4-2024

Hierarchical Parameterization in Spherical Domains for Deforming Feature Alignment

Lizhou Cao
lc1248@rit.edu

Hierarchical Parameterization in Spherical Domains for Deforming Feature Alignment

by

Lizhou Cao

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
April 2024

Hierarchical Parameterization in Spherical Domains for Deforming Feature Alignment

by

Lizhou Cao

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Chao Peng — Date
Advisor

Dr. Joe Geigel — Date
Dissertation Committee Member

Dr. David Long — Date
Dissertation Committee Member

Dr. David Schwartz — Date
Dissertation Committee Member

Dr. Yunbo Zhang — Date
Dissertation Committee Member

Dr. David Long — Date
Dissertation Defense Chairperson

Certified by:

Dr. Pengcheng Shi — Date
Ph.D. Program Director, Computing and Information Sciences

Hierarchical Parameterization in Spherical Domains for Deforming Feature Alignment

by

Lizhou Cao

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

The growing interest in immersive technology and lifelike virtual worlds has underscored the importance of creating automated methods to generate characters with a wide range of morphological variations. Traditional methods have either depended heavily on manual effort or simplified the challenge by only addressing static meshes or basic deformation transfers, without supporting dynamic shape morphing with deformation. In this work, we introduce a novel method that leverages cross-parameterization to semi-automate the creation of characters that are not only morphologically varied but also capable of synthesized deformation and animations. The key innovation of our work lies in the parameterization of deforming characters into a hierarchical spherical domain. This domain is methodically constructed to encapsulate the deforming features, ensuring that the hierarchical relationships among these features are preserved within this data structure. This approach takes into account the characteristics of mesh topology, deformation, and animation, thereby reducing parametric distortion, improving the bijectivity of the parameterization process, and ensuring better alignment quality of deforming features. Our alignment algorithm simplifies the process by concentrating on principal joint pairs, making it significantly easier and more intuitive than previous methods that required the manual pinpointing of feature points on meshes. Our method stands out by delivering high-quality outcomes in 3D morphing, texture mapping, character creation, and deformation transfer, marking a significant advancement over recent developments in the field. These results have the potential to significantly decrease the workload associated with asset generation and enhance visual diversity across various domains, including film, animation, virtual societies, and interactive entertainment.

Acknowledgments

I am grateful to my advisor, Dr. Chao Peng, for his exceptional guidance, unwavering support, and invaluable mentorship throughout my doctoral studies. His expertise and encouragement have been instrumental in shaping my research journey and in the completion of this thesis. I am truly fortunate to have had the opportunity to learn from him.

I extend my heartfelt appreciation to my dissertation committee members, Dr. Joe Geigel, Dr. David Long, Dr. David Schwartz, and Dr. Yunbo Zhang, for their insightful feedback, constructive criticism, and dedication to advancing my scholarly endeavors. Their expertise and encouragement have enriched my work immeasurably. I'd like to express my gratitude to Dr. Pengcheng Shi, the director of the Ph.D. program, for his invaluable support.

I am also deeply thankful to the MAGIC Spell Studio, the School of Interactive Games and Media, and the Golisano College of Computing and Information Sciences for generously providing me with workspace and research resources. Their support has been instrumental in the advancement of my research endeavors.

Last but not least, to my family and friends, whose unwavering love, encouragement, and understanding have sustained me throughout this journey, I am profoundly grateful. Your support has been my greatest source of strength.

To my family and friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

1.1.1 Geometry and Deforming Characters

Geometry serves as the foundational data of the digital world, enabling the seamless integration of 3D meshes, animations, and textures to construct intricate representations. This synthesis enriches the digital landscape, infusing it with diversity and vitality. Across a spectrum of industries, including video games, animations, and movies, these elements play an important role in captivating audiences and conveying complex narratives [38, 64]. Moreover, their significance extends far beyond mere entertainment. In domains such as scientific visualization, simulations, educational tools, and virtual training environments, the utilization of geometric data is essential for understanding, innovation, and progress in the increasingly digital society [19, 20, 21].

In geometry processing, the characters with animations are called deforming characters. A deforming character is a complex structure, mainly made up of a mesh and a skeleton, which together enable the character's animation in a digital setting [38, 64, 104]. The mesh defines the character's external 3D shape, featuring an intricate topology that arranges vertices, edges, and faces to shape the character's surface. This topology is crucial as it determines the mesh's flexibility and how well it can preserve the visual effect through deformation. Conversely, the skeleton serves as an internal framework, similar to an organism's bones, manipulating the mesh in specific regions to generate poses and movements. The skeletal structure is connected to the mesh via skinning, a process that assigns weights to each vertex to govern the bones' influence on mesh deformation. An example of

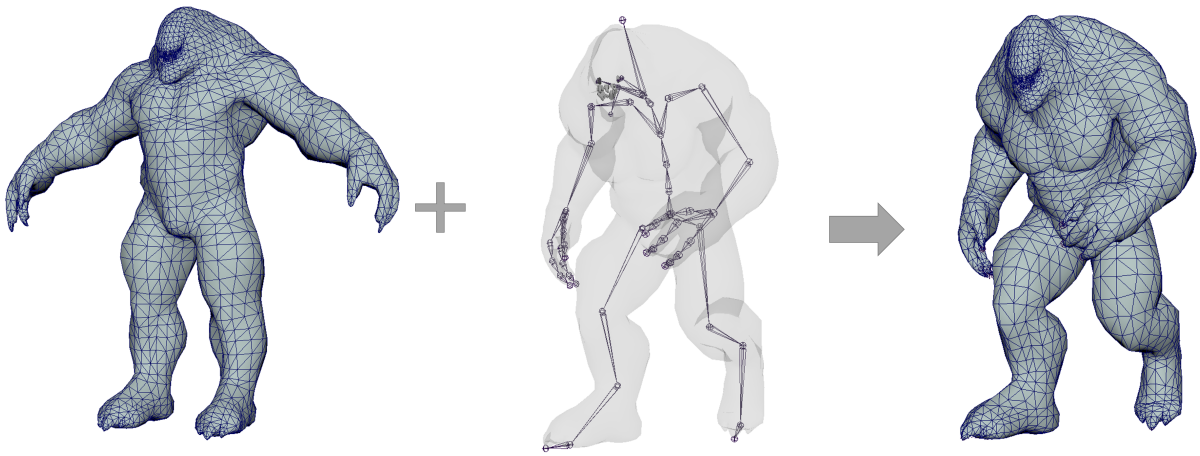


Figure 1.1: A deforming character (Troll Model). (a) represents the mesh surface, which defines the shape of the character. (b) illustrates the underlying skeleton that provides structural support. (c) showcases the walking animation, created through the process of rigging.

the deforming character is presented in Figure 1.1.

Typically, geometry data is crafted with precision by skilled artists, involving laborious processes such as pixel-by-pixel adjustments and vertex-by-vertex alignments. This attention to detail is particularly evident in the creation of digital environments, where collaboration between 3D artists, animators, and digital painters is essential. Beginning with the construction of a mesh to form the character's surface, artists establish a skeletal framework through a process known as rigging. This step requires aligning the mesh onto the underlying structure and articulating distinct regions in response to various joints within the skeleton. Subsequently, meticulous manipulation of keyframes controls the movement of the skeleton, causing the surface to deform and ultimately bringing the animation to life.

1.1.2 Enhancing Visual Diversity

Enhancing visual diversity is crucial in the creation of engaging, inclusive, and realistic digital environments. This goal becomes particularly significant in the realm of digital media, including film production, interactive entertainment, virtual reality, simulations, digital training, and visualization [51, 95, 116].

The development of deforming characters, each able to perform complex animations, is essential in expanding the range of visual diversity within digital platforms. By incorporating diverse and dynamic characters into digital content, creators enhance their ability to connect with a broader audience, ensuring that the digital environment reflects the complex variety of real-life experiences.

This process extends beyond the creation of a single character; instead, it requires replicating this labor-intensive workflow to produce numerous characters consistently, highlighting the significant challenges inherent in digital content creation. In the pursuit of crafting a visually immersive environment, creating a diverse array of characters with distinct appearances and animations is crucial. Consider, for instance, the scenario of a virtual reality game where a large amount of monsters charges toward the player—an experience that blends apprehension and exhilaration. However, if each of these monsters shared identical geometry, lacking any variation, the immersive quality would suffer, reducing the realism of the experience. To truly enhance virtual diversity and foster a heightened sense of authenticity, introducing morphological variations among characters becomes essential, enriching the virtual landscape and intensifying player engagement.

One effective method to achieve this diversity is by morphing multiple characters together. By blending the characteristics of different models, artists can create unique hybrids that retain individual traits while presenting new, captivating forms. This approach not only adds variety to the character roster but also allows for the exploration of innovative designs that transcend conventional boundaries. Through meticulous experimentation and iteration, artists can unleash their creativity, breathing life into the digital world and providing players with an immersive and unforgettable experience.

1.1.3 Blendshape and Cross-Parameterization

In the realm of commercial solutions, blendshapes emerge as a widely-used method for achieving the morphing effect between multiple characters [4, 65]. Blendshapes involve creating a series of predefined shapes or targets for a character's mesh. These targets encompass various facial expressions, poses, or even entirely different character designs. By smoothly interpolating between these targets, animators can seamlessly transition characters, allowing for fluid transformations. Blendshapes offer a flexible and efficient approach to introducing morphological variations, empowering artists to craft captivating character animations that enrich the overall narrative experience.

However, it's important to note that blendshapes have their limitations. One significant constraint is that they require the input meshes to be isomorphic, meaning they must have the same topology

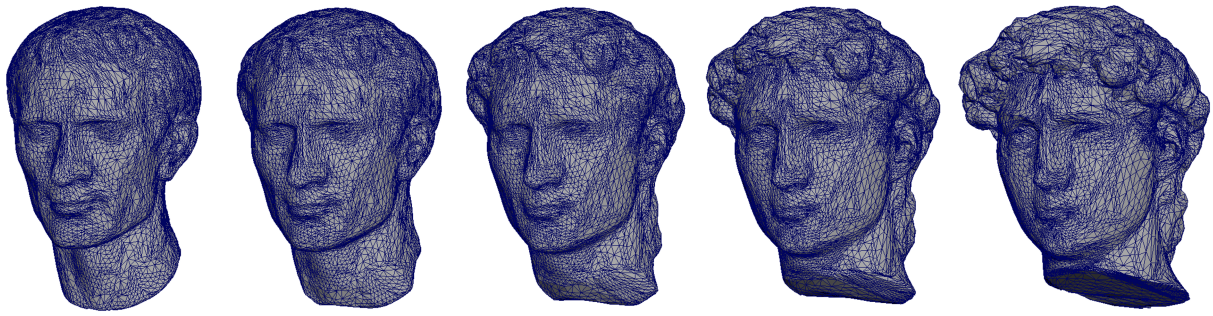


Figure 1.2: The morphing effect between two human head models is achieved through the use of blendshape technology in Maya. Both input meshes are isomorphic, each containing 12,937 vertices and 25,870 triangles, and they share identical surface topology.

and vertex count. This restriction can pose challenges when working with characters of varying complexity or when attempting to morph between characters with significantly different anatomies. Adapting meshes to meet these requirements can be labor-intensive, often requiring manual adjustments and meticulous tweaking to ensure compatibility. Despite this limitation, blendshapes remain a powerful tool in the animator's toolkit, providing a flexible and efficient means of creating dynamic character animations. A morphing effect between two isomorphic meshes is shown in Figure 1.2.

In order to automate the creation of characters with diverse morphological features from non-isomorphic sources, significant research has focused on refining *cross-parameterization techniques* [76, 77, 92, 106, 122]. This process is critical for facilitating the seamless integration and morphing of deforming characters within digital narratives. Cross-parameterization aligns meshes within a unified domain, ensuring accurate alignment and adaptation of each character, despite their differences in shape, size, or structure, for cohesive narrative representations. This pivotal alignment ensures essential characteristics and animations are preserved and effectively translated across different characters. Mapping and morphing characters within a common domain underpins creating visually diverse and dynamic environments, enabling a cohesive yet varied character presentation. Cross-parameterization gives creators tools to fluidly blend and transition between characters, enhancing immersion in digital content while ensuring visual diversity and realism.

Spherical domains are frequently chosen for the parameterization of genus-zero meshes, given their simplicity and efficiency in handling uniform surfaces. In the realm of cross-parameterization, seminal works primarily leverage the surface characteristics of characters to establish feature correspondences [61, 106, 111]. These methods typically involve mapping the mesh onto a singular spherical representation, within which feature alignment is directly executed. This approach, by

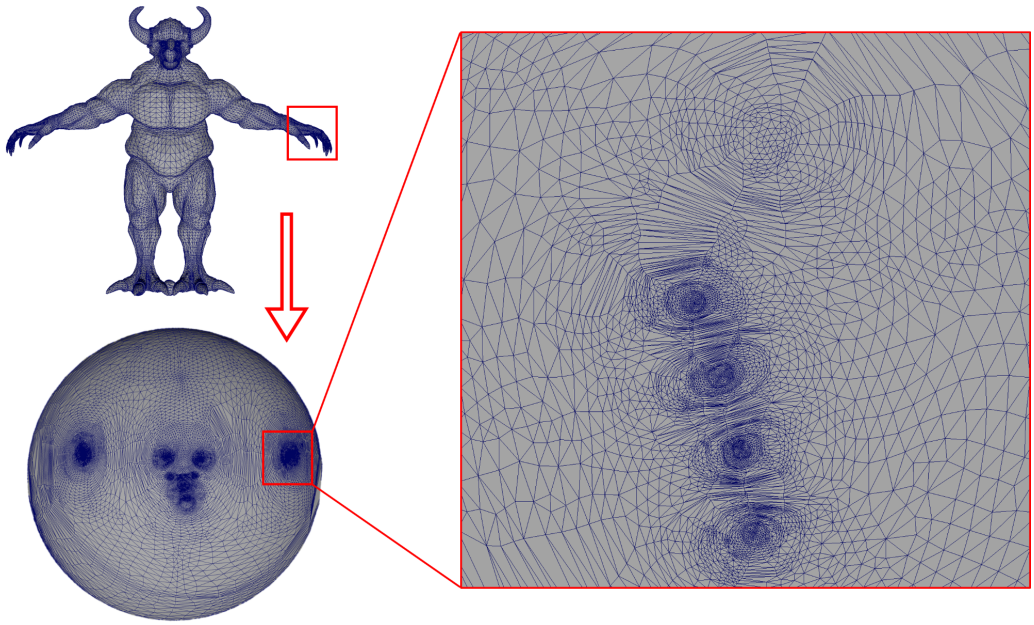


Figure 1.3: In traditional spherical parameterization approaches, certain areas, particularly the hands and fingers, may experience significant distortion. This distortion can lead to stretching and twisting effects within the feature alignment algorithm, adversely impacting the accuracy of the alignment. Additionally, the high density of vertices in these areas can introduce numerical issues, further complicating the parameterization process.

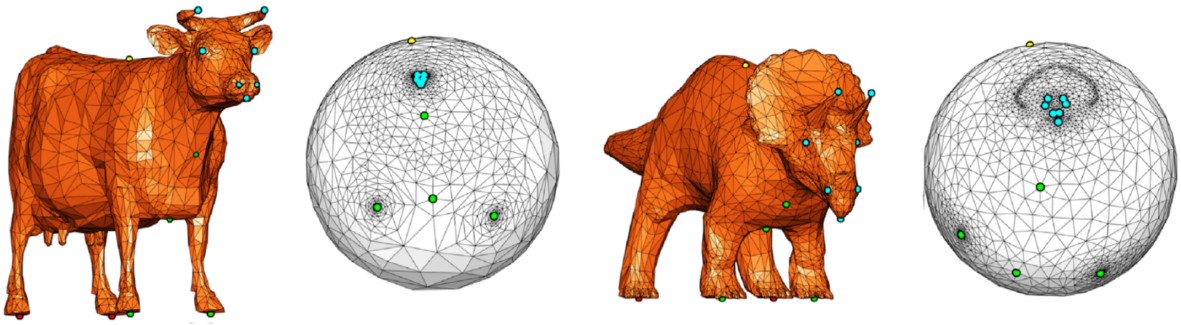


Figure 1.4: In the study presented by Peng and Timalsena [106], users manually select feature points on cow and triceratops models within the spherical domain.

employing a uniform parameterization resolution across the entire mesh, often inadvertently applies the same level of detail to every area of the model. While effective in basic scenarios, this strategy can introduce significant issues in regions dense with features. The uniform application of resolution may result in disproportionate stretching or compression of the mesh during parameterization, thereby compromising geometric accuracy and introducing unwanted parametric distortions, as shown in Figure 1.3. Such distortions are not just minor inconveniences; they can fundamentally alter the perceived quality and integrity of the character's surface, potentially detracting from its visual appeal and realism.

Traditional spherical parameterization methods involve mapping a mesh onto a sphere as a preliminary step towards achieving feature alignment. However, another significant drawback of these methods is the necessity for manual intervention: users are required to meticulously select corresponding feature points on the mesh to ensure accurate alignment, as shown in Figure 1.4. This task of selecting feature points is not only challenging but also lacks intuitiveness, as it demands a high level of precision and a deep understanding of the mesh's geometric and topological intricacies. The manual selection process can be time-consuming and prone to errors, particularly for complex meshes or those with subtle features, making it a less efficient step in the workflow of mesh manipulation and analysis.

Moreover, these existing cross-parameterization methods primarily focus on surface properties, neglecting the hierarchical structure of characters' deforming features. This limitation renders them inadequate for handling the detailed dynamics of character deformation, especially in scenarios where detailed, deformable features are crucial. These features encompass both surface details and deeper volumetric changes, necessitating a more advanced approach to handle the various layers of character morphology.

In this study, we introduce an innovative hierarchical spherical cross-parameterization method that intricately incorporates deformation characteristics, including joints and skin weights, into the feature identification process for characters. By adopting a hierarchical spherical representation, our approach adeptly acknowledges and integrates the inherent parent-child relationships among deforming features. This facilitates an alignment of features across the various layers of the parametric hierarchy, ensuring a coherent and structured process.

1.2 Contributions

Our technique improves upon traditional methods by significantly reducing parametric distortions and effectively mitigating twisting artifacts during the feature alignment phase. This focus on the hierarchical structure and deformation properties preserves the integrity and fidelity of the original deforming features.

Initially, our process begins by segmenting the meshes, focusing on their deforming features to divide them into distinct body parts. This step involves an analysis to identify critical features and establish a hierarchical relationship among them, ensuring a structured organization of the mesh's components based on their deformative significance.

Subsequently, we parameterize these mesh segments onto a hierarchical spherical domain, adopting a hierarchical and progressive approach that progresses from child segments to parent ones. This strategy allows for the parent parameterization to use the features identified in the child parameterization, thereby ensuring that the relationships among the features are preserved through the process.

Finally, our parameterization result enables the mapping of multiple deforming characters into a unified domain. We have innovated a parent-child feature alignment method designed to synchronize corresponding features across characters. Once these features are precisely aligned on the spherical surface, it gives us the possibility for extensive manipulation—be it modifying, blending, transferring, or augmenting features—to achieve varied visual effects and enhance morphological diversity.

This comprehensive approach significantly enhances the quality of outcomes in a wide array of applications, such as 3D morphing, texture transfer, character synthesis, and deformation transfer. Through the hierarchical alignment and manipulation of deforming features, our method facilitates the creation of dynamic and visually diverse digital environments, showcasing the capabilities of our

parameterization and alignment techniques in enriching digital content with details and realism.

The hierarchical nature of our method allows for a more nuanced and controlled manipulation of character's deforming features, thereby enabling a richer expression of morphological variations without compromising the model's structural coherence. By addressing the limitations inherent in existing cross-parameterization techniques—specifically, their neglect of the hierarchical structure of deforming features—our approach opens new avenues for the creation and manipulation of digital characters. This progress allows for more dynamic and realistic character animations, setting the stage for innovations in digital content creation and the broader field of computer graphics.

Overall, our contributions can be divided into two parts:

(1) We have developed a child-to-parent parameterization approach that constructs an organized hierarchy of spheres. This hierarchy mirrors the character's morphology with high fidelity and ensures the integrity of deforming features throughout the parameterization process. This method enables the preservation of critical deformation properties—such as joints and skin weights—thereby ensuring that the character's dynamic qualities are maintained even after extensive morphological transformations.

(2) We introduce a parent-to-child feature alignment method that systematically identifies and matches deforming feature correspondences across the various layers of the hierarchical structures. This strategy employs advanced algorithms to automate the alignment process, significantly enhancing the efficiency and accuracy of feature correspondence identification. By doing so, it bridges the gap between different levels of the hierarchy, ensuring that deforming features are cohesively integrated and accurately represented across the entire character models.

1.3 Organization

The structure of this dissertation is organized as follow. Chapter 2 provides a detailed literature review, covering spherical parameterization, cutting-edge cross-parameterization methods, and significant studies pertaining to deforming features.

In Chapter 3, we present an overview of this dissertation's contributions alongside a concise introduction to our methodologies, outlining the core principles that guide our research. Chapter 4 delves into our mesh segmentation process, detailing the approach we employ to dissect meshes and pinpoint the critical deforming features essential for our analysis. Chapter 5 is dedicated to illustrating

the construction of spherical representations, explaining how we transform segmented meshes into a spherical domain while preserving their correlations. Following this, Chapter 6 introduces our novel parent-child alignment methods, which are pivotal for maintaining structural integrity across the hierarchical data structure we utilize.

A thorough evaluation of our methodology is provided in Chapter 7, where we assess the effectiveness and applicability of our approach through various applications, demonstrating the versatility and robustness of our methods. Chapter 9 offers an in-depth discussion on the implications, limitations, and potential future directions of our work.

Finally, Chapter 10 concludes this dissertation, summarizing the key findings, contributions, and the broader impact of our research.

Chapter 2

Literature Review

In this work, we introduces an innovative approach to semi-automatically generate characters with morphological variations and animations through a hierarchical spherical cross-parameterization method. This method is designed to deform characters by establishing hierarchical representations. It accomplishes this by mapping input meshes into a spherical parametric domain and employing a parent-to-child feature alignment strategy across these representations.

The chapter begins with a comprehensive review of current spherical parameterization methods, highlighting their strengths and weaknesses. Following this, we delve into an examination of existing studies that focus on identifying and aligning feature correspondences within the parameterized domain. Lastly, we conduct a review of existing works of several potential applications that could be realized with our methods.

2.1 Cross-parameterization

Cross-parameterization is an important technique in computer graphics and geometric processing that establishes a correspondence between two or more geometric in a common domain, enabling the transfer of information such as textures, geometric detail, or deformation between them [35, 76]. This method is pivotal in applications ranging from texture mapping, morphing, and mesh editing to more complex tasks such as shape analysis and recognition. The domains involved in cross-parameterization can vary widely, encompassing not only different 3D models but also varying geometric representations.

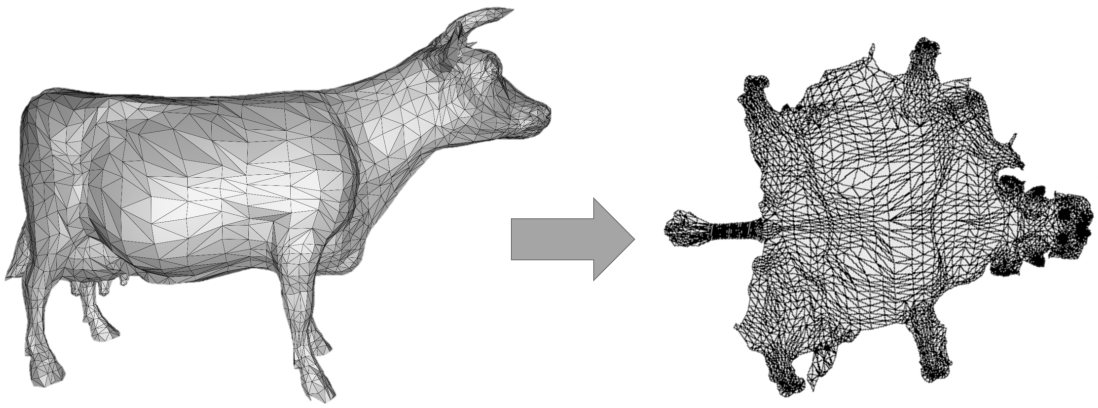


Figure 2.1: A planar parameterization example with the Cow model composed of 2,904 vertices and 5,804 triangles, excerpted from Sheffer et al. [125]. The open boundaries are created by cutting and unfolding the mesh at the high curvature regions.

2.1.1 Parameterization

In general, mesh parameterization algorithms transform a 3D mesh from its original domain into a parametric space [59]. This transformation encodes the mesh into a new format by establishing a bijective correspondence between all vertices [126]. The primary goal of such parameterization efforts is to transpose a 3D mesh into a more conducive domain, facilitating more efficient processing of the mesh's surface attributes and geometric features compared to operations within the 3D space [16]. A significant innovation presented in this dissertation lies in the development of hierarchical spherical parameterization and alignment techniques. These methods are distinctively designed to adeptly manage not only the geometric characteristics inherent to a character's mesh but also the dynamic features extracting from the character's articulation and animations, thus advancing beyond the capabilities of existing solutions.

Planar Parameterization

Planar parameterization is the most common method that maps a mesh into a 2D UV domain and has been used for various applications such as texture mapping

Planar parameterization is a fundamental concept in geometric modeling, graphics, and mesh processing. It is the most common parameterization method, enabling the flattening of 3D surfaces onto 2D planes with minimal distortion. This technique is crucial for texture mapping, mesh optimization, and various other applications in computer-aided design and graphics [24, 125, 126].

Sheffer and de Sturler [124] contributed a milestone technique focusing on angle-preserving mappings, known as the ABF++ method. Their work addressed the limitations of earlier methods by reducing texture stretching and providing more control over the parameterization process.

The development of planar parameterization methods has been significantly influenced by the work of Floater [36], who introduced Mean Value Coordinates for closed triangular meshes. Floater's method paved the way for parameterizations that effectively minimize angular distortion, thereby preserving the local geometry.

Levy et al. [85] introduced the Least Squares Conformal Map (LSCM), which formulated the parameterization as a least squares problem. This approach significantly improved the uniformity of the parameterization and was pivotal in handling complex topologies.

The progress in planar parameterization also encompasses the works of Sander et al. [117], who focused on optimizing the texture mapping process. Their approach allowed for more efficient utilization of texture space, reducing the wasted texture area and improving the visual quality of the mapped textures.

Despite these advancements, planar parameterization continues to face challenges such as dealing with highly irregular topologies. The quest for more efficient and less distortion-prone algorithms is ongoing, with current research exploring the use of machine learning and advanced optimization techniques [129]. Figure 2.1 shows a planar parameterization example produced by Sheffer et al. [125], in which high curvature regions were cut and unfolded into the planar domain. The angle and area ratios of triangles in these regions were preserved by adjusting the positions of boundary vertices in order to mitigate the parametric distortion. However, the planar parameterization creates the seams after cutting the mesh, so the continuity of the surface is not guaranteed in its planar representation. The open boundaries at the seams complicate the representation of the features, and make it difficult to align feature correspondences between multiple input meshes.

Surface Conformal Parameterization

Surface conformal parameterization is a key process in computational geometry that involves mapping a surface that preserves angles, aiming to maintain the original shape's local geometry. This technique is crucial in various applications, from medical imaging to computer graphics, where preserving the surface's intrinsic properties is essential.

Early work in the field by Haker et al. [50] set a precedent with their algorithm for conformal surface flattening. Their approach provided a way to flatten brain surfaces while preserving the cortical patterns, which is critical for neurological studies.

Gu and Yau's groundbreaking work [49] on computing conformal structures of surfaces revolutionized the approach to parameterization by introducing the discrete Ricci flow method. This work offered a robust computational framework for surface parameterization that has since been applied in various domains.

Another significant contribution to surface conformal parameterization was made by Desbrun et al. [29], who proposed an intrinsic parameterization method to tackle the issue of minimizing distortion during the mapping process. This method was particularly notable for its ability to parameterize complex shapes with high fidelity to the original geometry.

Reserachers have leveraged advanced techniques like the zipper algorithm by Marshall and Rohde [98], which provides a powerful tool for constructing conformal mappings. This method has proven to be effective for a wide range of applications, further extending the utility of conformal parameterization. Springborn et al. [134] introduced a variational principle for conformal parameterization with applications to surface remeshing.

Despite the progress made, challenges in conformal parameterization remain, particularly when dealing with high-genus surfaces or surfaces with boundaries. Research efforts continue to focus on developing more efficient and flexible algorithms.

Authalic Parameterization

Authalic parameterization, which strives to preserve the surface area in the mapping process, is an essential technique in computational geometry and computer graphics, especially for applications where maintaining the proportion of areas is critical. This form of parameterization finds use in various fields, such as medical imaging, where accurate area representation is necessary for meaningful

analysis.

A pivotal study by Hormann and Greiner [60] introduced MIPS (Most Isometric Parameterizations of Surfaces), which is a technique aiming to minimize area distortion in mappings. This technique paved the way for further research in authalic parameterization by providing a framework for achieving near-isometric planar domains.

Further advancements were made by Sheffer [124], who explored angle-preserving parameterizations and discussed the applications and limitations regarding area preservation. This work highlighted the importance of considering both angle and area preservation to achieve a more holistic parameterization.

The work of Snyder et al. [131] provided insights into mapping textures onto 3D surfaces with minimal area distortion, which is especially important in avoiding texture stretching and compression. This method was one of the early attempts to directly address the issue of area preservation in texture mapping.

Zou et al. [156] made significant contributions to the field by developing an algorithm for authalic parameterization of remeshing applications, addressing the computational challenges and providing a robust solution for preserving the surface area during the remeshing process.

These seminal works laid the foundation for subsequent innovations in authalic parameterization. Yet, the challenges persist in optimizing algorithms for complex surfaces and integrating them into real-time applications. Recent research continues to explore computational methods and tools to improve the accuracy and efficiency of authalic parameterization.

Spherical Parameterization

The field of spherical parameterization has seen significant advancements, largely due to its effectiveness in creating seamless and continuous mappings for genus-zero models [111]. This method has been crucial for various applications in computer graphics, including texture mapping, mesh deformation, and 3D model visualization.

Foundational research [47, 155] has been instrumental in establishing the theoretical and practical frameworks for spherical parameterization, highlighting its potential and versatility in handling complex geometries. Praun and Hoppe [111] introduced a spherical parameterization method designed specifically for remeshing applications. Their approach utilized signal processing techniques

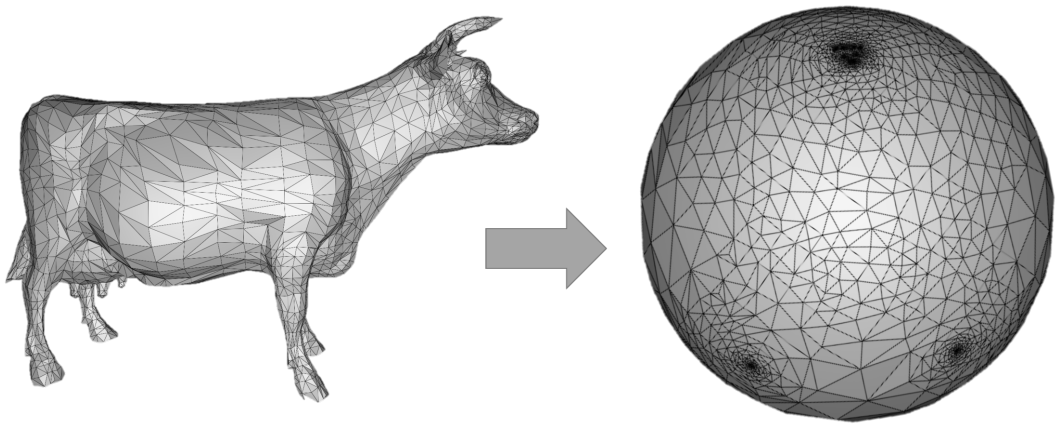


Figure 2.2: A spherical parameterization result of the cow model generated by the spherical parameterization method published by Peng and Timalsena [106]. The Cow model contains a mesh with 2,904 vertices and 5,804 triangles. The high curvature regions near the end joints, like the head and legs, are projected into a small area with a dense mesh.

to achieve uniform remeshing, significantly enhancing the quality and efficiency of the parameterization process.

In an effort to further reduce distortion in spherical mappings, Kharevych et al. [73] proposed a discrete conformal mapping technique. By leveraging advanced mathematical models, their work enabled more accurate representations of complex geometries on a spherical domain, preserving angles and minimizing area distortion. Floater and Hormann [37], who developed a method for surface parameterization based on convex combinations of barycentric coordinates, facilitating the mapping of arbitrary surface meshes onto a sphere with minimal distortion.

In recent developments, spherical parameterization methodologies have been broadly categorized into two main approaches: progressive embedding and energy minimization.

Progressive Embedding: This approach incrementally constructs the parameterization by embedding portions of the mesh onto the sphere step by step [56]. It focuses on maintaining local geometric properties during each embedding phase, ensuring that the mesh unfolds naturally onto the spherical surface without overlaps or distortions. Progressive embedding is particularly advanta-

geous for its intuitive construction and ability to closely monitor the quality of the parameterization at each step. However, it may require careful handling to avoid local minima that could lead to suboptimal overall parameterization.

Inspired by the well-known progressive meshes algorithm [57, 58], progressive embedding methods simplify the mesh to a base shape (e.g., a tetrahedron) by collapsing edges and removing vertices one by one. The base shape is then embedded on the sphere, and vertices are progressively inserted back and projected onto the sphere.

A notable contribution to this area was made by Crane et al. [25], who introduced a robust framework for spherical parameterizations that adaptively refines the mesh to minimize distortion. Following this, Tian et al. [139] developed a progressive method that iteratively adjusts the parameterization to reduce distortion in real-time applications. Their work demonstrates significant improvements in the speed and quality of parameterizations for dynamic meshes, showcasing the potential of progressive strategies in interactive environments.

Shen et al. [127] proposed a progressive embedding method emerges as a novel solution, designed to preserve the theoretical integrity of the Tutte embedding while enhancing resilience against the inaccuracies of floating-point arithmetic. This method is inspired by the concept of progressive meshes, where the strategy involves systematically collapsing edges of an initially invalid embedding into a simplified yet valid mesh configuration. The process then progresses by incrementally reintroducing points, carefully maintaining the embedding's overall validity.

Peng and Timalsena [106] introduced a novel metric for measuring the distortion of vertex insertion in the local kernel on the sphere and also introduced the concept of rough and exact alignments of feature correspondences. Hu et al. [61] presented a flat-to-extrusive simplification strategy to generate a spherical parameterization with local refinement. They improved the metric for distortion measurement to optimize isometric distortion and obtain a valid local embedding. However, the dense vertices in the progression can cause numerical issues when refining in the kernel, as its size may be too small.

Energy Minimization: Contrasting with the stepwise nature of progressive embedding, energy minimization approaches seek to optimize a global energy function that reflects the quality of the parameterization across the entire mesh. This method often involves solving a system of equations derived from the energy function, aiming to minimize distortions such as stretching or compression of the mesh surface [88]. Energy minimization is praised for its potential to achieve globally optimal solutions, providing high-quality parameterizations that faithfully represent the original mesh's

geometric and topological features.

The concept behind energy minimization is to perform a global non-linear optimization to assign energy values to edges and triangles, which in turn allows the mesh to be transformed into a more spherical shape that can be mapped onto the sphere [71]. For instance, Friedel et al. [39] introduced a non-linear inverse distance-weighted energy solution. Kazhdan et al. [70] presented a mean-curvature flow. Nadeem et al. [101] proposed an energy function to balance angle and area distortions. Wang et al. [143] developed a non-linear constrained optimization method. Aigerman et al. [3] minimized the spherical Dirichlet energy under boundary conditions of the orbifold structure. However, global optimizations are not typically used in a local space hierarchy, as the iterative optimization steps are applied to the entire mesh surface. This can result in difficulty controlling the parametric quality, requiring sophisticated experimental setups. Furthermore, a desirable parameterization may not have zero energy, and an overly reduced energy can cause issues with triangle degeneration.

The choice between progressive embedding and energy minimization depends on various factors, including the specific requirements of the application, the complexity of the mesh, and the desired balance between computational efficiency and parameterization quality. While progressive embedding offers more control over the embedding process, energy minimization is adept at handling large-scale optimizations, making it suitable for complex or highly detailed models.

The distinctive feature of our method is its employment of a hierarchical arrangement of spheres within the parametric domain, diverging from traditional approaches that utilize a singular, flat spherical domain. This multi-level framework inherently favors the progressive embedding method due to its structured and stepwise nature. The hierarchical spheres allow for a more nuanced and detailed parameterization process, accommodating complex geometries with greater precision and efficiency. By leveraging this approach, our method enhances the adaptability and fidelity of the parameterization, making progressive embedding particularly advantaged for managing the complexity of the hierarchical domain.

In our approach, we avoid the conventional practice of mapping dense mesh information onto a singular sphere. Instead, we segment the mesh into distinct segments, guided by the character's deformation characteristics, and systematically embed each segment into its own spherical representation. This segmentation is pivotal, with the boundaries between segments acting as essential deforming features that facilitate the coherent relationship between parent and child spheres.

A key advantage of our method lies in its capacity to preserve the continuity of embedding across segments, even those presenting "holes" due to the boundaries. This ensures that the advantages of

the spherical representation is maintained, allowing for seamless transitions and feature alignments between corresponding segments. Through this approach, we achieve a more detailed and dynamic parameterization, accommodating complex deformations while ensuring a cohesive and continuous embedding of the character's mesh.

Parameterization on More Domains

Cylindrical and box parameterization are specialized techniques within the field of surface parameterization, designed to map 3D surfaces to cylindrical and box-like domains, respectively. These methods are crucial for efficiently mapping textures and generating patterns on surfaces that approximate or can be enclosed by cylindrical or box shapes.

Cylindrical parameterization is particularly suited for objects with an elongated, tubular structure, such as limbs, bottles, or pipes. A notable approach to cylindrical parameterization was introduced by Maillot et al. [94], who proposed an interactive technique that allows for the intuitive mapping of textures onto cylindrical surfaces. This method laid the groundwork for further developments in automating and refining cylindrical mappings.

Box parameterization, on the other hand, is aimed at objects with a predominantly boxy structure, making it ideal for buildings, furniture, or any man-made objects with flat surfaces and sharp edges. A significant contribution in this area was made by Lévy et al. [85], who developed techniques for mapping complex geometries onto box-like domains, optimizing the use of texture space and minimizing distortion.

The work of Sheffer and de Sturler [124] on angle-based flattening also had implications for box parameterization, offering a methodology for unfolding surfaces onto a plane in a way that can be adapted to box-like unfoldings. Their research contributed to the understanding of how to manage the trade-offs between area distortion and angle preservation in box parameterization.

More recently, advancements in algorithmic approaches have allowed for more automated and efficient parameterization processes. Smith et al. [129] explored the use of bijective parameterization techniques that ensure a one-to-one mapping, reducing artifacts in both cylindrical and box parameterizations.

Despite these advancements, challenges remain in handling complex topologies and achieving real-time performance. The ongoing development of computational techniques and tools aims to address these challenges, pushing the boundaries of what can be achieved with cylindrical and box param-

eterization.

2.1.2 Feature Identification

Part-based Mesh Segmentation

In order to analyze the deforming features of characters and construct hierarchical representations within this work, segmentation emerges as an important process [66]. This segmentation is not merely about dividing the mesh into manageable pieces; it's about understanding how different parts of a character move and interact, which is crucial for capturing the features of their motion. By decomposing the mesh into segments, we can seperated specific areas of deformation, allowing for a detailed evaluation of how each segment contributes to the overall animation [1, 90, 91]. This analysis is fundamental in building accurate and dynamic hierarchical representations that represent the complex movements inherent in deforming characters.

Mesh decomposition stands as a cornerstone of mesh processing research, aiming to partition a mesh into distinct parts. This endeavor is categorized primarily into chart-based and part-based segmentation, as detailed in the literature [115, 123]. Our methodology is designed to segment the mesh according to deforming features which, representing the deformation across various body parts, aligns our approach with part-based segmentation.

Part-based segmentation encompasses three principal techniques: volume-based, surface-based, and skeleton-based methods [115]. The volume-based technique focuses on segmenting the mesh by its shape characteristics, offering a geometrically intuitive approach. On the other hand, the surface-based method, primarily suited for 2D meshes, finds utility in 3D mesh applications under certain conditions, leveraging surface attributes for segmentation. Lastly, the skeleton-based method aims to extract a skeletal structure from the mesh, providing a framework that reflects the mesh's inherent form and structure. These methods, as suggested by Rodrigues et al. (2018), can be effectively combined to leverage their unique advantages, offering a comprehensive strategy for part-based mesh segmentation.

Many existing part-based segmentation approaches primarily target the partitioning of meshes based on static features, overlooking the dynamics of deforming characters. Zhang et al. [149] introduced a space partitioning technique utilizing the minima rule for feature extraction to identify boundaries, albeit focusing solely on static models. Attene et al. [9] proposed a hierarchical segmentation strategy that clusters meshes into segments through primitive fitting. Initially, each triangle is

considered an individual primitive cluster, which, depending on the required level of segmentation, undergoes consolidation with other clusters for optimal fit.

Zhang et al. [153] developed a segmentation method employing mean-shifted curvature, adopting a region-growing algorithm. This technique, commonly applied in mesh segmentation [14, 34, 78], incrementally forms segments starting from a single vertex or seed. In each iteration, adjacent vertices or triangles that best match the criteria are integrated into the segment until no further additions are possible, ensuring the resultant segments are topologically cohesive.

Furthermore, Chen et al. [22] established a benchmark for assessing the effectiveness of mesh segmentation methods. This comprehensive review not only evaluated segmentation quality but also compiled 4300 manually segmented meshes derived from 380 base models. Despite its extensive coverage, this benchmark exclusively considered static meshes, highlighting a gap in resources for evaluating dynamic or deforming mesh segmentation.

These methodologies, while foundational, underscore the need for advanced segmentation techniques that can adeptly handle the complexities of deforming characters, capturing both their static and dynamic attributes to achieve more nuanced and accurate representations.

Feature Identification based on Mesh Segmentation

One of the foundational approaches in feature identification was proposed by Katz and Tal [68], who introduced a hierarchical scheme for feature extraction on 3D meshes. Their method efficiently identifies salient geometric features, laying the groundwork for subsequent segmentation and simplification techniques.

Building on this, Sun et al. [136] developed a concise representation for 3D shapes that emphasizes feature lines, enabling more effective shape analysis and recognition. Their work represents a significant step toward understanding and utilizing the intrinsic properties of surface features for shape manipulation and classification.

In the realm of high-level feature identification, Gal and Cohen-Or [40] introduced a method for detecting salient geometric features on 3D models. By focusing on the perceptual aspects of surface features, their approach aids in the visualization and analysis of complex 3D shapes, enhancing the capabilities of content creation tools and algorithms.

Several researchers have delved into the challenge of segmenting deforming meshes to better capture

their dynamic properties. Lee and Wang [83] introduced a technique to partition a mesh into nearly rigid sub-meshes by analyzing deformation patterns, specifically measuring the deformation degree among adjacent triangles. This approach uniquely required animation data as input to effectively segment the mesh based on its dynamic behavior.

Skraba et al. [128] adopted a different strategy by employing a persistence diagram for mesh clustering, utilizing the concept of topological persistence [32] to pinpoint deforming features. While their method excelled in generating clusters of varying sizes and demonstrated robustness in identifying significant deformations, it faced challenges in achieving smooth boundaries among the segmented regions.

Recent advancements have leveraged deep learning techniques to revolutionize feature identification. Qi et al. [112] presented PointNet, a deep neural network capable of directly processing point clouds to classify and segment 3D objects. This breakthrough has opened new avenues for feature identification, allowing for the direct analysis of raw 3D data without the need for preprocessing or meshing.

Refinement typically succeeds segmentation in mesh processing, addressing a common limitation of existing methods: their inability to ensure smooth boundaries between segments [123]. This step is crucial for enhancing the quality of segmentation by smoothing out the transitions between different parts of the mesh, thereby improving the mesh's overall aesthetic and functional integrity.

Kaplansky and Tal [67] introduced a sophisticated boundary refinement technique aimed at enhancing the delineation of mesh segments. Their algorithm strategically adjusts the segment boundaries to more accurately reflect the underlying structure of the mesh. It achieves this by enabling vertices to shift across faces, thereby relocating boundaries to positions that better align with the natural contours and features of the mesh. This method significantly improves the segmentation outcome by ensuring that boundaries between segments are not only smoother but also more representative of the mesh's geometric and topological nuances.

Building upon these insights, our proposed method will leverage skin weights as the key deforming feature for mesh segmentation. We aim to adopt a decomposition approach, recognizing the necessity for the segmented regions to maintain topological continuity to facilitate effective parameterization. This method promises to advance the segmentation of deforming meshes by ensuring a cohesive and continuous mapping of the mesh's dynamic aspects, enhancing the accuracy and utility of the segmentation process for complex animations.

2.1.3 Feature Correspondence

Surface Feature Correspondence

Establishing meaningful correspondences is pivotal for the effective transfer of mesh information across different characters, involving the identification of points or patches that represent geometric signatures unique to each character. These correspondences, critical for maintaining geometric and topological consistency, can exist across various domains, enhancing the versatility of mesh information transfer.

Van Kaick et al. [140] provided a comprehensive review of methods for computing correspondences between triangular meshes, covering both 2D and 3D parametric domains. This survey underscores the diversity of approaches available for establishing mesh correspondences, highlighting their applicability in different scenarios.

Schmidt et al. [120] introduced a discrete surface parameterization technique for disk-topology meshes within the surface domain. Their method focuses on minimizing mesh distortion, presenting a mapping algorithm designed to preserve the original mesh's structural integrity during the parameterization process.

In a subsequent study, Schmidt et al. [121] developed a surface mapping method that aims to minimize mesh distortion by ensuring constant Gaussian curvature. This approach facilitates mapping between meshes of the same genus, emphasizing distortion minimization as a crucial factor in maintaining the fidelity of the mapped meshes.

Peng and Timalsena [106] proposed a two-step process for aligning feature correspondences. Initially, spheres are rotated to approximate the alignment of correspondences closely. This is followed by an exact alignment step, where vertices are adjusted based on their proximity to designated feature points. This method requires the manual specification of feature points, highlighting the importance of human input in achieving precise alignment.

Lee and Kazhdan [82] adopted a similar two-step alignment strategy, beginning with rotational alignment to exploit correlations between spheres. Subsequently, they apply an optimal flow technique, employing a coarse-to-fine approach to refine the registrations further. This method adheres to the principle of iterative refinement, ensuring that the alignment maximizes the correspondences' utility and accuracy.

Least Squares Conformal Maps (LSCM) have emerged as a powerful tool in the domain of cross-parameterization, offering a method to preserve the local angles of a mesh when it is unfolded or flattened. This technique is particularly valuable for texture mapping and mesh editing, where preserving the intrinsic geometry of the surface is crucial.

The foundational principle of LSCM, based on conformal mapping theory, seeks to minimize the angular distortion during the parameterization process. The method was first introduced by Lévy et al. [85], proposing an algorithm that efficiently computes a 2D parameterization of 3D surfaces by solving a sparse linear system. The LSCM approach ensures that each triangle in the mesh is mapped to the plane with minimal angular distortion, making it highly suitable for applications that require the geometric integrity of the surface to be maintained.

Subsequent research has expanded upon the original LSCM framework, introducing various optimizations and applications. For example, Sheffer and de Sturler [124] proposed improvements to the algorithm that enhance the stability and speed of the parameterization process. Furthermore, applications of LSCM have been explored in fields beyond traditional graphics, such as in the parameterization of anatomical surfaces in medical imaging [144].

Together, these studies illustrate the ongoing development and refinement of methods for establishing and utilizing feature correspondences in mesh processing. By focusing on minimizing distortion and maximizing geometric fidelity, these approaches contribute significantly to the field of 3D modeling and animation, enabling more seamless and accurate transfer of mesh information between characters.

Deforming Feature Correspondence

Numerous studies in the realm of deformation transfer have delved into methodologies for transferring the deformation from a source mesh to a target mesh, primarily through the identification of triangle correspondences [12, 41, 114, 135]. While these methods offer significant insights into deformation transfer, they generally do not extend to the concurrent transfer of shape alongside deformation, marking a distinct boundary from our research focus.

In alignment with the objectives of our work, several studies have ventured into the domain of simultaneously transferring both shape and deformation. Xu et al. [147] leveraged the gradient domain to edit deforming mesh sequences, facilitating spacetime morphing that allows for both shape and deformation alterations. Zhang et al. [150] introduced an automatic feature correspondence

algorithm aimed at pose matching, noting that the success of this method depends heavily on the similarity between the input poses, suggesting a limitation in its application to varied poses.

Further contributing to this field, Chen et al. [23] approached the challenge by segmenting each pose within a mesh sequence into functional pieces, applying the correspondence algorithm and alignment process within each piece's parametric domain independently. This piecewise approach offers a nuanced method for handling complex deformations by focusing on the specific functions of different mesh segments.

Medalha et al. [99] presented a novel application of the least-squares technique to morph deforming mesh sequences, showcasing a methodological advancement in the precise and efficient handling of mesh deformations over time.

Nuvoli et al. [103] introduces a novel approach to creating new 3D animated models, such as video game characters, by utilizing components from existing models. This method is applicable to production-ready models that are rigged, skinned, and animated, facilitating the assembly of new characters through mix-and-match operations on skeletons.

These studies collectively underscore the evolving nature of deformation and shape transfer techniques, highlighting a variety of approaches designed to address the complexities of accurately mapping the intricate details of deforming meshes. Our work seeks to build upon these foundational efforts, aiming to further refine the processes of simultaneous shape and deformation transfer to achieve more seamless and authentic animation results.

Previous research has centered on pose matching within mesh sequences, tailoring their methodologies to facilitate the transfer or blending of specific poses. However, these techniques often fall short in terms of flexibility when it comes to editing structures that articulate dynamically. In contrast, our approach seeks to identify deforming correspondences between mesh segments across different inputs, akin to traditional methods of establishing point or patch correspondences based on geometric signatures. The critical distinction in our method lies in the foundation of these correspondences, which are determined by the dynamic influence of joint signatures on the mesh.

Our technique begins by adjusting the orientation of parametric spheres to achieve a preliminary alignment of boundary correlations. Subsequent steps involve a refinement of this alignment, focusing on boundary-to-boundary adjustments through vertex displacement. This process diverges significantly from earlier methods by employing a hierarchical parent-to-child operation. This not only enhances local correspondence refinement but also facilitates the discovery of shape and deformation transitions between connected segments. Our method's reliance on this hierarchical strategy

offers a novel approach that surpasses traditional techniques in capturing the nuanced interplay of shape and deformation within articulated structures.

2.2 Applications of Feature Alignment in Deforming Characters

The application of feature alignment in deforming characters presents a wide array of possibilities, significantly improving digital animation and modeling. The integration of feature alignment in deforming characters not only advances technological capabilities but also introduces new opportunities for impactful applications across multiple sectors.

2.2.1 Morphing Techniques

Mesh morphing techniques have evolved significantly, driven by their applications in computer graphics, animation, and medical imaging. These techniques enable the smooth transition of one mesh into another, preserving geometric details and topology.

A seminal contribution to mesh morphing was presented by Alexa et al. [5], who introduced a method for interpolating between two arbitrary meshes of the same topology. This approach laid the groundwork for subsequent morphing algorithms by emphasizing the importance of maintaining geometric detail during the morph.

Sorkine et al. [133] proposed a method based on Laplacian coordinates that allows for flexible editing of mesh geometry, including morphing. Their approach is notable for its ability to preserve local mesh details while enabling significant transformations.

To address the challenge of morphing between meshes of differing topologies, Sumner and Popović [135] developed a technique that employs a multi-resolution representation of the source and target meshes. This method significantly expanded the applicability of mesh morphing to more complex scenarios.

Recent advancements by Jacobson et al. [62] introduced a bounded biharmonic weights method for real-time deformation and morphing. This technique provides a robust solution for interactive applications, offering both speed and quality.

These contributions represent key milestones in the development of mesh morphing techniques. Our

methods enable the implementation of mesh morphing, with our contributions primarily focusing on the deformation features that have received limited attention in prior research.

2.2.2 Texture Transfer

Texture transfer for 3D meshes has emerged as a pivotal area of research within computer graphics. It focuses on applying textures from one model to another, adapting to the geometric nuances of the target mesh while preserving the texture's fidelity. This technique is vital for enhancing the visual realism of 3D models, finding extensive application in animation, gaming, and virtual reality [81, 84, 110].

Early work in texture transfer was presented by Efros and Freeman [33], who introduced an algorithm for transferring texture from one image to another based on image quilting. This technique laid the groundwork for subsequent developments by emphasizing the importance of maintaining consistency in texture appearance while adapting to new geometries.

Hertzmann et al. [53] expanded upon this concept with their technique for image analogies, which allowed for the application of filters or styles from one image to another, including texture transfer. Their method demonstrated versatility in various applications, from artistic stylization to photorealistic rendering.

Praun et al. [110] introduced a groundbreaking technique for seamless texture wrapping over complex geometries, leveraging surface parameterization and signal processing. This method set the stage for subsequent research by demonstrating the potential of enhancing 3D models' visual quality through texture transfer without manual editing. Lee et al. [84] advanced the methodology by proposing a semi-automatic approach that significantly mitigates distortion through feature matching and harmonic maps.

Interactive methods, such as the one proposed by Sorkine et al. [132], allow users to control the texture transfer process directly, offering greater artistic freedom. Bénard et al. [13] extended these techniques to animated meshes, tackling the challenges posed by dynamic deformation and motion.

Recently, the incorporation of machine learning into texture transfer, as explored by Tan et al. [137], has opened new pathways for automating the process, highlighting the evolving nature of texture transfer research. Gatys et al. [44] introduced a method that leverages convolutional neural networks to separate and recombine the content and style of images, enabling high-quality texture transfer that captures the essence of both source and target textures. Another contribution comes from Luan

et al. [89], who improved upon the neural style transfer framework to achieve photorealistic texture transfer. Their approach ensures that the transferred textures adhere to the structural integrity of the target image, resulting in more realistic and coherent outputs.

Our method facilitates the parameterization and alignment of textures from various characters within hierarchical spherical domains, achieving low distortion akin to other features. Subsequently, these textures can be seamlessly mapped back onto a unified UV domain, enabling the implementation of texture blending and transfer across multiple characters.

2.2.3 Character Synthesis

One of the foundational techniques in character synthesis is procedural modeling, which allows for the automatic generation of character models based on predefined rules or algorithms. Early work by Perlin [107] introduced noise functions that could be used for procedural texture and shape generation, laying the groundwork for more complex character synthesis.

The advent of skeletal animation and skinning techniques, as discussed by Magnenat-Thalmann et al. [93], revolutionized character animation by providing a framework for articulating character movements in a more natural and realistic manner. This approach has been further refined through the introduction of inverse kinematics and physics-based simulations, enhancing the realism of character motions.

Recent developments in machine learning and deep learning have opened new avenues for character synthesis. Techniques such as generative adversarial networks [26] and variational autoencoders [31] have been applied to generate detailed and diverse character models.

Furthermore, the integration of motion capture data into character synthesis workflows, as explored by Holden et al. [54], allows for the generation of dynamic and realistic character animations based on human movements. This method leverages deep learning to map motion capture data onto synthesized characters, significantly improving the quality and realism of animations.

With our hierarchical representations, we enable seamless character synthesis by adjusting the morphing ratios of different body parts and applying boundary interpolation between these parts, resulting in a cohesive and easily controllable process.

2.2.4 Deformation Transfer

Deformation transfer is a pivotal concept in the realm of 3D graphics and animation, enabling the adaptation of complex deformations from one mesh to another, irrespective of their differing topological structures. This technique has been extensively researched and developed, leading to various methodologies and applications ranging from character animation to the creation of dynamic simulations.

Sumner and Popović [135] pioneered the deformation transfer process, introducing a method that allows the deformation exhibited by a source mesh to be transferred to a target mesh, thus facilitating the animation of characters with significantly different anatomical structures. Their work laid the foundation for subsequent research in the field, emphasizing the importance of preserving the original artistic intent while adapting deformations to new models.

Following this, various researchers have expanded on the concept. For instance, Baran and Popović [11] developed an algorithm for automatic rigging and skinning of 3D models, which, when combined with deformation transfer techniques, significantly streamlines the animation process. This development has been crucial for animators, reducing the time and effort required to produce realistic animations.

Moreover, the work by Botsch and Kobbelt [15] on real-time shape editing using radial basis functions presents another dimension to deformation transfer, offering tools for more flexible and intuitive manipulation of mesh deformations. Their approach enhances the practicality of deformation transfer in real-time applications, such as virtual reality and interactive gaming.

The exploration of deformation transfer has also led to advancements in the understanding and processing of mesh topologies, as seen in the work by James and Twigg [64]. They introduced a method for dynamic skinning, improving the quality and realism of deformations for animated characters, especially around complex articulations.

In recent years, the application of machine learning techniques to deformation transfer has opened new avenues for research. Variational autoencoders, as discussed by Tan et al. [137], offer a promising approach for learning deformation representations, enabling the automatic transfer of deformations between meshes without the need for explicit correspondence mapping.

Our approach, which emphasizes deforming features, facilitates the simultaneous implementation of deformation transfer and shape morphing.

Chapter 3

Method Overview

To streamline the generation of morphologically diverse characters with animations, we introduce innovative algorithms designed to parameterize and align corresponding features across deforming characters via a hierarchical representation framework. This approach aims to facilitate a more efficient and flexible creation of animated characters, capturing the essence of their morphological diversity.

As depicted in Figure 3.1, our methodology unfolds through a series of structured processing stages, outlining the steps of our research method. The process is organized into three distinct stages, each pivotal in advancing towards the objective of automating the creation of animated characters with varied morphologies.

3.1 Part-based Segmentation by Deforming Properties

In our work, we employed a part-based segmentation method to segment the mesh according to the boundaries defined by skinning weights, ensuring that each segment corresponds to a specific joint. Utilizing the skinning weights as a measure of the likelihood that a vertex belongs to a particular set of joints, we applied the fuzzy decomposition method [68] to identify cuts with minimal cost. Subsequently, each vertex was assigned a joint index, grouping vertices with identical labels into distinct segments.

Challenges arose when non-adjacent regions received the same joint index, leading to multiple segments being inaccurately matched to a single joint. This scenario risked creating inconsistencies

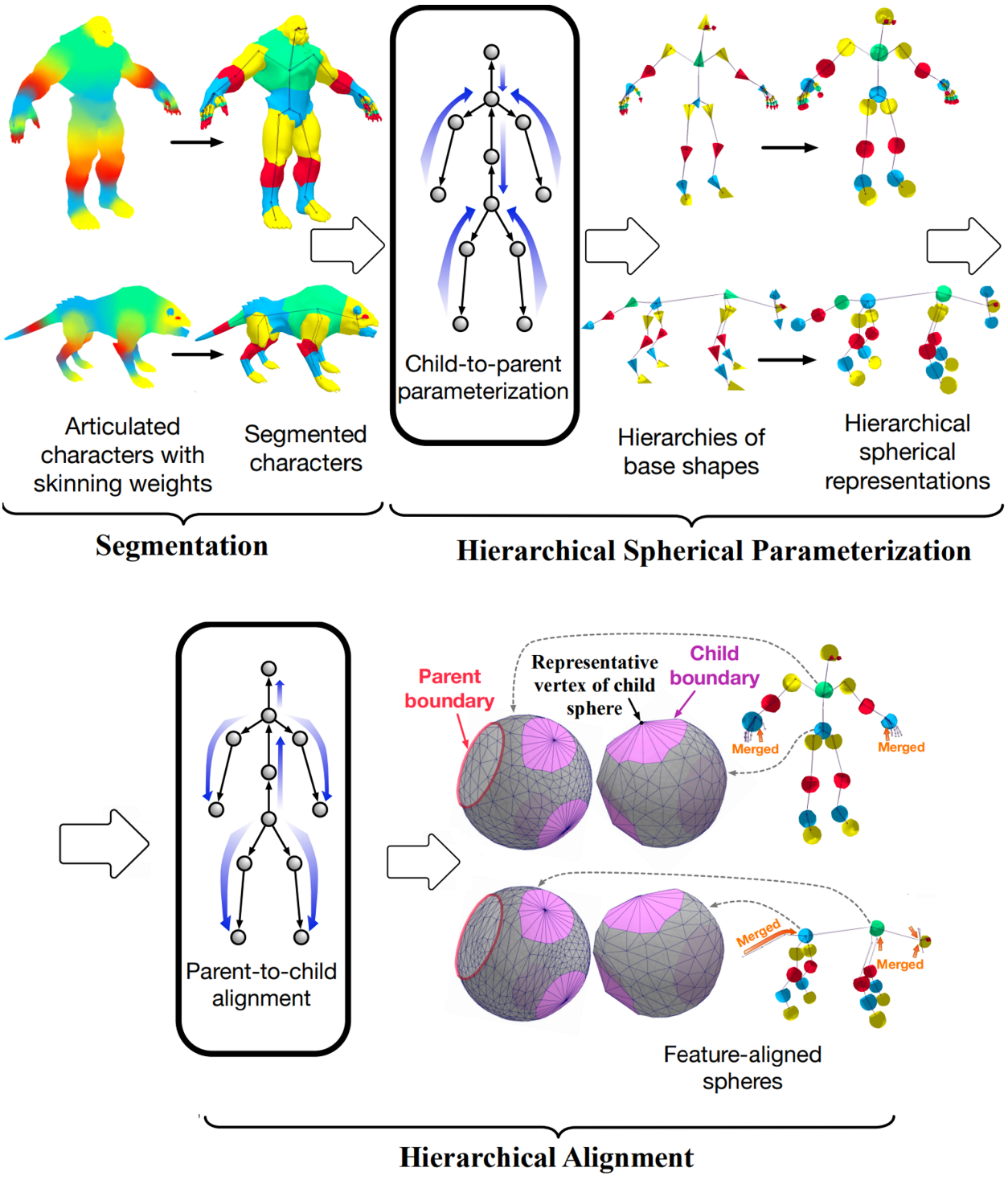


Figure 3.1: Our methodology encompasses three principal stages: segmentation, hierarchical spherical parameterization, and hierarchical feature alignment.

in the parent-child relationships essential for constructing spherical representations. Additionally, there were instances where two adjacent vertices were assigned non-adjacent joint indices, resulting in adjacent regions being erroneously treated as separate segments.

To address these labeling challenges, we developed a label refinement method aimed at consolidating separated regions that share a joint index. This process involved reassigning these regions to the nearest ancestor joint that could seamlessly connect through a sequence of adjacent vertices. Our algorithm meticulously re-evaluated the labels of adjacent vertices to ensure that they were assigned to adjacent joint indices, thereby preserving the integrity and continuity of the mesh segmentation. This refinement process played a crucial role in maintaining the logical structure of the segmentation, ensuring that each segment accurately reflected the underlying skeletal structure of the character.

3.2 Hierarchical Spherical Parameterization

For each segment, we implemented a vertex decimation method to simplify the segment down to a fundamental shape and subsequently project it onto a sphere. This process was conducted hierarchically, beginning with the segments tied to leaf joints. A leaf segment is characterized by a single *parent boundary* that connects to the *child boundary* within its parent segment, as illustrated in Figure 3.1. The non-boundary vertices of the leaf segment were reduced to a solitary vertex, termed the **rv** or *representative vertex*. Consequently, the foundational shape of the leaf segment was formed, comprising the **rv** and the vertices along the parent boundary, resembling a pyramid with an open base. This **rv** was then replicated into the parent segment and linked with every vertex of the associated child boundary. Employing this leaf-to-root parameterization strategy enabled the seamless filling of child boundaries in each segment with the **rv**s, allowing vertices on these filled child boundaries to be decimated similarly to the regular vertices. For the root segment, which lacks a parent, its base shape is constituted as a solid tetrahedron.

Subsequently, each base shape underwent projection onto an individual sphere. Following this, the remaining vertices were methodically reintegrated, each being projected to the least-distorted location within their immediate one-ring vicinity, ensuring a precise and coherent reconstruction of the original mesh structure within the spherical domain. This meticulous process underscores our approach's innovative strategy in mesh segmentation and parameterization, facilitating a structured and efficient transformation of complex meshes into their spherical representations.

3.3 Hierarchical Feature Alignment

To achieve precise alignment and facilitate the matching of deforming correspondences between hierarchical sphere structures, we executed a two-phase alignment process.

In the initial phase, our strategy involved applying a rotation matrix to one sphere to align it with another. The calculation of this rotation matrix aimed to minimize the sum of deformation distances, with a specific reference to the child boundary within the parent sphere. For the root sphere, which lacks a parent boundary, we defined the deformation distance as the cumulative spherical distance between each pair of corresponding **rv**s found within the child boundaries. This approach ensured that, for any two corresponding non-root spheres, the spheres were first rotated to position the parent boundaries within the same plane. Subsequently, we rotated the first sphere around the axis perpendicular to this plane section of the parameterized parent boundary. This adjustment aimed to bring the child boundaries of the two corresponding spheres as close to each other as possible, ensuring a more coherent alignment.

The second phase of the alignment focused on precisely positioning the child boundaries, followed by interpolating the remaining vertices using a radial basis function (RBF). This method allowed for a meticulous and exact alignment of the boundaries, ensuring that the transition between corresponding spheres was seamless and accurately reflected the deforming correspondences. Through this two-step alignment, we effectively bridged the gaps between sphere hierarchies, enhancing the fidelity of the deforming correspondences and ensuring a smooth transition across the segmented mesh.

3.4 Applications

This comprehensive approach markedly improves the quality of results across a diverse spectrum of applications, delivering enhanced precision and realism, as shown in Figure 3.2. In the realm of 3D morphing, it facilitates the seamless transformation of one object into another, ensuring smooth transitions and maintaining integrity of the underlying structures. In texture transfer, it allows for the accurate mapping of textures from one model to another, preserving detail and alignment even across complex geometries. For character synthesis, this method enables the creation of richly detailed and varied character models, supporting intricate animations and dynamic interactions. Lastly, in deformation transfer, it ensures that deformations applied to one model are accurately replicated on another, regardless of differences in shape or size. This breadth of application un-

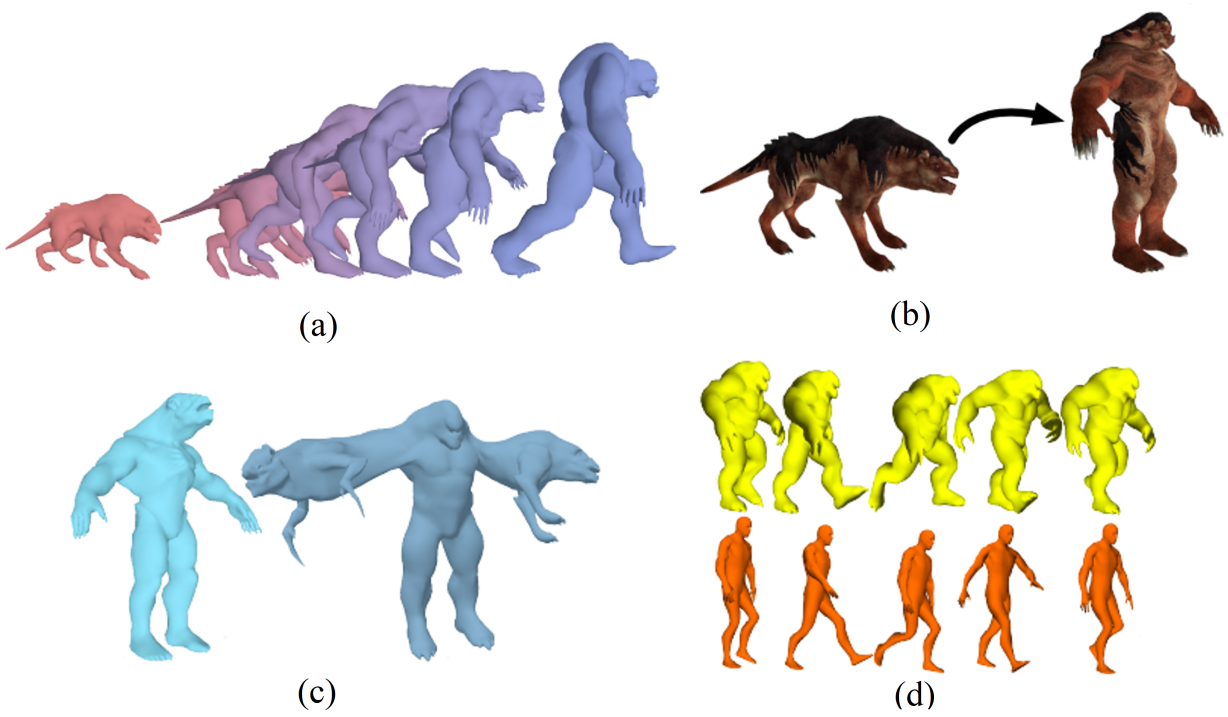


Figure 3.2: Potential applications enabled by our methods: (a) Morphing Effect, showcasing seamless transformation between models; (b) Texture Transfer, demonstrating the application of textures from one model to another; (c) Character Synthesis, highlighting the creation of new character models through the combination of features; and (d) Deformation Transfer, exemplifying the transfer of deformations.

derscores the method's versatility and its potential to revolutionize workflows in animation, game development, virtual reality, and beyond, by offering a level of detail previously unattainable.

Chapter 4

Segmentation and Deforming Feature Identification

As an articulated character animates, each joint in its structure manipulates a specific area of the surface, either altering segments or whole body parts through its movement. For instance, consider the animation of a human character waving their hand. The shoulder, elbow, and wrist joints each play a distinct role in this simple action. In a coordinated manner, the shoulder joint rotates to lift the arm, the elbow joint bends to raise the hand, and the wrist joint twists to wave. This process showcases a complex interaction between the character's skeleton and its mesh surface, which behaves like flexible skin, stretching, compressing, or twisting in response to the bone movements beneath. To precisely determine which mesh parts are influenced by specific joints, we used a part-based mesh segmentation technique, designed to handle the intricacies of character animation.

This technique utilizes skinning weights, a critical element in character rigging, which specifies how much influence each joint has on various parts of the mesh. Skinning weights are assigned to ensure that the mesh deforms smoothly and believably around the joints. Our part-based mesh segmentation method analyzes skinning weights to efficiently divide the character's mesh into distinct segments, optimizing the animation process. This close association between each segment and specific joints allows for precise correlation, ensuring accurate joint movement and mesh deformation. Through this method, the mesh is divided into clearly defined regions, each linked to the transformation properties of individual joints.

4.1 Foundation of Character Articulation

Rigging in the context of 3D animation and computer graphics involves creating the skeleton or structure needed to manipulate a character or object. This skeleton, comprised of joints, enables animators to move the character realistically or as intended. Rigging is where each joint in the skeleton is connected to the mesh in a way that when the joints are moved, the mesh moves with them. This setup allows for the creation of complex movements and poses, from the subtle expressions of a character's face to the dynamic action of a full-body leap. Rigging is both a technical and artistic discipline, requiring a deep understanding of anatomy, physics, and geometry, as well as a keen sense for motion and behavior.

Joints and skinning weights serve as the foundational numerical representations for the deformation properties of a character's mesh, facilitating the precise control over how each part of the character's surface reacts to skeletal movements [64, 69]. Skinning weight quantitatively measures a joint's influence on a specific vertex in the mesh, determining the extent of the vertex's movement in response to joint motion. This influence is represented as a percentage, with a higher percentage indicating a stronger influence of the joint on the vertex's position during deformation. For instance, a skinning weight of 80% for a joint means that the joint dictates 80% of the associated vertex's movement.

These crucial skinning weight parameters can be established through a variety of methods. Procedural approaches involve automated or semi-automated rig estimation techniques that efficiently generate skinning weights and joint associations across mesh sequences or individual frames. By leveraging existing animations or static poses, this process calculates the optimal distribution of weights, ensuring deformations appear natural and realistic [63, 79, 80, 105].

Alternatively, skinning weights and joints can also be crafted by artists, a process that allows for highly customized and nuanced control over character deformation. This manual approach enables animators and riggers to apply their deep understanding of anatomy and motion to create rigs that offer specific artistic or functional advantages [17, 92]. Whether through procedural methods or manual adjustment, the creation and fine-tuning of skinning weights and joints are pivotal in the development of dynamic, realistic character animations.

We followed a similar notation for skinning as described in [100]. Given the set of mesh vertices $\mathbf{V}_d = \mathbf{v}_{i,d}$ in the dress pose of a character, the set of 4×4 dress-pose joint matrices $\mathbf{M}_{j,d}$, and the set of 4×4 joint transformation matrices $\mathbf{M}_{j,c}$ (a canonical pose), a deformed vertex $\bar{\mathbf{v}}_{i,c}$ is expressed as

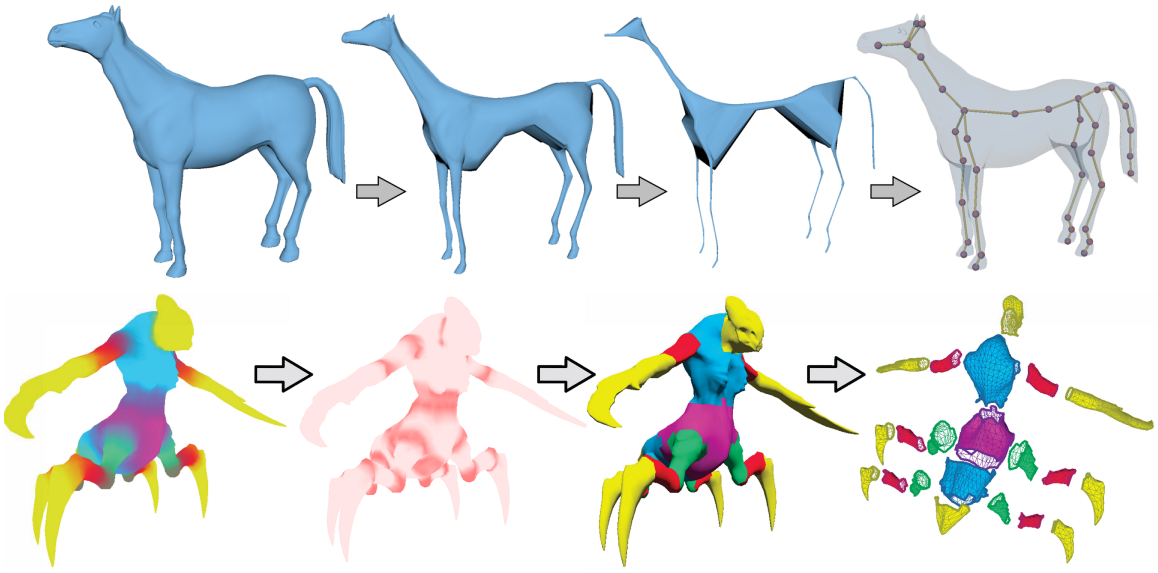


Figure 4.1: The first row shows the process that a skeleton is constructed from the mesh of the Horse and rigged to the Horse using the skeleton extraction algorithm. The second row shows the process that the Spider character, which is rigged, is segmented by the fuzzy decomposition algorithm.

$$\bar{\mathbf{v}}_{i,c} = \sum_{k=1}^{n} w_k \mathbf{M}_{\varrho(w_k),c} (\mathbf{M}_{\varrho(w_k),d})^{-1} \mathbf{v}_{i,d} \tag{4.1}$$

Where w_k is a skinning weight, and $\varrho(w_k)$ returns the joint index corresponding to the weight. The set of weights is a convex combination where $\sum w_k = 1$ and $w_k > 0$.

For the character presented only with meshes, such as mesh sequence data, the skeleton and skinning weights can be generated by the skeleton extraction algorithm [119], like the example in the first row in Figure 4.1.

4.2 Part-based Segmentation

Mesh segmentation stands out as a key technique in dividing character models into distinct body parts, closely reflecting the skeleton's anatomical structure [45, 115, 151]. This method plays an important role in dividing the character's mesh into manageable segments, each matching a specific skeletal part, thus simplifying the animation and rigging processes. Essentially, encapsulating each body part within its own mesh segment permits precise manipulation and deformation, ensuring

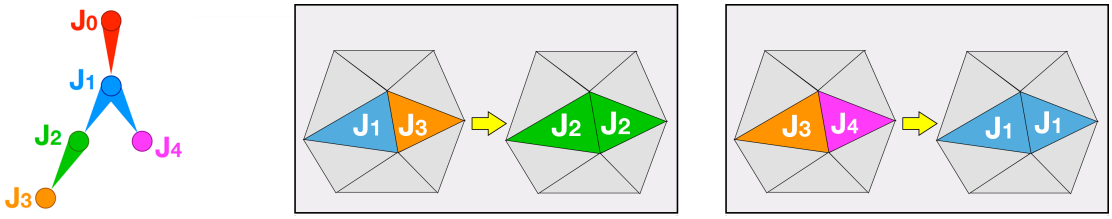
alignment with the character's natural movements.

The process of dividing the mesh into these segments is achieved through sophisticated algorithms designed to recognize and demarcate the boundaries of different anatomical features. A particularly notable approach in this domain is the fuzzy decomposition algorithm introduced by Katz and Tal [68]. This algorithm employs a hierarchical clustering technique that analyzes the mesh's geometry and topology to identify natural divisions within the model. By evaluating factors such as curvature, distance from skeletal joints, and the continuity of surface features, the algorithm can intelligently segment the mesh into coherent parts that correspond closely to the character's physical body parts.

This segmentation is not merely geometric but also considers the underlying skeletal structure, ensuring that each mesh segment is optimally aligned with the corresponding bones. This alignment is important for the subsequent assignment of skinning weights, as it ensures that the deformation of each segment is accurately influenced by the movements of the associated joints. The segments created through this method provide a structured framework upon which skinning weights can be precisely applied.

The fuzzy decomposition method is particularly useful for tasks that require the segmentation of complex structures into more manageable sub-components. This technique employs fuzzy logic principles to handle the inherent ambiguity and uncertainty in defining the boundaries and relationships between different parts of a model. Unlike traditional crisp decomposition methods that assign each element to a single, definitive segment, fuzzy decomposition allows elements to belong to multiple segments with different probabilities. In hence, the use of the fuzzy decomposition algorithm for mesh segmentation represents a significant advancement in the field of character animation and rigging. By facilitating the creation of mesh segments that accurately reflect the character's anatomical structure, animators and riggers are equipped with a powerful tool for producing highly detailed animations.

In this work, we determine the number of segments by setting it equal to the number of joints in the skeleton. Our approach involves assigning probabilities of belonging to a joint to each triangle in the mesh. When computing probabilities, in addition to adopting geodesic distances and angular distances introduced by Katz and Tal [68], we assign each triangle deforming compliance values calculated based on the skinning weights of its three vertices. If a joint does not influence any vertex of a triangle, the triangle's deforming compliance to that joint is zero. If a joint influences more than one vertices of a triangle, the deforming compliance to that joint is the sum of the corresponding skinning weights of the vertices. The distances and the deforming compliance values are calculated once in a pre-processing step using the bind poses of the characters.



Joints in skeleton (a) Ancestral Relation (b) Inosculation

Figure 4.2: The two conditions of triangles that require joint index relabeling in Fuzzy decomposition. The left image shows the reference skeletal structure. (a) illustrates that the triangles in the condition of ancestral relation are relabeled, and (b) illustrates that the triangles in the condition of inosculation are relabeled.

To compute an initial decomposition, we use the k-way iterative clustering scheme (with k equal to the joint count), which is the same method as the work presented by Katz and Tal [68]. We then refine the probabilities of each triangle of belonging to a joint to compute fuzzy regions, and eventually use them to construct exact boundaries between the segments by Laplacian smoothing. The result of fuzzy decomposition for a character is shown in the second row of Fig 4.1. Our assumption for generating the fuzzy decomposition is that triangles that are distant or whose most significant deforming compliance values are associate with different joints are less likely to be clustered into the same segment than closer triangles or those deformed primarily by the same joint.

4.3 Validation and Relabeling

In the process of mesh segmentation, particularly during the decomposition phase, a potential complication arises when two neighboring triangles within the mesh are allocated to joint indices that do not share adjacency. This scenario, illustrated in Figure 4.2, underscores a critical challenge: without careful refinement, the segments derived from this decomposition may not adhere to a tree structure. Instead, they could inadvertently form loops, transforming the segmentation into a graph structure. This shift from a tree to a graph structure poses significant difficulties, as the inherent hierarchical relationship between segments—a cornerstone of our algorithm—could be compromised.

To mitigate this issue and maintain the integrity of the segmentation process, our methodology integrates a validation step. During this phase, we examine the labels assigned to neighboring triangles. This scrutiny is aimed at verifying that these labels correspond to adjacent joint indices, thereby preserving the contiguity essential for a coherent tree structure. Let's say a triangle T_1 is

Algorithm 1 Fix Ancestral Relation

```
1: procedure FixAncestral(mesh)
2:     for i = 0 to mesh → tn − 1 do
3:         t1 ← mesh → triangles[i]
4:         for j = 0 to t1 → neighbors.size() − 1 do
5:             t2 ← t1 → neighbors[j]
6:             if GetRelationship(t1 → segmentID, t2 → segmentID) = DESCENDANT then
7:                 t1 → new_segmentID ← t2 → new_segmentID ← FindAncestorChild(t1 → segmentID, t2 → segmentID)
8:                 break
9:             end if
10:        end for
11:    end for
12: end procedure
```

Algorithm 2 Fix Inosculation Condition

```
1: procedure FixInosculation(mesh)
2:     for i = 0 to mesh → tn − 1 do
3:         t1 ← mesh → triangles[i]
4:         for j = 0 to t1 → neighbors.size() − 1 do
5:             t2 ← t1 → neighbors[j]
6:             if GetRelationship(t1 → segmentID, t2 → segmentID) = SIBLING then
7:                 t1 → new_segmentID ← FindNearestCommonAncestor(t1 → segmentID, t2 → segmentID)
8:                 break
9:             end if
10:        end for
11:    end for
12: end procedure
```

labeled with joint j_1, and it has an adjacent triangle T_2 labeled with joint j_2. The two conditions of triangles that require relabeling are described below:

- *Ancestral relation:* If j_1 is an ancestor of j_2's parent, both T_1 and T_2 are relabeled with the index of j_1's child which should also be j_2's ancestor.
- *Inosculation:* If j_1 and j_2 are the joints on different branches in the skeletal structure, both T_1 and T_2 are relabeled with the index of the nearest common ancestor joint.

The pseudo-code of relabeling are shown in Algorithm 1 and 2. By implementing this validation step, we preempt the formation of loops and the resultant graph structure, thereby safeguarding the hierarchical organization critical for our algorithm's success. This validation extends beyond a basic check to become an integral component of the segmentation process, guaranteeing that each segment aligns precisely with the underlying skeletal structure. The justification for this method lies in the necessity of a hierarchical, tree-like structure, which simplifies the animation process and supports the implementation of further algorithms. The tree structure provides a distinct, straightforward definition of parent-child relationships among segments, essential for precisely animating articulated characters. This approach ensures deformations follow a logical path along the skeleton, accurately mirroring natural movement patterns.

4.4 Boundaries as Deforming Features

In this work, we use the *boundaries* between parent and child segments as deforming features. These boundaries are not merely static demarcations but serve as dynamic deforming features, pivotal in the articulation process.

In the skeleton of a deforming character, joint transformation matrices (described in Section 4.1) act like deflection forces that change the shape of the character. This method ensures that a segment's deformation is inherently linked to and constrained by its neighboring segments. The boundaries between these segments emerge as critical zones, marking the areas where a joint's influence on the mesh surface sees its most significant gradient. This gradient is a direct reflection of how deformation propagates across the character's surface, transitioning from one segment to another. Such boundaries are invaluable for capturing the essence of deformation transmission, showing how motion and force are distributed across the character's anatomy.

Utilizing these boundaries as deforming features allows us to extract and analyze the complex

interplay of deforming elements across the character's surface. This method facilitates a more granular control over the deformation process, akin to isolating locally approximated near-rigid components within a predominantly non-rigid motion framework [83, 97, 146, 148]. By applying this analogy to character animation, we can achieve a more detailed understanding of how segments deform in relation to each other and the skeleton as a whole. This understanding is crucial for creating animations that not only look realistic but also faithfully replicate the complex dynamics of real-world movement and deformation.

To perform hierarchical parameterization and feature correspondence alignment, we utilized the boundaries extracted from the segmented mesh. The boundaries define a self-contained zone that encompasses the surface details of each segment, and they can be used to establish parent-child correlations (Chapter 5). By parameterizing the surface details of each segment onto a coarse geometric shape within this zone, we can achieve a hierarchical representation of the character. Moreover, we utilized the boundaries as deforming features to constrain the alignment process in the parametric domain, thereby reducing the occurrence of twisted geometries when synthesizing a new character (Chapter 6).

Chapter 5

Building Hierarchical Spherical Representations

We parameterized the mesh segments into hierarchical spherical representations. The segmentation in Section 4.2 results in three types of segments as shown in Figure 5.1: (1) a *tube* is a cylinder-like shape with top and bottom boundaries, one connecting to the parent mesh segment while the other connecting to the only child mesh segment; (2) a *bag* is the mesh segment associated to an end joint, and it has only one boundary that connects to the parent; and (3) a *vest* contains multiple boundaries, where one connects to the parent mesh segment, and each of others represents a connection to a child mesh segment.

Existing solutions often handle different types of segments with different parameterization methods, such as fitting tubes onto a cylinder domain and cutting bags and vests onto a planar domain [23], or mapping individual segments onto non-regular approximation domains [145]. However, the use of different parametric domains can be difficult for re-stitching the mesh segments to create a synthesized shape as a whole or transfer deformations.

Our parameterization method provides a universal solution by mapping them onto a hierarchy of spheres. The method can reduce the parametric distortion and benefit feature correspondence applications. Besides, compared to the non-regular approximation domains for cross-parameterization, a spherical domain ensure the mesh is mapped with a convex embedding, which is more flexible for the characters with complex body parts. One advantage of our method is the ability to retain convex, continuous parameterization on spheres for the mesh segments that have “holes" created by

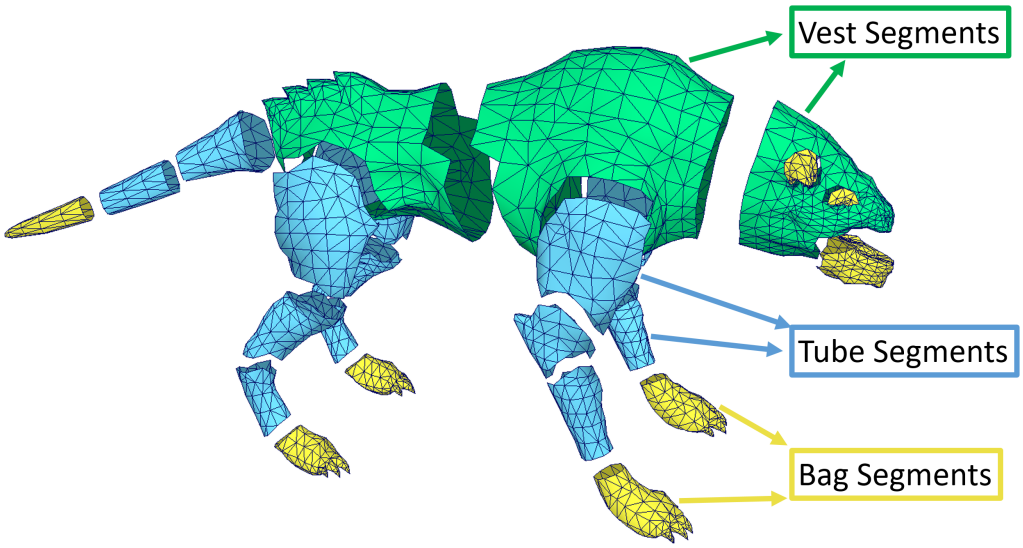


Figure 5.1: The segmentation outcome for the Ripper Dog distinct segment types with color-coded classifications: tube segments are depicted in blue, bag segments in yellow, and vest segments in green.

boundaries. The boundaries can be represented as circles on the sphere and aligned with those on the spheres of neighboring segments.

5.1 Child-to-Parent Decimation

For each segment, we implemented a vertex decimation method to collapse the segment down to a base shape and project it onto a sphere, as shown in Figure 5.2. The child-to-parent decimation starts with *bag* segments which are associated with leaf joints. To better communicate our ideas, we denote the boundaries on a segment as $\mathbf{B}_{chd}$, $\mathbf{B}_{pat} \in \mathbb{R}^3$ for a child boundary and the parent boundary, respectively. $\mathbf{B}_{chd} = \{\mathbf{BV}_{chd}, \mathbf{BE}_{chd}\}, \mathbf{B}_{pat} = \{\mathbf{BV}_{pat}, \mathbf{BE}_{pat}\}$, where $\mathbf{BV}_{chd}, \mathbf{BE}_{chd}, \mathbf{BV}_{pat}, \mathbf{BE}_{pat}$ are the sets of boundary vertices and boundary edges that form a single-cycle edge loop in each respective boundary type. A mesh segment has a set of boundaries $\{\mathbf{B}_{pat}, \mathbf{B}_{chd_1}, \mathbf{B}_{chd_2}, \cdots, \mathbf{B}_{chd_n}\}$, n is the number of child joints of the joint this mesh segment associates with.

In the process of decimation, we used the half-edge collapsing method [75] to collapse one vertex of the edge to the other until reaching the base shape. The approach involves iteratively merging pairs

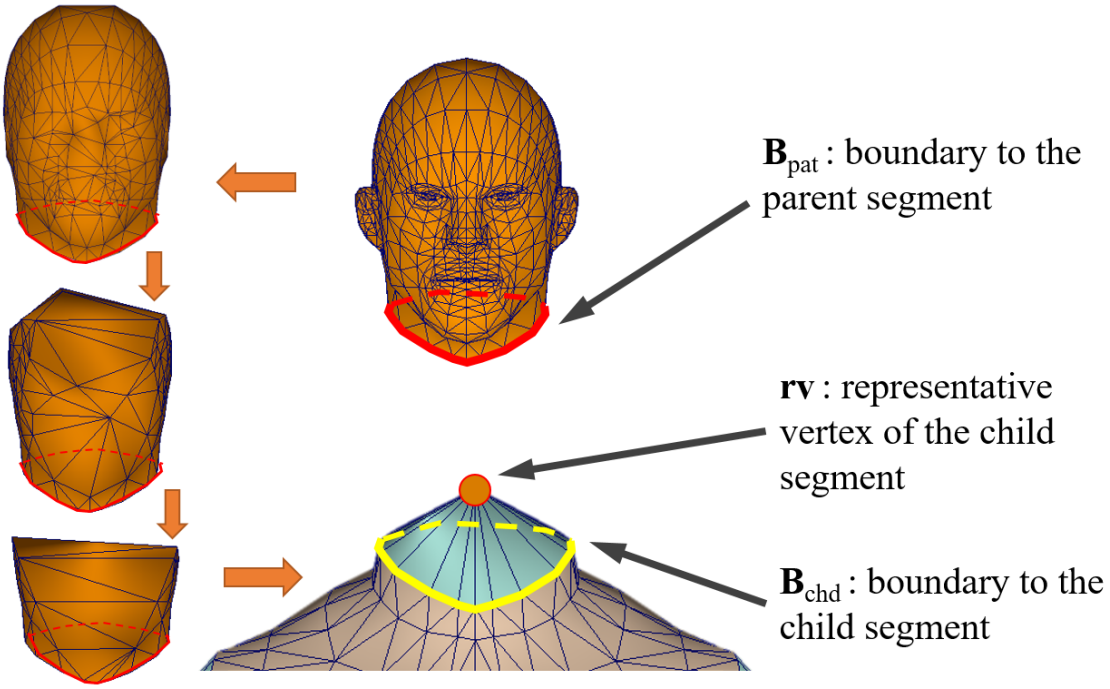


Figure 5.2: The boundaries between segments are denoted as **B***chd* and **B***pat*, representing the boundaries connecting to the child and parent segments, respectively. The *bag* segment associated with the leaf joint is decimated into a *representative vertex*, denoted as **rv**. Once all child segments are decimated into **rv**, the resulting *tube* or *vest* segment becomes a *bag* segment, and the algorithm can be applied again to process it.

of vertices towards the midpoint—or "half"—of an edge. In this process, the collapsing criterion is a set of metrics that guide the selection of edges for collapsing, based on factors such as edge length, surface curvature, visual importance, or error metrics that estimate the deviation from the original mesh geometry [6]. For instance, shorter edges might be prioritized for collapsing to minimize geometric distortion, or edges that contribute least to the overall shape or feature details might be selected to maintain the visual fidelity of the model [42, 43, 86]. Additionally, the criterion may incorporate aspects like maintaining manifoldness, avoiding distortion, or preserving critical feature lines. The careful formulation of the collapsing criterion is crucial, as it directly influences the quality, efficiency, and applicability of the mesh simplification.

Our collapsing criterion is in favor of removing rich features (e.g., high curvatures, dense point sets) as early as possible, so that they are embedded into a more flattened region that is more suitable for mapping to the spherical domain. The collapsing criteria is a cost function on an edge. Given an edge denoted as $\mathbf{e}(\mathbf{v}_1, \mathbf{v}_2)$, the cost function for collapsing vertex $\mathbf{v}_1$ to $\mathbf{v}_2$ is presented in Equation 5.1.

$$cost(\mathbf{v}_1 \rightarrow \mathbf{v}_2) = \frac{\sum_{\mathbf{t} \in neighbor(\tilde{\mathbf{v}})} AR(\mathbf{t})}{g(\mathbf{v}_1)} \tag{5.1}$$

$\tilde{\mathbf{v}}$ is the vertex if the collapsing is performed. $AR(\mathbf{t})$ returns the aspect ratio of the triangle $\mathbf{t}$. The equation calculates the sum of aspect ratios of all triangles in the neighborhood of $\tilde{\mathbf{v}}$. $g(\mathbf{v}_1)$ returns the Gaussian curvature at vertex $\mathbf{v}_1$.

In our method, edge collapsing has a constraint that does not allow a vertex to move onto a boundary vertex in $\mathbf{BV}_{pat}$. This is to prevent modifying the boundary vertices or having the boundary embed a non-boundary vertex.

Algorithm 3 elaborates the recursive steps of the mesh decimation. A mesh segment $\mathbf{Seg}_i$ is the input of the algorithm. A segment associated to a child joint of the $\mathbf{Seg}_i$'s joint can be accessed by $\mathbf{Seg}_i.\mathbf{Seg}_{chd_j}$. All base shapes are built recursively by initializing the call $Decimate(\mathbf{Seg}_0)$, where $\mathbf{Seg}_0$ is the mesh segment of the root joint. The algorithm first traverses down to leaf nodes (line 3), which are *bag* segments. For a *bag* segment, edges for $\mathbf{e} \notin \mathbf{BE}_{pat}$ are collapsed iteratively and decimated to a single vertex that we call the *representative vertex*, denoted as $\mathbf{rv}$. The base shape is created with the $\mathbf{rv}$ connecting to each vertex in $\mathbf{BV}_{pat}$ (lines 6-9). This base shape is a pyramid with an open bottom. As the child-to-parent decimation process continues to the next upper-level segment (the parent), the $\mathbf{rv}$ is duplicated to the parent segment and connects to every vertex in

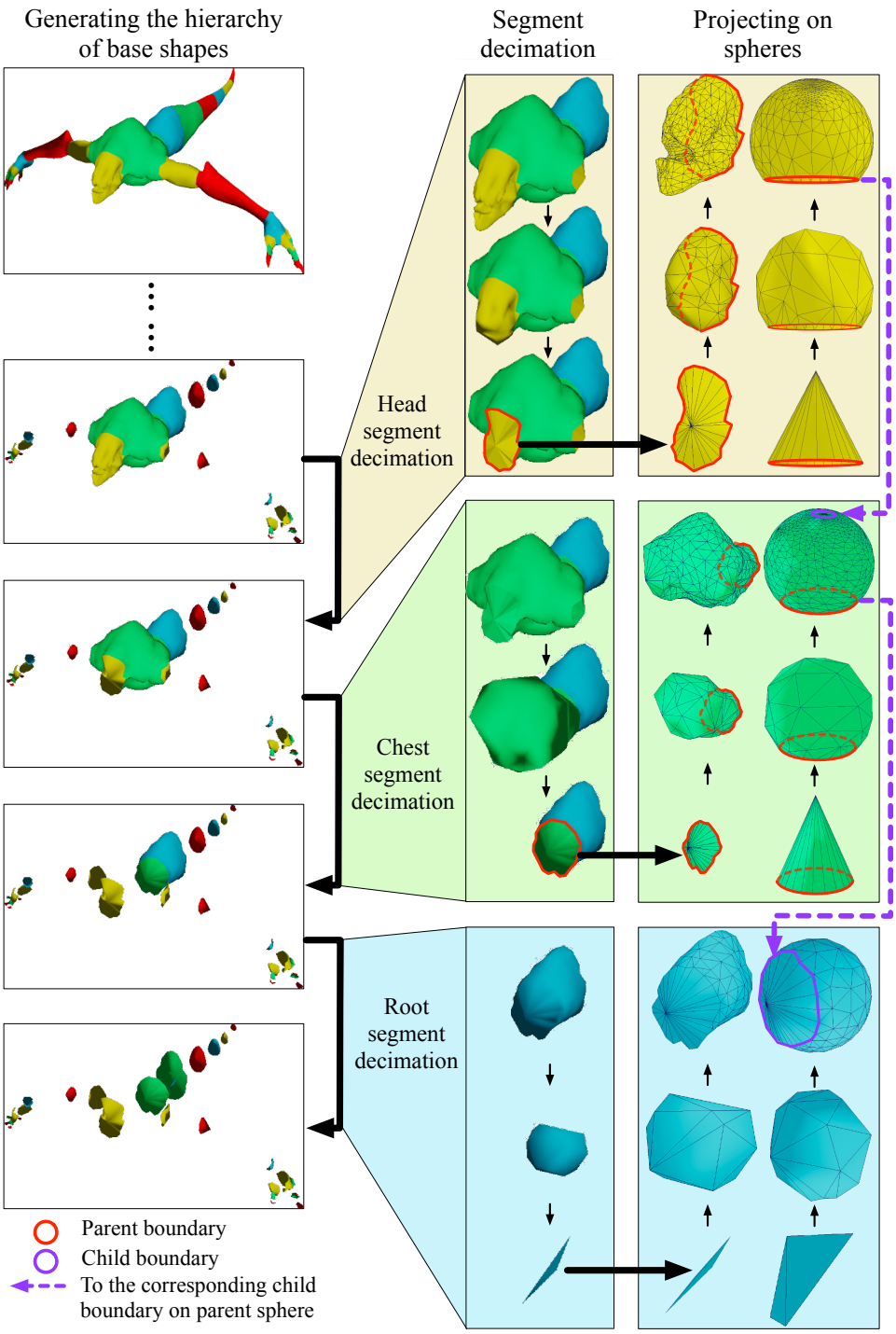


Figure 5.3: A parameterization example with the Creepy character. The left column shows the character is child-to-parent decimated to a hierarchy of base shapes. The middle column presents the intermediate results of segment decimation towards obtaining a base shape. The right column shows the process of building the spherical representations of the segments by first projecting the base shapes onto the sphere and then inserting vertices back in the reverse order of decimation. The base shapes of non-root segments are projected to a pyramid with the parent boundary defining the open bottom, and the base shape of the root segment is a solid tetrahedron.

Algorithm 3 Decimation

```
1: procedure DECIMATE(Seg_i)
2:     for the jth child of Seg_i do
3:         rv_{chd_j} ← Decimate(Seg_i.Seg_{chd_j});
4:         Triangulate(rv_{chd_j}, B_{chd_j});                ▷ Form a 1-ring fan
5:     end for
6:     BaseShape ← CollapseEdges({V, E} − B_pat);
7:     Save BaseShape to disk;
8:     if Seg_i is not associated to the root joint then
9:         return rv in the BaseShape;
10:    else
11:        return null;                                       ▷ All base shapes are finished
12:    end if
13: end procedure
```

the corresponding $\mathbf{BV}_{chd_j}$ such that the $\mathbf{B}_{chd_j}$ is filled and triangulated to form a 1-ring fan (lines 2-5). All child boundaries can be filled as 1-ring fans, so *tubes* and *vests* are converted to *bags* and eventually decimated to pyramids. For the root segment, the base shape is a solid tetrahedron. An example is shown in Fig. 5.3 (middle column).

5.2 Projecting onto Sphere Surfaces

The primary objective of spherical parameterization is to establish a bijective mapping between the vertices of a mesh in three-dimensional space and the vertices on the surface of a sphere, denoted as $\mathbb{S}^2$. This mapping, represented by the function $\eta : \mathbf{V}, \mathbf{E} \in \mathbb{R}^3 \mapsto \mathbf{V}^s, \mathbf{E}^s \in \mathbb{S}^2$, is designed to preserve the original topology of the mesh. Such preservation ensures that the spatial relationships and connectivity between vertices ($\mathbf{V}$ and $\mathbf{E}$) are faithfully maintained in their spherical representations ($\mathbf{V^s}$ and $\mathbf{E^s}$).

This technique facilitates the independent projection of decimated mesh segments onto the surface of a sphere, allowing for a localized analysis and manipulation of complex geometries. As illustrated in Figure 5.3 (specifically in the right column), each segment of the mesh can be individually mapped onto a sphere, demonstrating the versatility and precision of spherical parameterization.

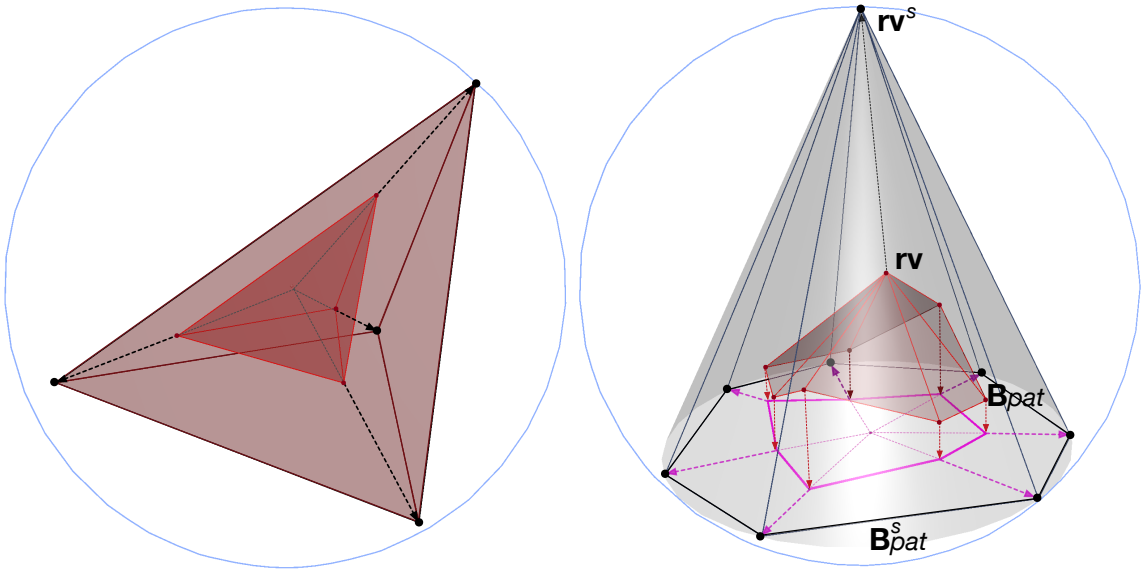


Figure 5.4: The illustrations showing how base shapes are projected onto the sphere. The left one is projecting the solid tetrahedron base shape of a root segment. The right one is projecting the pyramid base shape of a non-root segment onto the right circular cone inscribed in the sphere.

Figure 5.4 shows the methods to project the base shapes on the sphere. For the root mesh segment, the tetrahedron is first centered inside the sphere, and the rays are emitted from the center of the sphere and pass through the vertices of the tetrahedron. Each ray casts the vertex at the intersection point with the spherical surface.

For a non-root mesh segment, the pyramid geometry is parameterized into a right circular cone that maximizes the volume inscribed in the sphere. To do this, we first use the singular value decomposition [46] to find the local least-square plane based on the vertices in $\mathbf{BV}_{pat}$. The right circular cone is oriented to align with the direction of the plane's normal vector, and moved such that the base circle of the cone is on the plane. Then, we projected the vertices in $\mathbf{BV}_{pat}$ onto the plane, and then further projected them onto the base circle of the cone. The **rv** is moved to the cone's apex position. After that, the parameterized pyramid is rotated such that the plane is aligned to the joint's dress-pose orientation.

The rest of vertices are inserted back one by one and projected to the location inside the 1-ring region on the spherical surface, as shown in Figure 5.5. Reintroducing a vertex into a mesh establishes connections with its neighboring vertices, thereby creating a structure known as a 1-ring domain, which resembles a 3D umbrella in form. This process of adding a vertex effectively reverses the

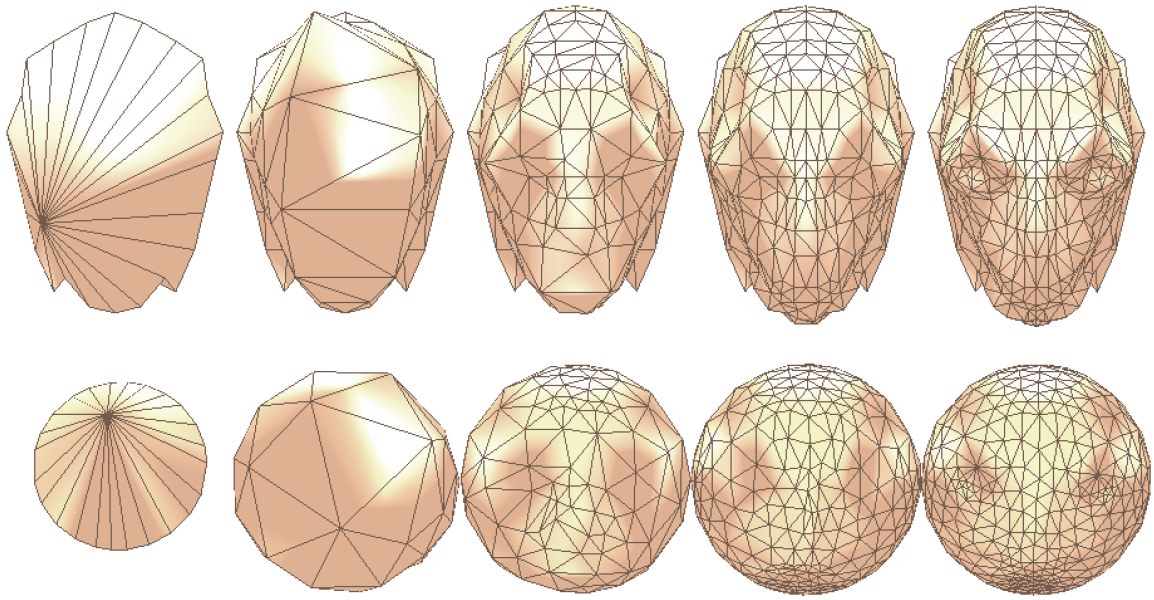


Figure 5.5: Projection of inserted vertices on the Alien Head segment onto a sphere. The top row depicts intermediate 3D meshes, showcasing the gradual recovery process as vertices are reintroduced. The bottom row illustrates the spherical parameterizations, highlighting how these inserted vertices are mapped onto the sphere.

operation of edge collapsing. For instance, referring to Figure 5, what appears as vertex v_2 in the right image would be divided into two separate vertices, v_1 and v_2, in the left image. Consequently, splitting the vertex in this manner generates a 1-ring domain around v_2. The intersection of open hemispheres, delineated by the edges of the spherical polygon that result from projecting the 1-ring edges in relation to the sphere's centroid, forms a spherical kernel. As detailed by Praun and Hoppe, this kernel constitutes a spherical polygon on the sphere's surface, ensuring the preservation of a valid embedding provided the vertex's projection falls within the spherical kernel.

To efficiently determine the optimal position of a vertex, a barycentric interpolation sampling technique is utilized, avoiding the direct sampling within the spherical kernel due to potential interpolation issues and high computational demands. Instead, sampling occurs within the parameterized triangles of the 1-ring using the formula:

$$p = t_1 \cdot v_1 + t_2 \cdot v_2 + (1 - t_1 - t_2) \cdot v_3$$

Here, p represents a point interpolated from the triangle's three vertices v_1, v_2, and v_3, with t_1 and t_2 being fractions within the range (0,1) and their sum not exceeding 1. These fractions adjust based on a predetermined step value.

However, some sampled points may lead to triangle foldovers if they fall outside the spherical kernel and thus need to be excluded. This determination is made by testing the sample points against the planes defined by the 1-ring edges and the sphere's centroid. A sample point is deemed valid if it lies above any of these planes; otherwise, it is considered outside the kernel and is discarded. Valid sample points, when connected by rays from the centroid to the sphere's surface, intersect at locations termed as kernel points. These points represent potential positions for vertex placement.

Equation 5.2 is used to measure the parametric distortion when projecting the vertex.

$$
\begin{aligned}
d(v) &= \sum_{\mathbf{t} \in neighbor(\mathbf{v})} \left(\frac{kArea(\mathbf{t}^s)}{Area(\mathbf{t})} + \frac{Area(\mathbf{t})}{kArea(\mathbf{t}^s)}\right)^{\theta_1} AR(\mathbf{t}^s)^{\theta_2}, \\
\text{where } k &= \frac{\sum_{\mathbf{t} \in neighbor(\mathbf{v})} Area(\mathbf{t})}{\sum_{\mathbf{t} \in neighbor(\mathbf{v})} Area(\mathbf{t}^s)}
\end{aligned}
\tag{5.2}
$$

This distortion measurement equation considers the area distortion and the regularity after projecting one vertex. $\mathbf{v}^s$ is the inserted vertex that has been projected on the sphere. $\mathbf{v}$ is the inserted vertex on the 3D mesh. Similarly, $\mathbf{t}^s$ is a triangle in the spherical domain and $\mathbf{t}$ is a triangle on the 3D mesh. k represents the ratio between the area of the 1-ring kernel on the 3D mesh and the area on the sphere. In this work, we set $\theta_1 = 1$ and $\theta_2 = 4$.

5.3 Optimization

Continuously adding vertices to the sphere will result in the densification of specific areas [106]. Over time, this leads to a distribution where small regions are heavily populated with geometric features, while larger areas remain sparse, making it challenging to maintain feature integrity in these less dense zones. The optimization of vertex distribution on the spherical surface plays a crucial role in enhancing the fidelity and accuracy of spherical parameterization processes.

To address this imbalance, we have devised a global optimization strategy aimed at effectively redistribating density. This method focuses on relaxing the overcrowded areas and consolidating the features in the sparsely populated regions, ensuring a more uniform distribution of geometric characteristics across the sphere.

This optimization is carried out periodically, especially as the count of projected vertices on the

Algorithm 4 Vertex Optimization Process

```
1: while not done and iterations < maxIterations do
2:     iterations ← iterations + 1
3:     Update verticesbyneighbourchange
4:     Sort vertices by neighbor change
5:     for all vertices do
6:         if NeedsOptimization(v) then
7:             Optimize(v, sample_step)
8:         end if
9:         if v.displacement > maxDisplacement then
10:            maxDisplacement ← v.displacement
11:        end if
12:    end for
13:    if maxDisplacement < minOffset then
14:        done ← true
15:    end if
16: end while
```

sphere reaches specific thresholds defined by a geometric progression. This strategy ensures that the distribution of vertices remains uniform and optimal, even as the complexity of the mesh increases. The optimization process is shown in Algorithm 4.

$$AMSE(\mathbf{v}_i^s) = \frac{1}{n} \sum_{\mathbf{v}_j \in \text{neighbor}(\mathbf{v}_i)}^{n} \text{arc}^2(\mathbf{v}_j^s - \widehat{\mathbf{v}_j^s}) \tag{5.3}$$

To achieve an optimal distribution, the projected vertices undergo a re-positioning process aimed at minimizing the Arc Mean Squared Error (AMSE) relative to their neighboring vertices. The AMSE for a vertex $\mathbf{v}_i^s$ is calculated in Equation 5.3, where $arc(\mathbf{v}_j^s - \widehat{\mathbf{v}_j^s})^2$ represents the squared arc distance between the current position of a vertex $\mathbf{v}_j^s$ and its optimal position $\widehat{\mathbf{v}_j^s}$, and n is the number of neighbors. This measure effectively captures the local distortion around each vertex, providing a quantitative basis for adjusting the vertices' positions to reduce overall surface distortion. The implementation of optimization method is shown in Algoirthm 5.

The goal of minimizing AMSE is to ensure that each vertex is positioned as accurately as possible relative to its neighbors, thereby preserving the mesh's local and global geometric properties. This optimization process not only aids in maintaining the integrity of the original model's features but

Algorithm 5 Optimize Vertex Distribution

```
1: RE_GENERATE_VALID_FEATURE_POINTS()
2: position_old ← v.parameterization_position
3: for i ← 0 to samplePoints_valid.size do
4:     v.parameterization_position ← samplePoints_valid[i]
5:     err ← CALCULATEPROJECTIONERROR(v)
6:     if err < minError then
7:         minError ← err
8:         m ← i
9:     end if
10: end for
11: if m exsits then
12:     v.parameterization_position ← sphere_r × NORMALIZE(samplePoints_valid[m])
13:     v.is_parameterization ← true
14: end if
15: v.displacement ← DISTANCE(position_old, v.parameterization_position)
16: for all neighbor in v.neighborVertexs do
17:     neighbor.UPDATENEIGHBORDISPLACEMENT
18: end for
```

also significantly improves the parameterization quality by ensuring that the spherical mapping accurately reflects the mesh's topological and geometric characteristics.

Chapter 6

Hierarchical Deforming Feature Alignment

Our method parameterizes deforming characters into hierarchical spherical representations. By encapsulating their complex geometries within a more manageable, spherical framework, these representations serve as intermediaries. This facilitates the uniform parameterization of diverse characters and is essential for the use of cross-parameterization technologies. Such technologies are crucial for sophisticated animation effects, including mesh morphing.

Once the character models are parameterized within this common domain, a critical step follows: aligning corresponding features across the models. This alignment is essential for ensuring that significant features and structural elements of the characters match up correctly. For instance, it is crucial that the eyes, mouths, and limbs of different characters are properly aligned to avoid distortions or anomalies in the resulting animation. The process of feature alignment involves identifying and mapping key points or landmarks on each character model that correspond to similar points on the other models. By establishing these correspondences, we can create a set of rules or mappings that guide the morphing process, ensuring that as one character transitions into another, their features transform in a coherent manner.

For deforming characters, during the feature alignment phase, the main focus is on the deforming features that define the animation's dynamics. The critical advantage of our parameterization lies in its hierarchical approach to handling these deforming features. Our hierarchical parameterization excels by providing a structured framework that captures the complexity of these deforming features

at various levels of detail. This multi-level approach allows for a more stable control over the animation process, ensuring that the motion and transformation are accurately represented.

This hierarchical organization is particularly advantageous when aligning features across characters for cross-parameterization techniques. It allows us to match not just the static positions of key landmarks, but also the dynamic behaviors of deforming features across different models. Moreover, the hierarchical nature of our parameterization facilitates the identification and alignment of corresponding deforming features across characters by providing a clear, organized structure that guides the matching process.

6.1 Sphere Merging based on Major Joints

In the field of character animation and morphing, the flexibility of our method is evident because it does not require the input characters to have identical skeletal structures to be successfully morphed or animated together. This flexibility is essential, considering the broad range of character designs and skeletal structures found in animation projects, from human figures to fantastical creatures. Nonetheless, for a smooth and consistent morphing process, it is crucial to set a standard for matching the characters' skeletons.

In this work, this is achieved through the identification of a coarse set of joint pairs, which we refer to as the *major joint* pairs. These pairs serve as the anchors for the alignment processes, ensuring that key anatomical and structural features are preserved across the different characters.

Our method simplifies this task by requiring user intervention to annotate these major joint pairs between the input characters. Despite the potential complexity of the task, it is designed to be straightforward, relying on basic understandings of the characters that most users can easily acquire. This user-guided annotation process ensures that the essential joints critical for maintaining the character's structure and motion dynamics are correctly identified and matched. The **major joint** pairs typically include the root joint, which is fundamental for defining the character's overall orientation and position in space, along with additional joints specified by the user based on the characters' anatomy and the specific requirements of the animation.

The sphere of a non-major joint is merged into its parent sphere until eventually merged into the sphere of a major joint. On that parent sphere, the set of $\{\mathbf{rv}^s_{chd_j}, \mathbf{B}^s_{chd_j}\}$ forms a triangle fan, which is a "shield" region that the sphere of the non-major joint (the child) can be merged into. One way to conduct such child-to-parent merging is using the stereographic projection [101, 126] to embed a

spherical surface into the shield region; however, the parametric distortion around the **rv** would be high. In our approach, the sphere merging process is equivalent to splitting the **rv**. Similar to the projection in Section 5.2, the latter collapsed vertices in the child mesh segment are merged into the shield region on the parent sphere earlier than those collapsed earlier. AMSE-based optimization is applied during the merging process.

After merging, the vertex density in the shield region increases, but none of the merged vertices in the shield region would be treated as feature points, so that they will not influence the process of feature alignment. In addition, the merging retains locality and monotonicity, and such local embedding is represented as a manifold spherical surface with the skin weights monotonically decreasing toward zero near the edge of the shield region. Upon completion of the merging process, both input characters share identical hierarchical spherical representations.

6.2 Alignment

Deforming features, as defined in Chapter 4, are the boundaries of the segments. After the spherical parameterization, such deforming feature points are the boundary vertices of a sphere. Like the notation in Section 5.1, these boundary vertices of a sphere are $\mathbf{BV}^s_{pat}$ and $\{\mathbf{BV}^s_{chd_1}, \mathbf{BV}^s_{chd_2}, \cdots, \mathbf{BV}^s_{chd_m}\}$, where the superscript s denotes the spherical domain. Note that the spheres of non-major joints have been merged. In the process of alignment, the spheres are those associated to the major joints. Given the sphere hierarchies of two characters, two spheres from different hierarchies and associated to the same major joint are called *corresponding spheres*. The alignment aims to:

- Align the orientations of all corresponding spheres through the hierarchies (Section 6.2.1).
- Align the boundary vertices between the corresponding spheres to the exact target positions (Section 6.2.2).

For a deforming character, a joint inherits the transformation from the parent joint, which is multiplied by this joint's local transformation and passed to the child joints. Thus, the alignment of a pair of corresponding spheres will affect the alignments of their parent and child pairs of corresponding spheres. However, existing feature alignment methods have not explored such parent-child correlations (e.g., relying solely on global geometric features like curvatures [106] or relying fully on users to determine the part correspondences [23]).

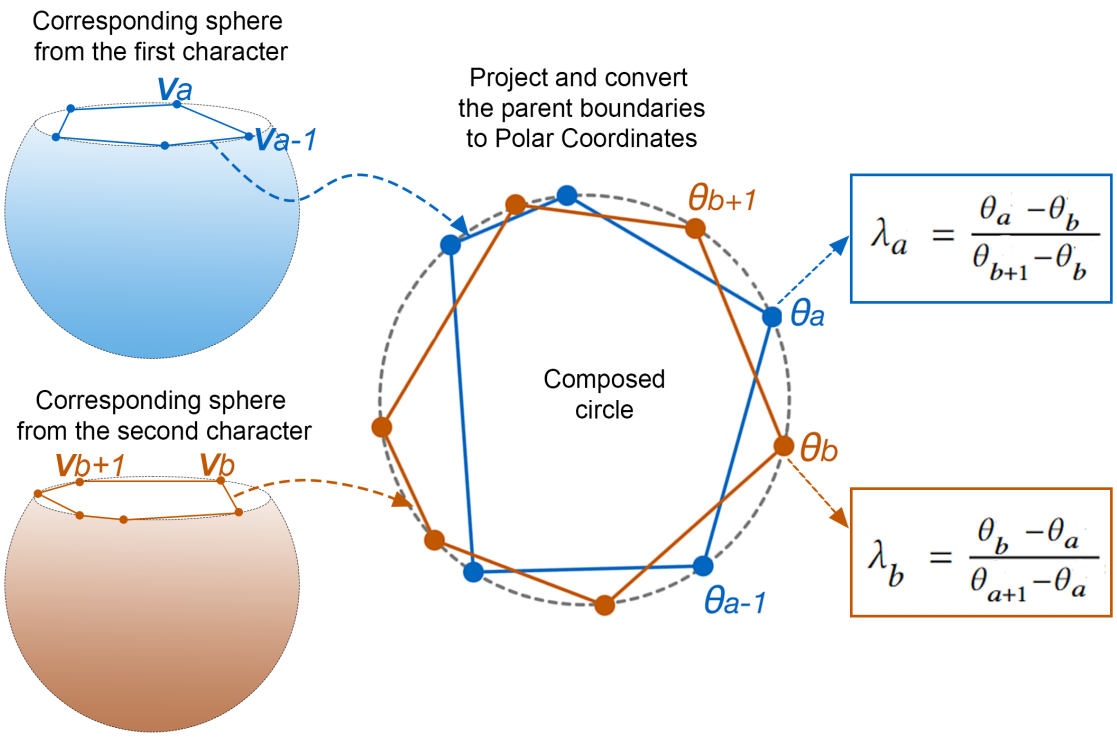


Figure 6.1: For two non-root corresponding spheres, the illustration of projecting the parent boundaries to the polar coordinate and calculate the ratio λ of its two allopatric neighboring vertices, which are used to calculate the target position in the following process.

We conducted a two-step alignment to coincide deforming feature correspondences between the sphere hierarchies. In the first step, we find a best-fit rotation matrix for every pair of corresponding spheres, which can perform the rigid alignment to the orientation of the spheres. The rigid alignment aims to bring the deforming features of one sphere as close as possible to the features on the other sphere it corresponds with. Then, based on the rotation matrix, the target positions of the feature points are calculated. In the second step, the feature points are moved to the exact target positions. Accordingly, the rest of vertices on the spheres are displaced along the sphere surface direction by the radial basis function interpolation.

6.2.1 Parent-to-Child Rigid Alignment

The process of alignment initiates at the root spheres of the two hierarchical structures and progresses iteratively towards the leaf spheres. Notably, root spheres are unique in that they lack

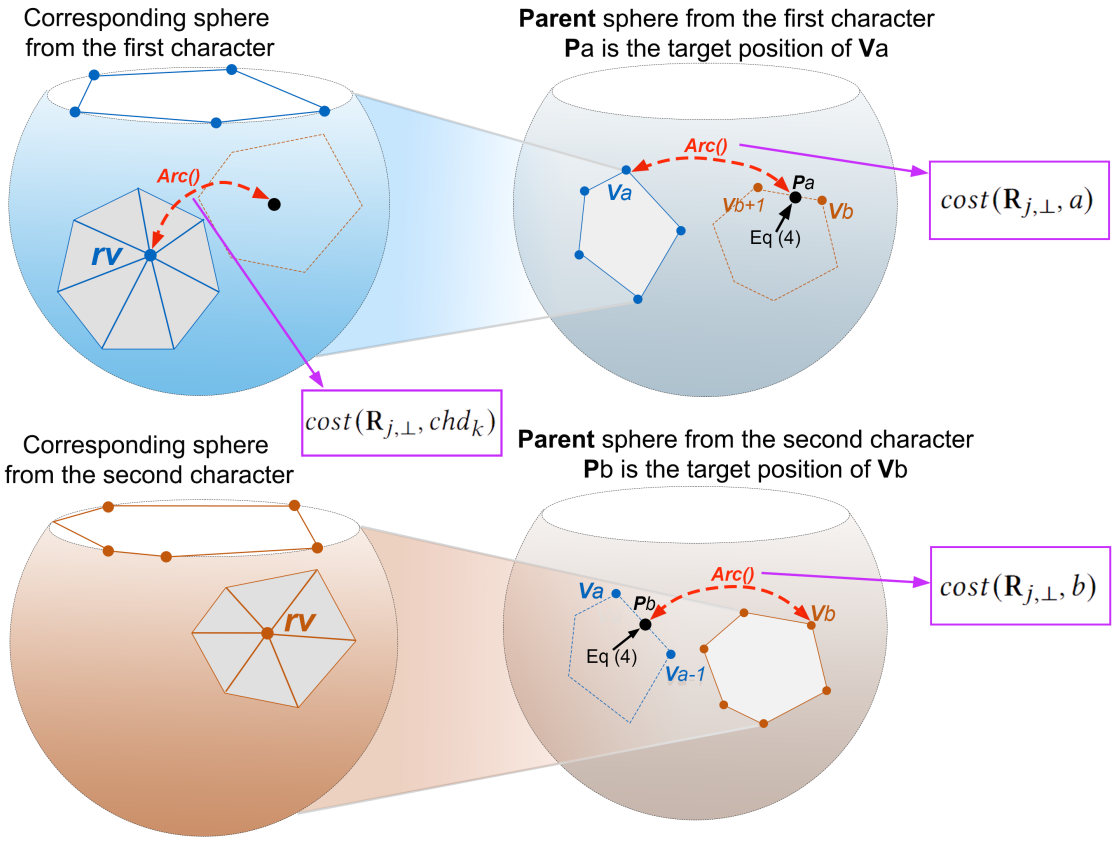


Figure 6.2: The illustration of the alignment cost evaluation for the corresponding non-root spheres during the process of the rigid alignment. The target positions $\mathbf{p}_a^{s_1}$ and $\mathbf{p}_b^{s_s}$ are estimated from allopatric neighbor vertices by Equation 6.3. $Arc()$ returns an arc distance between a vertex and its target position.

parent boundaries, setting them apart in the alignment procedure.

For root spheres, the alignment cost is quantitatively defined by aggregating the arc distances between corresponding pairs of representative vertices, denoted as $\mathbf{rv}^{s_1}chd_j$ and $\mathbf{rv}^{s_2}chd_j$. Arc distance is preferred for its precision in measuring distances on a spherical domain, offering more accurate interpolation results, as highlighted by Smolik et al. [130]. The goal is to minimize this alignment cost, as formulated in Equation 6.1, to derive the optimal rotation matrix $\mathbf{R}_{root}$. This matrix facilitates the precise alignment of the root sphere from the second hierarchy s_2 to that of the first hierarchy s_1. The variable m represents the total count of child spheres associated with the root, reflecting the breadth of connectivity and the complexity of the hierarchical structure at its base level. Through this alignment process, we ensure a coherent and structurally sound integration of the hierarchical models, laying a solid foundation for subsequent alignments throughout the hierarchy.

$$\underset{\mathbf{R}_{root}}{\mathbf{arg\ min}} = \frac{1}{m}\sum_{j=1}^{m} arc(\mathbf{rv}_{chd_j}^{s_1} - \mathbf{R}_{root} \times \mathbf{rv}_{chd_j}^{s_2})^2 \tag{6.1}$$

Our hierarchical alignment strategy is designed to address the complexities involved in aligning deforming features across corresponding spheres and their parent spheres. This comprehensive approach is crucial for attaining precise alignment of deforming features, which, in turn, plays a vital role in minimizing mesh distortions, including undesirable stretching and twisting. By carefully considering the alignment at both the child and parent levels, we ensure that the integrity of the deforming features is maintained throughout the process, preserving the original animations and movements of the characters.

The task of aligning corresponding non-root spheres presents additional challenges due to the presence of parent boundaries. These boundaries are directly correlated with the child boundaries in the parent sphere, making the alignment process more intricate. Failure to account for this correlation in the calculation of the rotation matrix can lead to over alterations of the sphere, potentially resulting in torsion and misalignment between the segments in three-dimensional space. Such distortions can significantly impact the visual and functional quality of the mesh, underscoring the importance of a nuanced approach to sphere alignment.

Moreover, aligning non-root spheres is complicated by the fact that corresponding spheres may have a varying number of parent boundary vertices. This discrepancy introduces a significant challenge in establishing direct point-to-point correspondences between spheres, further complicating

the alignment process. Our method eliminates the need for corresponding boundaries to have an identical number of vertices. Instead, we interpolate the boundaries of both spheres and compute the target positions to create target points for all vertices. Then, we could build the point-to-point correspondences between the original boundaries vertices and interpolated target vertices.

To align pairs of non-root spheres across two hierarchies, our method needs rotating each non-root sphere in the second hierarchy so that its parent boundary circle aligns with the plane of the corresponding sphere's parent boundary in the first hierarchy. This strategic alignment of parent boundaries considerably streamlines the process of establishing point-to-point correspondences among the boundary vertices, facilitating a more precise and coherent mesh alignment.

This alignment is achieved through the application of joint matrices. Referencing the definition provided in Equation 4.1, a joint matrix $\mathbf{M}_j$ encapsulates the rotation and translation transformations for a joint, which can be expressed as $\mathbf{M}j = [\mathbf{R}j, \mathbf{t}j]$. Here, $\mathbf{R}j$ represents the 3×3 rotational matrix, and $\mathbf{t}j$ signifies the xyz translation values corresponding to the jth joint. The rotation matrix necessary for aligning the parent boundaries onto the same plane, denoted $\mathbf{R}j, pat$, is calculated using the formula $\mathbf{R}j, pat = \mathbf{R}^{s_1}j, d \times (\mathbf{R}^{s_2}j, d)^{-1}$. In this equation, $\mathbf{R}j, d$ refers to the jth joint's rotation matrix derived from the dress pose.

As a result of this process, all parent boundary vertices, $\mathbf{BV}^{s_1}pat$ and $\mathbf{BV}^{s_2}pat$, are accurately positioned onto the circle of $\mathbf{BV}^{s_1}_{pat}$. Furthermore, the plane on which these vertices are placed is perpendicular to the direction of the bone to which they have been rotated.

On the circle of the parent boundaries, the positions of $\mathbf{BV}^{s_2}_{pat}$ on the circle of $\mathbf{BV}^{s_1}_{pat}$ can be estimated by $\overline{\mathbf{BV}}^{s_2}_{pat} = \mathbf{R}_{j,\perp} \times \mathbf{R}_{j,pat} \times \mathbf{BV}^{s_2}_{pat}$, where $\mathbf{R}_{j,\perp}$ is the rotation matrix around the bone-axis that we want to find. The calculation of $\mathbf{R}_{j,\perp}$ should minimize a correlation-based cost function that takes into account arc distance measurements on both the current spheres and their parent spheres, which is an important step to reduce the mesh distortion when establishing deforming feature correspondences between the characters that have significant morphological variations.

Figures 6.1 and 6.2 showcase the implementation of arc distances ($arc()$) within our cost function for aligning non-root spheres. This incorporation is pivotal for accurately calculating the spatial and deforming feature relationship and alignment between corresponding spheres.

To lay the groundwork for our cost calculation equations, we begin by representing the boundary vertices in $\mathbf{BV}^{s_1}pat$ and $\overline{\mathbf{BV}}^{s_2}pat$ using polar coordinates. These coordinates are denoted as $r, \theta_1, \theta_2, \cdots, {\theta_p}^{s_1}$ and $r, \theta_1, \theta_2, \cdots, {\theta_q}^{s_2}$, respectively, where θ values are organized in ascending order, and r signifies the radius of the circle. The use of polar coordinates for representing points on

a sphere is particularly advantageous due to the natural fit of polar coordinates with the spherical geometry. Unlike Cartesian coordinates, which are defined on a flat plane, polar coordinates express a point's location in terms of its angle and distance from a central point, aligning seamlessly with the spherical surface's curved nature. This system simplifies the mathematical description and manipulation of points on a sphere, facilitating operations such as rotation. This arrangement facilitates a straightforward method for establishing point correspondences based on angular positions, thus simplifying the alignment process.

Subsequently, we establish point correspondence for a given angle $\theta_a^{s_1}$ by employing a ratio λ, which is calculated relative to its two neighboring vertices $\theta^{s_2}b$ and $\theta^{s_2}b+1$ in the other sphere, as detailed in Equation 6.2. This method of using polar coordinates streamlines the alignment process, focusing it solely on the comparative edge lengths between the two corresponding spheres without complicating factors. The utilization of polar coordinates and arc distances in our alignment strategy offers an efficient way to achieve precise alignment. By focusing on angular relationships and radial distances, we ensure that the alignment accurately reflects the geometric properties of the spheres, thereby minimizing potential distortions and enhancing the overall quality of the mesh deformation.

$$\begin{aligned}\lambda_a^{s_1} &= \frac{\theta_a^{s_1}-\theta_b^{s_2}}{\theta_{b+1}^{s_2}-\theta_b^{s_2}}, \text{where } \theta_b^{s_2} \le \theta_a^{s_1} < \theta_{b+1}^{s_2} \\ \lambda_b^{s_2} &= \frac{\theta_b^{s_2}-\theta_a^{s_1}}{\theta_{a+1}^{s_1}-\theta_a^{s_1}}, \text{where } \theta_a^{s_1} \le \theta_b^{s_2} < \theta_{a+1}^{s_1}\end{aligned} \tag{6.2}$$

Then, given any rotation degree of θ, Equation 6.3 finds the *target position* $\mathbf{p}_k^s$ in the child boundary of the corresponded parent sphere that a boundary vertex intends to align to. $\mu(\theta_k, R_{j,\perp})$ returns the θ_k's corresponding vertex in the corresponding $\mathbf{BV}_{chd}^s$ in the parent sphere.

$$\begin{aligned}\mathbf{p}_a^{s_1} &= (1-\lambda_a^{s_1}) \times \mu(\theta_b^{s_2}, R_{j,\perp}) + \lambda_a^{s_1} \times \mu(\theta_{b+1}^{s_2}, R_{j,\perp}) \\ \mathbf{p}_b^{s_2} &= (1-\lambda_b^{s_2}) \times \mu(\theta_a^{s_1}, R_{j,\perp}) + \lambda_b^{s_2} \times \mu(\theta_{a+1}^{s_1}, R_{j,\perp})\end{aligned} \tag{6.3}$$

After that, the rotation matrix $\mathbf{R}_{j,\perp}$ is obtained to find the best θ value that minimizing the arc distance costs between the boundary vertices and the target positions (Equations 6.4 and 6.5).

In the context of feature alignment, it's crucial to mitigate large distance costs between corresponding vertices to prevent undue mesh distortion. Our algorithm addresses this by minimizing costs not only based on the alignment of individual corresponding spheres but also within the overarching

hierarchical structure of the character. Equation 6.4 introduces the first two cost functions, which assess arc distances on the parent spheres, while the third cost function evaluates the arc distances for the representative vertices (**rv**s) on the current spheres.

Equation 6.5 integrates these cost functions through a non-linear combination, applying coefficients α and β to balance the alignment cost relative to the parent space (gauged by arc distances on the parent spheres) against the cost pertinent to the child space (determined by the **rv** distances on the current spheres). This blend allows for the determination of the optimal rotation matrix $R_{j,\perp}$, achieving a balanced alignment that minimizes overall distortion by considering both the hierarchical context and the specific characteristics of each pair of corresponding spheres.

$$\begin{aligned} cost(\mathbf{R}_{j,\perp}, a) &= \tfrac{1}{2}|arc(\mu(\theta_a^{s_1}, \mathbf{R}_{j,\perp}) - \mathbf{p}_a^{s_1})| \\ cost(\mathbf{R}_{j,\perp}, b) &= \tfrac{1}{2}|arc(\mu(\theta_b^{s_2}, \mathbf{R}_{j,\perp}) - \mathbf{p}_b^{s_2})| \\ cost(\mathbf{R}_{j,\perp}, chd_k) &= |arc(\mathbf{rv}_{chd_k}^{s_1} - \mathbf{R}_{j,\perp} \times \mathbf{rv}_{chd_k}^{s_2})| \end{aligned} \tag{6.4}$$

$$\begin{aligned} \underset{\mathbf{R}_{j,\perp}}{\arg\min} =& (\frac{1}{p}\sum_{a=1}^{p} cost(\mathbf{R}_{j,\perp}, a)^2 + \frac{1}{q}\sum_{b=1}^{q} cost(\mathbf{R}_{j,\perp}, b)^2)^{\alpha} \times \\ & (\frac{1}{m}\sum_{k=1}^{m} cost(\mathbf{R}_{j,\perp}, chd_k)^2)^{\beta} \end{aligned} \tag{6.5}$$

Algorithm 6 shows the parent-to-child steps to perform the entire rotational alignment by executing $Align(\mathbf{Sph}_0^{s_1}, \mathbf{Sph}_0^{s_2})$, where $\mathbf{Sph}_0^s$ is the sphere associating to the root joint.

6.2.2 Exact Alignment with Vertex Displacement along Sphere Surface Directions

After the corresponding spheres and corresponding boundaries are aligned in parent-to-child rotational alignment, the exact alignment moves the boundary vertices in $\mathbf{BV}_{pat}^s$ and $\{\mathbf{BV}_{chd_1}^s, \cdots, \mathbf{BV}_{chd_m}^s\}$ onto their target positions $\{\mathbf{p}_k^s\}$. The target positions for exact alignment are calculated with Equation 6.3. In our method, each vertex on the sphere moves based on the interpolation from a radial basis function (RBF). Particularly, the new position of a vertex is computed from $\bar{\mathbf{v}}_i^s = \mathbf{v}_i^s + f(\mathbf{v}_i^s)$, $\mathbf{v}_i^s \in \mathbf{V}^s$. $f(\mathbf{v}_i^s)$ is a displacement function defining the movement of the vertex.

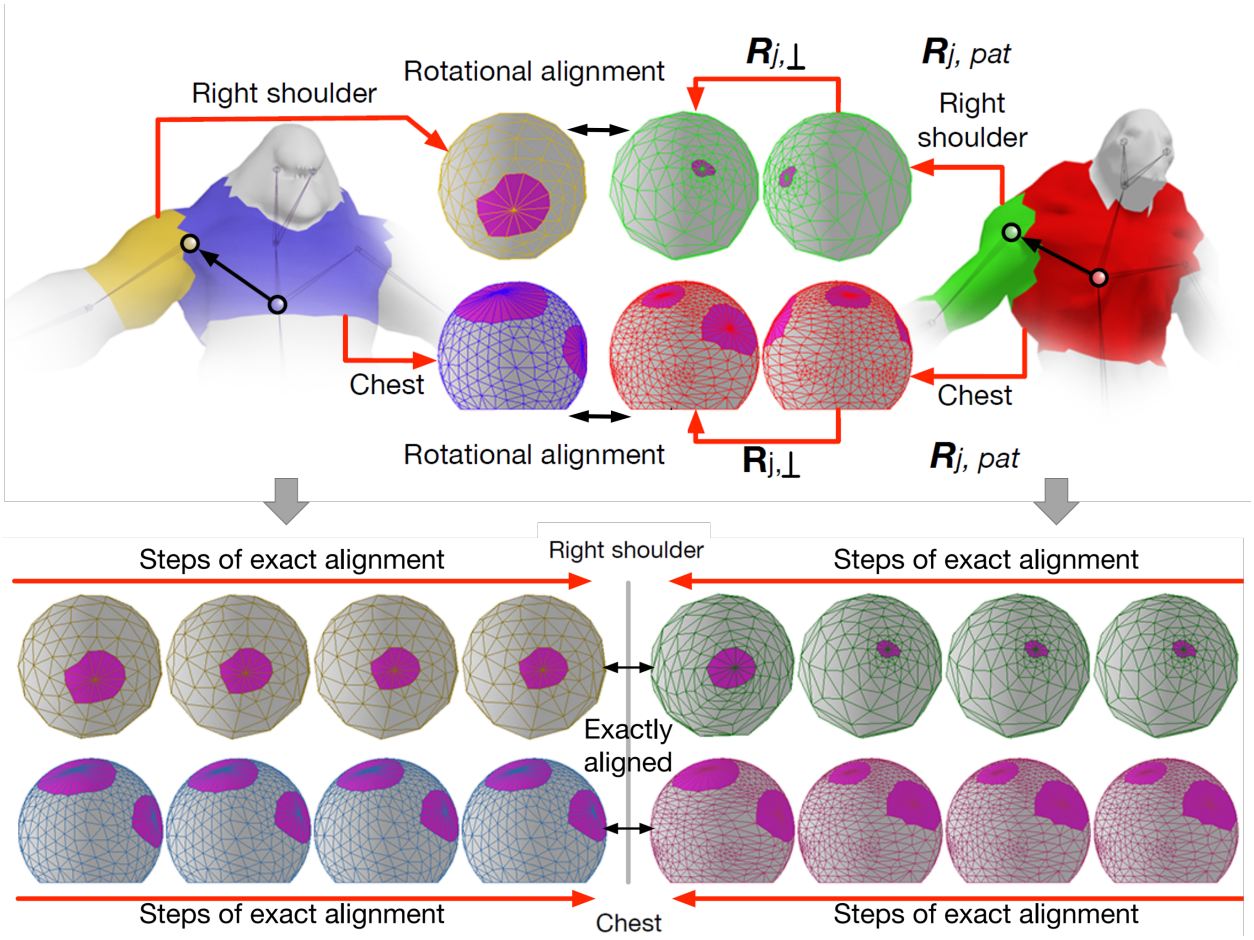


Figure 6.3: Intermediate results of aligning the chest segment (parent) and right shoulder segment (child) between the Troll and Alien models. Highlighted in purple are the child boundaries, delineating the area of focus for alignment. The upper section of the figure illustrates the rotational alignment phase, where corresponding spheres undergo rotation via a matrix that achieves the lowest alignment cost. The lower portion of the figure details the precise alignment stage, where the corresponding boundaries are aligned through mesh deformation, employing Radial Basis Function (RBF) interpolations.

Algorithm 6 Parent-to-Child Rigid Alignment

procedure ALIGN($\mathbf{Sph}_j^{s_1}$, $\mathbf{Sph}_j^{s_2}$)
  if the jth joint is the root **then**
    $\mathbf{R}_{root} \leftarrow$ Equation 6.1;
    Rotate $\mathbf{Sph}_j^{s_2}$ by $\mathbf{R}_{root}$;
  else
    $\mathbf{R}_{j,pat} \leftarrow \mathbf{R}_{j,d}^{s_1} \times (\mathbf{R}_{j,d}^{s_2})^{-1}$;
    $\mathbf{R}_{j,\perp} \leftarrow$ Equations 6.2, 6.3, 6.4, 6.5;
    Rotate $\mathbf{Sph}_j^{s_2}$ by $\mathbf{R}_{j,\perp} \times \mathbf{R}_{j,pat}$;
  end if
  if the jth joint is a leaf **then**
    return;
  else
    for the kth child of $\mathbf{Sph}_j$ **do**
      $Align(\mathbf{Sph}_j^{s_1}.\mathbf{Sph}_{chd_k}^{s_1}, \mathbf{Sph}_j^{s_2}.\mathbf{Sph}_{chd_k}^{s_2})$;
    end for
  end if
end procedure

The intermediate steps of exact alignment are illustrated in Figure 6.3. For a boundary vertex $\mathbf{bv}_i^s \in$ all $\mathbf{BV}^s$s, its displacement $f(\mathbf{bv}_i^s) = (\mathbf{p}_i^s - \mathbf{bv}_i^s)/2$. For non-boundary vertices, their displacement function is expressed in Equation 6.6. m and n are the total number of boundary vertices in the respective spheres.

$$
\begin{aligned}
f(\mathbf{v}_i^{s_1}) &= \sum_{k=1}^{m} \varphi(|arc(\mathbf{v}_i^{s_1} - \mathbf{bv}_k^{s_1})|)\gamma_k^{s_1} + \mathbf{P}^{s_1}(\mathbf{v}_i^{s_1}) \\
f(\mathbf{v}_i^{s_2}) &= \sum_{k=1}^{n} \varphi(|arc(\mathbf{v}_i^{s_2} - \mathbf{bv}_k^{s_2})|)\gamma_k^{s_2} + \mathbf{P}^{s_2}(\mathbf{v}_i^{s_2})
\end{aligned}
\tag{6.6}
$$

Radial Basis Function (RBF) interpolation is a mathematical approach well-known for its precision and flexibility in fitting and interpolating multidimensional data. RBF interpolation excels in handling complex deformations and spatial transformations, making it particularly well-suited for the intricate process of mesh alignment in 3D modeling and animation.

The core of RBF interpolation lies in its ability to construct a smooth and continuous function that precisely passes through a given set of points. This characteristic is invaluable for mesh alignment, where the goal is to map the vertices of one mesh onto another while preserving geometric and topological consistency. By utilizing RBF, we can accurately align mesh vertices, even in the presence of significant deformations or variations in mesh topology. RBF interpolation operates by defining a radial basis function centered on each point in the source mesh. The influence of each function diminishes with distance, ensuring a local rather than global effect on the mesh alignment. This local control allows for the detailed manipulation of mesh regions, facilitating a more nuanced and precise alignment process. RBF's adaptability to different kinds of deformations, whether linear or nonlinear, makes it a powerful tool for achieving high-quality mesh alignments.

In our work, the sets $\{\gamma_k^{s_1}\}$ and $\{\gamma_k^{s_2}\}$ are coefficient vectors, and $\mathbf{P}^{s_1}(\mathbf{v}_i^{s_1})$ and $\mathbf{P}^{s_2}(\mathbf{v}_i^{s_2})$ are linear polynomials. The details of the standard displacement function for RBF interpolation have been studied in the literature [28, 130]. In regards to the basis function $\varphi()$, we choose to use the CTPS C_b^2 version of compactly supported function [28]. This is because, comparing to other well-known versions of the basis function, the distortion caused by CTPS C_b^2 is rather small even if the boundaries encounter a large rotation.

6.3 Remeshing

Remeshing is a computational process used to modify the mesh structure of a 3D model, altering the distribution and connectivity of its vertices, edges, and faces while preserving the overall shape and features of the model [7,72,96]. This technique is pivotal in optimizing mesh quality for various applications, such as simulation, animation, and rendering, by improving the uniformity of the mesh, enhancing the resolution in areas of interest, or simplifying the geometry for computational efficiency.

In the process of achieving isomorphism between two surfaces following parameterization and feature alignment, remeshing is crucial. It is employed after parameterizing and aligning features between two non-isomorphic surfaces to modify both models' mesh structures, ensuring they attain identical topology and thus become isomorphic [111]. This process entails carefully adding, removing, or redistributing vertices and faces to guarantee that corresponding features on both models are matched by meshes with similar structure and density. Consequently, remeshing enables a smooth transition between models, supporting advanced techniques such as morphing, texture mapping, and uniform deformation across models with initially distinct geometric configurations.

Algorithm 7 Edge Intersection

for i in $edgesfrommodelA.size$ **do**
  for j in $edgesfrommodelB.size$ **do**
    if IS_MERGED($edgeA, edgeB$) **then**
      continue
    end if
    if CHECKINTERSECTION($v[A1].pos, v[A2].pos, v[B1].pos, v[B2].pos$) = true **then**
      $d11 \leftarrow$ DISTANCE($v.pos, v\_A1.pos$)
      $d12 \leftarrow$ DISTANCE($v.pos, v\_A2.pos$)
      $d21 \leftarrow$ DISTANCE($v.pos, v\_B1.pos$)
      $d22 \leftarrow$ DISTANCE($v.pos, v\_B2.pos$)
      $t1 \leftarrow d11/(d11 + d12)$
      $t2 \leftarrow d21/(d21 + d22)$
      $v.attributeA \leftarrow v[A1].attributeA \times (1.0 - t1) + t1 \times v[A2].attributeA$
      $v.attributeB \leftarrow v[B1].attributeB \times (1.0 - t2) + t2 \times v[B2].attributeB$
    end if
  end for
end for

Remeshing within a spherical domain is notably efficient and advantageous due to the inherent properties of spherical geometry, which offers a continuous, uniform surface without boundaries [111, 152]. This uniformity simplifies the process of distributing vertices evenly across the surface, ensuring a more consistent mesh quality and facilitating the preservation of geometric details. The spherical domain simplifies the remeshing process, making it easier to achieve isomorphic meshes between different models after feature alignment. Moreover, the spherical geometry inherently supports global parameterization techniques, which are crucial for maintaining the integrity of the mesh's topological relationships while accommodating the detailed features of the original models. Consequently, remeshing in a spherical domain is not only more straightforward but also produces high-quality results, making it a preferred approach for tasks requiring accurate and efficient mesh manipulation and alignment.

Remeshing in our hierarchical spherical domain introduces specific challenges, notably managing the boundaries between adjacent spheres. The primary challenge lies in ensuring seamless transitions across these boundaries, where the mesh densities, vertex distributions, and geometric details of adjacent spheres may significantly differ. In addressing the challenges of remeshing within a hierarchical spherical domain, particularly the issue of managing boundaries between adjacent spheres,

we have introduced an innovative edge intersection method. This technique focuses on the precise intersection of edges between two spheres, enabling the segmentation of these edges to facilitate a seamless transition across sphere boundaries. The core of this method lies in its ability to accurately detect and resolve the interactions between mesh edges at the boundaries, thereby allowing for the meticulous preservation of features even in these complex transition zones.

By employing edge intersection [18], the method generates additional vertices and edges at the boundaries, enhancing the mesh's ability to represent detailed features and maintain continuity across adjacent spheres. This increase in mesh complexity is a direct response to the need for higher fidelity in the representation of boundary features, ensuring that the remeshed model accurately reflects the original geometry and topology. The core of edge intersection is shown in Algorithm 7.

However, the introduction of more vertices and edges raises concerns about the resulting mesh density and computational efficiency. To mitigate these issues, we incorporate simplification techniques post-intersection. These techniques are designed to intelligently reduce the mesh complexity, removing unnecessary vertices and edges while preserving the critical geometric and topological features essential for the character's morphological integrity. The simplification process can be applied to maintain the balance between achieving a seamless boundary transition and retaining a manageable mesh size, ensuring that the remeshed model remains both high in quality and efficient for further processing and applications [87, 109, 138].

Chapter 7

Evaluations

Our approach was deployed on a workstation equipped with an Intel Xeon 3.7 GHz CPU and an Nvidia GTX 1080 GPU. To evaluate the effectiveness of our methodology, we selected a diverse set of deforming characters for testing. This collection included 5 characters that were articulated using the skeleton extraction algorithm as outlined by Homann [55], while the remaining 8 characters were acquired from CG Trader created by different artists.

The morphological diversity of these characters introduces significant variations in their hierarchical structures and deformation properties, which is a critical aspect of our experiment. The selected characters showcased a wide range of features, from differences in shapes (such as the presence or absence of tails) to variations in movement (bipedal and quadrupedal) and articulations involving a varying number of limbs (two, four, or eight). This variety was instrumental in rigorously testing the adaptability and effectiveness of our approach in handling complex deformations.

The implementation was carried out using C++ for its robust performance and flexibility, while OpenGL was utilized for rendering and visualization purposes. This combination allowed for efficient processing and high-quality visual output, facilitating a detailed examination of the morphing effects generated by our method. The results of our morphing experiments, showcasing the transformative capabilities of our approach across a range of character morphologies, are illustrated in Figure 7.1, highlighting the dynamic morphing effects achieved.

Table 7.1 presents the mesh and rig configurations for the characters used in this study, detailing the diversity of our test models.

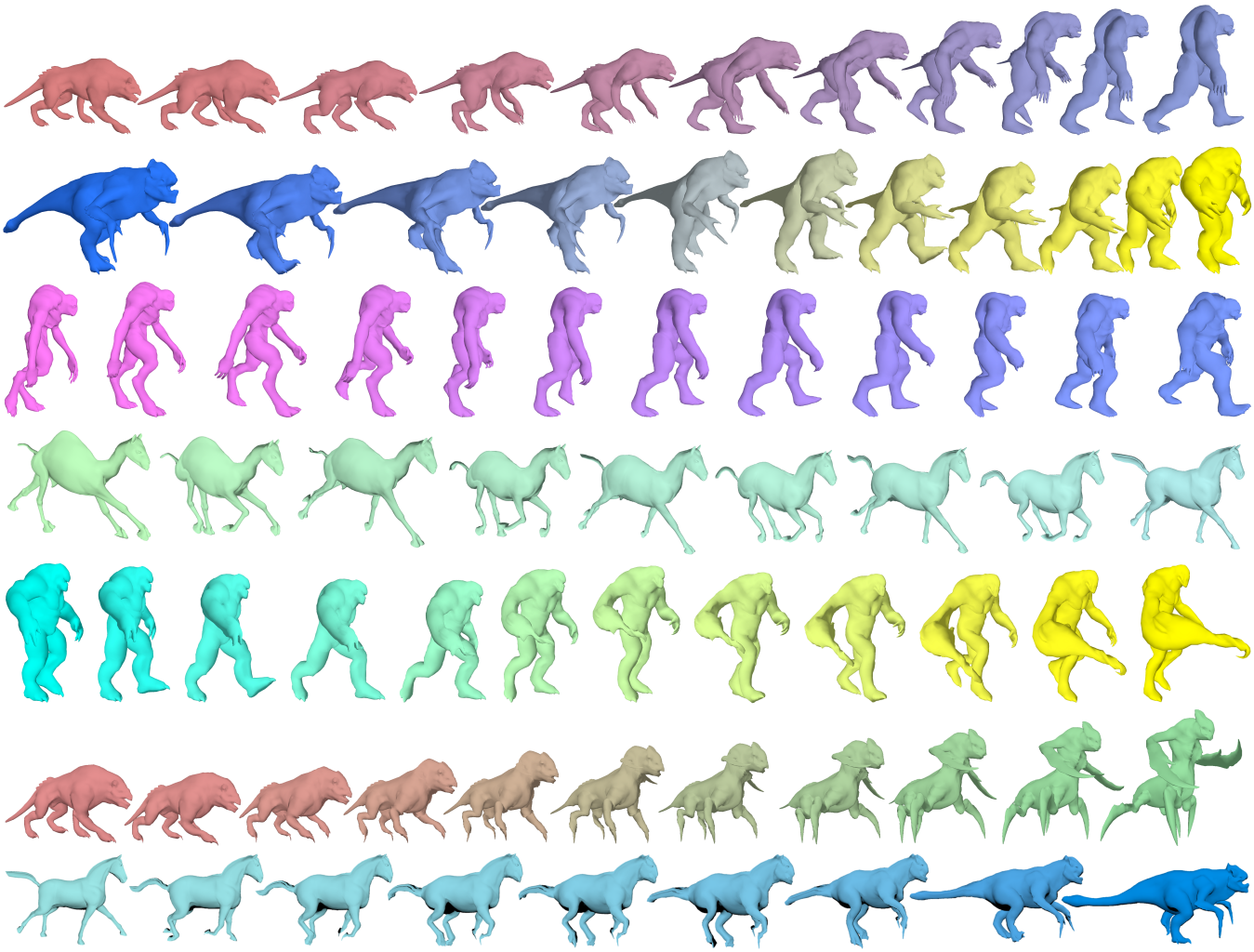


Figure 7.1: Morphing results with blended walking animations. From top to bottom, they are: Ripper Dog →Exterminator, Raptor→Troll, Alien→Exterminator, Camel→Horse, Troll→Mutant, Ripper Dog→Spider, and Horse→Raptor.

This work conducts a thorough evaluation of our methodology in distinct sections. Section 7.1 focuses on assessing the multi-sphere parameterization quality by measuring parametric distortion, offering objective metrics for its advantages. Subsequently, Section 7.2 examines the effectiveness of our hierarchical alignment method, comparing it with different alignment strategies and exploring the factors that impact its performance. Section 8 presents a variety of applications implemented using our methods, demonstrating the robustness and wide-ranging impact of our approach. This comprehensive analysis aims to illuminate the advantages of our approach and identify opportunities for further refinement in character animation and morphing techniques.

Table 7.1: Characters used in the experiment. * means the rigs are generated from the skeleton extraction method.

Characters	**Vertex Count**	**Triangle Count**	**Joint Count**
Horse*	8,317	16,630	38
Camel*	21,773	43,542	32
Sedan*	6,783	13,546	9
SUV*	6,108	12,196	9
Alien	2,846	5,688	43
Troll	2,481	4,958	40
Ripper Dog	2,489	4,974	27
Exterminator	3,175	6,346	50
Spider	2,936	5,868	23
Mutant	2,265	4,526	33
Creepy	3,233	6,266	34
Pangolin	2,762	5,520	32
Human	4,588	9,172	56

7.1 Parameterization Quality Evaluation

Our parameterization technique produces a series of spheres, upon which we conducted a detailed analysis to quantify the resulting parametric distortions. We did a comparative study to evaluate these distortions against those generated by traditional single-sphere parameterization methods. The specific types of parametric distortions we assessed encompass angular distortion, which relates to conformality, and area distortion, pertaining to authalicity.

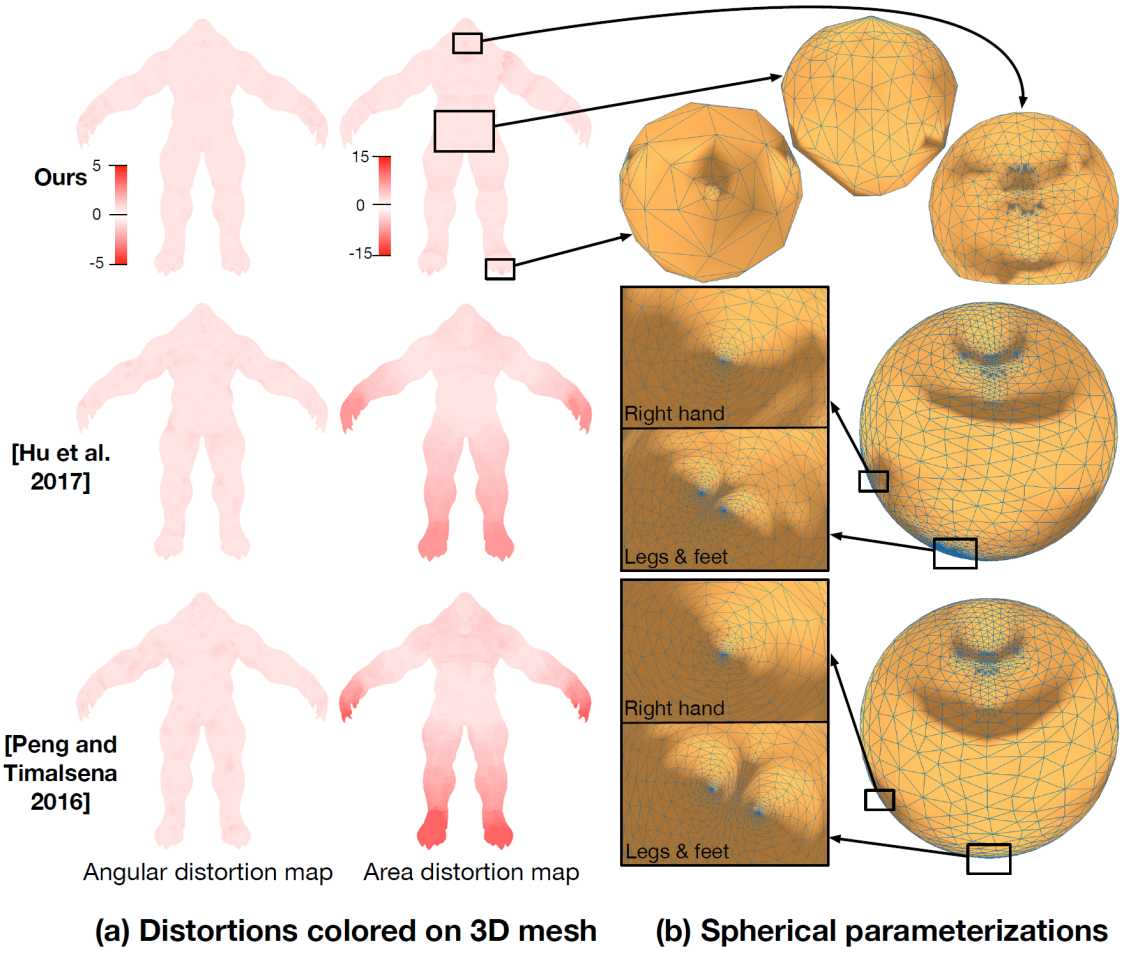


Figure 7.2: The parametric distortion comparison using the Troll character. In **(a)**, the parametric distortions are colored to the triangles of the 3D mesh. **(b)** shows three example spheres from our parameterization method and two enlarged parametric regions from the single-sphere parameterization methods.

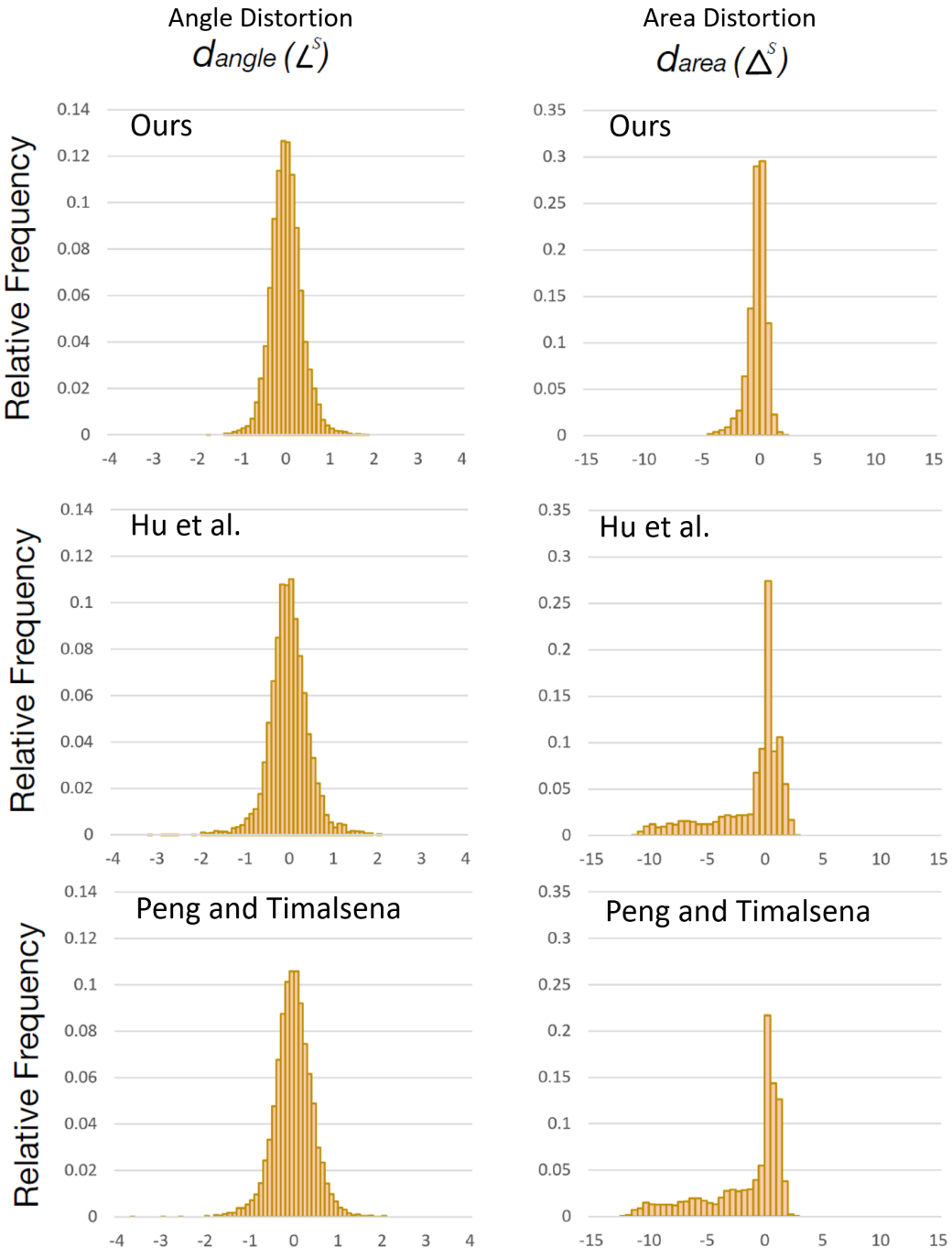


Figure 7.3: The parametric distortion histograms.

According to several studies in the literature [61, 100, 106, 111], it is established that minimizing angular and area distortions leads to parametric domains that more accurately mirror the geometry of the original mesh. These distortions, pertaining to conformality (angular distortion) and authalicity (area distortion), play a critical role in determining the fidelity of the parametric domains to the original mesh geometry. Angular distortion affects the angles within the mesh, potentially leading to skewed or misrepresented shapes when improperly managed. Area distortion impacts the relative sizes of different regions in the mesh, which can cause disproportionate scaling during the mapping process. The control of these distortions is therefore paramount for preserving the geometric and topological integrity of the original mesh in its parametric representation. Lower levels of these distortions ensure that the mapped representation more accurately reflects the details and overall structure of the source model. High fidelity in the parameterized domain ensures that features are aligned with minimal deviation, facilitating a more accurate and visually coherent transfer of details. This is particularly crucial in applications requiring precise feature correspondence, such as texture mapping, morphing, and deformation transfer.

Thus, achieving low parametric distortion is not merely a technical goal but a fundamental requirement for ensuring that the parameterized model retains the essence of the original mesh. This fidelity is instrumental in enhancing the quality of feature alignment, ultimately contributing to the creation of more accurate, realistic, and visually appealing digital representations.

In this evaluation, we implemented the single-sphere parameterization methods presented in previous work presented by Peng and Timalsena [106], adn Hu et al. [61]. Both of the methods decimate the mesh into a base shape and then insert the vertices back while computing bijective mappings onto the surface of the single sphere. In contrast, our parameterization method segments the mesh according to deforming properties. Mesh segments are decimated to the cone base shapes (except the root using a tetrahedron as the base shape). Then, each base shape is refined and mapped bijectively to a sphere, making the parent and child boundaries connect to the parent sphere and child spheres, respectively.

$$d_{\text{angle}}(\angle^s) = \ln\left(\frac{\text{Angle}(\angle^s)}{\text{Angle}(\angle)}\right) \tag{7.1}$$

$$d_{\text{area}}(\Delta^s) = \ln\left(\frac{\text{Area}(\Delta^s)/\sum_{t^s\in\text{sphere}}\text{Area}(t^s)}{\text{Area}(\Delta)/\sum_{t\in\text{mesh}}\text{Area}(t)}\right) \tag{7.2}$$

The angular and area distortions are calculated for each triangle as shown in Equation The angular distortion is given by Equation 7.1, and area distortion is given by 7.2. Note that the superscript s

means the spherical domain. The triangle being evaluated for the distortions is denoted as Δ, which has three values of $d_{angle}(\angle^s)$ and a singular value of $d_{area}(\Delta^s)$. t and t^s stand for every triangle of the 3D mesh and the spherical representation, respectively.

In assessing angular distortion, our parameterization method yields results comparable to those of single-sphere methods, as illustrated by the angular distortion map in Figure 7.2 (a). The angular distortion encountered is predominantly influenced by the cost function used during projection onto the spheres. Both our approach and single-sphere methods strive to preserve the equilaterality of triangles, adhering to recommendations found in [111, 142]. This preservation is crucial for maintaining a smooth and convex spherical surface. However, as noted in [48], this approach can lead to the compression of vertices in high-curvature areas into densely packed regions.

The innovative aspect of our method lies in its utilization of multiple spheres, which effectively mitigates the issue of dense region compression observed in single-sphere approaches. By employing multiple spheres derived from our method, we can alleviate the congestion in these dense areas, thereby better preserving the local shapes inherent to each deforming region, as depicted in Figure 7.2 (b). This advantage underscores the enhanced capability of our approach in maintaining the integrity and distinctiveness of local geometries within the parameterized model.

During the evaluation of area distortion, our methodology significantly reduces parametric distortion compared to traditional single-sphere methods, with notable improvements observed in the hands and feet, as highlighted in the area distortion map of Figure 7.2 (a). Single-sphere parameterization methods often struggle with pronounced area distortions in regions of the mesh characterized by high density and curvature. Such distortions pose challenges for subsequent alignment processes, particularly in accurately handling feature points within these complex areas.

Moreover, the condensed regions found in the single-sphere domain frequently lead to numerical challenges in the projection phase of parameterization. This is primarily due to the 1-ring kernel size being too small to effectively manage these areas, complicating the computation and potentially impacting the quality of the parameterization. Our approach, by contrast, alleviates these issues, offering a more robust and reliable solution for preserving the mesh's integrity during parameterization and facilitating a smoother alignment process.

We used the histograms to illustrate the distribution of the parametric distortions as shown in Figure 7.3. We did not find a significant difference regarding the relative frequency towards the lowest angular distortion. However, the relative frequency towards the lowest area distortion is significant: our parameterization method gathers 96.5% of the triangles to the area distortion range of $[-2, 2]$,

while the gathering is 81.26% with the method of Hu et al. [61] and 67.79% with the method of Peng and Timalsena [106]. Low area distortion in the parametric domain is important for feature alignment because it ensures that the relative sizes and proportions of features on the original surface are preserved when mapped to the parametric domain. This fidelity is vital for maintaining the integrity of the model's geometry and ensuring that detailed attributes are accurately represented. For deforming characters, when features are aligned with minimal area distortion, it results in more natural and visually coherent transitions between different model states, enhancing the realism and effectiveness of animations.

7.2 Alignment Quality Evaluation

An effective strategy for assessing the quality of feature alignment involves leveraging a 3D morphing application, offering a visually intuitive representation of the semantic matches' accuracy across blended shapes within the three-dimensional space. This method highlights the precision of feature correspondences and the preservation of semantic integrity across transitions. By observing how distinct features merge and transform from one shape to another, evaluators can discern the effectiveness of the alignment in maintaining consistent and coherent representations throughout the morphing sequence.

$$\varepsilon(t, \bar{t}) = 1 - \text{size}(t, \bar{t}) \times \sqrt{\text{shape}(t, \bar{t})} \tag{7.3}$$

To evaluate the quality of feature alignment in our morphed characters, we employed the size-shape metric, a well-regarded standard for evaluating the quality of triangular mesh deformation [28, 74]. This metric calculates the size-shape value for each triangle as shown in Equation 7.3, where t represents the triangle in its original state and $\bar{t}$ denotes the triangle after deformation, as defined in [74]. Notably, the size-shape metric is invariant to spatial transformations, making $\varepsilon(t, \bar{t})$ an effective measure of a triangle's contribution to the overall mesh distortion in a morphed shape. A lower average size-shape value across all triangles indicates a smoother and more natural transition between the input characters, reflecting a high-quality deformation and alignment process.

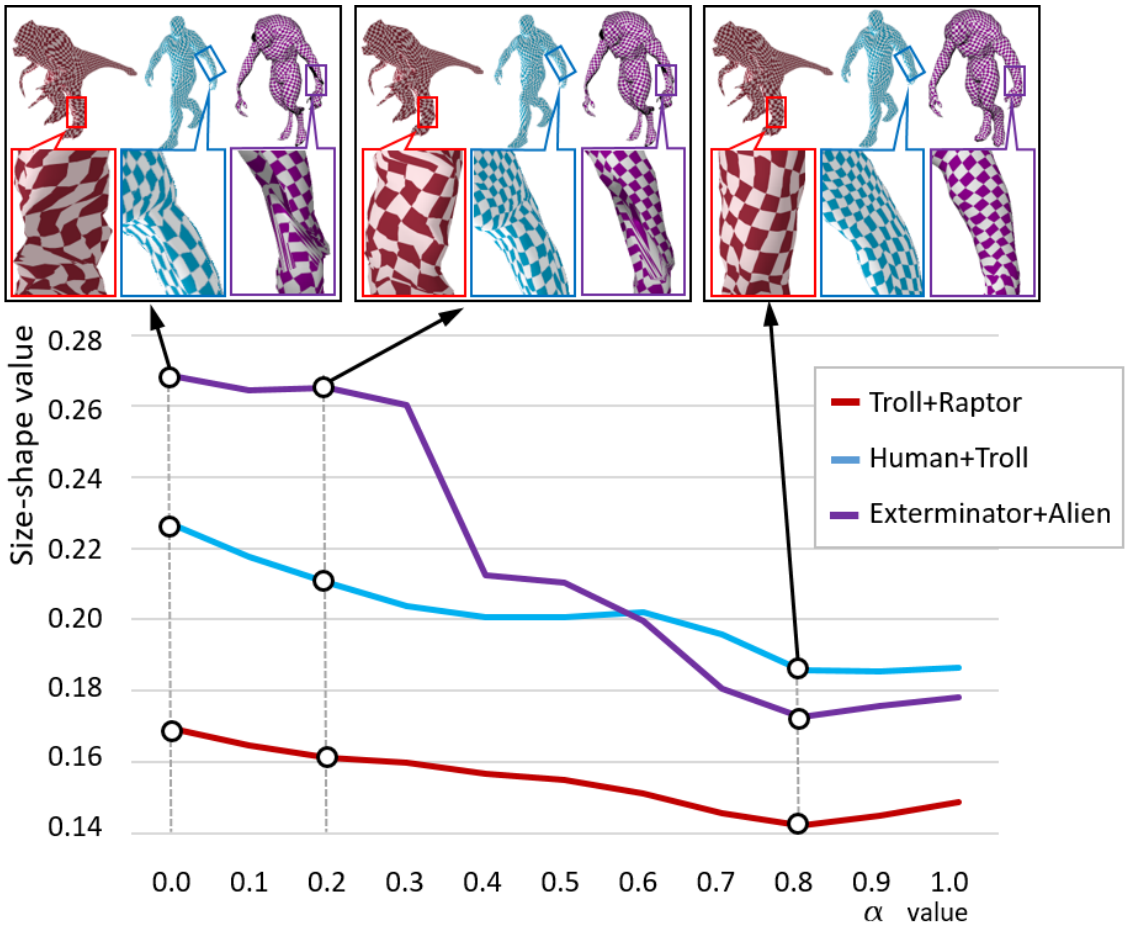


Figure 7.4: The average size-shape value over different α values from 0.0 to 1.0 with 0.1 increment (where β is $1 - \alpha$). The morphed shapes are from a pose in the walking animation with the linear blending ratio (0.5, 0.5). As indicated on the checkerboards, when $\alpha = 0.8$ (minimal size-shape value), the distortion on the mesh surface is the lowest.

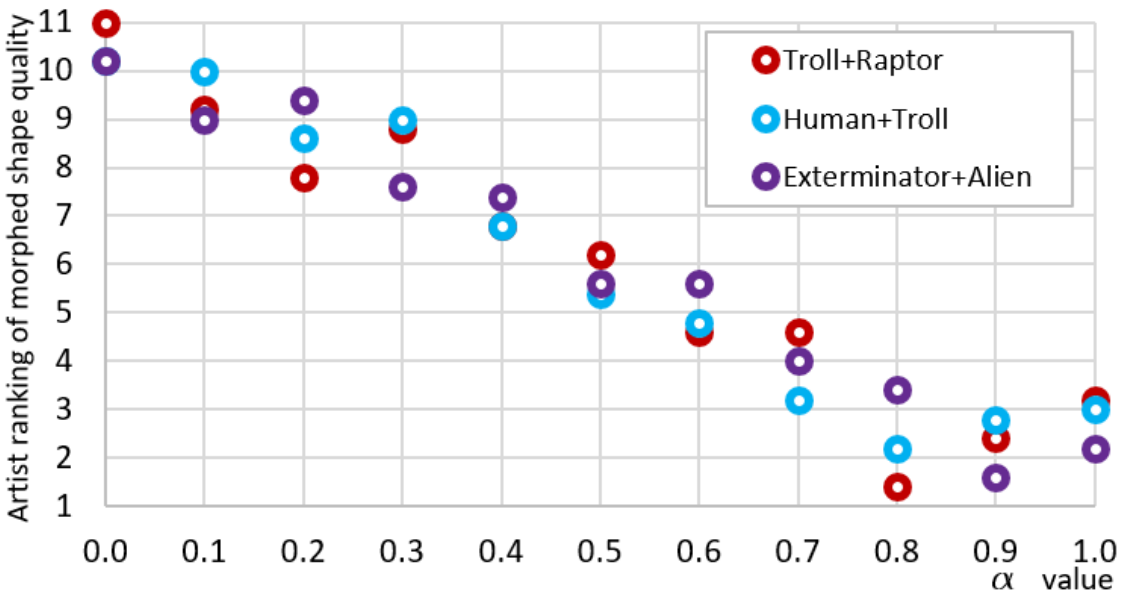


Figure 7.5: The artist ranking of the morphed shapes over different α values from 0.0 to 1.0 with 0.1 increment (where β is $1 - \alpha$). The linear blending ratio for morphing is (0.5, 0.5). Each artist ranks 11 morphed shapes in each morphing pair of the characters as shown in Figure 7.4. The plot data are the average ranks.

7.2.1 The Influence of the Global and Local Factors

In our study, we explored the impact of varying the parameters α and β within Equation 6.4, which are important in balancing the alignment cost related to the parent space (arc distances on the corresponding parent spheres) and the child space (**rv** distances on the current spheres). Our objective was to evaluate how different settings of α and β influence the morphed shape's quality. To this end, we applied the size-shape metric to evaluate all triangles within the spherical domain. The base status refers to the sphere post-rigid alignment (once α and β have been adjusted), while the updated status corresponds to the sphere following precise alignment, with the shapes primed for morphing.

Figure 7.4 presents the variation in the average size-shape value as α shifts from 0.0 to 1.0 in increments of 0.1, with β set to 1-α. The data reveal that exclusively prioritizing alignment costs in either the parent space ($\alpha = 1.0$, $\beta = 0.0$) or the child space ($\alpha = 0.0$, $\beta = 1.0$) fails to yield the lowest size-shape value. Intriguingly, the optimal balance, resulting in minimal mesh distortion during character morphing, occurs when $\alpha = 0.8$. This finding underscores the significance of

carefully calibrating the balance between parent and child space alignment costs to achieve the most natural and distortion-free morphed shapes.

Given the subjective nature of morphed shape quality, which varies according to individual perception, we complemented our statistical analysis with a qualitative study involving artists. We engaged five artists, aged between 26 to 32, with backgrounds in 3D character modeling, to assess the impact of control parameter variations on the perceived quality of morphed shapes. They evaluated three sets of morphed bind-pose characters at a linear blending ratio of (0.5, 0.5), spanning α values from 0.0 to 1.0. The artists ranked each model within the sets, with rankings from 1 to 11, where 1 signified the highest quality. The rankings from the artists of different combinations of the characters with different alpha value are listed in Table 7.2.

The artists' evaluations, as shown in Figure 7.5, indicated that the optimal morphed shape quality corresponded to an average α value of 0.86, closely aligning with the α value of 0.8 derived from the size-shape metric. This convergence of findings from both the size-shape metric and artists' perceptions underscores the reliability of these methods in assessing the quality of morphed shapes, offering valuable insights for refining morphing techniques.

7.2.2 Rotational Rigid Alignment Evaluation

Incorporating the bone-axis rotation ($\mathbf{R}j, \perp$) during the rigid alignment phase represents a novel algorithmic enhancement unique to our approach, distinguishing it from existing part-based methods [23,35]. To elucidate the advantages conferred by $\mathbf{R}j, \perp$, we conducted a thorough assessment of the sphere alignment quality. We contrasted the outcomes of our alignment technique, set at $\alpha = 0.8$ and $\beta = 0.2$, against those achieved without the implementation of bone-axis rotation.

As illustrated in Figure 7.6, although both methods successfully align deforming features on pairs of corresponding spheres to their precise target positions, the absence of $\mathbf{R}j, \perp$ (depicted in the second row of the figure) markedly diminishes the quality of the aligned spheres. This deficiency leads to significant triangle degeneration and folding, adversely affecting the mesh's integrity. Such parametric issues undermine the convexity of the mesh, resulting in excessive stretching on the morphed shapes. This comparison underscores the critical role of $\mathbf{R}j, \perp$ in maintaining the quality and structural integrity of aligned spheres, thereby enhancing the overall fidelity of the morphed character models.

Table 7.2: Artists' Rankings and Average for Different Character Combinations with Alpha from 0.0-1.0

Troll and Raptor						
Alpha	Ranking1	Ranking2	Ranking3	Ranking4	Ranking5	Average R.
0.0	11	11	11	11	11	11
0.1	10	10	7	10	9	9.2
0.2	9	8	9	9	4	7.8
0.3	7	9	10	8	10	8.8
0.4	8	5	8	6	7	6.8
0.5	5	7	6	7	6	6.2
0.6	6	6	3	3	5	4.6
0.7	4	4	5	2	3	4.6
0.8	1	2	1	1	2	1.4
0.9	2	1	4	4	1	2.4
1.0	3	3	2	5	3	3.2
Human and Troll						
Alpha	Ranking1	Ranking2	Ranking3	Ranking4	Ranking5	Average R.
0.0	11	10	11	8	11	10.2
0.1	10	11	9	10	10	10.0
0.2	9	9	7	9	9	8.6
0.3	8	8	10	11	8	9.0
0.4	7	6	8	7	6	6.8
0.5	6	7	1	6	7	5.4
0.6	4	5	6	4	5	4.8
0.7	5	4	3	1	3	3.2
0.8	3	2	2	2	2	2.2
0.9	1	3	4	5	1	2.8
1.0	2	1	5	3	4	3.0
Exterminator and Alien						
Alpha	Ranking1	Ranking2	Ranking3	Ranking4	Ranking5	Average R.
0.0	9	10	11	10	11	10.2
0.1	11	6	10	9	9	9.0
0.2	7	11	8	11	10	9.4
0.3	8	5	9	8	8	7.6
0.4	10	8	7	5	7	7.4
0.5	4	9	2	7	6	5.6
0.6	6	7	6	6	3	5.6
0.7	3	4	5	3	5	4.0
0.8	2	3	4	4	4	3.4
0.9	1	2	1	2	2	1.6
1.0	5	1	3	1	1	2.2

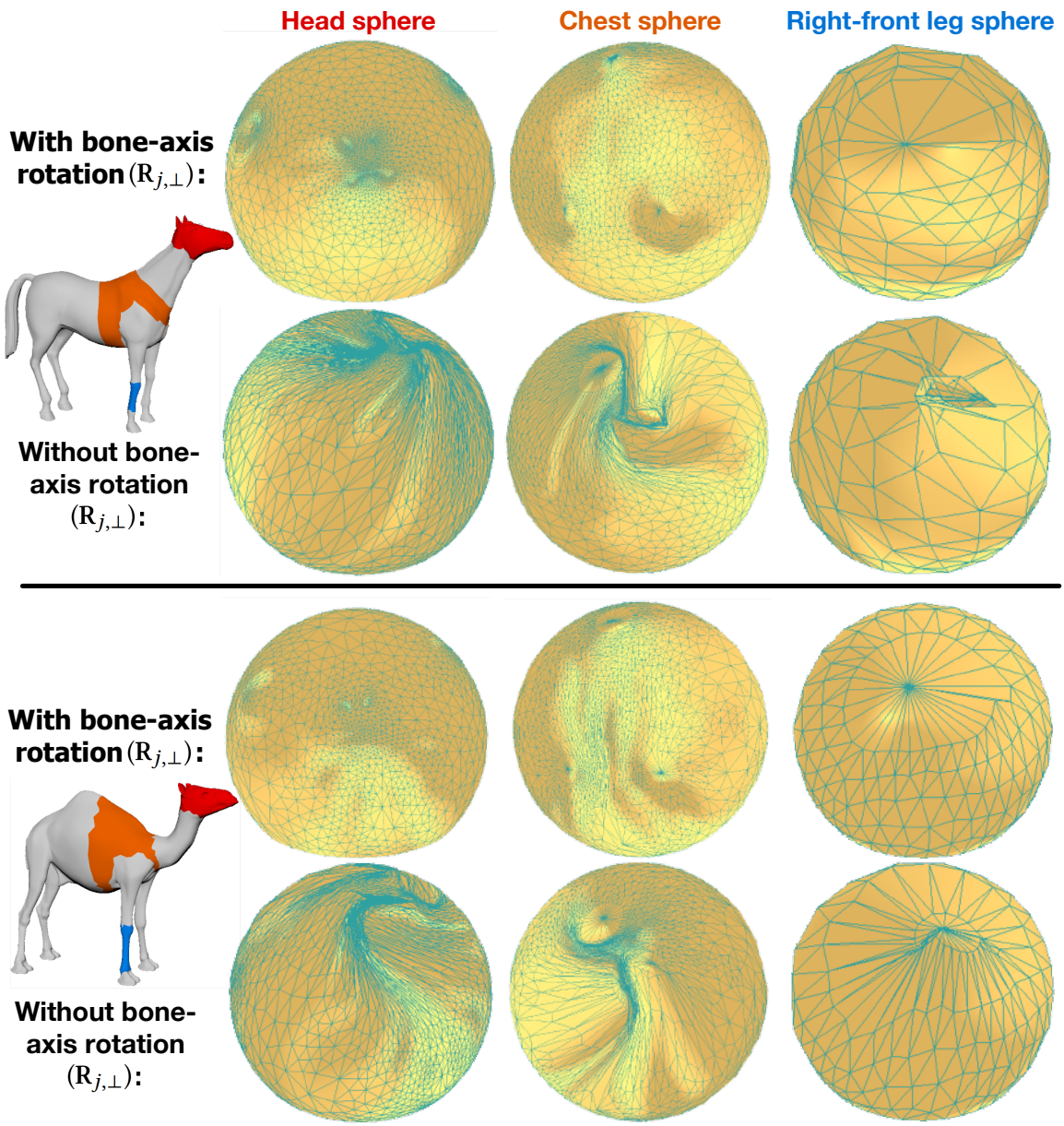


Figure 7.6: We compared the quality of aligned spheres using alignment methods with and without bone-axis rotation. To illustrate the differences in alignment quality, we utilized horse and camel models. Through the analysis of three example pairs of corresponding spheres, we found that the alignment method without bone-axis rotation resulted in feature points that were significantly further apart from each other. Consequently, during alignment, many triangles surrounding the feature points were pulled towards each other, leading to severe degeneration and folding issues.

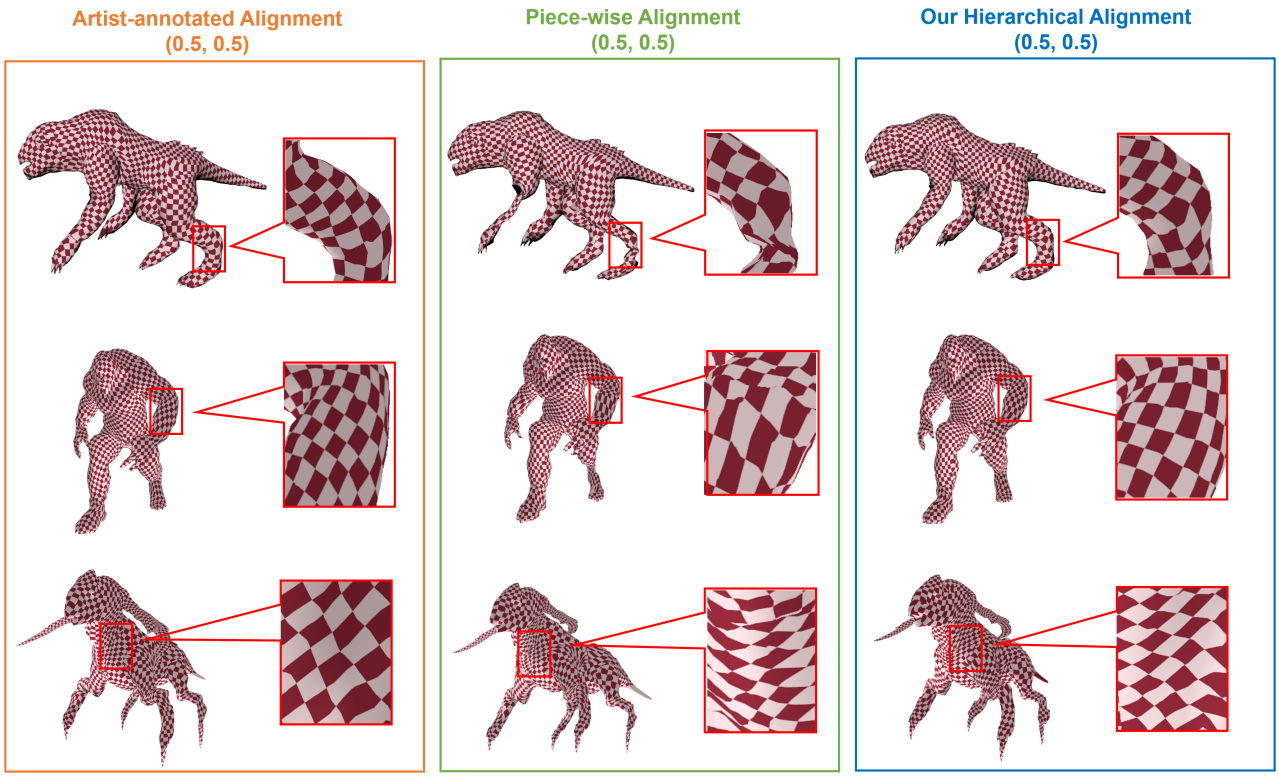


Figure 7.7: Comparison of artist-annotated alignment, piece-wise alignment, and our hierarchical alignment. The images shows the mesh distortion on the morphed shapes using the checkerboard. The piece-wise alignment cause a high degree of unwanted stretching and even tearing problems on the morphed shapes.

7.2.3 Comparison with Existing Methods

To evaluate the quality of alignment in our morphed shapes, we applied the size-shape metric, a measure well-suited for evaluating the deformation quality of morphed characters with their bind poses within 3D space. This evaluation technique blends the size-shape values according to the linear blending ratio $(u,\ 1-u)$ of the morphing process between the input characters, formulated as $u \times \varepsilon^{(1)}(t,\bar{t}) + (1-u) \times \varepsilon^{(2)}(t,\bar{t})$. Here, the base status is defined at $u = 1.0$, representing the exact shape of the first character, and at $u = 0.0$, denoting the exact shape of the second character, with the updated status corresponding to any u value within the range of 0.0 to 1.0.

For this particular experiment, we utilized the same pairs of characters and analyzed the blended size-shape values to measure the quality of alignment. Our evaluation involved a comparison between

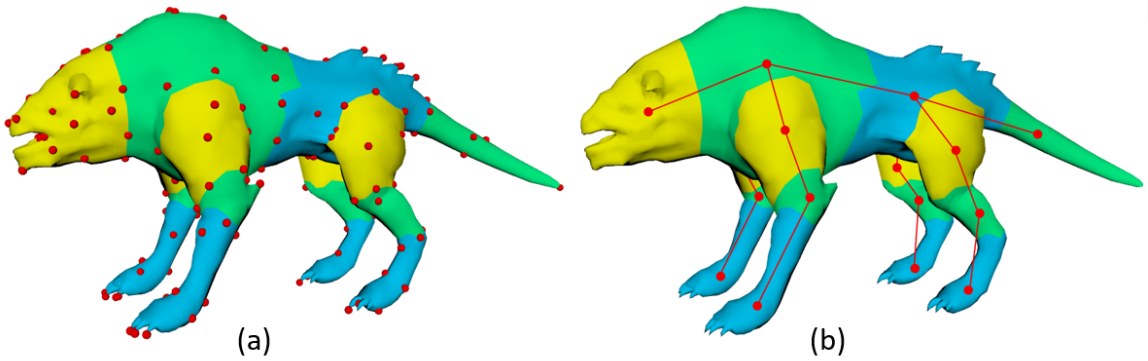


Figure 7.8: (a) shows 133 feature points were selected by artists carefully for the artists-annotated alignment. (b) shows the major joints selected in the piece-wise alignment method and our hierarchical alignment method. Artists only need to identify the matching of major joints.

our hierarchical alignment method and recent alternatives, including the artists-annotated method [10] and the piece-wise method [23]. Furthermore, we examined the manual effort required by these various alignment strategies, offering insights into their practical application. Below, we outline the methodologies implemented for the comparative methods in our study, providing a comprehensive overview of the techniques assessed alongside our own.

- **Artist-annotated alignment:** We used an artist-created morphing result to achieve a high level of alignment accuracy in the comparison. Following the methodology adopted in Avril et al. [10], a team of 3 artists collaborated to specify the feature correspondence points. The artists were tasked with carefully observing both the static and deforming features of the input characters. They were allowed to discuss and make adjustments to the feature correspondence points after observing the morphing results by following the expert elicitation guide, as presented by Hemming et al. [52]. The morphing effects of artist-annotated alignment were implemented based on the same multi-sphere domain as used in our approach.

- **Piece-wise alignment:** We followed the concept of piece-wise alignment as introduced in Chen et al. [23]. Their approach minimizes alignment costs by considering the local geometry features of the corresponding segments, without explicitly taking into account the hierarchical relationship between adjacent segments. In their approach, the parameterized segments belong to different domain types. To ensure a fair comparison, we implemented their approach within the same multi-sphere domain as our own. For the piece-wise alignment, manual effort to identify the matching of major joints is required.

To ascertain the quality of alignment in our morphed shapes, we applied the size-shape metric, evaluating the triangles positioned in the characters' bind poses within 3D space. The size-shape values were calculated using a linear blending ratio (u, $1-u$) between the input characters, formulated as $u \times \varepsilon^{(1)}(t, \bar{t}) + (1-u) \times \varepsilon^{(2)}(t, \bar{t})$. The experiment set the base status at $u = 1.0$ for the exact shape of the first character and at $u = 0.0$ for the second character, with the updated status at any u value ranging between 0.0 and 1.0.

Employing the same character pairs for this experiment, we leveraged the blended size-shape values to gauge the alignment quality. Our evaluation juxtaposed our hierarchical alignment method against recent techniques, including the artist-annotated method [10] and the piece-wise method [23]. Additionally, we examined the manual effort required for different alignment methods, providing insight into the implementation of compared methodologies.

Figure 7.7 presents three examples from this size-shape experiment, illustrating the comparison of alignment quality on morphed characters at a linear blending ratio of $(0.5, 0.5)$. The size-shape value starts and ends at 0, peaking near $u = 0.5$. Our hierarchical alignment method consistently outperforms the piece-wise alignment technique, achieving significantly lower size-shape values. The piece-wise method's shortcomings, particularly its neglect of parent-child segment correlations, resulted in 3D twisting artifacts, notably in characters with extensive morphological variations.

Moreover, our method drastically reduces the effort required from users compared to the artist-annotated alignment approach, as depicted in Figure 7.8. Rather than necessitating extensive feature point annotations by artists, our method simplifies the process to identifying major joint matches, a task easily managed even by those without expert knowledge. For instance, in morphing between the Ripper Dog and the Spider, artists needed 54 minutes and 46 seconds to annotate 133 feature points across four discussion rounds. In contrast, our alignment approach required only 52 seconds for artists to match major joints.

Notably, in the artist-annotated method, a significant number of feature points were located near segment boundaries, underscoring the importance of maintaining parent-child relationships in animation. This observation highlights the compatibility and effectiveness of our hierarchical alignment approach in addressing key areas critical to preserving animation integrity.

Chapter 8

Applications

Our method significantly enhances the precision and realism of outcomes in a wide range of applications, underscoring its versatility and revolutionary potential in digital workflows. It transforms 3D morphing by enabling seamless object transitions while preserving structural integrity. In texture transfer, this approach ensures precise texture mapping between models, maintaining fidelity across complex shapes. Character synthesis benefits from the creation of detailed and diverse models, bolstered by support for complex animations and interactions. Furthermore, deformation transfer is refined to accurately replicate model deformations, bridging differences in geometry. Collectively, these advancements highlight the method's capability to redefine standards in animation, game development, virtual reality, and more, introducing a new level of morphological variations.

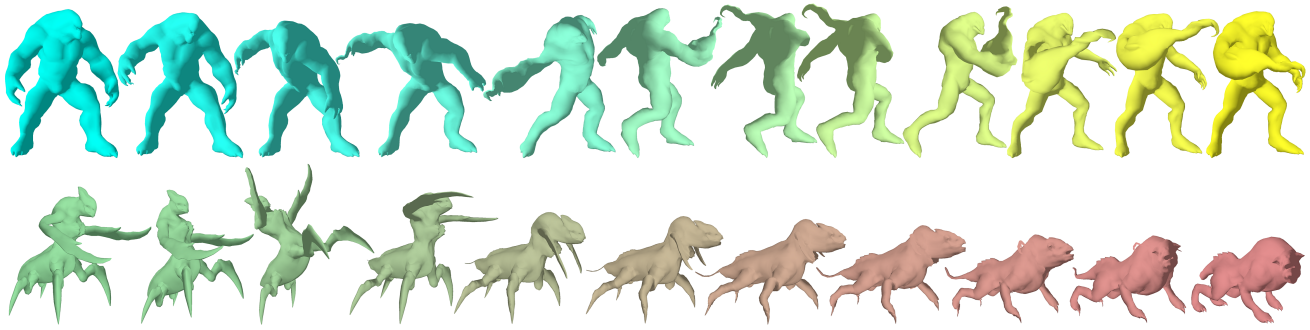


Figure 8.1: More morphing examples with different motion styles. The top image is Troll → Alien with the attacking animation, and the bottom image is Spider → Ripper Dog with the shouting animation.

8.1 3D Morphing

Utilizing our approach enables the implementation of morphing that captures a broad spectrum of motion styles and morphing techniques. This adaptability facilitates the transformation of characters, ensuring the transition is fluid and maintains the integrity of their distinctive movements and expressions. Our method extends the creative possibilities, allowing for the exploration of unique visual narratives and aesthetic expressions by blending different characters and motion styles effectively.

A morph between two models, M_1 and M_2, can be animated by linearly interpolating the positions of their vertices. The intermediate position v_i of a vertex during the animation is determined by the equation

$$v_i = t \cdot v_{i1} + (1 - t) \cdot v_{i2}, \tag{8.1}$$

where v_{i1} and v_{i2} are the positions of the vertex in M_1 and M_2, respectively, and t is the interpolation parameter ranging from 0 to 1.

By integrating 3D morphing, researchers and developers are provided with a powerful tool for iterative refinement. It creates a direct feedback loop, enabling the fine-tuning and optimization of algorithms to more adeptly navigate the complex challenges posed by intricate geometries and diverse morphological features. The incorporation of 3D morphing for quality assessment highlights the critical role of sophisticated feature alignment techniques in producing smooth and visually compelling morphs. Such morphs not only seamlessly transition between states but also faithfully convey the original design intent and artistic vision, demonstrating the pivotal contribution of our method to the advancement of digital animation and morphing technologies.

8.1.1 Different Motion Styles

Traditional morphing techniques have largely relied on blending mesh sequences by pairing specific poses, as highlighted in previous studies [23, 113, 147]. These approaches are adept at handling predefined motions but falter when applied to new mesh sequences that exhibit distinct motion styles. Each new sequence demands a fresh round of parameterization and alignment, introducing significant workflow inefficiencies and limiting the adaptability of the morphing process to new content.

Our methodology, as depicted in Figure 8.1, showcases an advanced capability to generate morphing

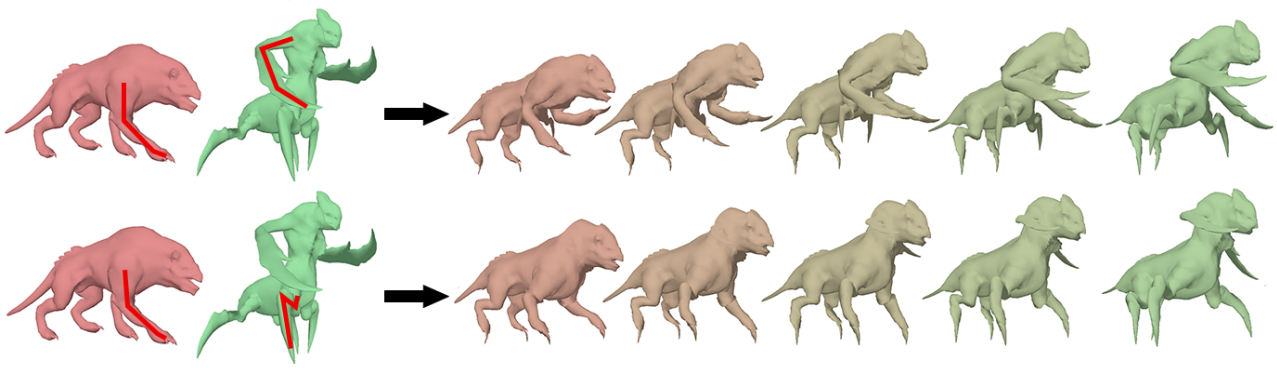


Figure 8.2: Two different morphing styles generated with different major joint pairs in Ripper Dog → Spider. In the first row, the Ripper Dog's front legs are aligned to the blade arms of the Spider. In the second row, the parameterizations of the blade arms are merged to the neck sphere, and in morphing they grow out from the neck segment.

effects across an array of motion styles utilizing a unified parameterization and alignment within a sphere-based framework. This adaptability originates from our innovative hierarchical parameterization and alignment strategy. This strategy is designed to intricately embed and safeguard the deforming features and structural relationships that are crucial to the character's articulation, ensuring that the essence of the motion is captured and preserved.

Contrastingly, the traditional focus on specific poses can restrict the scope of morphing effects, confining them to the initially selected sequences. Our method, however, establishes a durable and adaptable framework. It successfully navigates the challenges posed by different motion styles without necessitating the repetitive tasks of parameterization and alignment for each new sequence. This not only significantly simplifies the morphing process but also broadens the potential for creativity, allowing for the generation of more diverse and dynamic morphing effects.

8.1.2 Different Morphing Styles

Building on the method's flexibility, our framework allows for a wide range of shape transformations, encouraging creative experimentation by pairing key joint pairs, which makes our method be able to achieve different morphing styles. This feature is clearly shown in Figure 8.2, where the primary joints of the Spider model can be innovatively matched with those of the Ripper Dog model. This level of adaptability opens up possibilities for creative morphing scenarios, such as smoothly

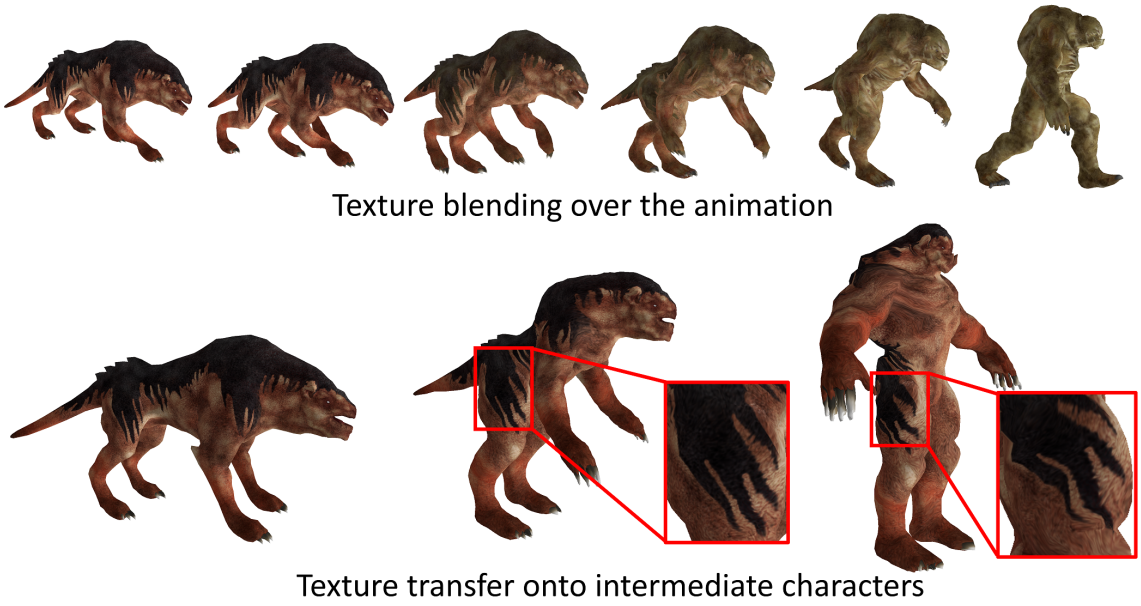


Figure 8.3: A texture transfer and blending example (Ripper Dog → Exterminator). The UV coordinates as part of the attributes of the vertices are parameterized and aligned in the same way as the vertices; therefore, the pieces of the UV texture is parameterized onto the corresponding spheres. As a result, the textures of the characters can be blended while they are morphed over the animation sequence. It also shows the texture of Ripper Dog can be transferred onto Exterminator or an intermediate character based on the matched structure from the alignment.

attaching the Spider's blade arms to the Ripper Dog's neck or combining the Spider's front legs with the Ripper Dog's torso.

This approach simplifies the adjustment of major joint pairings to facilitate unique morphological outcomes, significantly broadening the scope for creativity without necessitating a complete overhaul of the existing parameterization. Nonetheless, to preserve the cohesiveness and visual integrity of these novel configurations, a meticulous reevaluation of the alignment process is imperative. This ensures that, despite introducing unprecedented morphological variations, the morphing transition remains smooth, and the fundamental structural relationships between the character models are well preserved.

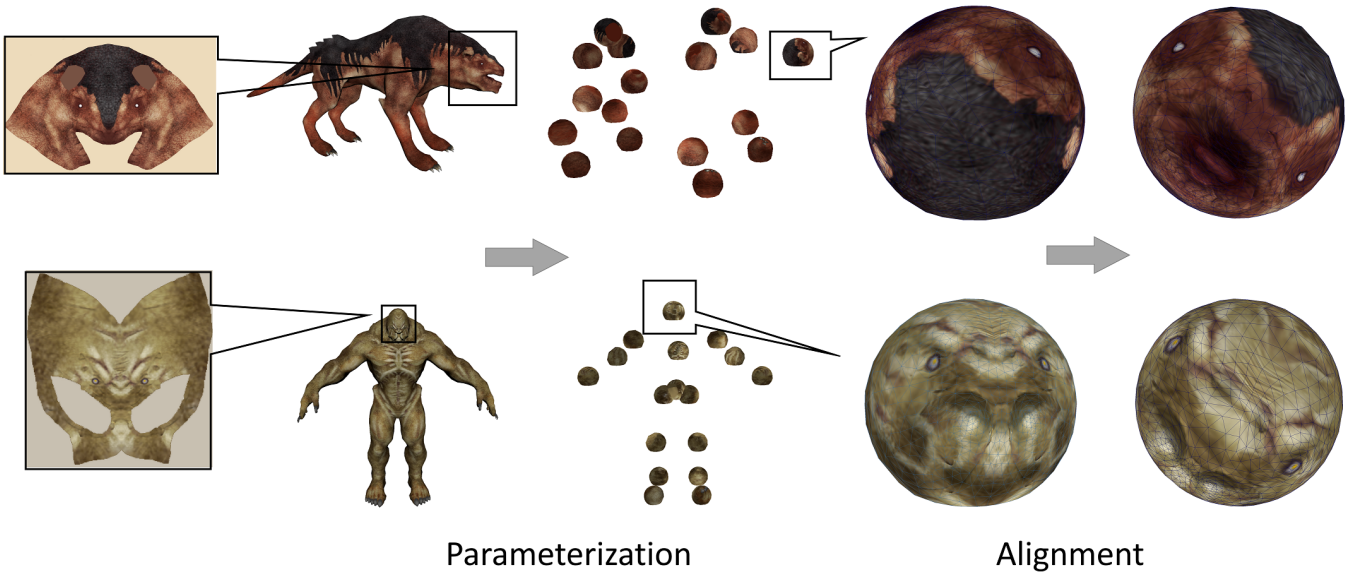


Figure 8.4: Our texture mapping process involves the application of diverse UV textures onto the surfaces of various input characters. Subsequently, these textures undergo parameterization within a spherical domain. This is followed by the feature alignment stage, wherein the textures are aligned, ensuring a seamless integration of visual elements across the character models.

8.2 Texture Transfer and Blending

Seamless texture transfer and texture blending effects can be achieved using our approach, like the example shown in Figure 8.3.

The process of texture transfer presents several challenges, primarily due to the varying UV sets among different characters and the difficulty in controlling the boundaries within UV textures during alignment [8,30]. In our approach, as shown in Figure 8.4, we begin by mapping distinct UV textures onto the surfaces of various input characters. This is followed by the parameterization phase with the texture features, where the textures are projected onto a spherical domain. Subsequent to this, we engage in the deforming feature alignment process to ensure the textures are precisely aligned. The culmination of this process occurs when we reproject the textures from the spherical domain back to the UV domain. This final step allows us to seamlessly integrate two disparate textures into a unified UV layout, overcoming the initial challenges and achieving coherent texture transfer.

The existing methods often establish the planar parameterization to embed the texture UVs of different models into the common parametric domain (e.g., [125]). However, the planar parame-

terization creates additional domain seams that may cut the texture UVs into more pieces. The open boundaries at the seams complicate the feature representations, make the feature alignment different, and ultimately may cause the the transferred textures to be stretched or intermittent on the mesh surface of the target character.

Spherical parameterization can provide the seamless texture transfer and blending, like the work presented by Dinh et al. [30] that represent the mesh in a single sphere. The single sphere represents the mesh at high parametric distortion and often require a significant amount of manual work to label corresponding features. These are the same limitations of the single sphere as we observed in the evaluation of this work. Our parameterization method creates a multi-sphere representation that ensures a low parametric distortion ratio, and the hierarchical alignment method is able to identify and align the structural features for the corresponding body segments, not only for maintaining the continuity of the textures but also conveying the semantic mapping of the textural features when being transferred and blended between the corresponding segments.

8.3 Character Synthesis

Morphological variations are crucial for enhancing visual diversity in digital environments, significantly contributing to the overall realism of scenes. For example, introducing a wide array of morphological variations among characters in a crowd scene can dramatically improve its authenticity, moving away from the monotony of duplicated figures. The challenge, however, lies in the extensive design and authoring efforts required to create a unique character for every instance in such scenes, a task that is both time-consuming and labor-intensive, as discussed in works by Hamed et al. [51] and Maim et al. [95].

Our innovative approach simplifies the creation of diverse characters through the strategic mixing and matching of specific body parts along with their associated rigging branches. This method is enhanced by our support for non-linear blending techniques, which allow for the precise alignment and integration of selected spherical segments, adding depth and variety to the character creation process. A key feature of our methodology is the use of skinning weights related to the joints of these selected spheres. These weights serve as gradient maps that facilitate smooth interpolation between segments, ensuring that transitions in shape and deformation between adjacent segments of a selected sphere are both seamless and natural.

Illustrated in Figure 8.5, our approach enables the synthesis of new, unique characters by combining

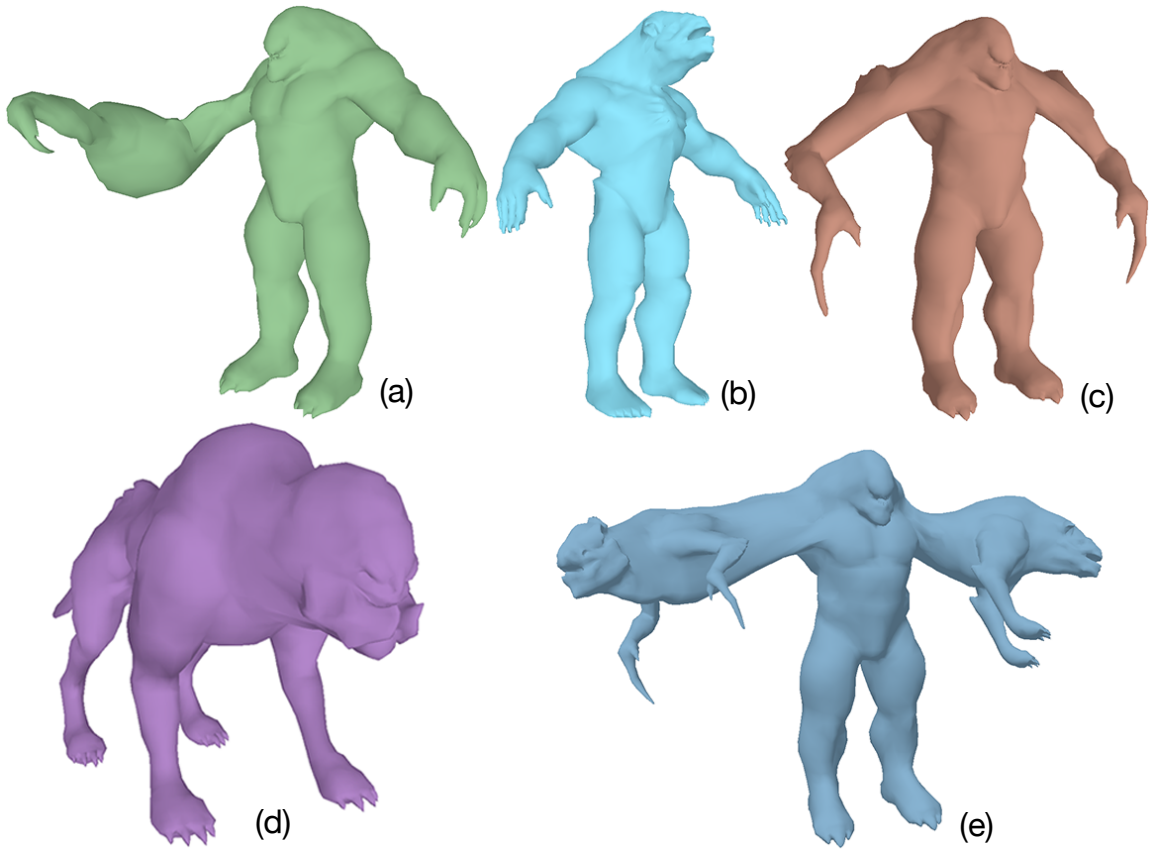


Figure 8.5: New characters synthesized by mixing body parts. (a) is the Troll with the Mutant's right hand. (b) is the Exterminator with the Ripper Dog's head. (c) is the Troll with the Raptor's hands. (d) is the Ripper Dog with the Exterminator's head. (e) is a character synthesized from three: Troll (body), Ripper Dog (left hand), and Raptor (right hand).

different body parts from a pool of input characters. This innovative technique not only alleviates the need for the exhaustive manual creation of each character but also opens up new possibilities for generating diverse and dynamically varied character populations. The ability to blend and interpolate between segments using gradient maps based on skinning weights introduces a level of sophistication and flexibility in character design, leading to richer, more compelling digital scenes. This method represents a significant advancement in digital animation, offering creators the tools to efficiently produce varied character ensembles with enhanced realism and visual appeal.

8.4 Deformation Transfer

In our study, we explored the application of our approach to deformation transfer, a technique that enables the transference of deformations and animations from a source character to a target character [12,41,114]. Our method enhances this process by facilitating the transfer of deformations and animations to both the target character and to intermediate characters interpolated between the source and target. This capability is vividly demonstrated in Figure 8.6, showcasing the seamless transition of deformations across characters.

The core of our approach's effectiveness lies in the precise alignment of major joints and deforming features between the source and target characters within the parametric domain. This alignment permits the accurate morphing and interpolation of bind-pose shapes and skinning weights, leading to the creation of intermediate characters that embody a linear blend of the source and target. Subsequently, the joint transformations observed in the animation are applied to the newly modeled characters by analyzing the displacements from the bind-pose, as discussed by Vlasic et al. [141]. This process ensures the preservation of the original animation's essence and dynamics across the derived characters.

By incorporating deformation transfer into our cross-parameterization framework, we enhance character animation's versatility. This integration not only simplifies the adaptation of animations across a wide range of characters but also enables the creation of a continuum of variably morphed characters. Each character inherits the dynamic expressions and movements of the original, enriching the animation sequence with a diversity of character designs while maintaining the original animation's integrity and fluidity.

Figure 8.6: A deformation transfer example that transfers Troll's walking to the Human. The first row is the walking sequence of the Troll. The last row is the Human with Troll's walking style. With our approach, the deformation transfer can be accompanied by a shape transfer. The middle row is the intermediate characters between the Troll and Human and with Troll's walking style.

Chapter 9

Discussions

We have introduced an innovative approach to cross-parameterization that establishes a bijective mapping between models, effectively reducing parametric distortions, including both angular (conformal) and area (authalic) distortions, while meticulously preserving the integrity of deforming features. Central to our methodology is the utilization of a novel hierarchical spherical domain. This approach proves to be more effective than traditional single spherical domain methods and non-hierarchical multi-spherical domains.

By leveraging a hierarchical structure, our method enhances the precision in the alignment of features and minimizes distortion, ensuring a more accurate representation of the original models and the fidelity of deformation features throughout the parameterization process. This advancement in cross-parameterization techniques marks a significant improvement in maintaining the geometric and topological integrity of models during complex transformations.

Traditional parameterization methods utilizing spherical domains often struggle with maintaining bijectivity, leading to significant distortion and potential foldovers on the sphere's surface [76, 143]. These issues are frequently exacerbated by the presence of high-curvature regions within the mesh, such as fingers and armpits. In contrast, our hierarchical domain approach presents an innovative solution that significantly enhances bijectivity. By structuring the parametric space across multiple levels, our method adeptly handles features distributed across different spheres.

This multi-level organization enables us to target a piecewise cost function rather than tackling a more complex global one. As a result, mapping is conducted on spheres corresponding to submeshes, which are inherently simpler to manage compared to a singular sphere mapping the entire mesh.

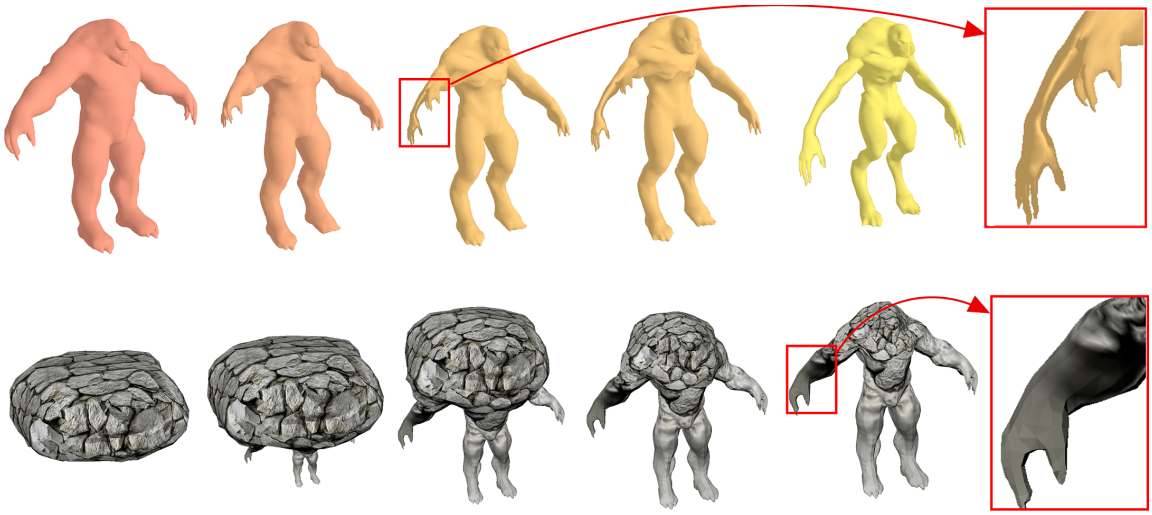


Figure 9.1: The first row demonstrates that poorly applied skinning weights can result in misalignment. For experimental purposes, we asked artists to apply inadequate skinning weights to the right arm and hand of the Alien, primarily influenced by the chest joint. As observed, when morphed with the well-skinned Troll, the arm and hand appear to grow from the shoulder. The second row demonstrates an extreme example of morphing between a rock model and the Troll. Given that rock contains only one joint, aligning features necessitates merging all spherically parameterized segments of the Troll to the root sphere. Consequently, this merges those spheres into a single spherical domain, causing significant distortions as evidenced on the arms and legs.

This strategic approach not only reduces the complexity of the mapping process but also significantly improves the accuracy and integrity of the parameterization, effectively addressing the challenges posed by high-curvature regions.

In the realm of cross-parameterization, the task of accurately mapping deforming characters presents significantly more complexity than dealing with static models. For static models, the success of cross-parameterization largely hinges on the precise alignment of geometric features such as surface curvatures and angles. These elements can be conformally projected into the parametric domain, ensuring the preservation of normal vectors for curvature and minimizing parametric distortions. However, translating the intrinsic attributes of deforming characters, which include skinning weights articulated with a skeleton or derived from a mesh sequence, into a coherent vector representation within the parametric domain poses a substantial challenge.

Our methodology addresses this challenge by adeptly extracting and utilizing the parent-child relationships inherent in the deforming characters' intrinsic information. We then integrate these relationships as the defining boundaries of part segments. These boundaries serve as near-rigid approximations, closely aligned with the hierarchical domain's structure, ensuring they are well-suited for accurately reflecting the characters' dynamic nature. This innovative approach allows for a more effective and precise alignment of deforming features, markedly enhancing the quality of cross-parameterization for characters in motion.

9.1 Suboptimal Skinning Weights

The cross-parameterization method we introduce proves highly effective in handling characters that exhibit substantial morphological differences, as demonstrated by our results. Such morphological variations significantly influence the variability in hierarchical structures and play a crucial role in shaping the skinning weights that are essential for mesh envelopment. Our methodology leverages a part-based segmentation technique adept at addressing challenges posed by suboptimally applied skinning weights, ensuring the generation of coherent mesh segments.

It's important to acknowledge, however, that suboptimal skinning weights can lead to a hierarchical representation that may lack in structural integrity. This is largely due to the intrinsic deformation challenges that inherently accompany poorly executed skinning weights. When skinning weights are referred to as suboptimal, it implies that these weights are not ideal or are less than perfect for the intended animation effect. This could lead to issues such as unnatural deformations, poor

articulation, or clipping errors in the animated model.

As shown in the first row of Figure 9.1, the hand and arm, which were skinned to the shoulder joint, do not establish adequate correspondences with the other well-skinned character during alignment, resulting in an unnatural morphing effect.

9.2 Morphing of Highly Varied Models

When the rigs of two models have very different number of levels of joints, it is possible of losing fine deforming features during the alignment. In the most extreme cases where, for example, one model is rigged with only one joint while the other model is rigged in a skeleton with tens of joints, the only joint of the first model likely have to be matched with the root joint of the second model, such as the example shown in the second row of Figure 9.1. In such a situation, all other spheres parameterized for the second model must be merged to the root sphere.

As a result, the hierarchical spheres degenerate into a single sphere, which makes the process regress to a version almost identical to the traditional spherical parameterization. Therefore, distortions observed in single-sphere cross-parameterization approaches [27, 106] would stay as problematic in this context. For example, this problem can be observed in the arm region depicted in the second row of Figure 9.1.

9.3 Morphing of Multiple Mesh Objects

Apparently, our approach is limited to shapes with a genus of zero, a constraint that is common to the general concept of sphere-based parameterization. A further drawback of a genus zero mesh is its inability to accommodate multiple mesh objects. Comparing with traditional single sphere parameterization methods, while our approach was not specifically developed for meshes with multiple objects, it can be adapted to this scenario by treating each object as a pre-segmented mesh. In fact, we have illustrated this adaptability in Figure 9.2, where the car with six separate objects (two body parts and four wheels) were parameterized into six spheres and represented in a hierarchy. However, creating feature correspondences between the sphere of an object and the boundaries of a corresponding sphere requires manual intervention from the user. Further investigation of automated techniques for processing meshes with multiple objects are needed to address this limitation.

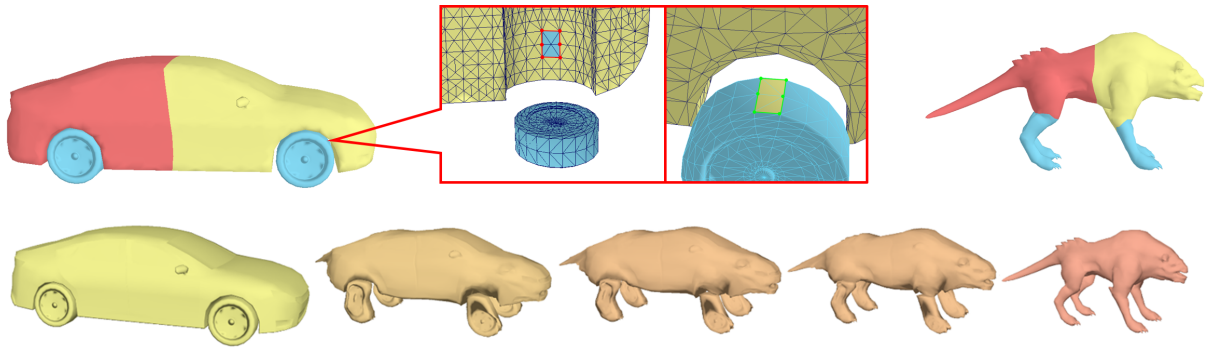


Figure 9.2: A car-to-dog morphing example demonstrating the adaptability of our approach to morph a multi-object model with a deforming character. The car consists of a body part and four wheels. We manually organized the body part and wheels into a hierarchical representation. The root sphere represents the parameterization of the body part, and four child spheres represent the wheels. Since the spheres have no boundary due to their rigid binding, we have implemented a method where the pair of nearest vertices between the parent-child adjacent spheres are selected as representative vertices, and their one-ring edges are used as the parent and child boundaries on the respective spheres, which are highlighted in green color (parent boundary) and red color (child boundary), respectively.

9.4 Limitation in Producing Split Effect

Although our approach enables the creation of customizable morphing styles, it does have a limitation in that it cannot produce an anatomic split effect, such as a mermaid splitting their tail fin into two legs or a person transforming into a mermaid by fusing two legs to form a tail. Similarly, it cannot support effects that involve splitting one finger into two. This is because our approach handles non-major spheres by following a hierarchical order, meaning that spheres can only merge with or grow from their parent and ancestor spheres. This hierarchical structure precludes the possibility of creating connections between sibling branches in the hierarchy, which should be necessary to achieve the aforementioned visual effects. However, our approach is able to create an effect of morphing the tail into one leg and growing the other leg from the pelvis.

Chapter 10

Conclusion and Future Work

In this dissertation, we introduce an innovative approach to cross-parameterization that transforms deforming characters into hierarchical spherical representations. This method not only aligns character features across these spherical domains but also preserves their structural interrelationships. Our technique stands out for its semi-automatic nature, simplifying the process for users by requiring them only to identify major joints, which then facilitates the automated alignment of features. Additionally, our method enhances the precision of alignment through the utilization of arbitrary surface feature points, enabling highly specific matches such as nail-to-nail or toe-to-toe correspondences for the utmost alignment accuracy.

Our comprehensive evaluation showcases the effectiveness of our approach in significantly mitigating distortion in areas of dense geometry and high curvature, thereby overcoming a notable challenge faced by conventional sphere-based parameterization techniques. Furthermore, our results reveal a marked improvement in alignment quality, attributed to the meticulous consideration of the hierarchical parent-child relationships among deforming features. This nuanced approach ensures a more coherent and structurally consistent alignment across different characters.

The versatility and robustness of our method have been thoroughly demonstrated across a wide array of applications. These include 3D morphing, where our approach enables seamless transitions between characters; texture transfer and blending, allowing for the creative reimagining of character appearances; character synthesis, facilitating the creation of new characters with diverse attributes; and deformation transfer, ensuring that character movements are realistically preserved across different models. Through these applications, our approach not only enhances the fidelity of character animations but also opens up new possibilities for 3D assets generation with morphological

variations.

In the future, we identify three promising research directions that could significantly enhance the efficiency of our cross-parameterization method. Firstly, the integration of advanced rig retargeting techniques could automate the selection of major joints - a process that currently requires manual input as part of our semi-automatic approach. The process of rig retargeting begins with the mapping of the source character's skeletal structure onto the target character. This involves identifying corresponding joints between the two characters and adjusting the target's skeletal rig to match the source's proportions and joint orientations as closely as possible. The challenge here lies in the inherent differences between characters, such as height, limb length, and body shape, which can significantly impact how animations are transferred. Advanced algorithms and methods have been developed to address these challenges, focusing on maintaining the fidelity of the transferred animations while accommodating anatomical differences [2, 10, 108]. By leveraging existing methodologies in rig retargeting , we anticipate streamlining the feature alignment phase, thereby reducing manual labor and increasing the overall efficiency of our system.

A second promising direction for future research that could further enhance the efficiency and effectiveness of our cross-parameterization method involves leveraging machine learning for feature matching [102, 118]. This advanced approach utilizes machine learning algorithms to automate the complex process of transferring skeletal rigs and animations from one character model to another, irrespective of differences in size, proportions, or geometry. By incorporating machine learning-based feature matching, we can streamline the identification of major joints and the subsequent feature alignment processes, making them more accurate and less labor-intensive. Machine learning models, particularly those based on deep learning, can be trained on extensive datasets comprising various character rigs and animation sequences. Through this training, the models develop an understanding of the intricate relationships between different skeletal structures and how to adapt animations to preserve motion dynamics and character integrity across diverse models. Furthermore, the application of machine learning in feature matching opens up the possibility of identifying and leveraging subtle, nuanced correlations in animation data that might be overlooked in manual or semi-automated processes. This could lead to more nuanced and lifelike animations, as the machine learning model would be capable of capturing and replicating the essence of the source animation with high fidelity, even when applied to a target character with a significantly different anatomical structure.

Thirdly, optimizing the execution performance of our approach through a parallel implementation presents a substantial opportunity for improvement [154]. Our current methodology involves a re-

cursive mesh decimation strategy to pinpoint the representative vertex during the vertex embedding stage, with an average processing rate of approximately 1.5k triangles per second. We aim to devise a metric capable of determining the representative vertex's position prior to initiating decimation. Such an advancement would facilitate the parallelization of the parameterization process across all segments, potentially elevating the execution speed significantly.

We believe that delving into these three research areas will markedly augment the practicality and applicability of our method. Automating the identification of major joints not only streamlines the initial setup but also aligns with the goal of reducing the manual intervention required. Concurrently, adopting a parallel processing approach could drastically reduce computation times, making our method even more appealing for a broad spectrum of computer graphics and animation applications. These enhancements promise to elevate our approach, making it a more efficient and user-friendly tool for professionals and researchers alike.

Bibliography

[1] Andreas A. Vasilakis and Ioannis Fudos. Pose partitioning for multi-resolution segmentation of arbitrary mesh animations. In *Computer Graphics Forum*, volume 33, pages 293–302. Wiley Online Library, 2014.

[2] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)*, 39(4):62–1, 2020.

[3] Noam Aigerman, Shahar Z Kovalsky, and Yaron Lipman. Spherical orbifold tutte embeddings. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[4] Marc Alexa. Recent advances in mesh morphing. In *Computer graphics forum*, volume 21, pages 173–198. Wiley Online Library, 2002.

[5] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. As-rigid-as-possible shape interpolation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, 2000.

[6] Maria-Elena Algorri and Francis Schmitt. Mesh simplification. In *Computer Graphics Forum*, volume 15, pages 77–86. Wiley Online Library, 1996.

[7] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. *Shape analysis and structuring*, pages 53–82, 2008.

[8] N Ashikhmin. Fast texture transfer. *IEEE computer Graphics and Applications*, 23(4):38–43, 2003.

[9] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[10] Quentin Avril, Donya Ghafourzadeh, Srinivasan Ramachandran, Sahel Fallahdoust, Sarah Ribet, Olivier Dionne, Martin de Lasa, and Eric Paquette. Animation setup transfer for 3d characters. In *Computer Graphics Forum*, volume 35, pages 115–126. Wiley Online Library, 2016.

[11] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics (TOG)*, 26(3):72, 2007.

[12] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. In *ACM SIGGRAPH 2009 papers*, pages 1–6. 2009.

[13] Pierre Bénard, Ares Lagae, Peter Vangorp, Sylvain Lefebvre, George Drettakis, and Joëlle Thollot. Dynamic solid textures for real-time coherent stylization. *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–8, 2011.

[14] Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello. Semi-supervised segmentation of 3d surfaces using a weighted graph representation. In *International workshop on graph-based representations in pattern recognition*, pages 225–234. Springer, 2011.

[15] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. *Computer Graphics Forum*, 24(3):611–621, 2006.

[16] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[17] Cheryl Briggs. *An Essential Introduction to Maya Character Rigging*. CRC Press, 2021.

[18] Guillermo D Canas and Steven J Gortler. Surface remeshing in arbitrary codimensions. *The Visual Computer*, 22:885–895, 2006.

[19] Lizhou Cao, Chao Peng, and Yangzi Dong. Ellic's exercise class: promoting physical activities during exergaming with immersive virtual reality. *Virtual reality*, 25:597–612, 2021.

[20] Lizhou Cao, Chao Peng, and Jeffrey T Hansberger. Usability and engagement study for a serious virtual reality game of lunar exploration missions. In *Informatics*, volume 6, page 44. MDPI, 2019.

[21] R Castilla, PJ Gamez-Montero, N Ertürk, A Vernet, M Coussirat, and E Codina. Numerical simulation of turbulent flow in the suction chamber of a gearpump using deforming mesh and mesh replacement. *International Journal of Mechanical Sciences*, 52(10):1334–1342, 2010.

[22] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. *Acm transactions on graphics (tog)*, 28(3):1–12, 2009.

[23] Xue Chen, Jieqing Feng, and Dominique Bechmann. Mesh sequence morphing. In *Computer Graphics Forum*, volume 35, pages 179–190. Wiley Online Library, 2016.

[24] Gary PT Choi, Amita Giri, and Lalan Kumar. Adaptive area-preserving parameterization of open and closed anatomical surfaces. *Computers in Biology and Medicine*, 148:105715, 2022.

[25] Keenan Crane, Ulrich Weischedel, and Max Wardetzky. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 32(4):61, 2013.

[26] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[27] Li Cui, Xin Qi, Chengfeng Wen, Na Lei, Xinyuan Li, Min Zhang, and Xianfeng Gu. Spherical optimal transportation. *Computer-Aided Design*, 115:181–193, 2019.

[28] Aukje De Boer, Martijn S Van der Schoot, and Hester Bijl. Mesh deformation based on radial basis function interpolation. *Computers & structures*, 85(11-14):784–795, 2007.

[29] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002.

[30] Huong Quynh Dinh, Anthony Yezzi, and Greg Turk. Texture transfer during shape transformation. *ACM Transactions on Graphics (TOG)*, 24(2):289–310, 2005.

[31] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[32] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

[33] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.

[34] Lubin Fan, Ligang Lic, and Kun Liu. Paint mesh cutting. In *Computer graphics forum*, volume 30, pages 603–612. Wiley Online Library, 2011.

[35] Zhengwen Fan, Xiaogang Jin, Jieqing Feng, and Hanqiu Sun. Mesh morphing using polycube-based cross-parameterization. *Computer Animation and Virtual Worlds*, 16(3-4):499–508, 2005.

[36] Michael S Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of computation*, 72(242):685–696, 2003.

[37] Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling*, pages 157–186, 2005.

[38] Pascal J Frey and Houman Borouchaki. Geometric surface mesh optimization. *Computing and visualization in science*, 1(3):113–121, 1998.

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

[40] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 130–150. ACM, 2006.

[41] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[42] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[43] Michael Garland and Paul S Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 263–269. IEEE, 1998.

[44] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[45] Abubakar Sulaiman Gezawa, Qicong Wang, Haruna Chiroma, and Yunqi Lei. A deep learning approach to mesh segmentation. *CMES-Computer Modeling in Engineering & Sciences*, 135(2), 2023.

[46] G. H. Golub and C. Reinsch. *Singular Value Decomposition and Least Squares Solutions*, pages 134–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 1971.

[47] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. Fundamentals of spherical parameterization for 3d meshes. In *ACM SIGGRAPH 2003 Papers*, pages 358–363. 2003.

[48] Xianfeng Gu, Yalin Wang, Tony F Chan, Paul M Thompson, and Shing-Tung Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE transactions on medical imaging*, 23(8):949–958, 2004.

[49] Xianfeng Gu and Shing-Tung Yau. Computing conformal structure of surfaces. *arXiv preprint cs/0212043*, 2002.

[50] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro, and Michael Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[51] Yasser Hamed, John Kahwaty, Andy Lin, Evan Goldberg, and Lawrence Chai. Crowd character complexity on big hero 6. In *ACM SIGGRAPH 2015 Talks*, pages 1–1. 2015.

[52] Victoria Hemming, Mark A Burgman, Anca M Hanea, Marissa F McBride, and Bonnie C Wintle. A practical guide to structured expert elicitation using the idea protocol. *Methods in Ecology and Evolution*, 9(1):169–180, 2018.

[53] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[54] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):138, 2016.

[55] Hanno Homann. Implementation of a 3d thinning algorithm. *Insight Journal*, 421, 2007.

[56] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, 1996.

[57] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, 1997.

[58] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, 1998.

[59] Kai Hormann, Bruno Lévy, and Alla Sheffer. Mesh parameterization: Theory and practice. 2007.

[60] Klaus Hormann and Günther Greiner. Mips: An efficient global parameterization method. *Curve and surface design*, 1:153–162, 2000.

[61] Xin Hu, Xiao-Ming Fu, and Ligang Liu. Advanced hierarchical spherical parameterizations. *IEEE transactions on visualization and computer graphics*, 24(6):1930–1941, 2017.

[62] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine-Hornung. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (TOG)*, 34(4):1–8, 2015.

[63] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and John P Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*. 2014.

[64] Doug L James and Christopher D Twigg. Skinning mesh animations. *ACM Transactions on Graphics (TOG)*, 24(3):399–407, 2005.

[65] Pushkar Joshi, Wen C Tien, Mathieu Desbrun, and Frédéric Pighin. Learning controls for blend shape based realistic facial animation. In *ACM Siggraph 2006 Courses*, pages 17–es. 2006.

[66] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. In *ACM SIGGRAPH 2010 papers*, pages 1–12. 2010.

[67] Lotan Kaplansky and Ayellet Tal. Mesh segmentation refinement. In *Computer Graphics Forum*, volume 28, pages 1995–2003. Wiley Online Library, 2009.

[68] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 954–961. ACM, 2003.

[69] Ladislav Kavan, P-P Sloan, and Carol O'Sullivan. Fast and efficient skinning of animated meshes. In *Computer Graphics Forum*, volume 29, pages 327–336. Wiley Online Library, 2010.

[70] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. Can mean-curvature flow be modified to be non-singular? In *Computer Graphics Forum*, volume 31, pages 1745–1754. Wiley Online Library, 2012.

[71] Khaled Khairy and Jonathon Howard. Spherical harmonics-based parametric deconvolution of 3d surface images using bending energy minimization. *Medical image analysis*, 12(2):217–227, 2008.

[72] Dawar Khan, Alexander Plopski, Yuichiro Fujimoto, Masayuki Kanbara, Gul Jabeen, Yongjie Jessica Zhang, Xiaopeng Zhang, and Hirokazu Kato. Surface remeshing: A systematic literature review of methods and research directions. *IEEE transactions on visualization and computer graphics*, 28(3):1680–1713, 2020.

[73] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics (TOG)*, 25(2):412–438, 2006.

[74] Patrick M Knupp. Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elements in Analysis and Design*, 39(3):217–241, 2003.

[75] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In *Graphics interface*, volume 98, pages 43–50. Citeseer, 1998.

[76] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (TOG)*, 23(3):861–869, 2004.

[77] Tsz-Ho Kwok, Yunbo Zhang, and Charlie CL Wang. Efficient optimization of common base domains for cross parameterization. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1678–1692, 2011.

[78] Guillaume Lavoué and Christian Wolf. Markov random fields for improving 3d mesh analysis and segmentation. In *3DOR@ Eurographics*, pages 25–32, 2008.

[79] Binh Huy Le and Zhigang Deng. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012.

[80] Binh Huy Le and Zhigang Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014.

[81] Hochang Lee, Sanghyun Seo, Seungtaek Ryoo, and Kyunghyun Yoon. Directional texture transfer. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 43–48, 2010.

[82] Sing Chun Lee and Misha Kazhdan. Dense point-to-point correspondences between genus-zero shapes. In *Computer Graphics Forum*, volume 38, pages 27–37. Wiley Online Library, 2019.

[83] Tong-Yee Lee, Yu-Shuen Wang, and Tai-Guang Chen. Segmenting a deforming mesh into near-rigid components. *The Visual Computer*, 22(9):729–739, 2006.

[84] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design*, 22(5):444–465, 2005.

[85] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002.

[86] Minglei Li and Liangliang Nan. Feature-preserving 3d mesh simplification for urban buildings. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:135–150, 2021.

[87] Yaqian Liang, Fazhi He, and Xiantao Zeng. 3d mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. *Integrated Computer-Aided Engineering*, 27(4):417–435, 2020.

[88] Wei-Hung Liao, Tsung-Ming Huang, Wen-Wei Lin, and Mei-Heng Yueh. Convergence of dirichlet energy minimization for spherical conformal parameterizations. *Journal of Scientific Computing*, 98(1):29, 2024.

[89] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4990–4998, 2017.

[90] Guoliang Luo, Frederic Cordier, and Hyewon Seo. Compression of 3d mesh sequences by temporal segmentation. *Computer Animation and Virtual Worlds*, 24(3-4):365–375, 2013.

[91] Guoliang Luo, Zhigang Deng, Xiaogag Jin, Xin Zhao, Wei Zeng, Wenqiang Xie, and Hyewon Seo. 3d mesh animation compression based on adaptive spatio-temporal segmentation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–10, 2019.

[92] Martin Madaras, Adam Riečickỳ, Michal Mesároš, Martin Stuchlík, and Michal Piovarči. Skeletex: Skeleton-texture co-representation for topology-driven real-time interchange and manipulation of surface regions. In *Computer Graphics Forum*, volume 37, pages 325–336. Wiley Online Library, 2018.

[93] Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. *Proceedings on Graphics interface '88*, pages 26–33, 1988.

[94] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34, 1993.

[95] Jonathan Maim, Barbara Yersin, Julien Pettre, and Daniel Thalmann. Yaq: an architecture for real-time navigation and rendering of varied crowds. *IEEE Computer Graphics and Applications*, 29(4):44–53, 2009.

[96] Emilie Marchandise, Jean-François Remacle, and Christophe Geuzaine. Optimal parametrizations for surface remeshing. *Engineering with Computers*, 30:383–402, 2014.

[97] Stefano Marras, Michael M Bronstein, Kai Hormann, Riccardo Scateni, and Roberto Scopigno. Motion-based mesh segmentation using augmented silhouettes. *Graphical Models*, 74(4):164–172, 2012.

[98] Donald E Marshall and Steffen Rohde. Convergence of the zipper algorithm for conformal mapping. *SIAM Journal on Numerical Analysis*, 45(6):2577–2609, 2007.

[99] André Medalha, Lucas Pagliosa, Afonso Paiva, and Paulo Pagliosa. Least-squares morphing of dynamic meshes. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 23–30. IEEE, 2017.

[100] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics (TOG)*, 22(3):562–568, 2003.

[101] Saad Nadeem, Zhengyu Su, Wei Zeng, Arie Kaufman, and Xianfeng Gu. Spherical parameterization balancing angle and area distortions. *IEEE transactions on visualization and computer graphics*, 23(6):1663–1676, 2016.

[102] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017.

[103] Stefano Nuvoli, Nico Pietroni, Paolo Cignoni, Riccardo Scateni, and Marco Tarini. Skinmixer: Blending 3d animated models. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022.

[104] Tina O'Hailey. *Rig it right! Maya animation rigging concepts*. Routledge, 2018.

[105] Junjun Pan, Lijuan Chen, Yuhan Yang, and Hong Qin. Automatic skinning and weight retargeting of articulated characters using extended position-based dynamics. *The Visual Computer*, 34(10):1285–1297, 2018.

[106] Chao Peng and Sabin Timalsena. Fast mapping and morphing for genus-zero meshes with cross spherical parameterization. *Computers & Graphics*, 59:107–118, 2016.

[107] Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

[108] Martin Poirier and Eric Paquette. Rig retargeting for 3d animation. In *Graphics interface*, pages 103–110, 2009.

[109] Rolandos Alexandros Potamias, Stylianos Ploumpis, and Stefanos Zafeiriou. Neural mesh simplification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18583–18592, 2022.

[110] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, 2000.

[111] Emil Praun and Hugues Hoppe. Spherical parameterization and remeshing. *ACM Transactions on Graphics (TOG)*, 22(3):340–349, 2003.

[112] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[113] Yi-Ling Qiao, Yu-Kun Lai, Hongbo Fu, and Lin Gao. Synthesizing mesh deformation sequences with bidirectional lstm. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[114] Richard A Roberts, Rafael Kuffner dos Anjos, Akinobu Maejima, and Ken Anjyo. Deformation transfer survey. *Computers & Graphics*, 94:52–61, 2021.

[115] Rui SV Rodrigues, José FM Morgado, and Abel JP Gomes. Part-based mesh segmentation: a survey. In *Computer Graphics Forum*, volume 37, pages 235–274. Wiley Online Library, 2018.

[116] Jose Luis Rubio-Tamayo, Manuel Gertrudix Barrio, and Francisco García García. Immersive environments and virtual reality: Systematic review and advances in communication, interaction and simulation. *Multimodal technologies and interaction*, 1(4):21, 2017.

[117] Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416, 2001.

[118] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[119] Scott Schaefer and Can Yuksel. Example-based skeleton extraction. In *Symposium on Geometry Processing*, pages 153–162, 2007.

[120] Patrick Schmidt, Janis Born, Marcel Campen, and Leif Kobbelt. Distortion-minimizing injective maps between surfaces. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[121] Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics (TOG)*, 39(4):119–1, 2020.

[122] Patrick Schmidt, Dörte Pieper, and Leif Kobbelt. Surface maps via adaptive triangulations. In *Computer Graphics Forum*, volume 42, pages 103–117. Wiley Online Library, 2023.

[123] Ariel Shamir. A survey on mesh segmentation techniques. In *Computer graphics forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008.

[124] Alla Sheffer and Eric de Sturler. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with computers*, 17(3):326–337, 2001.

[125] Alla Sheffer and John C Hart. Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Visualization, 2002. VIS 2002.*, pages 291–298. IEEE, 2002.

[126] Alla Sheffer, Emil Praun, Kenneth Rose, et al. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171, 2007.

[127] Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. Progressive embedding. *ACM Transactions on Graphics*, 38(4), 2019.

[128] Primoz Skraba, Maks Ovsjanikov, Frederic Chazal, and Leonidas Guibas. Persistence-based segmentation of deformable shapes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 45–52. IEEE, 2010.

[129] Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015.

[130] Michal Smolik and Vaclav Skala. Spherical rbf vector field interpolation: experimental study. In *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000431–000434. IEEE, 2017.

[131] John Snyder, Daniel N Wood, Steven J Gortler, Hugues Hoppe, and Robert Perry. Texture mapping progressive meshes. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 409–416, 1997.

[132] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 22(3):362–371, 2004.

[133] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[134] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics (TOG)*, 27(3):77, 2008.

[135] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[136] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer graphics forum*, 28(5):1383–1392, 2008.

[137] Jie Tan, Ignas Budvytis, and Roberto Cipolla. Variational autoencoders for deforming 3d mesh models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5841–5850, 2018.

[138] Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. Practical quad mesh simplification. In *Computer Graphics Forum*, volume 29, pages 407–418. Wiley Online Library, 2010.

[139] Haishan Tian, Yuanjun He, and Yong Wu. A new approach of progressive spherical parameterization. In *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, pages 5–pp. IEEE, 2005.

[140] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer graphics forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.

[141] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008.

[142] Shenghua Wan, Tengfei Ye, Maoqing Li, Hongchao Zhang, and Xin Li. Efficient spherical parametrization using progressive optimization. In *International Conference on Computational Visual Media*, pages 170–177. Springer, 2012.

[143] Chunxue Wang, Xin Hu, Xiaoming Fu, and Ligang Liu. Bijective spherical parametrization with low distortion. *Computers & Graphics*, 58:161–171, 2016.

[144] Yalin Wang, Xianfeng Gu, Shing-Tung Yau, and Xianfeng Guo. Conformal geometry and its applications on 3d shape matching, recognition, and stitching. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 29, pages 1209–1220. IEEE, 2007.

[145] Huai-Yu Wu, Chunhong Pan, Hongbin Zha, Qing Yang, and Songde Ma. Partwise cross-parameterization via nonregular convex hull domains. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1531–1544, 2010.

[146] Stefanie Wuhrer and Alan Brunton. Segmenting animated objects into near-rigid components. *The Visual Computer*, 26(2):147–155, 2010.

[147] Weiwei Xu, Kun Zhou, Yizhou Yu, Qifeng Tan, Qunsheng Peng, and Baining Guo. Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics (TOG)*, 26(3):84–es, 2007.

[148] Qing Yuan, Guiqing Li, Kai Xu, Xudong Chen, and Hui Huang. Space-time co-segmentation of articulated point cloud sequences. In *Computer Graphics Forum*, volume 35, pages 419–429. Wiley Online Library, 2016.

[149] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics (TOG)*, 24(1):1–27, 2005.

[150] Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Quan Zhou, Oliver Van Kaick, and Andrea Tagliasacchi. Deformation-driven shape correspondence. In *Computer Graphics Forum*, volume 27, pages 1431–1439. Wiley Online Library, 2008.

[151] Huadong Zhang, Lizhou Cao, and Chao Peng. Spherical parametric measurement for continuous and balanced mesh segmentation. 2023.

[152] Ruqin Zhang, Eliot Winer, and James H Oliver. Subdivision-based 3d remeshing with a fast spherical parameterization method. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 44113, pages 1039–1048, 2010.

[153] Xi Zhang, Guiqing Li, Yunhui Xiong, and Fenghua He. 3d mesh segmentation using mean-shifted curvature. In *International conference on geometric modeling and processing*, pages 465–474. Springer, 2008.

[154] Jieyi Zhao, Min Tang, and Ruofeng Tong. Mesh segmentation for parallel decompression on gpu. In *Computational Visual Media: First International Conference, CVM 2012, Beijing, China, November 8-10, 2012. Proceedings*, pages 83–90. Springer, 2012.

[155] Kun Zhou, Hujun Bao, and Jiaoying Shi. 3d surface filtering using spherical harmonics. *Computer-Aided Design*, 36(4):363–375, 2004.

[156] Guo-Jin Zou, Jun Hu, Xianfeng David Gu, and Jing Hua. Authalic parameterization of general surfaces using lie advection. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2005–2014, 2012.

Appendices

Appendix A

C++ Implementation of Hierarchical Alignment Equations

In Section 6.2.1, we performed a parent-to-child rigid alignment based on our multi-sphere representations. The C++ implementation of Equation 6.2, 6.3, 6.4, and 6.5, which calculates the alignment cost, are presented in listing A.1, A.2, and A.3. These functions are utilized in Algorithm 2 to determine the rotation matrix that minimizes the alignment cost.

Listing A.1: The implementation of Equation 6.2 calculates the lambda representations of the parent boundary for two corresponding spheres.

```
// Equation 6.2, calculate lamda representations
// Input: two corresponding spheres.
// Output: the lamda representation of the first sphere on the second
    sphere.
LamdaREP[] Alignment::CalculateLR(Sphere sphere1, Sphere sphere2) {
    LamdaREP lr_arr[] = new LamdaRep[sphere1.pb_SIZE]; // pb_Size is the
        size of the parent boundary
  for (int i = 0; i < sphere_1.pb_SIZE; i++) {
    float theta = sphere_1.parent_boundary_polarCoor[i];
        int k1 = sphere_2.findCombinationIndex(theta);
        int k2 = (k1 + 1) % sphere_2.pb_SIZE;
        theta1 = sphere_2. parent_boundary_polarCoor[k1];
        theta2 = sphere_2. parent_boundary_polarCoor[k2];

        LamdaRep[i].lamda = (theta2 - theta) / (theta2 - theta1);
```

```
        LamdaRep[i].index1 = k1;
        LamdaRep[i].index2 = k2;
  }
  return lr_arr;
}
```

Listing A.2: The implementation of Equation 6.3 calculates the target positions for the boundary vertices.

```
// Equation 6.3, calculate target positions
// Input: two corresponding spheres and the lamda representation of the
    first sphere on the second sphere
// Output: the target positions of the boundary vertices of the first
    sphere.
vec3[] Alignment::CalculateTP(Sphere sphere1, Sphere sphere2, LamdaREP
    lr_arr[]) {
    vec3 tp_arr[] = new vec3[sphere1.pb_SIZE];
    for (int i = 0; i < sphere_1.pb_SIZE; i++) {
        vec3 p1 = sphere2. parent_boundary[lr_arr[i].index1].pos;
        vec3 p2 = sphere2. parent_boundary[lr_arr[i].index2].pos;
        tp_arr[i] = p1 * lr_arr[i].lamda + p2 * (1.0f - lr_arr[i].lamda);
    }
    return tp_arr;
}
```

Listing A.3: The implementation of Equation 6.4 and 6.5 calculates the alignment cost.

```
// Equation 6.4 and 6.5, calculate alignement cost
// Input: two corresponding spheres and the target positions of the
    boundary vertices of both spheres.
// Output: the blended alignment cost.
float Alignment::CalculateAlignmentCost(Sphere sphere_A, Sphere sphere_B,
    vec3 tp_arr_A[], vec3 tp_arr_B[]) {
    // cost(a), equation 6.4
    float cost_A = 0.0f;
    for (int i = 0; i < size_A; i++) {
        vec3 p_start = sphere_A.parentSph.parent_boundary[i] .pos;
        vec3 p_target = tp_arr_A[i];
        cost_A += pow(ArcDistance(p_start, p_target), 2.0f);
    }

```

```
13      // cost(b), equation 6.4
14      float cost_B = 0.0f;
15      for (int i = 0; i < size_B; i++) {
16          vec3 p_start = sphere_B.parentSph.parent_boundary[i] .pos;
17          vec3 p_target = tp_arr_B[i];
18          cost_B += pow(ArcDistance(p_start, p_target), 2.0f);
19      }
20
21      // cost(rv), equation 6.4
22      float cost_rv = 0.0f;
23      for (int i = 0; i < sphere_A.rv.size(); i++) {
24          vec3 rv_A = sphere_A.rv[i].pos;
25          vec3 rv_B = sphere_B.rv[i].pos;
26          cost_rv += pow(ArcDistance(rv_A, rv_B), 2.0f);
27      }
28
29      // blend, equation 6.5
30      float cost = pow(1.0f + cost_A / size_A + cost_B / size_B, ALPHA) *
            pow(1.0f +  cost_rv / sphere_A.rv.size(), BETA);
31
32      return cost;
33  }
```