.com, accessed September 2009.

[4] Solidworks, http://www.solidworks.com, accessed September 2009.

[5] UGS Teamcenter, http://www.plm.automation.siemens.com/en_us/products/teamcenter/index.shtml, accessed September 2009.

[6] Ullman, David G., Mechanical Design Process, 3rd edition, McGraw-Hill, New York, 2003.

[7] Otto, Kevin N. and Kristin L. Wood, Product Design:Techniques in Reverse Engineering and New Product Development, Upper Saddle River, NJ: Prentice Hall, 2001.

[8] PTC: Pro/CONCEPT, http://www.ptc.com/appserver/mkt/products/home.jsp?&k=701, accessed September 2009.

[9] Dassault Systemes: CATIA V6 PLM, http://www.3ds.com/products/v6/welcome/, accessed September 2009.

[10] Dieter, G. E. and Schmidt, L. C., Engineering Design, 4th Edition, McGraw-Hill, New York, 2009.

[11] Wang, L., Shen, W., Xie, H., Neelamkavil, J., and Pardasani, A., Collaborative Conceptual Design—State of the Art and Future Trends, Journal of Computer-Aided Design, Vol. 34, No. 13, pp. 981-996, 2002.

[12] Sahin, A., Studd, A., and Terpenny, J. P., A Graphical Modeling Tool for Conceptualizing and Analyzing Modular Products, International Design Research Symposium, South Korea, 2006.

[13] Chang, X., Sahin, A., and Terpenny, J. P., An Ontology-Based Support for Product

Conceptual Design, Journal of Robotics and Computer-Integrated Manufacturing, Vol. 24, No. 6, pp. 755-762, Sp. Iss., December 2008.

[14] Cao, D., Ramani, K., Fu, M., and Zhang, R., Port-Based Ontology Semantic Similarities for Module Concept Creation, ASME International Design Engineering Technical Conferences 35th Design Automation Conference, paper no. DETC2009-86470, Aug 30-Sept 2, 2009.

[15] Christophe, F., Sell, R., Bernard, A., and Coatenéa, E., OPAS: Ontology Processing for Assisted Synthesis of Conceptual Design Solutions, ASME International Design Engineering Technical Conferences 35th Design Automation Conference, paper no. DETC2009-87776, Aug 30-Sept 2, 2009.

[16] Takahashi, T. T., Fanciullo, T., and Ridgely, D. B., Incorporation of Flight Control Design Tools into the Multi-Disciplinary Conceptual Design Process, 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan 8-11, 2007.

[17] Noon, C. J. and Winer, E. H., A Study of Different Metamodeling Techniques for Conceptual Design, ASME International Design Engineering Technical Conferences 35th Design Automation Conference, paper no. DETC2009-86496, Aug 30-Sept 2, 2009.

[18] Suh, N. P., Axiomatic Design—Advances and Applications, Oxford University Press, New York, 2001.

[19] Hazelrigg, G., Systems engineering: An approach design, Prentice-Hall, Upper Saddle River, NJ, 1996.

[20] Hauser, J. R. and Clausing, D., The House of Quality, Harvard Business Review, Vol. 5, 1988.

[21] Keeney, R. L. and Raiffa, H., Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, Inc., New York, 1976.

[22] Keeney, R. L., Value-Focused Thinking— A Path to Creative Decision Making, Harvard University Press, 1992.

[23] Jin, Y., Kim, D. and Danesh M. R., Value based design: an objective structuring approach to design concept generation, ASME Proceedings of IDETC/CIE, Philadelphia, PA, 2006.

[24] Danesh M. R. and Jin, Y., An Agent-Based Decision Network for Concurrent Engineering Design, Journal of Concurrent Engineering: Research and Applications, Vol. 9, No. 1, 2001.

[25] Hoyle, C. J., and Chen, W., Product Attribute Function Deployment (PAFD) for Decision-Based Conceptual Design, Journal of IEEE Transactions on Engineering Management, Vol. 56, No. 2, May 2009.

[26] Sutherland, I. E., The Ultimate Display, Proceedings of IFIPS Congress, New York City, NY, Vol. 2, pp. 506-508, May 1965.

[27] Cruz-Neira, Carolina, Sandin, Daniel J., and DeFanti, Thomas A., Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE, Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 135-142, September 1993.

[28] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., Hart, J. C., The CAVE: Audio Visual Experience Automatic Virtual Environment, Communications of the ACM, ACM, Vol. 35, No. 6, pp. 64-72, June 1992.

[29] Sony SRXT110 Projector: http://pro.sony.com/bbsc/ssr/product-SRXT110/, accessed September 2009.

[30] Kindratenko, V., A Comparison of the Accuracy of an Electromagnetic and a Hybrid Ultrasound-inertia Position Tracking System, Presence: Teleoperators and Virtual Environments, MIT Press, Cambridge, MA, Vol. 10, Issue 6, pp. 657-663, December 2001.

[31] InterSense: http://www.intersense.com/, accessed September 2009.

[32] Burdea, G. G. and Coiffet, P., Virtual reality technology, Wiley-Interscience, 2003.

[33] Bordegoni, M. and Cugini, U., A conceptual design tool based on innovative haptic interfaces, ASME Proceedings of IDETC/CIE, Philadelphia, PA, 2006.

[34] Fischer, A. and Vance, J. M., PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment, Proceedings of the Workshop on Virtual Environments, EUROGRAPH, Zurich, Switzerland, 2003.

[35] Duncan, T. J. and Vance, J. M., Development of a Virtual Environment for Interactive Interrogation of Computational Mixing Data, Journal of Mechanical Design, Vol. 129, No.

361, 2007.

[36] Abdul-Jalil, M. K. And Bloebaum, C. L., Development of a Distributed Collaborative Virtual Environment for Engineering Design Application, 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, Sept 6-8, 2000.

[37] Zhang, R., Noon, C., Oliver, J., Winer, E., Gilmore, B., and Duncan, J., Development of a Software Framework for Conceptual Design of Complex Systems, 3rd Annual AIAA Multidisciplinary Design Optimization Specialists Conference, paper no. AIAA-2007-1931, April 23-26, 2007.

[38] Zhang, R., Noon, C., Oliver, J., Winer, E., Gilmore, B., and Duncan, J., Immersive Product Configurator for Conceptual Design, ASME International Design Engineering Technical Conferences 33rd Design Automation Conference, paper no. DETC2007-35390, September 4-7, 2007.

[39] OpenSceneGraph, http://www.openscenegraph.org/projects/osg, accessed September 2009.

[40] wxWidgets, http://www.wxwidgets.org/, accessed September 2009.

[41] VR Juggler, http://www.vrjuggler.org, accessed September 2009.

[42] A. Sheffer, E. Praun, and K. Rose, Mesh parameterization methods and their applications, Foundations and Trends in Computer Graphics and Vision, 2(2):105-171, 2006.

[43] BENNIS, C., VÉZIEN, J.-M., AND IGLÉSIAS, G., Piecewise surface flattening for nondistorted texture mapping, ACM SIGGRAPH, pp.237-246, 1991.

[44] FLOATER, M. AND HORMANN, K., Surface Parameterization: a Tutorial and Survey, Advances in Multiresolution for Geometric Modelling, pp.157-186, 2005.

[45] MAILLOT, J., YAHIA, H., AND VERROUST, A., Interactive texture mapping, ACM SIGGRAPH, pp. 27-34, 1993.

[46] HORMANN, K., AND GREINER, G., MIPS: An efficient global parameterization method, Curve and Surface Design, pp.153-162, 2000.

[47] SHEFFER, A., AND DE STURLER, E., Surface Parameterization for Meshing by

Triangulation Flattening, Proc. 9th International Meshing Roundtable, pp.161-172, 2000.

[48] LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J., Least squares conformal maps for automatic texture atlas generation, ACM SIGGRAPH, pp.362-371, 2002.

[49] DESBRUN, M., MEYER, M., AND ALLIEZ, P., Intrinsic parameterizations of surface meshes, Computer Graphics Forum, 21, pp.209-218, 2002.

[50] DEGENER, P., MESETH, J., AND KLEIN, R., An adaptable surface parameterization method, Proceedings, 12th International Meshing Roundtable, pp. 201-213, 2003.

[51] Ed Catmull, A Subdivision Algorithm for Computer Display of Curved Surfaces, PhD thesis, Dept. of CS, U. of Utah, 1974.

[52] James F. Blinn, Martin E. Newell, Texture and Reflection in Computer Generated Images, CACM, Vol. 19, No. 10, pp. 542-547, 1976.

[53] James F. Blinn, Simulation of Wrinkled Surfaces, Computer Graphics, (SIGGRAPH ’78 Proceedings), Vol. 12, No. 3, pp. 286-292, 1978.

[54] James F. Blinn, Computer Display of Curved Surfaces, PhD thesis, CS Dept., U. of Utah, 1978.

[55] Geoffrey Y. Gardner, Visual Simulation of Clouds, Computer Graphics, (SIGGRAPH ’85 Proceedings), Vol. 19, No. 3, pp. 297-303, 1985.

[56] Gene S. Miller, C. Robert Hoffman, Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments, SIGGRAPH Advanced Computer Graphics Animation seminar notes, 1984.

[57] Robert L. Cook, Shade Trees, Computer Graphics, (SIGGRAPH ’84 Proceedings), Vol. 18, No. 3, pp. 223-231, 1984.

[58] James T. Kajiya, Anisotropic Reflection Models, Computer Graphics, (SIGGRAPH ’85 Proceedings), Vol. 19, No. 3, pp.15-21, 1985.

[59] SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A., ABF++: fast and robust angle based flattening, ACM Transactions on Graphics 24(2), pp.311-330, 2005.

[60] LENGYEL, J., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H., Real-time fur on arbitrary surfaces, Symposium on Interactive 3D Graphics, pp.227-232, 2001.

[61] PENG, J., KRISTJANSSON, D., AND ZORIN, D., Interactive modeling of Topologically Complex Geometric Detail, ACM SIGGRAPH, 2004.

[62] PORUMBESCU, S, BUDGE, B., FENG, L., AND JOY, K. I., Shell Maps, ACM SIGGRAP, 2005.

[63] PEDERSON, H.-K., Decorating Implicit Surfaces, ACM SIGGRAPH, 1995.

[64] PRAUN, E., FINKELSTEIN, A., AND HOPPE, H., Lapped Textures, ACM SIGGRAPH, pp.465-470., 2000.

[65] WEI, L.-Y., AND LEVOY, M., Texture Synthesis over Arbitrary Manifold Surfaces, ACM SIGGRAPH, 2001.

[66] TURK, G., Texture synthesis on surfaces, ACM SIGGRAPH, pp. 347-354, 2001.

[67] YING, L., HERTZMANN, A., BIERMANN, H., ZORIN, D., Texture and Shape Synthesis on Surfaces, Eurographics Workshop on Rendering, 2001.

[68] SOLER, C., CANI, M.-P., ANGELIDIS, A., Hierarchical Pattern Mapping, ACM SIGGRAPH, pp 673-680, 2002.

[69] IGARASHI, T., AND COSGROVE, D., Adaptive Unwrapping for Interactive Texture Painting, ACM Symposium on Interactive 3D Graphics, pp. 209-216, 2001.

[70] CARR, N., AND HART, J., Painting Detail, ACM SIGGRAPH, 2004.

[71] LÉVY, B., Dual domain extrapolation, ACM SIGGRAPH, 2003.

[72] ALLEN, B., CURLESS, B., AND POPOVIĆ, Z., The space of human body shapes: reconstruction and parameterization from range scans, ACM SIGGRAPH, pp.587-594, 2003.

[73] ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J. DAVIS, J., SCAPE: Shape Completion and Animation of People, ACM SIGGRAPH, 2005.

[74] KRAEVOY, V., AND SHEFFER, A., Template Based Mesh Completion, Proc. Symposium on Geometry Processing (SGP), 2005.

[75] BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D., Cut-and-paste editing of multiresolution surfaces, ACM SIGGRAPH, pp.312-321, 2002.

[76] SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P., Laplacian Surface Editing, Proceedings of Eurographics/ACM Symposium on Geometry Processing, pp.179-188, 2004.

[77] GU, X., GORTLER, S., AND HOPPE, H., Geometry images, ACM SIGGRAPH, pp.356-361, 2002.

[78] GUSKOV, I., VIDIMČE, K., SWELDENS, W., AND SCHRÖDER, P., Normal meshes, ACM SIGGRAPH, pp.95-102, 2000.

[79] LEE, A., HOPPE, H., MORETON, H., Displaced Subdivision Surfaces, ACM SIGGRAPH, 2000.

[80] KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P., Globally smooth parameterizations with low distortion, ACM SIGGRAPH, pp.350-357, 2003.

[81] SURAZHKY, V. AND GOTSMAN, C., Explicit surface remeshing, ACM/Eurographics Symposium on Geometry Processing, 2003.

[82] RAY, N., LI, W-C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P., Periodic global parameterization, ACM Transactions on Graphic, 25(3), 2006.

[83] DONG, S., BREMER, P.-T., GARLAND, M. , PASCUCCI, V., AND HART, J. C., Spectral Surface Quadrangulation, ACM SIGGRAPH, 2006.

[84] ALLIEZ, P., AND GOTSMAN, C., Recent advances in compression of 3D meshes, Advances in Multiresolution for Geometric Modelling, pp.3-26, 2005.

[85] HOPPE, H., AND PRAUN, E., Shape compression using spherical geometry images, Advances in multiresolution for geometric modeling, pp.3-26, 2005.

[86] BLANZ, V., AND VETTER, T., A morphable model for the synthesis of 3D faces, ACM SIGGRAPH, 1999.

[87] MARSCHNER, S., GUENTER, B., AND RAGUPATHY, S., Modeling and rendering for realistic facial animation, Proceedings of the Eurographics Workshop on Rendering Techniques, pp.231-242, 2000.

[88] BLANZ, V., BASSO, C., POGGIO, T. AND VETTER, T., Reanimating Faces in Images and Video, Computer Graphics Forum 22(3), EUROGRAPHICS, pp.641-650, 2003.

[89] BLANZ, V., SCHERBAUM, K., VETTER, T., AND SEIDEL, H.-P., Exchanging faces in images, EUROGRAPHICS, 2004.

[90] HURDAL, M. K., BOWERS, P. L., STEPHENSON, K., SUMNERS, D. W. L., REHM, K., SCHAPER, K., ROTTENBERG, D. A., Quasi-conformally flat mapping the human cerebellum, Medical Image Computing and Computer-Assisted Intervention 1679, pp.279-286, 1999.

[91] HAKER, S., ANGENENT, S., TANNENBAUM, S., KIKINIS, R., SAPIRO, G., AND HALLE, M., Conformal surface parameterization for texture mapping, IEEE TVCG, 6(2), pp.181-189, 2000.

[92] GU, X., WANG, Y., CHAN, T. F., THOMPSON, P. M., AND YAU, S.-T., Genus Zero Surface Conformal Mapping and Its Application to Brain Surface Mapping, IEEE Transaction on Medical Imaging 23(7), 2004.

[93] MITANI J., AND SUZUKI H., Making papercraft toys from meshes using strip-based approximate unfolding, ACM SIGGRAPH, pp.259-263, 2004.

[94] JULIUS D., KRAEVOY, AND SHEFFER, A., D-charts: Quasi-developable mesh segmentation, Proceedings of Eurographics, 24(3), pp.981-990, 2005.

[95] DO CARMO, M., Differential geometry of curves and surfaces, Prentice Hall, 1976.

[96] TUTTE, W., T., How to draw a graph, London Mathematical Society 1963, 13, pp.743-768, 1963.

[97] FLOATER, M., Parameterization and smooth approximation of surface triangulations, Computer Aided Geometric Design, 14(3), pp. 231-250, 1997.

[98] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W., Multiresolution analysis of arbitrary meshes, ACM SIGGRAPH, pp. 173-182, 1995.

[99] KHAREVYCH, L., SPRINGBORN, B., AND SCHRÖDER, P., Discrete conformal mappings via circle patterns, ACM Transactions on Graphics 25(2), 2006.

[100] GUSKOV, I., An anisotropic mesh parameterization scheme, Proceedings of the 11th International Meshing Roundtable, pp.325-332, 2002.

[101] FLOATER, M., Mean value coordinates. Computer Aided Geometric Design, 20(1), pp.19-27, 2003.

[102] LEE, Y., KIM, H. S., AND LEE, S., Mesh parameterization with a virtual boundary, Computers and Graphics 26(5), pp.677-686, 2002.

[103] ZHANG, E., MISCHAIKOW, AND TURK, G., Feature-Based Surface Parameterization and Texture Mapping, ACM Transaction on Graphics 24(1), pp.1-27, 2005.

[104] SHEFFER, A. AND DE STURLER, E., Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening, Engineering with Computers 17(3), pp.326-337, 2001.

[105] ZAYER, R., ROSSL, C., AND SEIDEL, H.-P., Variations on angle based flattening, Proceedings of Multiresolution in Geometric Modelling, pp.285-296, 2003.

[106] ZAYER, R., RÖSSL, C., AND SEIDEL, H.-P., Setting the Boundary Free: A Composite Approach to Surface Parameterization, Symposium on Geometry Processing, pp.91-100, 2005.

[107] LÉVY, B. AND MALLET, J.-L., Non-distorted texture mapping for sheared triangulated meshes, ACM SIGGRAPH, pp.343-352, 1998.

[108] SANDER, P., SNYDER, J., GORTLER, S., AND HOPPE, H., Texture mapping progressive meshes, ACM SIGGRAPH, pp.409-416, 2001.

[109] ZHOU, K., SNYDER, J., GUO, B., AND SHUM, H.-Y., Iso-charts: Stretch-driven Mesh Parameterization using Spectral Analysis, Eurographics Symposium on Geometry Processing, pp.47-56, 2004.

[110] SANDER, P., GORTLER, S., SNYDER, J., AND HOPPE, H., Signal-specialized parameterization, Eurographics Workshop on Rendering, pp.87-100, 2002.

[111] TEWARI, G., SNYDER, J., SANDER, P., GORTLER, S., AND HOPPE, H., Signal-Specialized Parameterization for Piecewise Linear Reconstruction, Eurographics Symposium on Geometry Processing, pp.57-66, 2004.

[112] SANDER, P., WOOD, Z., GORTLER, S., SNYDER, J. AND HOPPE, H., Multi-chart geometry images, ACM Symposium on Geometry Processing, 2003.

[113] GU, X., AND YAU, S.-T., Global conformal surface parameterization, Symposium on Geometry Processing, pp.127-137, 2003.

[114] TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C., PolyCube-Maps, ACM SIGGRAPH, pp.853-860, 2004.

[115] SHEFFER, A., AND HART, J., Seamster: Inconspicuous low-distortion texture seam layout, IEEE Visualization, pp.291-298, 2002.

[116] SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D., Bounded-distortion piecewise mesh parameterization, IEEE Visualization, pp.355-362, 2002.

[117] LAZARUS, F., POCCHIOLA, M., VEGTER, G., AND VERROUST, A., Computing a canonical polygonal schema of an orientable triangulated surface, In Proceedings of the Seventeenth Annual Symposium on Computational Geometry, 2001.

[118] ERICKSON, J. AND HAR-PELED, S., Optimally Cutting a Surface into a Disk, Discrete Computational Geometry 31(1), pp.37-59, 2004.

[119] NI, X., GARLAND, M., AND HART, J. C., Fair Morse functions for extracting the topological structure of a surface mesh, ACM SIGGRAPH, pp.613-622, 2004.

[120] EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A., Topological persistence and simplification, Discrete and Computational Geometry 28, 4, pp.511-533, 2002.

[121] GUSKOV, I., KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W., Hybrid meshes: multiresolution using regular and irregular refinement, ACM Symposium on Computational Geometry, pp.264-272, 2002.

[122] PRAUN, E., SWELDENS, W. AND SCHRÖDER, P., Consistent mesh parameterizations, ACM SIGGRAPH, pp.179-184, 2001.

[123] SCHREINER, J., ASIRVATHAM, A., PRAUN, E, AND HOPPE, H., Inter-Surface Mapping, ACM SIGGRAPH, 2004.

[124] KRAEVOY, V., AND SHEFFER, A., Cross-parameterization and compatible

remeshing of 3D models, ACM SIGGRAPH, 2004.

[125] BOIER-MARTIN, I., RUSHMEIER, H., JIN, J., Parameterization of Triangle Meshes over Quadrilateral Domains, Eurographics Symposium on Geometry Processing, pp.197-208, 2004.

[126] ISENBURG, M. GUMHOLD, S. AND GOTSMAN, C., Connectivity Shapes, Proceedings of IEEE Visualization, pp.135-142, 2001.

[127] ZAYER, R., RÖSSL, C., AND SEIDEL, H.-P., Curvilinear Spherical Parameterization, Proceedings of Shape Modeling and Applications, pp.57-64, 2006.

[128] DAS, G., AND GOODRICH, M. T., On the Complexity of Optimization Problems for 3-Dimensional Convex Polyhedra and Decision Trees, Computational Geometry, 8, pp.123-137, 1997.

[129] SHAPIRO, A. AND TAL, A., Polyhedron realization for shape transformation, The Visual Computer14 (8), pp.429-444, 1998.

[130] BIRKHOLZ. H., Shape-preserving parameterization of genus 0 surfaces, Proc. Winter Conference on Computer Graphics (WSCG), 2004.

[131] KOBBELT, L. P., VORSATZ, J., LABISK, U. AND SEIDEL, H.-P., A shrink-wrapping approach to remeshing polygonal surfaces, Proceedings of Eurographics, 1999.

[132] ALEXA, M., Merging polyhedral shapes with scattered features, The Visual Computer, 16(1), pp.26-37, 2000.

[133] ALEXA, M., Recent advances in mesh morphing, Computer Graphics Forum 21(2), pp.173-196, 2002.

[134] GOTSMAN, C., GU, X. AND SHEFFER, A., Fundamentals of spherical parameterization for 3D meshes, ACM SIGGRAPH, pp.358-364, 2003.

[135] SABA, S., YAVNEH, I., GOTSMAN, C. AND SHEFFER, A., Practical Spherical Embedding of Manifold Triangle Meshes, Proceedings of Shape Modeling International, 2005.

[136] PRAUN, E. AND HOPPE, H., Spherical parameterization and remeshing, ACM SIGGRAPH, pp.340-350, 2003.

[137] Sheffer, A., Gotsman, C., Dyn, N., Robust spherical parameterization of triangular meshes. Computing, 72(1–2): 185–193, 2003.

[138] Wolberg, G., Recent Advances in Image Morphing, Proceedings of the Conference on Computer Graphics International, pp 64, 1996.

[139] Daniel Cohen-Or, Amira Solomovici, and David Levin, Three dimensional distance field metamorphosis, ACM Transactions on Graphics, 17(2):116–141, 1998.

[140] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy, Feature based volume metamorphosis, Computer Graphics, 29:449–456, 1995.

[141] Xiang Fang, Hujun Bao, Pheng-Ann Heng, Tien-Tsin Wong, and Qunsheng Peng, Continuous field based free-form surface modeling and morphing, Computer and Graphics, 25(2):235–243, 2001.

[142] Greg Turk and James F. O'Brien, Shape transformation using variational implicit functions, In Proceedings of ACM SIGGRAPH 1999, pages 335–342, 1999.

[143] J.R. Kent,W.E. Carlson, and R.E. Parent, Shape transformation for polyhedral objects. Computer Graphics, 26(2):47.54, July 1992.

[144] Kanai, T., Suzuki, H. and Kimura, F., Three-Dimensional Geometric Metamorphosis Based on Harmonic Maps, The Visual Computer, vol. 14, pp 166-176, 1998.

[145] Kanai, T., Suzuki, H. and Kimura, F., Metamorphosis of Arbitrary Triangular Meshes, IEEE Computer Graphics and Applications, pp 62-75, 2000.

[146] Lee, A., Dobkin, D., Sweldens, W. and Schroder, P., Multiresolution Mesh Morphing, Proceedings of SIGGRAPH 99, pp 343-350, 1999.

[147] Lee, T. and Huang, P., Fast and Intuitive Metamorphosis of 3D Polyhedral Models Using SMCC Mesh Merging Scheme, IEEE Transactions on Visualization and Computer Graphics, vol. 9, no. 1, 2003.

[148] Hujun Bao and Qunsheng Peng, Interactive 3d morphing, Computer Graphics Forum, 17(3):23–30, 1998.

[149] Arthur Gregory, Andrei State, Ming C. Lin, Dinesh Manocha, and Mark A. Livingston, Feature-based surface decomposition for correspondence, Computer Animation 98, pages

64–71, 1998.

[150] Jin-Bey Yu and Jung-Hong Chuang, Consistent mesh parameterizations and its application in mesh morphing, Computer Graphics Workshop, 2003.

[151] Shlafman, S., Tal, A. and Katz, S., Metamorphosis of Polyhedral Surfaces Using Decomposition, EUROGRAPHICS, vol. 21, no. 3, pp 219-228, 2002.

[152] Janine Bennett, Valerio Pascucci, Kenneth Joy, A genus oblivious approach to cross parameterization, Computer Aided Geometric Design, vol. 25, no. 8, pp 592-606, 2008.

[153] Lee, T.-Y., Yao, C.-Y., Chu, H.-K., Tai, M.-J., Chen, C.-C., Generating genus-n-to-m mesh morphing using spherical parameterization, Journal of Visualization and Computer Animation, 17 (3–4), 433–443, 2006.

[154] L.-G. Liu, G.-J. Wang, Three-dimensional shape blending: intrinsic solutions to spatial interpolation problems, Comput. Graph., 23 (4) 535-545, 1999.

[155] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Trans. Graphics, vol. 27, no. 3, pp. 44:1–44:10, 2008.

[156] E. Wachspress, A rational finite element basis, Academic Press, 1975.

[157] Huong Quynh Dinh, Anthony Yezzi, Greg Turk, Texture transfer during shape transformation, ACM Transactions on Graphics (TOG), v.24 n.2, pp 289-310, 2005.

ACKNOWLEDGEMENTS

I will keep the diploma but many share the credit for this work.

First of all, I would like to thank my parents, without whom I would not be what I am today, my wife for her constant encouragement and the one person who has been my strongest source of strength and inspiration.

I would like to thank my major professors Eliot Winer and James Oliver, who have always given me all the support and affirmation of confidence that a graduate student could ask for. I really appreciate their patience and instructions over the past years. From them I learned the basics of graduate research and how to probe things deeply. And they are always a constant source of inspiration and wisdom.

Here at Iowa State University, I'd like thank Professors Judy Vance, Song Zhang and Chris Harding for being my thesis readers. I must thank my colleagues at Virtual Reality Applications Center, who have been giving me support and assistance. I want to also thank my colleagues Dr. Alex Zhou and Dr. Grace Chen at CFDRC for helping in implementation and debugging. I am greatly thankful for their assistance and suggestion for my research work.

For their support and sacrifice, I dedicate this work to them.