Efficient Ordinary Differential Equation-based Modelling and Skin Deformations for Character Animation

Shaojun Bian

A thesis submitted in partial fulfilment of the requirements
of Bournemouth University for the degree of

Doctor of Philosophy

National Centre for Computer Animation
Bournemouth University

January, 2018

Copyright statement

Acknowledgements

My sincere appreciation and thanks goes to my respected supervisors Professor Lihua You and Professor Jian Jun Zhang for giving me their invaluable advices and support throughout the duration of my Ph.D, helping me to solve tough problems and keep on making improvements. It is the most fortunate in my life to become one student of them. Completing this research is inseparable from their tireless instructions and help. I shall never forget the abundant time and effort Professor Lihua You put into my thesis and the direction from Professor Jian Jun Zhang in my research. I would also like to thank Bournemouth University and the National Centre for Computer Animation (NCCA) for providing me with a great learning environment, Ms. Sunny Choi and Ms. Jan Lewis, for all their supports.

This research is supported by grants of the project on human body animation funded by the Royal Society, United Kingdom (Grant Ref: IE131367), and China Scholarship Council, China. In this research, I have also gained knowledge and developed research skills during a two-month research project at Houston University, USA, and one-month research visit at Chinese Academy of Sciences, China.

I would like to thank my friends and research colleagues for their help and accompany, which make my research period more memorable.

Last but not least, I would also like to thank my parents and husband, who always give me unconditional love, support and courage to face challenges and cherish time to do right things.

Abstract

In the area of character animation, skin surface modelling, rigging and skin deforamtion are three essential aspects. Due to the different complexity of the characters, the time cost on creating corresponding skin surface model, animation skeleton in order to achieve diverse skin deformations, fluctuates from several hours to several weeks. More importantly, the data size of skin deformations could sharply influence the efficiency of generating animation. Smaller data size can also speed up character animation and transmission over computer networks. Over years, researchers have developed a variety of skin deformation techniques. Geometric skin deformation approaches have high efficiency but low realism. Example-based skin deformation approaches interpolate a set of given example poses to improve realism and effects that cannot be easily produced by geometric approaches. Physics-based skin deformation methods can greatly improve the realism of character animation, but require non-trivial training, intensive manual intervention, and heavy numerical calculations. Due to these limitations, many recent activities have initiated the research of integrating geometric, example-based, and physics-based skin deformation approaches.

The current research is to develop techniques based on Ordinary Differentical Equations (ODE) to efficiently create C2 continuous skin surfaces through two boundary curves, automatically generate skeleton to make the rigging process fast enough for highly efficient computer animation applications, and achieve physically realistic skin deformations for character animation by integrating geometric, physical and data-driven methods. Meanwhile, it is the first attempt to obtain an analytical solution to realistic physics-based skin deformations for highly efficient computation, to avoid the solving of a large set of linear equations, which largely reduces data size and computing time. The basic idea is to build ODE mechanics model, involve iso-parametric curves and Fourier Series representation, develop accurate and efficient solutions to calculate physical skin deformations through interpolating input realistic reconstructed 3D models. The proposed techniques will greatly avoid tedious

manual work, reduce data size, improve skin deformation realism, and raise efficiency of producing character animation.

Contents

List of Figures

List of Tables

Declaration

This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

Thesis Outline

This thesis consists of seven chapters.

- **Chapter 1 - Introduction:** This chapter provides a brief introduction to the importance of creating C2 ODE-based continuous skin surfaces, automatical skeleton generation, efficient physically realistic skin deformation, followed by conventional approaches for physics-based skin deformation, and finally the aims and framework of this research. At last, this chapter highlights the contributions of this research followed by a list of publications.
- **Chapter 2 - Related Work:** In this chapter, a thorough literature review of various skin deformation approaches is provided, following by a survey on automatically rigging and skin surface modelling techniques which are neccessary processes in this work's aimd ODE-based efficient physical realistic skin deformation method. These surveys review the strengths and weaknesses of state of the art methods, identifies some research challenges, and presents suggestions for future work.
- **Chapter 3 - Controllable C1 Surface Method for Blending:** This chapter proposes a new surface blending method. It can not only exactly satisfy blending boundary constraints but also achieve a desirable shape of blending surfaces. This blending method is based on the concept of sweeping surfaces controlled by fourth order ordinary differential equations. It creates blending surfaces by sweeping a three-dimensional curve called a generator along two three-dimensional trajectories and making the generator exactly satisfy the tangential constraints at the trajectories at the same time. The shape of blending surfaces is controlled by manipulating the generator with the solution to fourth order ordinary differential equations.
- **Chapter 4 - Interactive Creation of C2 Continuous ODE Skin Surfaces:** In this chapter, one efficient and simple technique to create tangential or curvature continuous surface and achieve the

smoothness between adjacent surface patches, through two boundary curves is introduced. Additionally, the proposed technique can be manipulated easily by simply changing first and second partial derivatives, and shape control parameters. This proposed technique also provides one direct ODE representation of character model for automatically generating skeleton to accelerate skin deformation process.

- **Chapter 5 - Automatic generation of animation skeleton to assist dynamic skin deformation:** This chapter outlines a new rigging algorithm for automatic generation of dynamic skin deformation to quickly identify iso-parametric curves and create an animation skeleton in a few milliseconds, which can be seamlessly used in curve-based skin deformation methods to make the rigging process fast enough for highly efficient computer animation applications.

- **Chapter 6 - Application of Fourier Transformation in physics-based skin deformation:** This chapter presents a new, simple, and efficient analytical approach for physics-based skin deformations, which is the first attempt to obtain an analytical solution to realistic physics-based skin deformations for highly efficient computation. Integrating physics-based and example-based approaches leads to more realistic skin deformations. This proposed technique utilize the physics-based mathematical model, in order to determine the correct force field acting on skin surfaces and achieve the realism of skin deformations. With the obtained analytic solution to the formulated ODE-based physics model, this approach transforms a discrete example mesh into its continuous Fourier series representations to avoid the solving of a large set of linear equations, which largely reduces data size and computing time.

- **Chapter 7 - Conclusion and Future Work:** This chapter demonstrates the conclusion of research presented in this thesis with its impact. It also discussed the limitations of the contributions presented and the possible future directions that can improve

this research in the further.

- **References**

Chapter 1

Introduction

1.1 Background

As essential and standardized parts of many character animation applications, various rigging and skinning techniques have been developed both in academia research and industry practices.

Existing rigging methods can be roughly classified into four different categories: *skeleton embedding*, *single shape-based skeleton extraction*, *motion-based skeleton extraction*, and *combination of skeleton embedding and extraction*. Skeleton embedding methods require optimally embedding pre-designed skeletons to skin meshes, which is often achieved through optimization Baran & Popović [2007]. However, its main disadvantage is the requirement for pre-designed skeletons. Rhodin et al. [2015] introduced one approach to address the problem that retargeting is hard for nonhuman characters, with locomotion bound to the sensing volume; and pose mappings are ambiguous with difficult dynamic motion control. Single shape-based skeleton extraction methods obtain skeletons from a single static pose Au et al. [2008]; Pan et al. [2009]. Since only one static pose is used, such down-sampling based approaches may fail when converting a curve skeleton into its corresponding animation skeleton. Motion-based skeleton extraction methods use motion data or multiple example poses to determine the skeleton and joint lo-

cations, and overcome the disadvantage of single shape-based skeleton extraction De Aguiar et al. [2008]; Le & Deng [2014]. The combination of skeleton embedding and extraction tackles the disadvantage of single shape-based skeleton extraction by locating a suitable template skeleton in the extracted curve skeleton through classification rules, derived from the general characteristics of each character class Pantuwong & Sugimoto [2012]. These methods mostly are not computationally efficient in general. For example, on a typical off-the-shelf computer, they often require at least a few seconds to complete automatic rigging for a model with several thousand vertices and tens of minutes (or even hours) for a model with tens of thousands of vertices.

Over the years, researchers also have developed a variety of skinning techniques Jacobson et al. [2014], including geometric skinning (e.g., the well-known linear blend skinning), example-based skinning (e.g., the pose space deformation Lewis et al. [2000]), and physics-based skinning. Traditionally, animators need to spend non-trivial efforts to set up rigs before a 3D model can be animated and deformed. As a result, not surprisingly, numerous previous efforts have been focused on generating suitable rigs for high quality skinning deformation. The usefulness of a skinning technique are generally measured in terms of three major characteristics: *realism*, *efficiency*, and *control*.

All of existing skinning techniques were designed to achieve certain trade-offs among the three factors. Purely geometric methods do not consider any physics of skin deformations. This type of methods are efficient in creating deformed skin shapes, but less realistic.

Data-driven methods create new skin deformations from known example skin shapes. When example skin shapes are sufficient, this type of methods can generate high realistic skin deformations. How to reduce the number of example skin shapes but still keep good realism is a main issue of data-driven methods. Since data-driven methods do not consider any physics of skin deformations, they require sufficient skin shapes to achieve realism.

Physics-based methods consider physics of skin and other tissues de-

formations. This type of methods can generate more realistic skin deformations at the cost of heavy calculations. On the other hand, Curve-based methods have also been proposed for modelling and simulating 3D character models in recent years. For example, Hyun et al. [2005] proposed sweep-based human deformation.

You et al. [2007, 2008] presented a curve-based sweeping surface and dynamic skin deformation method. Bartoň et al. [2013] discussed the theory, discretization, and numerics of curves which are evolving such that part of their shape, or at least their curvature as a function of arc length, remains unchanged. Bartoň et al. [2014] sought congruent planar curves to generate or approximately generate a freeform surface. Chaudhry et al. [2015] investigated a finite difference solution to curve-based dynamic skin deformations. Brandt et al. [2016] introduces techniques for the processing of motion and animations of non-rigid shapes. Although curve-based approaches simplify physics-based skinning, analytical solutions have not yet been presented to achieve both high efficiency and good realism of physics-based skin deformations.

1.2 Main Challenges

Skin deformation technique in character animation stays an essential and standardized part of many character animation applications these days including academia and industry practices. Over years, researchers have developed a variety of skin deformation techniques, including geometric skin deformation, example-based skin deformation, and physics-based skin deformation.

Geometric skin deformation approaches have high efficiency but low realism. Example-based skin deformation approaches interpolate a set of given example poses to improve realism or produce certain skin deformation effects that cannot be easily produced by geometric skinning approaches. Therefore, example-based skin deformation approaches are often used together with geometric skinning. In order to achieve satisfactory realism, how to optimally design or obtain such a set of example

poses is still considered as a widely open research problem. Physics-based skin deformation methods can greatly improve the realism of character animation, but require intensive manual intervention, and heavy numerical calculations. Due to these limitations, it is generally time-consuming to implement them, and difficult to achieve a high runtime efficiency. Integrating geometric and physics-based skin deformation into a same simulation framework can alleviate the computational burden of physics-based skin deformation and improve realism of geometric skin deformation.

Many recent activities have initiated the research of integrating geometric, example-based, and physics-based skin deformation approaches. In order to tackle the above limitations caused by numerical calculations of physics-based skin deformation, this research aims to develop more effective combination operations which could remedy the shortages of each kind of methods; especially to develop an fast analytical skin deformation approach with corresponding automatically generating skeleton. Since character surface generating and character rigging affect the preprocessing time, I also propose an interactive creation of C2 continuous ODE skin surface technique and one more efficient automatic rigging algorithm to efficiently reduce the preprocessing time and data size. These techniques can produce physically-realistic deformations with higher efficiency while does not need specialized physics-based knowledge, tedious manual operations and time-consuming numerical calculations.

1.3 Research Aims

As described above, physics-based skin deformation methods can greatly improve the realism of character animation, but require intensive manual intervention, and heavy numerical calculations. Due to these limitations, it is generally time-consuming to implement them, and difficult to achieve a high runtime efficiency. In order to tackle the above limitations caused by numerical calculations of physics-based skin deformation, this research aims to propose a simple, and efficient iso-parametric curve-based ana-

lytical approach to generate physically realistic skin deformations.

Since non-automatic rigging requires heavy human involvements, and various automatic rigging algorithms are less efficient in terms of computational efficiency. Especially for current curve-based skin deformation methods, identifying the iso-parametric curves and creating the animation skeleton need tedious and time-consuming manual work. Although several automatic rigging methods have been developed, but not aim at curve-based models. To tackle this issue, this research aims to develop an automatic rigging algorithm to accelerate the simulation of physically dynamic skin deformation for character animation.

In addition, when implementing Differential Equation physical model to generate realistic skin deformation for character animation, the technique of interactively creating C2 continuous character surface on the basis of ODE iso-parametric curves is beneficial and aimed to provide a direct representation of character model for automatically generating animation skeleton.

1.4 Contributions

To fulfill the aims and objectives of the research, this thesis have made three novel contributions, which are as follows.

- This research has developed a new surface blending method with positional and tangential continuities based on sweeping surfaces controlled by fourth order ODEs. The generator used to construct blending surfaces is created with the closed form solution to fourth order ODEs and its shape is manipulated by the shape control parameters involved in the equations. Blending surfaces are generated by moving the generator along two three-dimensional trajectories and making it exactly satisfy the tangential constraints at the trimlines. Due to the analytical nature of the proposed approach, it can generate various blending surfaces quickly. Since blending boundary constraints are explicitly included in the mathematical

expressions of the blending surfaces, the proposed approach is easy to use. The shape of the blending surfaces constructed with the proposed approach is controlled effectively by manipulating shape control parameters while still maintaining the same continuities at the trimlines. Apart from its capacity in creating a blending surface between two primary surfaces, it is also effective in blending more than two primary surfaces together.

- This work also proposes a new technique to interactively create C2 continuous surfaces for DE-based simulation of character skin deformation, providing one direct ODE representation of character model for efficiently automatically generating skeleton. With this approach, the creation of a three-dimensional surface is transformed into generating 2 boundary curves or 2 boundary curves plus 4 control curves and solving a vector-valued sixth order ordinary differential equation. Unlike the existing patch modelling approaches which require tedious and time-consuming manual operations to stitch two separate patches together and achieve the tangential or curvature continuity between two stitched patches, this proposed technique always maintains the C2 continuity between adjacent surface patches naturally which avoids manual stitching operations to produce character models. Besides, the technique presented in this work can achieve more shape variations defined by the same boundary constraints since the proposed technique can manipulate surfaces through shape control parameters, and the first and second partial derivatives.

- In the research, an efficient rigging algorithm for automatic generation of physically dynamic skin deformation is developed, including vertex identification on iso-parametric curves, automatic skeleton creation combined with curve-based skin deformation approach, in order to quickly identify iso-parametric curves and create an animation skeleton in a few milliseconds, which can be seamlessly used in DE curve-based skin deformation methods to make the rigging process fast enough for highly efficient computer animation applications.

- In this proposed approach, a new, simple, and efficient analytical approach for physics-based skin deformations is developed. This technique utilize the physics-based mathematical model, in order to determine the correct force field acting on skin surfaces and achieve the realism of skin deformations. With the obtained analytic solution to the formulated ODE-based physics model, this approach transforms a discrete example mesh into its continuous Fourier series representations to avoid the solving of a large set of linear equations, which largely reduces data size and computing time. Specifically, (1) employ Fourier series to convert 3D mesh models into continuous parametric representations through a conversion algorithm, which largely reduces data size and computing time but still keeps high realism, (2) introduce a Ordinary Differential Equation (ODE)-based skin deformation model to describe the underlying deformation dynamics between given poses, which is the first attempt to obtain an analytical solution to physics-based skin deformations for highly efficient computation.

1.5 Publications

The research presented in this thesis has led to the following publications:

1. **Shaojun Bian**, Zhigang Deng, Ehtzaz Chaudhry, Lihua You, Xiaosong Yang, Hassan Ugail, Xiaogang Jin, Zhidong Xiao, Jian Jun Zhang. Efficient and realistic character animation through analytical physics-based skin deformation, accepted and will be published by journal of Graphical Models, 2019.

2. **Shaojun Bian**, Anzong Zheng, Ehtzaz Chaudhry, Lihua You, Jian J. Zhang, Automatic generation of dynamic skin deformation for animated characters, Journal of Symmetry, 2018;

3. Ruibin Wang, Jian J. Zhang, **Shaojun Bian**, Lihua You, A survey of parametric modelling methods in high-speed train head design, Journal of Rail and Rapid Transit, 2017;

4. **Shaojun Bian**, Lihua You, Jian J. Zhang, Recent Developments in Skin Deformation for Character Animation, in the Proceedings of GRAPP 2015, published by SCITEPRESS, pp. 122-129;

5. E. Chaudhry, **Shaojun Bian**, Hassan Ugail, Xiaogang Jin, L. H. You, Jian J. Zhang, Dynamic Skin Deformation using Finite Difference Solutions for Character Animation, Computers and Graphics 46, pp. 294-305, 2015;

6. Hui Liang, Jian Chang, Xiaosong Yang, Lihua You, **Shaojun Bian**, Jian Jun Zhang, Advanced ordinary differential equation based head modelling for Chinese marionette art preservation, Computer Animation and Virtual Worlds 26, pp. 207-218, 2015.

Papers under revison:

1. **Shaojun Bian**, L.H. You, H. Ugail, Jian J. Zhang, Interactive creation of C2 continuous ODE surfaces, to be submitted.

2. O. Li, **Shaojun Bian**, I. Kazmi, L.H. You, Efficient sketch-based character modelling with primitive deformer and detail generator, to be submitted.

3. L.H. You, X.S. Yang, J.J. Pan, **Shaojun Bian**, .etc, Fast character modeling with sketch-based PDE surfaces, to be submitted.

Chapter 2

Related Work

The focus of my research is to develop a fast solution to achieve efficient physically realistic skin deformations. Since character surface generating and character rigging affect the preprocessing time, I also propose a more efficient automatic rigging algorithm and interactive creation of C2 continuous ODE skin surface technique to efficiently reduce the preprocessing time and data size. Accordingly, in what follows, I review existing work on skin deformations, character modelling and automatic rigging approaches.

2.1 Related Work on Skin Surface Creation

Polygon modeling and skin surface creation Russo [2006] have been widely applied in commercial available graphics package. This modeling approach can produce detailed or branched models, assign UV texture coordinates, and create hard edges more readily than NURBS modeling. However, polygons are incapable of accurately representing curved surfaces. Therefore, a large number of polygons must be used to approximate curved surfaces in a visually appealing manner. The typical patch modeling approach is NURBS modeling Piegl & Tiller [2012]Farin [2014]. With NURBS modeling, smooth curved objects can be created and edited with few control points. When rendered, both

low and high tessellation setting can be obtained from a same model. One obvious disadvantage for NURBS modeling is that a lot of manual operations are required to stitch adjacent patches together and deal with the continuity problem between different patches. Subdivision modeling Stam [1998]DeRose et al. [1998]Warren & Weimer [2001]Cashman et al. [2009] start the modeling with a coarse polygonal model, subdivides its polygonal faces into smaller faces through approximating or interpolating schemes, and generates a denser polygon mesh of the model. Subdivision makes the modeling of complex geometry more easily and rendering more efficiently. Some disadvantages of subdivision modeling are difficult to specify precision and lack of underlying parametrization.

Physics-based modeling considers the underlying physics of surface deformation. It has a capacity to create more realistic appearances. Various physics-based modeling approaches are reviewed by Nealen et al. [2006]. These approaches include finite element method, finite difference method, finite volume method, mass-spring systems, mesh-free methods, coupled particle systems, and reduced deformable models using modal analysis. Partial differential equation based geometric modeling was pioneered by Bloor & Wilson [1989].

The PDE based methods, since their advent two decades ago, have found their applications in a lot of surface modeling tasks, including free-form surface generation Bloor & Wilson [1990b], n-sided patch modeling Bloor & Wilson [1989], surface blending Bloor & Wilson [1989], and industrial applications Athanasopoulos et al. [2009]. Compared with the conventional surface modeling methods, the PDE-based methods provide the user with a higher level control of the shape of the generated surfaces using the parameters and the boundary conditions of the PDE instead of many hundreds of control points. Therefore they can be easily implemented as an easy to use interactive modeling package. However, before that can be realized, one serious hurdle need to be overcome, that is to solve the corresponding PDE efficiently. Currently, it is done either ad hoc or only for simple problems. For complicated problems, expensive numerical methods are still the only available choice, such as the finite element method Li [1998]Li & Chang [1999]Li & Chang [1999], finite dif-

ference method Cheng et al. [1990], and collocation point method Bloor & Wilson [1990a]. In order to improve the computational efficiency, the Fourier series method was proposed Bloor & Wilson [1996] although it is effective only when the high frequency modes are not strongly represented in the boundary conditions. In addition, another issue to be addressed is that the existing PDE based approaches only considered static modeling of surfaces. Dynamic modeling of curves and surfaces with up to curvature continuities using analytical Differentical Equations has not been investigated yet.

2.2 Related Work on Dynamic Skin Deformation and Automatic Rigging

Various approaches of skin deformations can be classified into four different types. They are: geometric skin deformation, example-based skin deformation, physics-based skin deformation, and hybrid skin deformations.

LBS is the most well-known **geometric skin deformation algorithm** Magnenat-Thalmann et al. [1988] due to its efficiency and simplicity. However, its limitations include collapsing elbow, candy-wrapper effects, and failure of secondary deformation Lewis et al. [2000]; Kavan et al. [2008]. In recent years, various geometric skinning methods Jacobson & Sorkine [2011]; Saito et al. [2015] have been proposed to overcome these limitations. In spite of its high efficiency, geometric skinning is less capable of creating highly realistic skin deformations.

Example-based skin deformation is employed to address the realism issue of geometric skinning, by learning deformation dynamics from a set of given examples Kavan et al. [2008]; Li et al. [2013]; Le & Deng [2012]. One of its major disadvantages is the design of a sufficient set of example skin shapes in order to produce realistic skin deformations, which is often a non-trivial task in practices.

Physics-based skin deformation simulate the underlying physics

to create realistic skin deformations Lee et al. [2009]; Kim & Pollard [2011]; Li et al. [2014]; Wang et al. [2015]. Physics-based skinning including dynamic skin deformation, is usually solved by numerical methods that require specialized knowledge and skills, making it difficult and time-consuming to implement and hard to reach a high animation rate. Therefore, it cannot be used for many real-time graphics applications.

2.2.1 Geometric skin deformation methods

Geometric skin deformation methods Linear Blend Skinning (LBS) also called skeleton subspace deformation (SSD) is the most well-known geometric skinning algorithm Magnenat-Thalmann et al. [1988] due to its efficiency and simplicity. However, its limitations include collapsing elbow, candy-wrapper effects, and failure of secondary deformation Lewis et al. [2000]. Sauvage et al. [2008] proposed one approach to address the deformation of B-spline surfaces while constraining the volume enclosed by the surface. Kilian et al. [2007] presented a novel framework to treat shapes in the setting of Riemannian geometry. A novel skinning algorithm based on linear combination of dual quaternions is presented in Kavan et al. [2008] to tackle some serious drawbacks of linear blend skinning. By introducing an extra scalar weight function per bone, a simple modification of the linear blend skinning (LBS) formulation was presented in Jacobson & Sorkine [2011] that enables stretching and twisting without changing the existing skeleton rig or bone weights. To remedy the problems like collapsing elbow and candy wrapper joint, curve skeletons along with the joint-based skeletons was used in Saito et al. [2015] to animate the skin shape. Le & Hodgins [2016] choose to pre-compute optimized centre of rotation for each vertex, and use these centres to interpolate rigid transformations. Vaillant et al. [2013] firstly provide one pure geometric method which could handle skin contact and muscle bulge problem in real-time, but fails to address deep self-intersections. To solve this problem, Vaillant et al. [2014] use new composition operators enabling blending effects and local self-contact between implicit surfaces, as well as a tangential relaxation scheme derived

from the as-rigid-as possible energy to solve the tracking problem. As above mentioned, in recent years, various geometric skinning methods have been proposed to overcome these limitations. But, in spite of its high efficiency, geometric skinning is still less capable of creating highly realistic skin deformations.

Due to the exiting problems of unrealistic deformation created by geometric based methods, such as the collapsing-joint, candy-wrapper, bulging-joint and distorted normal, more explorations have been launched. Here, four influential algorithms are illustrated, which could effectively improve skin deformation shapes generated geometrically.

Implicit Skinning with Contact Modelling

As traditional geometric-based skinning techniques, linear blending skinning(LBS)Magnenat-Thalmann et al. [1988] or dual-quaternion skinning(DQS) Kavan et al. [2008] are good at performances, which could meet the need of industry, but the deformations generated are less realistic, because of the collapsing-joint and candy-wrapper Magnenat-Thalmann et al. [1988] and bulging joint and distorted normal Kavan et al. [2008].

Implicit skinning with contact modeling is one purely geometric method, which could effectively address skin contact artifacts at joints and muscular bulges in real-time without using time-consuming collision detection (Vaillant et al. [2013]).

The merits of this method include maintaining the character volume after deformation, generating contact shapes and bulging near joint without any optimization and collision, so that the computing time could be saved.

The demerit of this method is that the deformed shape quality depends on the option of the initial geometric skinning method. When the method could avoid deep self-intersections, the results will be of high-quality.

Bulging-Free Dual Quaternion Skinning

This bulging free dual quaternion skinning method Kim & Han [2014] also considers both the shortage of LBS in collapsing-joint and candy-wrapper effects, and the problems of DQS in bulging joint and distorted normal.

In order to tackle the above mentioned skinning shortage, the first step, is to concern on correcting the positions of vertex. It pre-computes every vertex distance in the rest pose. When the vertex is in the bone-zone, the distance means to the bone. While the vertex in the joint-zone, the distance is to the joint. Then, use the run-time algorithm to correct vertex positions, pushing the red curve toward the corresponding bone or joint.

The second step is to correct the distorted-normals. Firstly, give every vertex a vector for reference. Then after deformation, another reference vector is rotated by a transform for every vertex. Finally, use the transform to correct the distorted normal. Some unnatural shading of the deformed skin could be eliminated.

This method mainly uses two procedures to solve the bulging joint and distorted problems of DQS. It is simple and easy to implement but the normal correction algorithm still faces computation overhead problem.

Stretchable and Twistable Bones Skinning

Stretchable and Twistable Bones for Skeletal Shape Deformation approach (STBS) Jacobson & Sorkine [2011] makes some modifications on the current popular method, skeleton-based linear blend skinning (LBS), to tackle the problems on elbow-collapse and candy-wrapper effects.

This approach could keep the original model skeleton rig and bone weights after stretching and twisting deformation, and still maintains a good performance.

Differential Blending Deformation

Creating a realistic character model and generating the diverse poses of the model in computer is increasingly difficult and time-consuming, generally because of two reasons. One is the character rig system may limit the space of achievable poses, and the other is that manipulating a character rig system to obtain desirable poses requires huge manual work, due to lots of the rigging parameters.

The Differential Blending approach Öztireli et al. [2013] introduced here, deals with the above mentioned shortages of skeletal deformation by using the 2D hand-drawn animation as a guide Blair [1994].

The core of this novel blending method stays in blending skeletal large and disparate transformations into small ones. Firstly, represent all transformations differentially. Then, calculate the averages of these transformations. Finally, obtain the desirable blended transformations between animation key-frames with much lower time and labour cost. The user draws a stroke to select a bone. Then transformations from the frames on the drawn curve to the select bone are computed to get the final deformed model.

2.2.2 Example-based skin deformation methods

Example-based skin deformation methods is employed to address the realism issue of geometric skinning, by learning deformation dynamics from a set of given examples Sloan et al. [2001]. An automated framework was presented in Mohr & Gleicher [2003] to fit the parameters of a deformation model using a set of examples consisting of skeleton configurations paired with the deformed geometry as static meshes. Rhee et al. [2006] develop one parallel deformation method using the GPU fragment processors. Joint weights for each vertex are automatically calculated from sample poses, thereby reducing manual effort and enhancing the quality of WPSD (Weighted Pose Space Deformation) as well as SSD (Skeletal Subspace Deformation). Park & Hodgins [2008] presents a data-driven technique for synthesizing skin deformation from

skeletal motion. Eulerian representation of skin was proposed in Li et al. [2013] to simulate thin hyperelastic skin that can stretch and slide over underlying body structures such as muscles, bones, and tendons. One automated algorithm, called Smooth Skinning Decomposition with Rigid Bones (SSDR), was introduced in Le & Deng [2012] to extract the linear blend skinning (LBS) from a set of example poses. It outperforms the state-of-the-art skinning decomposition schemes in terms of accuracy and applicability. One major disadvantage of example-based skin deformation methods is the design of a sufficient set of example skin shapes in order to produce realistic skin deformations, which is often a non-trivial task in practice.

Data-driven methods generate new skin deformations through example character skin models, without considering any underlying physics. Once example models are sufficient, this category of methods could create highly realistic skin deformations. Here, I mainly take research on four data-driven algorithms developed for decreasing the input character example data but still could accomplish high realistic results.

Smooth Skinning Decomposition

Smooth Skinning Decomposition with Rigid Bones (SSDR), is one effective approach that automatically extract linear blend skinning (LBS) from input example models.

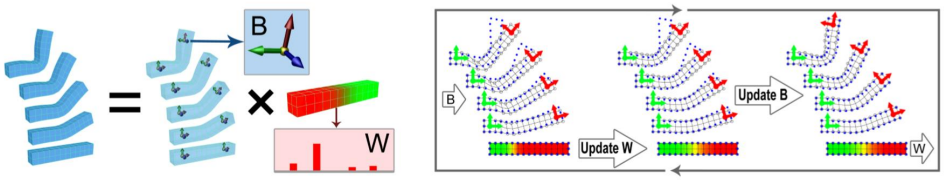


Figure 2.1: *Overview of this skinning decomposition method.(Le & Deng [2012])*

As shown in Fig. 2.1, B means rigid bone transformations and W means a sparse, convex bone-vertex weight map W Le & Deng [2012].a set of example models are decomposed into bone transformations and a sparse, convex bone-vertex weight map by the block coordinate descent algorithm (right hand side). During the process, the example poses (indicated as blue dots) can be reconstructed more accurately by alternatively updating W and B while the other is kept fixed. Only these little rigid bones and the weight map are used to simulate the skin deformations of character models by SSDR Le & Deng [2012]. More specifically, this skinning decomposition is solved as one constrained optimization problem.

The skinning methods in James & Twigg [2005] and Hasler et al. [2010] make a novel treatment called soft constraints which are the constraint of bone orthogonal transformation and bone-vertex convex weight map. But this SSDR technique treats these constraints as hard constraints to avoid the collision between totally satisfying the constraints and minimizing the reconstructing error. By employing the SSDR model, the bone transformation could be obtained simply and the deformed shapes are accurate but it needs high computational cost.

Enriching Coarse Interactive Deformation

Simulating elastic object is really necessary in character modeling area. Many efficient approximate deformation method have been developed, but they always cannot do good on simulating complex geometric models with nonlinear materials and dissatisfied computing cost.

Considering on the aforementioned problems, the enriching coarse method follows the idea that, the non-linearly deformation of geometric object could be decomposed as a superposition of an approximate model and displacements on deviation between approximate model and real geometric model Zhong et al. [2005]. It proposes one efficient dynamic interactive coarse model coupled with enriching details form a high-resolution quasi-static model in a data-driven way Seiler et al. [2012].

The first stage of this algorithm is the pre-computation. During this stage, theres an interactive tool which could be used to act on the object and create object deformation. The aforementioned procedure acts again but with higher resolution quasi-static simulation. After every example interaction, the difference between the two models is calculated as a displacement field for next times use, called stamp in this method.

The second stage of this algorithm is to obtain the approximated character deformation model by coarse simulating. And weight (w) is extracted to blend the stamps using non-linear correlation. Then according to the coarse model of object and the blended stamp, the high resolution model with enriching details could be produced Seiler et al. [2012]. This approach proposes the stamping way to enhance the quality of interpolation for simulating elastic object with details. But the usage direction on dynamic deformation should still be exploded.

Sparse Localized Components Deformation

This Sparse Localized deformation method decomposes a whole model deformation into some sparse and spatially localized modes through an animated sequence Neumann et al. [2013].

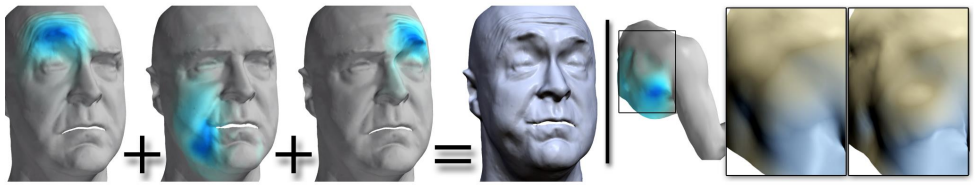


Figure 2.2: *decomposed sparse and local deformations (blue) could be added to create a new deformation needed (Neumann et al. [2013]).*

As shown in Fig. 2.2, summing several deformations of localized components produces one new facial expression. This method automatically decomposes any mesh animations like performance captured faces (left) or muscle deformations (right) into sparse and localized deformation modes (shown in blue). Left: a new facial expression is generated by summing deformation components. This method automatically separates spatially confined effects like separate eyebrow motions from the data. Right: This algorithm extracts individual muscle and bone deformations. The deformation components can then be used for convenient editing of the captured animation. Here, the deformation component of the clavicle is over-exaggerated to achieve an artistically desired look.

Firstly, a sparsity-inducing regularizer is edited for mesh deformation setting. Then design one mechanism to automatically decompose sparse and localized mesh components efficiently which could be guided by input data from user. Besides, one effective decomposition optimization way has also been developed.

Based on the important theories on matrix decomposition such as Non-Negative Matrix Factorization (NMF) Lee & Seung [2001], Robust PCA Candès et al. [2011], and Sparse PCA Zou et al. [2006], this new efficient data-driven algorithm could decompose mesh sequence into sparse

deformation components without considering the real underlying physical movements.

The sparse localized decomposition method highly deals with some tough mesh processing and editing tasks, such as animation editing on faces, body, cloth and statistical geometry processing.

Non-Linear Heterogeneous Soft Tissue Deformation

Recently, the methods on simulating soft object deformation have been developed to solve the heterogeneous materials problem. But it exactly is still a time-consuming work and another tough problem is material nonlinearities.

This data-driven method proposes one novel way to simulate the deformations of non-linear heterogeneous soft object. Finite element methods and a range of measured example objects deformation have been used, saving lots cost of choosing material parameters Bickel et al. [2009]. As always, a four stage process applies:

(1) Every measured example deformation of objects is transformed into a local element-wise strain space. (2) Model the stress-strain relation of material deformation into locally linear sample. (3) Through radial basis functions (RBFs) Buhmann [2003], interpolate and simulate the non-linear deformation of material in strain space. (4) Finally, by using an easy-to-implement elastostatic finite-element solution of the non-linear material examples based on incremental loading, the accurate soft object deformation models could be generated on lower computation cost Bickel et al. [2009].

2.2.3 Dynamic Skin Deformation and Automatic rigging

Dynamic Skin Deformation, Chaudhry et al. [2015] integrate geometric transformations, example-based, and physics-based approaches to simulate dynamic skin deformations. Xu & Barbič [2016] integrate

physics-based and pose-space skin deformations, where the latter combines SSD Magnenat-Thalmann et al. [1988] with artist-corrected pose shapes, and introduce pose-dependent model reduction to accelerate the finite element simulation for real-time dynamic skin deformation applications. Physics-based dynamic skin deformation techniques play a very important role in computer modelling and animation currently. This kind of methods could produce more realistic skin deformed shapes but commonly need high computing cost.

Automatic rigging, Generation and placement of an animation skeleton, is developed to avoid tedious and time-consuming manual rigging. Here, an animation skeleton typically consisting of bones and joints is used to drive the movements and deformations of skin meshes, while a curve skeleton is defined as the medial axis Pan et al. [2009] or a curve connecting the centres of closed curves on a skin mesh.

Existing automatic rigging methods can be roughly classified into four different categories: *skeleton embedding*, *single shape-based skeleton extraction*, *motion-based skeleton extraction*, and *combination of skeleton embedding and extraction*. Skeleton embedding methods require optimally embedding pre-designed skeletons to skin meshes, which is often achieved through optimization Baran & Popović [2007]. However, its main disadvantage is the requirement for pre-designed skeletons. Rhodin et al. [2015] introduced one approach to address the problem that retargeting is hard for nonhuman characters, with locomotion bound to the sensing volume; and pose mappings are ambiguous with difficult dynamic motion control. Single shape-based skeleton extraction methods obtain skeletons from a single static pose Au et al. [2008]; Pan et al. [2009]. Since only one static pose is used, such down-sampling based approaches may fail when converting a curve skeleton into its corresponding animation skeleton. Motion-based skeleton extraction methods use motion data or multiple example poses to determine the skeleton and joint locations, and overcome the disadvantage of single shape-based skeleton extraction De Aguiar et al. [2008]; Le & Deng [2014]. The combination of skeleton embedding and extraction tackles the disadvantage of single shape-based skeleton extraction by locating a suitable template skeleton

in the extracted curve skeleton through classification rules, derived from the general characteristics of each character class Pantuwong & Sugimoto [2012]. These methods mostly are not computationally efficient in general. For example, on a typical off-the-shelf computer, they often require at least a few seconds to complete automatic rigging for a model with several thousand vertices and tens of minutes (or even hours) for a model with tens of thousands of vertices. Rhodin et al. [2014] presented one way to control charactors for real-time animation, avoiding the rigging-skinning pipeline for arbitrary motion mapping. And Vögele et al. [2012] introduced one fast method to rapidly combine available sparse motion capture data with existing mesh sequences to produce a large variety of new animations.

2.3 Related Work on Physics-based Skin Deformation

2.3.1 Physics-based skin deformation methods

Physics-based skin deformation methods try to simulate the underlying physics to create more realistic skin deformations. Nedel & Thalmann [1998] propose a method to simulate human beings based on anatomy concepts. In Angelidis & Singh [2007], volume preserving method was presented to avoid extra bulge or wrinkle when deformation, and use vector field integration to avoid self-collision, however is still computationally expensive. A comprehensive biomechanical model of the human upper body was developed in Lee et al. [2009] that uses a coupled finite element model with the appropriate constitutive behavior to simulate biomechanically realistic flesh deformations and investigates an associated physics-based animation controller. A physically based simulation system for skeleton-driven deformable body characters is developed in Kim & Pollard [2011] that gives a well-coordinated combination of a reduced deformable body model with nonlinear finite elements, a linear-time algorithm for skeleton dynamics, and explicit integration

can boost simulation speed. An efficient algorithm based on a novel discretization of corotational elasticity over a hexahedral lattice is examined in McAdams et al. [2011] to achieve near-interactive simulation of skeleton driven, high resolution elasticity models, but it still need several seconds per animation frame. Hahn et al. [2012] formulate the equations of motions in the subspace of deformations defined by animator's rig, bringing the benefits of physics-based simulation to enhance the realism of traditional animation pipelines, but need tens of seconds when simulate one normal character model. A closed-form skinning method is proposed in Kavan & Sorkine [2012] to generate higher quality deformations than both linear and dual quaternion skinning through optimize skinning weights for the standard linear and dual quaternion skinning techniques and introducing joint-based deformers. Elastic animation editing with spacetime constraints was discussed in Li et al. [2014] that not only optimizes control forces added to a linearized dynamic model, but also optimizes material properties to better match user constraints and provide plausible and consistent motion. By minimizing quadratic deformation energy, built via a discrete Laplacian inducing linear precision on the domain boundary, a method was presented in Wang et al. [2015] to design linear deformation subspaces, unifying linear blend skinning and generalized barycentric coordinates. A fast physically based simulation system for skeleton-driven deformable body characters is developed in Jacobson et al. [2012] that gives a well-coordinated combination of a reduced deformable body model with nonlinear finite elements, a linear-time algorithm for skeleton dynamics, and explicit integration, to boost simulation speed. In Kry et al. [2012], EigenSkin has been presented to correct SSD results, it achieves high frames per second, but constructing one proper error-optimal eigen displacement basis requires sufficient experience and background knowledge. Since it is not easy to model the different elastic behaviors of muscle, fat and skin using simple volumetric mesh, Bender et al. [2013] introduce one novel multi-layer model to simulate them, but it fails handle collisions during fast motions. Heeren et al. [2012] presents one computational model for geodesics in the space of thin shells and incorporate bending contributions into deformation

energy, besides, Heeren et al. [2014] develop a time- and space-discrete geodesic calculus to shoot geodesics with prescribed initial data, for simulation realistically trade off, they all need heavy computation. Although physics-based skin deformations improve the realism, they require specific knowledge and skills, big computer memory, high computational cost, and low efficiency. In order to reduce the computational cost of physics-based skin deformations, model reduction has been introduced into dynamic Teng et al. [2014] deformation simulations. It can achieve real-time performance, but reduce the computational accuracy and increase the implementation complexity. Elastic animation editing with spacetime constraints was proposed in Li et al. [2014], where optimized control forces are added to a linearized dynamic model, and material properties are optimized to better match user constraints and provide plausible and consistent motion. Recently, Wang et al. [2015] design linear deformation subspaces by minimizing quadratic deformation energy.

Physics-based techniques play a very important role in computer modelling and animation currently. This kind of methods could produce more realistic skin deformed shapes but commonly need high computing cost. Here, I mainly review four physics algorithms which fairly solve the tough tasks on deformation of complex heterogeneous objects and soft materials.

Sparse Meshless Model

Physics-based methods consider the physics principles of skin and the material attributes. Because of the complex heterogeneous material of real objects, common methods often regard it into one homogeneous material for modelling. Once taking use of current method for modelling the complex heterogeneous objects realistically, it needs to deal with lots of varying material parameters which seems unfeasible previously.

The Sparse Meshless model Faure et al. [2011] of complex deformable solids deals with above questions using various stiffnesses to simulate complex heterogeneous objects. By maintaining the frame-based meshless framework introduced in Gilles et al. [2011], this method obtains

the physical realism of character animation by using skeleton subspace deformation (SSD) on character volume and continuum mechanics.

Compared with previous approaches, this Faure et al. [2011] model adapts coarse deformation functions to efficiently simulate objects of complex heterogeneous material at a high performance and less control nodes but the accuracy should be improved.

Efficient Elasticity Technique

As for the high computational cost of physically based approach to generate the life-like human and animal models, geometric or data-driven skinning approaches are always used. But in that case, the pinch-free geometry could not be preserved. Therefore, some previous works have been done to simplify the physical simulation. The principle component analysis of off-line elasticity simulation Kry et al. [2002] is use to enhance the interaction of physics-based SSD.

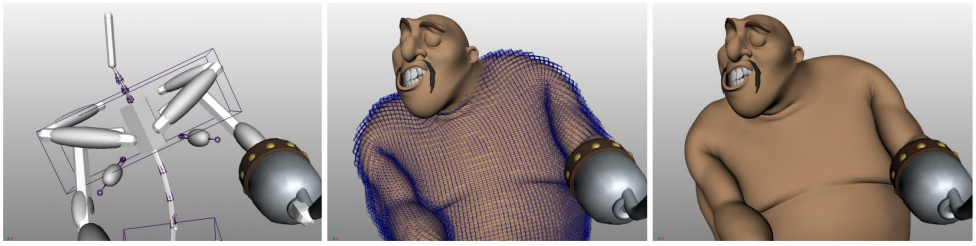


Figure 2.3: *Overview of the efficient elasticity model (McAdams et al. [2011]).*

The novel elasticity model introduced in McAdams et al. [2011] focuses on solving the soft tissue deformation problems. It innovatively discretizes co-rotational elasticity over a hexahedral lattice to diminish the self-collision artifacts and maintains soft-constraints for character real-

ism.

As shown in Fig. 2.3, there's a character mesh and its skeleton (left). Then a corresponding hexahedral lattice is defined (middle). The original mesh is deformed by the rules of self-collision and volumetric elasticity.

Taking a reference on Chao et al. [2010], this corotational elasticity discretization method accurately treats the force into derivatives to get a more robust solver than the simplified warped-stiffness approaches with little manual cost.

Skeleton and Skin Coupled Physics Framework

Recently some outcomes promote the controlling of human-like rigid characters Yin et al. [2007],Coros et al. [2010] and highly dynamic motions Liu et al. [2012], Brown et al. [2013]. But sometimes the motions of character skin and soft body always influence the dynamic of skeleton. In the long run, biomechanical algorithms that truly simulate human anatomy are exactly necessary to avoid the problem Lee et al. [2009], unfortunately still need high computing cost.

Here, this physically based framework for simulating and controlling life-like soft material characters could couples the dynamics of skeleton and soft body Liu et al. [2013]. In detail, this simulation and control system works as:

(1) Take the character skeleton and surface mesh as input data. (2) In order to couple the skeleton dynamic and skin geometry, user should construct one coarse volumetric mesh with a reference configuration X. More exactly, one soft body dynamics solver is obtained to construct the volumetric mesh. And a rigid body dynamics solver is obtained to simulate the character skeleton.

By coupling the dynamics of skeleton and skin, this physics-based framework shows good performance on character large deformation and joint effects.

Embedded Thin Shells Wrinkle Deformation

Wrinkles simulation takes a very important part in object deformation. When the material properties of the surface and underlying volume change, wrinkles will happen, causing by a force. The wrinkle appearances commonly occur on human skin Danielson [1973], but also could appear at other kinds of objects, like fruits and mountain Genzer & Groenewold [2006].

The embedded thin shells framework showed in Rémillard & Kry [2013] could highly simulate complex object with a soft interior and a harder skin. The core of it is to combine high resolution thin shells with coarse finite element lattices and confirm frequency based constraints. And it could generate the predicted wrinkle by calculating the physical parameters of characters. This method also adapts one novel two-way coupled model to eliminate the computational cost of internal volumetric elements Rémillard & Kry [2013]. To elaborate, this technique has the following phases:

(1) Taking use of the embedded mesh method and replaces the embedded mesh with a thin shell, combining both systems just with position constraints. (2) Considering the constraints. They should be non-intervention with wrinkle formation or large character deformation. (3) C1 quadratic shape functions to represent the interior deformations, achieving seamless effect on discretization boundaries.

This solver produces static solutions for the shell. These shells are thin enough and cannot cause visual dynamics. The high-resolution deformation of these shells could be used to contribute forcing on the low-resolution interior dynamics Rémillard & Kry [2013]. Thus, the process could largely eliminate the cost of deforming the interior of objects.

2.3.2 Curve-based skin deformation methods

Curve-based methods have also been proposed for modeling and simulating 3D character models in recent years. For example, Hyun et al.

[2005] proposed sweep-based human deformation. You et al. [2007, 2008] presented a curve-based sweeping surface and dynamic skin deformation method. Bartoň et al. [2013] discussed the theory, discretization, and numerics of curves which are evolving such that part of their shape, or at least their curvature as a function of arc length, remains unchanged. Bartoň et al. [2014] sought congruent planar curves to generate or approximately generate a freeform surface. Chaudhry et al. [2015] investigated a finite difference solution to curve-based dynamic skin deformations. Brandt et al. [2016] introduces techniques for the processing of motion and animations of non-rigid shapes. Although curve-based approaches simplify physics-based skinning, analytical solutions have not yet been presented to achieve both high efficiency and good realism of physics-based skin deformations.

The above discussions indicate that geometric, example-based, and physics-based skin deformations have their own strengths and weaknesses. Integrating them together can maximize their strengths and minimize their weaknesses. Inspired by You et al. [2008] which uses a curve defined model to animate skin deformation, Chaudhry et al. [2015] integrates geometric transformations, example-based and physics-based approaches to simulate dynamic skin deformations. Xu & Barbič [2016] integrates physics-based and pose-space skin deformations where the latter combines skeleton subspace deformation (SSD) Magnenat-Thalmann et al. [1988] with artist-corrected pose shapes, and proposes pose-dependent model reduction to accelerate the finite element simulation for hard real-time applications. Similar to Xu & Barbič [2016] in spirit, this paper integrates SSD, example-based and physics-based skin deformations. Different from Xu & Barbič [2016] which develops a complicated numerical approach to deal with linear and nonlinear elastic dynamic skin deformations including transient and steady-state skin vibrations, this paper proposes a simple analytical solution to cope with linear elastic dynamic skin deformations and steady-state skin vibrations which is highly realistic and efficient, easy to learn, implement and use, and requires much fewer manual operations and far smaller computer memory requirement.

Chapter 3

Controllable C1 Surface Method for Blending

Surface blending with tangential continuity is most widely applied in computer aided design etc. For such surface blending, two issues should be addressed. One is to exactly satisfy blending boundary constraints, and the other is to achieve a desirable shape of blending surfaces. Surface blending with tangential continuity can be divided into G^1 and C^1. G^1 continuous surface blending achieves a desirable shape of blending surfaces by adjusting the magnitude of boundary tangents in blending boundary constraints. C^1 continuous surface blending can exactly satisfy blending boundary constraints but may have no capacity to adjust the shape of blending surfaces.

This chapter proposes a new surface blending method. It can not only exactly satisfy blending boundary constraints but also achieve a desirable shape of blending surfaces. This blending method is based on the concept of sweeping surfaces controlled by fourth order ordinary differential equations. It creates blending surfaces by sweeping a three-dimensional curve called a generator along two three-dimensional trajectories and making the generator exactly satisfy the tangential constraints at the trajectories at the same time. The shape of blending surfaces is controlled by manipulating the generator with the solution to fourth order ordinary differential equations. Since the blending boundary constraints

are explicitly incorporated in the analytical mathematical equations of the blending surfaces, it is easy to use and generates blending surfaces quickly. In addition, it can blend more than two primary surfaces together, and manipulate the shape of the blending surfaces effectively while keeping the same continuity at the trimlines between the blending surface and primary surfaces.

3.1 Introduction

Surface blending is to generate a transition surface which connects two or more surfaces together seamlessly and smoothly. The surfaces to be connected are called primary surfaces or base surfaces and the transition surface is called a blending surface. The interface curves between the blending surface and the primary surfaces are called trimlines or boundary curves. The constraints which a blending surface must satisfy at the trimlines are called blending boundary constraints.

Surface blending is widely used in many applications such as computer-aided design, manufacturing systems, and character modeling to achieve a smooth transition between two or more separate primary surfaces for strength, manufacturing, aesthetic and usage purposes. Due to its wide range of applications, various surface blending approaches have been proposed Vida et al. [1994].

Rolling-ball blending is the most popular method. It was proposed by Rossignac and Requicha Rossignac & Requicha [1984]. Rolling-ball blending can be used to blend both implicit and parametric surfaces. Depending on whether the radius of the rolling ball changes or not, rolling-ball blending can be divided into constant-radius and variable-radius blends. Constant-radius blending was investigated in Rossignac & Requicha [1984]; Choi & Ju [1989]; Barnhill et al. [1993]; Farouki & Sverrisson [1996]; Kós et al. [2000]. Variable-radius blending was examined in Chuang et al. [1995]; Chuang & Hwang [1997]; Lukács [1998]. Recently, Whited and Rossignac introduced the concept of relative blending and a set theoretic formulation for variable-radius blending Whited &

Rossignac [2009]. Blending surfaces constructed with rolling ball methods are circular. Some other blending methods can create noncircular blending surfaces such as branching blends with Pythagorean normal surfaces Krasauskas [2008], vertex blending using S-patches Zhou & Qian [2009, 2010], N-sided hole filling Schichtel [1993]; Hsu & Tsay [1998]; Piegl & Tiller [1999]; Li & Li [2002]; Hwang et al. [2003]; Yang et al. [2006]; Shi et al. [2010], and the following partial differential equation-based blends.

Surface blending using partial differential equations is an effective approach especially in dealing with various blending problems between two separate primary surfaces. With such an approach, a blending surface can be constructed from the solution to a vector-valued partial differential equation which satisfies the positional, tangential or higher order continuities at trimlines. Partial differential equation-based surface blending was pioneered in Bloor & Wilson [1989]. Since closed form solutions to partial differential equations are difficult to obtain, some numerical and approximate analytical solutions were investigated. Li Li [1998]; Li & Chang [1999] and Li and Chang Li & Chang [1999] proposed boundary penalty finite element methods of surface blending. Bloor and Wilson developed a perturbation method to generate blending surfaces Bloor et al. [2000]. They also investigated a pseudo-spectral method for the construction of regular 4-sided patches Bloor & Wilson [2005]. You et al. presented approximate analytical methods You et al. [2004b,a]. In addition to partial differential equation-based surface blending, solid modelling using partial differential equations has also been investigated in Bloor & Wilson [1993]; You et al. [2011].

In contrast to partial differential equations which are usually solved by numerical or approximate analytical methods, the accurate analytical solution to ordinary differential equations (ODEs) involving x, y and z components can be obtained easily. The geometric representation of such a solution is a three-dimensional curve.

Any two and three-dimensional surfaces can be defined with curves. Therefore, these surfaces can be created and manipulated through curves.

Some approaches have been developed for this purpose. Singh and Fiume used wire curves to describe the shape of an object, and introduced domain curves to define the domain of deformation on the object Singh & Fiume [1998]. Pyun et al. extracted wire curves and deformation parameters from a facial model for facial animation Pyun et al. [2004]. Qin and Wright proposed an approach to construct freeform surfaces from unorganized curves Qin & Wright [2006]. Yoon and Kim generated freeform deformations by sweeping key cross-section curves Yoon & Kim [2006]. Nealen et al. designed freeform surfaces from 3D curves Nealen et al. [2007]. You et al. used characteristic curves to describe dynamic skin deformations You et al. [2008]. Liu et al. discussed how to use nonparallel curve networks to construct surfaces Liu et al. [2008]. Gal et al. employed intelligent wire curves as basic primitives to edit surface models Gal et al. [2009].

How to manipulate the shape of a blending surface is also an interesting topic. It has been investigated in the existing literature. For example, the work reported in Filip [1989]; Hoschek et al. [1993] achieves different shapes of a blending surface by scaling the length of boundary tangents. Although such a treatment cannot satisfy blending boundary tangents exactly, it can achieve visually continuity (G^1 continuity) between the blending and primary surfaces.

The main aim of this method is to propose a new surface blending method which not only manipulates the shape of blending surfaces effectively, but also satisfies blending boundary constraints exactly. This new blending method constructs a C^1 continuous blending surface by sweeping a curve called a generator along two trimlines called trajectories and exactly satisfies positional and tangential continuities at the trimlines. The generator is created and manipulated by the solution to fourth order ODEs subjected to blending boundary constraints consisting of trimlines and first partial derivatives of the primary surfaces to be blended at the trimlines.

In what follows, firstly discuss the mathematical model and closed form solution of curve-based surface blending in Section 3.2. Then im-

plement and validate the proposed closed form solution in Section 3.3. Next, examine shape manipulation of blending surfaces and make a comparison between the proposed approach and surface blending based on the cubic Hermite interpolation in Section 3.4. Finally, use an example to demonstrate the application of the proposed method in blending more than two separate primary surfaces together in Section 3.5 and 3.6. This work also investigates how to approximate the proposed approach for CAD-system processing in Section 3.7 and conclude the work given in Section 3.8.

3.2 Mathmatical Model and Solution

The smoothness between primary and blending surfaces can be described with different continuities. In most situations, blending surfaces with up to tangential continuities are required. For the parametric representation of blending surfaces, the positional continuity is described by the mathematical equation of primary surfaces at trimlines, and the tangential continuity is defined by first partial derivatives of primary surfaces at the trimlines.

Therefore, for surface blending with positional and tangential continuities, blending boundary constraints can be written as,

$$\begin{aligned} u = 0 \quad & S(u,v) = C_0(v), \quad \partial S(u,v)/\partial u = \bar{C}_0(v) \\ u = 1 \quad & S(u,v) = C_1(v), \quad \partial S(u,v)/\partial u = \bar{C}_1(v) \end{aligned} \tag{3.1}$$

where the vector-valued function $S(u,v) = [S_x(u,v)S_y(u,v)S_z(u,v)]^T$ is the mathematical description of a blending surface, u and v are parametric variables, and $C_0(v) = [C_{0x}(v)C_{0y}(v)C_{0z}(v)]^T$ and $C_1(v) = [C_{1x}(v)C_{1y}(v)C_{1z}(v)]^T$ are the vector-valued positional functions of primary surfaces at the trimlines, $\bar{C}_0(v) = [\bar{C}_{0x}(v)\bar{C}_{0y}(v)\bar{C}_{0z}(v)]^T$ and $\bar{C}_1(v) = [\bar{C}_{1x}(v)\bar{C}_{1y}(v)\bar{C}_{1z}(v)]^T$ are the first partial derivatives of primary surfaces at the trimlines which stand for the tangential constraints.

The determination of the functions $C_0(v)$, $\bar{C}_0(v)$, $C_1(v)$ and $\bar{C}_1(v)$ in blending boundary constraints (1) has been solved in the existing literature. If the trimline $C_i(v)(i = 0or1)$ is an isoparametric line of a primary surface $P(u, v)$, $C_i(v)$ can be easily determined by taking the parametric variable U to be the parametric value u_m of the isoparametric line, i.e., $C_i(v) = P(u_m, v)$. Similarly, the first partial derivative of the primary surface at the isoparametric line can be formulated as $\bar{C}_i(v) = \partial P(u_m, v)/\partial u(i = 0or1)$.

If the trimline is not an isoparametric line of a primary surface $P(s, t)$ where s and t are two parametric variables, the methods described in Hoschek et al. [1993] can be used to determine an arbitrary trimline on the primary surface and those presented in Filip [1989]; Hoschek et al. [1993]; Koparkar [1991]; Bardis & Patrikalakis [1989] can be used to find a tangent in the tangent plane of the primary surface $P(s, t)$ at a point of the trimline. The arbitrary trimline and tangent can be formulated as $C_i(v) = P(s(v), t(v)) = P(v)$ and $\bar{C}_i(v) = T(v)$ such as $T(v) = l(v)\partial P(s(v), t(v))/\partial s + k(v)\partial P(s(v), t(v))/\partial t$ in Koparkar [1991] where V is a curve parameter, and $l(v)$ and $k(v)$ are two functions or constants which can be selected to manipulate the tangent $T(v)$.

In order to manipulate the shape of blending surfaces, the parameters which can affect the shape but have no influence on the blending boundary constraints should be introduced into the mathematical description of the blending surfaces. A vector-valued ordinary differential equation (ODE) provides a good way of introducing such parameters due to the following reason.

The geometric representation of the solution to a vector-valued ODE is a three-dimensional curve and a blending surface can be created by sweeping the curve, called a generator, along two trimlines and exactly satisfying the boundary tangent at the trimlines at the same time. The parameters involved in the differential equation have an influence on the shape of the curve and can be used as a shape manipulator of the blending surface. In this chapter, blending surfaces created with this approach are called ODE blending surfaces.

Since fourth order differential equations involve four unknown constants which can be used to satisfy the four positional and tangential functions in 3.1, I propose to use the following equations for surface blending with positional and tangential continuities.

$$\alpha\frac{d^4G(u)}{du^4}+\beta\frac{d^2G(u)}{du^2}+\gamma G(u)=0 \tag{3.2}$$

where $G(u)=[G_x(u)G_y(u)G_z(u)]^T$ is a vector-valued function defining a generator which is used to create sweeping surfaces, and α, β and γ are called shape control parameters which serve as shape control handles for the generator.

The vector-valued ODE Eq. 3.2 can be changed into an algebraic equation by taking each component of the vector-valued function of the generator to be

$$\begin{aligned} &G_\gamma(u)=e^{ru}\\ &(\gamma=x,y,z) \end{aligned} \tag{3.3}$$

Substituting Eq. 3.3 and the second and fourth derivatives of $G(u)$ with respect to the parametric variable u into Eq.3.2 and deleting e^{ru}, the following algebra equation is reached,

$$\alpha r^4+\beta r^2+\gamma=0 \tag{3.4}$$

Depending on different combinations of shape control parameters, Eq. 3.4 has different solutions. Here only give the solution for $\beta^2=4\alpha\gamma$ and $\alpha/\beta<0$. All other solutions are given in Appendix A.

For $\beta^2 = 4\alpha\gamma$ and $\alpha/\beta < 0$, the roots of Eq. 3.4 are,

$$r_{1,2,3,4} = +- q_1 \tag{3.5}$$

where

$$q_1 = \sqrt{-\beta/(2\alpha)} \tag{3.6}$$

With the roots given in Eq. 3.5, the solution of Eq. 3.2 is,

$$G(u) = d_1 e^{q_1 u} + d_2 u e^{q_1 u} + d_3 e^{-q_1 u} + d_4 u e^{-q_1 u} \tag{3.7}$$

where d_1, d_2, d_3 and d_4 are vector-valued unknown constants.

In order to determine the unknown constants in Eq. 3.7, substitute it into Eq. 3.1, conduct the sweeping operation by solving for the four vector-valued unknown constants d_1, d_2, d_3 and d_4, and obtain,

$$\begin{aligned}
d_1 &= [a_{22} f_1(v) - a_{12} f_2(v)]/a_0 \\
d_2 &= [a_{11} f_2(v) - a_{21} f_1(v)]/a_0 \\
d_3 &= C_0(v) - [a_{22} f_1(v) - a_{12} f_2(v)]/a_0 \\
d_4 &= \bar{C}_0(v) + q_1 C_0(v) - 2q_1 [a_{22} f_1(v) - a_{12} f_2(v)]/a_0 \\
&\quad -[a_{11} f_2(v) - a_{21} f_1(v)]/a_0
\end{aligned} \tag{3.8}$$

where,

$$a_0 = a_{11}a_{22} - a_{12}a_{21} \tag{3.9}$$

and,

$$\begin{aligned}
a_{11} &= e^{q_1} - (1+2q_1)e^{-q_1} \\
a_{12} &= e^{q_1} - e^{-q_1} \\
a_{21} &= q_1 e^{q_1} - q_1 e^{-q_1} + 2q_1^2 e^{-q_1} \\
a_{22} &= (1+q_1)e^{q_1} - (1-q_1)e^{-q_1} \\
f_1(v) &= -(1+q_1)e^{-q_1}C_0(v) - e^{-q_1}\bar{C}_0(v) + C_1(v) \\
f_2(v) &= [q_1 e^{-q_1} - (1-q_1)q_1 e^{-q_1}]C_0(v) - (1-q_1)e^{-q_1}\bar{C}_0(v) + \bar{C}_1(v)
\end{aligned} \tag{3.10}$$

Substituting Eq. 3.8 back into Eq. 3.7, the mathematical equation of the swept surfaces for and can be written as,

$$\begin{aligned}
S(u,v) = [e^{-q_1 u} + q_1 u e^{-q_1 u} - (1+q_1)e^{-q_1}g_1(u) + [q_1 e^{-q_1} \\
-(1-q_1)q_1 e^{-q_1}]g_2(u)]C_0(v) + [ue^{-q_1 u} - e^{-q_1}g_1(u) \\
-(1-q_1)e^{-q_1}g_2(u)]\bar{C}_0(v) + g_1(u)C_1(v) + g_2(u)\bar{C}(v)
\end{aligned} \tag{3.11}$$

where,

$$\begin{aligned}
g_1(u) &= [q_1[e^{q_1} + (2q_1 - 1)e^{-q_1}]u(e^{-q_1 u} - e^{q_1 u}) + [(1+q_1)e^{q_1} \\
&\quad -(1-q_1)e^{-q_1}][e^{q_1 u} - (1+2q_1 u)e^{-q_1 u}]]/A_0 \\
g_2(u) &= [[e^{q_1} - (1+2q_1)e^{-q_1}]u(e^{q_1 u} - e^{-q_1 u}) + 2q_1(e^{q_1} - e^{-q_1}) \\
&\quad ue^{-q_1 u} + (e^{q_1} - e^{-q_1})(e^{-q_1 u} - e^{q_1 u})]/A_0
\end{aligned} \tag{3.12}$$

and,

$$A_0 = [e^{q_1} - (1+2q_1)e^{-q_1}][(1+q_1)e^{q_1} - (1-q_1)e^{-q_1}] \\ -(e^{q_1} - e^{-q_1})[q_1 e^{q_1} - q_1 e^{-q_1} + 2q_1^2 e^{-q_1}] \tag{3.13}$$

With the mathematical equations of blending surfaces obtained in Eqs. 3.11, 3.29, 3.34 and 3.40, it can tackle various surface blending problems with positional and tangential continuities. I will demonstrate this in the following sections.

Since the boundary constraints $C_0(v)$, $\bar{C}_0(v)$, $C_1(v)$ and $\bar{C}_1(v)$ are explicitly incorporated in the mathematical expressions 3.11, 3.29, 3.34 and 3.40 of the blending surfaces, the main task of surface blending is to determine the boundary curves and first partial derivatives at the trimlines which can be readily obtained from the primary surfaces. Therefore, the proposed surface blending method is easy to use.

Once the boundary constraints at the trimlines are known, blending surfaces are analytically determined from one of 3.11, 3.29, 3.34 and 3.40. Even boundary constraints are represented at the discrete points at the trimlines, the proposed method can also construct blending surfaces quickly because of the explicit mathematical expressions of blending surfaces.

3.3 Implementation and Validation

In this section, I implement the proposed method and validate it with various surface blending tasks. The first task is to create a blending surface between an open surface and a closed conic surface. This example is used to validate Eq. 3.11 in surface blending. Denoting $S = S(u,v) = [S_x S_y S_z]^T$, the boundary constraints for this blending task can be written

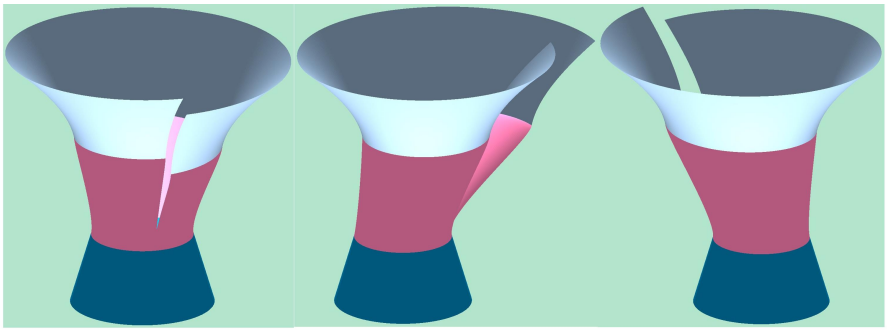


Figure 3.1: *Blending surface constructed with Eq. 3.11*

as,

$$
\begin{aligned}
u = 0 \qquad & S_x = 0.1\sinh(17y + 0.1) + 1.1297\sin v \qquad \partial S_x/\partial u = -0.027\sin v \\
& S_y = 0.3\cosh 0.3v + 1.1297\cos v \qquad \partial S_y/\partial u = -0.027\cos v \\
& S_z = -1.5 + e^0.3 \qquad \partial S_z/\partial u = -0.5e^0.3 \\
u = 1 \qquad & S_x = 0.8\sin v \qquad \partial S_x/\partial u = 0.2\sin v \\
& S_y = 0.8\cos v \qquad \partial S_y/\partial u = 0.2\cos v \\
& S_z = -1.9 \qquad \partial S_z/\partial u = -3.6
\end{aligned}
\tag{3.14}
$$

Substituting the vector-valued functions $C_0(v)$, $\bar{C}_0(v)$, $C_1(v)$ and $\bar{C}_1(v)$ determined by Eq. 3.14 into Eq. 3.11, and setting the shape control parameters to: $\alpha = \gamma = 1$, and $\beta = -2$, the blending surface is obtained and depicted in Fig. 1 where Fig. 3.1(b) and Fig. 3.1(c) are from different views of the blending surface in Fig. 3.1(a).

The second blending task is to generate a blend between a circular torus and an elliptic hyperboloid. This blending task is used to validate Eq. 3.29 in surface blending. The boundary constraints for this blending

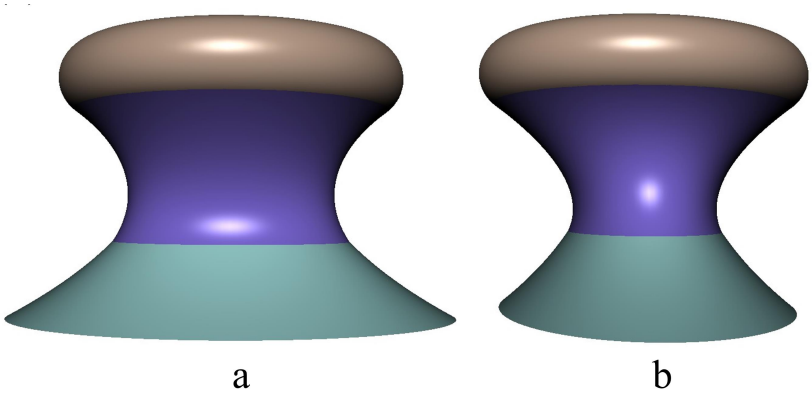


Figure 3.2: *Blending surface constructed with Eq. 3.29*

task are,

$$
\begin{aligned}
u = 0 \qquad & S_x = (1.7 + 0.45\cos 0.85)\cos v \qquad \partial S_x/\partial u = 2.45 \sin 0.85 \cos v \\
& S_y = (1.7 + 0.45\cos 0.85)\sin v \qquad \partial S_y/\partial u = 2.45 \sin 0.85 \sin v \\
& S_z = 0.45 \sin 0.85 \qquad \partial S_z/\partial u = -2.45 \cos 0.85 \\
u = 1 \qquad & S_x = 1.5 \cosh 0 \cos v \qquad \partial S_x/\partial u = (\sinh 0 + \cosh 0)\cos v \\
& S_y = \cosh \sin v \qquad \partial S_y/\partial u = 0.7(\sinh 0 + \cosh 0)\sin v \\
& S_z = -1.5 \sinh 0 \qquad \partial S_z/\partial u = -1.5 \cosh 0
\end{aligned}
\tag{3.15}
$$

For this blending task, I set the shape control parameters to: $\alpha = \gamma = 1$ and $\beta = 2$. The obtained blending surface was depicted in Fig. 3.2 where Fig. 3.2(a) is from the front view and Fig. 3.2(b) is from the side view.

The third example is to construct a blending surface between two conic frustums: one has some wrinkles on its surface and the other has a circular cross section. This example is used to validate Eq. 3.34 in surface blending. The boundary constraints for this blending task have the form of,

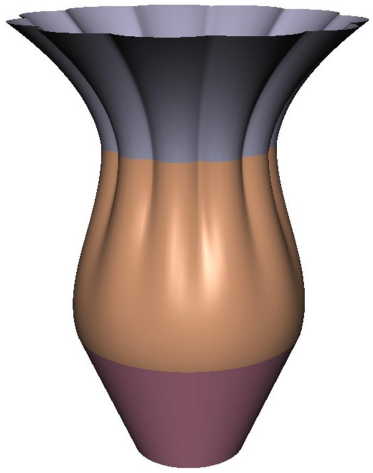


Figure 3.3: *Blending surface constructed with Eq. 3.34*

$$
\begin{aligned}
u = 0 \qquad & S_x = 1.013[0.909\cos v + 0.051\cos(12v)] \\
& \partial S_x/\partial u = 0.26[0.909\cos v + 0.051\cos(12v)] \\
& S_y = 1.013[0.909\sin v + 0.051\sin(12v)] \\
& \partial S_y/\partial u = 0.26[0.909\sin v + 0.051\sin(12v)] \\
& S_z = 0.95 \qquad \partial S_z/\partial u = -2.1 \\
u = 1 \qquad & S_x = 1.3\cos v \qquad \partial S_x/\partial u = -0.5\cos v \\
& S_y = 1.3\sin v \qquad \partial S_y/\partial u = -0.5\sin v \\
& S_z = -1.2 \qquad \partial S_z/\partial u = -2
\end{aligned}
\tag{3.16}
$$

Setting the shape control parameters to: $\alpha = \gamma = 1$ and $\beta = 3$, the obtained blending surface was depicted in Fig. 3.3.

The fourth example is to produce a blending surface between a circular cylinder and an elliptic cylinder of two sheets. This example is used to validate Eq. 3.40 in surface blending. The boundary constraints for this blending task are,

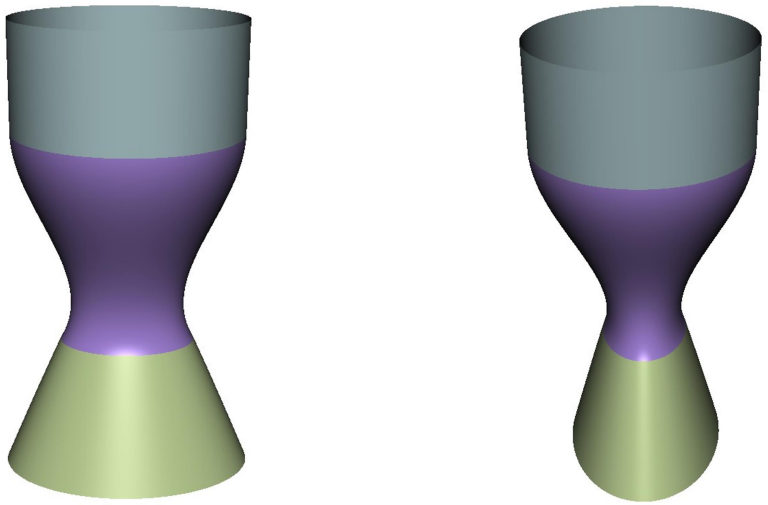


Figure 3.4: *Blending surface constructed with Eq. 3.40*

$$
\begin{array}{lll}
u = 0 & S_x = 0.9\cos v & \partial S_x/\partial u = 0 \\
 & S_y = 0.9\sin v & \partial S_y/\partial u = 0 \\
 & S_z = 0.4 & \partial S_z/\partial u = -1.8 \\
u = 1 & S_x = 0.6\sinh 1.4\cos v & \partial S_x/\partial u = 0.6\cosh 1.4\cos v \\
 & S_y = 0.4\sinh 1.4\sin v & \partial S_y/\partial u = 0.4\cosh 1.4\sin v \\
 & S_z = -1.5\cosh 1.4 & \partial S_z/\partial u = -1.35\sinh 1.4
\end{array}
\tag{3.17}
$$

With Eqs. 3.40 and 3.17, and setting the shape control parameters to: $\alpha = \beta = \gamma = 1$, the generated blending surface is given in Fig. 3.4 where Fig. 3.4(a) is from the front view and Fig. 3.4(b) is from the side view.

3.4 Shape Manipulation of Blending Surfaces

The advantage of the proposed blending approach is that the shape of the blending surfaces is controllable by changing the shape control parameters in 3.2. With this shape manipulation method, the continuities between the primary surfaces and the blending surface are well maintained. I will use an example below to demonstrate this. It is to blend a petal-like surface to the frustum of an elliptic cone. The boundary constraints for this example are,

$$
\begin{array}{ll}
u = 0 & S_x = 1.015[0.9\cos v + 0.1\cos(7v)] \\
 & \partial S_x/\partial u = 0.02[0.9\cos v + 0.1\cos(7v)] \\
 & S_y = 1.015[0.9\sin v + 0.1\sin(7v)] \\
 & \partial S_y/\partial u = 0.02[0.9\sin v + 0.1\sin(7v)] \\
 & S_z = 1.65 \qquad \partial S_z/\partial u = -0.1 \\
u = 1 & S_x = 1.6\cos v \qquad \partial S_x/\partial u = 1.2\cos v \\
 & S_y = 1.2\sin v \qquad \partial S_y/\partial u = 1.2\sin v \\
 & S_z = 0 \qquad \partial S_z/\partial u = -1.8
\end{array}
\tag{3.18}
$$

When setting the shape control parameters to: $\alpha = \gamma = 1$, and $\beta = 0.5$, the obtained blending surface was indicated in Fig. 3.5(a). If I changed these shape control parameters to: $\alpha = 3.7$, $\beta = 2.5$ and $\gamma = 1$, the blending surface given in Fig. 3.5(b) was produced. The blending surface shown in Fig. 3.5(c) was created by taking the shape control parameters to be: $\alpha = 3.7$, $\beta = 2$ and $\gamma = 18$, and that presented in Fig. 3.5(d) was achieved by setting the shape control parameters to: $\alpha = 2$, $\beta = 0.5$, and $\gamma = 1$.

In order to demonstrate the shape changes of the blending surface indicated in Fig. 3.5(a)-(d) more clearly, I depicted the profile curves

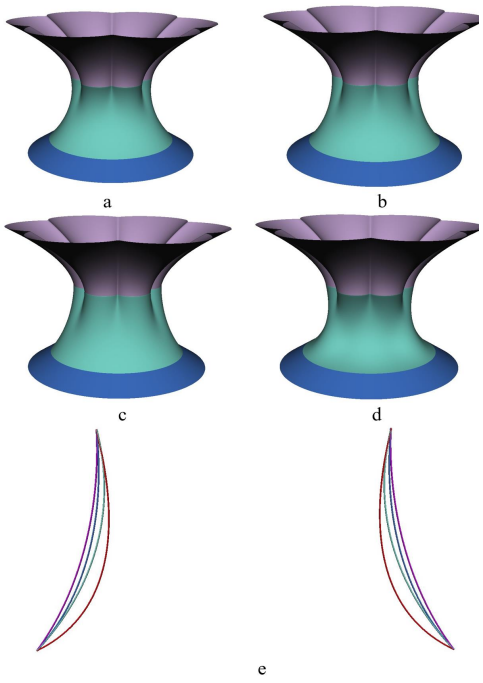


Figure 3.5: *Blending surfaces created with different shape control parameters (a)* $\alpha = \gamma = 1$, *and* $\beta = 0.5$, *(b)*$\alpha = 3.7$, $\beta = 2.5$ *and* $\gamma = 1$, *(c)* $\alpha = 3.7$, $\beta = 2$ *and* $\gamma = 18$, *(d)* $\alpha = 2$, $\beta = 0.5$, *and* $\gamma = 1$, *(e) profile curves of (a)-(d).*

of the blending surface in Fig. 3.5(e) where the profile curves in blue, green, purple and red are from Fig. 3.5(a), Fig. 3.5(b), Fig. 3.5(c) and Fig. 3.5(d), respectively.

The images in Fig. 3.5(a)-(e) indicate that shape control parameters are very effective in controlling the shape of blending surfaces. By manipulating these shape control parameters, the shape of blending surfaces can be controlled effectively while the continuities at the trimlines between the blending surface and primary surfaces remain the same.

Now, the proposed method investigate how the shape control parameters affect the shape of blending surfaces. There are two factors affecting the shape of a blending surface: one is the blending boundary tangents, i. e., the first partial derivatives, and the other is shape control parameters. In order to exclude the influence of the blending boundary tangents on the blending surface, I keep the blending boundary constraints 3.18 unchanged. With this treatment, it can be investigated that the effects

of the shape control parameters on the shape of the blending surface only.

First, fix the second and third shape control parameters $\beta = \gamma = 1$, take the first shape control parameter $\alpha = 1$, and obtain the red profile curve of the blending surface shown in Fig. 3.6(a). Next, change the first shape control parameter to $\alpha = 2$, the green profile curve of the blending surface is generated. Further raising the shape control parameters to $\alpha = 2.5$, the blue profile curve of the blending surface is created. These profile curves indicate that raising the first shape control parameters, the whole blending surface becomes uniformly more concave.

Next, fix the first and third shape control parameters $\alpha = \gamma = 1$. When the second shape control parameter is set to $\beta = 0.1$, the red profile curve indicated in Fig. 3.6(b) is produced. Increasing the parameter to $\beta = 1.5$ and $\beta = 3$, respectively, the green and blue profile curves of the blending surface are obtained. These profile curves demonstrate that increasing the second shape control parameter, the blending surface becomes less concave. When the increase of the parameter is small ($\beta = 1.5$), the influence mainly occurs in the lower part of the blending surface. However, when the parameter is large ($\beta = 3$), the influence becomes more uniform.

Finally, fix the first and second shape control parameters $\alpha = \beta = 1$, and change the third shape control parameter only. When $\gamma = 0.1$, the achieved red profile curve of the blending surface is depicted in Fig. 3.6(c). Changing the parameter to 0.3 and 0.5, respectively, the green and blue profile curves are created and shown in the same figure. These profile curves suggest that increasing the third shape control parameter, the blending surface becomes more concave. When the increase is not big ($\gamma = 0.3$), the influence of the parameter on the top part of the blending surface is more significant. However, when the increase becomes larger ($\gamma = 0.5$), the influence changes to more uniform.

In order to demonstrate the advantage of the proposed approach in effectively manipulating the shape of blending surfaces and exactly satisfying blending boundary constraints, here make a comparison between

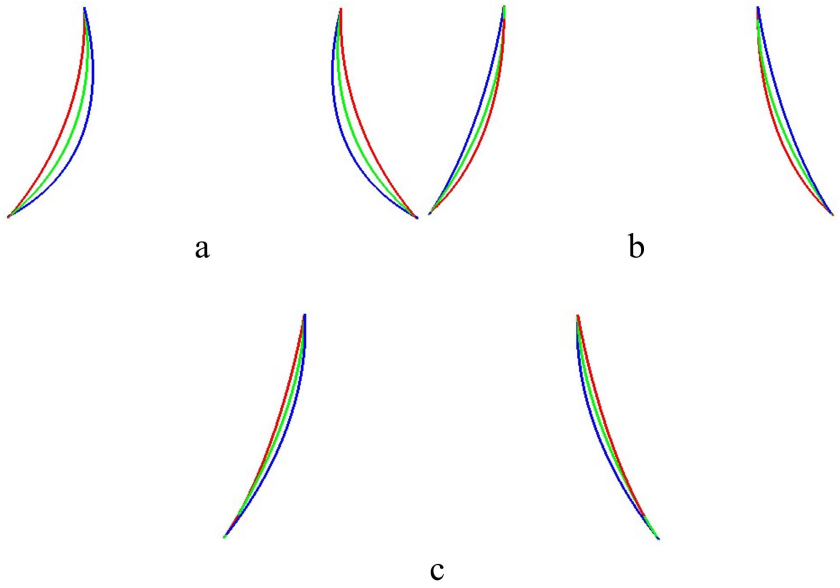


Figure 3.6: *Relationships between shape control parameters and the shape of blending surfaces (a) , (red profile), (green profile), (blue profile), (b) , (red profile), (green profile), (blue profile), (c) , (red profile), (green profile), (blue profile).*

the proposed approach and surface blending using the cubic Hermite interpolation.

For the blending example defined by 3.18, the blending surface using the cubic Hermite interpolation in cubic Bzier form is described by Farin [2014]

$$S(u,v) = H_0^3(u)C_0(v) + H_1^3(u)\bar{C}_0(v) + H_2^3(u)\bar{C}_1(v) + H_3^3(u)C_1(v) \tag{3.19}$$

where

$$
\begin{aligned}
H_0^3(u) &= B_0^3(u) + B_1^3(u) \\
H_1^3(u) &= B_1^3(u)/3 \\
H_2^3(u) &= -B_2^3(u)/3 \\
H_3^3(u) &= B_2^3(u) + B_3^3(u)
\end{aligned}
\tag{3.20}
$$

and

$$
\begin{aligned}
B_i^3(u) &= 3! u^i (1-u)^{3-i} / [i!(3-i)!] \\
&(i = 0, 1, 2, 3)
\end{aligned}
\tag{3.21}
$$

Substituting Eq. 3.18 into Eq. 3.19, to obtain the mathematical equations of the blending surface and use them to produce the blending surface in Fig. 3.7(a) whose profile curve is coloured red and indicated in Fig. 3.7(c). With the proposed approach, the blending surface obtained with the cubic Hermite interpolation Eq. 3.19 can be reproduced by taking the shape control parameters to be $\alpha = \beta = 1$ and $\gamma = 0.1$ in 3.34 whose profile curve is coloured green and shown in Fig. 3.7(c) as well. By changing these parameters to $\alpha = \gamma = 1$ and $\beta = 11$, the blending surface is changed into that whose profile curve is coloured in pink in Fig. 3.7(c). Further changing the parameter β to 17 while keeping the parameters α and γ unchanged, a different shape of the blending surface is obtained and depicted in Fig. 3.7(b) whose profile curve in blue is shown in Fig. 3.7(c).

According to Eq. 3.19, the shape of bending surfaces determined by the cubic Hermite interpolation can be manipulated by scaling the magnitude of the boundary tangent at the trimlines. However, such a manipulation method cannot achieve the same magnitude of boundary tangents of primary surfaces at the trimlines. Consequently, it cannot

exactly satisfy the constraints of boundary tangents in 3.1 required by primary surfaces. In contrast, the proposed approach can obtain different shapes of the blending surface through manipulating shape control parameters without changing boundary tangents, i. e. $\bar{C}_0(v)$ and $\bar{C}_1(v)$ in 3.1 as demonstrated in Fig. 3.7(c) and Fig. 3.6.

The above discussions explain why one of the analytical closed form solutions 3.11, 3.29, 3.34 and 3.40 is required for surface blending. Although they are more complicated than 3.19 obtained with cubic Hermite interpolation, they have the advantage of exactly satisfying blending boundary constraints and effectively manipulating the shape of blending surfaces which cannot be achieved by the surface blending using cubic Hermite interpolation.

The blending surfaces depicted in Figs. 3.7(a) and 3.7(b) are from the same view. Examining the profiles at the upper trimline highlighted by the red circles, it can be found that a smoother transition between the top primary and middle blending surfaces is achieved in Fig. 3.7(b). It indicates that although both the proposed approach and the cubic Hermite interpolation satisfy the blending boundary constraints exactly, the blending surface created with the method achieves better visual smoothness due to the shape change of the blending surface caused by the shape control parameters.

The different shapes depicted in Fig. 3.7(c) were created by 3.34 only. It indicates that each of 3.11, 3.29, 3.34 and 3.40 has a capacity to create different shapes of a blending surface without changing boundary constraints.

Apart from its advantage of effective shape manipulation and exact satisfaction of boundary constraints, the method can also create blending surfaces quickly. On a laptop with a 1.66 GHz CPU and using $100 * 100$ surface points, the cubic Hermite interpolation took 0.031 second to create the blending surface in Fig. 3.7(a). The proposed approach took 0.063 second to generate the blending surface in Fig. 3.7(b). Although the mathematical expressions of the blending method look more complicated than the cubic Hermite interpolation, the time used to generate

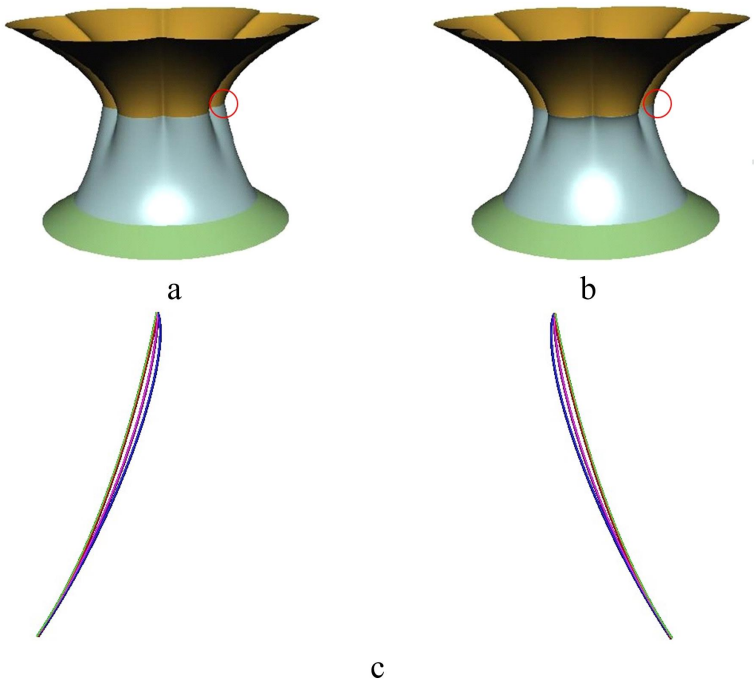


Figure 3.7: *Comparison of blending surfaces between the proposed approach and cubic Hermite interpolation (a) Hermite interpolation (b) ODE-based blending (c) profile curves of blending surfaces: Hermite interpolation (red profile), ODE-based blending (green profile* $\alpha = \beta = 1$, $\gamma = 0.1$*; pink profile* $\alpha = \gamma = 1$, $\beta = 11$*; blue profile* $\alpha = \gamma = 1$, $\beta = 17$*).*

the blending surface with the proposed approach is only about twice of that using the cubic Hermite interpolation, suggesting that the proposed approach can create blending surfaces fast.

In order to facilitate the application of the proposed approach in current commercial CAD systems, in Section 3.7 I demonstrate how the mathematical expressions of the proposed ODE blending surfaces are approximated with cubic splines.

3.5 Blending Between More Than Two Primary Surfaces

The proposed approach is not only powerful in blending two primary surfaces together, but is also effective in constructing blending surfaces between more than two primary surfaces, which is usually a difficult task

for some existing surface blending approaches such as partial differential equation-based surface blending. I will demonstrate this with an example below.

In this example, I will blend three planes together, as shown in fig. 3.8(a). First, construct blending surfaces between the top and front planes, between the top and side planes, and between the front and side planes. Then, generate a blending surface between the three constructed blending surfaces.

The boundary constraints for the blending between the top and front planes can be written in the form below,

$$\begin{aligned}
u = 0 \qquad & S_x(u,v) = l_x v \qquad \partial S_x(u,v)/\partial u = 0 \\
& S_y(u,v) = y_0 + r \qquad \partial S_y(u,v)/\partial u = -0.5 l_y \\
& S_z(u,v) = z_0 + l_z + r \qquad \partial S_z(u,v)/\partial u = 0 \\
u = 1 \qquad & S_x(u,v) = l_x v \qquad \partial S_x(u,v)/\partial u = 0 \\
& S_y(u,v) = y_0 \qquad \partial S_y(u,v)/\partial u = 0 \\
& S_z(u,v) = z_0 + l_z \qquad \partial S_z(u,v)/\partial u = -0.5 l_z
\end{aligned} \tag{3.22}$$

The boundary constraints used to construct the blending surface between the top and side planes can be written as,

$$\begin{aligned}
u = 0 \qquad & S_x(u,v) = l_x v \qquad \partial S_x(u,v)/\partial u = 0.5 l_x \\
& S_y(u,v) = y_0 + r + l_y y \qquad \partial S_y(u,v)/\partial u = 0.0 \\
& S_z(u,v) = z_0 + l_z + r \qquad \partial S_z(u,v)/\partial u = 0 \\
u = 1 \qquad & S_x(u,v) = l_x + r \qquad \partial S_x(u,v)/\partial u = 0 \\
& S_y(u,v) = y_0 + r + l_y v \qquad \partial S_y(u,v)/\partial u = 0 \\
& S_z(u,v) = z_0 + l_z \qquad \partial S_z(u,v)/\partial u = -0.5 l_z
\end{aligned} \tag{3.23}$$

Similarly, the boundary constraints employed to construct the blending surface between the front and side planes take the form of,

$$\begin{array}{lll} u=0 & S_x(u,v)=l_x & \partial S_x(u,v)/\partial u = 0.5l_x \\ & S_y(u,v)=y_0 & \partial S_y(u,v)/\partial u = 0.0 \\ & S_z(u,v)=z_0+l_z v & \partial S_z(u,v)/\partial u = 0 \\ u=1 & S_x(u,v)=l_x+r & \partial S_x(u,v)/\partial u = 0 \\ & S_y(u,v)=y_0+r & \partial S_y(u,v)/\partial u = 0.5l_y \\ & S_z(u,v)=z_0+l_z v & \partial S_z(u,v)/\partial u = 0 \end{array} \tag{3.24}$$

In the above equation, y_0, z_0, l_x, l_y, l_z and r are geometric parameters used to define the top, front and side planes.

Taking the geometric parameters in the above equations to be: $y_0 = l_x = l_z = 1$, $z_0 = 0.5$, $l_y = 1.2$ and $r = 0.25$, and setting the shape control parameters to: $\alpha = \gamma = 1$ and $\beta = 2$, I constructed the blending surfaces between these planes using 3.29 and depicted these blending surfaces in Fig. 3.8(b).

Finally, generate the blending surface between the three constructed blending surfaces shown in Fig. 3.8(b). This blending surface is a 3-sided patch. The boundary constraints for this 3-sided patch were determined from 3.29 of the constructed blending surfaces.

The top boundary at $u = 0$ for this blending problem is a point which is determined by setting $u = 0$ and $v = 1$ in 3.29 of the blending surface between the top and front planes, i. e., $S^{TF}(0, 1)$ or by setting $u = 0$ and $v = 0$ in 3.29 of the blending surface between the top and side planes, i. e., $S^{TS}(0, 0)$ where the superscripts TF and TS indicate the blend surface between the top and front planes and between the top and side planes, respectively. Although the top boundary of the 3-sided patch is a point, the boundary tangent changes continuously from that determined by $S^{TF}(u, v)$ to the one determined by $S^{TS}(u, v)$, i. e., from

$T^0 = \partial S^{TF}(0,1)/\partial u$ to $T^1 = \partial S^{TS}(0,0)/\partial u$. The boundary tangent $T^t(v)$ at any position v can be obtained by interpolating T^0 and T^1 which is $T^t(v) = T^0 + (T^1 - T^0)$ where the superscript t indicates the top boundary and $l(v)$ is the normalized arc length of the bottom boundary curve.

The normalized arc length of the bottom boundary curve can be determined as follows. First, calculate the total length of the bottom boundary curve with the equation
$L = \int_0^1 \sqrt{[dS_x^{FS}(u,1)/du]^2 + [dS_y^{FS}(u,1)/du]^2 + [dS_z^{FS}(u,1)/du]^2}du$ where $S^{FS}(u,1)$ is the vector-valued equation of the bottom boundary curve and the superscript FS indicates the blending surface between the front and side planes. Then, calculate the length of the bottom boundary curve from $u = 0$ to any position u which is
$L(u) = \int_0^u \sqrt{[dS_x^{FS}(u,1)/du]^2 + [dS_y^{FS}(u,1)/du]^2 + [dS_z^{FS}(u,1)/du]^2}du$.
The normalized arc length $l(v)$ of the bottom boundary curve is determined by $l(v) = L(v)/L$ where the parametric variable u in $L(u)$ has been replaced by the parametric variable v.

The bottom boundary is a curve represented by $C^b(u) = S^{FS}(u,l)$ where the superscript b indicates the bottom boundary of the 3-sided blending surface. The boundary tangent at the bottom boundary is determined by $T^b(u) = \partial S^{FS}(u,1)/\partial v$.

When constructing the 3-sided blending surface, the parametric direction u for the bottom boundary is changed into the parametric direction v. Therefore, the boundary curve and boundary tangent for the bottom boundary become $C^b(v)$ and $T^b(v)$.

With the above treatment, the boundary constraints for the 3-sided blending surface are represented by the following equation

$$\begin{array}{lll} u = 0 & S(u,v) = S^{TF}(0,1) & \partial S(u,v)/\partial u = T^t(v) \\ u = 1 & S(u,v) = C^b(v) & \partial S(u,v)/\partial u = T^b(v) \end{array} \tag{3.25}$$

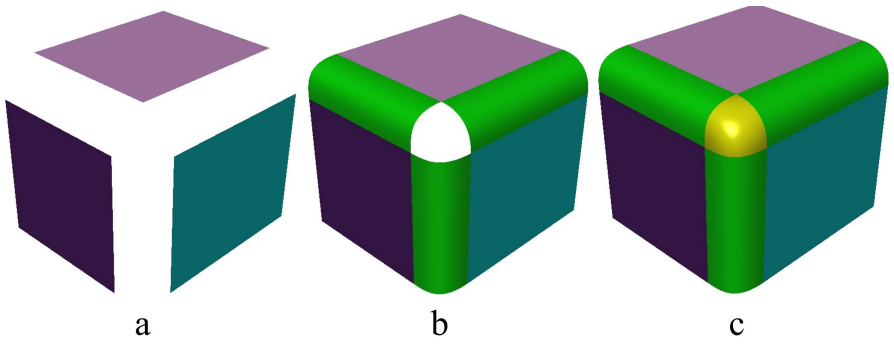


Figure 3.8: *Blending between more than 2 primary surfaces*

Still using the same shape control parameters, obtained the 3-sided blending surface and depicted it in Fig. 3.8(c).

This example demonstrates that the proposed approach can blend more than two primary surfaces together easily and effectively. In contrast, it is not easy to construct such blending surfaces by using some existing surface blending methods such as partial differential equation-based surface blending approaches.

3.6 Other Solutions of EQ. 3.2

For $\beta^2 = 4\alpha\gamma$ and $\alpha/\beta > 0$, the roots of Eq. 3.4 are,

$$r_{1,2,3,4} = \pm i q_2 \tag{3.26}$$

where i is an imaginary unit and,

$$q_2 = \sqrt{\beta/(2\alpha)} \tag{3.27}$$

With the roots given in Eq. 3.26, the solution of Eq. 3.27 becomes,

$$G(u) = d_1 \cos q_2 u + d_2 \sin q_2 u + d_3 u \cos q_2 u + d_4 u \sin q_2 u \tag{3.28}$$

where d_1, d_2, d_3 and d_4 are vector-valued unknown constants.

In order to determine the unknown constants in 3.28, perform the same sweeping operation by substituting it into 3.1, and solving for the four unknown constants d_1, d_2, d_3 and d_4. Then, substitute the unknown constants back into 3.28, and obtain,

$$\begin{aligned} S(u,v) = (&\cos q_2 u + A_2 \sin q_2 u - q_2 A_2 u \cos q_2 u - u \cot q_2 \sin q_2 u + \\ &A_2 A_5 u \sin q_2 u) C_0(v) + (A_3 \sin q_2 u + u \cos q_2 u - q_2 A_3 u \cos q_2 u \\ &- u \cot q_2 \sin q_2 u + A_3 A_5 u \sin q_2 u) \bar{C}_0(v) + (-A_4 \sin q_2 u \\ &+ q_2 A_4 u \cos q_2 u + u \sin q_2 u / \sin q_2 - A_4 A_5 u \sin q_2 u) C_1(v) \\ &+ (\sin q_2 u - q_2 u \cos q_2 u + A_5 u \sin q_2 u) \bar{C}_1(v) / A_1 \end{aligned} \tag{3.29}$$

where,

$$
\begin{aligned}
A_1 &= q_2/\sin q_2 - \sin q_2 \\
A_2 &= [q_2 \sin q_2 + \cot q_2(\sin q_2 + q_2 \cos q_2)]/A_1 \\
A_3 &= [-(\cos q_2 - q_2 \sin q_2) + \cot q_2(\sin q_2 + q_2 \cos q_2)]/A_1 \\
A_4 &= (\sin q_2 + q_2 \cos q_2)/(A_1 \sin q_2) \\
A_5 &= (q_2 \cos q_2 - \sin q_2)/\sin q_2
\end{aligned}
\tag{3.30}
$$

For $\beta^2 > 4\alpha\gamma$, the roots of 3.4 are found to be,

$$
\begin{aligned}
r_{1,2} &= \pm i q_3 \\
r_{3,4} &= \pm i q_4
\end{aligned}
\tag{3.31}
$$

where i is an imaginary unit and,

$$
\begin{aligned}
q_3 &= \sqrt{\beta(1 - \sqrt{1 - 4\alpha\gamma/\beta^2})/(2\alpha)} \\
q_4 &= \sqrt{\beta(1 + \sqrt{1 - 4\alpha\gamma/\beta^2})/(2\alpha)}
\end{aligned}
\tag{3.32}
$$

With the roots given in 3.32, the solution of 3.2 takes the form of,

$$
G(u) = d_1 \cos q_3 u + d_2 \sin q_3 u + d_3 \cos q_4 u + d_4 \sin q_4 u
\tag{3.33}
$$

where d_1, d_2, d_3 and d_4 are vector-valued unknown constants.

Same as above, substitute 3.33 into 3.1, carry out the sweeping operation, and determine the four unknown constants d_1, d_2, d_3 and d_4. After

substituting these unknown constants back to 3.33, the mathematical equation of blending surfaces becomes,

$$\begin{gathered} S(u,v) = [-g_3(u)\cos q_4/A_6 - g_4(u)q_4 \sin q_4/A_6 + \cos q_4 u]C_0(v) \\ [-g_3(u)\sin q_4/q_4/A_6 + g_4(u)\cos q_4/A_6 + \sin q_4 u/q_4] \\ \bar{C}_0(v) + g_3(u)C_1(v)/A_6 - g_4(u)\bar{C}_1(v)/A_6 \end{gathered} \tag{3.34}$$

where,

$$A_6 = q_3(\cos q_3 - \cos q_4)^2 - (\sin q_3 - q_3 \sin q_4/q_4)(q_4 \sin q_4 - q_3 \sin q_3) \tag{3.35}$$

and,

$$\begin{gathered} g_3(u) = q_3(\cos q_3 - \cos q_4)(\cos q_3 u - \cos q_4 u) + \\ (q_4 \sin q_4 - q_3 \sin q_3)(q_3 \sin q_4 u/q_4 - \sin q_3 u) \\ g_4(u) = (\cos q_3 - \cos q_4)(q_3 \sin q_4 u/q_4 - \sin q_3 u) + \\ (\sin q_3 - q_3 \sin q_4/q_4)(\cos q_3 u - \cos q_4 u) \end{gathered} \tag{3.36}$$

For $\beta^2 < 4\alpha\gamma$, the roots of 3.4 are found to be,

$$r_{1,2,3,4} = \pm q_5 \pm i q_6 \tag{3.37}$$

where i is an imaginary unit and,

$$q_5 = \sqrt[4]{\gamma/\alpha}\sqrt{0.5 - \beta/(4\sqrt{\alpha\gamma})}$$
$$q_6 = \sqrt[4]{\gamma/\alpha}\sqrt{0.5 + \beta/(4\sqrt{\alpha\gamma})} \tag{3.38}$$

With the roots given in 3.38, the solution of 3.2 takes the form of,

$$G(u) = d_1 e^{q_5 u}\cos q_6 u + d_2 e^{q_5 u}\sin q_6 u + d_3 e^{-q_5 u}\cos q_6 u + d_4 e^{-q_5 u}\sin q_6 u \tag{3.39}$$

where d_1, d_2, d_3 and d_4 are vector-valued unknown constants.

Substituting 3.39 into 3.1 and doing the sweeping operation, the 4 unknown constants are determined and blending surfaces satisfying 3.1 and 3.2 are found to be,

$$\begin{aligned} S(u,v) = [&-A_9 g_5(u)/q_6 - (A_7 q_5/q_6 - A_8)g_6(u) + e^{-q_5 u}\cos q_6 u \\ &+ q_5 e^{-q_5 u}\sin q_6 u/q_6]C_0(v) + [-A_7 g_6(u)/q_6 - e^{-q_5} \\ &\sin q_6 g_5(u)/q_6 + e^{-q_5 u}\sin q_6 u/q_6]\bar{C}_0(v) \\ &+ g_5(u)C_1(v) + g_6(u)\bar{C}_1(v) \end{aligned} \tag{3.40}$$

where,

$$
\begin{gathered}
A_7 = q_6 e^{-q_5}\cos q_6 - q_5 e^{-q_5}\sin q_6 \\
A_8 = q_5 e^{-q_5}\cos q_6 + q_6 e^{-q_5}\sin q_6 \\
A_9 = q_5 e^{-q_5}\sin q_6 + q_6 e^{-q_5}\cos q_6 \\
g_5(u) = 1/A_{14} \\
\{A_{12}[(e^{-q_5 u} - e^{q_5 u})\sin q_6 u] + A_{13}[(e^{q_5 u} - e^{-q_5 u})\cos q_6 u - 2q_5 e^{-q_5 u}\sin q_6 u/q_6]\} \\
g_6(u) = 1/A_{14} \\
\{A_{10}[(e^{q_5 u} - e^{-q_5 u})\sin q_6 u] + A_{11}[(e^{-q_5 u} - e^{q_5 u})\cos q_6 u + 2q_5 e^{-q_5 u}\sin q_6 u/q_6]\}
\end{gathered}
\tag{3.41}
$$

and,

$$
\begin{aligned}
A_{10} &= (e^{q_5} - e^{-q_5})\cos q_6 - 2q_5 e^{-q_5}\sin q_6/q_6 \\
A_{11} &= (e^{q_5} - e^{-q_5})\sin q_6 \\
A_{12} &= (q_5\cos q_6 - q_6\sin q_6)e^{q_5} + A_8 - 2q_5 A_7/q_6 \\
A_{13} &= (q_5\sin q_6 + q_6\cos q_6)e^{q_5} - A_9 \\
A_{14} &= A_{10}A_{13} - A_{11}A_{12}
\end{aligned}
\tag{3.42}
$$

3.7 Approximation of One Blending Surfaces for CAD-system Processing

Compared to algebraic functions, some transcendental functions can describe the curves or surfaces with special and useful shapes. Therefore, they have been widely applied in engineering. Involute gears are such an example which are commonly used for motion and power transmission, whose tooth profiles are involutes. Similarly epicycloids and hypocycloids are used in cycloidal gears. The proposed approach also demonstrates that without changing boundary tangents, the surface blending involving transcendental functions proposed in this chapter can create different

shapes of a blending surface.

Transcendental functions cannot be directly incorporated into current commercial CAD systems. However, they can be approximated by the functions accepted by the systems as investigated by Ling et al. Ling et al. [2010] and Higuchi et al. Higuchi et al. [2007]. Here, I will approximate the mathematical expressions of the blending surfaces proposed in this chapter with the cubic spline interpolation method. It can be seen that this treatment is simple and computationally efficient, and has high approximation accuracy.

Observing the mathematical expressions Eq. 3.11, 3.17, 3.22 of the blending surfaces created with the proposed approach, all of them can be written as

$$S(u,v) = r_1(u)C_0(v) + r_2(u)\bar{C}_0(v) + r_3(u)\bar{C}_1(v) + r_4(u)\bar{C}_1(v) \tag{3.43}$$

where $C_0(v)$, $\bar{C}_0(v)$, $C_1(v)$ and $\bar{C}_1(v)$ are boundary curves and boundary tangents defined in Eq. 3.1.

If the functions $r_1(u)$, $r_2(u)$, $r_3(u)$ and $r_4(u)$ can be approximated with one of those accepted by the current CAD systems, surface approximation of the proposed approach can be transformed into curve approximation.

There are many methods for curve approximation. Here use the clamped cubic spline interpolation which has good approximation accuracy, and more importantly, the first derivative at the two boundaries of the domain where a function is defined can be exactly satisfied. There are a lot of resources on the internet including computer source code for the clamped cubic spline interpolation method. Here, take $r_1(u)$ as an example, and briefly introduce how to determine the cubic spline approximating $r_1(u)$.

In order to use one subscript in the following discussions, drop the subscript 1 of $r_1(u)$ and change it into $r(u)$. If the values of $r(u)$ at $u_0 = 0$, $u_1 = 1/n$, $u_2 = 2/n$, Λ, $u_{n-1} = (n-1)/n$, and $u_n = 1$ are

known to be r_0, r_1, r_2, λ, $r_n - 1$, and r_n where n is the total number of the intervals on $[0, 1]$, can determine a cubic spline $f(u)$ consisting of n cubic functions $f_j(u) = a_j + b_j u + c_j u^2 + d_j u^3 (j = 1, 2, \lambda, n-1, n)$ which approximates $r(u)$ by satisfying the following conditions:

(1). $f(u)$ is a cubic function $f_j(u) = a_j + b_j u + c_j u^2 + d_j u^3$
on $[u_{j-1}, u_j](j = 1, 2, \lambda, n-1, n)$,

(2). $f_j(u_{j-1}) = r_{j-1}$ and $f_j(u_j) = r_j (j = 1, 2, \lambda, n-1, n)$,

(3). $f_j'(u_j) = f_{j+1}'(u_j)(j = 1, 2, \lambda, n-1)$,

(4). $f_j"(u_j) = f_{j+1}"(u_j)(j = 1, 2, \lambda, n-1)$,

(5). $f_1'(u_0) = r'(u_0)$ and $f_n'(u_n) = r'(u_n)$.

In the above equations, $f_j'(u_j) = df_j(u_j)/du (j = 1, 2, \lambda, n-1, n)$ and $f_j"(u_j) = df_j^2(u_j)/du^2 (j = 1, 2, \lambda, n-1), f_1^(u_0) = df_1(u_0)/du$, and $r'(u_k) = d_r(u_k)/du (k = 0, n)$.

The convergence, accuracy and efficiency of the clamped cubic spline interpolation are investigated by using a cubic spline with n (n=2, 4 and 6) cubic functions to approximate each of the functions $r_1(u)$, $r_2(u)$, $r_3(u)$ and $r_4(u)$ involved in Eq. 3.22. The values of these functions at $u = i/n (n = 2, 4, 6; i = 0, 1, \lambda, n)$ and their first derivatives with respect to the parametric variable u at the boundaries $u = 0$ and $u = 1$ are used to determine the n cubic functions.

Under the blending boundary constraints, the blending surface created using the combination of the original functions $r_1(u)$, $r_2(u)$, $r_3(u)$ and $r_4(u)$ with the boundary functions $C_0(v)$, $\bar{C}_0(v)$, $C_1(v)$ and $\bar{C}_1(v)$ through Eq. 3.43 is the left one in Fig. 3.9 and that obtained using the combination of 2 cubic functions with the boundary functions is the middle one in the same figure. Examining the shape in both images, I cannot find any visual differences. I have also depicted the profile curve

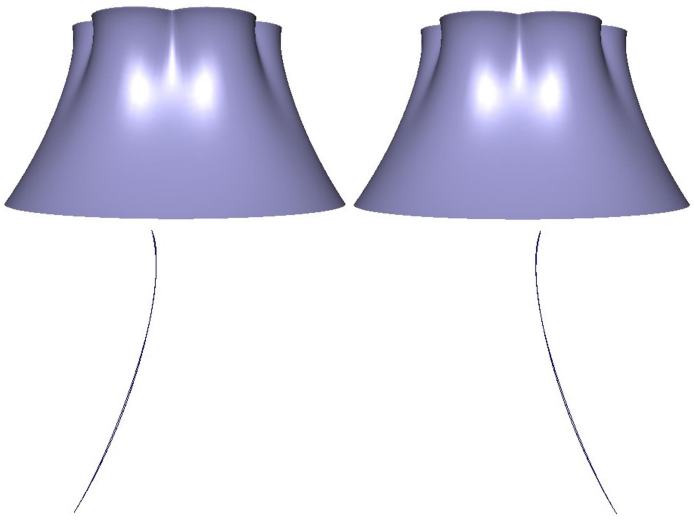


Figure 3.9: *Comparison between the original blending surface and that from the cubic spline interpolation*

of the blending surface created with the original functions $r_1(u)$, $r_2(u)$, $r_3(u)$ and $r_4(u)$ and those approximated with 2, 4 and 6 cubic functions in the right one of Fig. 3.9. Once again, all the profile curves look the same and no visual differences can be found.

In order to quantify the difference between the original and approximation blending surfaces from the cubic spline interpolation and between their profiles, the following error formula is used

$$E = \sum_{i=1}^{I} \sqrt{[(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2 + (z_i - \bar{z}_i)^2]/[x_i^2 + y_i^2 + z_i^2]}/I \tag{3.44}$$

where x_i, y_i and z_i without an overbar are the coordinate values of the I uniformly distributed points on the original blending surface or its profile curve, and those with an overbar are on the approximation blending surface or its profile curve created with the cubic spline interpolation, and I is the number of all points. For the blending surface, $J = \sqrt{I}$ points are uniformly collocated in both parametric directions u and v.

Table 3.1: *Computational error and time of various methods.*

	TA	CSI(n=2)	CSI(n=4)	CSI(n=6)
EPC	0	2.907e-3	1.854e-4	3.006e-5
EBS	0	3.7507e-3	2.300e-4	3.749e-5
T(second)	6.3e-2	4.7e-2	6.2e-2	7.8e-2

For the calculation of the errors between the original blending surface and the approximation one, $J = 100$ points are uniformly collocated in both parametric directions u and v of the blending surface with the total points $I = 10000$. For the determination of the error of the profile curve between both blending surfaces, $I = 100$ points are uniformly collocated in the parametric direction u.

The error between the original blending surface and the approximation blending surface together with the error between their profile curves are listed in Table 3.1 where TA stands for the proposed approach, CSI means the cubic spline interpolation with the number of the used cubic functions being given in the round parenthesis, EPC indicates the error for profile curves, EBS denotes the error for blending surfaces, and T is the computational time. The errors shown in Table 3.1 are very small compared to the maximum radius of the blending surface which is 1.937. Examining these errors, it can be concluded that the cubic spline interpolation converges very quickly, and very few cubic functions (2 cubic functions in this example) can also generate good approximation results.

In addition to its good accuracy, the cubic spline interpolation also has high computational efficiency. On the same laptop and including the time used to generate the values of the functions $r_1(u)$, $r_2(u)$, $r_3(u)$ and $r_4(u)$ at $u = i/n(n = 2, 4, 6; i = 0, 1, \lambda, n)$ and their first derivatives at the boundaries $u = 0$ and $u = 1$ with the proposed approach, the total time for the cubic spline interpolation with n (n=2, 4 and 6) cubic functions to create the approximation blending surfaces is also given in Table 3.1. These timing data indicate the cubic spline interpolation can produce good approximation of blending surfaces efficiently.

3.8 Discussion and Conclusion

This chapter has developed a new surface blending method with positional and tangential continuities based on sweeping surfaces controlled by fourth order ODEs. The generator used to construct blending surfaces is created with the closed form solution to fourth order ODEs and its shape is manipulated by the shape control parameters involved in the equations. Blending surfaces are generated by moving the generator along two three-dimensional trajectories and making it exactly satisfy the tangential constraints at the trimlines. A number of examples were presented to demonstrate the applications of the proposed approach in surface blending.

Due to the analytical nature of the proposed approach, it can generate various blending surfaces quickly. Since blending boundary constraints are explicitly included in the mathematical expressions of the blending surfaces, the proposed approach is easy to use. By making blending surfaces exactly meet the constraints of boundary curves and first partial derivatives at the trimlines, surface blending with positional and tangential continuities is achieved. The shape of the blending surfaces constructed with the proposed approach is controlled effectively by manipulating shape control parameters while still maintaining the same continuities at the trimlines. Apart from its capacity in creating a blending surface between two primary surfaces, it is also effective in blending more than two primary surfaces together.

Chapter 4

Interactive Creation of C2 Continuous ODE Skin Surfaces

In this chapter, a new technique of interactive creation of C2 continuous surfaces is introduced. With this approach, the creation of a three-dimensional surface is transformed into generating 2 boundary curves or 2 boundary curves plus 4 control curves and solving a vector-valued sixth order ordinary differential equation subjected to boundary constraints consisting of boundary curves, first and second partial derivatives at the boundary curves. Unlike the existing patch modeling approaches which require tedious and time-consuming manual operations to stitch two separate patches together and achieve the continuity between two stitched patches, the proposed technique maintains the C2 continuity between adjacent surface patches naturally which avoids manual stitching operations. The proposed technique not only creates those by Maya loft operation but also much more surface shapes since the shape of the surfaces created with this proposed technique can be manipulated easily by simply changing first and second partial derivatives, and shape control parameters.

4.1 Introduction

Surfaces creation approaches are widely applied in creative, digital and design industries to create external appearances of various objects. Depending on different mathematical representations, surfaces can be divided into implicit, explicit and parametric ones. Among them, parametric surfaces are the most common.

Current popular surface creation techniques are polygon, patch modeling such as NURBS, and subdivision. The polygon technique creates complicated models from simple geometric primitives through manipulating surface points. The patch technique divides a complex model into a lot of patches, creates all these patches separately, and stitches them together to produce the model. The subdivision technique uses approximating or interpolating schemes to subdivide the polygonal faces of a coarse polygonal model into smaller faces and generates a denser polygon mesh of the model. Among them, the patch technique is especially suitable for smooth curved surfaces with a small data size.

In spite of the suitability of the patch technique in creating smooth curved surfaces, further improvements are required in the following aspects. First, when using this technique to create a three dimension (3D) objects, tedious and time-consuming manual operations are required to stitch adjacent surface patches together and achieve the smoothness between adjacent patches. Second, when the control points of a surface patch are plenty to keep the details, the global shape of the patch is difficult to manipulate. Thirdly, the traditional patch modeling technique is purely geometric which does not consider any underlying physics of object deformations. Introducing physics into the patch technique usually involves computationally expensive numerical calculations leading to slow surface modeling.

Ordinary differential equation (ODE)-based surface creation generates a surface from the solution to a vector-valued Ordinary differential equation subjected to boundary constraints. This technique is simple and efficiency in manipulating the global shape of a surface patch. However,

how to achieve the solution is not an easy work. For complicated surface creation, it also depends on heavy numerical calculations. In addition, it is very difficult to analytically achieve tangent or curvature continuity in two different parametric directions of a 4-sided surface patch at the same time.

In the work of this chapter, I develop a new surface creation technique which transforms creation of curvature continuous surfaces into the task to find the closed form solution of a vector-valued sixth order ordinary differential equation subjected to the constraints of boundary curves, and first and second partial derivatives at the boundary curves. With this technique, only two boundary curves or two boundary curves and four control curves are required. Different surface shapes can be created easily and efficiently through the shape control parameters.

Compared to the existing patch modeling technique, the proposed technique achieves up to curvature continuities naturally. No manual operations are required to stitch adjacent patches together and deal with the continuity problem between them. The global shape of a surface patch can be manipulated easily and efficiently through simply changing one of the first and second partial derivatives at boundary curves and shape control parameters. Even a surface patch must exactly satisfy the constraints of position functions, and first and second partial derivatives at boundary curves, the shape of surface patches can also be manipulated by changing one of shape control parameters. Since the curves used to define a surface are controlled by differential equations usually used to characterize natural physical process, the proposed technique can be regarded as physics-based. Compared with ODE based surface creation, the proposed technique is simpler and more efficient due to its analytical nature.

4.2 Mathematical Model

Any 3D parametric surfaces always can be described by the vector-valued mathematical equation $X = S(u, v)$ where u and v are two parametric

variables usually defined in the region: $0 \leqslant u \leqslant 1$ and $0 \leqslant v \leqslant 1$, X is a vector-valued position function which has three components x, y and z, and $S(u,v)$ also has three components $S_x(u,v)$, $S_y(u,v)$ and $S_z(u,v)$. When the parametric variable v takes the constant v_i, the surface at the position becomes a parametric curve $C_i = S(u,v_i)$ where the vector-valued function C_i has three components C_{xi}, C_{yi} and C_{zi}. Therefore, the surface $X = S(u,v)$ can be regarded as the set of the parametric curves C_0, C_1, C_2, ..., which are obtained by changing v_i from 0 to 1 continuously. Based on this consideration, the modelling of 3D parametric surfaces can be transformed into that of a set of parametric curves.

In order to achieve the C2 continuity between two adjacent surface patches, the two surface patches should share the same position function, and first and second partial derivatives with respect to the parametric variable u at their joint interface. That is to say, a C2 continuous surface patch in the parametric direction u should exactly satisfy the boundary constraints at $u = 0$ and $u = 1$ which consist of position functions, and the first and second partial derivatives determined by the adjacent surface patches at the positions. If the position functions, and the first and second partial derivatives at the positions $u = 0$ and $u = 1$ are $c_j(v)(j = 1,2,3,4,5,6)$, the boundary constraints which a C2 continuous surface patch should satisfy can be written as

$$\begin{array}{llll} u=0 & S(0,v)=c_1(v), & \partial S(0,v)/\partial u = c_2(v), & \partial S^2(0,v)/\partial u^2 = c_3(v) \\ u=1 & S(1,v)=c_4(v), & \partial S(1,v)/\partial u = c_5(v), & \partial S^2(1,v)/\partial u^2 = c_6(v) \end{array} \tag{4.1}$$

where $c_1(v)$ and $c_4(v)$ are position functions, $c_2(v)$ and $c_5(v)$ are the first partial derivatives, and $c_3(v)$ and $c_6(v)$ are the second partial derivatives at the boundaries.

With the above treatment, the task of surface modeling is to determine the mathematical equation of the set of the parametric curves which

meets Eq. 4.1 at their two ends.

A parametric curve can be described by some popular curve functions such as NURBS. However, these curve functions are purely geometric and did not involve any underlying physics. It has been realized that the physics can be introduced into geometric modeling to improve the realism of geometric modeling and many research studies have addressed physics-based geometric modeling and deformations. Ordinary differential equations (ODEs) usually can be used to describe the deformations of curve-like objects such as beams and members. It is known that the accurate solution of second order ODEs has two unknown constants only which are used to satisfy the constraints of two position functions, and that of fourth order ODEs has four unknown constants which can be used to meet the constraints of two position functions and two first partial derivatives at boundaries. In contrast, the solution of sixth order ODEs contains six unknown constants which can be used to satisfy all the position functions, and the first and second partial derivatives given in boundary constraints Eq. 4.1. Therefore, the following vector-valued sixth order ODE is introduced

$$\rho d^6 S(u, v_i)/du^6 + \eta d^4 S(u, v_i)/du^4 + \lambda d^2 S(u, v_i)/du^2 = F(u) \qquad (4.2)$$

where ρ, η and λ are shape control parameters, $S(u, v_i)$ represents a 3D parametric curve C_i whose components are $S_x(u, v_i)$, $S_y(u, v_i)$ and $S_z(u, v_i)$, and $F(u)$ is the virtual sculpting force function whose components are $F_x(u)$, $F_y(u)$ and $F_z(u)$.

The set of parametric curves can be obtained by finding the solution to Eq. 4.2 subjected to boundary constraints Eq. 4.1. The solution consists of two parts: complementary solution of the associated homogeneous equation of ODE Eq. 4.2 and particular solution. In this work, the interactive surface creation using the complementary solution is investigated. The surface manipulation using the particular solution will be addressed in the following work.

Before investigating the complementary solution to ODE 4.2, first discuss how to generate the boundary constraints 4.1. Here two different approaches are proposed.

The first approach is to draw two boundary curves $c_1(v)$ and $c_4(v)$ as shown in Figure 4.2(a) and Figure 4.4(a). Then boundary tangents and boundary curvature are taken to be the same forms as the boundary curves modified by different coefficients, i.e., $c_2(v) = w_1 c_1(v)$, $c_3(v) = w_2 c_1(v)$, $c_5(v) = w_3 c_4(v)$, and $c_6(v) = w_4 c_4(v)$ where $w_k (k = 1, 2, 3, 4)$ are different coefficients. Some surfaces generated with this approach were depicted in Figure 4.2(a) and Figure 4.4(a).

As shown in Figure 4.3 and Figure 4.5, the second approach is to create two boundary curves $c_1(v)$ and $c_4(v)$ at $u = 0$ and $u = 1$, respectively, and generate the other four control curves $\bar{c}_2(v)$ at $u_2 (0 < u_2 < u_3)$, $\bar{c}_3(v)$ at $u_3 (u_2 < u_3 < u_4)$, $\bar{c}_5(v)$ at $u_5 (\bar{u}_4 < u_5 < u_6)$ where $\bar{u}_4$ is different from u_4, and $\bar{c}_6(v)$ at $u_6 (u_5 < u_6 < 1)$ by duplicating the boundary curves and deforming the duplicated curves through some of geometric transformations: translation, scaling and rotation.

For an arbitrary point on the boundary curve $c_1(v)$ at the position v_i, firstly use the following forward difference formula to calculate the first derivative at the points $c_1(v_i)$ and $\bar{c}_2(v_i)$, i.e.,

$$\begin{aligned} c_2(v_i) &= [\bar{c}_2(v_i) - c_1(v_i)]/u_2 \\ \bar{c}_2'(v_i) &= [\bar{c}_3(v_i) - \bar{c}_2(v_i)]/(u_3 - u_2) \end{aligned} \tag{4.3}$$

Then the same forward difference formula is used to calculate the second derivative at the point $c_1(v_i)$

$$\begin{aligned} c_3(v_i) &= [\bar{c}_2'(v_i) - c_2(v_i)]/u_2 \\ &= \{[\bar{c}_3(v_i) - \bar{c}_2(v_i)]/(u_3 - u_2) - [\bar{c}_2(v_i) - c_1(v_i)]/u_2\}/u_2 \\ &= [u_2 \bar{c}_3(v_i) - u_3 \bar{c}_2(v_i) + (u_3 - u_2) c_1(v_i)]/[u_2^2 (u_3 - u_2)] \end{aligned} \tag{4.4}$$

Similarly, for an arbitrary point on the boundary curve $c_4(v)$ at the position v_i, the following backward difference formula is used to calculate the first derivative at the points $c_4(v_i)$ and $\bar{c}_6(v_i)$, i.e.,

$$\begin{aligned} c_5(v_i) &= [c_4(v_i) - \bar{c}_6(v_i)]/(1-u_6) \\ \bar{c}_6'(v_i) &= [\bar{c}_6(v_i) - \bar{c}_5(v_i)]/(u_6-u_5) \end{aligned} \tag{4.5}$$

Next, use the same backward difference formula and the first derivative at the points $c_4(v_i)$ and $\bar{c}_6(v_i)$ to calculate the second derivative at the point $c_4(v_i)$

$$\begin{aligned} c_6(v_i) &= [c_5(v_i) - \bar{c}_6'(v_i)]/(1-u_6) \\ &= \{[c_4(v_i) - \bar{c}_6(v_i)]/(1-u_6) - [\bar{c}_6(v_i) - \bar{c}_5(v_i)]/(u_6-u_5)\}/(1-u_6) \\ &= [(u_6-u_5)c_4(v_i) + (1-u_6)\bar{c}_5(v_i) - (1-u_5)\bar{c}_6(v_i)]/[(1-u_6)^2(u_6-u_5)] \end{aligned} \tag{4.6}$$

When v_i changes from $v_i = 0$ to $v_i = 1$, the boundary tangents and boundary curvature in the above equations are continuous functions of the parametric variable v. Therefore, the functions $c_2(v)$, $c_3(v)$, $c_5(v)$ and $c_6(v)$ in Eq. 4.1 can be written as

$$\begin{aligned} c_2(v) &= [\bar{c}_2(v_i) - c_1(v_i)]/u_2 \\ c_3(v) &= \{[u_2\bar{c}_3(v_i) - u_3\bar{c}_2(v_i) + (u_3-u_2)c_1(v_i)]/[u_2^2(u_3-u_2)] \\ c_5(v) &= [c_4(v_i) - \bar{c}_6(v_i)]/(1-u_6) \\ c_6(v) &= \{[(u_6-u_5)c_4(v_i) + (1-u_6)\bar{c}_5(v_i) - (1-u_5)\bar{c}_6(v_i)]/[(1-u_6)^2(u_6-u_5)] \end{aligned} \tag{4.7}$$

In Figure 4.3 and Figure 4.5, some surfaces which are generated with the second approach are presented.

The surface created by the complementary solution of the associated homogeneous equation of ODE 4.2 subjected to boundary constraints Eq. 4.1 can be manipulated by the shape control parameters in Eq. 4.2, and the first and second partial derivatives in Eq. 4.1. In order to increase the capacity of surface manipulation, I can further manipulate the created surface through deforming some curves on the surface using the particular solution of ODE Eq. 4.2 which involves a sculpting force represented by the term of the right-hand side of ODE Eq. 4.2. I will address this issue in the following work. The sections below discuss how to achieve the closed form complementary solution of ODE Eq. 4.2 subjected to boundary constraints Eq. 4.1 and use the solution to create various parametric surfaces.

4.3 Closed Form Complementary Solution

After drawing two boundary curves or 2 boundary curves plus 4 control curves used for the determination of the first and second partial derivatives, the above two approaches can be used to obtain the boundary constraints Eq. 4.1, and create 3D parametric surfaces with the complementary solution $\bar{S}(u, v_i)$ of the associated homogeneous equation of ODE Eq. 4.2, i.e.,

$$\rho d^6 S(u, v_i)/du^6 + \eta d^4 S(u, v_i)/du^4 + \lambda d^2 S(u, v_i)/du^2 = 0 \qquad (4.8)$$

subjected to the boundary constraints Eq. 4.1.

The sixth order ODE Eq. 4.2 can be changed into a fourth order ODE by introducing the following vector-valued second order ordinary differential equation

$$\bar{S}(u, v_i) = d^2 S(u, v_i)/du^2 \qquad (4.9)$$

Calculating the second and fourth derivatives of $\bar{S}(u, v_i)$ with respect

to the parametric variable u and substituting them as well as Eq. 4.9 into Eq. 4.8, the following fourth order ordinary differential equation is reached

$$\rho d^4\bar{S}(u,v_i)/du^4 + \eta d^2\bar{S}(u,v_i)/du^2 + \lambda\bar{S}(u,v_i) = 0 \tag{4.10}$$

The vector-valued ordinary differential equation Eq. 4.10 can be transformed into an algebra equation by considering

$$\begin{aligned} \bar{S}_\phi(u,v_i) &= e^{ru} \\ (\phi &= x,y,z) \end{aligned} \tag{4.11}$$

Substituting Eq. 4.11 together with the second and fourth derivatives of $\bar{S}_\phi(u,v_i)$ with respect to the parametric variable u into Eq. 4.10 and deleting e^{ru}, the following quartic equation is obtained

$$\rho r^4 + \eta r^2 + \lambda = 0 \tag{4.12}$$

If $q = r^2$ is further introduced, the quartic equation Eq. 4.12 is changed into a quadratic equation below

$$\rho q^2 + \eta q + \lambda = 0 \tag{4.13}$$

whose roots are

$$q_{1,2} = -\eta(1 \pm \sqrt{1 - 4\rho\lambda/\eta^2})/(2\rho) \tag{4.14}$$

For the sake of conciseness, here only consider the situation of $4\rho\lambda/\eta^2 < 1$. The other situations can be treated with the same methodology. After substituting Eq. 4.14 into the relation $q = r^2$, and introducing two new

constants ξ_1 and ξ_2 which are determined by

$$\xi_{1,2} = \sqrt{\eta(1 \pm \sqrt{1 - 4\rho\lambda/\eta^2})/(2\rho)} \tag{4.15}$$

the following four roots of the quartic equation Eq. 4.12 are obtained

$$\begin{aligned} r_{1,2} &= \pm i\xi_1 \\ r_{3,4} &= \pm i\xi_2 \end{aligned} \tag{4.16}$$

According to the above four roots, the solution to Eq. 4.10 can be written into the following form

$$\bar{S}(u, v_i) = \bar{b}_1 \cos\xi_1 u + \bar{b}_2 \sin\xi_1 u + \bar{b}_3 \cos\xi_2 u + \bar{b}_4 \sin\xi_2 u \tag{4.17}$$

where $\bar{b}_k (k = 1, 2, 3, 4)$ are vector-valued unknown constants.

Substituting the above equation into Eq. 4.9 and solving the second order ordinary differential equation, the following solution to the sixth order ordinary differential equation Eq. 4.8 is achieved

$$S(u, v_i) = b_1 \cos\xi_1 u + b_2 \sin\xi_1 u + b_3 \cos\xi_2 u + b_4 \sin\xi_2 u + b_5 u + b_6 \tag{4.18}$$

where $b_k (k = 1, 2, ..., 6)$ are vector-valued unknown constants. Inserting Eq. 4.18 into the boundary constraints Eq. 4.1, solving for the 6 vector-valued unknown constants $b_k (k = 1, 2, ..., 6)$, and substituting them back into Eq. 4.18, the following functions which define a 3D parametric surface satisfying the ODE Eq. 4.8 and the boundary constraints Eq. 4.1 exactly is reached, and organized as below equation:

$$S(u, v) = g_1(u)c_1(v) + g_2(u)c_2(v) + g_3(u)c_3(v) + g_4(u)c_4(v) + g_5(u)c_5(v) + g_6(u)c_6(v) \tag{4.19}$$

where

$$
\begin{aligned}
g_1(u) &= -d_1\cos\xi_1 u - d_4\sin\xi_1 u + d_1(e_{11}+1)\cos\xi_2 u \\
&+e_9\sin\xi_2 u - (\xi_2 e_9 - \xi_1 d_4)u - (e_{11}d_1 - 1) \\
g_2(u) &= -(d_1+d_2)\cos\xi_1 u - (d_3+d_4)\sin\xi_1 u + (e_9+e_1 0)\sin\xi_2 u - \\
&[\xi_2(e_9+e_1 0) - \xi_1(d_3+d_4) - 1]u - e_{11}(d_1+d_2) \\
g_3(u) &= -d_5\cos\xi_1 u - d_7\sin\xi_1 u + d_{10}\cos\xi_2 u + (e_5e_9 + e_6e_{10} \\
&+e_{12}/\xi_2)\sin\xi_2 u - [\xi_2(e_5e_9+e_6e_{10}) - \xi_1 d_7 + e_{12}]u - (e_{11}d_5 - 1/\xi_2^2) \\
g_4(u) &= d_1\cos\xi_1 u + d_4\sin\xi_1 u - d_1(e_{11}+1)\cos\xi_2 u - e_9\sin\xi_2 u + (\xi_2 e_9 - \xi_1 d_4)u + e_{11}d_1 \\
g_5(u) &= d_2\cos\xi_1 u + d_3\sin\xi_1 u - d_2(e_{11}+1)\cos\xi_2 u - e_{10}\sin\xi_2 u + (\xi_2 e_{10} - \xi_1 d_3)u + e_{11}d_2 \\
g_6(u) &= -d_6\cos\xi_1 u - d_8\sin\xi_1 u + d_6(e_{11}+1)\cos\xi_2 u + (e_7e_9 + e_8e_{10} - \\
&e_{13}/\xi_2)\sin\xi_2 u - [\xi_2(e_7e_9+e_8e_{10}) - \xi_1 d_8 - e_{13}]u - e_{11}d_6
\end{aligned}
\tag{4.20}
$$

here, the $d_i(i = 1, 2, 3...10)$ in Eq. 4.20 are obtained by:

$$
\begin{aligned}
d_1 &= e_1/(e_1e_3 - e_2e_4) \\
d_2 &= -e_2/(e_1e_3 - e_2e_4) \\
d_3 &= e_3/(e_1e_3 - e_2e_4) \\
d_4 &= -e_4/(e_1e_3 - e_2e_4) \\
d_5 &= d_1e_5 + d_2e_6 \\
d_6 &= d_1e_7 + d_2e_8 \\
d_7 &= d_4e_5 + d_3e_6 \\
d_8 &= d_4e_7 + d_3e_8 \\
d_9 &= (e_{11}+1)(d_1+d_2) \\
d_{10} &= -1/\xi_2^2 + (e_{11}+1)d_5
\end{aligned}
\tag{4.21}
$$

and the $e_i(i = 1, 2, 3...13)$ in Eq. 4.20 and Eq. 4.21 are obtained by:

$$
\begin{aligned}
e_1 &= \xi_1(\cos\xi_1 - 1) + \xi_1^2 \sin\xi_1(1-\cos\xi_2)/(\xi_2 \sin\xi_2) \\
e_2 &= \sin\xi_1 - \xi_1 + \xi_1^2 \sin\xi_1(1/\sin\xi_2 - 1/\xi_2)/\xi_2 \\
e_3 &= \cos\xi_1 - 1 + \xi_1^2(1-\cos\xi_2)/\xi_2^2 + \xi_1^2(\cos\xi_2 - \cos\xi_1)(1/\xi_2 - 1/\sin\xi_2)/\xi_2 \\
e_4 &= \xi_1(-\sin\xi_1 + \xi_1 \sin\xi_2/\xi_2) + \xi_1^2(\cos\xi_2 - \cos\xi_1)(\cos\xi_2 - 1)/(\xi_2 \sin\xi_2) \\
e_5 &= (1/\xi_2 - \cos\xi_2/\sin\xi_2)/\xi_2 \\
e_6 &= (\sin\xi_2 + \cos^2\xi_2/\sin\xi_2 - \cos\xi_2/\sin\xi_2)/\xi_2 \\
e_7 &= (1/\sin\xi_2 - 1/\xi_2)/\xi_2 \\
e_8 &= (1-\cos\xi_2)/(\xi_2 - \sin\xi_2) \\
e_9 &= \xi_1^2[d_4 \sin\xi_1 - d_1(\cos\xi_2 - \cos\xi_1)]/(\xi_2^2 \sin\xi_2) \\
e_{10} &= \xi_1^2[d_3 \sin\xi_1 - d_2(\cos\xi_2 - \cos\xi_1)]/(\xi_2^2 \sin\xi_2) \\
e_{11} &= \xi_1^2/\xi_2^2 - 1 \\
e_{12} &= \cos\xi_2/(\xi_2 \sin\xi_2) \\
e_{13} &= 1/(\xi_2 \sin\xi_2)
\end{aligned}
\tag{4.22}
$$

4.4 Continuity between Adjacent Surface Patches

The mathematical equations describing parametric surfaces involve two parametric variables u and v. When parametric surface patches are connected together, they should maintain specified continuities in both u and v parametric directions. In what follows, I will investigate the continuity in the u parametric direction, and then the continuity in the v parametric direction.

4.4.1 Continuity in Parametric Direction U

For the continuity of two connected surface patches in the u parametric direction, assuming the two parametric surface patches to be connected together are $S(u,v)$ and $\hat{S}(u,v)$, respectively. They are defined

by Eq. 4.1 and Eq. 4.8. For the surface patch $\hat{S}(u,v)$, the vector-valued function $S(u,v)$, known vector-valued functions $c_j(v)(j=1,2,3,4,5,6)$, and shape control parameters ρ, η and λ in Eq. 4.1 and Eq. 4.2 are replaced by $\hat{S}(u,v)$, $c_j(v)(j=1,2,3,4,5,6)$, and $\hat{\rho},\hat{\eta},\hat{\lambda}$, respectively.

If the two surface patches are required to maintain up to the curvature continuities at the interface where the two surface patches are connected together, the constraints $S(1,v)=\hat{S}(0,v)$, $\partial S(1,v)/\partial u=\partial\hat{S}(0,v)/\partial u$, and $\partial S^2(1,v)/\partial u^2=\partial\hat{S}^2(0,v)/\partial u^2$ must be satisfied. If the surface patch $\hat{S}(u,v)$ with the following boundary constraints are created,

$$\begin{aligned}
&\hat{S}(0,v)=c_4(v) \quad \partial\hat{S}(0,v)/\partial u=c_5(v) \quad \partial\hat{S}^2(0,v)/\partial u^2=c_6(v)\\
&\hat{S}(1,v)=\hat{c}_4(v) \quad \partial\hat{S}(1,v)/\partial u=\hat{c}_5(v) \quad \partial\hat{S}^2(1,v)/\partial u^2=\hat{c}_6(v)
\end{aligned} \tag{4.23}$$

the continuities of the position, tangent and curvature between the two surface patches in the u parametric direction are achieved.

The above conclusion is also demonstrated visually in Figure 4.1(a). In the figure, the top surface patch is created by introducing the following boundary conditions into Eq. 4.19 and Eq. 4.20.

$$\begin{aligned}
&S(0,v)=c_1(v) \quad \partial S(0,v)/\partial u=w_1c_1(v) \quad \partial S^2(0,v)/\partial u^2=w_2c_1(v)\\
&S(1,v)=c_4(v) \quad \partial S(1,v)/\partial u=w_3c_4(v) \quad \partial S^2(1,v)/\partial u^2=w_4c_4(v)
\end{aligned} \tag{4.24}$$

and the bottom surface patch is generated by using the following boundary constraints

$$\begin{aligned}
&\hat{S}(0,v)=c_4(v) \quad \partial\hat{S}(0,v)/\partial u=w_3c_4(v) \quad \partial\hat{S}^2(0,v)/\partial u^2=w_4c_4(v)\\
&\hat{S}(1,v)=\hat{c}_4(v) \quad \partial\hat{S}(1,v)/\partial u=w_5\hat{c}_4(v) \quad \partial\hat{S}^2(1,v)/\partial u^2=w_6\hat{c}_4(v)
\end{aligned} \tag{4.25}$$

4.4.2 Continuity in Parametric Direction V

For the continuity of two connected surface patches in the v parametric direction, the first and second partial directives of
$S(v,u) = [S_x(u,v), S_y(u,v), S_z(u,v)]^T$ with respect to the parametric variable v are required and can be determined from Eq. 4.19. The first and second partial directives of $S(u,v)$ and the mathematical equations of the surface $S(u,v)$ can be unified as

$$\begin{aligned} S^{(i)}(u,v) = g_1(u)c_1^{(i)}(v) + g_2(u)c_2^{(i)}(v) + g_3(u)c_3^{(i)}(v) \\ +g_4(u)c_4^{(i)}(v) + g_5(u)c_5^{(i)}(v) + g_6(u)c_6^{(i)}(v) \end{aligned} \tag{4.26}$$

where $i = 0, 1, 2$ indicates the surface of $S(u,v)$, its first and second partial derivatives with respect to the parametric variable v, respectively, and $c_k^{(i)}(v) = (i = 0, 1, 2; k = 1, 2, ..., 6)$ can be determined from boundary constraints 4.1.

I still denote the two surface patches to be connected together with $S(u,v)$ and $\hat{S}(u,v)$, and use Eq. 4.26,4.20,4.21,4.22,4.15 to determine them and their first and second partial derivatives with respect to the parametric variable v. In the equations, with and without the symbol $\wedge$ on top, denotes for the two surface patches $\hat{S}(u,v)$ and $S(u,v)$ and their first and second partial derivatives, respectively.

At the interface where the two surface patches $S(u,v)$ and $\hat{S}(u,v)$ are to be connected together, the continuity of the position function, first and second partial derivatives requires $S^{(i)}(u,1) = \hat{S}^{(i)}(u,0)(i = 0, 1, 2)$ to be met. According to Eq. 4.26, if $g_k^{(i)}(u) = \hat{g}_k^{(i)}(u)(i = 0, 1, 2; k = 1, 2, ..., 6)$ and $c_k^{(i)}(1) = \hat{c}_k^{(i)}(0)$ the constraints $S^{(i)}(u,1) = \hat{S}^{(i)}(u,0)$ are always satisfied.

If setting $\xi_n = \hat{\xi}_n$, have $e_{jp} = \hat{e}_{jp}(p = 1, 2, ..., 13)$ according to Eq. 4.22, $d_l = \hat{d}_l(l = 1, 2, ..., 10)$ from Eq. 4.21, and $g_k^{(i)}(u) = \hat{g}_k^{(i)}(u)(i = 0, 1, 2; k = 1, 2, ..., 6)$ according to Eq. 4.20. Therefore, two adjacent surface patches achieve up to C2 continuities at their interface if $\xi_n = \hat{\xi}_n(n = 1, 2)$

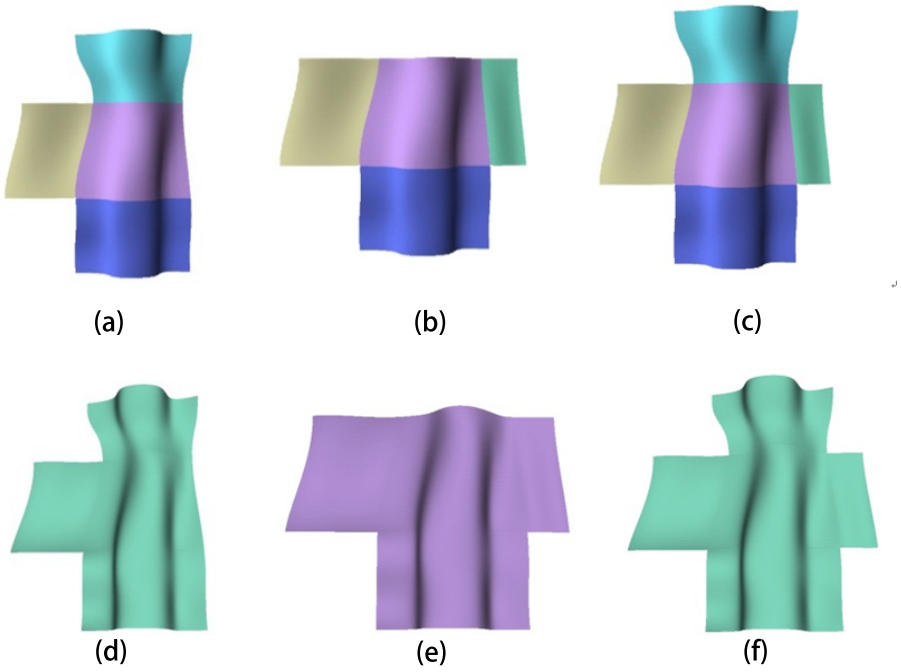


Figure 4.1: *Connected surface patches with up to C2 continuities*

and the position of the boundary curve and its first and second partial derivatives with respect to the parametric variable v for the first surface patch are equal to the ones for the second surface patch at the interface point of the two boundary curves.

Image in Figure 4.1(b) is also used to visually indicate two surface patches are connected together in the parametric direction v and achieve up to C2 continuities. The shape control parameters for the two surface patches are: $\rho = 1$, $\lambda = 1$, $\eta = 3$, $\hat{\rho} = 0.1$, $\hat{\lambda} = 0.1$ and $\hat{\eta} = 3$ and the function of the boundary curves for the two surface patches are $c_1(v)$ and $\hat{c}_1(v)$.

More examples are presented in Figure 4.1 to demonstrate various patches can be smoothly connected together with up to C2 continuities, both in direction u and direction v.

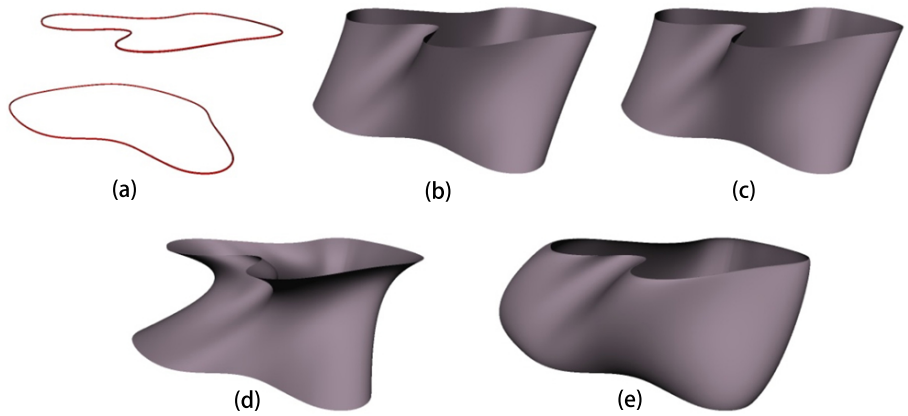


Figure 4.2: *Closed surface modelling by using two boundary curves*

4.5 Creation of Single Surface

With the implemented users interface, firstly used the first approach to create a closed surface. I draw two boundary curves which were depicted in Figure 4.2(a). The surface created using Maya loft operation was shown in Figure 4.2(b).

Then, took the mathematical equation of the first and second partial derivatives to be the same form as the corresponding boundary curves but modify them with the modifying coefficients $w_k (k = 1, 2, 3, 4)$. When $w_k = 0 (k = 1, 2, 3, 4)$, the surface is generated in Figure 4.2(c) which is the same as that achieved with Maya loft operation.

If only the first partial derivative is changed and kept the second partial derivative unchanged, i. e. take $w_1 = -1$, $w_3 = 10$ and $w_2 = w_4 = 0$, the surface indicated in Figure 4.2(d) was produced. If the modifying coefficients were changed into: $w_1 = w_3 = 0$, $w_2 = 1$ and $w_4 = -10$ which means only the second partial derivative was modified, the surface shown in Figure 4.2(e) was obtained.

These images indicate that the proposed surface creation method not only can produce surface shapes generated by Maya loft operation, but also can create different shape changes through manipulating the first and second partial derivatives.

Next, used the second approach to create a closed surface. Firstly, draw two boundary curves which are the top and bottom ones on the image shown in Figure 4.3(a). Then use the top boundary curve as reference to create another two curves as the control curves of the top boundary curve. Similarly, generate the other two control curves of the bottom boundary curve. Using Maya loft operation, obtain the surface given in Figure 4.3(a) which interpolates the 2 boundary curves and four control curves. Using Eq. 4.2 to determine the first and second partial derivatives at the top and bottom boundary curves and first taking the shape control parameters to be $\rho = \lambda = 1$, and $\eta = 3$, the surface created with the proposed second approach is given in Figure 4.3(b).

Comparing Figure 4.3(a) with Figure 4.3(b), it can be concluded that this approach can also create the similar surfaces to those by Maya loft operation. However, this approach has two advantages over Maya loft operation.

First, this approach can achieve up to C2 continuities between the adjacent patches and there are no manual operations to stitch these adjacent patches together. This is because that the two adjacent patches created with this approach share the same first and second partial derivatives.

Second, the shape of the surfaces created with this approach is controllable. This can be well demonstrated by comparing Figure 4.3(b) and Figure 4.3(c) where the surface in Figure 4.3(c) is obtained with the same boundary and control curves as those in Figure 4.3(b) but the shape control parameters ρ and λ are reduced to 0.0001. The shape change between Figure 4.3(b) and Figure 4.3(c) can be more clearly observed from Figure 4.3(d) where the surfaces in Figure 4.3(b) and Figure 4.3(c) are shown in the same figure.

If the four control curves in Figure 4.3(a) are scaled down, different surfaces are achieved, given in Figure 4.3(e) to Figure 4.3(h) where the surface in Figure 4.3(e) is from Maya loft operation, that in Figure 4.3(f) is from the proposed second approach with the shape control parameters $\rho = \gamma = 1$ and $\eta = 3$, the one in Figure 4.3(g) is from $\rho = \lambda = 0.0001$ and

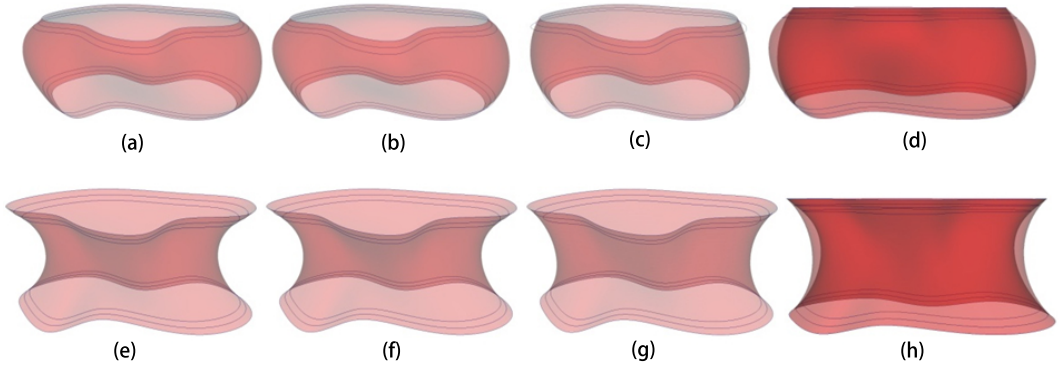


Figure 4.3: *Closed surface modelling by using two boundary and four control curves*

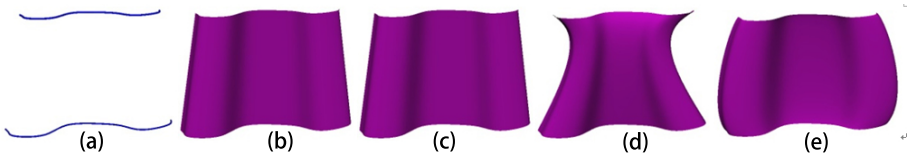


Figure 4.4: *Open surface modelling by using two boundary curves*

$\eta = 3$. Figure 4.3(h) is used to compare the surfaces given in Figure 4.3(f) and Figure 4.3(g).

The images in Figure 4.3 also indicate that the surfaces created with this approach may or may not pass through the four control curves depending on different values of the shape control parameters.

Apart from the applications in creating closed surfaces, the proposed two approaches also apply to open surfaces. I demonstrate this in the following Figure 4.4 and Figure 4.5.

In Figure 4.4, use two boundary curves and the first approach to create different shapes of the surface defined by the two boundary curves and different first and second partial derivatives. Figure 4.4(a) indicates the two boundary curves, Figure 4.4(b) is from Maya loft operation, Figure 4.4(c) to Figure 4.4(e) is from the first approach where Figure 4.4(c) is from zeroed first and second partial derivatives, Figure 4.4(d) is from the first partial derivative $w_1 = -1$, $w_3 = 10$ and zeroed second partial derivative, and Figure 4.4(e) is from zeroed first partial derivative and the second partial derivative $w_2 = 1$ and $w_4 = -10$. These images also demonstrate that the proposed first approach not only can create those

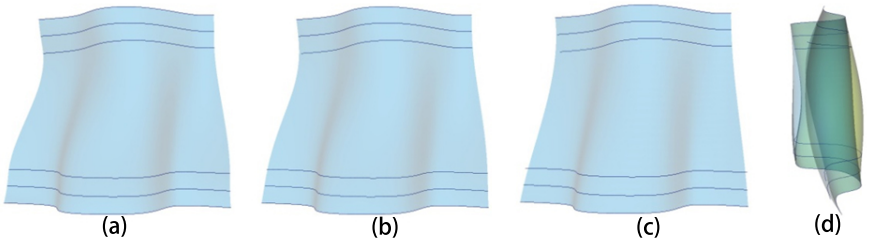


Figure 4.5: *Open surface modelling by using two boundary and four control curves*

generated by Maya loft operation but also those which cannot obtained from Maya loft operation.

In Figure 4.5, use two boundary curves, four control curves shown in the figure and the second approach to generate different surface shapes. Figure 4.5(a) is generated by Maya loft operation, Figure 4.5(b) is from the second approach and the shape control parameters $\rho = \lambda = 1$ and $\eta = 3$, and Figure 4.5(c) is from the second approach and the shape control parameters $\rho = \lambda = 0.0001$ and $\eta = 3$. The influence of the shape control parameters on surface shapes can be clearly observed from the side view Figure 4.5(d) of the surfaces in Figure 4.5(b) and Figure 4.5(c) where the surface in yellow is from Figure 4.5(b) and the other is from Figure 4.5(c). These images also indicate that the proposed second approach not only creates the surface shapes by Maya loft operation, but also other surface shapes which cannot be obtained through Maya loft operation.

4.6 Creation of Surface Objects

This section introduces how to create complicated objects with the above developed approach. Two approaches will be discussed below.

The first approach is to decompose a surface object into parts. For some complicated parts, they are further decomposed into simple surface patches. The proposed approach is used to create each surface patch which shares the same boundary constraints of position and first and second partial derivatives with the adjacent patch at its four edges.

Figure 4.6: *Creation of surface model*

Taking the dog model in Figure 4.6 as an example, first decompose the dog model into parts of head, neck, torso, tail, ears, eyes, front legs and rear legs. Each of the front legs is further decomposed into three surface patches and each of the rear legs is divided into two surface patches. Then, each surface patch is created and two adjacent patches share the same constraints of the position and the first and second partial derivatives.

The second approach is to generate some sketched curves on the model to be created. These sketched curves define some 4-sided and 3-sided patches. For each of 4-sided or 3-sided patches, an ODE surface is created with the constraints of the position and the first and second partial derivative being the same as those of the adjacent surface patches. Here take the creation of a female face as an example. Some sketched curves in Figure 4.7(a)(b) are used to define the female model, and accordingly, ODE surface patches are used to build the female model, shown in Figure 4.7(c)(d).

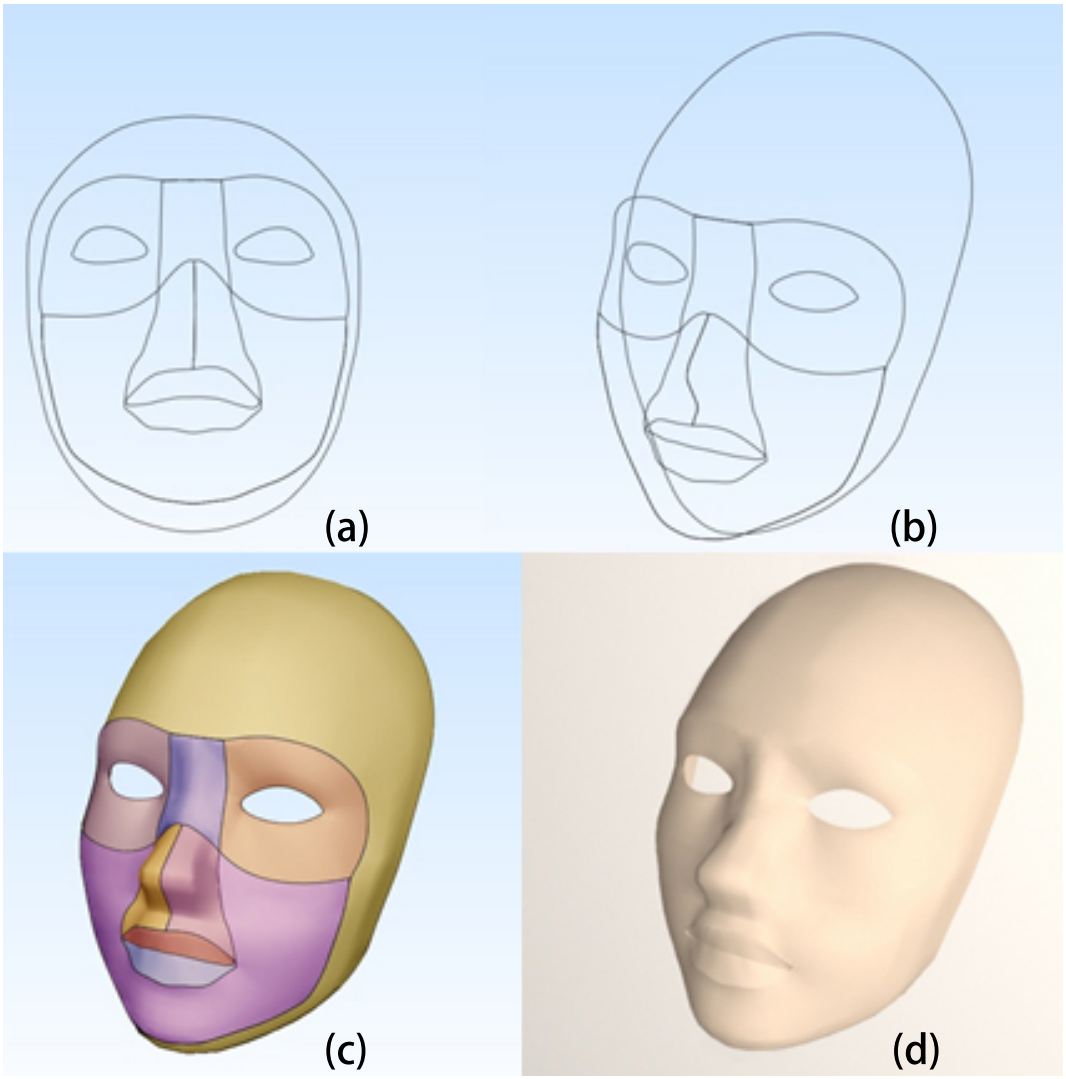


Figure 4.7: *Creation of facial surface*

4.7 ODE based Curve Network for facial Modeling

In this section, one application of C2 continuous surface based on the Ordinary Differential Equation(ODE) is demonstrated to build one curve network structure to represent 3D facial mesh, which could be used to obtain diverse faces in world by changing corresponding parameters.

4.7.1 Facial Action Coding System

Facial Action Coding System (FACS) is a system to taxonomize human facial movements by their appearance on the face, based on a system originally developed by a Swedish anatomist named Carl-Herman Hjortsj Hjortsjö [1969]. It was later adopted by Paul Ekman and Wallace V. Friesen, and published in 1978 Ekman & Friesen. Ekman, Friesen, and Joseph C. Hager published a significant update to FACS in 2002 HAGER [2002]. Movements of individual facial muscles are encoded by FACS from slight different instant changes in facial appearance Hamm et al. [2011]. It is a common standard to systematically categorize the physical expression of emotions, and it has proven useful to psychologists and to animators. Due to subjectivity and time consumption issues, FACS has been established as a computed automated system that detects faces in videos, extracts the geometrical features of the faces, and then produces temporal profiles of each facial movement Hamm et al. [2011]. The pioneer F-M Facial Action Coding System 3.0 (F-M FACS 3.0) Freitas-Magalhães [2018] was created in 2018 by Dr. Freitas-Magalhes, and presents 4,000 segments in 4K, using 3D technology and automatic and real-time recognition (FaceReader 7.1).The F-M FACS 3.0 features 8 pioneering action units (AUs) and 22 pioneering tongue movements (TMs), in addition to functional and structural nomenclature Freitas-Magalhães [2018].

Using FACS, Freitas-Magalhães [2013] human coders can manually code nearly any anatomically possible facial expression, deconstructing it into the specific action units (AU) and their temporal segments that produced the expression. As AUs are independent of any interpretation, they can be used for any higher order decision making process including recognition of basic emotions, or pre-programmed commands for an ambient intelligent environment. The FACS Manual is over 500 pages in length and provides the AUs, as well as Ekman's interpretation of their meaning.

FACS defines AUs, which are a contraction or relaxation of one or more muscles. It also defines a number of Action Descriptors, which

differ from AUs in that the authors of FACS have not specified the muscular basis for the action and have not distinguished specific behaviors as precisely as they have for the AUs.

For example, FACS can be used to distinguish two types of smiles as follows Del Giudice & Colle [2007]:

Insincere and voluntary Pan-Am smile: contraction of zygomatic major alone Sincere and involuntary Duchenne smile: contraction of zygomatic major and inferior part of orbicularis oculi. Although the labeling of expressions currently requires trained experts, researchers have had some success in using computers to automatically identify FACS codes, and thus quickly identify emotions. Computer graphical face models, such as CANDIDE or Artnatomy, allow expressions to be artificially posed by setting the desired action units.

The use of FACS has been proposed for use in the analysis of depression Reed et al. [2007], and the measurement of pain in patients unable to express themselves verbally Lints-Martindale et al. [2007].

FACS is designed to be self-instructional. People can learn the technique from a number of sources including manuals and workshops, and obtain certification through testing Hager [2003]. The original FACS has been modified to analyze facial movements in several non-human primates, namely chimpanzees Parr et al. [2007], rhesus macaques Parr et al. [2010], gibbons and siamangs Waller et al. [2012] and orangutans Caeiro et al. [2013]. More recently, it was adapted for a domestic species, the dog Waller et al. [2013].

Thus, FACS can be used to compare facial repertoires across species due to its anatomical basis. A study conducted by Vick and others (2006) suggests that FACS can be modified by taking differences in underlying morphology into account. Such considerations enable a comparison of the homologous facial movements present in humans and chimpanzees, to show that the facial expressions of both species result from extremely notable appearance changes. The development of FACS tools for different species allows the objective and anatomical study of facial expressions

Table 4.1: *Emotion-related facial action units*

Emotion	Action units
Happiness	6+12
Sadness	1+4+15
Surprise	1+2+5B+26
Fear	1+2+4+5+7+20+26
Anger	4+5+7+23
Disgust	9+15+16
Contempt	R12A+R14A

in communicative and emotional contexts. Furthermore, a cross-species analysis of facial expressions can help to answer interesting questions, such as which emotions are uniquely human Vick et al. [2007].

EMFACS (Emotional Facial Action Coding System)Friesen et al. [1983] and FACSAID (Facial Action Coding System Affect Interpretation Dictionary) consider only emotion-related facial actions. Examples of these are:

For clarification, FACS is an index of facial expressions, but does not actually provide any bio-mechanical information about the degree of muscle activation. Though muscle activation is not part of FACS, the main muscles involved in the facial expression have been added here for the benefit of the reader.

Action units (AUs) are the fundamental actions of individual muscles or groups of muscles.

Action descriptors (ADs) are unitary movements that may involve the actions of several muscle groups (e.g., a forwardthrusting movement of the jaw). The muscular basis for these actions hasn't been specified and specific behaviors haven't been distinguished as precisely as for the AUs.

For most accurate annotation, FACS suggests agreement from at least two independent certified FACS encoders.

Intensities of FACS are annotated by appending letters AE (for minimal-maximal intensity) to the action unit number (e.g. AU 1A is the weakest trace of AU 1 and AU 1E is the maximum intensity possible for the in-

Table 4.2: *Intensity scoring*

A	Trace
B	Slight
C	Marked or pronounced
D	Severe or extreme
E	Maximum

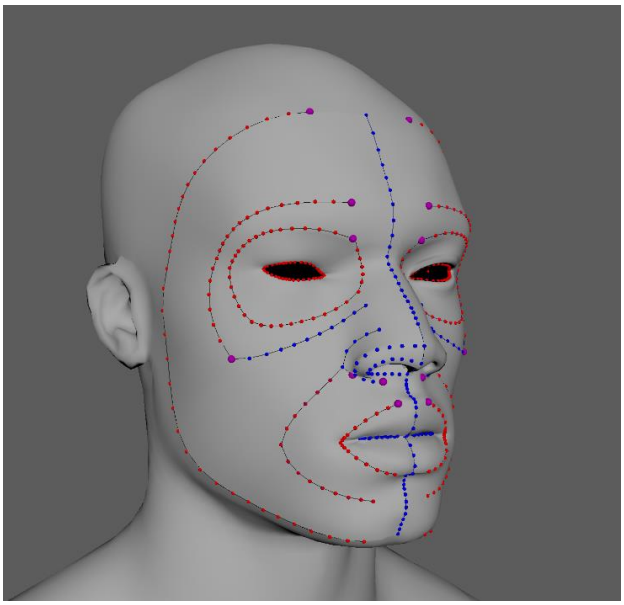


Figure 4.8: *One practical facial Curve Network*

dividual person).

There are other modifiers present in FACS codes for emotional expressions, such as "R" which represents an action that occurs on the right side of the face and "L" for actions which occur on the left. An action which is unilateral (occurs on only one side of the face) but has no specific side is indicated with a "U" and an action which is unilateral but has a stronger side is indicated with an "A".

4.7.2 Curve Network Structure

As analysis above, Facial Action Coding System could be used as powerful rules to achieve any anatomically possible facial expression.

But how FACS could be implemented into computer application, to

Table 4.3: *List of action units and action descriptors (with underlying facial muscles: Main codes*

AU No.	FACS name	Muscular basis
0	Neutral face	
19	Tongue show	
14	Dimpler	buccinator
15	Lip corner depressor	depressor anguli oris
4	Brow lowerer	depressor glabellae, depressor supercilii
16	Lower lip depressor	depressor labii inferioris
25	Lips part	depressor labii inferioris
2	Outer brow raiser	depressor labii inferioris
1	Inner brow raiser	frontalis (pars medialis)
18	Lip pucker	incisivii labii superioris
13	Sharp lip puller	levator anguli oris (also known as caninus)
9	Nose wrinkler	levator labii superioris alaeque nasi
10	Upper lip raiser	levator labii superioris, caput infraorbitalis
5	Upper lid raiser	levator palpebrae superioris, superior tarsal muscle
26	Jaw drop	masseter; relaxed temporalis and internal pterygoid
17	Chin raiser	mentalis
6	Cheek raiser	orbicularis oculi (pars orbitalis)
7	Lid tightener	orbicularis oculi (pars palpebralis)
8	Lower lip depressor	depressor labii inferioris
22	Lip funneler	orbicularis oris
23	Lip tightener	orbicularis oris
24	Lip pressor	orbicularis oriss
28	Lip suck	orbicularis oris
21	Neck tightener	platysma
27	Mouth stretch	pterygoids, digastric
20	Lip stretcher	risorius w/ platysma
12	Lip corner puller	zygomaticus major
11	Nasolabial deepener	zygomaticus minor

Table 4.4: *Eye movement codes*

AU No.	FACS name
61	Eyes turn left
M61	Eyes left
62	Eyes turn right
M62	Eyes right
63	Eyes up
64	Eyes down
65	Walleye
66	Cross-eye
M68	Upward rolling of eyes
69	Eyes positioned to look at other person
M69	Head and/or eyes look at other person

Table 4.5: *Visibility codes*

AU No.	FACS name
70	Brows and forehead not visible
71	Eyes not visible
72	Lower face not visible
73	Entire face not visible
74	Unsociable

replace tedious manual work with automatically processing.

In order to solve the above-mentioned problem, this work here develops one curve network as the mathematical representation of FACS.

Based on the Facial Action Coding System (FACS) mentioned in above section, one Curve Network is build to represent both the neutral human face and approximate diverse facial actions only by several curves, and could reconstruct the whole facial model from C2 continuous ODE surface method. The contributions include greatly reducing the data size for facial models and provide one new quick approach to build different human faces.

One practical facial Curve Network structure is created by implementing ordinary differential equation (ODE)-based parameterization method with C2 continuity, and carry on experiments to create a curve network as the mathematical representation of Facial Action Coding System (FACS), in order to replace tedious manual work with automatically

Table 4.6: *Gross behavior codes, these codes are reserved for recording information about gross behaviors that may be relevant to the facial actions that are scored.*

AU No.	FACS name
29	Jaw thrust
30	Jaw sideways
31	Jaw clencher
32	[Lip] bite
33	[Cheek] blow
34	[Cheek] puff
35	[Cheek] suck
36	[Tongue] bulge
37	Lip wipe
38	Nostril dilator
39	Nostril compressor
40	Sniff
41	Lid droop
42	Slit
43	Eyes closed
44	Squint
45	Blink
46	Wink
50	Speech
80	Swallow
81	Chewing
82	Shoulder shrug
84	Head shake back and forth
85	Head nod up and down
91	Flash
92	Partial flash
97	Shiver/tremble
98	Fast up-down look

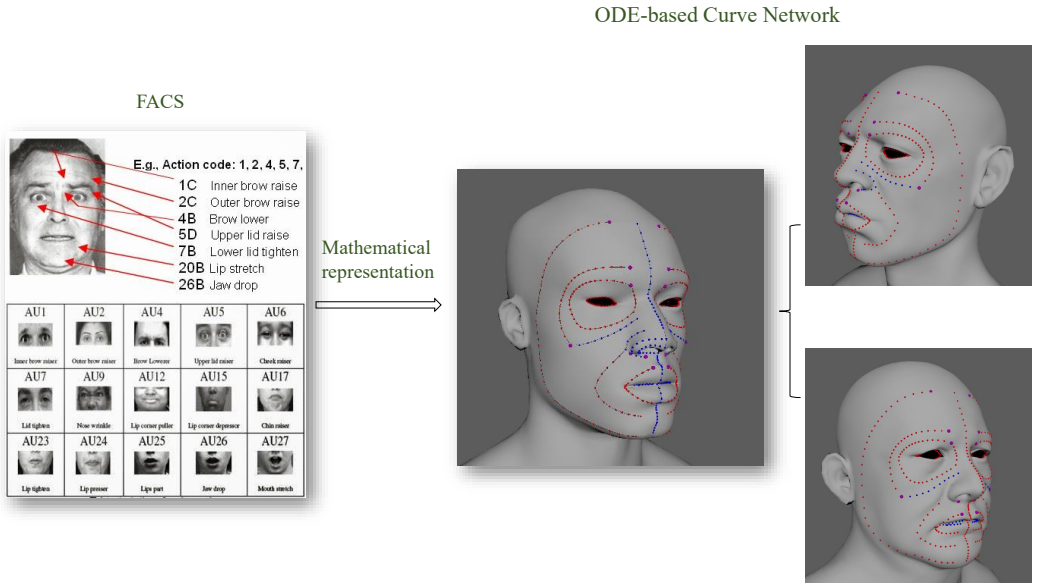


Figure 4.9: *Mathematical representation of Facial Action Coding System (FACS)*

processing for achieving various facial expressions, shown as below Fig. 4.8

Using this curve-net work, (1) could keep the essential feature of human face, shown as Fig. 4.9; (2) provide a way to set values to FACS Unit attributes for more accurate computing; (3) use it as one underlying driver to transmit the diverse expressions between different human faces, shown as Fig. 4.10:

The proposed process of presenting different Facial Action units by curve network include:

(1) Extract Curve Network from one template mesh, Fig. 4.11.

(2) Generate mesh by Curve Network based on C2 continuous ODE surface method, Fig. 4.12.

(3) Comparison between original mesh and the reconstructed mesh using curve network based on C2 continuous ODE surface method, shown

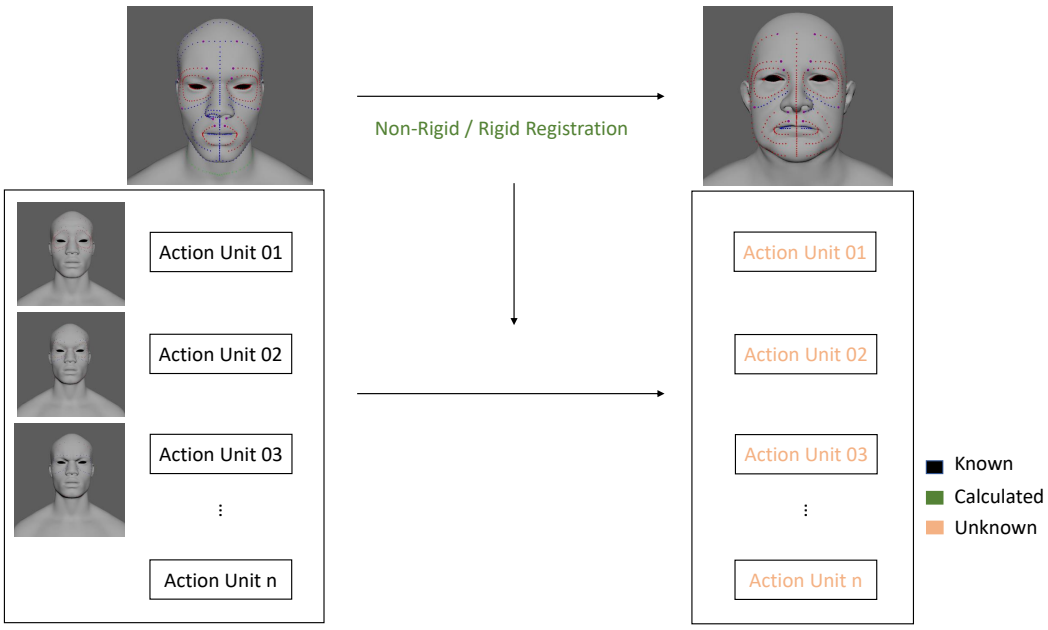


Figure 4.10: *Use rigid/ non-rigid registration to transmit the diverse expressions between different human faces*

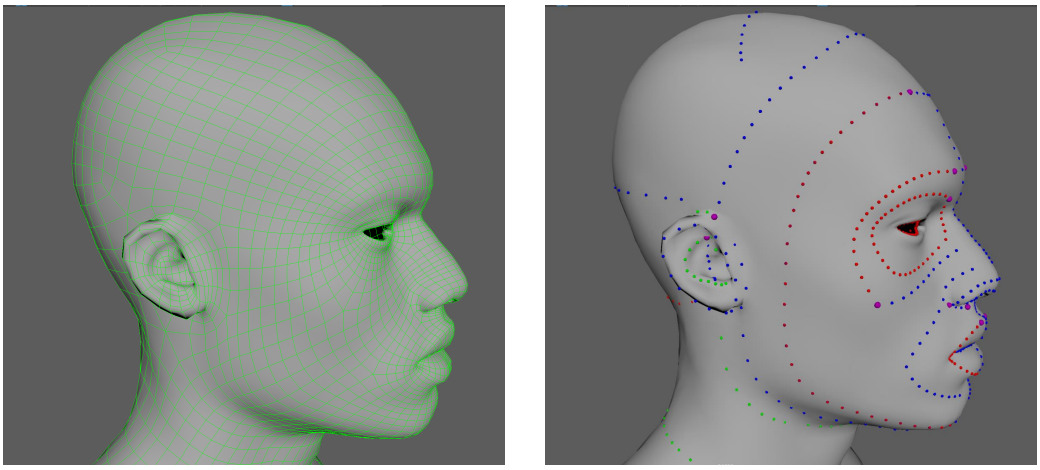
as Fig. 4.13 and Fig. 4.14.

(4) Detailed part of the reconstructed mesh, in this experiment, use 4 curves to reconstruct the ear part, shown as Fig. 4.15.

(5) Use the defined facial curve network to generate different Action Units, examples shown as Fig. 4.16, Fig. 4.17, Fig. 4.18 and Fig. 4.19.

The implementation of GUI for showing the curve network structure and use default parameters to generate initialized surfaces is shown as below Fig. 4.20 and Fig. 4.21. This implementation of the ODE based continuous surface algorithm is accomplished by C++(11) in Visual studio 2017, and Microsoft operating system.

Instead of using default values, could change controllers value in GUI to optimize generated surface, the difference could be shown as Fig. 4.22.



(a) Original Mesh (b) Curve Network

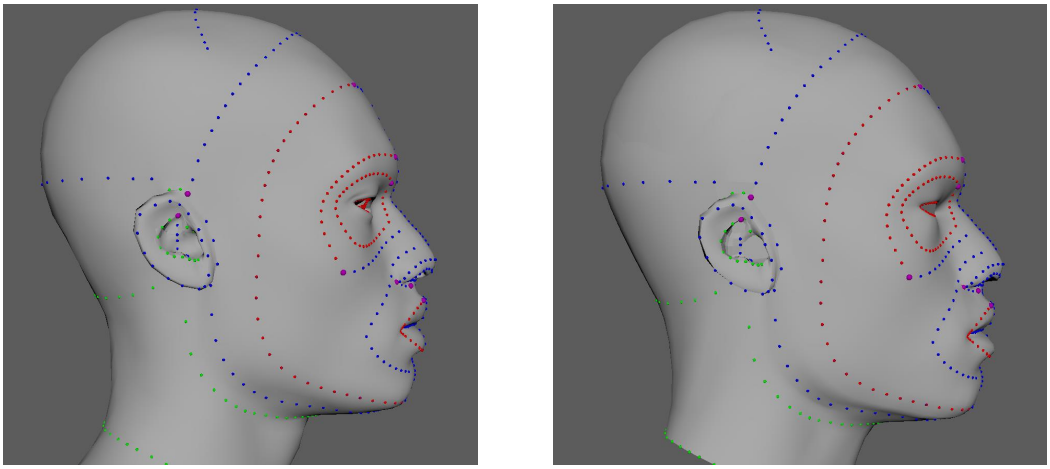
Figure 4.11: *Extract Curve Network (b) from one template mesh (a)*

4.8 Discussion and Conclusion

In this chapter, an efficient ODE-based surface generating technique has been proposed to create C2 continuous surface. This technique is based on the mathematical model consisting of a vector-valued sixth order ordinary differential equation and C2 continuous boundary constraints.

The solution to the vector-valued sixth order ordinary differential equation is a three-dimensional curve which is used to define an isoparametric line of surface models. By making the isoparametric line satisfying the constraints of the position, and the first and second partial derivatives, a surface patch is created. The surface models consisting of such surface patches always maintain C2 continuity between different surface patches.

In order to create surface patches quickly, the analytical solution to the vector-valued sixth order ordinary differential equation subjected to the constraints of position and the first and second partial derivatives is developed. With the obtained analytical solution, the users only generate two boundary curves or two boundary curves plus four control curves,

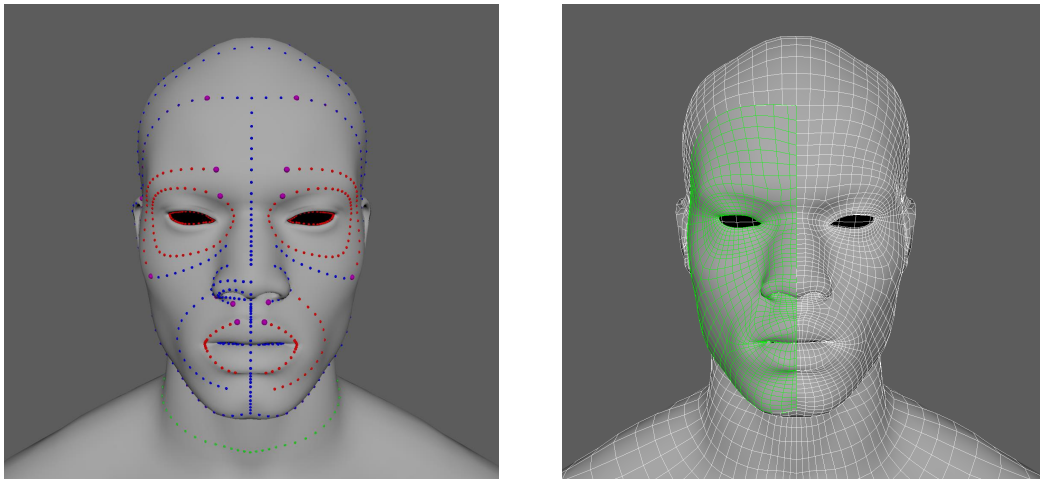


(a) Curve Network on Original Mesh (b) Generated Mesh

Figure 4.12: *Generate mesh (b) by Curve Network based on C2 continuous ODE surface method (a)*

the proposed approach transforms them into the boundary constraints of the position and the first and second derivatives, and create a surface patch satisfying the boundary constraints exactly.

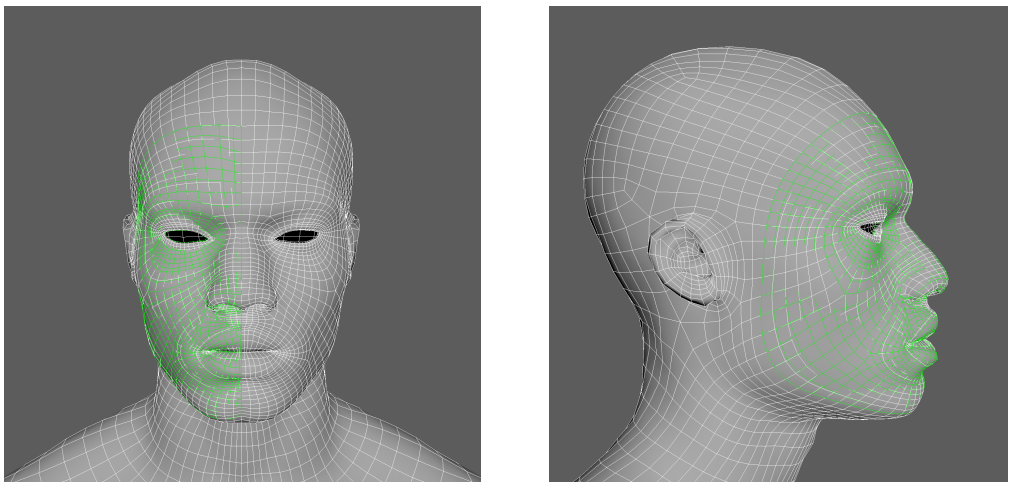
When building surface models, existing patch modeling techniques require tedious and time-consuming manual operations to stitch two separate patches together and achieve tangential or curvature continuity. The technique proposed in this work solves this problem. All created surface patches are connected together automatically with C2 continuity. Besides, the technique presented in this work can achieve more shape variations defined by the same boundary constraints since the proposed technique can manipulate surfaces through shape control parameters, and the first and second partial derivatives.



(a) Original Mesh

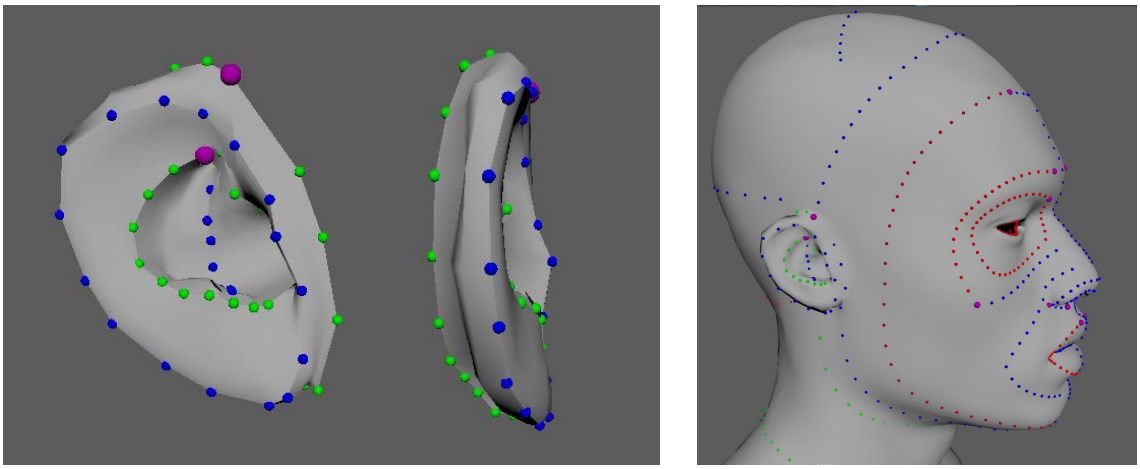
(b) Generated Mesh (Green), Original mesh (white).

Figure 4.13: *(a) shows the Curve Network on original mesh, (b) compare the difference between generated mesh by C2 continuous ODE surface method in green and the original mesh in white.*



Generated Mesh (Green), Original mesh (white).

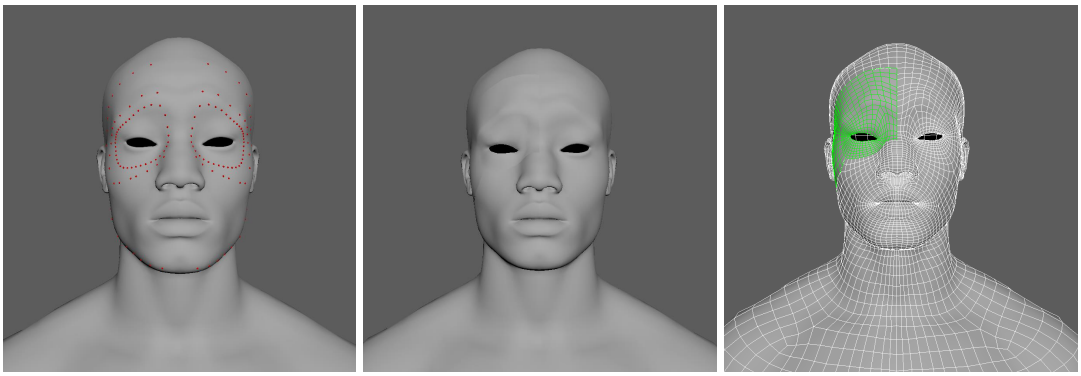
Figure 4.14: *Comparison the difference between generated mesh by C2 continuous ODE surface method in green and the original mesh in white.*



(a) Reconstruct the detailed part (b) Original Mesh

Figure 4.15: *(a) shows the reconstructed detailed ear part of facial mesh, (b) shows the original mesh.*

Action Uint 01



(a) (b) (c)

Figure 4.16: *(a) shows the curve network on facial action unit 01, (b) shows the combination of reconstructed part and the rest facial mesh of action unit 01, (c) shows the comparison of reconstructed mesh by curve network and C2 continuous ODE surface method in green and original mesh in white.*

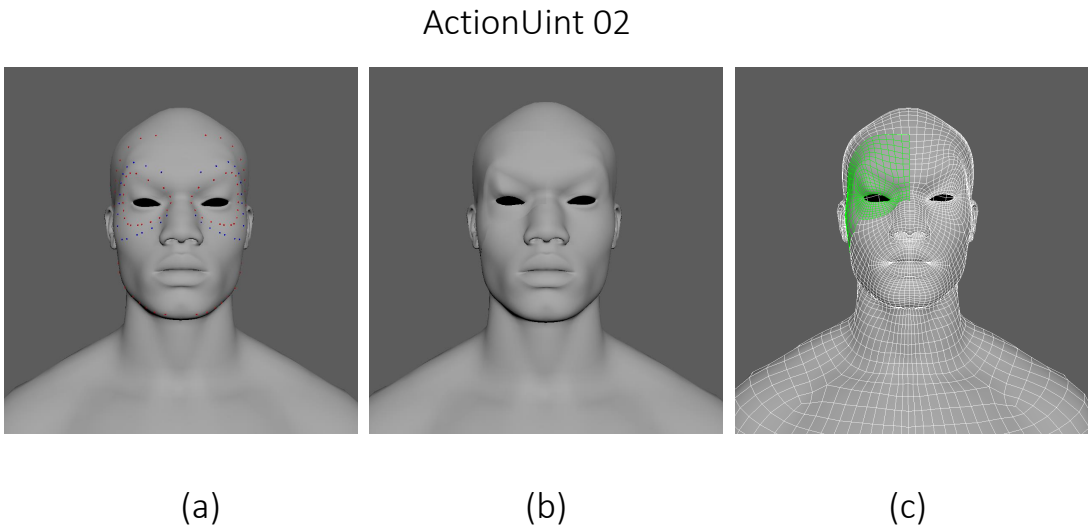
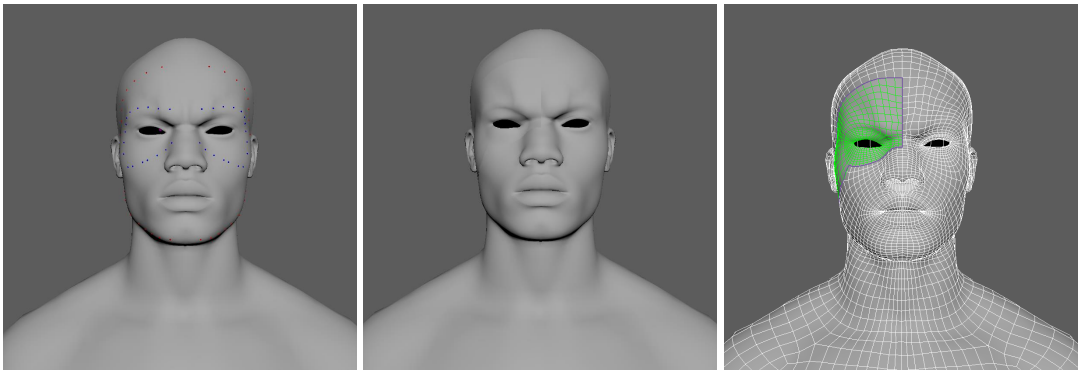


Figure 4.17: *(a) shows the curve network on facial action unit 02, (b) shows the combination of reconstructed part and the rest facial mesh of action unit 02, (c) shows the comparison of reconstructed mesh by curve network and C2 continuous ODE surface method in green and original mesh in white.*

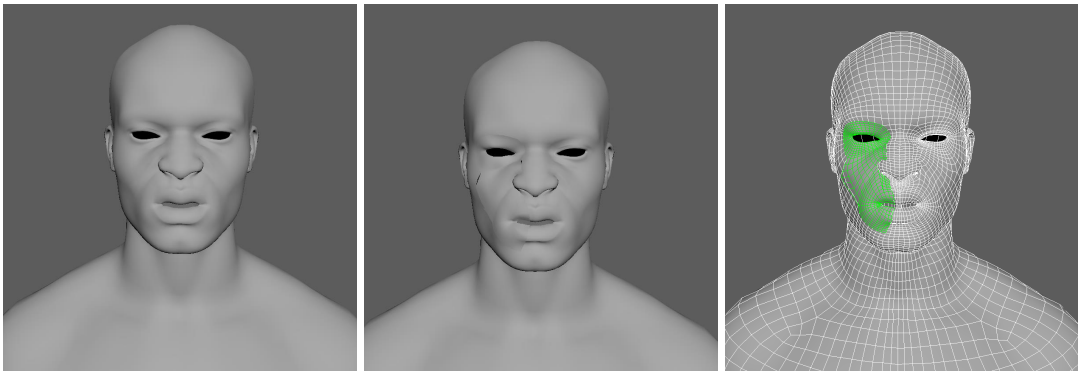
Action Uint 04



(a) (b) (c)

Figure 4.18: *(a) shows the curve network on facial action unit 04, (b) shows the combination of reconstructed part and the rest facial mesh of action unit 04, (c) shows the comparison of reconstructed mesh by curve network and C2 continuous ODE surface method in green and original mesh in white.*

Action Uint 09



(a) (b) (c)

Figure 4.19: *(a) shows the facial action unit 09, (b) shows the combination of reconstructed part and the rest facial mesh of action unit 09, (c) shows the comparison of reconstructed mesh by curve network and C2 continuous ODE surface method in green and original mesh in white.*

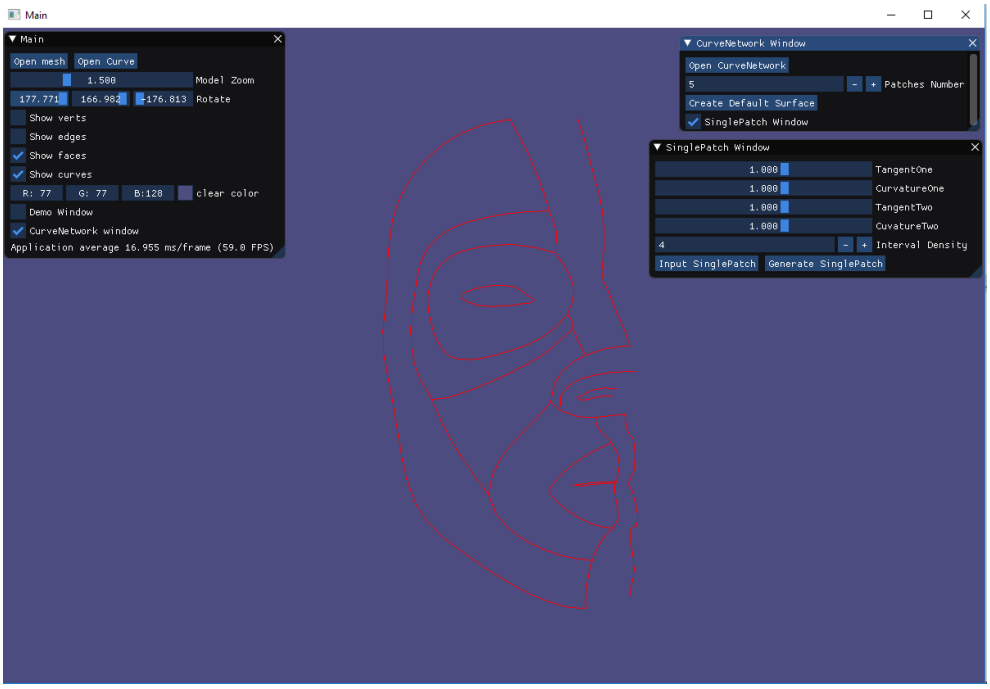


Figure 4.20: *GUI for showing the curve network structure and use default parameters to generate initialized surfaces.*

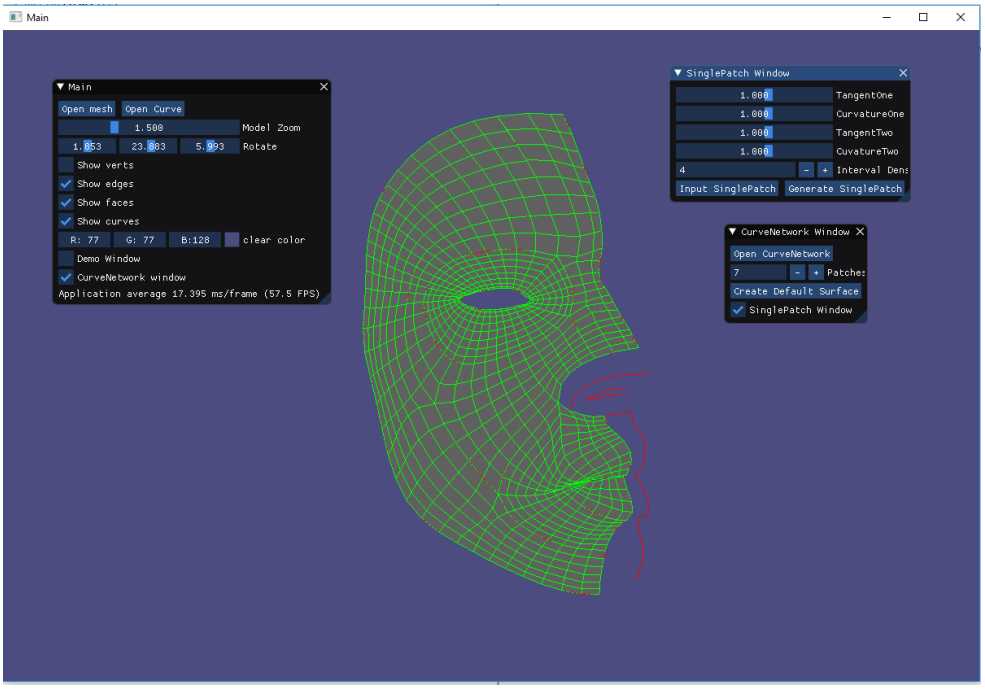


Figure 4.21: *Use the GUI and default parameters to reconstruct seven initialized patches of the facial mesh.*

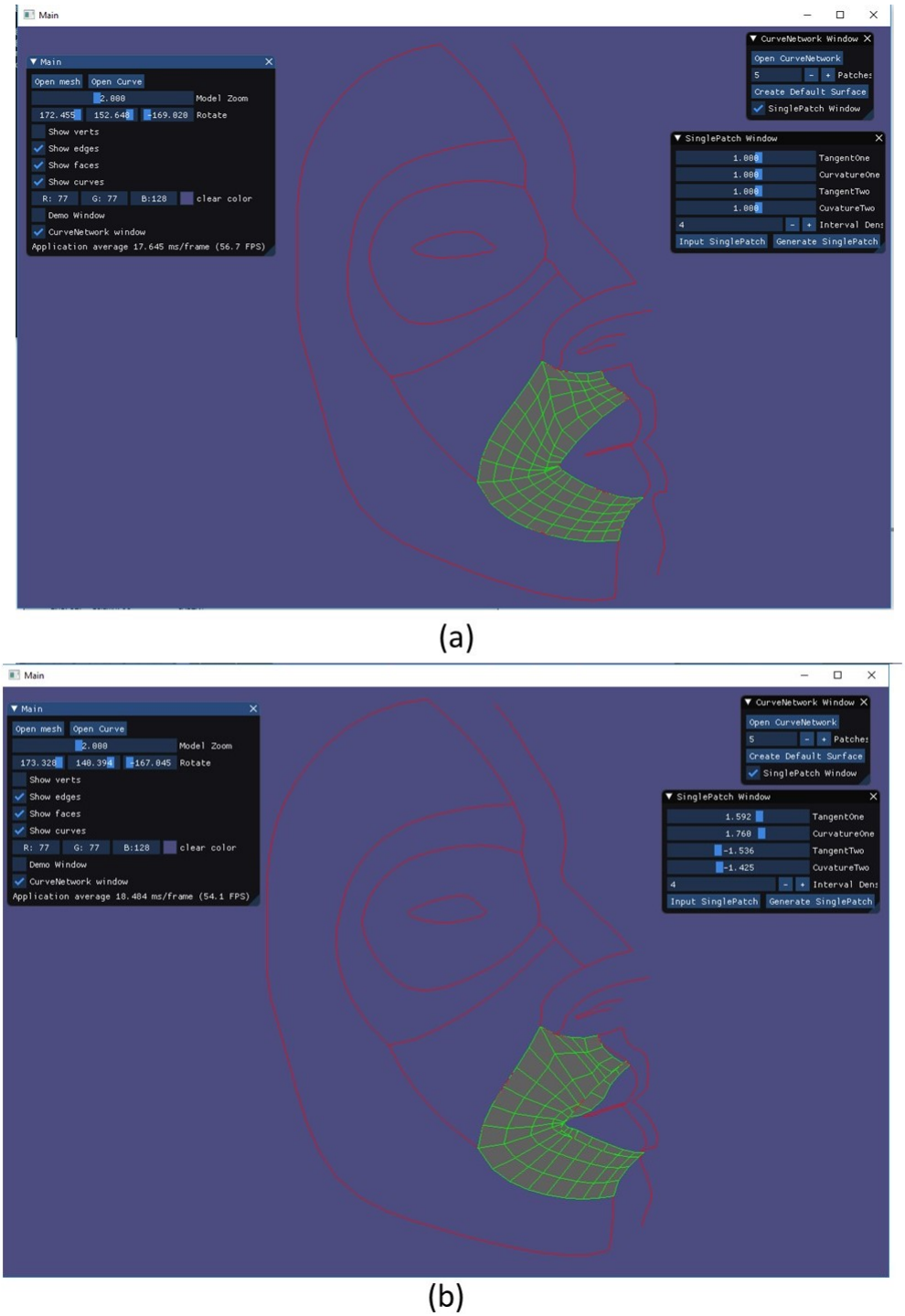


Figure 4.22: *(a) shows the patch surface generated by default shape control values, (b) shows changing the values of shape control paremeters will generate different surfaces.*

Chapter 5

Automatic Generation of Animation Skeleton to Assist Dynamic Skin Deformation

Since non-automatic rigging requires heavy human involvements, and various automatic rigging algorithms are less efficient in terms of computational efficiency. Especially for current curve-based skin deformation methods, identifying the iso-parametric curves and creating the animation skeleton need tedious and time-consuming manual work. Although several automatic rigging methods have been developed, but not aim at curve-based models. To tackle this issue, this chapter propose a new rigging algorithm for automatic generation of dynamic skin deformation to quickly identify iso-parametric curves and create an animation skeleton in a few milliseconds, which can be seamlessly used in curve-based skin deformation methods to make the rigging process fast enough for highly efficient computer animation applications.

5.1 Introduction

Skin deformation technique stays an essential and standardized part of many character animation applications these days including academia

and industry practices. The usefulness of a skin deformation technique is generally measures in terms of three major characteristics: efficiency, realism, and easiness. Over years, researchers have developed a variety of skin deformation techniques to improve them. These skin deformation techniques can be roughly classified into geometric skin deformation, example-based skin deformation, and physics-based skin deformation.

Geometric skin deformation approaches directly bind skin deformation with the underlying skeleton movement without taking any underlying physics into consideration. Despite their simplicity and efficiency, they cannot produce highly realistic skin deformation without non-trivial additional efforts.

Example-based skin deformation approaches interpolate a set of given example poses to improve realism or produce certain skin deformation effects that cannot be easily produced by geometric skinning approaches. Therefore, example-based skin deformation approaches are often used together with geometric skinning. In order to achieve satisfactory realism, sufficient example skin shapes are required. In addition, how to optimally design or obtain such a set of example poses is still considered as a widely open research problem. For example, how many example poses are sufficient for high quality skinning of a specific 3D model; and where those example poses should be positioned in the deformation space.

Physics-based skin deformation approaches introduce physics rules or musculoskeletal systems to drive and simulate the movement of the skin surface, which probably have produced more physically-realistic skin deformation results to date. Although physics-based skin deformation approaches bring in good realism, they rely on heavy numerical calculations and have the following limitations: 1) numerical calculations such as finite element simulations require specialized knowledge and skills, making them more difficult to learn and use, 2) a lot of manual operations such as mesh generation of finite elements and specification of boundary conditions are involved leading to more pre-processing time, 3) large computer memory requirements and high computational cost making them more difficult to achieve high efficiency. Low efficiency caused by numerical

calculations of physics-based simulations is due to large data size or high dimension. Model reduction can be used to reduce the data size or dimension and obtain an approximation of lower accuracy in significantly less time but more complicated implementation.

Animation quality of virtual characters without clothes is determined by efficiency and realism of skin deformations. Automatic rigging can raise the efficiency, and data-driven and physics-based skinning techniques can improve the realism. In existing literature, a rigging process includes generating and placing a skeleton inside a skin mesh and relating skin shape changes to the skeleton movements.

The above discussions indicate that geometric, example-based, and physics-based skin deformation approaches have their own strengths and limitations. How to maximize their strengths and minimize their limitations to achieve more realistic and efficient skin deformations easily requires further investigations. Besides, rigging is usually done by hand, especially for current curve-based skin deformation methods, identifying the iso-parametric curves and create the animation skeleton need tedious and time-consuming manual work. Although several automatic rigging methods have been developed, but not aim at curve-based models.

To tackle this issue, this work proposes a new rigging algorithm to quickly identify iso-parametric curves and create an animation skeleton in a few milliseconds, which can be seamlessly used in physically curve-based skin deformation methods to make the rigging process fast enough for highly efficient computer animation applications.

Since modelling of skin surfaces and their deformations can be simplified as those of the curves defining skin surfaces. When physics-based approaches are used to deal with skin deformations, such a simplification can significantly reduce computer resource and raise computational efficiency.

Chaudhry et al. [2015] investigated a finite difference solution to curve-based dynamic skin deformations, the curves are extracted manually from these reconstructed skin deformation models, which is tedious and

time-consuming. Thus, this chapter develops:

1) an automatic and quick vertex identification algorithm which quickly identifies all the vertices at iso-paramteric curves on the example skin meshes at two extreme poses for following physics-based skin deformation calculations. 2) an automatic and highly efficient rigging algorithm which quickly creates a new skeleton and embeds it in the two example skin meshes through shape matching with a template skeleton. 3) integration of curve-based presentations, automatic rigging, ODE-based simulation to achieve good realism, high efficiency for skin deformation with small data size.

The chapter is structured below. First, an overview of the proposed approach and developed animation system is given in Section 5.2. Then, identifying iso-parametric curves is investigated in Section 5.3, and creating animation skeleton is developed in Section 5.4. After that, ODE-based dynamic skin deformation experimental results are given in Section 5.5 and evaluations are given in Section 5.6. Finally, the concluded and future work is discussed in Section 5.7.

5.2 Overview

This section gives an overview of the proposed approach. As shown in Fig. 5.1, the proposed approach consists of three main parts: identification of iso-parametric curves, creation of animation skeleton, and dynamic skin deformations. The purpose of identifying the iso-parametric curves has two: one is to convert polygon meshes into curve-defined models to be applied in ODE-based dynamic skin deformations discussed in Section 5.6, and the other is to create curve skeleton which will be changed into animation skeleton.

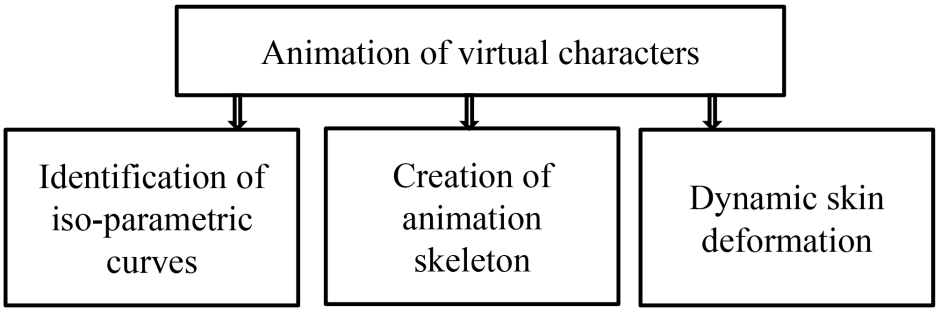


Figure 5.1: *Overview of the proposed approach*

Most skin surfaces are polygon meshes defined by discrete vertices, since volumetric models would require the information inside skin meshes that is often unavailable in given skin examples. Due to the deformation complexity of muscle, fat, and skin, current physics-based models, in spite of better realism than geometric skin deformations, are still unable to accurately predict skin shape changes. Thus, assuming skin deformations are similar to elastic bending of thin plates and develop a surface model of physics-based skin deformations, represented by Ordinary Differential equations (ODE) on the basis of iso-parametric curves, to describe the underlying physics for skin shape creation between given examples to achieve high realistic skin deformation results, which largely reduces data size and computing time.

Meanwhile, the identified isoparametric curves naturally provide a good basis to automatically and high efficiently determine the curve skeletons of characters, using down-sampling algorithm and shape matching algorithm.

Finally, the ODE based dynamic deformation model is proposed to describe physics of curve deformations and its finite difference solution is developed to determine shape changes of the iso-parametric curves. So that the proposed dynamic skin deformation technique can create realistic deformed skin shapes for character animation efficiently with largely reduced computing time.

5.3 Identify Iso-parametric Curves

The proposed automatic rigging method is specifically designed and tailored for the purpose of character animation. It is not targeted for general rigging on meshes with arbitrary topologies. Given that converting a triangle mesh to a quad mesh has been well supported in many software packages such as Autodesk Maya, without loss of generality, this approach only focus on quad meshes as the input character poses.

Assuming that a parametric variable u is used to denote the circumferential direction ($u = u_i$ is an iso-parametric curve), and use v to indicate the axial direction of each of the parts of a character model, the specific goal of vertex identifier in the proposed approach is to identify all the vertices on each iso-parametric curve, $u = u_i$ $(i = 0, 1, 2, ..., I)$.

The proposed iso-parametric curves identification process starts from the farthest end of the character parts such as limbs and tail, which can be manually specified or automatically identified through the analysis of curvature distribution. Specifically, several seed points are first selected to start the construction of iso-parametric curves in their circumferential directions. Then, their neighboring vertices in the axial direction are continuously added as new seed points. In this process, a vertex is visited at most once, which would lead to closed loops at the beginning and gradually open the loops especially at the connection place between different body parts.

The first step is to identify the axial and circumferential directions of each of the parts of a 3D character mesh. Fig. 5.2 illustrates an example how this process is done. A ray is firstly drawn starting from the vertex 2 of a cat model in the normal direction of the facet 2-17-30-11-2, which intersects another facet at a point P_0 (shown in Fig. 5.2(c)). Then, 10 lines are drawn, including the line 2-P_0, at the same angle of 18^o, that pass the middle point P_1 of the line 2-P_0 in the plane determined by the triangle 2-P_0-11-2, and obtain 20 intersecting points $P_k^1(k = 1, 2, ..., 20)$ illustrated in Fig. 5.2(d). Afterwards, the centre P_0^1 of the 20 points are calculated and all the Euclidean distances $d(P_0^1, P_k^1)$ (k=1,2,...,20),

among which the minimal and maximal Euclidean distances are found, d^1_{min} and d^1_{max}. At the end, the difference between the minimal and maximal Euclidean distances can be calculated, $d^1 = d^1_{max} - d^1_{min}$.

Similarly, in the plane determined by the triangle 2-17-P_0-2, do the same operations, i.e., drawing 10 lines in the plane, obtaining 20 intersecting points $P^2_k(k = 1, 2, ..., 20)$, calculating the minimal and maximal Euclidean distances d^2_{min} and d^2_{max}, and calculating the difference $d^2 = d^2_{max} - d^2_{min}$. If $d_1 < d_2$, the direction determined by the 20 intersecting points $P^1_k(k = 1, 2, ..., 20)$ is the *circumferential* direction. Otherwise, the direction determined by the 20 intersecting points $P^2_k(k = 1, 2, ..., 20)$ is the circumferential direction.

An accurate method is proposed below to automatically identify the vertices on the iso-parametric curve u_i in the circumferential direction, and use their indexes to obtain their coordinate values on the mesh.

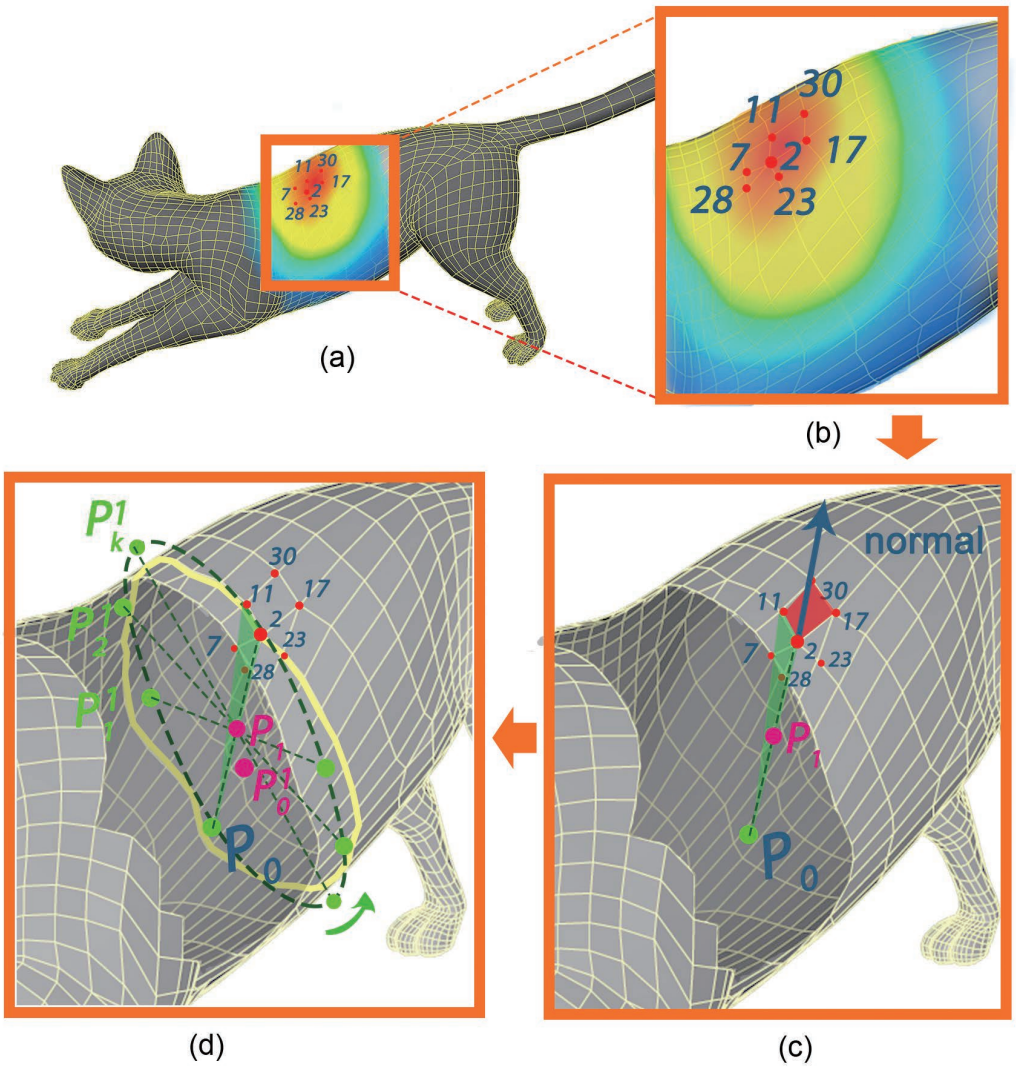


Figure 5.2: *Illustration of Identifying Iso-parametric Curves on example model*

Starting with the vertex 2, and find all the edges which directly connect the vertex 2 to some of its 1-ring vertices, i.e, 2-17, 2-11, 2-28 and 2-23 as illustrated in Fig. 5.2(a). Then, calculate the angles between these edges and the plane in the circumferential direction. The one with the minimum angle (e.g., the vertex 11 in Fig. 5.2) is selected to connect to the vertex 2. Afterward, the same operations are performed on the new selected vertex in order to select next vertex. The process is repeated until connecting back to the vertex 2 to find all the vertices on the iso-parametric curve u_i.

Fig.5.3 shows the identified vertices on iso-parametric curves of a cat

model (a). The computational times for the vertex identifier process are given in Table 5.1.

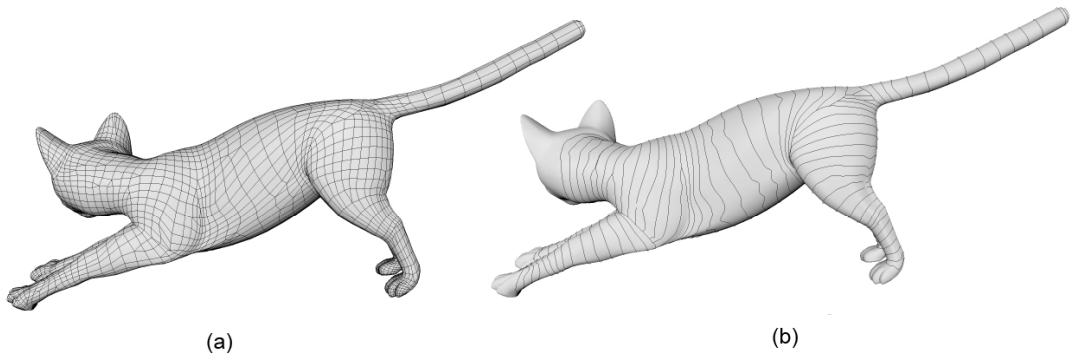


Figure 5.3: *The proposed iso-parametric curves identifying step transforms a quad mesh (a) into discrete vertices on iso-parametric curves(b).*

5.4 Creating Animation Skeleton

The identified isoparametric curves provide a very good basis to automatically and quickly determine the curve skeletons of the decomposed parts. To this end, calculate the centres of all isoparametric curves. The curve skeleton is obtained by connecting these centres. Then, use the following down-sampling algorithm and shape matching algorithm to create an animation skeleton from the obtained curve skeleton and a template skeleton as discussed below.

First, the centre of the points $\bar{\mathbf{S}}_0(u_i, v_j)(j = 0, 1, 2, ..., J)$ on one isoparametric curve are calculated to obtain the curve skeletons for the limbs and the torso including neck and head. The curve skeleton of a human arm consisting of the calculated centres is shown as Figure 5.4(a).

Inspired by the work of Au et al. [2008]; Pan et al. [2009], a down-sampling algorithm is introduced, to find the joints. Here set 20^o shown in Figure 5.4(b) as the minimal angle threshold between two connected line segments of a curve skeleton. In other words, if the relative rotation between two connected line segments of a curve skeleton is greater than this threshold, a joint is specified at this point shared by the two adjacent

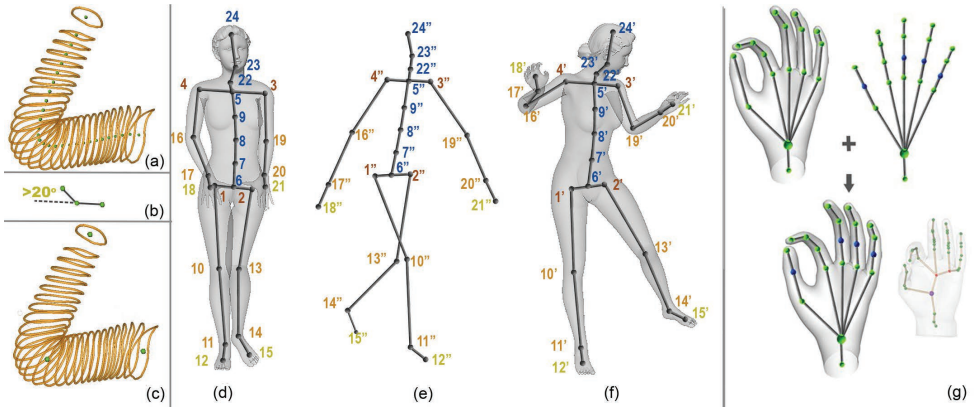


Figure 5.4: *Creation of the animation skeleton*

line segments. Figure 5.4(c) depicts the joint between the forearm and upper arm determined by the down-sampling algorithm.

Also, the centres of seam-curves are denoted as the joints between the limbs and the torso labeled as 1, 2, 3 and 4. And the centres closest to the fingers and toes are labeled as 12, 15, 18 and 21 on the curve skeletons shown in Figure 5.4(d). With the down-sampling algorithm, the joints 11, 13, 14, 16, 17, 19, 20 has the larger angle between two adjacent line segments from the curve skeletons, thus can be identified, shown in Figure 5.4(d).But for other joints, such as 10, which cannot be identified with the above down-sampling algorithm, a template skeleton and a shape matching algorithm can be used to identify it.

This approach uses the human skeleton in Bharaj et al. [2012], shown in Figure 5.4 (e), as a template skeleton. Besides, the shape matching algorithm is based on the matching of four feature joints 1, 2, 3, 4 generated by seam-curves and a symmetric axis from joint 6 to joint 24 in the skeleton template Figure 5.4(e), and the similarity measure between the curve skeleton and the template skeleton. Once a template skeleton is selected, firstly check whether the joints 3, 4, and 5 are on one same straight line, and the joints 1, 2, and 6 are on another straight line. In Figure 5.4 (e), the double prime symbol is used to indicate the joints of the template skeleton. For this template skeleton, the joints 3, 4, and 5 are on one straight line, and the joints 1, 2, and 6 are on another straight line. The joints 5 and 6 are determined by 5.1

$$
\begin{aligned}
t_5 &= t_4 + (t_3 - t_4)/2 \\
t_6 &= t_1 + (t_2 - t_1)/2 \\
&(t = x, y, z)
\end{aligned}
\tag{5.1}
$$

After the joints 5 and 6 have been determined, they are connected to the curve skeleton between the joint 5 and 6 to obtain the symmetric axis. Then, determine the locations of the joints 7, 8, and 9 on torso and the locations of the joints 22, 23 , 24 on neck and head according to the length similarity measure. For example, the joint 9 is determined by $L_{59}/L_{56} = L_{5"9"}/L_{5"6"}$ where L_{56} is the arc length of the curve skeleton from the joint 5 to joint 6, and $L_{5"6"} = L_{5"9"} + L_{9"8"} + L_{8"7"} + L_{7"6"}$.

Similarly, animation skeletons for other models such as a human hand can be automatically generated in Figure 5.4(g). In Figure 5.4(g), the top left hand shows the skeleton after the down-sampling algorithm, but four fingers are almost spreading, use the template in the middle and the matching algorithm to supplement the four joints (blue) shown as the top right hand. The hand on the bottom right of Figure 5.4(g) shows the skeleton extracted by Au et al. [2008] is based on mesh contraction. The mesh contraction presented in Au et al. [2008] takes about 10 seconds to obtain the skeleton of the mesh with 25,000 vertices. In contrast, the proposed method takes less than 5 milliseconds to create the skeleton of the mesh with 32,185 vertices.

If there are two or more skin meshes for one model at different poses, the above algorithm can be used to obtain a skeleton for each of them for revision. If some joints obtained from different meshes are different, the joints with larger angles are used. For example, the joint 19 for the skin mesh at the starting pose shown in Figure 5.4(d) is on the isoparametric curve $\bar{\mathbf{S}}_0(u_i, v)$, but the corresponding joint $19'$ on the skin mesh at the ending pose shown in Figure 5.4(f) is on the isoparametric curve $\mathbf{S}_1(u_j, v)$, the joint $19'$ is taken to be the joint between the forearm and upper arm

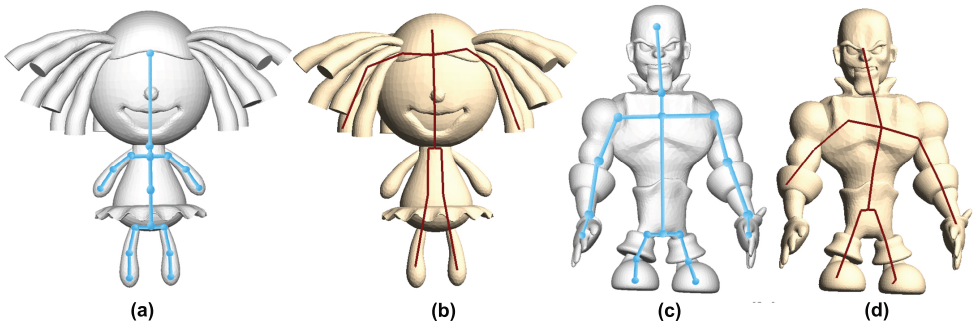


Figure 5.5: *Skeleton results generated by the proposed automatic rigging algorithm.*

since it has a larger angle. Here, the prime symbol is used to indicate the skin mesh at the ending pose.

Only taking the mesh at one pose shown in Figure 5.5 as input (models from Baran & Popović [2007]), the proposed automatic rigging algorithm including both down-sampling and the shape matching with the template skeleton are used to generate the skeletons in Figure 5.5(a) and (c), comparing them with the skeletons in Figure 5.5(b) and (d) obtained by the rigging method in Baran & Popović [2007]. As shown in the figure, our method can generate comparable skeletons as Baran & Popović [2007], if not better.

5.5 ODE-based Dynamic Skin Deformation

After automatically identifying the iso-parametric curves and creating the animation skeleton in high efficiency, the curve-based skin deformation method in Chaudhry et al. [2015] is employed to create the dynamic deformation model for character animation which demonstrates the proposed method can be used in curve-based skin deformation approach to create physically realistic animation high efficiently. For the sake of completeness, the physics-based skin deformation method presented in Chaudhry et al. [2015] is introduced briefly.

The govern equation of dynamic bending of isotropic elastic beams

can be written as Chaudhry et al. [2015]:

$$EI[\partial^4 w/\partial x^4] + \rho A[\partial^2 w/\partial t^2] = F(x) \tag{5.2}$$

where w is the deformation, x is a coordinate variable, t is a time variable, $F(x)$ is the bending force, E is Youngs modulus, I is the second moment, ρ is the density, and A is the cross-section area of the beam.

If the skin mesh is described with a parametric representation, the deformation will involve three components x, y, and z, the bending force will involve three components f_x, f_y, and f_z, and the coordinate variable will become the parametric variable v. Accordingly, the above equation is changed into:

$$EI[\partial^4 q(v,t)/\partial v^4] + \rho A[\partial^2 q(v,t)/\partial t^2] = fq(x) \quad (q = x, y, z) \tag{5.3}$$

Since considering physicals-base skin deformations of human character in this approach, I determine E and ρ with Youngs modulus and Poisson ratio of human skin. If the skin shapes at the rest pose $t = 0$ and the deformed pose $t = 1$ are known, the deformation and deformation rate at the rest pose $t = 0$ are zero, and the deformation at the deformed pose $t = 1$ is the difference between the skin shape at the deformed pose $t = 1$ and the skin shape at the rest pose $t = 0$. Therefore, the following constraints are obtained:

$$\begin{gathered} t = 0 \quad q = q_1 - q_0 = 0 \quad d_q/d_t = 0 \\ t = 1 \quad q = q_1 - q_0 \\ (q = x, y, z) \end{gathered} \tag{5.4}$$

Substituting the central finite difference formula of the fourth derivative of q with respect to the parametric variable v and the first and second derivatives of q with respect to the time variable t into the above

two equations, a system of linear algebra equations are obtained which can be written in the matrix form below

$$[K_{nm}]\{Q_{nm}\} = \{F_{nm}\} \tag{5.5}$$

where n indicates the nth time integration, and m stands for the mth node of the finite difference grid.

For the undeformed skin mesh at the rest pose Fig. 5.6(a) and the deformed skin mesh at the deformed pose Fig. 5.6(e), the above linear algebra equations are solved to obtain the intermediate meshes shown in (b), (c) and (d) of Fig. 5.6.

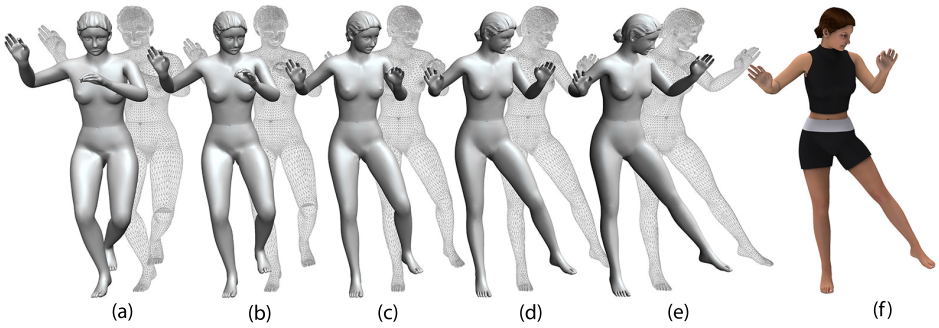


Figure 5.6: *The models (a) and (e) are input as examples, intermediate models (b)-(d) are created efficiently, (f) is the rendered model.*

Given multiple poses as the input, the proposed method can also produce physically-realistic skin deformations. For example, the skin shapes at the three poses shown in (a), (f) and (k) of Fig. 5.7 are known. The skin shapes at (a) and (c) are used to create the deformed skin shapes at the pose (b), and use the skin shapes at the pose (c) and (e) to create the deformed skin shapes at the poses (d) of Fig. 5.7.

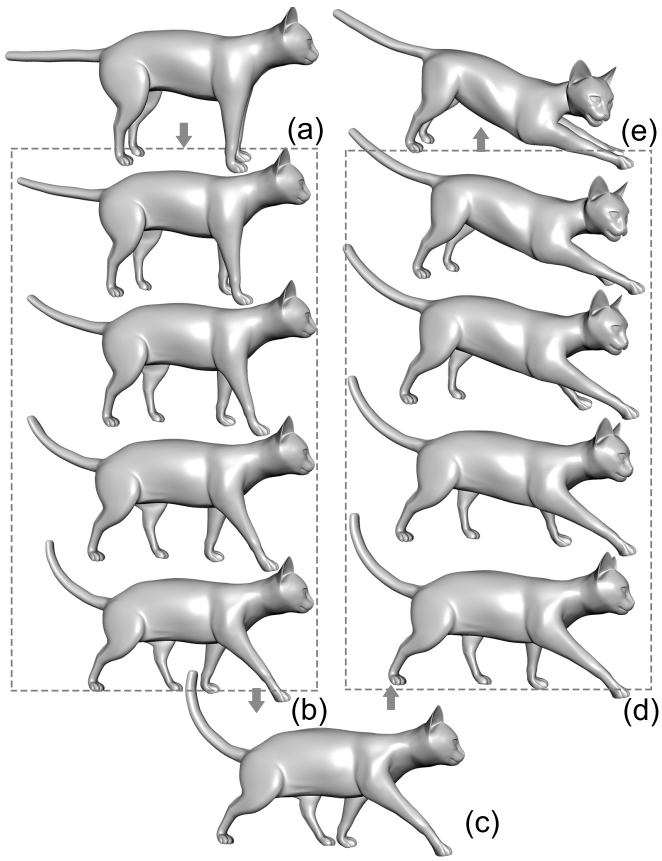


Figure 5.7: *(a), (c), and (e) are three input example poses, all the frames in (b) are in-between poses generated by the proposed approach based on (a) and (c), all the frames in (d) are in-between poses generated by the proposed approach based on (c) and (e).*

5.6 Evaluation

In this section, the proposed iso-parametric curve identification, skeleton creation, and ODE-based dynamic skin deformation are evaluated.

5.6.1 Evaluation of Iso-Parametric Curve Identification

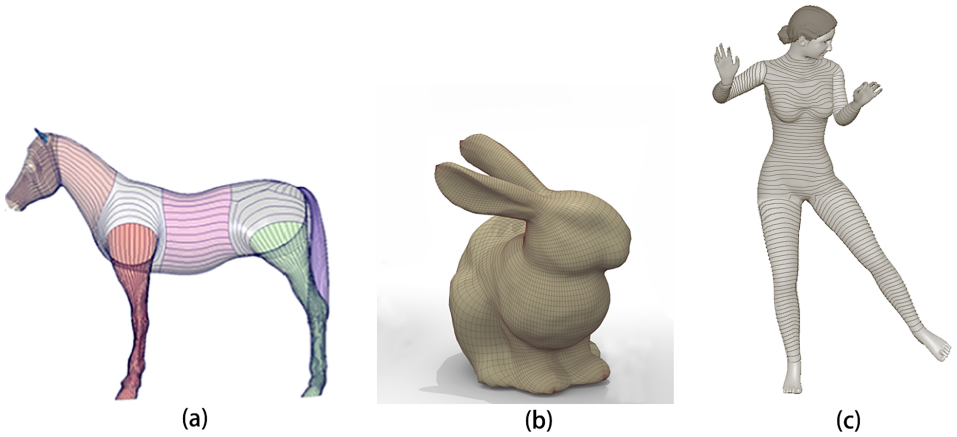


Figure 5.8: *Character models and extracted curves with different methods. (**a**) manual extraction Chaudhry et al. [2015], (**b**) skeleton-mesh co-representation Bærentzen et al. [2014], (**c**) iso-parametric curves identified by the proposed approach.*

There are few research studies which investigate identification or extraction of iso-parametric curves. In the existing curve-based geometric modeling and deformation simulation, iso-parametric curves were manually extracted. As said in Chaudhry et al. [2015], even a Maya Embedded language (MEL) script were used to assist the manual extraction of iso-parametric curves, the whole extraction of a horse model with 30,134 vertices shown in Figure 5.8 (a) took two hours.

The curves defining skin surfaces are manually extracted, leading to heavy and time-consuming human involvements. The method proposed in Bærentzen et al. [2014] can be used to extract iso-parametric curves from a triangular mesh. However, this method: (1) requires users to manually specify the extreme points of a harmonic function to guide the conversion, (2) its computational efficiency is less optimal. For example, it requires 6.41 s to process a bunny mesh with 6966 triangles shown in Figure 5.8 (b) into its polar-annular mesh representation using 100 slices on an off-the-shelf computer Bærentzen et al. [2014]. The proposed iso-parametric cure identification extracts a female model with a similar vertex number shown in Figure 5.8 (c) only took 2.03 milliseconds.

5.6.2 Evaluation of animation skeleton creation

In Chaudhry et al. [2015]; You et al. [2008], the animation skeleton used for calculating skin deformation is manually created. It is unsuitable for real-time animation or some applications that require high animation frame rates. In contrast, the approach proposed in this chaper can generate skeleton automatically.

As shown in Figure 5.9 (a), the mesh contraction method proposed in Au et al. [2008] takes about 10 s to obtain the skeleton of the raptor mesh with 25,000 vertices. The method proposed in this chaper takes less than 5 milliseconds to create the skeleton of the human mesh with 32,185 vertices shown in Figure 5.9 (b).

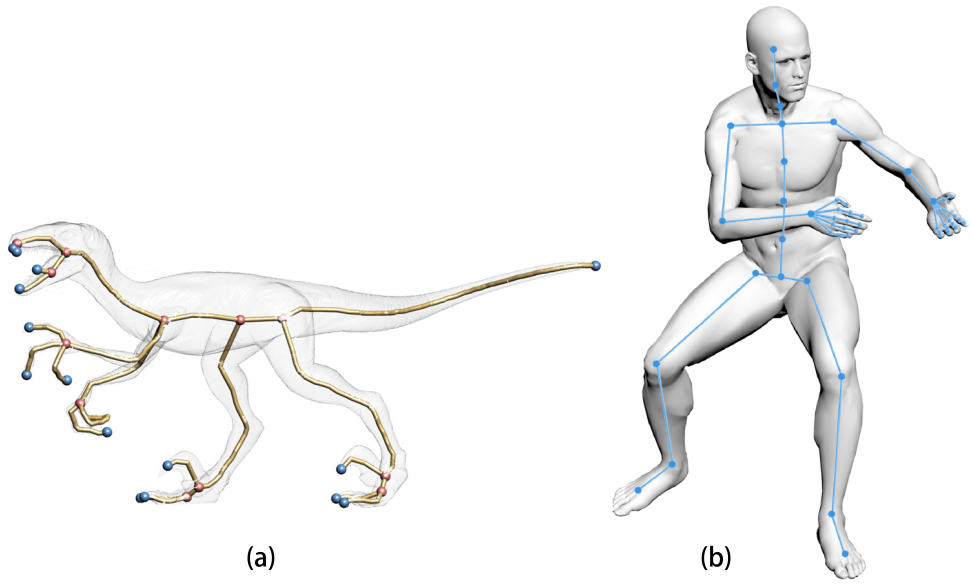


Figure 5.9: *Skeleton creation with different methods. (**a**) skeleton extraction by mesh contraction Au et al. [2008] , (**b**) skeleton generated by the proposed approach.*

5.6.3 Evaluation of ODE dynamic skin deformation

The runtime time shows the proposed approach is highly efficient while producing physically realistic skin deformations. Most existing physics-based skin deformation techniques work in a discrete vertex space to obtain discrete numerical solutions of skin deformation, causing the limita-

tions of intensive manual intervention, and heavy numerical calculations. The method in Kavan & Sorkine [2012] generates higher quality deformations than both linear and dual quaternion skinning through skinning weights optimization, but requires at least a few minutes to precompute the deformation weights. The method in McAdams et al. [2011] still needs at least several seconds for torso and arms simulation per frame on GPU, it is still not fast enough for interactive posing.

In contrast, this method only needs few milliseconds to automatically extract iso-parametric curves, generate skeleton and achieve dynamic in-between skin deformations, as shown in Table 5.1, giving a runtime breakdown of the proposed approach when it is applied to generate the animation of the dancer model in Fig. 5.6 and a cat model in Fig. 5.7. The runtime time statistics show this approach is highly efficient while producing physically realistic skin deformations.

Table 5.1: *Runtime breakdown of the proposed approach when it was used to automatically process the different models. Here, VI: Vertex Identifier on iso-parametric curves, SC: Skeleton Creating, SD: Skin Deformer.*

		Runtime (ms)			
Model	**Vertices**	**VI**	**SC**	**SD**	**Total**
Cat	7,207	1.32	0.269	1.517	3.106
Dancer	13,201	2.03	0.431	2.830	5.291

5.7 Discussion and Conclusion

This chapter presents an automatic rigging algorithm for ODE-based simulation of dynamic skin deformation for character animation. It includes vertex identification on iso-parametric curves, automatic skeleton creation combined with curve-based skin deformation approach. Several experiments have been conducted to demonstrate the developed approach can make the rigging process fast enough for highly efficient computer animation applications.

The method proposed in this chapter is easy to learn, implement,

and use. Among the three steps of the proposed approach, the isoparametric curve identification and automatic animation skeleton creation are straightforward and easiest to implement. Although the finite difference solution of ODE-based dynamic skin deformation is a little more difficult to implement, it is still much easier than the implementation of the corresponding finite element solution.

Several improvements can be made in the future. The proposed isoparametric curve identification is applicable to quad meshes with regular topology. Future work will investigate triangule meshes and arbitrary topologies. The proposed automatic rigging requires a template skeleton. In order to make the proposed automatic rigging applicable to various character models, different character models will be classified into different categories and a small template skeleton database with one skeleton in the database for one category of character models will be created.

Chapter 6

Application of Fourier Transformation in Physics-based Skin Deformation

Physics-based skin deformation methods can greatly improve the realism of character animation, but require non-trivial training, intensive manual intervention, and heavy numerical calculations. Due to these limitations, it is generally time-consuming to implement them, and difficult to achieve a high runtime efficiency. Recent model reduction efforts accelerate numerical calculations, providing an effective solution to achieving real-time performance at the expense of lowering computational accuracy and increasing implementation complexity. In order to tackle the above limitations caused by numerical calculations of physics-based skin deformation, a new, simple, and efficient analytical approach for physics-based skin deformations are proposed. Specifically, I: (1) employ Fourier series to convert 3D mesh models into continuous parametric representations through a conversion algorithm, which largely reduces data size and computing time but still keeps high realism, (2) introduce a Ordinary differential equation (ODE)-based skin deformation model, which is the first attempt to obtain an analytical solution to physics-based skin

deformations for highly efficient computation. This proposed approach is efficient, easy to use, and capable to create physically realistic skin deformations.

6.1 Introduction

Skin deformation techniques have been used as a standardized part of many character animation applications these days in both academia and industry practices. The usefulness of a skin deformation technique in terms of three major characteristics are generally measured: *efficiency*, *realism*, and *ease of use*. Over the years, researchers have developed a variety of skin deformation techniques to improve these aspects. These skin deformation techniques can be roughly categorized to geometric skin deformation, example-based skin deformation, and physics-based skin deformation, described below.

Geometric skin deformation approaches directly bind skin deformation with the underlying skeleton movement without taking any underlying physics into consideration. Despite their simplicity and efficiency, they typically fall short of producing highly realistic skin deformations without non-trivial additional efforts. ***Example-based skin deformation*** approaches interpolate a set of given example poses to improve realism or produce certain skin deformation effects that cannot be easily produced by geometric skinning approaches. Therefore, example-based skin deformation approaches are often used together with geometric skinning. In order to achieve high realism, a sufficient number of skin examples are often required. In addition, how to optimally design or acquire such a sufficient set of example poses is still considered as a widely open research problem. ***Physics-based skin deformation*** approaches introduce physics rules or musculoskeletal systems to drive and simulate the movement of the skin surface, which arguably have produced the most realistic skin deformation results to date. Although physics-based skin deformation approaches can generally lead to high realism, they rely on heavy numerical calculations, specialized knowledge

and skills; the requirements of numerical calculations including finite element simulations for a large memory footprint and high computational cost make them difficult to achieve a high efficiency. Model reduction can be used to reduce the data size or dimension and obtain an approximation of lower accuracy in significantly less time but more complicated implementation.

Although some recent research efforts have pursued the direction of integrating geometric, example-based, and physics-based skin deformation approaches, any approaches that are capable of describing physics-based skin deformation (static or dynamic) with analytical solutions have not been found, to the best of my knowledge. This poses an interesting open question: *Develop one skin deformation approach with analytical solution, which is capable of producing physically-realistic deformations with high efficiency (e.g., much faster than the numerical approaches employed by conventional physics-based skin deformations without model reduction), without requiring specialized knowledge, skills, and heavy manual involvements.*

Inspired by the above challenges, in this chapter a new analytical approach are proposed to generate physically-realistic character skin deformation in a highly efficient manner. This proposed method can be used only requiring at least two example meshes obtained from either captured or artist-sculpted shapes; any two adjacent examples constitute one sequence to create intermediate skin deformations (an example is shown in Figure 6.1). The advantages of this approach include: high efficiency, good realism, and ease of use, as detailed below.

High efficiency. With the obtained analytic solution to the formulated ODE-based physics model, this approach transforms a discrete example mesh into its continuous Fourier series representations to avoid the solving of a large set of linear equations, which largely reduces data size and computing time.

Good realism. Integrating physics-based and example-based approaches leads to more realistic skin deformations. It utilize a small number of provided example meshes in this physics-based mathemati-

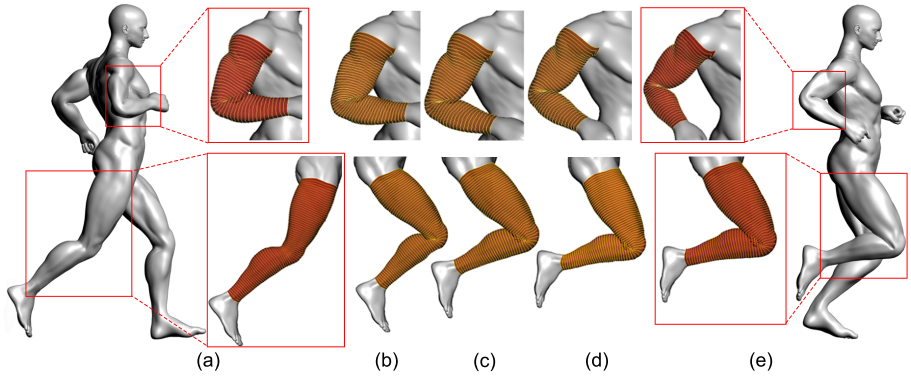


Figure 6.1: *This proposed method can generate physically realistic character animation efficiently given two example poses, shown in (a) and (e). (b)-(d) are intermediate poses generated by this proposed approach.*

cal model, in order to determine the correct force field acting on skin surfaces and achieve the realism of skin deformations.

Explicit solver. The solution of physics-based skin deformations is fully analytical and able to explicitly and quickly determine physically realistic skin deformations with high automation. Due to the fully analytical nature and few manual operations, it is easy to learn, implement, and use without much physics-based knowledge and skills.

6.2 Approach Overview

Most existing physics-based skin deformation techniques work in a discrete vertex space to obtain discrete numerical solutions of skin deformation, causing the aforementioned limitations. Analytical solutions, if obtainable, can overcome the limitations of numerical calculations of physics-based methods. However, obtaining an efficient analytical solution to physics-based skin deformation is a non-trivial task. Currently, no analytical solution is available to create physically-realistic skin deformation.

Most skin surfaces are polygon meshes defined by many discrete vertices. Unlike numerical solutions that are essentially discrete representations, analytical solutions are continuous representations. In order

to develop analytical solutions, discrete polygon models must be firstly transformed into continuous surfaces. Two mathematical representations can be used to define continuous surfaces: parametric and implicit surfaces. Since parametric surfaces are easier to develop analytical solutions than implicit surfaces, I select parametric surfaces to describe 3D skin surfaces.

Parametric skin surfaces can be mathematically written as $\mathbf{S}(u, v)$, where u and v are two parametric variables. If $\bar{\mathbf{S}}_0(u, v)$ and $\mathbf{S}_1(u, v)$ are used to stand for the meshes at any two adjacent poses called starting pose and ending pose, respectively, the shape change from $\bar{\mathbf{S}}_0(u, v)$ to $\mathbf{S}_1(u, v)$ can be decomposed into two parts: *geometric shape change* that is resulted from affine transformations, and *physical deformation* that is due to the force field acting on the skin surfaces.

Physics-based skin deformation approaches determine how skin deformations change with external forces. When external forces are applied slowly, skin deformations can be treated as static (called static skin deformations), which does not consider the effects of acceleration (inertial forces) and velocity (damping forces) on skin shape changes. Otherwise, skin deformations can be treated as dynamic (called dynamic skin deformations) to include the influences of acceleration and velocity.

Introducing a time variable t, dynamic *physical deformation* is written as $\mathbf{D}(u, v, t)$ in Fourier series representation, where a time-dependent force field $\mathbf{F}(u, v, t)$ is applied onto the skin surface, through the underlying physics described by differential equations. Assuming a differential operator $\mathbf{L}$ stands for the operations of the differential equations, the relationship between dynamic skin deformation and the force field can be described as $\mathbf{L}(\mathbf{D}(u, v, t)) = \mathbf{F}(u, v, t)$, whose concrete form will be presented in Section 6.4. Above differential equation involves three variables u, v and t. Simultaneously solving the three variables is difficult. However, if u are fixed, i. e., $u = u_i$, so can transform it into $\mathbf{L}(\mathbf{D}(u_i, v, t)) = \mathbf{F}(u_i, v, t)$. The remaining problem is how to efficiently solve this reduced differential equation. Fourier series developed by the French mathematician Joseph Fourier provides a powerful tool for devel-

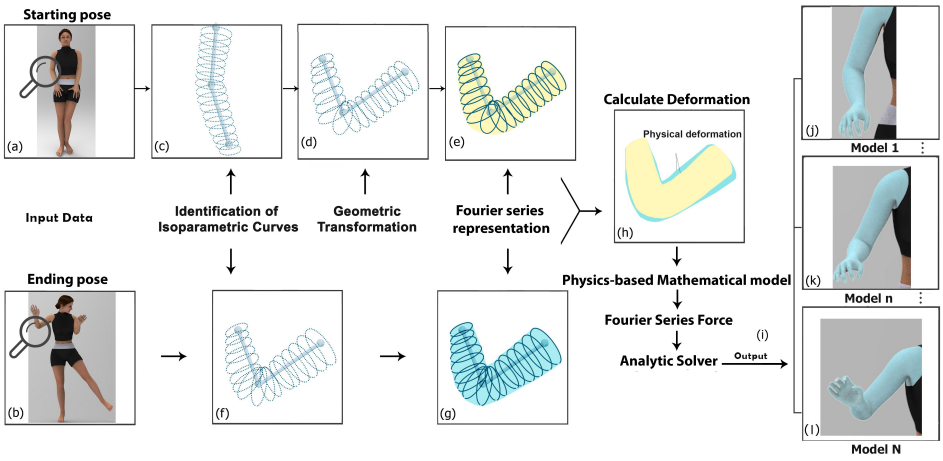


Figure 6.2: *The pipeline of the proposed method consists of two parts: Fourier series conversion, and physics-based skin deformation.*

oping an analytical solution to this differential equation.

Inspired by the above analysis, this proposed approach consists of two parts: the first part, **Fourier series conversion** (Section 6.3), mainly including *identification of vertices on isoparametric curves* and *Fourier series representation*. The second part, **physics-based skin deformation** (Section 6.4), mainly including *mathematical model of physics-based skin deformation* and *analytical solver*, which derives a closed form analytical solution to physics-based skin deformation. The pipeline of the proposed approach is illustrated in Figure 6.2 as below

In the **Fourier series conversion** part, first, two example meshes $\bar{\mathbf{S}}_0(u, v)$ in Figure 6.2(a) and $\mathbf{S}_1(u, v)$ in Figure 6.2(b) are inputted to the proposed system, which quickly identifies the vertices at the isoparametric curves $u = u_i$ and transforms $\bar{\mathbf{S}}_0(u, v)$ and $\mathbf{S}_1(u, v)$ into a series of isoparametric curves $\bar{\mathbf{S}}_0(u_i, v)$ in Figure 6.2(c) and $\mathbf{S}_1(u_i, v)$ in Figure 6.2(f). After that, the LBS is introduced to change $\bar{\mathbf{S}}_0(u_i, v)$ into a new shape $\mathbf{S}_0(u_i, v)$ as shown in Figure 6.2(d), which together with $\mathbf{S}_1(u_i, v)$ are transformed into continuous Fourier series representations indicated in Figure 6.2(e) and Figure 6.2(g), and are inputted into a deformation calculator to obtain the physical deformation shown in Figure 6.2(h). In the **Physics-based skin deformation** part, as shown in Figure 6.2(i), a physics-based mathematical model is first developed

for both dynamic and static skin deformations. Then, the force field causing skin deformation is represented with Fourier series. After that, the mathematical model is solved analytically and a closed form analytical solution is obtained. Finally, the obtained analytic solution is used to create skin deformations at any poses Figure 6.2(j)-(l) between the starting pose and the ending pose.

6.3 Fourier Series Conversion

The integration of geometric, example-based, and physics-based skin deformations includes the following steps: skeleton extraction, weights calculation, and physics-based deformation, maybe leading to manual interventions and non-negligible amount of preprocessing time. The first part of the proposed method, Fourier series conversion, is designed and tailored to avoid unnecessary manual operations, reduce the preprocessing time, and fast obtain the Fourier series representations that are needed for follow-up physics-based deformation determination.

6.3.1 Identification of Isoparametric Curves

The identification of isoparametric curves includes: i) automatic segmentation of skin surface models (decomposing a mesh into parts such as limbs and torso), ii) determination of the intersecting curves of the starting pose, and iii) extracting the isoparametric curves $\{u = u_i\}$ according to intersecting curves on the starting pose, further be used to determine their counterparts on the ending pose.

Automatic segmentation. Many existing algorithms can be used to automatically segment 3D mesh models. A comprehensive literature review on mesh segmentation algorithms can be found in Shamir [2008]. Quantitative evaluation of mesh segmentation algorithms was also reported in Chen et al. [2009]. Due to the reported efficiency of the random walks method Lai et al. [2008], it is employed to automatically segment 3D models (specifically, 3D character models in this work). The inter-

secting curves between two adjacent segmented parts are called *seam-curves*. After segmenting a 3D model into parts, including segmentation with one seam-curve (highlighted in green in Figure 6.3(b)), two seam-curves (illustrated in Figure 6.3(c), (d)), or more than two seam-curves (illustrated in Figure 6.3(e)).

Determination of intersecting curves. The determination of intersecting curves on one part model can be divided into cases with one seam-curve, two seam-curves, and more than two seam-curves.

For the case of one seam-curve on a part model (illustrated in Figure 6.3(b)), the centroid P (blue) of the seam-curve are firstly calculated. From the centroid and the normal of the plane closest to the seam-curve, a step size are specifiedto determine another point. Then, a new intersecting plane is created that passes through the point and is perpendicular to the normal, use it to intersect the part model, and obtain an intersecting curve. A new centroid is determined from the intersecting curve. The direction determined by the two adjacent centroids is taken to be the direction of the next step. This process repeats until arriving at the farthest end of the part model.

For the case of two seam-curves on a part model (Figure 6.3(c)), a dichotomy algorithm is used to obtain intersecting curves. For the first dichotomy, shown as the left arm of Figure 6.3(c), the centroid P and P' of the two green seam-curves is determined, and find the middle point P_1 of the line segment PP'. From the red middle point P_1, an intersecting plane T_1 perpendicular to the line PP' is created to intersect the part model, which leads to an intersecting curve (blue). Then, the black centroid P'_1 of the intersecting curve are calculated. For the second dichotomy, the black centroid P'_1 with P and P' are connected, calculate the red middle points P_2 of the line segments P'_1P and P'_1P', create two planes T_2 to intersect the model, and obtain two intersecting curves (blue) and calculate two corresponding centroid P'_2 of the blue curves. As shown in the second arm of Figure 6.3(c), for the third dichotomy, the three black centroids and the centers of green seam-curves PP'_2, $P'_2P'_1$, $P'_1P'_2$, and P'_2P' are continuously connected to obtain four line segments,

create four intersecting planes T_3 from the middle red points P_3 of the four line segments to intersect the part model, and obtain four blue intersecting curves and four black centroids P_3'. For the i-th dichotomy, $2^{(i-1)}$ centroid (black) can be obtained. Repeat the iteration until reaching a specified number of iterations or the threshold of the Euclidean distances between red center points and the black centroid. At last, all the black centroids after the i-th dichotomy are connected to generate a new curve skeleton shown as the last image of Figure 6.3(c), use it to create new intersecting planes (green), which is more sensitive with character surface shape, then obtain new intersecting curves. Figure 6.3(d) demonstrates application of the above dichotomy algorithm in a more irregular model segmentation.

In the situation of more than two seam-curves on a part model as indicated in Figure 6.3(e), each centroid of green seam-curves is calculated, and separate them into two clusters using K-means clustering method. Among each cluster, the average value of these seam-curve centers are computed, named as P_6 and P_7. After this treatment, this case can be transfered into two seam-curves problem (based on P_6 and P_7), and use the above dichotomy algorithm to obtain intersecting curves.

With the above three algorithms of determining intersecting curves, the number of intersecting curves can be easily controlled by taking different step sizes and effectively deal with irregular meshes with high frequency details and/or noise.

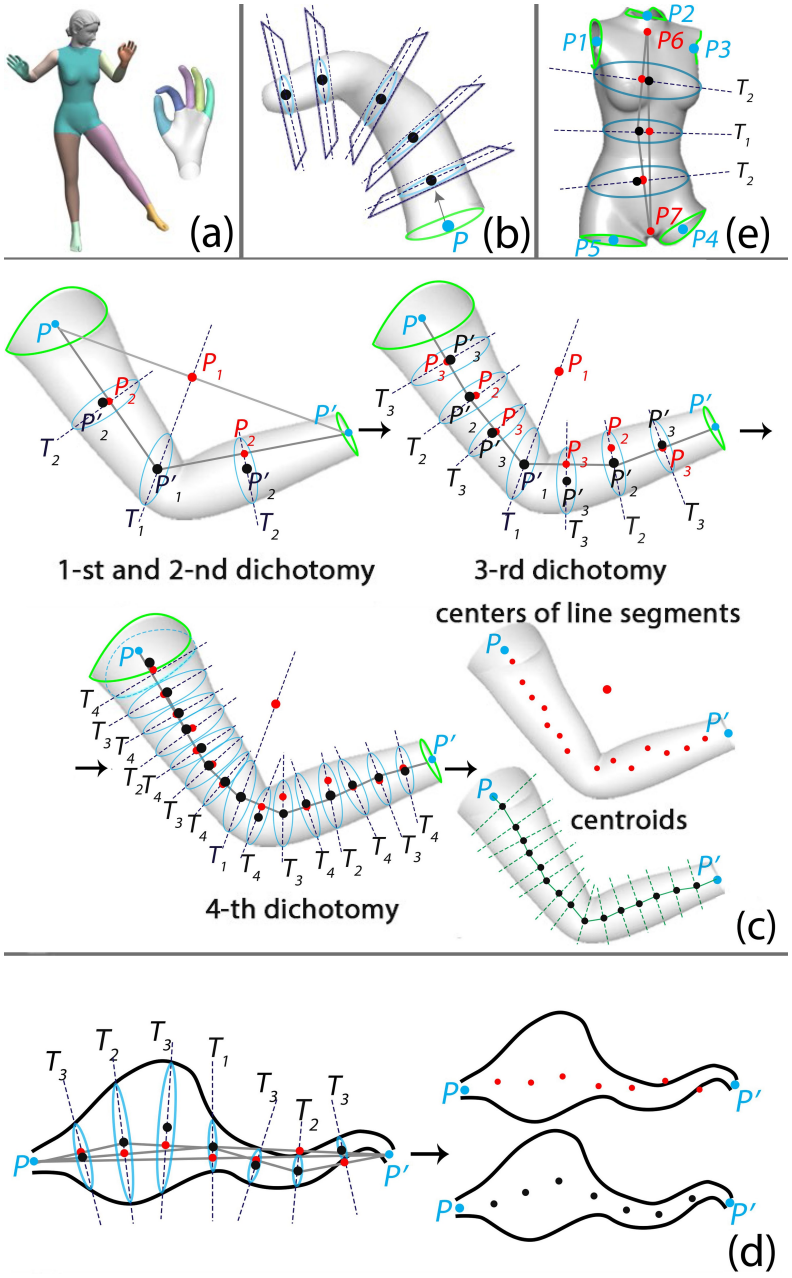


Figure 6.3: *Determination of intersecting curves: (a) segmentation of character models into parts, (b) determination of intersecting curves for parts with one seam-curve, (c) determination of intersecting curves for parts with two seam-curves, (d) determination of intersecting curves for irregular parts with two seam-curves, (e) determination of intersecting curves for irregular parts with more than two seam-curves. T_i means the intersecting plans when $i-th$ dichotomy, the red points means centers of lines, and the black points means centroid of intersecting curves.*

Extraction of isoparametric curves. The blue intersecting curve in Figure 6.4(a) is obtained by the aforementioned *Determination of in-*

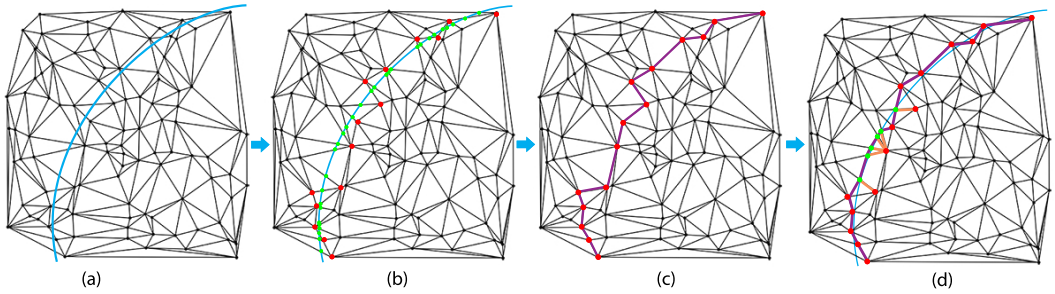


Figure 6.4: *Illustration of Extracting isoparametric curves: (a) shows one mesh intersected with one plane and get the intersected curve (blue). Green points in (b) are the intersected points, and the red vertex on each mesh edge is the edge end having smaller distance with green intersected points. Connecting these red ends of edges, isoparametric curve (purple) is extracted shown as (c). Improve the isoparametric curve by employing distance threshold, if distance between intersected point and ends (orange) are larger than threshold, keep the intersected point (green), more regular isoparametric curve (purple) can be obtained shown as (d).*

tersecting curves, consists of the green points in Figure 6.4(b) which are intersecting points generated by the intersecting plane and the edges of triangle facets. For each edge which is intersected by the intersecting plane, the distances from the intersecting point to the two end points of the edge are calculated, and take the end point with a smaller distance as an extracted vertex. Repeating this operation, the red vertices in Figure 6.4(b) are extracted and connect them together to get a purple isoparametric curve depicted in Figure 6.4(c). As can be seen from Figure 6.4(c), the extracted purple isoparametric curve is irregular when meshes are not regular. In order to obtain regular isoparametric curves, a threshold is introduced to discard some extracted vertices but keep green intersecting points as follows.

The distances from each of the extracted vertices to the intersecting curve are firstly calculated. If the distance is larger than the threshold, the extracted vertex is discarded. The intersecting point(s) on the intersecting curve (green) are kept to generate the new isoparametric curve (purple), shown as Figure 6.4(d).

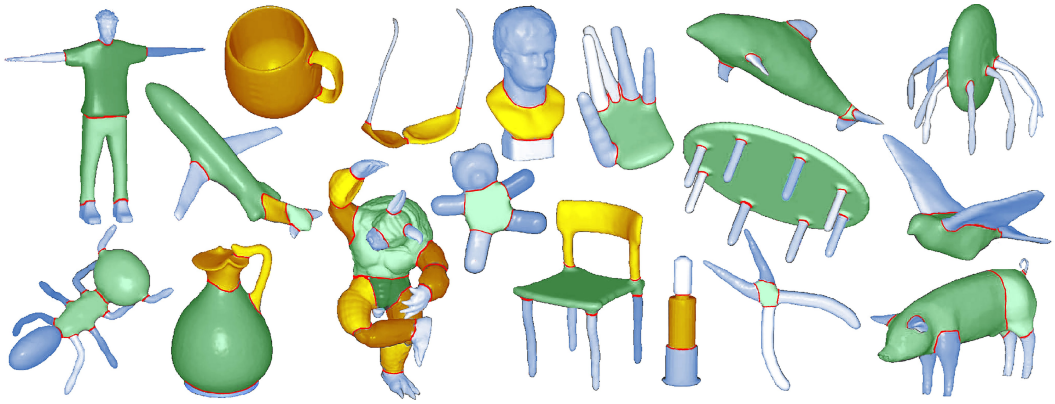


Figure 6.5: *Segmentation results of representative models in the Princeton Segmentation Benchmark Chen et al. [2009]. The segmentations could be addressed by the proposed approach according to three types, blue portions denote segmentations with one seam-curve, the yellow denotes segmentations with two seam-curves, and the green denotes segmentations with more than two seam-curves. It can be observed that the proposed approach could be employed for general models.*

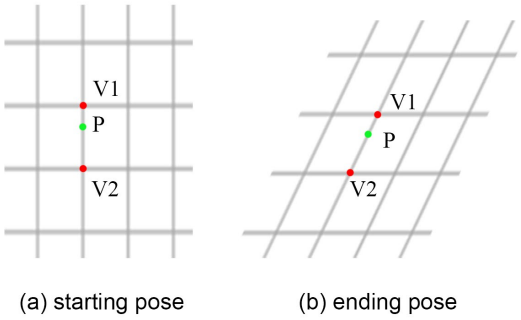


Figure 6.6: *Determine the isoparametric curves of the starting pose, and use the ratio to determine the position of the intersecting point on the ending example.*

The proposed method can be applied to various models, illustrated in Figure 6.5. Only need to determine the isoparametric curves of the starting pose, shown in Figure 6.6. Afterward, the corresponding isoparametric curves on other meshes can be automatically determined by the corresponding vertices with the same indexes. For any intersecting point P, find two vertices V_1 and V_2 which are on the same edge as the point P on the starting pose, shown as Figure 6.6(a). Then calculate the ratio of PV_1/PV_2, and use the ratio to determine the position of the intersecting point P on the ending example, shown as Figure 6.6(b).

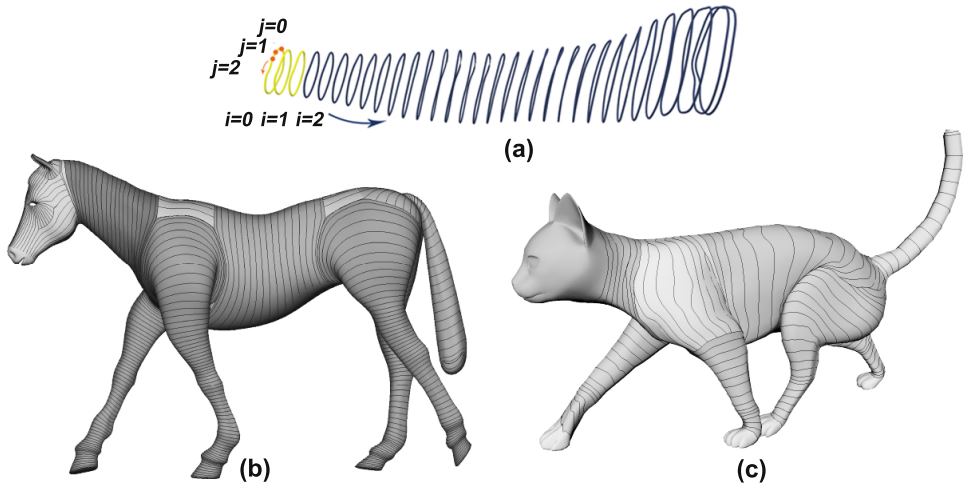


Figure 6.7: *The isoparametric curve identification algorithm transforms a mesh into discrete points on isoparametric curves* $\bar{\mathbf{S}}_0(u_i, v)(i = 0, 1, 2, ...I)$ *and the discrete points on the i-th isoparametric curve are numbered as* $\bar{\mathbf{S}}_0(u_i, v_j)(i = 0, 1, 2, ...I)(J = 0, 1, 2, ...J)$ *in an arm model (a), a horse model (b), amd a cat model (c).*

Figure 6.7 shows the identified isoparametric curves of a human arm (a), a horse (b) and a cat (c). The computational times for identifying isoparametric curves are given in Table 6.3.

The method proposed in Bærentzen et al. [2014] can be used to extract isoparametric curves from a triangular mesh. However, this method: 1) requires users to manually specify the extreme points of a harmonic function to guide the conversion, 2) its computational efficiency is less optimal. For example, it requires 6.21 seconds to process a mesh with 6,966 triangles into its polar-annular mesh representation using 100 slices on an off-the-shelf computer Bærentzen et al. [2014]. Therefore, above approach is proposed to better serve this specific aim for Fourier Series Representation.

6.3.2 Geometric Transformation

As discussed in the follow-up section, this physics-based skinning model does not include affine transformation such as rotation and translation, these transformations must be excluded firstly. Although various skin-

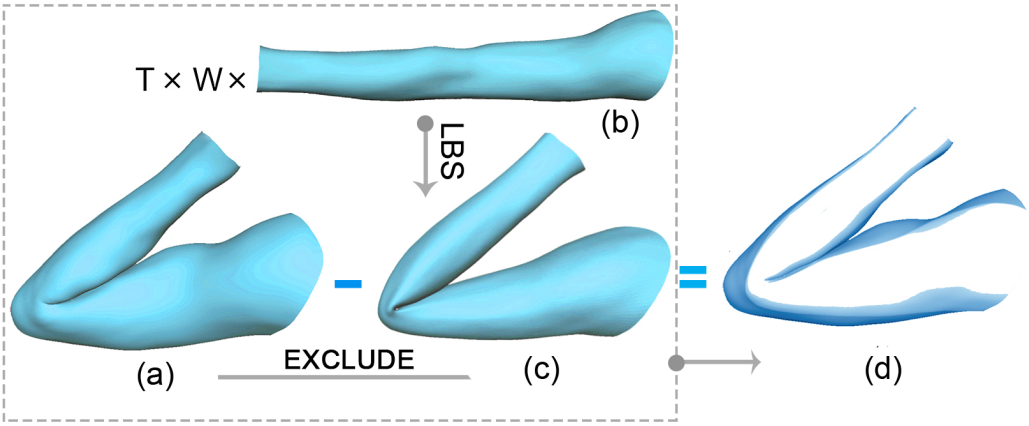


Figure 6.8: *Removal of transformations between two example poses. With the LBS model, transform the starting pose (b) to obtain the transformed mesh in (c). The purely geometric transformations are excluded by calculating the difference between the mesh of the ending pose (a) and the transformed mesh (c), as shown in (d).*

ning methods such as LBS, PSD (Pose Space Deformation), and dual quaternion can be used to achieve this goal, LBS is chosen to exclude these affine transformations due to its efficiency. The identified vertices $\bar{\mathbf{S}}_0(u_i, v_j)$ at the starting pose are transformed to new positions $\mathbf{S}_0(u_i, v_j)$ at the ending pose through a linear combination of bone transformation matrices T_m as follows.

$$\mathbf{S}_0(u_i, v_j) = (\sum_{m=1}^{M} w_{ij,m} T_m) \bar{S}_0(u_i, v_j) \tag{6.1}$$

where $w_{ij,m}$ are the blending weights, and its subscript indicates the j-th vertex on the isoparametric curve at u_i.

Since two example poses are given, the transformation matrices T_m are straightforwardly fitted by transforming the animation skeleton at the starting pose to fit the animation skeleton at the ending pose. Assuming the skeletons have been given and among various weights calculation methods, Euclidean distance weights usually serve as initial guess for further weights refinement; Bounded Biharmonic Weights (BBW) Jacobson et al. [2011] are smooth, localized and shape-aware but slower and need a volume discretization; the extra scalar weight in Jacobson & Sorkine

[2011] provides a way to determine points how much they could stretch and twist while need to paint weights manually or compute automatically using methods in Jacobson et al. [2011]; Baran & Popović [2007]. In comparison, the method Yang & Zhang [2006] is much simpler and more efficient, maximizing high computational efficiency of the analytical solution of physics-based skin deformation; thus, use it to calculate $w_{ij,m}$ of above Eq. 6.1, and illustrate this computing process in Figure 6.8.

6.3.3 Fourier Series Representation

In order to obtain an analytical solution to physics-based skin deformations, this method convert example character meshes consisting of discrete vertices $S_k(u_i, v_j)$(k=0,1) into continuous Fourier series representations $\mathbf{C}_k^F(u_i, v)$(k=0,1) as below.

$$\mathbf{C}_k^F(u_i, v) = \mathbf{c}_0^k(u_i) + \sum_{n=1}^{N} \left[\mathbf{c}_{2n-1}^k(u_i)\cos 2n\pi v + \mathbf{c}_{2n}^k(u_i)\sin 2n\pi v\right] \quad (k = 0, 1) \tag{6.2}$$

where $\mathbf{C}_k^F(u_i, v)(k = 0, 1)$ are continuous Fourier series representations of example models, $k = 0$ denotes the starting pose, $k = 1$ denotes the ending pose, the superscript F indicates Fourier series, u_i denotes the i-th isoparametric curve in each pose, v is a parametric variable in the range of [0, 1], $\mathbf{c}_n^k(u_i)(k = 0, 1; n = 0, 1, ..., N)$ are Fourier coefficients, and the subscript n denotes the n-th term of the Fourier series.

In order to determine the unknown Fourier coefficients $\mathbf{c}_n^k(u_i)$, first discuss how to relate the parametric variable v in Eq. 6.2 to the points on isoparametric curves. Assuming that the length of the isoparametric curves $S_k(u_i, v_j)$ (k=0, 1) from the point $j = 0$ to the points $j = 1, 2, ..., J$ is L_{ij}^k, and the total length of the isoparametric curves from the point $j = 0$ to the point $j = J$ is L_{iJ}^k where the subscript J indicates the last point on the isoparametric curves, the value v_{ij}^k of the parametric variable v at the points $S_k(u_i, v_j)$(k=0, 1) is determined by the ratio

between the length L_{ij}^k and the total length L_{iJ}^k, i. e.,

$$v_{i0}^k = 0 \quad v_{ij}^k = L_{ij}^k / L_{iJ}^k (j = 1, 2, 3, ..., J), \tag{6.3}$$

where

$$L_{im}^k = \sum_{l=1}^{m} d(\mathbf{S}_k(u_i, v_l), \mathbf{S}_k(u_i, v_{l-1})),$$
$$(k = 0, 1; i = 0, 1, 2, ..., I; m = i, J; j = 1, 2, ..., J) \tag{6.4}$$

where d is the Euclidean distance.

After relating the parametric variable v in Eq. 6.2 to the points on the isoparametric curves, the points $\mathbf{C}_k^F(u_i, v_j)$ given by the Fourier series representations corresponds to the points $\mathbf{S}_k(u_i, v_j)$ on the isoparametric curves are known. The error caused by the Fourier series representations can be quantified by the squared sum of all the Euclidean distances between the points$\mathbf{C}_k^F(u_i, v_j)$ and the points $\mathbf{S}_k(u_i, v_j)$ with the following equation:

$$\mathbf{E}_k = \sum_{j=0}^{J-1} [d(\mathbf{C}_k^F(u_i, v_j), \mathbf{S}_k(u_i, v_j)]^2 \tag{6.5}$$

Here, $\mathbf{E}_k$ denotes the errors which need to be minimized by least square method to obtain the $\mathbf{C}_k^F(u_i, v_j)$, the errors given in Eq. 6.5 can be minimized by differentiating the above equations with respect to $\mathbf{c}_n^k(u_i)(k = 0, 1; n = 0, 1, 2, 3, ..., 2n)$, respectively, and zeroing the derivatives, i. e., and where

$$\mathbf{C}_k^F(u_i, v_j) = \mathbf{c}_0^k(u_i) +$$
$$\sum_{n=1}^{N} \left[\mathbf{c}_{2n-1}^k(u_i) \cos 2n\pi v_j + \mathbf{c}_{2n}^k(u_i) \sin 2n\pi v_j\right] \tag{6.6}$$

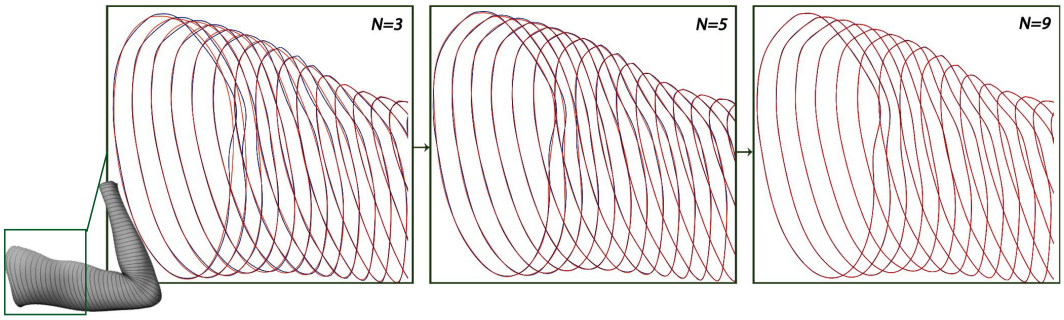


Figure 6.9: *Comparison between original isoparametric curves (blue) and Fourier series curves (red) (a), when different Fourier series terms are employed, such as N = 3, 5, 9, the Fourier series representation (red) from Eq.2 are increasingly approach to the isoparametric curves (blue) of one scanned arm. Even N is set as 9, Fourier series representation approximate isoparametric curves as the same.*

Thus, minimize Eq. 6.5 over the Fourier coefficients to determine the unknown Fourier coefficients $\mathbf{c}_n^k(u_i)$,

The above Fourier series conversion algorithm is applicable to both closed and open isoparametric curves. For closed isoparametric curves, $\mathbf{S}_k(u_i, v_J) = \mathbf{S}_k(u_i, v_0)$ and Eq. 6.5 can be used directly. For open isoparametric curves, can use $\mathbf{C}_k^F(u_i, v_0) = \mathbf{S}_k(u_i, v_0)$ and $\mathbf{C}_k^F(u_i, v_J) = \mathbf{S}_k(u_i, v_J)$ to achieve the positional continuity at the two end points and determine two vector-valued unknown Fourier coefficients in Eq. 6.2. If tangential or higher continuities at the two end points are required, the continuity conditions of the first or higher derivatives are introduced to determine more unknown constants in Eq. 6.2, then change j in Eq. 6.5 starting from 0 into $J-1$ to determine the remaining unknown constants.

The proposed Fourier series representations effectively describe local details well, shown in Figure 6.9, a comparison between the original isoparametric curves (blue) of a human arm and the corresponding Fourier series curves (red) with different terms N. It is evident that even if a small number of terms such as N=3 is taken, the Fourier series reconstruction approaches the original isoparametric curves very closely.

6.3.4 Deformation Calculation

The deformations from the starting pose to the ending pose after excluding purely geometric transformations are the difference between the transformed starting pose $\mathbf{C}_0^F(u_i, v)$ and the ending pose $\mathbf{C}_1^F(u_i, v)$, described below.

$$\widetilde{\mathbf{D}}(u_i, v) = \mathbf{C}_1^F(u_i, v) - \mathbf{C}_0^F(u_i, v) \tag{6.7}$$

Substituting Eq. 6.2 into Eq. 6.7 and putting the same coefficients together, the skin deformations $\tilde{\mathbf{D}}(u_i, v)$ can be rewritten as the following Fourier series:

$$\widetilde{\mathbf{D}}(u_i, v) = \mathbf{d}_0(u_i) + \sum_{n=1}^{N} [\mathbf{d}_{2n-1}(u_i) \cos 2n\pi v \ + \mathbf{d}_{2n}(u_i) \sin 2n\pi v] \tag{6.8}$$

$$\mathbf{d}_n(u_i) = \mathbf{c}_n^1(u_i) - \mathbf{c}_n^0(u_i) (n = 0, 1, 2, 3, \cdots, 2N) \tag{6.9}$$

where unknown constants $\mathbf{c}_n^0(u_i)$ and $\mathbf{c}_n^1(u_i)$ are obtained in Section 6.3.3.

6.4 Physics-based Skin deformation

Physics-based skin deformation determines how skin shapes change with external forces (or displacements). In order to tackle both dynamic and static skin deformations, the physics-based skin deformation model should include the contributions of acceleration, velocity, and static skin deformation. In this section, a physics-based skin deformation model to describe how skin surfaces change their shapes under the action of a time-dependent force field will be proposed. Since the deformation between the ending pose and starting pose has been represented in the Fourier space through Eq. 6.8, and also represent the force field in the same

Fourier space. Finally, an analytical solution of the model is developed to create realistic skin deformation efficiently.

6.4.1 Mathematical Model of Physics-based Skin Deformation

Similar to the treatment proposed by James & Pai [2002], the velocity and acceleration have an effect on the shape change of skin surfaces. This effect is formulated by using the vector-valued motion equation below.

$$(\rho\partial^2/\partial t^2 + \eta\partial/\partial t + \Re)\mathbf{D}(u,v,t) = \mathbf{F}(u,v,t) \tag{6.10}$$

where ρ and η are the density and damping coefficient of skin surfaces, and the corresponding terms describe the effects of acceleration and velocity on the shape variations of skin surfaces, $\Re$ is an internal deformation force which is the function of skin deformations $\mathbf{D}(u,v,t)$, and $\mathbf{F}(u_i,v,t)$ is a time-dependent force field driving the skin deformations.

Skin deformations are similar to the elastic bending of thin plates. Accordingly, the internal force for skin bending deformations is similar to the internal force of the elastic bending of thin plates, and it can be mathematically described by

$$\Re\mathbf{D}(u,v,t) = (\chi\partial^4/\partial u^4 + \gamma\partial^4/\partial u^2\partial v^2 + \xi\partial^4/\partial v^4)\mathbf{D}(u,v,t) \tag{6.11}$$

Here, χ, γ, ξ are called shape control parameters which control the shape change of skin surfaces. Apart from the internal force for bending deformations, another force which resists tensile or compressive deformations should also be considered. This force can be regarded as a restoring force. Like the deformation of a spring, the restoring force is proportional to the tensile or compressive deformations, which can be approximated as $\varsigma\mathbf{D}(u,v,t)$. From the above discussion, the internal deformation force $\Re$ should be the sum of the internal force for bending and the restor-

ing force for tensile or compressive deformations. Using them to replace $\Re$ in the linear elastodynamic equation Eq. 6.10 and introducing the differential operator $\mathbf{L}$ defined below.

$$\rho\partial^2/\partial t^2 + \eta\partial/\partial t + \chi\partial^4/\partial u^4 + \gamma\partial^4/\partial u^2\partial v^2 + \xi\partial^4/\partial v^4 + \varsigma = \mathbf{L} \quad (6.12)$$

The linear elastodynamic Eq. 6.10 is transformed into a vector-valued fourth order dynamic ordinary differential equation (ODE):

$$\mathbf{LD}(u, v, t) = \mathbf{F}(u, v, t) \quad (6.13)$$

As show in Figure 6.10, the forced vibrations include a transient part and a steady-state part. The transient part has an exponentially decreasing amplitude indicated by the blue curve in Figure 6.10. After sufficient time, the transient part is damped out, leaving the motion described by the steady-state part only indicated by the purple curve whose vibration frequency is totally determined by the frequency of the external forces Housner & Hudson [1980].

Since the transient part only occurs in a short time when the character motion is starting or stopping and the analytical solution to the transient part is more difficult to obtain, this work focus on the analytical solution to the steady-state part of the physics-based skin deformation model. In the future work, the proposed method will be extended to deal with the transient part. Besides, solving fourth order ODEs is very complicated to develop analytic solutions, numerical methods such as the finite element method and finite difference method have been used. These numerical methods are in general computationally costly. In order to reduce the computational complexity, this method propose to lower the fourth order dynamic ODE to the second order by using the following differential

operator:

$$\mathbf{L} = \rho\partial^2/\partial t^2 + \eta\partial/\partial t + \chi\partial^2/\partial u^2 + \gamma\partial^2/\partial u\partial v + \xi\partial^2/\partial v^2 + \varsigma \quad (6.14)$$

When considering the shape changes on the iso-parametric curve u_i, the parametric variable u becomes a constant. Accordingly, the deformation function $\mathbf{D}(u, v, t)$ and the force functions $\mathbf{F}(u, v, t)$ in Eq. 6.13 become $\mathbf{D}(u_i, v, t)$ and $\mathbf{F}(u_i, v, t)$, the partial derivatives of $\mathbf{D}(u, v, t)$ with respect to the parametric variable u drop, and the differential Eq. 6.13 becomes

$$\mathbf{LD}(u_i, v, t) = \mathbf{F}(u_i, v, t), \quad (6.15)$$

where

$$\mathbf{L} = \rho\partial^2/\partial t^2 + \eta\partial/\partial t + \xi\partial^2/\partial v^2 + \varsigma \quad (6.16)$$

It should be noted that lowering the dynamic ODE from the fourth order to the second order may reduce the prediction accuracy of skin deformations. However, this can be well compensated by introducing example skin shapes.

When two example meshes are known, the deformations $\mathbf{D}(u_i, v, t)$ can be set to zero at $t = 0$ and $\widetilde{\mathbf{D}}(u_i, v)$ at $t = 1$. If a mesh is motionless at $t = 0$, the deformation rate $\partial\mathbf{D}(u_i, v, t)/\partial t$ is zero. If a mesh is in motion at $t = 0$, the deformation rate $\partial\mathbf{D}(u_i, v, t)/\partial t$ should take the value of the deformation rate at the instant. If $\dot{\mathbf{D}}_0$ is used to indicate the deformation rate at $t = 0$, these constraint conditions can be formulated as

$$\begin{aligned} t = 0 \qquad & \mathbf{D}(u_i, v, t) = 0, \quad \partial\mathbf{D}(u_i, v, t)/\partial t = \dot{\mathbf{D}}_0 \\ t = 1 \qquad & \mathbf{D}(u_i, v, t) = \widetilde{\mathbf{D}}(u_i, v) \end{aligned} \quad (6.17)$$

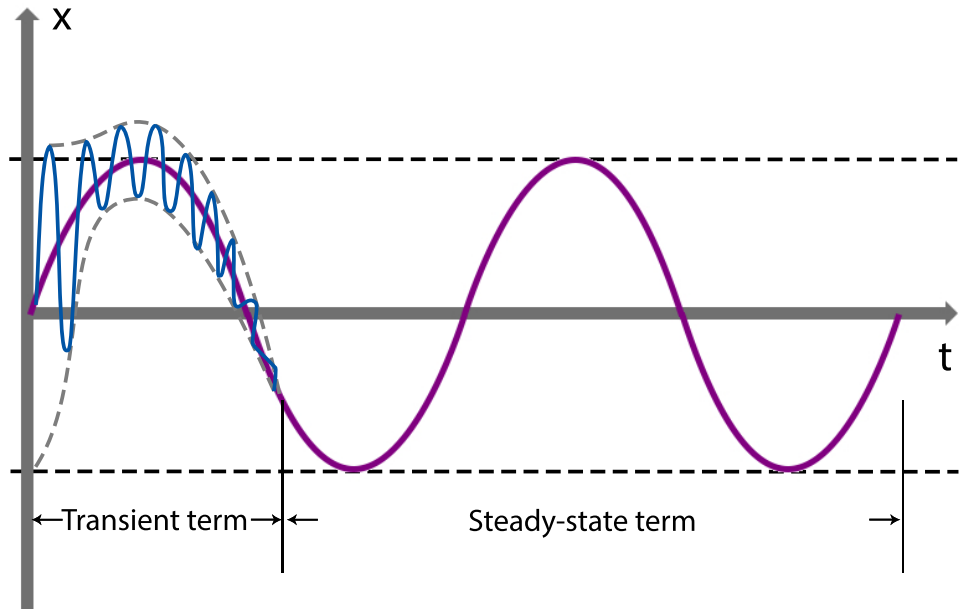


Figure 6.10: *Effects of damping and force on vibration Housner & Hudson [1980]*

After defining boundary conditions, the mathematical model is obtained by combining Eq. 6.15 with Eq. 6.17, which can be analytically solved and generate smooth deformation between two adjacent sequences as described below.

6.4.2 Fourier Series Force

The time-dependent force field $\mathbf{F}$ in $\mathbf{LD} = \mathbf{F}$ can be determined from the given example skin shapes. At the starting pose $t = 0$, there is no skin deformations. The external force at the starting pose should be zero: $\mathbf{F}(u_i, v, 0) = 0$. At the ending pose $t = 1$, the skin deformations is $\widetilde{\mathbf{D}}(u_i, v)$ determined by Eq. 6.8, which are represented with Fourier series of the parametric variable v. Naturally, the same Fourier series can be used to represent the external force at the ending pose, i. e.,

$$\widetilde{\mathbf{F}}(u_i, v) = \mathbf{e}_0(u_i) + \sum_{n=1}^{N} (\mathbf{e}_{2n-1}(u_i) \cos 2n\pi v + \mathbf{e}_{2n}(u_i) \sin 2n\pi v) \quad (6.18)$$

Here, the $\mathbf{e}_0(u_i)$, $\mathbf{e}_{2n-1}(u_i)$, $\mathbf{e}_{2n}(u_i)$ are unknown constants to be obtained by Analytic Solver in Section 6.4.3. From the known external forces at the starting and ending poses, the following linear variation of the force field can be obtained with respect to the time variable t.

$$\begin{aligned}\mathbf{F}(u_i, v, t) = t\widetilde{\mathbf{F}}(u_i, v) = \quad & t\mathbf{e}_0(u_i) + \sum_{n=1}^{N} (t\mathbf{e}_{2n-1}(u_i)\cos 2n\pi v \\ & + t\mathbf{e}_{2n}(u_i)\sin 2n\pi v) \end{aligned} \tag{6.19}$$

6.4.3 Analytic Solver

The mathematical model Eq. 6.15 can be used to deal with dynamic skin deformations including skin vibrations. The time-dependent force field represented by Eq. 6.19 involves three types of terms: t, $t\cos 2n\pi v$, and $t\sin 2n\pi v$. If it is introduced into Eq. 6.15, $\mathbf{D}(u_i, v, t)$ should also contain the three types of terms. If $\mathbf{D}_1(u_i, t)$, $\mathbf{D}_{2n}(u_i, v, t) = \mathbf{D}_2(t\cos 2n\pi v)$ and $\mathbf{D}_{3n}(u_i, v, t) = \mathbf{D}_3(t\sin 2n\pi v)$ are used to indicate the three types of terms, respectively, the deformation function $\mathbf{D}(u_i, v, t)$ can be rewritten as

$$\mathbf{D}(u_i, v, t) = \mathbf{D}_1(u_i, t) + \sum_{n=1}^{N} [\mathbf{D}_{2n}(u_i, v, t) + \mathbf{D}_{3n}(u_i, v, t)] \tag{6.20}$$

Substituting Eq. 6.20 into Eq. 6.15, the following equation can be obtained.

$$\mathbf{L_1}\mathbf{D_1}(u_i, t) + \sum_{n=1}^{N} \mathbf{L}_2[\mathbf{D}_{2n}(u_i, v, t) + \mathbf{D}_{3n}(u_i, v, t)] = \mathbf{F}(u_i, v, t), \tag{6.21}$$

where $\mathbf{L}_1$ and $\mathbf{L}_2$ are determined by

$$\mathbf{L}_1 = \rho\partial^2/\partial t^2 + \eta\partial/\partial t + \varsigma \qquad \mathbf{L}_2 = \mathbf{L}_1 + \xi\partial^2/\partial v^2 \tag{6.22}$$

Introducing Eq. (6.19) into Eq. (6.21) and comparing both sides of the equation, the following equations are obtained

$$\begin{aligned} \mathbf{L}_1\mathbf{D}_1(u_i,t) = &\quad t\mathbf{e}_0(u_i) \\ \mathbf{L}_2\mathbf{D}_{2n}(u_i,v,t) = &\quad t\mathbf{e}_{2n-1}(u_i)cos2n\pi \\ \mathbf{L}_2\mathbf{D}_{3n}(u_i,v,t) = &\quad t\mathbf{e}_{2n}(u_i)sin2n\pi \\ &\quad (n=1,2,3,...,N) \end{aligned} \tag{6.23}$$

The remaining details of analytically solving the homogenous general solutions and the particular solutions to Eq. 6.21 are given as following:

The homogenous general solution to each equation of Eq.(6.23) can be taken as $\widetilde{\mathbf{D}}_1(u_i,t) = e^{\mathbf{r}t}$,$\widetilde{\mathbf{D}}_2(u_i,v,t) = e^{\mathbf{q}t}\cos 2n\pi v$ and $\widetilde{\mathbf{D}}_3(u_i,v,t) = e^{\mathbf{s}t}\sin 2n\pi v$.

Substituting them into each of Eq.(6.23), and meet the constraints: at starting pose $t = 0$, the deformations determined by Eq. 6.20 should be zero; at the ending pose $t = 1$, the deformations determined by Eq. 6.20 should be the same as Eq. 6.9; and at the starting pose $t = 0$, the rate of deformation is $\partial D/\partial t = 0(t = 0)$. In order to solve the above constraints to get the homogenous general solution for $\mathbf{D}_1$, $\mathbf{D}_{2n}$ and $\mathbf{D}_{3n}$, the situations $4\rho\xi - \eta^2 > 0$ and $4(\xi - 4\zeta n^2\pi^2) - \eta^2 > 0$ should be met, and setting

$$\begin{aligned} \alpha_r = -\eta/(2\rho) \quad \beta_r = \alpha_r\sqrt{4\rho\zeta/\eta^2 - 1} \\ h_n = \zeta - 4n^2\pi^2\xi \quad \beta_n = \alpha_r\sqrt{4h_n/\eta^2 - 1} \end{aligned} \tag{6.24}$$

to obtain $r_{1,2} = \alpha_r + i\beta_r, \quad q_{n_{1,2}} = s_{n_{1,2}} = \alpha_r + i\beta_n$

and the homogenous general solutions below

$$\begin{aligned}
\widetilde{\mathbf{D}}_1(u_i,t) &= \mathbf{C}_1(u_i)f(t) + \mathbf{C}_2(u_i)\bar{f}(t) \\
\widetilde{\mathbf{D}}_{2n}(u_i,v,t) &= [\mathbf{C}_{3n}(u_i)g_n(t) + \mathbf{C}_{4n}(u_i)\bar{g}_n(t)]\cos 2n\pi v \\
\widetilde{\mathbf{D}}_{3n}(u_i,v,t) &= [\mathbf{C}_{5n}(u_i)g_n(t) + \mathbf{C}_{6n}(u_i)\bar{g}_n(t)]\sin 2n\pi v
\end{aligned} \tag{6.25}$$

where

$$\begin{aligned}
f(t) &= e^{\alpha_r t}\cos\beta_r t \quad \bar{f}(t) = e^{\alpha_r t}\sin\beta_r t \\
g_n(t) &= e^{\alpha_r t}\cos\beta_n t \quad \bar{g}_n(t) = e^{\alpha_r t}\sin\beta_n t
\end{aligned} \tag{6.26}$$

Then, the particular solutions to Eq.(6.23) can be taken to be $\bar{\mathbf{D}}_1(u_i,t) = a_1(u_i)+a_2(u_i)t$, $\bar{\mathbf{D}}_{2n}(u_i,v,t) = [a_3(u_i)+a_4(u_i)t]\cos 2n\pi v$ and $\bar{\mathbf{D}}_{3n}(u_i,v,t) = [a_5(u_i)+a_6(u_i)t]\cos 2n\pi v$. Also substituting them into Eq.(6.23), to obtain

$$\begin{aligned}
\bar{\mathbf{D}}_1(u_i,t) &= (\zeta t - \eta)e_0(u_i)/\zeta^2 \\
\bar{\mathbf{D}}_{2n}(u_i,v,t) &= (h_n t - \eta)e_{2n-1}(u_i)\cos 2n\pi v/h_n^2 \\
\bar{\mathbf{D}}_{3n}(u_i,v,t) &= (h_n t - \eta)e_{2n}(u_i)\sin 2n\pi v/h_n^2
\end{aligned} \tag{6.27}$$

Introducing Eq.(6.25) and Eq.(6.27) into Eq.(6.20), to obtain the function $\mathbf{D}(u_i,v,t)$ which involves the unknown constants $\mathbf{e}_0(u_i)$, $\mathbf{e}_{2n-1}(u_i)$, $\mathbf{e}_{2n}(u_i)$ and $\mathbf{C}_1(u_i)$, $\mathbf{C}_2(u_i)$ and $\mathbf{C}_{kn}(u_i)(k = 3,4,5,6)$. Using the boundary conditions Eq.(6.17), these unknown constants can be determined.

Then, substituting the obtained unknown constants back into $\mathbf{D}(u_i, v, t)$, and introducing

$$\begin{aligned}\delta_0 &= e^{\alpha_r}[-\eta\beta_r \cos\beta_r + \zeta(1+\eta\alpha_r/\zeta)\sin\beta_r] - \zeta\beta_r(1-\eta/\zeta) \\ \delta_n &= e^{\alpha_r}[-\eta\beta_n \cos\beta_n + (1+\eta\alpha_r)\sin\beta_n] - \beta_n(1-\eta/h_n)\end{aligned} \tag{6.28}$$

and

$$\begin{aligned}\gamma_0(t) &= [-\eta\beta_r f(t) + \zeta(1+\eta\alpha_r/\zeta)\bar{f}(t) - \zeta\beta_r(t-\eta/\zeta)]/\delta_0 \\ \gamma_n(t) &= [-\eta\beta_n g_n(t) + (1+\eta\alpha_r)\bar{g_n}(t) - \beta_n(t-\eta/h_n)]/\delta_n\end{aligned} \tag{6.29}$$

Eq.(6.30) is obtained.

To the end, the obtained analytical solution is as follows:

$$\begin{aligned}\mathbf{D}(u_i, v, t) = \quad & \gamma_0(t)d_0(u_i) + \sum_{n=1}^{N} \gamma_n(t)[d_{2n-1}(u_i)cos2n\pi v \\ & + d_{2n}(u_i)sin2n\pi v]\end{aligned} \tag{6.30}$$

The above Eq. 6.30 indicates that once the skin deformations $\mathbf{d}_n(u_i)(n = 0, 1, 2, ..., 2N)$, determined from the skin shapes at the starting pose $\mathbf{c}_n^0(u_i)$ and the ending pose $\mathbf{c}_n^1(u_i)$, are known, the skin deformation at any in-between poses can be analytically determined by Eq. 6.30.

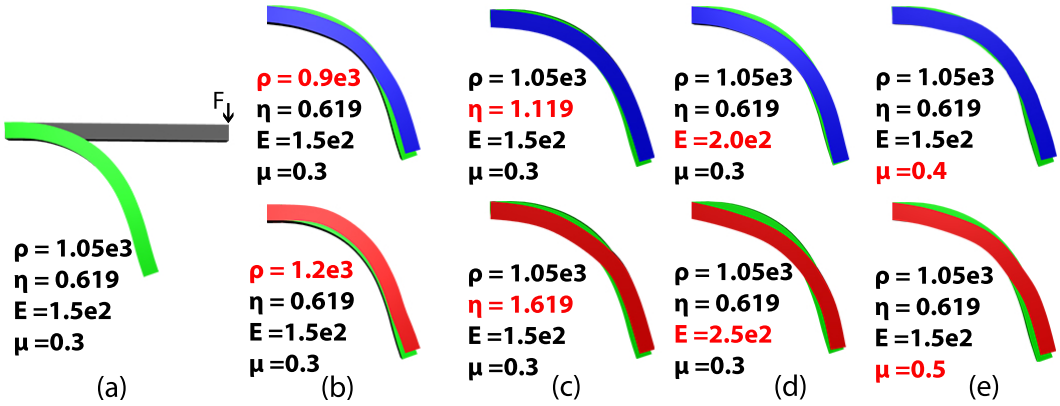


Figure 6.11: *In the approach, skin deformations are similar to the elastic bending of thin plates and the bending deformations are connected to Young's modulus E, Poisson ratio μ, and the material parameters, density ρ and damping coefficient η. (a) All of them have an influence on skin shapes. (b) density ρ, dynamic vibration enhances, when ρ increasing and when ρ reach 0, it achieves static deformation; (c) shows how damping coefficients η influence results when it increasing; (d) shows when Young's modulus E increasing, amplitude of object decrease; (e) shows how Poisson ratio μ affects the result.*

6.5 Experimental results and Comparisons

This approach is tested on some 3D models for experiments. In Subsection 6.1, first discuss some basic behaviors of the proposed physics-based skin deformations. In order to demonstrate this approach can produce realistic skin deformations efficiently, I compared the proposed approach with geometric skin deformation methods in Subsection 6.5.2, and with the example-based skin deformation method, i. e., pose space deformation (PSD) method in Subsection 6.5.3, and with physics-based dynamic skin deformation methods in Subsections 6.5.4 and 6.5.5 where Subsection 6.5.4 uses curve-defined skin models and Subsection 6.5.5 uses polygon skin models.

6.5.1 Basic Behaviors

Basic behaviors to be tested include the effects of material and mechanical parameters, differences between dynamic and static skin deforma-

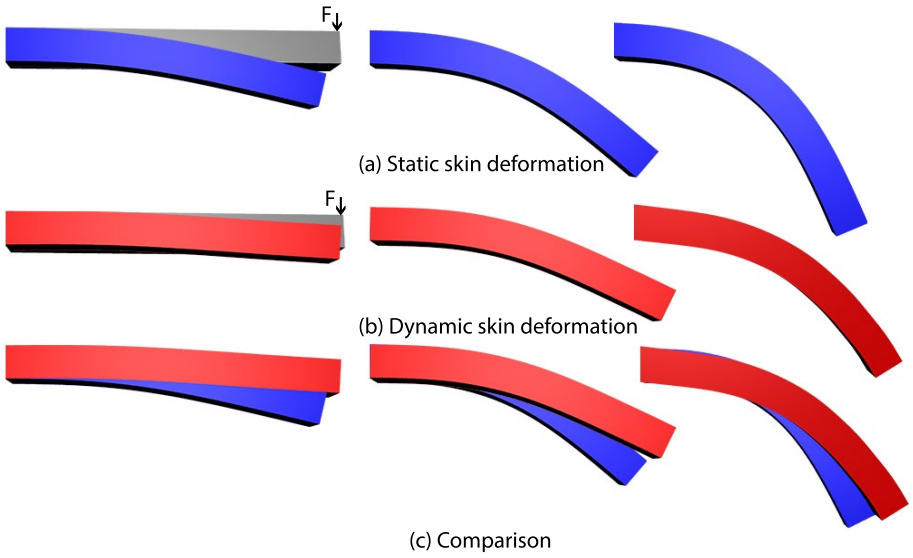


Figure 6.12: *In the proposed physics-based skin deformation model, setting the density $\rho = 0$ and the damping coefficient $\eta = 0$, the obtained analytical solution can be used to create static skin deformations. (a) indicates static skin deformations generated by this approach, (b) indicates dynamic skin deformations generated by this approach, and (c) demonstrates substantial differences between the generated static and dynamic skin deformations.*

tions, and the reversibility of the proposed approach.

As aforementioned, skin deformations are similar to the elastic bending of thin plates and the bending deformations are described by Eq. 6.11, where the coefficient ξ is similar to the flexural rigidity of elastic plate bending, which is connected to Young's modulus E, Poisson ratio μ, and skin thickness h through the equation $\xi = Eh^3/[12(1-\mu^2)]$. Young's modulus, Poisson ratio and the tensile stiffness ς are mechanical parameters of the skin. The density ρ and damping coefficient η are material parameters. All of them have influence on skin shapes and first investigate what are their effects.

Considering a skin strip with a rectangle cross-section of height h and unit width shown in Figure 6.11(a) where the left end of the skin strip is fixed and the right end is under the action of a force F. According to Amar [2010]; Elert [1998], these mechanical and material parameters for the skin strip can be taken to be: $E = 1.5e2$, $\mu = 0.3$, $\rho = 1.05e3$,

and $\eta = 0.619$. With these material and mechanical parameters, the straight skin strip in grey is deformed into a new shape in green shown in Figure 6.11(a). Then investigate how each parameter of E, μ, ρ and η affect skin deformations, shown in Figure 6.11(b)-(e).

In the proposed physics-based skin deformation model, setting the density $\rho = 0$ and the damping coefficient $\eta = 0$, the obtained analytical solution can be used to create static skin deformations. The differences between static and dynamic skin deformations are shown in Figure 6.12 where Figure 6.12(a) indicates static skin deformations, Figure 6.12(b) indicates dynamic skin deformations, and Figure 6.12(c) shows the differences between static and dynamic skin deformations. The images in Figure 6.12 demonstrate substantial differences between static and dynamic skin deformations.

The proposed analytical solution to physics-based skin deformations is reversible. I illustrate this in Figure 6.13. The upper red images show the deformations from the starting pose in Figure 6.13(a) to the end pose in Figure 6.13(f), the bottom blue images show the deformations from the end pose to the starting pose, and the middle thin curves are skin deformations in the upper local region which are taken from the upper and bottom images. These middle curves show the same deformations whether from the starting pose to the ending pose or from the ending pose to the starting pose. This shows the reversibility of the proposed analytical solution.

6.5.2 Comparison with Geometric Skin Deformation Methods

In this subsection, I compared the deformed shapes created by this approach with 3D scanned groundtruth and those from baseline methods including the classical LBS and the Dual Quaternion blending skinning Kavan et al. [2008] which are included in standard character animation pipelines. Since LBS and DQBS have been included in Maya, I use Maya's LBS and DQBS to determine skin deformations.

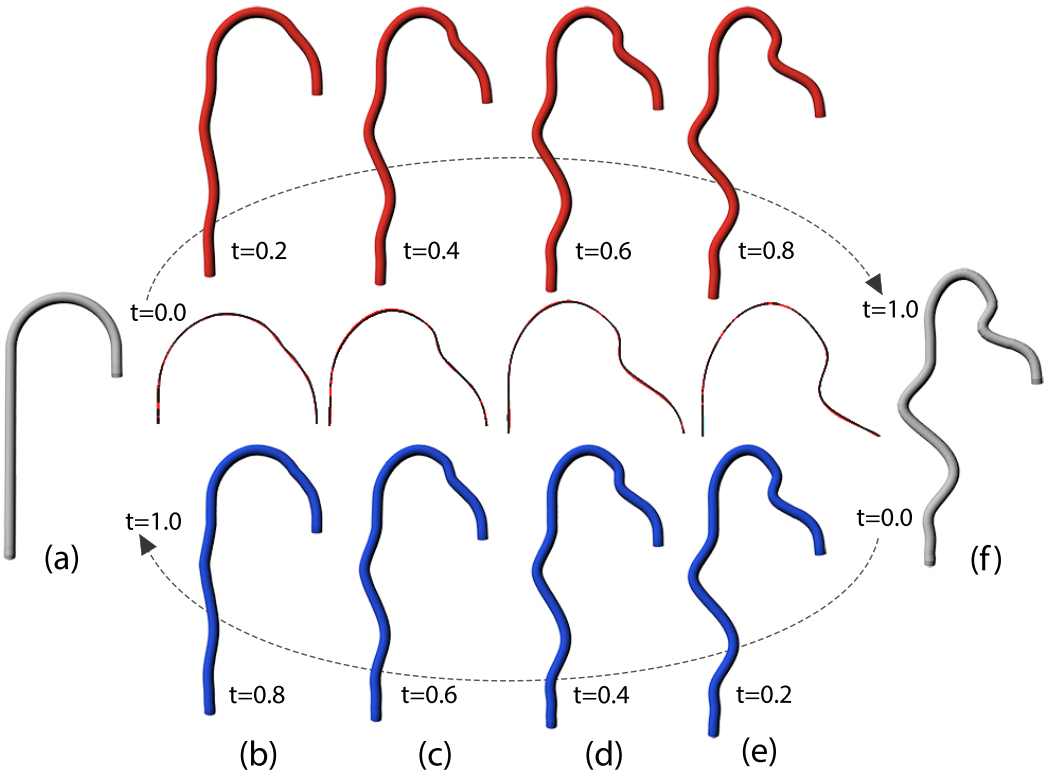


Figure 6.13: *Reversibility illustration. The red images in first row show the deformations from the starting pose (a) to the end pose (f), the blue images in bottom row show the deformations from the end pose (f) to the starting pose (a), and the overlapped red and blue thin curves in middle row are the skin deformations which are taken and magnified from the top local region of first row and bottom images. These middle curves show the deformations are the same, whether from the starting pose to the ending pose or from the ending pose to the starting pose. It indicates reversibility of this proposed analytical solution.*

Figure 6.14(a) - (e) show the skin deformation result of a male model obtained by the proposed approach. The comparison with those obtained by the two baseline methods is indicated in Figure 6.14(f) - (g). The proposed approach gives more realistic skin deformations at the arm and the leg joints.

In order to quantify the errors of various methods, I calculated the vertex errors between the 3D scanned groundtruth and those obtained by the proposed approach and the two baseline methods, and depicted the vertex errors with different colours in Figure 6.15. As shown in the figure, the skin deformation by the proposed approach is the closest to

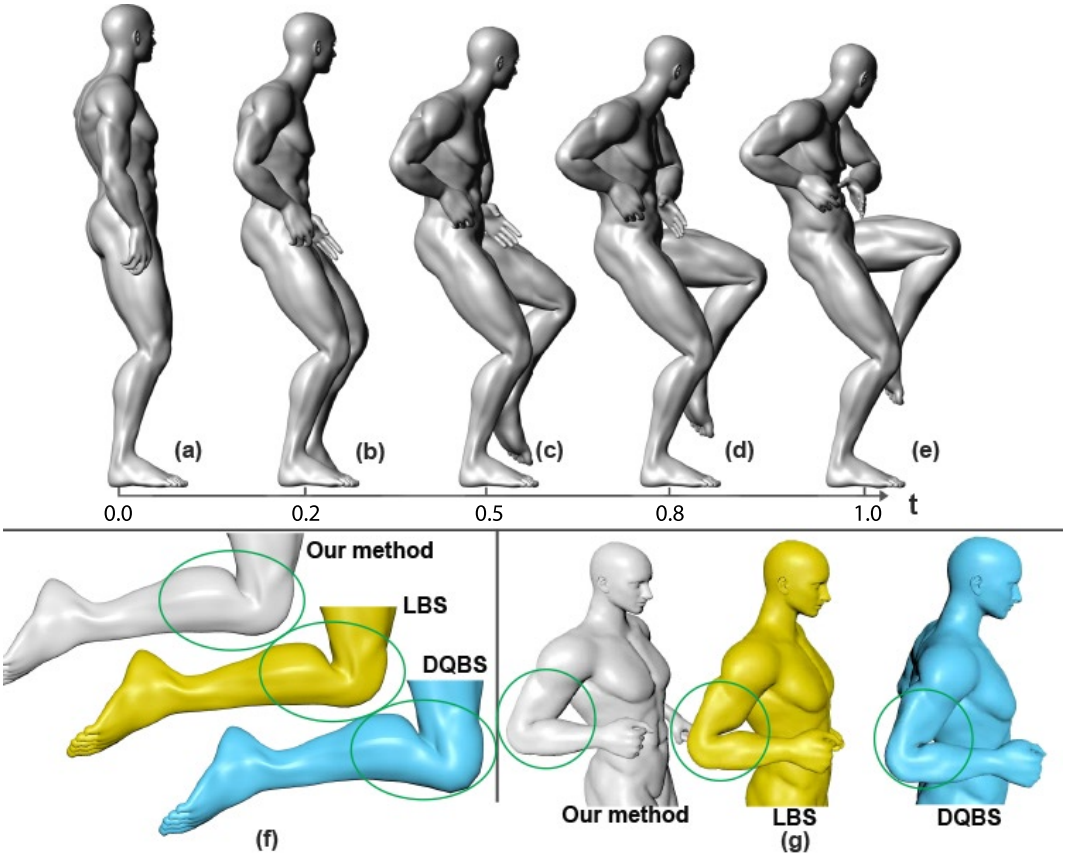


Figure 6.14: *Skin deformation result of a running male model by this proposed approach. (a) and (e) are the input example poses. (b)-(d) are generated intermediate shapes by this approach. (f) and (g) shows this approach gives more realistic skin deformations at leg joints and arm joints. LBS denotes the standard linear blending skinning model, and DQBS denotes the Dual Quaternion Blending Skinning model Kavan et al. [2008].*

the ground-truth among all the methods in this comparison. The visual perception of the difference between the ground-truth and the proposed approach is negligible at most.

In contrast, the meshes created by the classic LBS and dual quaternion blend skinning are noticeably different from the ground-truth. Here, Maximum Vertex Error $r = dif/len$ (calculated by the ratio r of Euler's difference dif between the deformed mesh and ground-truth mesh divided by the largest length of ground-truth mesh's bounding box len), is used for evaluation.The Maximum Vertex Error (MVE) by the proposed method is 0.026, while other methods have much larger MVEs: LBS

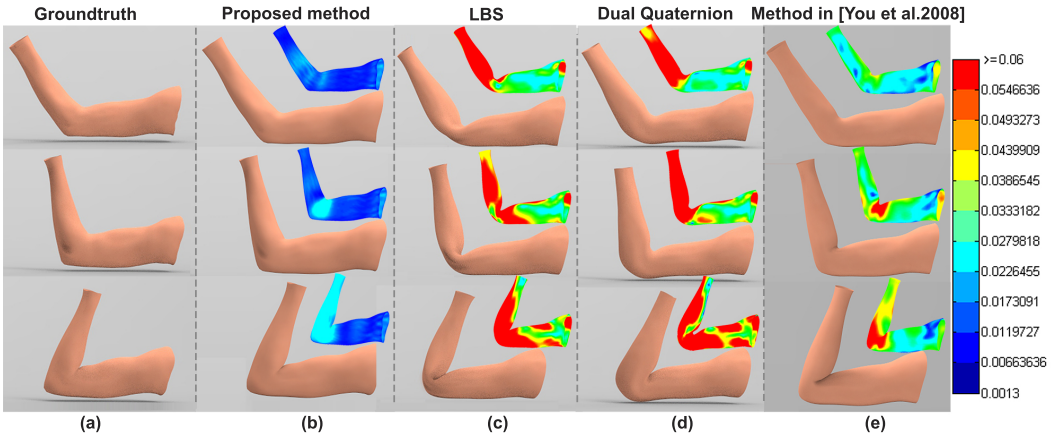


Figure 6.15: *Comparisons between the proposed method and the baseline skinning methods for an arm model. The leftmost column shows the ground-truth arm model reconstructed from laser scanning data. Three rows (from top to bottom) corresponding to poses at t=0.25, t=0.5, and t=0.75, respectively. The columns from left to right are: ground-truth, the proposed approach, the classical LBS, dual quaternion skinning Kavan et al. [2008], curve-based method You et al. [2008]. The color images at top-right corner are the visualization of the vertex errors, compared to the ground-truth.*

(0.385), the dual quaternion blend skinning (0.278) and the method of You et al. [2008](0.169).

6.5.3 Comparison with Example-based Skin Deformation Methods

As discussed in the Introduction Section, example-based skin deformation methods are often used together with geometric skinning, and realistic skin deformations require sufficient examples. Since the Pose Space Deformation (PSD) method Lewis et al. [2000] is a widely-used example-based skin deformation method, I compare this approach with PSD in this subsection and demonstrate the proposed physics-based approach can be used to improve the realism of example-based skin deformations.

Two arm skin shapes at the starting pose $t = 0$ and the ending pose $t = 1$ respectively shown in Figure 6.16(a) are used for the comparison. The obtained skin shapes at $t = 0.25$, $t = 0.5$ and $t = 0.75$ are depicted

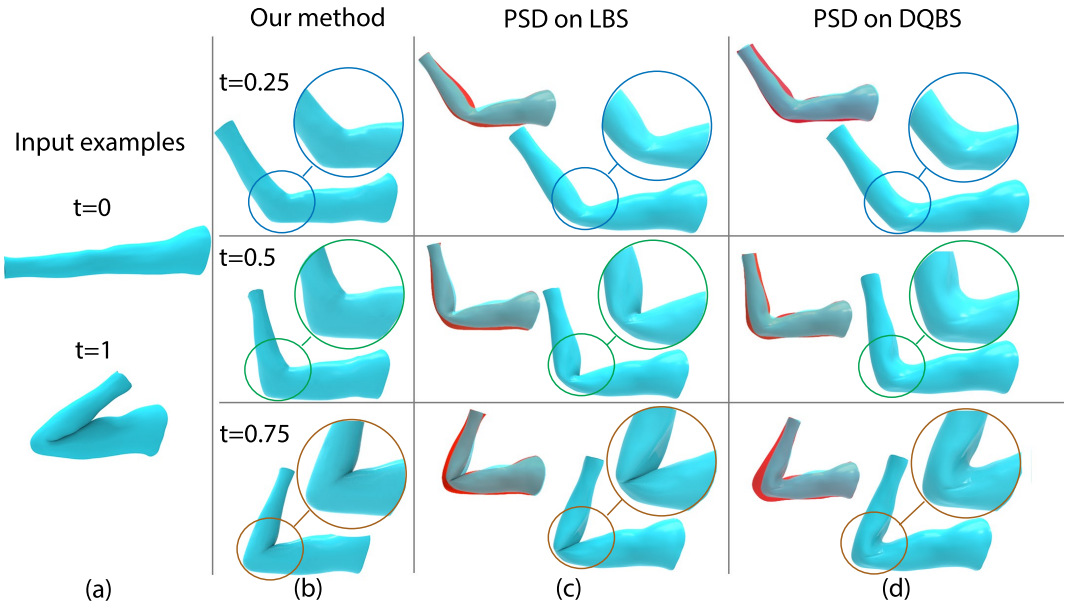


Figure 6.16: *Comparison with PSD, (a) shows the input data, two scan arm models: one at the starting pose and the other at the ending pose, (b) shows the models generated by this proposed method, without collapsing-joint, bulging-joint and distorted normal. (c) shows the models create by PSD (Pose Space Deformation) based on LBS. The left overlap parts show the differences between results of LBS (red) and PSD on LBS (blue). (d) shows the models generated by PSD based on DQBS. The left overlap parts also show the differences between results of DQBS (red) and PSD on DQBS(blue).*

in Figure 6.16(b)-(d). In the figure, (b) is from the proposed approach, and (c) and (d) are from the PSD tool added to Maya 2016 Extension 2 where (c) is the PSD on the linear blending skinning (LBS), and (d) is the PSD on the DQBS.

Comparing the images shown in the figure, it can be found that both PSD on LBS and PSD on DQBS improve the skin deformation results purely on LBS and DQBS, but still can not avoid the collapsing-joint, bulging-joint and distorted normal effects, whereas the proposed approach creates more realistic skin deformations without these artifacts.

6.5.4 Comparison with Physics-based Skin Deformation Methods Using Curve Defined Models

Although several previous curve-based skin deformation methods You et al. [2008]; Chaudhry et al. [2015] have been proposed, they have the following limitations. 1) The curves defining skin surfaces are manually extracted, leading to heavy and time-consuming human involvements. 2) The animation skeleton is manually created, resulting in non-trivial human efforts and preprocessing time. 3) A large linear system must be solved to obtain skin deformations at finite difference nodes Chaudhry et al. [2015] or unknown constants involved in analytical mathematical expressions of skin deformations You et al. [2008]. 4) The animation is not smooth at the connecting poses (that is, the ending pose of the current animation segment is also the starting pose of next animation segment).

The above limitations 1) to 3) make previous curve-based skin deformation methods less suitable for those applications requiring high animation frame rates, and the above limitation 4) would seriously affects the quality of resulting animations.

The proposed approach overcomes all the four limitations. Since both deformation continuity and deformation rate continuity are introduced into Eq. 6.17, the proposed method can be used only requiring at least two example meshes to produce smooth and physically realistic animations as demonstrated by the horse model in Figure 6.17. In contrast, the previous approaches such as You et al. [2008] would fail to produce smooth animations if multiple example poses are given, as demonstrated by the animation result in the accompanying video.

Among various existing curve-based skin deformation approaches, the work of You et al. [2008] is the most efficient. I compare the computing time of the proposed approach with You et al. [2008] through four different models: scanned arm, cat, horse, and human, presented in Table 6.1 shows that the proposed approach is about 12-18 times faster than You

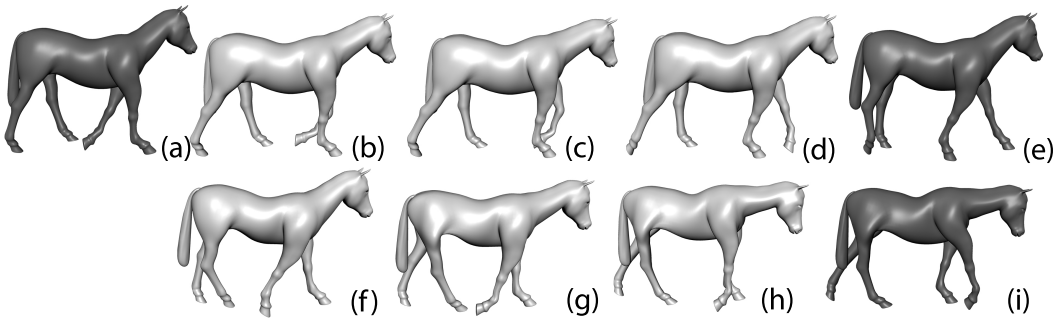


Figure 6.17: *Skin deformation result of a horse model. (a), (e), and (i) are three input example poses, (b) - (d) are in-between poses generated by the proposed method based on (a) and (e). (f) - (h) are in-between poses generated by the proposed approach based on (e) and (i). During these two adjacent sequences, the shapes show a smooth transition form the first sequence (a)-(e) to the second sequence (e)-(i).*

Table 6.1: *Runtime efficiency comparisons between the proposed method and You et al. [2008]*

Data	Verts	The proposed method(fps)	You et al. [2008](fps)
Arm	3840	531	23
Cat	7207	273	17
Horse	10128	205	14
Runner	32185	86	4

et al. [2008], and the calculations are carried out by C++(11), Visual Studio 2013, on a 3.2GHz Intel(R) Xeon(R) CPU E5-1650 Hewlett-Packard HP Z240 Workstation with 32 GB of memory.

6.5.5 Comparison with Physics-based Skin Deformation Methods Using Polygon Models

As discussed before, physics-based skin deformation methods produce high-quality and realistic results, but require heavy numerical calculations. The proposed method significantly reduces the computing cost, manual operations and realize high animation efficiency. This mainly depends on: (1) automatic isoparametric curve extraction, (2) Fourier series representation with much fewer variables, (3) fast analytical solution for the explicit determination of physics-based skin deformations. As Eq. 6.30 shows, once two example meshes are given, physically real-

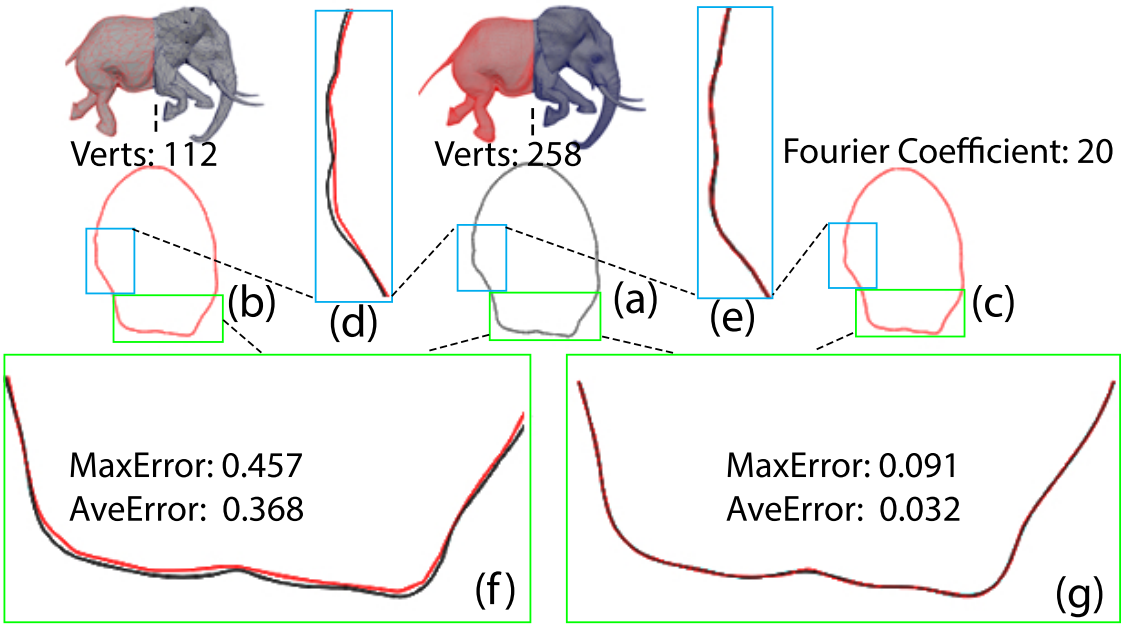


Figure 6.18: *The high-resolution curve(a) in polygon mesh reduced to (b) with large accuracy lose in (d), (f). But Fourier series representation (c) keep high realism with sharply decreasing data-size*

istic intermediate skin deformations could be obtained directly. In this subsection, I will compare the proposed approach with existing physics-based skin deformation methods using polygon models.

I first demonstrate the proposed Fourier series representation can greatly reduce design variables compared with polygon models but still keep high-realism, shown as Figure 6.18. Next, I make a comparison between the proposed proposed approach and the work described in Xu & Barbič [2016] since it is most closely relevant to ours. Finally, I compare the proposed approach to some other physics-based skin deformation methods which are less relevant to ours.

Table 6.2 gives a comparison of design variables. For polygon models, the total design variables are the total vertices. After representing polygon models with Fourier series, the total design variables become the total vector-valued Fourier coefficients.

The second row in Table 6.2 shows the total vertices of arm, cat, horse, dancer, and runner models. The third, fourth, and fourth rows show the total vector-valued Fourier coefficients when Fourier series terms used to

Table 6.2: *Comparison of design variables*

	Arm	Cat	Horse	Dancer	Runner
Verts	3840	7207	10128	13201	32185
FC N=3	448	1681	2363	3080	5632
FC N=5	704	2642	3713	4840	8850
FC N=9	1216	4563	6414	8360	15287

FC:Fourier Coefficients Terms

Table 6.3: *Runtime breakdown of the proposed approach when it was used to create the animations of different models.*

			Runtime (ms)				
Model	Verts	FC	IIL	GT	FSR+DC	AS+SD	Total
Arm	3840	1216	0.35	0.215	1.19	0.13	1.885
Cat	7207	2642	0.72	0.258	2.47	0.21	3.658
Horse	10128	3713	0.99	0.450	3.15	0.27	4.860
Runner	32185	15287	1.83	0.656	8.25	0.79	11.526

Note: FC:Fourier Coefficients, variables number of Fourier series representations used in these experiments.
IIL: Identification of isoparametric curves. GT: Geometric Transform. FSC: Fourier Series Representation.
DC: Deformation Calculation. AS: Analytic Solver. SD:Skin Deformer.

describe the models are set to be 3, 5, and 9, respectively. The data in this table illustrate the design variables of polygon models are 4.3-8.6, 2.7-5.5, and 1.6-3.2 times of the design variables of Fourier series representations for the same models when Fourier series terms N=3, 5, and 9, respectively. This demonstrates by using Fourier series representations, I reduce design variables sharply, and the specific value of Fourier coefficients used in these experiments are shown in Table 6.3 and all the calculations are carried out by C++(11), Visual Studio 2013, on a 3.2GHz Intel(R) Xeon(R) CPU E5-1650 Hewlett-Packard HP Z240 Workstation with 32 GB of memory.

Among various physics-based skin deformation approaches, the approach proposed in Xu & Barbič [2016] incorporate physically based simulation into rigging/skinning to automatically produce secondary skin motion and use model reduction to accelerate numerical calculations, which sacrificed the accuracy to a certain extent.

It should be noted that, since the proposed method focuses on in-

tegrating examples to physics-based model to create high-realistic skin deformation in steady-state part under extra vibration forces; in contrast, the main goal of Xu & Barbič [2016] is to achieve the balance between high-realism and forced vibration part. Therefore, I cannot do a fair comparison between the proposed method and Xu & Barbič [2016].

To accommodate large deformations around each pose, Xu & Barbič [2016] augments the linear model basis with modal derivatives proposed in Barbič & James [2005].

In Barbič & James [2005], after frame 1000, the vertical displacement of the spoon simulation vertex between full simulation and modal derivatives could reach the length of spoon. In order to obtain inertial forces, Xu & Barbič [2016] needs several samples for training, which brings in less than 3% relative training error. Although the pose-dependent model reduction Xu & Barbič [2016] divides a polar bear model of 6,876 vertices into 8 local regions and uses 8 threads with OpenMP, it still takes 1.43 ms to animate skin deformations of the polar bear model. Without any multithreading, the proposed approach only takes 1.627 ms to animate skin deformations of a horse model with 10,128 vertices. It indicates that the proposed approach is comparable with the pose-dependent model reduction, if not faster.

Other physics-based skin deformation approaches are substantially slower than the pose-dependent model reduction. The method in Kavan & Sorkine [2012] generates higher quality deformations than both linear and dual quaternion skinning through skinning weights optimization, but requires at least a few minutes to precompute the deformation weights. The method in McAdams et al. [2011] still needs at least several seconds for torso and arms simulation per frame on GPU, it is still not fast enough for interactive posing.

Through a quantitative comparison with ground-truth skin deformations shown in Figure 6.15, the high realism of the proposed approach has been clearly demonstrated. I also compare the proposed approach with the finite element simulation of steady-state skin vibrations in Figure 6.19, which further shows good realism achieved by the proposed

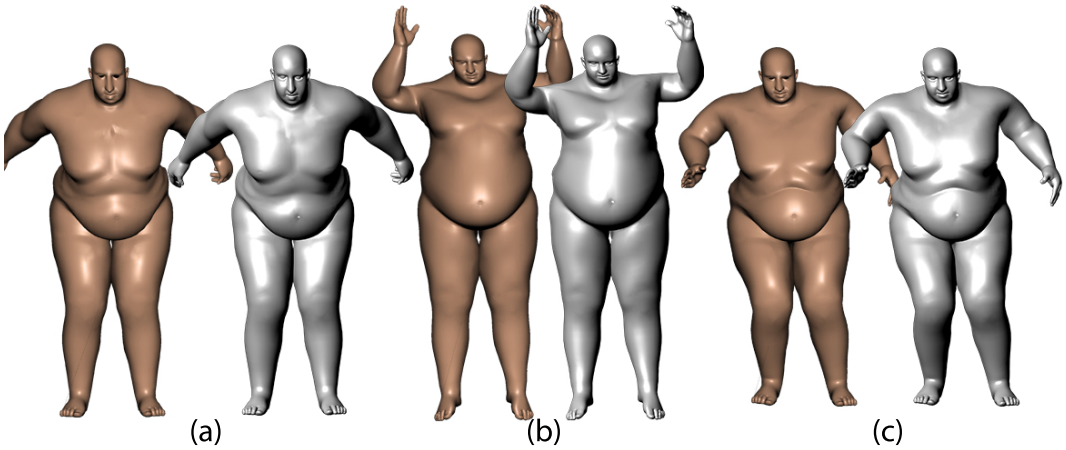


Figure 6.19: *Simulating skin vibration with steady-state dynamics:(a)(b)(c) are comparisons of three model poses created by ABAQUS steady-state dynamic simulation (gold) and by the steady-state dynamic simulation (silver).*

approach.

6.6 Discussion and Conclusion

In this chapter, an analytical approach is developed to efficiently create physically-realistic skin deformations. It includes several key steps: vertex identification on isoparametric curves, Fourier series conversion, and design of an analytic solution to the formulated physics-based skin deformation model. I have conducted many experiments to validate the proposed approach, discussed the effects of various factors, made direct comparisons with various skin deformation approaches including geometric skin deformation methods, example-based skin deformation methods, and physics-based skin deformation methods using curve defined models and model reduction. These results demonstrate the proposed approach avoids unnecessary manual operations and time-consuming numerical calculation that require specialized knowledge and skills, produces physically-realistic models and reaches high animation frame rates for real-time applications.

Several improvements can be made in the future. First, I will extend

this work to develop an analytical solution to fourth order differential equations to tackle linear elastic dynamic skin deformations including both transient and steady-state skin vibrations. Second, the current analytic solution to physics-based skin deformation is derived from linearly varying forces. It can be extended to nonlinearly varying forces. I will examine which force variation can generate more realistic skin deformations in the future work. Thirdly, an more adaptive value of step size during determination of intersecting curves will improve the realism of skin deformation. Lastly, the proposed approach can be extended to generate detailed clothing deformations. This can be achieved by combining the proposed analytic physics-based skinning with clothing examples obtained from the SOR scheme Xu et al. [2014].

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In the area of character animation, skin surface modelling, rigging and skin deforamtion are three essential aspects. Due to the different complexity of the characters, the time cost on creating corresponding skin surface model, animation skeleton to achieve diverse skin deformations fluctuates from several hours to several weeks. More importantly, the data size of skin deformations could sharply influence the efficiency of generating animation. Smaller data size also can speed up character animation and transmission over computer networks.

This thesis has reviewed the existing work on above-mentioned aspects. Although some recent research efforts have pursued the direction of integrating geometric, example-based, and physics-based skin deformation approaches, I have not found any approaches that are capable of describing physics-based skin deformation (static or dynamic) with analytical solutions, to the best of our knowledge. This research developed one skin deformation approach with analytical solution, which is capable of producing physically-realistic deformations with high efficiency (e.g., much faster than the numerical approaches employed by conventional physics-based skin deformations without model reduction), without requiring specialized knowledge, skills, and heavy manual involvements.

In order to address the issues of high-realism, efficiency and small data size in creating skin deformations for character animation, this thesis has developed techniques based on Ordinary Differentical Equations (ODE) to automatically generate skeleton, efficiently create C2 continuous skin surfaces and achieve physically realistic skin deformations for human body animation by integrating geometric and data-driven methods. Meanwhile, effectively reduce the data size by iso-parametric curves and Fourier Series representation. ODE model is built and accurate and efficient solutions are developed to calculate physical skin deformations through interpolating and traning of input realistic reconstructed 3D models. The proposed techniques will greatly reduce data size, improve realism, and raise efficiency of producing character animation.

7.2 Future Work

The current research work has some limitations which can be the focus of future work:

The work on an automatic rigging algorithm in this research could be improved in several directions. The proposed iso-parametric curve identification is applicable to quad meshes with regular topology. Future work will investigate triangule meshes and arbitrary topologies. The proposed automatic rigging requires a template skeleton. In order to make the proposed automatic rigging applicable to various character models, different character models will be classified into different categories and a small template skeleton database with one skeleton in the database for one category of character models will be created.

In this research, the analytical approach is developed to efficiently create physically-realistic skin deformations, avoiding unnecessary manual operations and time-consuming numerical calculation that require specialized knowledge and skills, produces physically-realistic models and reaches high animation frame rates for real-time applications. Several improvements can be made in the future. First, I will extend the current method to develop an analytical solution to fourth order differential

equations to tackle linear elastic dynamic skin deformations including both transient and steady-state skin vibrations. Second, current analytic solution to physics-based skin deformation is derived from linearly varying forces. It can be extended to nonlinearly varying forces. I will examine which force variation can generate more realistic skin deformations in the future work. Thirdly, an more adaptive value of step size during determination of intersecting curves will improve the realism of skin deformation. Lastly, the proposed approach can be extended to generate detailed clothing deformations. This can be achieved by combining the proposed analytic physics-based skinning with clothing examples obtained from the SOR scheme Xu et al. [2014].

Bibliography

M. R. Amar (2010). 'Estimation of mechanical properties of soft tissue subjected to dynamic impact' .

A. Angelidis & K. Singh (2007). 'Kinodynamic Skinning Using Volume-preserving Deformations'. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pp. 129–140, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

M. Athanasopoulos, et al. (2009). 'Parametric design of aircraft geometry using partial differential equations'. *Advances in Engineering Software* **40**(7):479–486.

O. K.-C. Au, et al. (2008). 'Skeleton extraction by mesh contraction'. *ACM Transactions on Graphics (TOG)* **27**(3):44.

J. A. Bærentzen, et al. (2014). 'Interactive shape modeling using a skeleton-mesh co-representation'. *ACM Transactions on Graphics (TOG)* **33**(4):132.

I. Baran & J. Popović (2007). 'Automatic rigging and animation of 3d characters'. In *ACM Transactions on Graphics (TOG)*, vol. 26, p. 72. ACM.

J. Barbič & D. L. James (2005). 'Real-time subspace integration for St. Venant-Kirchhoff deformable models'. *ACM transactions on graphics (TOG)* **24**(3):982–990.

L. Bardis & N. Patrikalakis (1989). 'Blending rational B-spline surfaces'. In *Eurographics*, vol. 89, pp. 453–462.

R. E. Barnhill, et al. (1993). 'Constant-radius blending of parametric surfaces'. In *Geometric modelling*, pp. 1–20. Springer.

M. Bartoň, et al. (2014). 'Detection and reconstruction of freeform sweeps'. In *Computer Graphics Forum*, vol. 33, pp. 23–32. Wiley Online Library.

M. Bartoň, et al. (2013). 'Circular arc snakes and kinematic surface generation'. In *Computer Graphics Forum*, vol. 32, pp. 1–10. Wiley Online Library.

J. Bender, et al. (2013). 'Physically-based character skinning' .

G. Bharaj, et al. (2012). 'Automatically rigging multi-component characters'. In *Computer Graphics Forum*, vol. 31, pp. 755–764. Wiley Online Library.

B. Bickel, et al. (2009). 'Capture and modeling of non-linear heterogeneous soft tissue'. In *ACM Transactions on Graphics (TOG)*, vol. 28, p. 89. ACM.

P. Blair (1994). *Animation: learn how to draw animated cartoons*. Walter T. Foster.

I. M. Bloor & M. J. Wilson (1996). 'Spectral approximations to PDE surfaces'. *Computer-Aided Design* **28**(2):145–152.

M. Bloor & M. Wilson (1989). 'Generating N-sided patches with partial differential equations'. In *New Advances in Computer Graphics*, pp. 129–145. Springer.

M. Bloor, et al. (2000). 'Generating blend surfaces using a perturbation method'. *Mathematical and Computer Modelling* **31**(1):1–13.

M. I. Bloor & M. J. Wilson (1990a). 'Representing PDE surfaces in terms of B-splines'. *Computer-Aided Design* **22**(6):324–331.

M. I. Bloor & M. J. Wilson (1990b). 'Using partial differential equations to generate free-form surfaces'. *Computer-Aided Design* **22**(4):202–212.

M. I. Bloor & M. J. Wilson (1993). 'Functionality in solids obtained

from partial differential equations'. In *Geometric modelling*, pp. 21–42. Springer.

M. I. Bloor & M. J. Wilson (2005). 'An analytic pseudo-spectral method to generate a regular 4-sided PDE surface patch'. *Computer Aided Geometric Design* **22**(3):203–219.

C. Brandt, et al. (2016). 'Geometric flows of curves in shape space for processing motion of deformable objects'. In *Computer Graphics Forum*, vol. 35, pp. 295–305. Wiley Online Library.

D. F. Brown, et al. (2013). 'Control of rotational dynamics for ground behaviors'. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 55–61. ACM.

M. D. Buhmann (2003). 'Radial basis functions: theory and implementations'. *Cambridge Monographs on Applied and Computational Mathematics* **12**:147–165.

C. C. Caeiro, et al. (2013). 'OrangFACS: a muscle-based facial movement coding system for orangutans (Pongo spp.)'. *International Journal of Primatology* **34**(1):115–129.

E. J. Candès, et al. (2011). 'Robust principal component analysis?'. *Journal of the ACM (JACM)* **58**(3):11.

T. J. Cashman, et al. (2009). 'NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes'. In *ACM Transactions on Graphics (TOG)*, vol. 28, p. 46. ACM.

I. Chao, et al. (2010). 'A simple geometric model for elastic deformations'. *ACM Transactions on Graphics (TOG)* **29**(4):38.

E. Chaudhry, et al. (2015). 'Dynamic skin deformation using finite difference solutions for character animation'. *Computers & Graphics* **46**:294–305.

X. Chen, et al. (2009). 'A benchmark for 3D mesh segmentation'. In *Acm transactions on graphics (tog)*, vol. 28, p. 73. ACM.

S. Cheng, et al. (1990). 'Blending between quadric surfaces using partial differential equations'. *Advances in design automation* **1**:257–263.

B. K. Choi & S. Ju (1989). 'Constant-radius blending in surface modelling'. *Computer-Aided Design* **21**(4):213–220.

J.-H. Chuang & W.-C. Hwang (1997). 'Variable-radius blending by constrained spine generation'. *The Visual Computer* **13**(7):316–329.

J.-H. Chuang, et al. (1995). 'Variable-radius blending of parametric surfaces'. *The Visual Computer* **11**(10):513–525.

S. Coros, et al. (2010). 'Generalized biped walking control'. In *ACM Transactions on Graphics (TOG)*, vol. 29, p. 130. ACM.

D. Danielson (1973). 'Human skin as an elastic membrane'. *Journal of biomechanics* **6**(5):539–546.

E. De Aguiar, et al. (2008). 'Performance capture from sparse multi-view video'. *ACM Transactions on Graphics (TOG)* **27**(3):98.

M. Del Giudice & L. Colle (2007). 'Differences between children and adults in the recognition of enjoyment smiles.'. *Developmental psychology* **43**(3):796.

T. DeRose, et al. (1998). 'Subdivision surfaces in character animation'. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 85–94. ACM.

P. Ekman & W. Friesen (????). 'Facial action coding system: a technique for the measurement of facial movement. 1978'. *Consulting Psychologists, San Francisco* .

G. Elert (1998). 'The Physics Hypertextbook'.

G. Farin (2014). *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier.

R. A. Farouki & R. Sverrisson (1996). 'Approximation of rolling-ball blends for free-form parametric surfaces'. *Computer-Aided Design* **28**(11):871–878.

F. Faure, et al. (2011). 'Sparse meshless models of complex deformable solids'. In *ACM transactions on graphics (TOG)*, vol. 30, p. 73. ACM.

D. J. Filip (1989). 'Blending parametric surfaces'. *ACM Transactions on Graphics (TOG)* **8**(3):164–173.

A. Freitas-Magalhães (2013). *The face of lies*. Leya.

A. Freitas-Magalhães (2018). *Facial Action Coding System-Manual of Scientific Codification of the Human Face*. Leya.

W. V. Friesen, et al. (1983). 'EMFACS-7: Emotional facial action coding system'. *Unpublished manuscript, University of California at San Francisco* **2**(36):1.

R. Gal, et al. (2009). 'iWIRES: an analyze-and-edit approach to shape manipulation'. In *ACM Transactions on Graphics (TOG)*, vol. 28, p. 33. ACM.

J. Genzer & J. Groenewold (2006). 'Soft matter with hard skin: From skin wrinkles to templating and material characterization'. *Soft Matter* **2**(4):310–323.

B. Gilles, et al. (2011). 'Frame-based elastic models'. *ACM transactions on graphics (TOG)* **30**(2):15.

J. C. Hager (2003). 'FACS Final Test (FFT) Overview'.

P. E.-W. F.-J. HAGER (2002). 'Facial Action Coding System. The Manual On CD ROM'.

F. Hahn, et al. (2012). 'Rig-space physics'. *ACM transactions on graphics (TOG)* **31**(4):72.

J. Hamm, et al. (2011). 'Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders'. *Journal of neuroscience methods* **200**(2):237–256.

N. Hasler, et al. (2010). 'Learning skeletons for shape and pose'. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 23–30. ACM.

B. Heeren, et al. (2014). 'Exploring the geometry of the space of shells'. In *Computer Graphics Forum*, vol. 33, pp. 247–256. Wiley Online Library.

B. Heeren, et al. (2012). 'Time-Discrete Geodesics in the Space of Shells'. In *Computer Graphics Forum*, vol. 31, pp. 1755–1764. Wiley Online Library.

F. Higuchi, et al. (2007). 'Approximation of involute curves for CAD-system processing'. *Engineering with Computers* **23**(3):207–214.

C.-H. Hjortsjö (1969). *Man's face and mimic language*. Studen litteratur.

J. Hoschek, et al. (1993). *Fundamentals of computer aided geometric design*. AK Peters, Ltd.

G. W. Housner & D. E. Hudson (1980). 'Applied mechanics dynamics' .

K. L. Hsu & D. M. Tsay (1998). 'Corner blending of free-form N-sided holes'. *IEEE Computer Graphics and Applications* **18**(1):72–78.

W.-C. Hwang, et al. (2003). 'N-sided hole filling and vertex blending using subdivision surfaces'. *Journal of Information Science and Engineering* **19**(5):857–879.

D.-E. Hyun, et al. (2005). 'Sweep-based human deformation'. *The Visual Computer* **21**(8-10):542–550.

A. Jacobson, et al. (2012). 'Fast automatic skinning transformations'. *ACM Transactions on Graphics (TOG)* **31**(4):77.

A. Jacobson, et al. (2011). 'Bounded biharmonic weights for real-time deformation.'. *ACM Trans. Graph.* **30**(4):78.

A. Jacobson, et al. (2014). 'Skinning: Real-time shape deformation'. *ACM SIGGRAPH Course* .

A. Jacobson & O. Sorkine (2011). 'Stretchable and twistable bones for skeletal shape deformation'. *ACM Transactions on Graphics (TOG)* **30**(6):165.

D. L. James & D. K. Pai (2002). 'DyRT: dynamic response textures

for real time deformation simulation with graphics hardware'. *ACM Transactions on Graphics (TOG)* **21**(3):582–585.

D. L. James & C. D. Twigg (2005). 'Skinning mesh animations'. In *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 399–407. ACM.

L. Kavan, et al. (2008). 'Geometric skinning with approximate dual quaternion blending'. *ACM Transactions on Graphics (TOG)* **27**(4):105.

L. Kavan & O. Sorkine (2012). 'Elasticity-inspired deformers for character articulation'. *ACM Transactions on Graphics (TOG)* **31**(6):196.

M. Kilian, et al. (2007). 'Geometric modeling in shape space'. In *ACM Transactions on Graphics (TOG)*, vol. 26, p. 64. ACM.

J. Kim & N. S. Pollard (2011). 'Fast simulation of skeleton-driven deformable body characters'. *ACM Transactions on Graphics (TOG)* **30**(5):121.

Y. Kim & J. Han (2014). 'Bulging-free dual quaternion skinning'. *Computer Animation and Virtual Worlds* **25**(3-4):321–329.

P. Koparkar (1991). 'Parametric blending using fanout surfaces'. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pp. 317–327. ACM.

G. Kós, et al. (2000). 'Methods to recover constant radius rolling ball blends in reverse engineering'. *Computer Aided Geometric Design* **17**(2):127–160.

R. Krasauskas (2008). 'Branching blend of natural quadrics based on surfaces with rational offsets'. *Computer Aided Geometric Design* **25**(4-5):332–341.

P. G. Kry, et al. (2002). 'Eigenskin: real time large deformation character skinning in hardware'. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 153–159. ACM.

P. G. Kry, et al. (2012). 'Inverse kinodynamics: Editing and constraining

kinematic approximations of dynamic motion'. *Computers & Graphics* **36**(8):904–915.

Y.-K. Lai, et al. (2008). 'Fast mesh segmentation using random walks'. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pp. 183–191. ACM.

B. H. Le & Z. Deng (2012). 'Smooth skinning decomposition with rigid bones'. *ACM Transactions on Graphics (TOG)* **31**(6):199.

B. H. Le & Z. Deng (2014). 'Robust and accurate skeletal rigging from mesh sequences'. *ACM Transactions on Graphics (TOG)* **33**(4):84.

B. H. Le & J. K. Hodgins (2016). 'Real-time skeletal skinning with optimized centers of rotation'. *ACM Transactions on Graphics (TOG)* **35**(4):37.

D. D. Lee & H. S. Seung (2001). 'Algorithms for non-negative matrix factorization'. In *Advances in neural information processing systems*, pp. 556–562.

S.-H. Lee, et al. (2009). 'Comprehensive biomechanical modeling and simulation of the upper body'. *ACM Transactions on Graphics (TOG)* **28**(4):99.

J. P. Lewis, et al. (2000). 'Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation'. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 165–172. ACM Press/Addison-Wesley Publishing Co.

D. Li, et al. (2013). 'Thin skin elastodynamics'. *ACM Transactions on Graphics (TOG)* **32**(4):49.

G. Li & H. Li (2002). 'Blending parametric patches with subdivision surfaces'. *Journal of computer science and technology* **17**(4):498–506.

S. Li, et al. (2014). 'Space-time editing of elastic motion through material optimization and reduction'. *ACM Transactions on Graphics (TOG)* **33**(4):108.

Z.-c. Li (1998). 'Boundary penalty finite element methods for blending

surfaces, I basic theory'. *Journal of Computational Mathematics* pp. 457–480.

Z.-C. Li & C.-S. Chang (1999). 'Boundary penalty finite element methods for blending surfaces, III. Superconvergence and stability and examples'. *Journal of Computational and Applied Mathematics* **110**(2):241–270.

C. C. Ling, et al. (2010). 'Approximating GCS by low energy Hermite curve'. *European Journal of Scientific Research* **46**(4):616–626.

A. C. Lints-Martindale, et al. (2007). 'A psychophysical investigation of the facial action coding system as an index of pain variability among older adults with and without Alzheimer's disease'. *Pain Medicine* **8**(8):678–689.

L. Liu, et al. (2008). 'Surface reconstruction from non-parallel curve networks'. In *Computer Graphics Forum*, vol. 27, pp. 155–163. Wiley Online Library.

L. Liu, et al. (2012). 'Terrain runner: control, parameterization, composition, and planning for highly dynamic motions.'. *ACM Trans. Graph.* **31**(6):154.

L. Liu, et al. (2013). 'Simulation and control of skeleton-driven soft body characters'. *ACM Transactions on Graphics (TOG)* **32**(6):215.

G. Lukács (1998). 'Differential geometry of G1 variable radius rolling ball blend surfaces'. *Computer Aided Geometric Design* **15**(6):585–613.

N. Magnenat-Thalmann, et al. (1988). 'Joint-dependent local deformations for hand animation and object grasping'. In *In Proceedings on Graphics interface88*. Citeseer.

A. McAdams, et al. (2011). 'Efficient elasticity for character skinning with contact and collisions'. In *ACM Transactions on Graphics (TOG)*, vol. 30, p. 37. ACM.

A. Mohr & M. Gleicher (2003). 'Building efficient, accurate character skins from examples'. In *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 562–568. ACM.

A. Nealen, et al. (2007). 'FiberMesh: designing freeform surfaces with 3D curves'. *ACM transactions on graphics (TOG)* **26**(3):41.

A. Nealen, et al. (2006). 'Physically based deformable models in computer graphics'. In *Computer graphics forum*, vol. 25, pp. 809–836. Wiley Online Library.

L. P. Nedel & D. Thalmann (1998). 'Modeling and deformation of the human body using an anatomically-based approach'. In *Computer Animation 98. Proceedings*, pp. 34–40. IEEE.

T. Neumann, et al. (2013). 'Sparse localized deformation components'. *ACM Transactions on Graphics (TOG)* **32**(6):179.

A. C. Öztireli, et al. (2013). 'Differential blending for expressive sketch-based posing'. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 155–164. ACM.

J. Pan, et al. (2009). 'Automatic rigging for animation characters with 3D silhouette'. *Computer Animation and Virtual Worlds* **20**(2-3):121–131.

N. Pantuwong & M. Sugimoto (2012). 'A novel template-based automatic rigging algorithm for articulated-character animation'. *Computer Animation and Virtual Worlds* **23**(2):125–141.

S. I. Park & J. K. Hodgins (2008). 'Data-driven modeling of skin and muscle deformation'. In *ACM Transactions on Graphics (TOG)*, vol. 27, p. 96. ACM.

L. A. Parr, et al. (2010). 'Brief communication: MaqFACS: a muscle-based facial movement coding system for the rhesus macaque'. *American journal of physical anthropology* **143**(4):625–630.

L. A. Parr, et al. (2007). 'Classifying chimpanzee facial expressions using muscle action.'. *Emotion* **7**(1):172.

L. Piegl & W. Tiller (2012). *The NURBS book*. Springer Science & Business Media.

L. A. Piegl & W. Tiller (1999). 'Filling n-sided regions with NURBS patches'. *The Visual Computer* **15**(2):77–89.

H. Pyun, et al. (2004). 'On extracting the wire curves from multiple face models for facial animation'. *Computers & Graphics* **28**(5):757–765.

S. Qin & D. K. Wright (2006). 'Progressive surface modelling scheme from unorganised curves'. *Computer-Aided Design* **38**(10):1113–1122.

L. I. Reed, et al. (2007). 'Impact of depression on response to comedy: A dynamic facial coding analysis.'. *Journal of abnormal psychology* **116**(4):804.

O. Rémillard & P. G. Kry (2013). 'Embedded thin shells for wrinkle simulation'. *ACM Transactions on Graphics (TOG)* **32**(4):50.

T. Rhee, et al. (2006). 'Real-Time Weighted Pose-Space Deformation on the GPU'. In *Computer Graphics Forum*, vol. 25, pp. 439–448. Wiley Online Library.

H. Rhodin, et al. (2014). 'Interactive motion mapping for real-time character control'. In *Computer Graphics Forum*, vol. 33, pp. 273–282. Wiley Online Library.

H. Rhodin, et al. (2015). 'Generalizing wave gestures from sparse examples for real-time character control'. *ACM Transactions on Graphics (TOG)* **34**(6):181.

J. Rossignac & A. Requicha (1984). 'Constant-radius blending in solid modelling' .

M. Russo (2006). *Polygonal modeling: basic and advanced techniques*. Jones & Bartlett Learning.

S. Saito, et al. (2015). 'Computational bodybuilding: anatomically-based modeling of human bodies'. *ACM Transactions on Graphics (TOG)* **34**(4):41.

B. Sauvage, et al. (2008). 'Detail preserving deformation of B-spline surfaces with volume constraint'. *Computer Aided Geometric Design* **25**(8):678–696.

M. Schichtel (1993). 'G/sup 2/blend surfaces and filling of N-sided holes'. *IEEE Computer Graphics and Applications* **13**(5):68–73.

M. Seiler, et al. (2012). 'Enriching coarse interactive elastic objects with high-resolution data-driven deformations'. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 9–17. Eurographics Association.

A. Shamir (2008). 'A survey on mesh segmentation techniques'. In *Computer graphics forum*, vol. 27, pp. 1539–1556. Wiley Online Library.

K.-L. Shi, et al. (2010). 'Filling n-sided regions with G 1 triangular Coons B-spline patches'. *The Visual Computer* **26**(6-8):791–800.

K. Singh & E. Fiume (1998). 'Wires: a geometric deformation technique'. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 405–414. ACM.

P.-P. J. Sloan, et al. (2001). 'Shape by example'. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 135–143. ACM.

J. Stam (1998). 'Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values'. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 395–404. ACM.

Y. Teng, et al. (2014). 'Simulating articulated subspace self-contact'. *ACM Transactions on Graphics (TOG)* **33**(4):106.

R. Vaillant, et al. (2013). 'Implicit skinning: real-time skin deformation with contact modeling'. *ACM Transactions on Graphics (TOG)* **32**(4):125.

R. Vaillant, et al. (2014). 'Robust iso-surface tracking for interactive character skinning'. *ACM Transactions on Graphics (TOG)* **33**(6):189.

S.-J. Vick, et al. (2007). 'A cross-species comparison of facial morphology and movement in humans and chimpanzees using the facial action coding system (FACS)'. *Journal of Nonverbal Behavior* **31**(1):1–20.

J. Vida, et al. (1994). 'A survey of blending methods that use parametric surfaces'. *Computer-Aided Design* **26**(5):341–365.

A. Vögele, et al. (2012). 'Interactive steering of mesh animations'. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 53–58. Eurographics Association.

B. M. Waller, et al. (2012). 'GibbonFACS: a muscle-based facial movement coding system for hylobatids'. *International Journal of Primatology* **33**(4):809–821.

B. M. Waller, et al. (2013). 'Paedomorphic facial expressions give dogs a selective advantage'. *PLoS one* **8**(12):e82686.

Y. Wang, et al. (2015). 'Linear subspace design for real-time shape deformation'. *ACM Transactions on Graphics (TOG)* **34**(4):57.

J. Warren & H. Weimer (2001). *Subdivision methods for geometric design: A constructive approach*. Morgan Kaufmann.

B. Whited & J. Rossignac (2009). 'Relative blending'. *Computer-Aided Design* **41**(6):456–462.

H. Xu & J. Barbič (2016). 'Pose-space subspace dynamics'. *ACM Transactions on Graphics (TOG)* **35**(4):35.

W. Xu, et al. (2014). 'Sensitivity-optimized rigging for example-based real-time clothing synthesis'. *ACM Trans. Graph.* **33**(4):107–1.

X. Yang & J. J. Zhang (2006). 'Stretch it-realistic smooth skinning'. In *Computer Graphics, Imaging and Visualisation, 2006 International Conference on*, pp. 323–328. IEEE.

Y.-J. Yang, et al. (2006). 'A rational extension of Piegls method for filling n-sided holes'. *Computer-Aided Design* **38**(11):1166–1178.

K. Yin, et al. (2007). 'Simbicon: Simple biped locomotion control'. In *ACM Transactions on Graphics (TOG)*, vol. 26, p. 105. ACM.

S.-H. Yoon & M.-S. Kim (2006). 'Sweep-based freeform deformations'. In *Computer Graphics Forum*, vol. 25, pp. 487–496. Wiley Online Library.

L. You, et al. (2011). 'Solid modelling based on sixth order partial differential equations'. *Computer-Aided Design* **43**(6):720–729.

L. You, et al. (2004a). 'PDE blending surfaces with C2 continuity'. *Computers & Graphics* **28**(6):895–906.

L. You, et al. (2007). 'Boundary constrained swept surfaces for modelling and animation'. In *Computer Graphics Forum*, vol. 26, pp. 313–322. Wiley Online Library.

L. You, et al. (2008). 'Dynamic skin deformation with characteristic curves'. *Computer Animation and Virtual Worlds* **19**(3-4):433–444.

L. You, et al. (2004b). 'Blending surface generation using a fast and accurate analytical solution of a fourth-order PDE with three shape control parameters'. *The Visual Computer* **20**(2-3):199–214.

H. Zhong, et al. (2005). 'A real time finite element based tissue simulation method incorporating nonlinear elastic behavior'. *Computer Methods in Biomechanics and Biomedical Engineering* **8**(3):177–189.

P. Zhou & W.-H. Qian (2009). 'A vertex-first parametric algorithm for polyhedron blending'. *Computer-Aided Design* **41**(11):812–824.

P. Zhou & W.-H. Qian (2010). 'Polyhedral vertex blending with setbacks using rational S-patches'. *Computer Aided Geometric Design* **27**(3):233–244.

H. Zou, et al. (2006). 'Sparse principal component analysis'. *Journal of computational and graphical statistics* **15**(2):265–286.