Efficient and detailed sketch-based character modelling with composite generalized elliptic curves and ODE surface creators

Ouwen Li

Thesis of Doctor of Philosophy

Bournemouth University

Supervised by Prof Lihua You, Prof JianJun Zhang

Copyright statement

Publications

Li, Ouwen, Zhigang Deng, Shaojun Bian, Algirdas Noreika, Xiaogang Jin, Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang. "ODE-Driven Sketch-Based Organic Modelling." In Computer Graphics International Conference, pp. 453-460. Springer, Cham, 2019.

Li, Ouwen, Shaojun Bian, Algirdas Noreika, Ismail Khalid Kazmi, Lihua You, and Jian Jun Zhang.Efficient creation of 3D organic models from sketches and ODE-based deformations", accepted by LNCS Transactions on Computational Science.

Abstract

Sketch-based modelling (SBM), dating back to 1980s, has attracted a lot of researches' attention due to its easy-to-use features and high efficiency in generating 3D models. However, existing sketch-based modelling approaches are incapable in creating detailed and realistic 3D character models. This project aims to propose new techniques which can create more detailed 3D character models with easiness and efficiency. The basic idea is to fit primitives to the sketches consisting of front view contours, side view contours and cross-section curves to obtain more detailed shape, propose ODE (ordinary differential equation) driven deformation to create more realistic shapes, and use surfaces defined by cross-sectional curves to represent sketch-based and ODE-driven 3D character models.

In order to achieve the above aim, this thesis firstly investigates curve fitting of cross-sectional shapes and solved the problem of representing cross-sectional curves with generalized ellipses or composite generalized elliptic segments. Then, this thesis proposes a new mathematical formula for defining a surface from the cross-sectional curves. A new sketch-guided and ODE-driven character modelling technique is proposed, consisting of two main components: primitive deformer and detail generator. With such a technique, I first draw 2D silhouette contours of a character model. Then, I select proper primitives and align them with the corresponding silhouette contours. After that, I develope a sketch-guided and ODE-driven primitive deformer. It uses ODE-based deformations to deform the cross-section curves of the primitive to exactly match the generated 2D silhouette contours in one view plane and with the curve-fitting method and surface reconstruction method mentioned above, a base mesh of a character model consisting of deformed primitive is obtained. In order to add various 3D details, I develop a local shape generator which uses sketches in different view planes to define a local shape and employs ODE-driven deformations to create a local surface passing through all the sketches. The experimental results demonstrate that the proposed approach can create 3D character models with 3D details from 2D sketches easily, quickly and precisely.

Cross-section contours are important in defining cross-section shapes and creating detailed models. In order to develop a cross-section contour-based modelling approach, how to mathematically represent cross-section curves must be first solved. The second aim of this project is to propose composite generalized elliptic curves and introduce them into character modelling to achieve an analytical and compact mathematical representation of cross-section contours.

Current template-based character modelling which creates 3D character models from sketches retrieves and then uses 3D template models directly. Since retrieving 3D models from sketches is not an easy task, the third aim of this project is to extract 2D cross-section contours from template models and use the extracted 2D cross section contours as templates to assist the creation of 3D character models for simplifying and accelerating the modelling process. Although there are many different approaches to interpret shapes with sketch strokes, but to our knowledge, no one utilises 2D template cross-section contours to quickly generate the shapes of human characters in a sketch-based system, which is one of the contributions of this project.

Contents

List of Figures

Thesis Outline

This thesis is structured by seven chapters.

- **Chapter 1 - Introduction:** In this chapter, a brief overview of 3D character modeling technologies, including their strengths and weaknesses, their applications and limitations is introduced. Followed by how sketch-based modeling approach can help to tackle the problems mentioned above. Then the contribution of this research is provided.
- **Chapter 2 - Related Work:** This chapter gives a literature review of sketch-based modeling system, modeling from geometric primitives and shape construction by cross-section curves that are related to this research.
- **Chapter 3 - Curve Fitting for Cross-Section:** This chapter presents a method to approximate a cross-section curve as a generalized ellipse or a group of end-to-end generalized elliptic segments with a parametric formula.
- **Chapter 4 - Cross-section Defined Surface Modeling** This chapter presents a surface reconstruction method. In addition, this method is extended to a surface blending method that maintains a set continuity between two primary surfaces whose edges are cross-sectional curves created by the method introduced in Chapter 3.
- **Chapter 5 - Primitive Generator:** In this chapter, an ODE-driven sketch-based modeling method for generating a primitive and deforming its cross-section curves so that the primitive's projected silhouette can fit the input sketch silhouette. This deformation ap-

proach can also be extended to further deform the primitive by fitting its cross-section contour(s) to the reference cross-section contour(s) extracted from reference meshes. Afterward, the deformed cross-sectional curves are represented by trigonometric series with the algorithm introduced in Chapter 3 and then the new surface is created with these cross-sectional curves by the method proposed in Chapter 4.

- **Chapter 6 - Free-form Patch Generator:** In this chapter, an ODE-driven sketch-based modeling method for generating a free-form patch is proposed. With this detail generator, details such as accessories and clothes can be added on human character models.

- **Chapter 7 - Conclusion and Future Work:** This chapter sums up the conclusion of this research and discusses the limitation of this work, and how to take this research to the next level.

Chapter 1

Introduction

1.1 Overview

Human character modeling is one of the central task in creative industries and medical simulation, such as film/animation/game industry and visual surgery for decades. However this task still remains very tedious, time-consuming and requires a lot of training time for novice users to become proficient modellers.

In most cases, modellers will import reference images from the concept artists, and then begin the modelling and sculpting operations. Those reference images, sometimes called model sheets, are the profiles from T-pose or A-pose in front view, side view and back view respectively, or various arbitrary poses in arbitrary perspective views. According to Hoffman & Singh [1997], for human vision systems, silhouettes alone are efficient for recognising everyday objects, by indexing their shapes in human memory. Also, I noticed that the most natural way for both novice modellers and veteran modellers is to generate a new shape is to sketch freehand strokes with paper-pen-like graphic input tablets, so does the shape modification. In addition, the recent surge of painting or modelling applications in virtual reality such as Arora et al. [2017], Gravity Sketch [2017], Tilt Brush Google Inc [2017], Quill Facebook [2019], Medium Oculus [2019] and A-Painter Mozilla [2019] also show it's viable for artists to sketch in 3D canvas. Therefore, modeling from the sketches, due to its intuitiveness and user-friendliness, becomes a promising research area and has attracted many researchers' interests for decades.

However, there are still challenges in current sketch-based modeling systems. First of all, they are mainly designed for modeling simple shapes such as rotund toys(Igarashi et al. [1999]; Yang & Wünsche [2010]; Nealen et al. [2007a]) and Constructive Solid Geometry (Shtof et al. [2013]; Trimble Inc. [2019]). To fill the gap of modeling complex and more detailed organic or human characters with SBM is the main challenge. Sketch-based editing technology like Nealen et al. [2005] doesn't support accurate editing. This research will address this issue so that the new mesh's projected contour can fully and accurately match the input sketch strokes. In addition to modeling the overall shape, functions of adding details to the organic and human characters such as accessories should be also supported. Finally, computational efficiency and storage cost should be taken into consideration as well as visual fidelity.

1.2 Research Objectives

In order to tackle the challenges mentioned above, following objectives should be obtained:

- Literature Review: Investigate existing researches on sketch-based modeling system, ODE-based geometric processing, and shape editing technology.
- Propose a new curve and a new surface representation to efficiently reconstruct a 3D shape from cross-sectional curves.
- Introduce a new sketch-based modelling technology to control the overall shape. Since ODE are widely used in simulating physic phenomenon, we'll introduce an ODE driven mesh editing technology which generates plausible skin deformations with tolerable computation costs.
- Integrate the local detail generation into the SBM system.

1.3 Contribution

A sketch-based modeling system is proposed in this research. This research not only focuses on the base mesh modelling phase, but also the editing/sculpting phase by deforming the shapes according to cross-sectional contours and silhouette contours. By inventing an ordinary differential equation (ODE) driven surface deformation technology, my research reduces the computational cost compared with the state-of-art physics-based surface deformation methods, meanwhile achieve a better physical plausibility than purely geometric shape deformation methods. The main contributions to the sketch-based modeling technique are listed as follows:

- **Reconstruct the 3D shape by interpolating cross-sectional contours that are extracted from 3D template models**: Extract (and edit if required) cross-sectional contours of template 3D models and curve fitting them with trigonometric series. Afterward, reconstruct 3D shapes based on the cross-sectional contours.

- **Shape deforming technology**: Find an efficient physical-based mesh deformation algorithm which works well on quadrilateral meshes. Deform the initial primitive according to the input sketch silhouette. If more silhouettes or cross-sectional contours are provided for local shape editing, edit the primitives by deforming the primitive so that the new shape can exactly match the input sketch strokes. Moreover, based on other sets of curves from the input sketches, generate details like nose, eyes, and ears on the base mesh.
- **Sketch-based modeling system**: Equipped with the primitive deformer and detail generator methods, develop a user-friendly human character sketch based modelling system.

Chapter 2

Literature Review

In this project, an ODE-driven sketch-based modelling system specialized in human character models is proposed. To be more exact, the data extracted from matching template 3D model are cross-sectional contours which contains critical shape information of human body. The actual shape generation and shape manipulations will be under an interactive sketch-based interface; and the shape deformation algorithms are the cornerstone of the whole modelling and editing system. Hence, this literature review will be divided into three parts: human character modelling related basic knowledge, overview of sketch-based modelling systems and shape deformation technologies.

2.1 Human Character modelling

2.1.1 Shell surface modelling vs Volumetric modelling

Shell surface modelling has been widely used and a large amount of researches have investigated into this realm. There are four types of shell surface presentations: polygon mesh, parametric surfaces (eg. nonuniform rational B-splines (Piegl & Tiller [2012]), Coons patch (Coons [1967]), etc), implicit surfaces (for example, metaball), and subdivision surfaces.

Volumetric modelling is very computational expensive. Unlike shell surfaces which only store surface data, it also stores interior space data under the surfaces. In spite of this, the volumetric modelling meets the requirement of medical research(Gibson et al. [1998]) where the underlying anatomy are important and hence worth keeping, as such, volumetric modelling is optimal for soft-tissue deformation and tissue cutting.

As computational cost and visual quality are often inversely proportional, in film and animation industry, NURBS surfaces are commonly used, given that high-end render package will render the smooth surface off-line, while in game industry, relatively low-resolution polygon mesh is the format to ensure real-time rendering but sometimes compromise the image quality (Parent [2012]).

As discussed above, shell surface modelling is most widely applied and volumetric modelling is very computationally expensive. My work will investigate shell surface modelling only.

2.1.2 Anatomy-based Human Character modelling

Good human character models should meet two requirements: they should look believable and have no weird artifacts when animating. By taking cues from the knowledge of human anatomy, a new genre of humanoid modelling encompassing skeleton, muscles, viscera, fat tissues, and skins has emerged, creating ready-to-animated models suitable for simulation.

A number of anatomy-based human character modelling techniques follow the paradigm of dividing a human representation into three different layers: the skeleton, the intermediate tissues like muscles, fat, fascia, and the external skin surface. Scheepers et al. [1997] considered the influence of the musculature on the exterior form, developed anatomy-based models of muscles which responds to the changes of the posture of an underlying articulated skeleton, and applied them to the torso and arm of a human figure. Jane & Allen [1997] proposed an improved, anatomically based approach to modelling and animating animals. It models muscles, bones, and generalized tissue as triangle meshes or ellipsoids, treats muscles as deformable discretized cylinders whose shapes change as the joints move, and finally creates polygonal isosurface for external skin by voxelizing, filtering and extracting from the underlying components. Nedel & Thalmann [1998], Porcher Nedel & Thalmann [2000], and Aubel & Thalmann [2001] presented the force exerted from muscles to bones as action lines, introduced a mass-spring system with angular springs to deform the action lines so as to physically simulate muscle deformations, and used ray-casting on semi-regular cylindrical grids to get the sample points which are later used directly as cubic B-spline control points to smoothen the skin surface.

Some anatomy-based modelling researches, on the other hand, model the internal anatomy elements based on physiological and bio-mechanical principles. Yang & Zhang [2006] presented a method to automatically extract major muscles from already modelled or scanned human characters. By using the constrained delaunay triangulation, the closed area formed by a cross-sectional curve is triangulated and then the triangles of the bulge region along the cross-sectional curve indicating the possible muscles will be collapsed with the method used in Igarashi et al. [1999]. Later the muscle's cross-section curves will be approximated by ellipses. With all the muscles' cross-sectional curves, the muscle shape can be lofted. Ali-Hamadi et al. [2013] proposed a method to transfer a template anatomical model with bones, muscles and fat tissues to a realistic or catoony target character, which is without anatomy elements yet. The template model skin and the target model skin are firstly registered, then the fat distribution, which is designed in a semi-automatic way: fat thickness information extracted and retargeted from MRI data, or the user can tune the fat tissues of the target character, erode the volume bounded by the skin to generate the fat tissue under the skin. Next, bone of the target model is interpolated, and is used as an attractor for constrained registration to get the correct bone shape for target model. Internal soft organs like viscera and muscles is then automatically created by volumetric interpolation between the bone and fat tissue. Saito et al. [2015] proposed a physical based muscle growth model and a fat growth model respectively for human characters, ready for simulation, with interactive performance and support extreme cases of hypertrophy and atrophy human characters. It takes into account volume conservation(plasticity), and uses Lagrangian approaches to solve the elasticity.

Other auxiliary researches concentrate on the topology and data structure for anatomical data. Like in the work of Beylot et al. [1996], topological modelling is introduced to address the issues of structural manipulation of the data acquired from medical imaging. Its main contribution is a new method called topological modelling which contains a new data structure storing information of structural, physical, geometrical and mechanical attributes for different organs, and 3D interactive tools for manipulate bones, joints and muscles to get motion simulation and dynamic analysis of the biomedical imaging data. Zou et al. [2015] presented the first surface reconstruction algorithm from an input set of planar cross-sections that allows global topology (genus) control. They proposed a definition of the function based on random walks, present algorithms for enumerating and scoring distinct level set topologies, and a bottom-up dynamic-programming algorithm to find the optimal solution meeting the topology constraint.

Although anatomy-based human character modelling can create realistic appearance, detect collision and dynamically correct, the computational cost of simulation the skin wrapped underneath anatomy elements is expensive.

2.2 Sketch-Based Modelling Systems Overview

Over the past several decades, sketch-based-modelling (SBM) has been widely studied in the computer graphic community (Olsen et al. [2009]). This section will firstly investigate the theories that prove sketches contain a good amount of 3D shapes' geometrical properties, therefore they can be used to reconstruct 3D shapes sufficiently. Then this section will dive into some typical and representative state-of-art sketch-based modelling systems that can be broadly divided into direct shape generation and template-based mesh creation.

2.2.1 Sketch Is a Reliable Source to Deduce 3D Shapes

Human's 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Therefore it is necessary to learn from human vision systems to improve the performance in computer graphic applications. DeCarlo et al. [2003] discovered that there are many other indicators for shape representations of organic objects include smooth feature curves, suggestive contours, etc. According to the research by Cole et al. [2008], human line drawings are quite consistent with each other. Lines drawn by artists largely overlapped one another, particularly along the occluding contours of the object. Most lines that do not overlap contours will overlap large gradients of the image intensity (as measured by image-space gradient magnitude) and correlate strongly with predictions made by recent line-drawing algorithms in computer graphics (CG). Those two kinds of locations mentioned above can be described by well-known, simple mathematical properties (Cole et al. [2008]). Moreover, their research shows that current non-photo-realistic-rendering technology has reached a state where most of the lines drawn by human can also be calculated by math algorithms. Their analysis lay the theoretical foundation that shapes can be deducted from lines drawing by human users when choose the appropriate line-drawing algorithms in SBM system.

2.2.2 Direct shape generation

In the category of direct shape generation, several systems have been proposed to generate organic models. Mainly, there are three types of mesh: geometric primitives, inflating surface and layered surface.

Geometric Primitive

An interesting phenomena is that many modelling systems' building bricks are geometrical primitives of all sorts.

To understand why geometrical primitives are chosen to be the starting points of modelling a figure, we must first know how human vision system understands figures and interprets their shapes.

Hoffman & Singh [1997] presented a theory of part salience. The theory builds on the minima rule for defining part boundaries. According to this rule, human vision defines part boundaries at negative minima of curvature on silhouettes, and along negative minima of the principal curvatures on surfaces. They propose that the salience of a part depends on (at least) three factors: its size relative to the whole object, the degree to which it protrudes, and the strength of its boundaries. Based on this rule, a human character has many component parts which are often differ in their visual salience. For example, those sizes relative to the whole object is substantial, like torso; those protrude much from the main shape, like head, limbs and foot; and those have boundaries of sharp curvature changes, like neck. They present the evidence to show that these factors influence visual processes which determine the choice of figure and ground. Hence, I divide human figure into different parts based on these factors.

Primitives-based systems decompose the modelling task as a process of creating a certain set of geometry primitives and further editing the primitives (Shtof et al. [2013]; Chen et al. [2013]; Xu et al. [2016]). The idea of assembling simple geometric primitives to form 3D models is very common in CSG (Constructive Solid Geometry) modelling. Shtof et al. [2013] introduced a snapping method to detect the feature curves and silhouette curves of both 2D sketches and 3D simple geometric primitives, i.e. box, sphere, cylinder, cone. Then automatically determined the core parameters of these simple geometric primitives so as to fit the 2D sketches, and then improved the model globally by inferring geosemantic constraints that define the relationship between different parts. The relationship includes: parallelism, orthogonality, collinear centers, concentric, and coplanar. In the work of Chen et al. [2013], the researchers proposed a tool to generate a cylinder from only 3 strokes: the first two strokes define the 2D profile and the last stroke defines the axis along which the profile curve sweeps. Copies of the profile are not only perpendicularly aligned to the axis, but also resized to snap to the input outlines. However, their work is only designed for man-made objects, where simple sweeping surfaces can meet the quality requirements of the modelling task.

Structured annotations for 2D-to-3D modelling (Gingold et al. [2009]), on the other hand, focus on organic modelling. It uses two sets of primitives. One is generalized cylinders, created by the input of a single open sketch stroke represented as a spline, and then modified by using simple gestures such as tilting cross sections, scaling local radius, rotating symmetrical plane, and changing cap size. The other is ellipsoids, generated according to the drawn closed ellipse sketch stroke. As its name indicates, there exist a set of annotation tools to further edit the surface shape using annotations such as same lengths, same angles, alignments, and mirror symmetries. Its main contribution is providing a solution to the problem of semantic information such as symmetric, equal length and angle, and alignment, which are quite obscure in the single view modelling system with only one arbitrary angled guide image.

Surface Inflation

Another way to generate a shape is inflating a surface represented by a closed 2D region/sketch to give it a volume (Kazmi et al. [2014]). The surface inflation technique extrudes a polygonal mesh from a given skeleton outwards and does a good job in modelling stuffed toys. One trend is to inflate free-form surfaces to create simple stuffed animals and other rotund objects in a SBM fashion (Igarashi et al. [1999]; Nealen et al. [2007a]; Karpenko & Hughes [2006]). The pioneering Teddy system (Igarashi et al. [1999]) takes closed curves as inputs, finds their chordal axes as the spines, then wraps the spines with a polygonal mesh.

Later, FiberMesh (Nealen et al. [2007a]) enriches editing operations for the inflated base mesh. This approach also presents two types of the control curves: smooth and sharp. A smooth curve constrains the surface to be smooth across it, while a sharp curve only places positional constraints with C^0 continuity. Sharp control curves appear when operations like cutting, extrusion, and tunnel take place. They also serve the creation of creases on the surface. The SmoothSketch system (Karpenko & Hughes [2006]) supports the creation of cusps and T-junctions, which Teddy and its successors fail to address. In addition, SmoothSketch extended Teddy's work to an extent that the strokes do not need to be closed.

Although the surface inflation approach is good at creating stuffed animals and simple toys, it is not versatile enough to express geometrical details on the surface. BendSketch(Li et al. [2017]) offers a technique which enables complex curvature patterns existing on surfaces. In order to give the bending information, users need to draw a set of lines that comply with what the BendSketch system has specified, which mimics the hatching technique artists often utilise to express the sense of volume and curvature information on the surfaces.

Layered modelling

From the observations of a large amount of sketching and concept art procedures, layered structure, defining by geometric dependence of layers on underlying layers, are commonly used in both organic and man-made objects.

SecondSkin (De Paoli & Singh [2015]) proposed a creation method for layered structure in a multi-view interactive fashion. They found that 91% of sketch strokes fall into these four curve-type categories: shell contour, projection, tangent and normal. Hence they developed an algorithm to identify the input sketch strokes' curve-type, and interpreted them as 3D curves on and around underlying 3D geometry, using other connected 3D curves for context. Curve loops are automatically surfaced and turned into volumes bound to the underlying layer, if necessary, additional curves and surfaces are created as well. This work assumes input sketched strokes are around the underlying object, rather than curves lying on the objects. Treating inputted sketches as curves on base mesh has been well investigated in prior works (Takayama et al. [2013]; Nealen et al. [2007a]; Kara &

Shimada [2007]).

Skippy (Krs et al. [2017]) is another sketch-based curve modelling system that predicts the depth information of 2D sketched curves by leveraging the existing geometries as guides. It works with a single-view. Like De Paoli & Singh [2015] it also recognises curve segments occluded by the existing geometries.

Others

There are other works about SKM don't fit in the aforementioned three catogories but still worth noting.

A set of intuitive creation rules help improve the easiness for end-users. ILoveSketch (Bae et al. [2008]) improved the gesture vocabulary, especially the roll-back gesture was added into the interaction gesture set, which is very inspiring. As a CAD tool, ILoveSketch provides smooth curve approximation to the input sketch stokes. It also addresses the problem of oversketch by develop a multi-stroke NURBS curve creation workflow.

Buchanan et al. [2013] presents a method for automatically constructing rigged, skinned and textured low-polygonal 3D meshes from a single piece of concept artwork, which can work in real-time graphic applications. This SBM system doesn't require any manual operations from 3D artists. Their algorithm casts several requirements to the input image: character is oriented in general towards the front; minimal occlusion or touching. By adapting the medial transform skeletonization method, they came up with a point contraction algorithm. They also proposed an algorithm to generate a swept shell from the cross-sectional slices which are also generated by computer. Based on their observation, concept artists draw a lot of pictures in three quarters view, so their orientation establishing has a promising applications.

There are certain limitations of the existing sketch-based-modelling technologies. For example, when creating complex organic character models, the geometric primitive method has the tendency of over-simplifying the initial form of the target shape with simple primitives like cube, cylinder, cone and sphere, which will cause more editing operations afterward to achieve the final shape. Moreover, the existing geometric primitive SKM systems offer few parameters for the primitives, which are mostly radius, rotation, and scale, as such, it limits the flexibility in the primitive creation process. As for the surface inflation methods, they only works when the input sketch curve is closed. My research will overcome this by introducing an ODE-based primitive deformer to automatically enable the silhouettes of primitive to exactly match the input sketches. I also propose an ODE-based detail generator to generate surfaces from both open and closed curves.

2.2.3 template-based mesh creation

Using geometric primitives to align 2D sketches provides an easy and efficient approach to generate rough base models. However, manipulating the generated rough base models in one image plane only is difficult to create detailed 3D models.

For template-based mesh creation, an elegant technique for sketch-based modelling has been proposed in Kraevoy et al. [2009] to find precise correspondences, which can be seen as an optimisation problem and can be solved by a hidden Markov model(HMM), and then an alternating correspond-

and-deform process determines the mesh deformations with mean-value encoding as the deformation method. Kazmi et al. [2015] proposed a real-time template-based SBM method. Their method finds the correspondence vertices between the input sketch contours and template model by KD-tree algorithm, which is faster than HMM. Then, by combining skeleton-based deformation and Laplacian mesh editing, an efficient approach quickly deforms a 3D template model to fit the user's drawn sketches.

Among various approaches of Direct shape generation from user-drawn sketches, a variety of sketch-based modelling tools are based on a "sketch-rotate-sketch" workflow. Such a workflow requires users to draw sketches from many views, causing the difficulties in matching input strokes with a guidance image and finding a good view (Gingold et al. [2009]).

2.3 Sketch-based Deformation

Deformation tools provide the interface for users to interact and modify mesh surfaces. A good deformation tool should meet the following principles (Botsch & Sorkine [2008]):

- Flexibility: permit users to change the mesh surface as they wish, meanwhile reserve correct modelling constraints.
- Shape quality: aesthetically pleasing
- Intuitive result: conform to the morphing happening in real world with real physical material, which makes no recognition confusion for users when they modify the shape. In other words, physically correct.

Sketch-based shape editing like Teddy and others (Karpenko & Hughes [2006]; Fleisch et al. [2004]; Terzopoulos et al. [1987]) featured inflation approaches use smooth silhouettes, thus they create only smooth shapes. In order to create aesthetically pleasing details on meshes, SBM should be equipped with shape editing method supporting inserting feature curves while preserve the global and local geometry.

Geometry-based editing system is based on the energy in a form of geometrical traits. A popular geometry-based frame is Laplacian/Poisson model, which represent the differential traits of the surface in various ways depending on how they are employed. In the discrete Laplacian/Poisson models, it is easy to displace a set of edges (e.g., sketch a new position of an identified contour) while preserving the geometric details of the surface as much as possible (Nealen et al. [2007b]). In order to address the problem that feature lines' differential properties are related to the viewing direction, and the feature lines are not coincide with edges on the mesh, Nealen et al. [2007a] extends the framework of Laplace/Poisson mesh modelling in 3 ways: (a) accommodate constraints on the normals and the curvature; (b) allow constraints to be placed on virtual vertices, i.e. vertices placed on edges that only serve to implement the constraints but are never added to the mesh. The virtual vertices are linearly interpolated on the edge between 2 vertices; (c) incorporate a tangential mesh regularization, which moves edges onto sharp features while ensuring well-shaped triangles. Their method supports to change the moderate noisy silhouette contours, to edit the sharp feature lines like ridge, ravine and crease, and to edit smooth features and suggestive contours.

Apart from deformation based on differential surface representations discuss in last paragraph. Another family make use of multi-resolution or subdivision editing as a way to perform global surface deformation while preserve local surface details. Subdivision mesh is firstly decomposed into a low-frequency base surface and a high-frequency detail information. Then after deforming the base surface, the detail information is added on the deformed base surface. In the case of interactive shape editing by means of manipulating control handles with 3 degrees of freedom for translation and 3 degrees of freedom for rotation, define the displacement vector as the change of position and orientation of the control handles. The final shape can be seen as the outcome of adding a basis function of displacement vector to the origin shape. The requested displacement has to be translated into coefficients of this basis functions. These coefficients include but not limited to smoothness, stiffness (order of energy function), fullness, etc. Besides, the basis functions require pre-computed inverse matrix to speed up the surface updating (Botsch & Kobbelt [2004]), which is another hint that these approaches are very computational-expensive.

Compared to the geometry-based technology, physic-based editing technology simulates the physical principle through physical energy function. It is more comply with the real-world physical fidelity. However, it is less flexible if the 3D artists want to achieve very drastic effects because the penalty of stretching or bending forces on the energy function will be very big. Another challenge of physically based skin deformation methods is that they are computationally expensive, at least they are not as efficient as geometry based approaches. Hence, my object is to improve the performance and efficiency of physical-based editing scheme by decreasing the 2-dimentional partial differential equation to 1-dimentional ordinary differential equation. Finite element method and finite difference method can be used to solve the partial derivative equation in this scenario. For the concern of computation efficiency, I choose the Finite Difference Method even though Finite Element Method produce more accurate approximation. And the obtained curve can be in the form of silhouettes and cross-sectional contours. Besides, it overcomes the shortage of geometry-based deformation method that the region affected by the geometry-based editing is limited (often within several rings) or defined by user as region of interesting. Whereas, the physical-based editing method this thesis proposes impacts the whole cross-sectional contour.

2.4 ODE-based geometric processing

Ordinary differential equations (ODEs) have been applied in 3D surface modelling, skin deformation and shape manipulation due to its high efficiency and physical-based nature.

You et al. [2007] introduced an ODE-based sweeping technology called boundary constrained swept surface(BCSS) to formulate the profile curves of a swept surface, and unlike usual sweeping surface which only has one trajectory and the generator is invariable, BCSS allows two trajectories and variable generator, and the result surface will meet the tangent conditions at both trajectories. Compares to ordinary modelling method, BCSS has the advantages of light weight in storage and computational cost, easy global shape manipulation, physically-based local shape manipulation, and has the potential to converting the complex models so that they become easier to be animated afterward.

You et al. [2008] deployed ODE in deforming the characters skin combining both the strength of physic-based and example-driven skin deformation. They first represent the surface by 3D curve network and represent the curves with time-dependent trigonometric series, and then constructing a deformation equation deriving from elastic beam bending, finally using the input poses from the examples, the the unknown constants of the equation can be found and the mathematical description of the time-dependent trigonometric series of the surface is determined. Chaudhry et al. [2013] simplified the formula of the external force that leads to the skin deformation into an ordinary differential equation that relates the position of point on open curve, the physical and geometric properties of the curve, to the external force. They calculated the forces for two consecutive poses, and interpolate the force for the in-between frames, finally get the positions for frames between the two input poses by solving the ODE. Since their model doesn't contain time variables, they call it static skin deformation. Based on the model describing the homogeneous elastic beam's bending behaviour provided by Ebrahimi [2011], Chaudhry et al. [2015] developed a dynamic deformation model with boundary conditions and using finite difference solution to solve the ordinary differential equation of this dynamic deformation model. Bian et al. [2018] introduced an automatic rigging system for curve-based skin deformation method proposed by Chaudhry et al. [2015] where the skeletons are created automatically. Liang et al. [2015] also uses elastic beam deformation model to approximate the shape of Chinese Marionette character's head. To solve the high order linear homogeneous ordinary differential equation with boundary constraints, they use complementary function to find the analytical solution of the profile curve mathematical description. Furthermore, they provided an editing technique by subdividing existing surface along with local refinement functions to maintain the positional and tangential constraints. Together, this editing and refinement method can refine the shape locally within a patch or close to a boundary region. Bian et al. [2019] proposed a example-driven, PDE-based physics model to describe the skin deformation in animation. They invented an extract isoparametric curves from trianglated model of pose 0 and use Fourier series representation for the extracted curve network, followed by applying Linear Blending Skinning to deform the model from input pose 0 to input pose 1. Then, by analysing the difference between the mesh after LBS to the mesh directly from input pose 1, the force that drives the deformation can be calculated. Finally, with the PDE-based physics model, the frames between pose 0 to pose 1 can be obtained.

To sum up the above discussion, differential equations are widely used to describe the physics model of geometric processing, both statically and dynamically. In addition, to facilitate the computational efficiency, one option is to choose the proper mathematical form for the geometry, to get analytical solution to the physics model is possible and the other is to take advantages of the light weightness of ODE to achieve real-time numerical solutions. Inspired by the state-of-art ODE-based geometric processing approaches, I introduce a new sculpting force as the disturbance to the physical system which balances the equation, and propose a new ODE-based FDM deformer to refine the course base mesh to accurately fit the input sketch silhouettes. To my knowledge, this is the first time ODE has been applied to solve a static (a.k.a. no time-related parameter is involved in the equation) modeling task.

Chapter 3

Curve Fitting for Cross-Section

The study of how cross-sectional contours help to define the human body can date back to 1500s. A German painter and artist Albrecht Dürer Dürer [1534] attempted to apply a selection of cross-sectional contours of human body to study human figures in his publications Four Books on Human Proportion (Vier Bücher von Menschlicher Proportion) of 1528. For the last two decades, the advances of medical imaging applications have encouraged the development of techniques for reconstruction digital human organ models from cross-sectional curves obtained by MRI, CT, and other medical scanning devices (Boissonnat & Memari [2007]). In this chapter, I will investigate a new approach to get the approximation mathematical formation of a given cross-section point list.

3.1 Generalized ellipses

In the work of Hyun et al. [2005], sweep surfaces with elliptic cross section have been used to approximate human arms, legs, torso and neck, and carry out human modelling and deformation. In their work, the mathematical description of sweep surfaces has the form of

$$\begin{aligned} S(u,v) &= R(u)E_u(v) + C(u) \\ &= \begin{bmatrix} r_{11}(u) & r_{12}(u) & r_{13}(u) \\ r_{21}(u) & r_{22}(u) & r_{23}(u) \\ r_{31}(u) & r_{32}(u) & r_{33}(u) \end{bmatrix} \times \begin{bmatrix} a\cos v \\ b\sin v \\ 0 \end{bmatrix} + \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} \end{aligned} \tag{3.1}$$

where S(u,v) is a sweep surface, $R(u)$ and $C(u)$ stand for rotation and translation, respectively, and $E_u(v)$ is a standard ellipse of variable size.

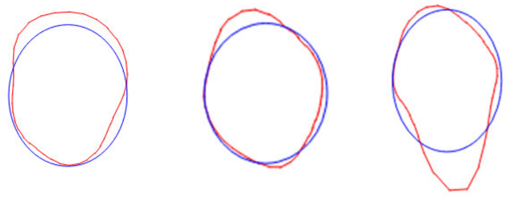
Since the cross sections of most parts of human body are irregular curves. Using standard ellipses to approximate these cross sections will bring in large differences leading to unrealistic human modelling.

Fig. 3.1 gives some cross-sectional curves of legs, arms, and torso from a human model built with the polygon modelling approach and those approximated by elliptic cross sections: Fig. 3.1a is taken from human right leg, Fig. 3.1b is from human left arm, and Fig. 3.1c is from human torso. It can be seen for these images that there are noticeable differences between the real cross sections and elliptic ones.

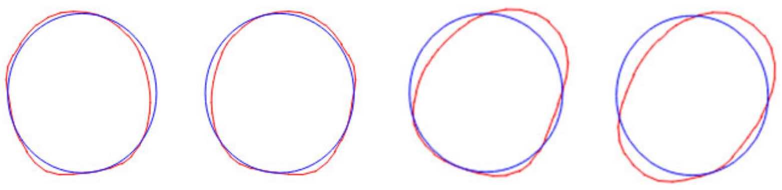
In addition, for a certain value of the parametric variable , Eq.(3.1) can be written as the following form

$$\begin{aligned} S_x(u_0,v) &= r_{11}(u_0)\cos v + r_{12}(u_0)\sin v + x(u_0) \\ S_y(u_0,v) &= r_{21}(u_0)\cos v + r_{22}(u_0)\sin v + y(u_0) \\ S_z(u_0,v) &= r_{31}(u_0)\cos v + r_{32}(u_0)\sin v + z(u_0) \end{aligned} \tag{3.2}$$

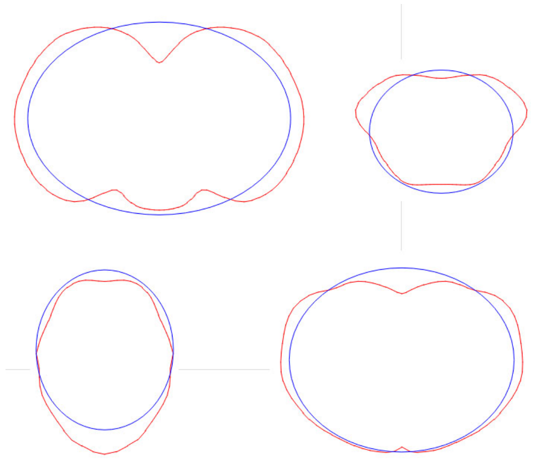
The above equation indicates that at the plane determined by a point $C(u_0)$



(a) *Cross section curves of left leg*



(b) *Cross section curves of right arm*



(c) *Various cross section curves of torso*

Figure 3.1: *Comparison between scanned real human cross sections and approximated ellipses*

and a unit normal vector $N(u_0) = \left[r_{13}(u_0), r_{23}(u_0), r_{33}(u_0)\right]^T$, the cross section of the sweep surface is still a simple curve which is described with two trigonometric functions $\cos v$ and $\sin v$.

In order to characterize the cross sections of human body more accurately, I introduce the conception of generalized elliptic curves which consist of a number of trigonometric functions $\cos jv$ and $\sin jv$ $(j = 0, 1, \ldots, 2J)$. With the application of more trigonometric functions, the cross sections of human body can be approximated very accurately.

Generalized elliptic curves can be divided into two classes: generalized ellipses and composite generalized elliptic segments. The former is used for the approximation cross section curves with a simple shape, and the latter aims at those with very complicated shapes. In the following, I will introduce the construction of generalized ellipses. The generation of composite generalized elliptic segments will be discussed in Section 3.2.

In the work of You et al. [2004], trigonometric series of two parametric variables has been applied to describe blending surfaces. By degenerating two dimensional problems to one dimensional ones, the trigonometric series in You et al. [2004] can be modified to represent cross section curves of human body. Assuming that a generalized ellipse is lying on one of x-y, x-z and y-z plane, with the centre $C = \left[x_c, y_c, z_c\right]$, and taking the one lying on x-y plane as an example, the mathematical equation of the generalized ellipse can be written as

$$\begin{aligned}
x(v) - x_c &= a_0 + \sum_{j=1}^{J} \left(a_{2j-1} \cos(j \times v) + a_{2j} \sin(j \times v)\right) \\
y(v) - y_c &= b_0 + \sum_{j=1}^{J} \left(b_{2j-1} \sin(j \times v) + b_{2j} \cos(j \times v)\right) \\
z(v) - z_c &= 0
\end{aligned} \tag{3.3}$$

where a_j and $b_j (j = 0, 1, 2, 3, \ldots, 2J)$ are unknown constants.

If there is a cross section curve represented by a number of discrete points $(x_i, y_i, z_c)(i = 1, 2, 3, \ldots, I)$, the centre of the generalized ellipse can be determined by the average value of each component x and y. That is

$$\begin{aligned}
x_c &= \frac{1}{I} \sum_{i=1}^{I} x_i \\
y_c &= \frac{1}{I} \sum_{i=1}^{I} y_i
\end{aligned} \tag{3.4}$$

Then, I use curve fitting and the least square algorithm to determine the unknown constants in Eq. (3.3). To this aim, I calculate the squares sum of the errors between the curve and the generalized ellipse at the points $(x_i, y_i, z_c)(i = 1, 2, 3, \ldots, I)$ for x and y position components, respectively. Theoretically the distance error should calculate the distance between the point and the curve. But practically, it does not make much difference. Hence, here the distance is calculated between corresponding points.

$$\begin{aligned}
E_x &= \sum_{i=1}^{I} \left[x_i - x_c - a_0 - \sum_{j=1}^{J} \left(a_{2j-1} \cos(j \times v_i) + a_{2j} \sin(j \times v_i)\right)\right]^2 \\
E_y &= \sum_{i=1}^{I} \left[y_i - y_c - b_0 - \sum_{j=1}^{J} \left(b_{2j-1} \sin(j \times v_i) + b_{2j} \cos(j \times v_i)\right)\right]^2
\end{aligned} \tag{3.5}$$

where $v_i (i = 1, 2, 3, \ldots, I)$ is the angle of point p_i on the curve. And the procedure of calculating v_i is like the following: 1. find the center of the curve; 2. draw a line l_i starts from the the center and directs to point p_i; 3. get the cosine value a_i of the angle v_i between l_i and the x-axis; 4. based on different quadrant the angle v_i belongs to, decide the value of v_i by calculating the arc cosine of a_i.

$$
\begin{aligned}
&a_i = \frac{p_i.y - center.y}{\sqrt{(p_i.y - center.y)^2 + (p_i.x - center.x)^2}} \\
&v_i = \arccos(a_i) \\
&(i = 1, 2, 3, \ldots, I)
\end{aligned}
\tag{3.6}
$$

The errors in Eq. (3.5) are minimized by setting the derivatives of the square sums with respect to the unknown constants to zero

$$
\begin{aligned}
\frac{\partial E_x}{\partial a_j} &= 0 \\
\frac{\partial E_y}{\partial b_j} &= 0 \\
(j = 0, 1, 2, 3, &\ldots, 2J)
\end{aligned}
\tag{3.7}
$$

which leads to the following linear algebraic equations

$$\sum_{i=1}^{I}(x_i - x_c) - \sum_{i=1}^{I} a_0 - \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j-1}\cos(j \times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j}\sin(j \times v_i) = 0$$

$$
\begin{aligned}
&\sum_{i=1}^{I}(x_i - x_c)\cos(k \times v_i) - a_0\sum_{i=1}^{I}\cos(k \times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j-1}\cos(j \times v_i)\cos(k \times v_i) \\
&- \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j}\sin(j \times v_i)\cos(k \times v_i) = 0
\end{aligned}
$$

$$\sum_{i=1}^{I}(x_i - x_c)\sin(k\times v_i) - a_0\sum_{i=1}^{I}\sin(k\times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I}a_{2j-1}\cos(j\times v_i)\sin(k\times v_i)$$
$$-\sum_{j=1}^{J}\sum_{i=1}^{I}a_{2j}sin(j\times v_i)\sin(k\times v_i) = 0$$
$$(k = 1,2,3,\ldots,J) \tag{3.8}$$

and

$$\sum_{i=1}^{I}(y_i - y_c) - \sum_{i=1}^{I}b_0 - \sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j-1}\sin(j\times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j}\cos(j\times v_i) = 0$$

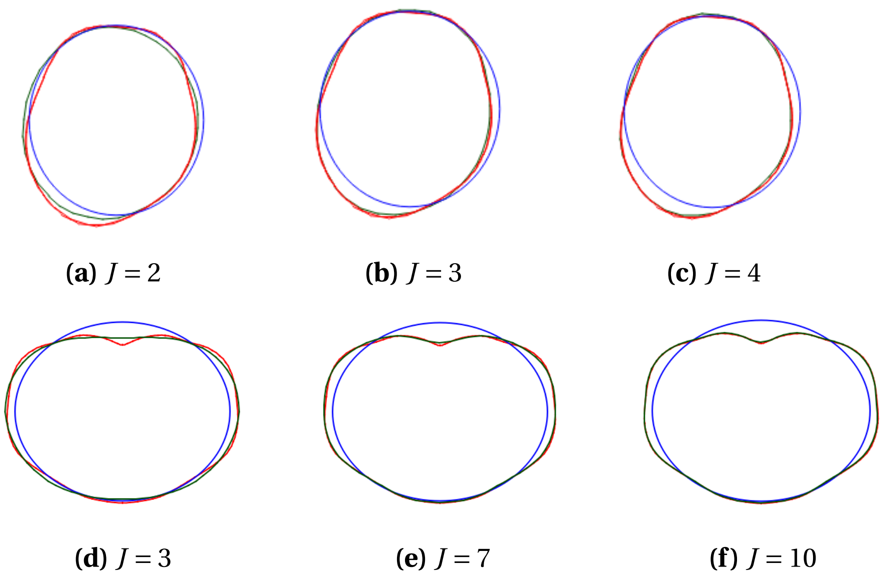
$$\sum_{i=1}^{I}(y_i - y_c)\sin(k\times v_i) - b_0\sum_{i=1}^{I}\sin(k\times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j-1}\sin(j\times v_i)\sin(k\times v_i)$$
$$-\sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j}\cos(j\times v_i)\sin(k\times v_i) = 0$$

$$\sum_{i=1}^{I}(y_i - y_c)\cos(k\times v_i) - b_0\sum_{i=1}^{I}\cos(k\times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j-1}\sin(j\times v_i)\cos(k\times v_i)$$
$$-\sum_{j=1}^{J}\sum_{i=1}^{I}b_{2j}\cos(j\times v_i)\cos(k\times v_i) = 0$$
$$(k = 1,2,3,\ldots,J) \tag{3.9}$$

The matrix equations for (3.8) and (3.9) are presented in Appendix E. Solving those two equation systems, we determine all unknown constants and obtain the mathematical representation of the generalized ellipse.

For the curve which is not perpendicular to any coordinate planes, we must carry out the coordinate transformation and find the mathematical description of the curve in the local coordinate system where one of the coordinate axes is perpendicular to the curve plane.

For some representative curves taken from human leg and torso, we use the generalized ellipse Eq. (3.3) and standard ellipse to regenerate them and the obtained results were depicted in Fig. 3.2 where J is defined in Eq.(3.3), the curve in red is the original one, the one in green is from the algorithm of the generalized ellipse, and that in blue is created with the equation of the standard ellipse.


(a) $J = 2$ **(b)** $J = 3$ **(c)** $J = 4$

(d) $J = 3$ **(e)** $J = 7$ **(f)** $J = 10$

Figure 3.2: *Curve generation with different approaches, with original cross section curve in red, standard ellipse in blue and generalized ellipse produced by Eq.*(3.3) *in green where (a), (b), and (c) show the cross section curves calculated by Eq.*(3.3) *with* $J = 2,3,4$, *respectively, and (d), (e), and (f) shows the cross section curves calculated by Eq.*(3.3) *with* $J = 3,7,10$, *respectively*

It can be seen from these figures that the algorithm of the generalized ellipse can approximate the cross section curves of human body quite well and the approximation can be greatly improved by increasing the terms of the generalized ellipse. Since most cross section curves of human body have the similar complexity to those in this figure, the proposed approach can use few terms - most human body cross-sectional curves can be represented by this trigonometric series with less than 30 terms ($J < 15$) involved, while using polygonal points it needs more than 50 sampled points for the simplest cross-sectional curve - to achieve high accuracy of curve modelling.

Due to different shapes of cross-sectional curves, different terms in Eq.(3.3) will be used to achieve the required accuracy. Too few terms in Eq.(3.3) will cause too large errors. Too many terms will increase the computational cost and slow down the human modelling process. Therefore, suitable terms should be used. However, in the situation of interactive modelling, it is inconvenient to find out which value of the term J is the best. In order to tackle this issue, I here propose the following strategy, and the results were shown in Fig.3.4.

First, I define an average error E_a and a maximum error E_m below, with the help of D - the diagonal distance of the curve's bounding box. The former is used to measure the global approximation of the generalized ellipse to the original curve and the latter quantifies the maximum difference between the generalized ellipse and the original curve.

$$E_a = \frac{1}{I}\sum_{i=1}^{I}\frac{d_i}{D} \tag{3.10}$$

where

$$
\begin{aligned}
X &= x_i - x_c - a_0 - \sum_{j=1}^{J} \left(a_{2j-1}\cos(j \times v_i) + a_{2j}\sin(j \times v_i)\right)\\
Y &= y_i - y_c - b_0 - \sum_{j=1}^{J} \left(b_{2j-1}\sin(j \times v_i) + b_{2j}\cos(j \times v_i)\right)\\
d_i &= \sqrt{X^2 + Y^2}
\end{aligned}
\tag{3.11}
$$

and

$$E_m = max\{\frac{d_1}{D}, \frac{d_2}{D}, \frac{d_3}{D}, \ldots, \frac{d_J}{D}\} \tag{3.12}$$

Then, I can set different errors to determine the required terms in Eq.(3.3). For example, I take the average error and maximum error not more than 1% and 2.5%, respectively. That is

$$
\begin{aligned}
E_a &\leq 1\%\\
E_m &\leq 2.5\%
\end{aligned}
\tag{3.13}
$$

Fig. 3.3 illustrates the procedure of finding a proper J: A linear interpolation operation is employed to find out the suitable terms efficiently. Initially, take $J = 3$ and $J = 10$, and calculate the average errors E_{a3} and E_{a10} and the maximum errors E_{m3} and E_{m10} where the subscripts 3 and 10 stands for the values of J. If both E_{a3} and E_{m3} have met Eq. (3.13), or one of E_{a10} and E_{m10} does not satisfy Eq. (3.13), a linear extrapolation operation is used to find a smaller J_1 for the former and a larger J_1 for the latter. Otherwise, a linear interpolation is applied to calculate the J_1 between $J = 3$ and $J = 10$. The obtained J_1 is usually not an integer. Round off it to the nearest integer. With J_1, calculate the average errors E_{aJ_1} and the maximum errors E_{mJ_1} and examine whether they have met Eq. (3.13). If both E_{aJ_1} and E_{mJ_1}

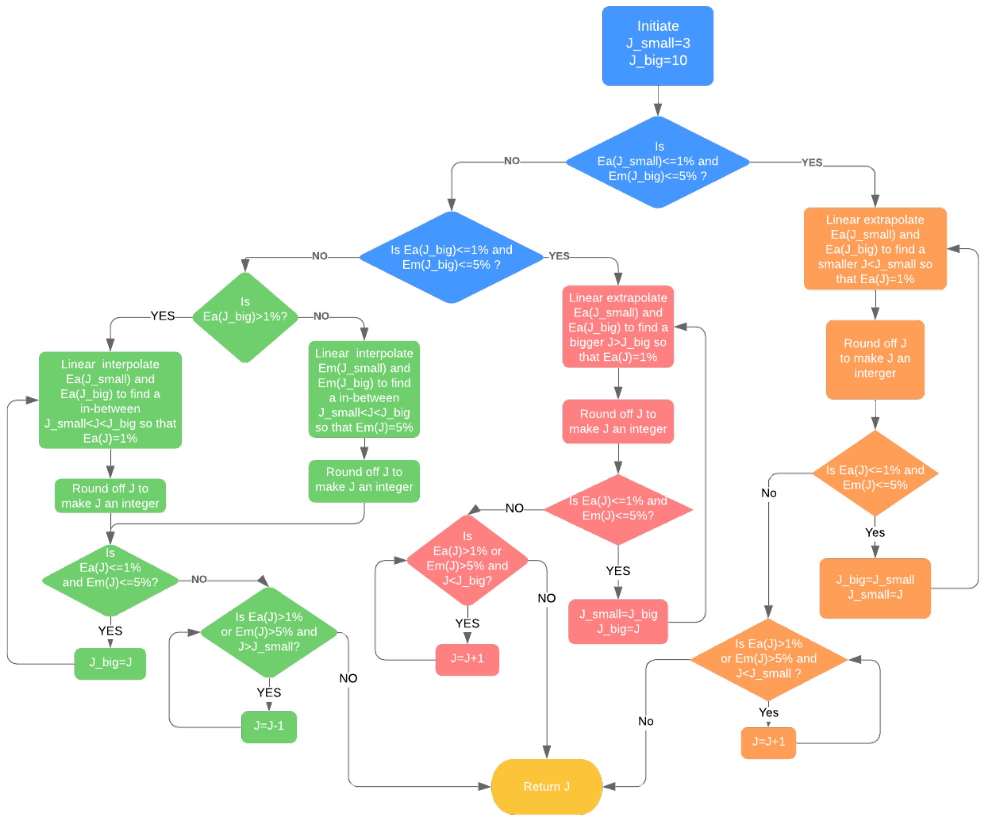
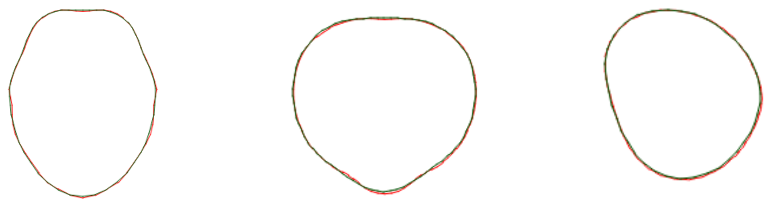

Initiate J_small=3 J_big=10
Is Ea(J_small)<=1% and Em(J_big)<=5% ?
NO
YES
Is Ea(J_big)<=1% and Em(J_big)<=5% ?
NO
YES
Linear extrapolate Ea(J_small) and Ea(J_big) to find a smaller J<J_small so that Ea(J)=1%
Is Ea(J_big)>1%?
YES
NO
Linear extrapolate Ea(J_small) and Ea(J_big) to find a bigger J>J_big so that Ea(J)=1%
Round off J to make J an interger
Linear interpolate Ea(J_small) and Ea(J_big) to find a in-between J_small<J<J_big so that Ea(J)=1%
Linear interpolate Em(J_small) and Em(J_big) to find a in-between J_small<J<J_big so that Em(J)=5%
Round off J to make J an integer
Is Ea(J)<=1% and Em(J)<=5%
No
Yes
Round off J to make J an integer
Round off J to make J an integer
NO
Is Ea(J)<=1% and Em(J)<=5%?
Is Ea(J)<=1% and Em(J)<=5%?
NO
Is Ea(J)>1% or Em(J)>5% and J<J_big?
J_big=J_small J_small=J
YES
NO
YES
Is Ea(J)>1% or Em(J)>5% and J>J_small?
YES
J_small=J_big J_big=J
J_big=J
J=J+1
Is Ea(J)>1% or Em(J)>5% and J<J_small ?
YES
NO
No
Yes
J=J-1
J=J+1
Return J

Figure 3.3: *Program flowchart of finding a proper J*

have met Eq. (3.13) and $J_1 < 3$, J is between J_1 and $J = 3$. If one of E_{aJ_1} and E_{mJ_1} does not satisfy Eq. (3.13) and $J_1 > 10$, J is between $J = 10$ and J_1. If both E_{aJ_1} and E_{mJ_1} have met Eq. (3.13) and $3 < J_1 < 10$, J is between $J = 3$ and J_1. If one of E_{aJ_1} and E_{mJ_1} does not satisfy Eq. (3.13) and $3 < J_1 < 10$, J is between J_1 and $J = 10$. It is worth to be noted that, since the human body cross-sectional curves in this study aren't very complicated, I found that after the first linear interpolation or extrapolation operation, it's more efficient if using $J = J + 1$ or $J = J - 1$ to reach the result, as shown in Fig.3.3.

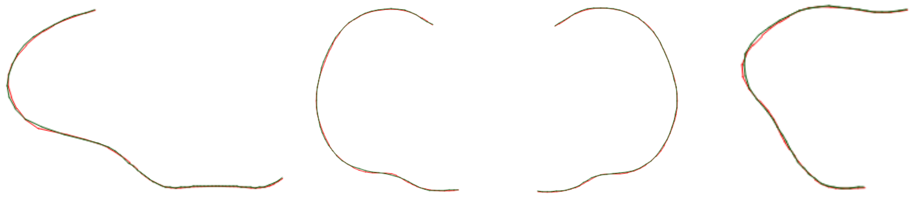
Since there are two quantities E_a and E_m which can be employed for the interpolation, which one should be used must be determined. Obviously, if only one of E_a and E_m does not meet Eq. (3.13), this error is used for the interpolation. If both of them do not satisfy Eq. (3.13), always use the average error for the interpolation since the average error is a global measurement of the difference between the original curve and the generalized ellipse.



(a) *head cross-section J=7, Ea=0.93%, Em=2.45%* **(b)** *neck cross-section J=6, Ea=0.79%, Em=2.15%* **(c)** *arm cross-section J=4, Ea=0.70%, Em=1.18%*

Figure 3.4: *Head, neck and arm cross-sectional curves in red, and their correspondent generalized ellipses produced by Eq.*(3.3) *with optimised J in green*

This trigonometric series representation is not only suitable for closed curves but also capable of representing open curves, as shown in Fig. 3.5.



(a) *J=10 Ea=0.19%, Em=2.24%* **(b)** *J=6 Ea=0.29%, Em=1.82%* **(c)** *J=8 Ea=0.08%, Em=0.70%* **(d)** *J=6 Ea=0.46%, Em=2.00%*

Figure 3.5: *Open curves in red, and their correspondent generalised curves produced by Eq.*(3.3) *with optimised J in green*

3.2 Split generalized ellipse into smoothly connected segments

A complete curve representation method should support curve segmentation. Hence, in this section, I will discuss the approach to split a generalized ellipse into multiple curve segments with the aforementioned trigonometric series representation discussed in Section.3.1 with C^1 continuity between adjacent segments.

A generalized ellipse can be decomposed into several simpler ones. In order to approximate these decomposed curve segments, let's still use Eq. (3.3) to represent new curve segments and call them generalized elliptic segments. The curve consisting of these generalized elliptic segments is named a composite generalized elliptic curve.

To guarantee two adjacent generalized elliptic segments are smoothly connected together, I investigate the continuity between these two segments.

As an example, let's consider how to use three generalized elliptic segments to approximate a closed cross section curve (Fig.3.6) with a complicated shape and how to guarantee the tangential continuity between the two adjacent generalized elliptic segments. For the curves consisting of more than three generalized elliptic segments and with higher order continuity between connected generalized elliptic segments, the basic principle is the same.

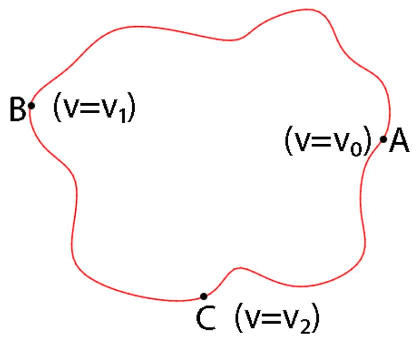


Figure 3.6: *Three segments of a randomly selected curve*

3.2.1 Connection of two adjacent generalized elliptic segments

Apart from the curve shown in Fig.3.6 which is the same as that in Fig.3.7, let's also consider a curve shown in Fig.3.9 and a cross section curve from human hip shown in Fig.3.10. Firstly, divide a cross section curve from human hip into three separate curve segments $\widehat{AB}$, $\widehat{BC}$ and $\widehat{CA}$ as indicated in Fig. 3.10. With Eq. (3.3) and curve fitting algorithm Eq. (7.1) and Eq. (7.2), the first generalized elliptic segment corresponding to the curve segment $\widehat{AB}$ which is defined within the region of the parametric variable v from $v = v_0$ to $v = v_1$ can be determined, see Fig.3.7a.

The next task is to determine the second generalized elliptic segment corresponding to the curve segment $\widehat{BC}$ which is defined in the region of the parametric variable v from $v = v_1$ to $v = v_2$. Using a bar to indicate the second elliptic segment, the positional values of the first generalized elliptic segment at $v = v_1$, as stated in Eq.(3.14), should be equal to that of the second generalized elliptic segment at $v = v_1$. Therefore, we have the following positional continuity conditions Eq.(3.15) for the x and y position functions. Note that since the first segment is already generated, P_x, P_y,

T_x, T_y are known constants.

$$\begin{aligned}
P_x &= x_c + a_0 + \sum_{j=1}^{J} a_{2j-1}\cos(j\times v_1) + a_{2j}\sin(j\times v_1)\\
P_y &= y_c + b_0 + \sum_{j=1}^{J} b_{2j-1}\sin(j\times v_1) + b_{2j}\cos(j\times v_1)\\
T_x &= \sum_{j=1}^{J} -a_{2j-1}\times j\times\sin(j\times v_1) + a_{2j}\times j\times\cos(j\times v_1)\\
T_y &= \sum_{j=1}^{J} b_{2j-1}\times j\times\cos(j\times v_1) - b_{2j}\times j\times\sin(j\times v_1)
\end{aligned}\tag{3.14}$$

$$\begin{aligned}
P_x &= \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1}\cos(j\times v_1) + \bar{a}_{2j}\sin(j\times v_1)\\
P_y &= \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1}\sin(j\times v_1) + \bar{b}_{2j}\cos(j\times v_1)\\
T_x &= \sum_{j=1}^{\bar{J}} -\bar{a}_{2j-1}\times j\times\sin(j\times v_1) + \bar{a}_{2j}\times j\times\cos(j\times v_1)\\
T_y &= \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1}\times j\times\cos(j\times v_1) - \bar{b}_{2j}\times j\times\sin(j\times v_1)
\end{aligned}\tag{3.15}$$

Re-arrange Eq.(3.15), we can get the formula for $\bar{a}_0$, $\bar{a}_1$, $\bar{b}_0$ and $\bar{b}_1$ as follows. Since the denominator cannot be zero, we specify that $v_1 \neq 0$ and $v_1 \neq \pi$, so that $\sin v_1$ doesn't equal zero. As for the cases when $v_1 = 0$ and

$v_1 = \pi$, I will discuss both scenarios later in this section, respectively.

$$\begin{aligned}
\bar{a}_0 =& \bar{P}_x - \bar{x}_c + T_x \times \frac{\cos v_1}{\sin v_1} + \sum_{j=2}^{\bar{J}} \bar{a}_{2j-1} \times \left[j \times \frac{\sin(j \times v_1)}{\sin v_1} \times \cos v_1 - \cos(j \times v_1) \right] \\
&+ \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times \left[j \times \frac{\cos(j \times v_1)}{\sin v_1} \times \cos v_1 - \sin(j \times v_1) \right] \\
\bar{a}_1 =& -\frac{\bar{T}_x}{\sin v_1} - \sum_{j=2}^{\bar{J}} \bar{a}_{2j-1} \times j \times \frac{\sin(j \times v_1)}{\sin v_1} + \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times j \times \frac{\cos(j \times v_1)}{\sin v_1} \\
\bar{b}_0 =& \bar{P}_y - \bar{y}_c - T_y \times \frac{\sin v_1}{\cos v_1} + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times \left[j \times \sin v_1 \times \frac{\cos(j \times v_1)}{\cos v_1} - \sin(j \times v_1) \right] \\
&- \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times \left[j \times \frac{\sin(j \times v_1)}{\cos v_1} \times \sin v_1 + \cos(j \times v_1) \right] \\
\bar{b}_1 =& \frac{\bar{T}_y(v_1)}{\cos v_1} - \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times j \times \frac{\cos(j \times v_1)}{\cos v_1} + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times j \times \frac{\sin(j \times v_1)}{\cos v_1}
\end{aligned} \tag{3.16}$$

Substituting Eq.(3.16) into Eq.(3.3), $\bar{x}$ and $\bar{y}$ become functions of $\bar{a}_2$,

$\bar{a}_3,\ldots,\bar{a}_{2\bar{J}}$ and $\bar{b}_2$, $\bar{b}_3,\ldots,\bar{b}_{2\bar{J}}$, respectively.

$$\begin{aligned}\bar{x}(v) =& \bar{P}_x + \bar{T}_x \times \frac{\cos v_1 - \cos v}{\sin v_1} \\ &+ \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times \left[j \times \frac{\sin(j \times v_1)}{\sin v_1} \times (\cos v_1 - \cos v) - \cos(j \times v_1) + \cos(j \times v) \right] \\ &+ \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times \left[j \times \frac{\cos(j \times v_1)}{\sin v_1} \times (\cos v - \cos v_1) + \sin(j \times v) - \sin(j \times v_1) \right]\end{aligned}$$

$$\begin{aligned}\bar{y}(v) =& \bar{P}_y + \bar{T}_y \times \frac{\sin v - \sin v_1}{\cos v_1} \\ &+ \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times \left[j \times \frac{\cos(j \times v_1)}{\cos v_1} \times (\sin v_1 - \sin v) - \sin(j \times v_1) + \sin(j \times v) \right] \\ &+ \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times \left[j \times \frac{\sin(j \times v_1)}{\cos v_1} \times (\sin v - \sin v_1) - \cos(j \times v_1) + \cos(j \times v) \right]\end{aligned} \tag{3.17}$$

Since the first point of generalized elliptic segment is fixed, the $\bar{E}_x$ and $\bar{E}_y$ can be defined as the squared sum of the errors between the curve and the generalized elliptic segment from the second point to the last point ($i =$

$2,3,\ldots,\bar{I}$) for x and y position components, respectively.

$$\begin{aligned}
E_{\bar{x}} &= \sum_{i=2}^{\bar{I}} \left[\bar{G}_x(i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j-1} \times \bar{M}_x(j,\bar{v}_i) + \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times \bar{N}_x(j,\bar{v}_i)\right]^2 \\
E_{\bar{y}} &= \sum_{i=2}^{\bar{I}} \left[\bar{G}_y(i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times \bar{M}_x(j,\bar{v}_i) + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times \bar{N}_y(j,\bar{v}_i)\right]^2
\end{aligned}$$

where

$$\begin{aligned}
&\bar{G}_x(i) = \bar{P}_x + \frac{\cos v_1 - \cos \bar{v}_i}{\sin v_1} \times \bar{T}_x - \bar{x}_i \\
&\bar{M}_x(j,\bar{v}_i) = j \times \frac{\sin(j \times v_1)}{\sin v_1} \times (\cos v_1 - \cos \bar{v}_i) - \cos(j \times v_1) + \cos(j \times \bar{v}_i) \\
&\bar{N}_x(j,\bar{v}_i) = j \times \frac{\cos(j \times v_1)}{\sin v_1} \times (\cos \bar{v}_i - \cos v_1) - \sin(j \times v_1) + \sin(j \times \bar{v}_i) \\
&\bar{G}_y(i) = \bar{P}_y + \frac{\sin \bar{v}_i - \sin v_1}{\cos v_1} \times \bar{T}_y - \bar{y}_i \\
&\bar{M}_y(j,\bar{v}_i) = j \times \frac{\cos(j \times v_1)}{\cos v_1} \times (\sin v_1 - \sin \bar{v}_i) - \sin(j \times v_1) + \sin(j \times \bar{v}_i) \\
&\bar{N}_y(j,\bar{v}_i) = j \times \frac{\sin(j \times v_1)}{\cos v_1} \times (\sin \bar{v}_i - \sin v_1) - \cos(j \times v_1) + \cos(j \times \bar{v}_i)
\end{aligned} \tag{3.18}$$

The errors in Eq.(3.18) are minimized by setting the derivatives of the squared sums with respect to the unknown constants to zero

$$\begin{aligned}
\frac{\partial E_{\bar{x}}}{\partial \bar{a}_j} &= 0 \\
\frac{\partial E_{\bar{y}}}{\partial \bar{b}_j} &= 0 \\
(j = 2,3,\ldots,2J)&
\end{aligned} \tag{3.19}$$

which leads to the following linear algebraic equations

$$\sum_{i=2}^{\bar{I}} \left[\bar{G}_x(i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j-1} \times \bar{M}_x(j, \bar{v}_i) + \sum_{j=1}^{\bar{J}} a_{2j} \times \bar{N}_x(j, \bar{v}_i) \right] \times \bar{M}_x(\frac{k+1}{2}, \bar{v}_i) = 0$$

$(k = 3, 5, \ldots, 2\bar{J} - 1)$

$$\sum_{i=2}^{\bar{I}} \left[\bar{G}_x(i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j-1} \times \bar{M}_x(j, \bar{v}_i) + \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times \bar{N}_x(j, \bar{v}_i) \right] \times \bar{N}_x(\frac{k}{2}, \bar{v}_i) = 0$$

$(k = 2, 4, \ldots, 2\bar{J})$

$$\sum_{i=2}^{\bar{I}} \left[\bar{G}_y(i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times \bar{M}_y(j, \bar{v}_i) + \sum_{j=1}^{\bar{J}} b_{2j} \times \bar{N}_y(j, \bar{v}_i) \right] \times \bar{M}_y(\frac{k+1}{2}, \bar{v}_i) = 0$$

$(k = 3, 5, \ldots, 2\bar{J} - 1)$

$$\sum_{i=2}^{\bar{I}} \left[\bar{G}_y(i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times \bar{M}_y(j, \bar{v}_i) + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times \bar{N}_y(j, \bar{v}_i) \right] \times \bar{N}_y(\frac{k}{2}, \bar{v}_i) = 0$$

$(k = 2, 4, \ldots, 2\bar{J})$

(3.20)

Solving the above Eq.(3.20), we can get $\bar{a}_2, \bar{a}_3, \ldots, \bar{a}_{2\bar{J}}$ and $\bar{b}_2, \bar{b}_3, \ldots, \bar{b}_{2\bar{J}}$, with which we obtain the mathematical description of the second generalized elliptic segments required by Eq.(3.17), and then draw the segment accordingly (Fig.3.7b).

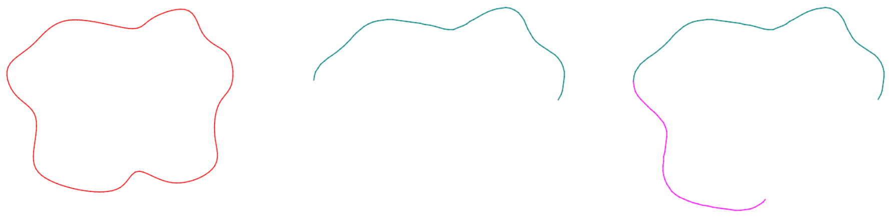

(a) *original curve* **(b)** *first segment* **(c)** *second segment*

Figure 3.7: *The generation of two adjacent segments*

Special case 1: Connection of two adjacent generalized elliptic segments at the joint point v=0

When the second segment $\widehat{BC}$ starts at $v_1 = 0$ (Fig.3.8), where $\bar{P}_x$, $\bar{P}_y$, $\bar{T}_x$ and $\bar{T}_y$ can be calculated as Eq.(3.21). And the positional and tangential constraints becomes Eq.(3.22)

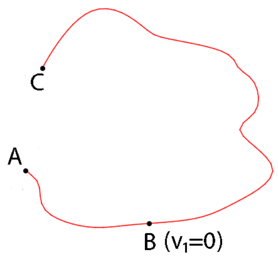


Figure 3.8: *Two adjacent segments connected at* $v_1 = 0$

$$\begin{aligned} \bar{P}_x &= x_c + a_0 + \sum_{j=1}^{J} a_{2j-1} \\ \bar{P}_y &= y_c + b_0 + \sum_{j=1}^{J} b_{2j} \\ \bar{T}_x &= \sum_{j=1}^{J} a_{2j} \times j \\ \bar{T}_y &= \sum_{j=1}^{J} b_{2j-1} \times j \end{aligned} \tag{3.21}$$

$$\begin{aligned}
\bar{P}_x &= \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \\
\bar{P}_y &= \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \\
\bar{T}_x &= \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times j \\
\bar{T}_y &= \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1} \times j
\end{aligned} \tag{3.22}$$

Re-arranging Eq.(3.22), we can get the formula for $\bar{a}_0$,$\bar{a}_2$,$\bar{b}_0$ and $\bar{b}_1$ as follows

$$\begin{aligned}
\bar{a}_0 &= \bar{P}_x - \bar{x}_c - \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \\
\bar{a}_2 &= \bar{T}_c - \sum_{j=2}^{\bar{J}} \bar{a}_{2j} \times j \\
\bar{b}_0 &= \bar{P}_y - \bar{y}_c - \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \\
\bar{b}_1 &= \bar{T}_y - \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times j
\end{aligned} \tag{3.23}$$

Substitute Eq.(3.23) into Eq.(3.3), $\bar{x}$ and $\bar{y}$ should become

$$\begin{aligned}
\bar{x}(v) &= \bar{P}_x + \bar{T}_x + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times [\cos(j \times v) - 1] + \sum_{j=2}^{\bar{J}} \bar{a}_{2j} [\sin(j \times v) - j \times \sin v] \\
\bar{y}(v) &= \bar{P}_y + \sin v \times \bar{T}_y + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times [\sin(j \times v) - j \times \sin v] + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times [\cos(j \times v) - 1]
\end{aligned} \tag{3.24}$$

As mentioned before, $E_{\bar{x}}$ and $E_{\bar{y}}$ are the squared sums of errors between the original curve and the generalized elliptic segment from the second point to the last point. Since the first point position serves as boundary constraint, it should not contribute to errors.

$$E_{\bar{x}} = \sum_{i=2}^{\bar{I}} [\bar{x}_i - \bar{x}(\bar{v}_i)]^2 = \sum_{i=2}^{\bar{I}} [\bar{G}_x(i) + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \bar{M}_x(j, \bar{v}_i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j} \times \bar{N}_x(j, \bar{v}_i)]^2$$

$$E_{\bar{y}} = \sum_{i=2}^{\bar{I}} [\bar{y}_i - \bar{y}(v_i)]^2 = \sum_{i=2}^{\bar{I}} [\bar{G}_y(i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \bar{M}_y(j, \bar{v}_i) + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times \bar{N}_y(j, \bar{v}_i)]^2$$

$where$

$$\bar{G}_x(i) = \bar{P}_x + \bar{T}_x \times \sin \bar{v}_i - \bar{x}_i$$

$$\bar{M}_x(j, \bar{v}_i) = \cos(j \times \bar{v}_i) - 1$$

$$\bar{N}_x(j, \bar{v}_i) = \sin(j \times \bar{v}_i) - j \times \sin \bar{v}_i$$

$$\bar{G}_y(i) = \bar{P}_y + \bar{T}_y \times \sin \bar{v}_i - \bar{y}_i$$

$$\bar{M}_y(j, \bar{v}_i) = \sin(j \times \bar{v}_i) - j \times \sin \bar{v}_i$$

$$\bar{N}_y(j, \bar{v}_i) = \sin(j \times \bar{v}_i) - j \times \sin \bar{v}_i$$

(3.25)

In order to minimize the $E_{\bar{x}}$, we take the partial derivative of $E_{\bar{x}}$ with respect to every $\bar{a}_j, (j = 1, 3, 4, \ldots, 2\bar{J})$ to be zero. The same goes with $\bar{y}$, i.e. the partial derivative of $E_{\bar{y}}$ with respect to every $\bar{b}_j, (j = 2, 3, 4, \ldots, 2\bar{J})$ should be zero, too.

$$\frac{\partial E_x}{\partial a_j} = 0$$

$$(j = 1, 3, 4, \ldots, 2\bar{J})$$

$$\frac{\partial E_y}{\partial b_j} = 0 \tag{3.26}$$

$$(j = 2, 3, 4, \ldots, 2\bar{J})$$

which leads to the following linear algebraic equations

$$\sum_{i=2}^{\bar{I}}[\bar{G}_x(i)+\sum_{j=1}^{\bar{J}}\bar{a}_{2j-1}\times\bar{M}_x(j,\bar{v}_i)+\sum_{j=2}^{\bar{J}}\bar{a}_{2j}\times\bar{N}_x(j,\bar{v}_i)]\times\bar{M}_x(\frac{k+1}{2},\bar{v}_i)=0$$

$(k=1,3,5,\ldots,2\bar{J}-1)$

$$\sum_{i=2}^{\bar{I}}[\bar{G}_x(i)+\sum_{j=1}^{\bar{J}}\bar{a}_{2j-1}\times\bar{M}_x(j,\bar{v}_i)+\sum_{j=2}^{\bar{J}}\bar{a}_{2j}\times\bar{N}_x(j,\bar{v}_i)]\times\bar{N}_x(\frac{k}{2},\bar{v}_i)=0$$

$(k=4,6,\ldots,2\bar{J})$

$$\sum_{i=2}^{\bar{I}}[\bar{G}_y(i)+\sum_{j=1}^{\bar{J}}\bar{b}_{2j-1}\times\bar{M}_y(j,\bar{v}_i)+\sum_{j=2}^{\bar{J}}\bar{b}_{2j}\times\bar{N}_y(j,\bar{v}_i)]\times\bar{M}_y(\frac{k+1}{2},\bar{v}_i)=0$$

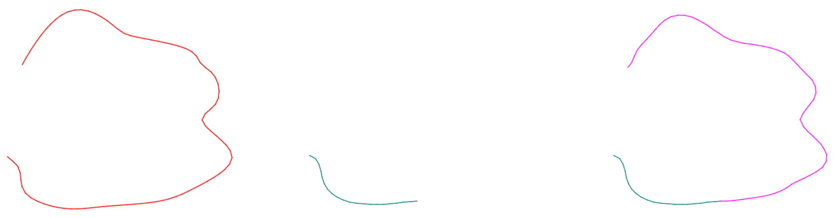
$(k=3,5,\ldots,2\bar{J}-1)$

$$\sum_{i=2}^{\bar{I}}[\bar{G}_y(i)+\sum_{j=1}^{\bar{J}}\bar{b}_{2j-1}\times\bar{M}_y(j,\bar{v}_i)+\sum_{j=2}^{\bar{J}}\bar{b}_{2j}\times\bar{N}_y(j,\bar{v}_i)]\times\bar{N}_y(\frac{k}{2},\bar{v}_i)=0$$

$(k=2,4,\ldots,2\bar{J})$

(3.27)

Eq.(3.27) determines $\bar{a}_1,\bar{a}_3,\bar{a}_4,\bar{a}_5,\ldots,\bar{a}_{2\bar{J}}$ and $\bar{b}_2,\bar{b}_3,\bar{b}_4,\ldots,\bar{b}_{2\bar{J}}$. Solving it, we get the mathematical description of the composite generalized segment. Its one end connected with the preceding segment at $v_1=0$, see Fig.3.9.



(a) *original curve* **(b)** *first segment* **(c)** *second segment*

Figure 3.9: *Two adjacent segments connected at $v=0$*

Special case 2: Connection of two adjacent generalized elliptic segments at the joint point $v = \pi$

When the second segment $\widehat{BC}$ starts at $v_1 = \pi$ (Fig.3.10), the positional and tangential constraints becomes

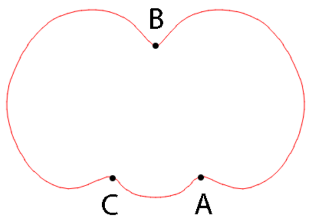


Figure 3.10: *Three segments of a hip cross-sectional curve*

$$
\begin{aligned}
\bar{P}_x &= x_c + a_0 + \sum_{j=1}^{J} a_{2j-1} \times (-1)^j \\
\bar{P}_y &= y_c + b_0 + \sum_{j=1}^{J} b_{2j} \times (-1)^j \\
\bar{T}_x &= \sum_{j=1}^{J} a_{2j} \times j \times (-1)^j \\
\bar{T}_y &= \sum_{j=1}^{J} b_{2j-1} \times (-1)^j \times j
\end{aligned}
\tag{3.28}
$$

$$\begin{aligned}
\bar{P}_x &= \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times (-1)^j \\
\bar{P}_y &= \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times (-1)^j \\
\bar{T}_x &= \sum_{j=1}^{\bar{J}} \bar{a}_{2j} \times j \times (-1)^j \\
\bar{T}_y &= \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1} \times (-1)^j \times j
\end{aligned} \tag{3.29}$$

Re-arranging Eq.(3.29), we could get the formula for $\bar{a}_0$,$\bar{a}_2$,$\bar{b}_0$ and $\bar{b}_1$

$$\begin{aligned}
\bar{a}_0 &= \bar{P}_x - \bar{x}_c - \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times (-1)^j \\
\bar{a}_2 &= -\bar{T}_x - \sum_{j=2}^{\bar{J}} \bar{a}_{2j} \times j \times (-1)^j \\
\bar{b}_0 &= \bar{P}_y - \bar{y}_c - \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times (-1)^j \\
\bar{b}_1 &= -\bar{T}_y + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times j \times (-1)^j
\end{aligned} \tag{3.30}$$

Substitute Eq.(3.30) into Eq.(3.3), $\bar{x}(v)$ and $\bar{y}(v)$ now become functions of $\bar{a}_1$,$\bar{a}_3$,$\bar{a}_4$,$\bar{a}_5$,...,$\bar{a}_{2\bar{J}}$ and $\bar{b}_2$,$\bar{b}_3$,$\bar{b}_4$,...,$\bar{b}_{2\bar{J}}$, respectively.

$$\begin{aligned}
\bar{x}(v) =& \bar{P}_x - \bar{T}_x \times \sin v + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times [\cos(j \times v) - (-1)^j] \\
&+ \sum_{j=2}^{\bar{J}} \bar{a}_{2j}[\sin(j \times v) - j \times (-1)^j \times \sin v] \\
\bar{y}(v) =& \bar{P}_y - \sin v \times \bar{T}_y + \sum_{j=2}^{\bar{J}} \bar{b}_{2j-1} \times [\sin(j \times v) + (-1)^j \times j \times \sin v] \\
&+ \sum_{j=1}^{\bar{J}} \bar{b}_{2j} \times [\cos(j \times v) - (-1)^j]
\end{aligned} \tag{3.31}$$

Similar to Eq.(3.25), the derivative of $E_{\bar{x}}$ with respect to $\bar{a}_j (j = 1,3,4,5,\ldots,\bar{J})$ equals zero. So is the derivative of $E_{\bar{y}}$ with respect to $\bar{b}_j (j = 2,3,4,\ldots,\bar{J})$. Which lead to the following equations

$$
\begin{aligned}
&\sum_{i=2}^{\bar{I}} [\bar{G}_x(i) + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times \bar{M}_x(j,\bar{v}_i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j} \times \bar{N}_x(j,\bar{v}_i)] \times \bar{M}_x(\frac{k+1}{2},\bar{v}_i) = 0 \\
&(k = 1,3,5,\ldots,2\bar{J}-1) \\
&\sum_{i=2}^{\bar{I}} [\bar{G}_x(i) + \sum_{j=1}^{\bar{J}} \bar{a}_{2j-1} \times \bar{M}_x(j,\bar{v}_i) + \sum_{j=2}^{\bar{J}} \bar{a}_{2j} \times \bar{N}_x(j,\bar{v}_i)] \times \bar{N}_x(\frac{k}{2},\bar{v}_i) = 0 \\
&(k = 4,6,\ldots,2\bar{J}) \\
&\sum_{i=2}^{\bar{I}} [\bar{G}_y(i) + \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1} \times \bar{M}_y(j,\bar{v}_i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j} \times \bar{N}_y(j,\bar{v}_i)] \times \bar{M}_y(\frac{k+1}{2},\bar{v}_i) = 0 \\
&(k = 3,5,\ldots,2\bar{J}-1) \\
&\sum_{i=2}^{\bar{I}} [\bar{G}_y(i) + \sum_{j=1}^{\bar{J}} \bar{b}_{2j-1} \times \bar{M}_y(j,\bar{v}_i) + \sum_{j=2}^{\bar{J}} \bar{b}_{2j} \times \bar{N}_y(j,\bar{v}_i)] \times \bar{N}_y(\frac{k}{2},\bar{v}_i) = 0 \\
&(k = 2,4,\ldots,2\bar{J}) \\
&\bar{G}_x(i) = \bar{P}_x - \bar{T}_x \times \sin\bar{v}_i - \bar{x}_i \\
&\bar{M}_x(j,\bar{v}_i) = \cos j \times \bar{v}_i - (-1)^j \\
&\bar{N}_x(j,\bar{v}_i) = j \times (-1)^j \times \sin\bar{v}_i + \sin j \times \bar{v}_i \\
&\bar{G}_y(i) = \bar{P}_y - \bar{T}_y \sin\bar{v}_i - \bar{y}_i \\
&\bar{M}_y(j,\bar{v}_i) = (-1)^j \times j \times \sin\bar{v}_i + \sin j \times \bar{v}_i \\
&\bar{N}_y(j,\bar{v}_i) = \cos j \times \bar{v}_i - (-1)^j
\end{aligned}
\tag{3.32}
$$

By solving the above equations, we could get the mathematical description for generating the composite generalized elliptic segment, see Fig.3.11b

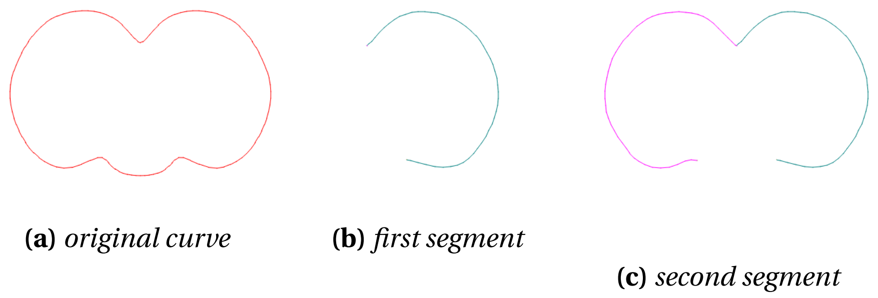


Figure 3.11: *Two adjacent segments connected at $v = pi$*

3.2.2 Connect the last generalized elliptic segment with its two adjacent segment

For the last segment $\widehat{CA}$ defined within the region of the parametric variable v from $v = v_2$ to $v = v_0$ (see Fig.3.12), using a double bar to indicate this elliptic segment, the positional values of the last generalized elliptic segment at $v = v_2$ and $v = v_0$ should be equal to that of the second generalized elliptic segment $\widehat{BC}$ at $v = v_2$ and to that of the first generalized elliptic segment $\widehat{AB}$ at $v = v_0$, respectively. Use P_x,P_y,T_x,T_y to denote the

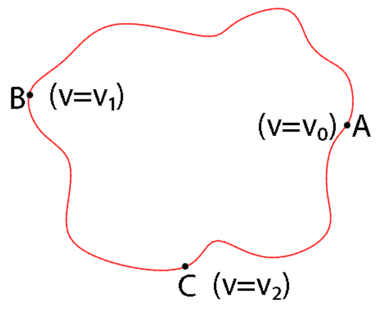


Figure 3.12: *last segment with both ends connected with others*

x component and y component of position and tangent of point A, and $\bar{\bar{P}}_x$,$\bar{\bar{P}}_y$,$\bar{\bar{T}}_x$,$\bar{\bar{T}}_y$ to denote the x component and y component of position and tangent of point C, we can get the value of these constants.

$$\begin{aligned}
P_x &= x_c + a_0 + \sum_{j=1}^{J} \left[a_{2j-1} \times \cos(j \times v_0) + a_{2j} \times \sin(j \times v_0)\right] \\
P_y &= y_c + a_0 + \sum_{j=1}^{J} \left[b_{2j-1} \times \sin(j \times v_0) + b_{2j} \times \cos(j \times v_0)\right] \\
T_x &= \sum_{j=1}^{J} \left[-a_{2j-1} \times j \times \sin(j \times v_0) + a_{2j} \times j \times \cos(j \times v_0)\right] \\
T_y &= \sum_{j=1}^{J} \left[b_{2j-1} \times j \times \cos(j \times v_0) - b_{2j} \times j \times \sin(j \times v_0)\right] \\
&\text{and} \qquad (3.33) \\
\bar{\bar{P}}_x &= \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{\bar{J}} \left[\bar{a}_{2j-1} \times \cos(j \times v_2) + \bar{a}_{2j} \times \sin(j \times v_2)\right] \\
\bar{\bar{P}}_y &= \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{\bar{J}} \left[\bar{b}_{2j-1} \times \sin(j \times v_2) + \bar{b}_{2j} \times \cos(j \times v_2)\right] \\
\bar{\bar{T}}_x &= \sum_{j=1}^{\bar{J}} \left[-\bar{a}_{2j-1} \times j \times \sin(j \times v_0) + \bar{a}_{2j} \times j \times \cos(j \times v_0)\right] \\
\bar{\bar{T}}_y &= \sum_{j=1}^{\bar{J}} \left[\bar{b}_{2j-1} \times j \times \cos(j \times v_0) - \bar{b}_{2j} \times j \times \sin(j \times v_0)\right]
\end{aligned}$$

Since the last segment $\widehat{CA}$ should connect smoothly to its adjacent segments $\widehat{AB}$ and $\widehat{BC}$ at Point A and Point C, the position and tangent of the last segment at Point A and Point C should be equal to those of $\widehat{AB}$ at Point A and $\widehat{BC}$ at Point C, we have the following positional continuity conditions for the $\bar{\bar{x}}$ and $\bar{\bar{y}}$, respectively

$$\begin{aligned}
&\bar{\bar{x}}_{\bar{\bar{I}}} = P_x = \bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} \left[\bar{\bar{a}}_{2j-1} \times \cos(j \times v_0) + \bar{\bar{a}}_{2j} \times \sin(j \times v_0)\right] \\
&\bar{\bar{x}}_0 = \bar{\bar{P}}_x = \bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} \left[\bar{\bar{a}}_{2j-1} \times \cos(j \times v_2) + \bar{\bar{a}}_{2j} \times \sin(j \times v_2)\right] \\
&T_x = \sum_{j=1}^{J} \left[-\bar{\bar{a}}_{2j-1} \times j \times \sin(j \times v_0) + \bar{\bar{a}}_{2j} \times j \times \cos(j \times v_0)\right] \\
&\bar{\bar{T}}_x = \sum_{j=1}^{J} \left[-\bar{a}_{2j-1} \times j \times \sin(j \times v_2) + \bar{a}_{2j} \times j \times \cos(j \times v_2)\right]
\end{aligned}$$

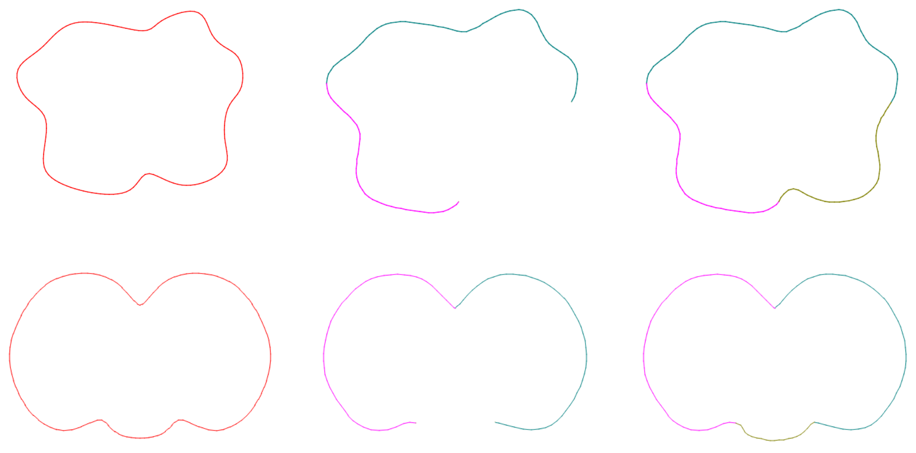
and (3.34)

$$\begin{aligned}
&\bar{\bar{y}}_{\bar{\bar{I}}} = P_y = \bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} \left[\bar{\bar{b}}_{2j-1} \times \sin(j \times v_0) + \bar{\bar{b}}_{2j} \times \cos(j \times v_0)\right] \\
&\bar{\bar{y}}_0 = \bar{\bar{P}}_y = \bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} \left[\bar{\bar{b}}_{2j-1} \times \sin(j \times v_2) + \bar{\bar{b}}_{2j} \times \cos(j \times v_2)\right] \\
&T_y = \sum_{j=1}^{J} \left[\bar{\bar{b}}_{2j-1} \times j \times \cos(j \times v_0) - \bar{\bar{b}}_{2j} \times j \times \sin(j \times v_0)\right] \\
&\bar{\bar{T}}_y = \sum_{j=1}^{J} \left[\bar{b}_{2j-1} \times j \times \cos(j \times v_2) - \bar{b}_{2j} \times j \times \sin(j \times v_2)\right]
\end{aligned}$$

The sum of squared errors between the $\widehat{AC}$ of original curve and the generalized elliptic segment $\widehat{AC}$ at all points except the first point (Point C) and the last point (Point A) because at these two points, the x,y components are fixed: $\bar{\bar{x}}_1 = \bar{\bar{P}}_x$, $\bar{\bar{y}}_1 = \bar{\bar{P}}_y$, $\bar{\bar{x}}_{\bar{\bar{I}}} = P_x$, $\bar{\bar{y}}_{\bar{\bar{I}}} = P_y$, according to Eq.(3.34). Hence, $E_{\bar{\bar{x}}}$ and $E_{\bar{\bar{y}}}$ are defined as

$$\begin{aligned}
E_{\bar{\bar{x}}} &= \sum_{i=2}^{\bar{\bar{I}}-1} [\bar{\bar{x}}_i - \bar{\bar{x}}(\bar{\bar{v}}_i)]^2 = \sum_{i=2}^{\bar{\bar{I}}-1} \left[\bar{\bar{x}}_i - \bar{\bar{x}}_c - \bar{\bar{a}}_0 - \sum_{j=1}^{\bar{\bar{J}}} \left[\bar{\bar{a}}_{2j-1} \cos(j \times \bar{\bar{v}}_i) + \bar{\bar{a}}_{2j} \sin(j \times \bar{\bar{v}}_i)\right]\right]^2 \\
E_{\bar{\bar{y}}} &= \sum_{i=2}^{\bar{\bar{I}}-1} [\bar{\bar{y}}_i - \bar{\bar{y}}(\bar{\bar{v}}_i)]^2 = \sum_{i=2}^{\bar{\bar{I}}-1} \left[\bar{\bar{y}}_i - \bar{\bar{y}}_c - \bar{\bar{b}}_0 - \sum_{j=1}^{\bar{\bar{J}}} \left[\bar{\bar{b}}_{2j-1} \sin(j \times \bar{\bar{v}}_i) + \bar{\bar{b}}_{2j} \cos(j \times \bar{\bar{v}}_i)\right]\right]^2
\end{aligned} \tag{3.35}$$

To minimize the errors between the generalized elliptic segment and the original curve segment subject to the boundary constraints specified in Eq.(3.34) is a typical nonlinear programming problem. I choose to use a sequential least squares programming algorithm named SLSQP proposed by Kraft [1988] to solve this minimization problem, because SLSQP supports constrained minimization problem and is well-developed by a Python Library called Scipy, and get the variables $\bar{\bar{a}}_j(j = 0,1,2,\ldots,2\bar{\bar{J}})$ and $\bar{\bar{b}}_j(j = 0,1,2,\ldots,2\bar{\bar{J}})$. The results are shown in Fig.3.13.



(a) *original curve* **(b)** *first segment and in-between segment combined* **(c)** *add the last segment*

Figure 3.13: *add the last segment*

Special case: when the last segment starts at 180 degree and ends at 0 degree

Although in preceding Subsection.3.2.2 SLSQP optimizer can find the solution of $\min E_x$ and $\min E_y$ subject to different equality constraints, the convergence of SLSQP is relatively slow. In order to improve the computational efficiency, it's better to detect special cases before resorting to SLSQP. One special case, for example, as shown on Fig.3.14, when the curve splits into two segment with the last segment starts from $v_1 = \pi$ and ends at $v_0 = 0$.

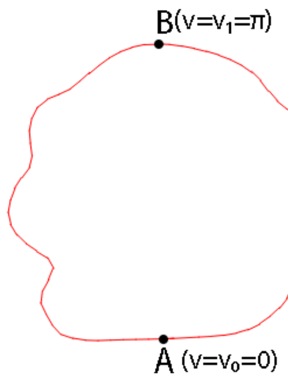


Figure 3.14: *three segments of a closed curve where its last segment starts from $v_2 = \pi$ and ends at $v_0 = 0$*

Using a bar and a double bar to indicate the first and second elliptic segments respectively, the positional values of the first generalized elliptic segment at $v = 0$ and $v = \pi$ should be equal to those of the second generalized elliptic segment at $v = 2\pi$ and $v = \pi$, respectively. Therefore, we have the following positional continuity conditions for the x position function

$$\begin{aligned}
&\bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} (-1)^j \bar{\bar{a}}_{2j-1} = \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{J} (-1)^j \bar{a}_{2j-1} \\
&\bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} \bar{\bar{a}}_{2j-1} = \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{J} \bar{a}_{2j-1} \\
&\bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} (-1)^j \bar{\bar{b}}_{2j} = \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{J} (-1)^j \bar{b}_{2j} \\
&\bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} \bar{\bar{b}}_{2j} = \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{J} \bar{b}_{2j}
\end{aligned} \tag{3.36}$$

Similarly, the tangential values of the first generalized elliptic segment at $v = 0$ and $v = \pi$ should be equal to those of the second generalized elliptic segment at $v = 2\pi$ and $v = \pi$, respectively, which leads to the following tangential continuity conditions

$$\begin{aligned}
&\sum_{j=1}^{J}(-1)^j j\bar{\bar{a}}_{2j} = \sum_{j=1}^{J}(-1)^j j\bar{a}_{2j} \\
&\sum_{j=1}^{J} j\bar{\bar{a}}_{2j} = \sum_{j=1}^{J} j\bar{a}_{2j} \\
&\sum_{j=1}^{J}(-1)^j j\bar{\bar{b}}_{2j-1} = \sum_{j=1}^{J}(-1)^j j\bar{b}_{2j-1} \\
&\sum_{j=1}^{J} j\bar{\bar{b}}_{2j-1} = \sum_{j=1}^{J} j\bar{b}_{2j-1}
\end{aligned} \tag{3.37}$$

From the first two of Eqs. (3.36) and (3.37), we can determine four unknown constants $\bar{\bar{a}}_1$, $\bar{\bar{a}}_2$, $\bar{\bar{a}}_3$ and $\bar{\bar{a}}_4$ for x position component, as are shown in Eqs.(3.38). To be more specifical, if we subtract first two of Eqs. (3.36), we can get the formula for $\bar{\bar{a}}_1$. By subtracting the first two of Eqs. (3.37), we can get the formula for $\bar{\bar{a}}_2$. Substituting $\bar{\bar{a}}_1$ from Eqs.(3.38) into the first equation in Eqs.(3.36) and we get the formula for $\bar{\bar{a}}_3$. Adding the first two of Eqs.(3.37), we can get the formula for $\bar{\bar{a}}_4$.

$$\begin{aligned}
&\bar{\bar{a}}_1 = \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\bar{a}_{2j-1} - \sum_{j=3}^{J}[1-(-1)^j]\bar{\bar{a}}_{2j-1}\Big\} \\
&\bar{\bar{a}}_2 = \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]j\bar{a}_{2j} - \sum_{j=3}^{J}[1-(-1)^j]j\bar{\bar{a}}_{2j}\Big\} \\
&\bar{\bar{a}}_3 = \bar{x}_c - \bar{\bar{x}}_c + \bar{a}_0 - \bar{\bar{a}}_0 + \frac{1}{2}\Big\{\sum_{j=1}^{J}[1+(-1)^j]\bar{a}_{2j-1} - \sum_{j=3}^{J}[1+(-1)^j]\bar{\bar{a}}_{2j-1}\Big\} \\
&\bar{\bar{a}}_4 = \frac{1}{4}\Big\{\sum_{j=1}^{J}[1+(-1)^j]j\bar{a}_{2j} - \sum_{j=3}^{J}[1+(-1)^j]j\bar{\bar{a}}_{2j}\Big\}
\end{aligned} \tag{3.38}$$

From the last two of Eqs.(3.36) and (3.37) , we can obtain four unknown constants $\bar{\bar{b}}_0, \bar{\bar{b}}_1, \bar{\bar{b}}_2, \ldots, \bar{\bar{b}}_{2J}$ for y position component, as listed in Eqs.(3.39).

More specifically, subtracting the last two of Eqs.(3.36), we can get the $\bar{\bar{b}}_2$; subtracting the last two of Eqs.(3.37), we can get the $\bar{\bar{b}}_1$; adding the last two of Eqs.(3.37), we can get the $\bar{\bar{b}}_3$; adding the last two of Eqs. (3.36), we can get $\bar{\bar{b}}_4$.

$$
\begin{aligned}
\bar{\bar{b}}_1 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,j\bar{b}_{2j-1} - \sum_{j=3}^{J}[1-(-1)^j]\,j\bar{\bar{b}}_{2j-1}\Big\} \\
\bar{\bar{b}}_2 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,\bar{b}_{2j} - \sum_{j=3}^{J}[1-(-1)^j]\,\bar{\bar{b}}_{2j}\Big\} \\
\bar{\bar{b}}_3 &= \frac{1}{4}\Big\{\sum_{j=1}^{J}[1+(-1)^j]\,j\bar{b}_{2j-1} - \sum_{j=3}^{J}[1+(-1)^j]\,j\bar{\bar{b}}_{2j-1}\Big\} \\
\bar{\bar{b}}_4 &= \bar{y}_c - \bar{\bar{y}}_c + \bar{b}_0 - \bar{\bar{b}}_0 + \frac{1}{2}\Big\{\sum_{j=1}^{\bar{J}}[1+(-1)^j]\,\bar{b}_{2j} - \sum_{j=3}^{J}[1+(-1)^j]\,\bar{\bar{b}}_{2j}\Big\}
\end{aligned}
\tag{3.39}
$$

Substituting Eqs. (3.38) and (3.39) into Eq. (3.3), the x and y components of the positional function for the second generalized elliptic segment are changed into

$$
\begin{aligned}
\bar{\bar{x}}(v) &= (\bar{x}_c + \bar{a}_0)\cos 2v + \frac{1}{2}\sum_{j=1}^{J}\left[D_1(j,v)\bar{a}_{2j-1} + jD_2(j,v)\bar{a}_{2j}\right] \\
&\quad + (\bar{\bar{x}}_c + \bar{\bar{a}}_0)(1-\cos 2v) \\
&\quad + \frac{1}{2}\sum_{j=3}^{J}\Big\{\left[2\cos jv - D_1(j,v)\right]\bar{\bar{a}}_{2j-1} + \left[2\sin jv - jD_2(j,v)\right]\bar{\bar{a}}_{2j}\Big\} \\
\bar{\bar{y}}(v) &= (\bar{y}_c + \bar{b}_0)\cos 2v + \frac{1}{2}\sum_{j=1}^{J}\left[jD_2(j,v)\bar{b}_{2j-1} + D_1(j,v)\bar{b}_{2j}\right] \\
&\quad + (\bar{\bar{y}}_c + \bar{\bar{b}}_0)(1-\cos 2v) \\
&\quad + \frac{1}{2}\sum_{j=3}^{J}\Big\{\left[2\sin jv - jD_2(j,v)\right]\bar{\bar{b}}_{2j-1} + \left[2\cos jv - D_1(j,v)\right]\bar{\bar{b}}_{2j}\Big\}
\end{aligned}
\tag{3.40}
$$

where $D_1(j,v) = [1-(-1)^j]\cos v + [1+(-1)^j]\cos 2v$, and $D_2(j,v) = [1-$

$(-1)^j]\sin v+\frac{1}{2}[1+(-1)^j]\sin 2v$.

Using Eq. (3.40) we can perform curve fitting for the curve segment $\overset{\frown}{CDA}$. Firstly, construct similar Errors for $\bar{\bar{x}}$ and $\bar{\bar{y}}$ as that in Eqs.(3.4), (3.5) and (3.7). Then by setting the derivatives of the Errors to zero, we can get the following equations involving $2J-3$ items $\bar{\bar{a}}_0$,$\bar{\bar{a}}_5$,$\bar{\bar{a}}_6$,…,$\bar{\bar{a}}_{2J-1}$,$\bar{\bar{a}}_{2J}$ as unknown variables for x position, and another $2J-3$ items $\bar{\bar{b}}_0$,$\bar{\bar{b}}_5$,$\bar{\bar{b}}_6$,…,$\bar{\bar{b}}_{2J-1}$, $\bar{\bar{b}}_{2J}$ for y position

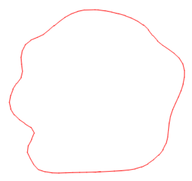
$$\begin{aligned}
&C_a(i)=x_i-(\bar{x}_c+\bar{a}_0)\cos 2v_i-\bar{\bar{x}}_c(1-\cos 2v_i)\\
&-\frac{1}{2}\sum_{j=1}^{J}[D_1(j,v_i)\bar{a}_{2j-1}+jD_2(j,v_i)\bar{a}_{2j}]\\
&A(i)=\bar{\bar{a}}_0(1-\cos 2v_i)+\sum_{j=3}^{J}[\cos jv_i-\frac{1}{2}D_1(j,v_i)]\bar{\bar{a}}_{2j-1}\\
&+\sum_{j=3}^{J}[\sin jv_i-\frac{j}{2}D_2(j,v_i)]\bar{\bar{a}}_{2j}\\
&P_1(k,i)=\cos(\frac{k+1}{2}\times v_i)-\frac{1}{2}D_1(\frac{k+1}{2},v_i),(k=5,7,9,\ldots,2J-1)\\
&P_2(k,i)=\sin(\frac{k}{2}\times v_i)-\frac{k}{4}D_2(\frac{k}{2},v_i),(k=6,8,10,\ldots,2J)\\
&\sum_{i=1}^{I}(1-\cos 2v_i)\times A(i)=\sum_{i=1}^{I}(1-\cos 2v_i)\times C_a(i)\\
&\sum_{i=1}^{I}P_1(k,i)\times A(i)=\sum_{i=1}^{I}P_1(k,i)\times C_a(i),(k=5,7,9,\ldots,2J-1)\\
&\sum_{i=1}^{I}P_2(k,i)\times A(i)=\sum_{i=1}^{I}P_2(k,i)\times C_a(i),(k=6,8,10,\ldots,2J)
\end{aligned} \tag{3.41}$$

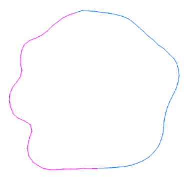
and

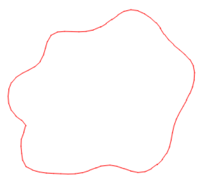
$$\begin{aligned}
&C_b(i)=y_i-(\bar{y}_c+\bar{b}_0)\cos 2v_i-\bar{\bar{y}}_c(1-\cos 2v_i)\\
&-\frac{1}{2}\sum_{j=1}^{J}[jD_2(j,v_i)\bar{b}_{2j-1}+D_1(j,v_i)\bar{b}_{2j}]
\end{aligned}$$

$$
\begin{aligned}
&B(i) = \bar{\bar{b}}_0(1-\cos 2v_i) + \sum_{j=3}^{J}[\sin jv_i - \frac{j}{2}D_2(j,v_i)]\bar{\bar{b}}_{2j-1} \\
&+ \sum_{j=3}^{J}[\cos jv_i - \frac{1}{2}D_1(j,v_i)]\bar{\bar{b}}_{2j} \\
&P_3(k,i) = \sin(\frac{k+1}{2}\times v_i) - \frac{k+1}{4}D_2(\frac{k+1}{2},v_i), (k=5,7,9,\ldots,2J-1) \\
&P_4(k,i) = \cos(\frac{k}{2}\times v_i) - \frac{1}{2}D_1(\frac{k}{2},v_i), (k=6,8,10,\ldots,2J) \\
&\sum_{i=1}^{I}(1-\cos 2v_i)\times B(i) = \sum_{i=1}^{I}(1-\cos 2v_i)\times C_b(i) \\
&\sum_{i=1}^{I}P_3(k,i)\times B(i) = \sum_{i=1}^{I}P_3(k,i)\times C_b(i), (k=5,7,9,\ldots,2J-1) \\
&\sum_{i=1}^{I}P_4(k,i)\times B(i) = \sum_{i=1}^{I}P_4(k,i)\times C_b(i), (k=6,8,10,\ldots,2J)
\end{aligned}
\tag{3.42}
$$

These two systems of linear equations Eqs.(3.41) and Eqs.(3.42) can be represented in matrix form, as in Appendix F. With the solution of these two systems of equations, we obtain all $2J-3$ unknown variables $\bar{\bar{a}}_0, \bar{\bar{a}}_5, \bar{\bar{a}}_6, \bar{\bar{a}}_7, \ldots, \bar{\bar{a}}_{2J}$ and $\bar{\bar{b}}_0$, $\bar{\bar{b}}_5$, $\bar{\bar{b}}_6$, $\bar{\bar{b}}_7$, ..., $\bar{\bar{b}}_{2J}$. We then achieve the mathematical description of the last generalized elliptic segment according to Eqs.(3.40). We generate the composite generalized elliptic segments depicted in the right column of Fig. 3.15 and the original curves are shown in the left column.


(a) *original curve*


(b) *composite generalized elliptic segments*


(c) *original curve*

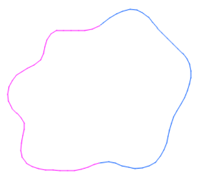

(d) *composite generalized elliptic segments*

Figure 3.15: *Two composite generalized elliptic segments meets at 0 and π*

Chapter 4

Cross-Section Defined Surface Modeling

In chapter 3, I investigated curve fitting of cross-section shapes and solved the problem of representing cross-sectional curves with generalized ellipses or composite generalized elliptic segments. The remaining problem is how to mathematically define surface models from these cross-sectional curves.

In this chapter, a modelling method of 3D models is presented. A 3D model is divided into different parts. For each part, characteristic cross section curves are generated and approximated by generalized ellipses and composite generalized elliptic segments. The parts are constructed from these generalized elliptic curves assembled together to create the 3D model. With the approach presented in this chapter, different orders of the continuities between adjacent curve segments and surface patches are well maintained. Since surface creation of 3D models is transformed into generation of cross-sectional curves and few unknown constants are required to describe these cross-sectional curves accurately, this approach can reduce data size of surface modeling.

4.1 Introduction

Since standard ellipses can only give a very rough approximation of the cross section curves of human body, the sweep surface of a moving ellipse is far from the practical shape of the human body. In order to remedy this shortcoming, sweep-based human deformation introduces the concept of displacement map (Hyun et al. [2005]).

For most cases, the vertex of human body is not exactly located on an ellipse. Therefore, the algorithm of the displacement map first brings the vertex back to the coordinate system of the ellipse. Then, the relative displacement of the vertex from the moving ellipse is measured and a continuous displacement map is constructed by approximating or interpolating a set of sampled relative displacements.

Although the introduction of the displacement map can improve the realism of human modeling, it can be observed from the above discussion that such a method is also a approximation and requires more computer storage and additional computational cost.

Since generalized ellipses and composite generalized elliptic segments can represent the cross section curves of human body very exactly, the accuracy improvement using the displacement map or other approaches is not required. Therefore, the method proposed in this chapter can significantly reduce the computer storage and obviously increase modelling efficiency.

First, this research uses the 3D models from Autodesk [2015] Character Generator dataset, and creates skeleton for the models. The benefits of this dataset is that the models are all in A-pose, which makes little to none mesh distortion compared to human character meshes in arbitrary poses. Next, I automatically extract the cross-sections on the template model by casting a ray from a bone, and getting the intersection position of the casting ray and the surface of template model with Maya's closestIntersection function. Later use the techniques in Chapter 3 to transform these cross-sectional curves into parametric curves. Then, repeat the last step for different part of the same bone, and organize the extracted cross-sections in a sequence and finally, a new surface is generated from the cross-sectional contour sets by the algorithm in this chapter. Motivated by the work of Hyun et al. [2003, 2005] but without using standard ellipses and displacement map, I will introduce generalized ellipses and composite generalized elliptic segments to approximate the cross-sectional curves of human body, and present a simple and efficient method with small data storage to build human models in this chapter.

4.2 3D models defined with generalized ellipses

In order to build a 3D model, the first step is drawing some cross-sectional curves which define the 3D model. Then approximate the original cross-sectional curves with the generalized ellipses. After that, surface patches can be constructed from these generalized ellipses with the following method.

As mentioned in Tokuyama [2000], among three surface interpolation methods, i. e., interpolating through distinct point data, skinning over a family of curves and interpolating the surface simultaneously over two families of intersection curves, the skinning method is generally considered to be the most frequently used technique for surface construction . Here I use this skinning method to construct surface patches of human parts.

If a surface patch(shown in Fig. 4.1) will be constructed from K generalized ellipses $\big(c_{xk}(v),\ c_{yk}(v), c_{zk}(v)\big)$ $(k=0,1,2,\ldots,K-1)$ where $c_{xk}(v)$, $c_{yk}(v)$ and $c_{zk}(v)$ are determined by Eq. (3.3), one can use the following equation to describe the surface to be constructed

$$\begin{aligned}
&t(u,v)=\sum_{m=0}^{K-1} u^m f_{tm}(v)\\
&f_{tm}(v)=\mathbf{F}_{tm} * [1, cos(v), sin(v), cos(2v), sin(2v), ..., cos(Jv), sin(Jv)]^T\\
&(t=x,y,z)
\end{aligned} \tag{4.1}$$

where $\mathbf{F}_{tm}(m=0,1,2,3,\ldots,K-1)$ are row vectors, and each contains $2J+1$ unknown constants.

Uniformly dividing the region $u=0$ to $u=1$ into $K-1$ equal intervals which gives the interval length to be $d_u=\frac{1}{K-1}$, we have $u_k=k\times d_u$ $(k=0,1,2,3,\ldots,K-1)$. The unknown constants $\mathbf{F}_{tm}$ $(m=0,1,2,3,\ldots,K-1)$ can be determined by solving the following linear algebraic equations

$$\begin{aligned}
&c_{tk}(v)=\sum_{m=0}^{K-1} u_k^m f_{tm}(v),\\
&(k=0,1,2,3,\ldots,K-1; t=x,y,z)
\end{aligned} \tag{4.2}$$

where $c_{tk}(v)$ is the functions obtained in Eq.(3.3).

Rewriting Eq. (4.2) into the form of matrix, we obtain the following

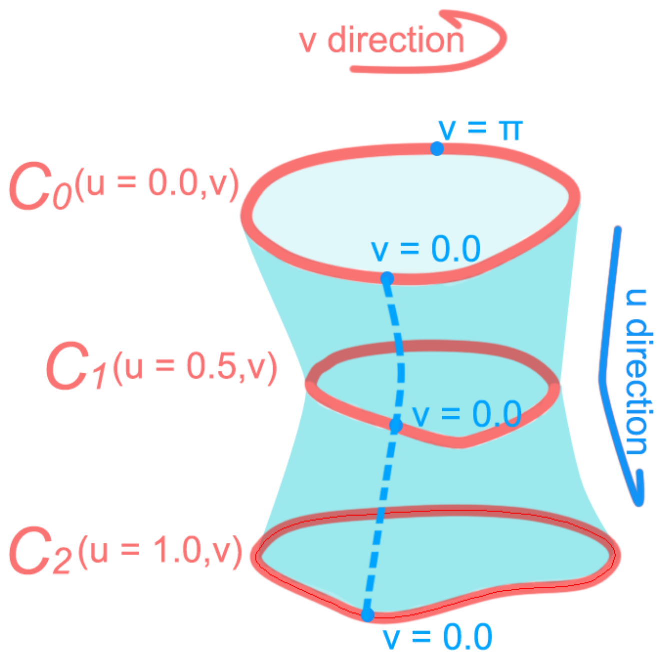


Figure 4.1: *Illustration of surface designed with generalised ellipses*

mathematical expression

$$\mathbf{R}_t(u_{km})\mathbf{F}_t = \mathbf{C}_t \qquad (t = x, y, z) \tag{4.3}$$

where $\mathbf{R}_t(u_{km})$ is a $K \times K$ square matrix with the elements $u_{km} = u_k^m$, $\mathbf{F}_t(v) = [\mathbf{F}_{t0}, \mathbf{F}_{t1}, \mathbf{F}_{t2}, \ldots, \mathbf{F}_{tK-1}]^T$ and $\mathbf{C}_t = [\mathbf{C}_{t0}, \mathbf{C}_{t1}, \mathbf{C}_{t2}, \ldots, \mathbf{C}_{t(K-1)}]^T$ are two matrices with K vectors, each vector has $2J+1$ elements. $\mathbf{C}_{tm}$ consists of the constants of the m^{th} generalized ellipse in Eq.(3.3) and $\mathbf{F}_{tm}$ is the unknown constants that defines the surface patch needed to be solved.

Using $\mathbf{R}_t(u_{km})^{-1}$ to indicate the inverse matrix of $\mathbf{R}_t(u_{km})$ and left multiplying both sides of Eq. (4.2) by this inverse matrix, we obtain the unknown functions with the following equation

$$\mathbf{F}_t(v) = \mathbf{R}_t(u_{km})^{-1} \times \mathbf{C}_t(v) \tag{4.4}$$

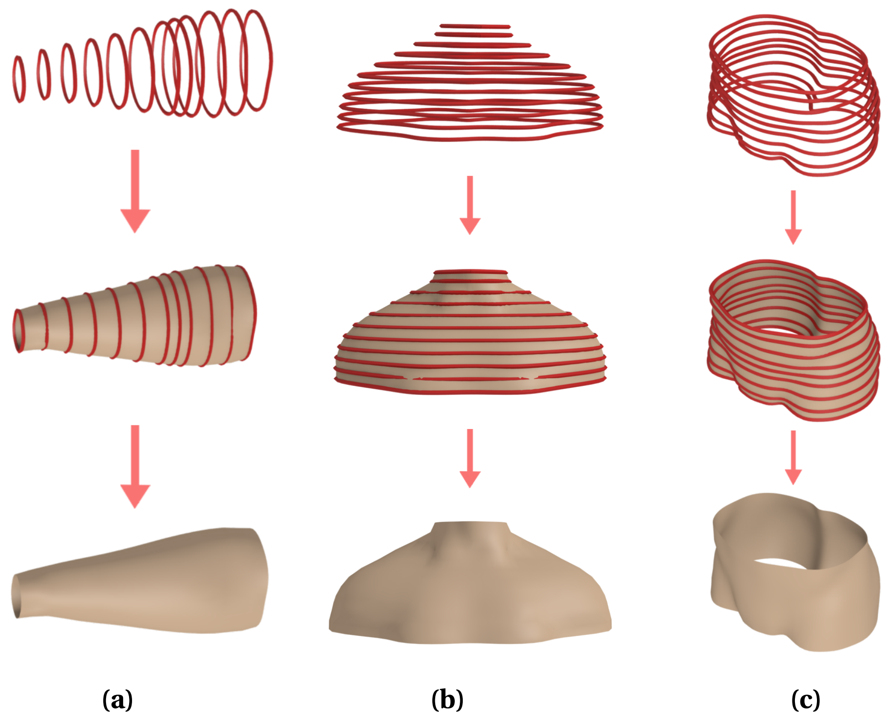


Figure 4.2: *Human forearm, neck and hip created from generalized ellipses. Cross-sectional curves described as generalized ellipses are in red.*

Substitute $\mathbf{F}_t(v)$ into Eq.(4.1), we generate the 3D models defined with generalized ellipses. Some examples are shown in Fig. 4.2.

At the boundary curves where two different surface patches are to be connected together, we must consider the continuity between the two surface patches. For an existing surface patch indicated by Eq. 4.1, different order continuities such as the boundary tangents and boundary curvature etc. at its boundaries can be determined from the different orders of partial derivatives ($n = 1, 2, 3, \ldots$) of the surface patch with respect to the parametric variable u. By introducing these partial derivatives into the above operation, different order continuities between two connected surface patches can be obtained.

For example, if we intend to connect two surface patches with the tangent continuity at $u = 1$ of the existing surface patch $\bar{t}(u, v)$, we obtain the mathematical expressions of the first partial derivative $\left[\frac{\partial \bar{t}(u,v)}{\partial u}\right]_{u=1} = \sum_{m=1}^{\bar{K}-1} m\bar{f}_{tm}(v)$ of the existing surface patch $\bar{t}(u,v)$ and $\left[\frac{\partial t(u,v)}{\partial u}\right]_{u=0} = f_{t1}(v)$ of the unknown surface patch $t(u,v)$ with respect to the parametric variable u from Eq. 4.1, respectively. Then the following tangential continuity constraint is added to Eq. 4.2.

$$
\begin{aligned}
&f_{t1}(v) = \sum_{m=1}^{\bar{K}-1} m\bar{f}_{tm}(v) \\
&(t = x, y, z)
\end{aligned}
\tag{4.5}
$$

Rewriting Eq.(4.5) into the form of vector, we have the following mathematical expression

$$
\begin{aligned}
&[0,1,0,0,\ldots,0] \cdot [\mathbf{F}_{t0}, \mathbf{F}_{t1}, \mathbf{F}_{t2}, \ldots, \mathbf{F}_{tK}]^T = \sum_{m=0}^{\bar{K}-1} m\bar{\mathbf{F}}_{tm} \\
&(t = x, y, z)
\end{aligned}
\tag{4.6}
$$

Add Eq.(4.6) into Eq. (4.4) as a new row. Since one more linear algebraic equation is introduced, the unknown functions $f_{tm}(v)$ in Eq. 4.1 should be increased from $f_{tK-1}(v)$ to $f_{tK}(v)$ and all K in Eq. 4.1 and thereafter will be replaced by $(K+1)$ in order to form an invertible square matrix. The expanding equation system in matrix form is as follow

$$\begin{bmatrix} \mathbf{C}_t \\ \sum_{m=1}^{\bar{K}-1} m\bar{\mathbf{F}}_{tm} \end{bmatrix} = \begin{bmatrix} u_0^0 & u_0^1 & u_0^2 & u_0^3 & \dots & u_0^K \\ u_1^0 & u_1^1 & u_1^2 & u_1^3 & \dots & u_1^K \\ u_2^0 & u_2^1 & u_2^2 & u_2^3 & \dots & u_2^K \\ \dots & & & & & \\ u_{K-1}^0 & u_{K-1}^1 & u_{K-1}^2 & u_{K-1}^3 & \dots & u_{K-1}^K \\ 0 & 1 & 0 & 0 & \dots & 0 \end{bmatrix} * \begin{bmatrix} \mathbf{F}_{t0} \\ \mathbf{F}_{t1} \\ \mathbf{F}_{t2} \\ \dots \\ \mathbf{F}_{t(K-1)} \\ \mathbf{F}_{tK} \end{bmatrix}$$

$$(t = x, y, z) \tag{4.7}$$

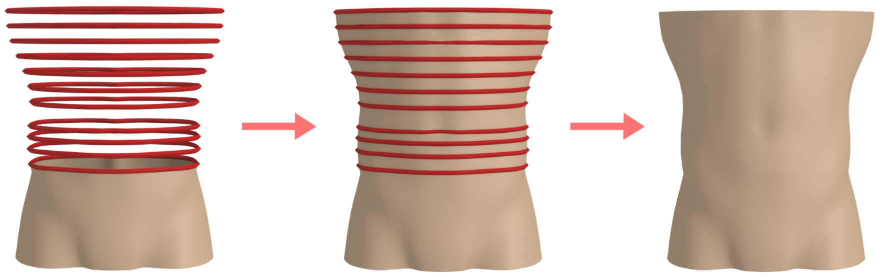
Solve the above equations, we get the constants in $\mathbf{F}_{tm}(m = 0, 1, 2, \dots, K)$. Therefore, the mathematical description of a surface with one shared edge with existing surface is obtained.

$$t(u, v) = \sum_{m=0}^{K} u^m \times \mathbf{F}_{tm} \times [1, \cos v, \sin v, \dots, \cos(Jv), \sin(Jv)]^T$$
$$(t = x, y, z) \tag{4.8}$$

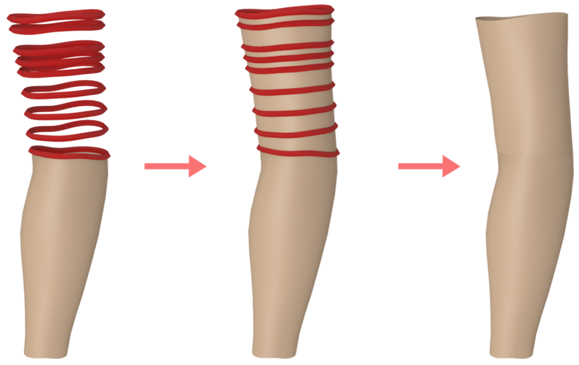
Some examples of a surface shares one edge with another surface patch are shown in Fig.4.3

If both opposite edges of a surface patch $t(u, v)$ will be connected to two separate surface patches $\bar{t}(u, v)$ and $\bar{\bar{t}}(u, v)$, the continuity at the boundary curve $u = 0$ of the exiting surface patch $\bar{\bar{t}}(u, v)$ should also be considered. Similar to the above treatment, the following boundary condition for the tangential continuity will be added to Eq. 4.2

$$\sum_{m=1}^{K-1} m f_{tm}(v) = \bar{\bar{f}}_{t1}(v)$$
$$(t = x, y, z) \tag{4.9}$$



(a)



(b)

Figure 4.3: *A human torso which shares a boundary with hip and an upper arm shares a boundary with a forearm are created from generalized ellipses. Cross-sectional curves described as generalized ellipses are in red.*

and the unknown functions $f_{tm}(v)$ in Eq. 4.1 will be increased from $f_{tK}(v)$ to $f_{tK+1}(v)$ and all $(K+1)$ in Eq. 4.1 and thereafter will be replaced by $(K+2)$. Rewrite Eq.(4.9) into a vector form, we get

$$\bar{\bar{f}}_{t1} = [0, 1, 2, \ldots, K] * [\mathbf{F}_{t0}, \mathbf{F}_{t1}, \mathbf{F}_{t2}, \ldots, \mathbf{F}_{tK}]^T \qquad (t = x, y, z) \tag{4.10}$$

Add Eq.(4.10) into Eq.(4.3), we have the following matrix equations

$$\begin{bmatrix} \mathbf{C}_t \\ \sum_{m=1}^{\bar{K}} m\bar{\mathbf{F}}_{tm} \\ \bar{\bar{\mathbf{F}}}_{t1} \end{bmatrix} = \begin{bmatrix} u_0^0 & u_0^1 & u_0^2 & u_0^3 & \ldots & u_0^{K+1} \\ u_1^0 & u_1^1 & u_1^2 & u_1^3 & \ldots & u_1^{K+1} \\ u_2^0 & u_2^1 & u_2^2 & u_2^3 & \ldots & u_2^{K+1} \\ \ldots & & & & & \\ u_{K-1}^0 & u_{K-1}^1 & u_{K-1}^2 & u_{K-1}^3 & \ldots & u_{K-1}^{K+1} \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 2 & 3 & \ldots & K+1 \end{bmatrix} * \begin{bmatrix} \mathbf{F}_{t0} \\ \mathbf{F}_{t1} \\ \mathbf{F}_{t2} \\ \ldots \\ \mathbf{F}_{t(K-1)} \\ \mathbf{F}_{tK} \\ \mathbf{F}_{t(K+1)} \end{bmatrix}$$

$$(t = x, y, z) \tag{4.11}$$

With the above constructed surface function, we create a human knee which shares two boundaries with a thigh and a calf from generalized ellipses as demonstrated in Fig.4.4

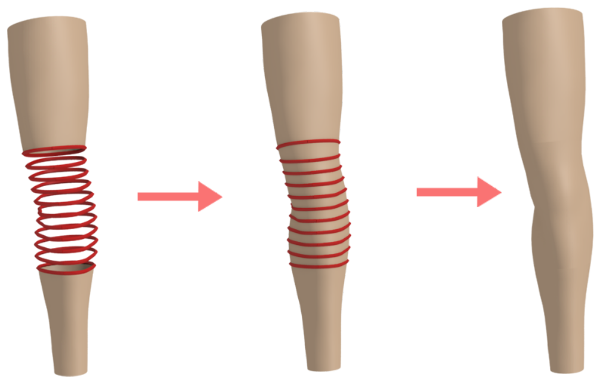


Figure 4.4: *A human knee shares two boundaries with a thigh and a calf, respectively, created from generalized ellipses. Generalized ellipses are illustrated in red.*

4.3 Surfaces defined with composite generalized elliptic segments

With the introduction of composite generalized elliptic segments, human parts can be constructed from these segments and the combinations of these segments and generalized ellipses in addition to those created from the generalized ellipses only.

Whether the curves are generated from composite generalized elliptic segments or generalized ellipses, they must have the same curve segments if a surface is to be constructed from these curves. Therefore, when some generalized ellipses are to be combined with other composite generalized elliptic segments to generate a surface, these generalized ellipses should be firstly divided into the same segments as those of composite generalized elliptic segments. For example, if we are required to construct a surface from three composite generalized elliptic segments and a closed generalized ellipse where the three composite generalized elliptic segments are connected together at $v = v_0$, $v = v_1$ and $v = v_2$, respectively, we simply divide the generalized ellipse into $v = v_0$ to $v = v_1$, $v = v_1$ to $v = v_2$, and $v = v_2$ to $v = v_0$.

The continuity of different surface patches along the v parametric direction can be achieved by taking the mathematical expressions for u parametric variable of the adjacent two surface patches to be the same form. In the following, we indicate that the surface patches constructed in this way can maintain the required continuity.

For $I+1$ generalized elliptic curves each of which consists of $J+1$ composite generalized elliptic segments or parts of a generalized ellipse, we use vector-valued position functions $\mathbf{X}_{ij}(i = 0,1,2,3,\ldots,I; j = 0,1,2,3,\ldots,J)$ to represent the jth generalized elliptic segment or the corresponding segment of a generalized ellipse for the ith generalized elliptic curves. Here $\mathbf{X}_{ij}$ has the three components x_{ij}, y_{ij} and z_{ij}. Since z_{ij} takes an identical value for all segments of the same generalized elliptic curve, the continuity for this component is always ensured. Therefore, the following treatment is for x and y components. Here I only consider positional and tangential continuities. Of course, the treatment discussed here is also applicable to the higher order continuities.

If up to tangential continuity is taken into account, both positional and tangential continuities at the joint between the jth and $(j+1)$th segments should be achieved when constructing these curve segments. That is

$$\begin{aligned}
&\mathbf{X}_{ij}(v = v_{ij}) = \mathbf{X}_{ij+1}(v = v_{ij}) \\
&\left[\frac{\partial \mathbf{X}_{ij}}{\partial v}\right]_{v=v_{ij}} = \left[\frac{\partial \mathbf{X}_{ij+1}}{\partial v}\right]_{v=v_{ij}} \\
&(i = 0,1,2,3,\ldots,I; j = 0,1,2,3,\ldots,J-1)
\end{aligned} \tag{4.12}$$

If the generalized elliptic curves are closed, the positional and tangential continuities at the closure should also be introduced which leads to the additional equations below

$$\begin{aligned}
&\mathbf{X}_{iJ}(v = 2\pi) = \mathbf{X}_{i0}(v = 0)\\
&\left[\frac{\partial \mathbf{X}_{iJ}}{\partial v}\right]_{v=2\pi} = \left[\frac{\partial \mathbf{X}_{i0}}{\partial v}\right]_{v=0}\\
&(i = 0, 1, 2, 3, \ldots, I)
\end{aligned} \tag{4.13}$$

If $(J+1)$ surface patches are constructed from the above generalized elliptic curves, we take their surface functions to be

$$\begin{aligned}
&\mathbf{S}_j(u, v) = \sum_{m=0}^{I} u^m \bar{\mathbf{X}}_{mj}(v)\\
&(j = 0, 1, 2, 3, \ldots, I)
\end{aligned} \tag{4.14}$$

where $\bar{\mathbf{X}}_{mj}(v)(m = 0, 1, 2, 3, \ldots, I; j = 0, 1, 2, 3, \ldots, J))$ are unknown functions.

At the position u_i of the ith generalized elliptic curve segment, the jth and $(j+1)$th surface patches should pass through the jth and $(j+1)$th curve segments, respectively, i. e.,

$$\begin{aligned}
&\sum_{m=0}^{I} u_i^m \bar{\mathbf{X}}_{mj}(v) = \mathbf{X}_{ij}(v)\\
&\sum_{m=0}^{I} u_i^m \bar{\mathbf{X}}_{mj+1}(v) = \mathbf{X}_{ij+1}(v)\\
&(i = 0, 1, 2, 3, \ldots, I)
\end{aligned} \tag{4.15}$$

Using the same treatment given in Section 4.2, we can determine $\bar{\mathbf{X}}_{mj}(v)$

and $\bar{\mathbf{X}}_{mj+1}(v)$ using the following matrix equations

$$\begin{aligned}
&\bar{\mathbf{X}}_j(v) = \mathbf{R}_j(u_{im})^{-1}\mathbf{X}_j(v) \\
&\bar{\mathbf{X}}_{j+1}(v) = \mathbf{R}_{j+1}(u_{im})^{-1}\mathbf{X}_{j+1}(v) \\
&(j = 0,1,2,3,\ldots,J)
\end{aligned} \tag{4.16}$$

where $\bar{\mathbf{X}}_l(v) = \left[\bar{\mathbf{X}}_{0l},\bar{\mathbf{X}}_{1l},\bar{\mathbf{X}}_{2l},\ldots,\bar{\mathbf{X}}_{Il}\right]^T$ and $\mathbf{X}_l(v) = \left[\mathbf{X}_{0l},\mathbf{X}_{1l},\mathbf{X}_{2l},\ldots,\mathbf{X}_{Il}\right]^T (l = j, j+1)$ are two column vectors.

According to Eq. (4.15), we know that $\mathbf{R}_j(u_{im})^{-1}$ and $\mathbf{R}_{j+1}(u_{im})^{-1}$ are identical which can be written as

$$\mathbf{R} = \mathbf{R}_j\left(u_{im}\right)^{-1} = \mathbf{R}_{j+1}\left(u_{im}\right)^{-1} = \mathbf{R}_{ij} \tag{4.17}$$

where $\mathbf{R}_{ij}(i = 0,1,2,3,\ldots,I; j = 0,1,2,3,\ldots,I)$ are the elements of the square matrix $\mathbf{R}$.

With Eq. (4.17), we can obtain the mathematical expressions of the elements in vectors $\bar{\mathbf{X}}_j(v)$ and $\bar{\mathbf{X}}_{j+1}(v)$

$$\begin{aligned}
&\bar{\mathbf{X}}_{lj}(v) = \sum_{i=0}^{I} R_{li}\mathbf{X}_{ij}(v) \\
&\bar{\mathbf{X}}_{lj+1}(v) = \sum_{i=0}^{I} R_{li}\mathbf{X}_{ij+1}(v) \\
&(l = 0,1,2,3,\ldots,I)
\end{aligned} \tag{4.18}$$

Substituting Eq.(4.18) into Eq.(4.14), the position functions of the jth and $(j+1)$th surface patches are found to be

$$\begin{aligned}\mathbf{S}_j(u,v) &= \sum_{m=0}^{I} u^m \sum_{i=0}^{I} R_{mi}\mathbf{X}_{ij}(v)\\ \mathbf{S}_{j+1}(u,v) &= \sum_{m=0}^{I} u^m \sum_{i=0}^{I} R_{mi}\mathbf{X}_{ij+1}(v)\end{aligned} \tag{4.19}$$

From Eq. (4.19), we can calculate the first partial derivatives of the jth and $(j+1)$th surface patches with respect to the parametric variable v. Taking into account the positional and tangential continuity conditions Eq.(4.12) of $\mathbf{X}_{ij}(v)$ and $\mathbf{X}_{ij+1}(v)$ at their joint v_{ij}, we find

$$\begin{aligned}&\mathbf{S}_j(u,v_{ij}) = \mathbf{S}_{j+1}(u,v_{ij})\\ &\left[\frac{\partial \mathbf{S}_j(u,v)}{\partial v}\right]_{v=v_{ij}} = \left[\frac{\partial \mathbf{S}_{j+1}(u,v)}{\partial v}\right]_{v=v_{ij}}\end{aligned} \tag{4.20}$$

Equation (4.20) indicates that along the shared boundary curve between the jth and $(j+1)$th surface patches, both positional and tangential continuities are guaranteed.

With the same method given in Section 4.2, the continuity of surface patches at $u=0$ and $u=1$ along u parameter direction can also be tackled.

Fig. 4.5 gives an example of generating composite generalised elliptic surface segments: original cross-sectional curves in Fig. 4.5(a) are splited into four parts and then transformed into composite elliptic curve segments in Fig. 4.5(b) with the methods discussed in Chapter 3, each segment is highlighted with a different color. Finally I deal with the composite elliptic curve segments using the approach introduced in this chapter and then build composite elliptic surface segments shown in Fig. 4.5(c).

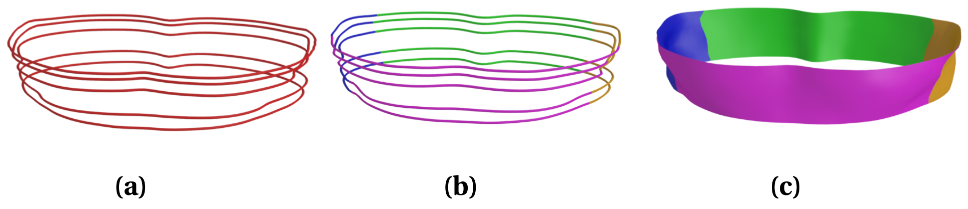


Figure 4.5: *Generation of surface patches from composite generalized elliptic segments*

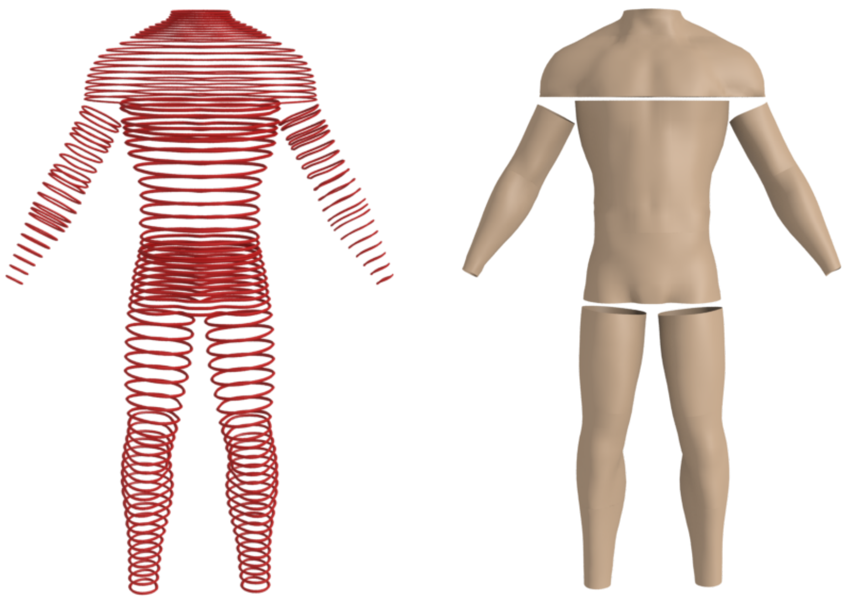
4.4 Conclusions

Detailed character models can be defined by cross-sectional curves. These cross-sectional curves can be mathematically represented with generalized ellipses and composite generalized elliptic segments. In this chapter, I have presented a method to create surface patches from cross-sectional curves represented with generalized ellipses and composite generalized elliptic segments.

The method starts from some characteristic cross-sectional curves of human body. With introduction of generalized elliptic curves, these cross sectional curves are approximated very accurately with generalized ellipses or composite generalized elliptic segments. Then human parts are constructed from these generalized elliptic curves. This chapter presented the algorithms to determine generalized ellipses and composite generalized elliptic segments, investigate the continuity between two adjacent segments of a generalized elliptic curve. This chapter examined how to construct human parts from the generalized ellipses and composite generalized elliptic segments and demonstrated that surface patches produced from generalized elliptic curves can maintain the required continuities between two adjacent patches.

Fig. 4.6 gave a human model built with my proposed approach. The left image contains the cross-sectional curves of a human body and the right image is the assembled human body made up of various generalized elliptic surfaces. The approach presented in this paper greatly lowers the computer storage. For the human body given in the following figures, the modelling with the polygon approach uses 12,426 vertices to keep the information of the human body. With my method, only 7530 constants are required to build the human body. The number of constants sorely depends on the cross-sectional curves that are fed into the algorithms as inputs. To be more specific, the amount of constants in $\mathbf{F}_{tm}$ $(m = 0, 1, \ldots, K-1; t = x, y, z)$ depends on the number of cross-sections(K) and the accuracy of cross-sectional curves as discussed in Chapter 3. Hence, this method can be used in the scenarios where LOD(level of detail) is applied. For example in games and animated films, the characters far from the viewers do not require high visual quality can be modeled with lower number of cross-sections(K) and less accuracy of cross-sectional curves to increase rendering computational efficiency. For now the optimal number of cross-sectional curves required to represent a surface is still a result of manual trials. It is recommended that those going through critical parts of the body (e.g. places with radical curvature changes, etc.) should be added to the cross-section set.

Unlike sweep-based human modelling, my method can represent cross-sectional curves of human body very accurately. Due to this reason, the additional operations of improving modelling realism using some approaches such as displacement map (Hyun et al. [2005]) are not necessary.



(a) **(b)**

Figure 4.6: *Human model assembled from different human parts: a) original cross-sectional curves sampled from base mesh, b) various generalised elliptic surfaces are generated from the cross-sectional curves*

Chapter 5

Primitive Generator

This chapter presents a novel sketch-guided and ODE-drive primitive deformer to address the thin-shell energy function algorithm described by Terzopoulos et al. [1987], with the implementation of the Finite Difference Method as numerical solution.

5.1 Pipeline Overview

As shown in the Fig.5.1, the first step of my proposed approach is to generate 2D sketches. Two methods are included in my developed system to generate 2D images. They are manually drawing without a reference image and extracting a 2D sketch from a reference image. After users extract 2D silhouette contours from sketches, or directly draw 2D silhouette contours, they put the super-ellipsoid at the place of the input sketch silhouette and set proper parameters of the super-ellipsoid. The input sketch can be messy and there may be problems like over-sketch, so the sketch strokes need to be pre-processed which includes stroke grouping and curve beautification. However the pre-processing of the sketches is beyond this thesis's scope, so here refer the readers to more specialized literature on these topics (Douglas & Peucker [1973]; Bae et al. [2008]). Afterward, an ODE-based deforming method will deform the cross-section curves of super-ellipsoid primitives so that the projection of deformed cross-section curves exactly matches the input sketch silhouette. Finally, the deformed cross-section curves will be fitted with the method introduced in Chapter 3 and new surface constructed from these cross-section curves will be created with the technique described in Chapter 4.

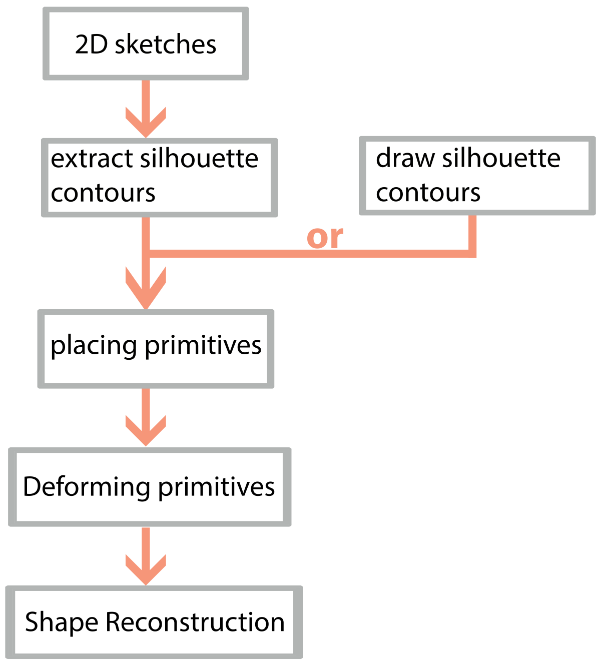


Figure 5.1: *pipeline overview.*

5.2 User Interface of Primitive Deformer

The user interface of my developed primitive deformer uses four windows as shown in Figure 5.2. The upper left window is used to display 3D base mesh without primitive deformations in the front view. The upper right window is used to draw and edit 2D silhouette contours for the primitive. The bottom left window is used to deform the primitive in the front view. The bottom right window is used to deform the primitive in the side view so that the 3D model obtains its realistic shape with its projected silhouettes exactly matching the input sketched silhouettes.

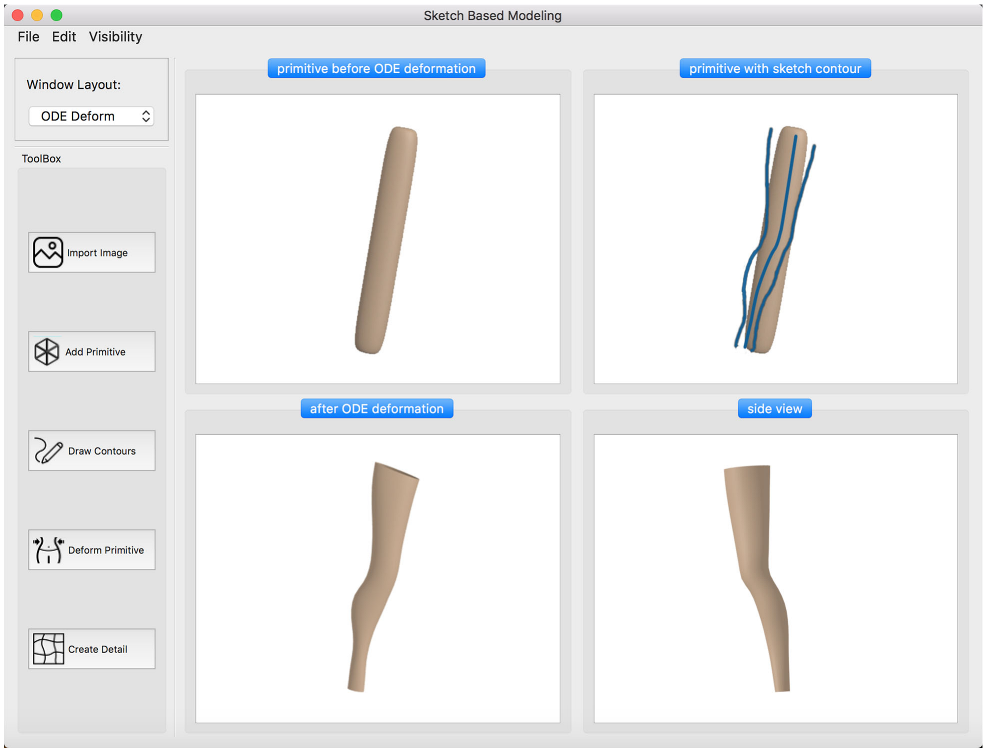


Figure 5.2: *Interface for primitive deformer: (a) 3D base mesh of the left leg of the female warrior without primitive deformations, (b) 2D silhouette contours of the female warrior leg sketch and primitive, (c) and (d) front view and side view of the 3D left leg base mesh after primitive deformations*

5.3 Thin Shell Deform Energy

After 3D primitives have been placed and aligned with the generated 2D silhouette contours, these 3D primitives should be deformed so that their 2D silhouette contours can match the generated 2D silhouette contours exactly. Here we use the example shown in Figure 5.3 to demonstrate the algorithm of my proposed primitive deformer and how it deforms a 3D primitive to match the 2D silhouette contours.

Figure 5.3: *Primitive deformer: a) female warrior torso represented with a cylinder and its 2D silhouette contour, b) deformed shape of the cylinder*

Figure 5.3a depicts a torso model of the female warrior represented with a cylinder-shaped superellipsoid. The 2D silhouette contour to be matched is also shown in the image. Figure 5.3b shows how the cylinder is deformed with the algorithm developed below to match the 2D silhouette contour exactly.

To tackle the above problem, we propose a sketch-guided and ODE-drive primitive deformer. It is developed from a simplified version of the Euler-Lagrange PDE (partial differential equation) which is widely used in physical-based surface deformations and briefly introduced below.

As discussed in Botsch & Sorkine [2008], the main requirement for physically-based surface deformations is an elastic energy which considers the local stretching and bending of two-manifold surfaces called thin-shells. When a surface $\mathbb{S} \subset \mathbb{R}^3$, parameterized by a function $\mathbf{P}(u,v) : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{S} \subset \mathbb{R}^3$, is deformed to a new shape $\mathbb{S}'$ through adding a displacement vector $\mathbf{d}(u,v)$ to each point $\mathbf{P}(u,v)$, the change of the first and second fundamental $I(u,v)$, $\Pi(u,v) \in \mathbb{R}^{2\times 2}$ forms in differential geometry (Do Carmo et al. [2017]) yields a measure of stretching and bending, as described in Terzopoulos et al. [1987]:

$$E_{shell}(\mathbb{S}') = \int_{\Omega} k_s \|I' - I\|_F^2 + k_b \|\Pi' - \Pi\|_F^2 \,\mathrm{d}u\,\mathrm{d}v \tag{5.1}$$

where I' and Π' are the first and second fundamental forms of the surface $\mathbb{S}'$, $\|.\|$ indicates a (weighted) Frobenius norm, and the stiffness parameters k_s and k_b are used to control the resistance to stretching and bending.

Generating a new deformed surface requires the minimization of the above Equation (5.1), which is non-linear and computationally expensive for interactive applications. In order to avoid the nonlinear minimization, the change of the first and second fundamental forms is replaced by the first and second order partial derivatives of the displacement function $\mathbf{d}(u,v)$ (Celniker & Gossard [1991]; Welch & Witkin [1992]), i. e.,

$$\tilde{E}_{shell}(\mathbf{d}) = \int_{\Omega} k_s(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2) + k_b(\|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\| + \|\mathbf{d}_{vv}\|^2) \,\mathrm{d}u\,\mathrm{d}v, \tag{5.2}$$

where $\mathbf{d}_x = \frac{\partial}{\partial x}\mathbf{d}$, $\mathbf{d}_{xy} = \frac{\partial^2}{\partial x \partial y}\mathbf{d}$ and $\mathbf{d}_{xx} = \frac{\partial^2}{\partial x^2}\mathbf{d}$. The minimization of the above equation can be obtained by applying variational calculus, which leads to the following Euler-Lagrange PDE:

$$-k_s \triangle \mathbf{d} + k_b \triangle^2 \mathbf{d} = 0, \tag{5.3}$$

where $\triangle$ and $\triangle^2$ are the Laplacian and the bi-Laplacian operators, respectively,

$$\begin{aligned} \triangle\mathbf{d} &= \nabla\cdot\nabla\mathbf{d} = \mathbf{d}_{uu} + \mathbf{d}_{vv}, \\ \triangle^2\mathbf{d} &= \triangle(\triangle\mathbf{d}) = \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv} \end{aligned} \tag{5.4}$$

Up till now, the minimization of the thin-shell deform energy is described in a continuous space which depicts the physical phenomenon happening in real world. In next section, we will discrete Eq. (5.3) for computers to simulate.

5.4 Finite Difference Method for Geometric Processing

In this section, a discretized ODE-driven and sketch-guided primitive deformation method is proposed. For a long time, ODEs have been widely applied in scientific computing and engineering analyses to describe the underlying physics. For example, fourth-order ODEs have been used to describe the lateral bending deformations of elastic beams. Introducing ODEs into geometric processing can create physically realistic appearances and deformations of 3D models. ODE-based sweeping surfaces (You et al. [2007]), ODE-based surface deformations (You et al. [2010]; Chaudhry et al. [2013]), and ODE-based surface blending (You et al. [2014]) have also been developed previously.

Although researchers studied ODE-based geometric surface creation and deformations, how to use ODE-based modelling to deform geometric primitives and create new shapes from the user's drawn sketches has been under-explored to date.

The work presented in this chapter falls within the category of direct mesh generation from sketches. It integrates quick creation of primitive-based base meshes, efficient ODE-driven primitive deformations, and shape generation from sketches in three orthographic views, in order to create detailed 3D models.

Using the sketched 2D silhouette contours shown in Fig. 5.5 to change the shape of the primitive can be transformed into the generation of a sweeping surface which passes through the two sketched 2D silhouette contours. The generator that creates the sweeping surface is a curve of the parametric variable u only, and the two silhouette contours are trajectories. If Equation (5.3) is used to describe the generator, the parametric variable v in Equation (5.3) drops, and we have $\mathbf{d}_{vv} = 0$ and $\mathbf{d}_{vvvv} = 0$. Substituting $\mathbf{d}_{vv} = 0$ and $\mathbf{d}_{vvvv} = 0$ into Equation (5.4), we obtain the following simplified version of the Euler-Lagrange PDE, seen as (5.5), which is actually a vector-valued ODE.

$$k_b \frac{\partial^4 \mathbf{d}}{\partial u^4} - k_s \frac{\partial^2 \mathbf{d}}{\partial u^2} = 0. \tag{5.5}$$

As pointed out in Chaudhry et al. [2013] and Chaudhry et al. [2015], the finite difference solution to ordinary differential equations is very efficient, we here investigate such a numerical solution to Equation (5.5).

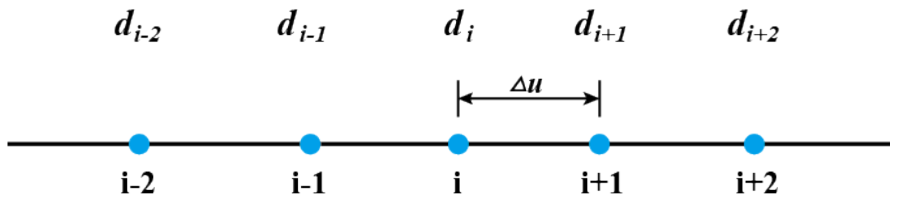


Figure 5.4: *Typical node i for the finite difference approximations of derivatives*

For a typical node shown in Figure 5.4, the central finite difference approximations of the second and fourth order derivatives can be written as Chaudhry et al. [2013]:

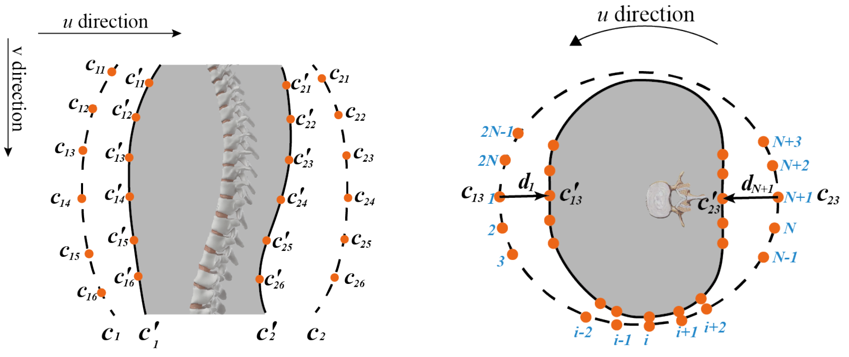
$$\begin{aligned}
\frac{\partial^2 \mathbf{d}}{\partial u^2}|_i &= \frac{1}{\triangle u^2}(\mathbf{d}_{i+1} - 2\mathbf{d}_i + \mathbf{d}_{i-1}), \\
\frac{\partial^4 \mathbf{d}}{\partial u^4}|_i &= \frac{1}{\triangle u^4}[6\mathbf{d}_i - 4(\mathbf{d}_{i-1} + \mathbf{d}_{i+1}) + \mathbf{d}_{i-2} + \mathbf{d}_{i+2}].
\end{aligned} \tag{5.6}$$

Introducing Equation (5.6) into Equation (5.5), the following finite difference equation at a representative node i can be written as:

$$(6k_b + 2k_s h^2)\mathbf{d}_i + k_b\mathbf{d}_{i-2} + k_b\mathbf{d}_{i+2} - (4k_b + k_s h^2)\mathbf{d}_{i-1} - (4k_b + k_s h^2)\mathbf{d}_{i+1} = 0. \tag{5.7}$$

5.5 Algorithm for Primitive Deformer

For character models, the 3D shape defined by two silhouette contours is closed in the parametric direction u as indicated in Figure 5.5b. Therefore, we can extract some closed curves each of which passes through the two corresponding points on the two silhouette contours. Taking the silhouette contours in Figure 5.5a as an example, we sample the two curves according to arc-length and find two corresponding points $\mathbf{c}_{13}$ and $\mathbf{c}_{23}$ (points with the same index) on the original silhouette contours $\mathbf{c}_1$ and $\mathbf{c}_2$, and two corresponding points $\mathbf{c}'_{13}$ and $\mathbf{c}'_{23}$ on the deformed silhouette contours $\mathbf{c}'_1$ and $\mathbf{c}'_2$ as shown in Figure 5.5b. Then, we extract a closed curve $\mathbf{c}(u)$ passing through the two corresponding points $\mathbf{c}_{13}$ and $\mathbf{c}_{23}$ from the 3D model in Figure 5.5a and depict it as a dashed curve in Figure 5.5b. Assuming that the deformed shape of the closed curve $\mathbf{c}(u)$ is $\mathbf{c}'(u)$, the displacement difference between the original closed curve and the deformed closed curve is $\mathbf{d}(u) = \mathbf{c}'(u) - \mathbf{c}(u)$. My task is to find the displacement difference $\mathbf{d}(u)$ and generate the deformed curve $\mathbf{c}'(u) = \mathbf{d}(u) + \mathbf{c}(u)$.



(a) *side view* **(b)** *top view*

Figure 5.5: *Finite difference nodes for local shape manipulation from sketches in different view planes.*

In order to use the finite difference method to find the displacement difference $\mathbf{d}(u)$, we uniformly divide the closed curve into $2N$ equal intervals as indicated in Figure 5.5b. The displacement difference at node 0 and node N is known, i. e. $\mathbf{d}_0 = \mathbf{c}'_{13} - \mathbf{c}_{13}$ and $\mathbf{d}_N = \mathbf{c}'_{23} - \mathbf{c}_{23}$.

We take the deformation of node 0 and node N as a consequence of applying external force f_0 and f_N and as such we invent a term for force f and add it to the right side of Eq.(5.7) as follows:

$$(6k_b + 2k_s h^2)\mathbf{d}_i + k_b \mathbf{d}_{i-2} + k_b \mathbf{d}_{i+2} - (4k_b + k_s h^2)\mathbf{d}_{i-1} - (4k_b + k_s h^2)\mathbf{d}_{i+1} = f_i. \tag{5.8}$$

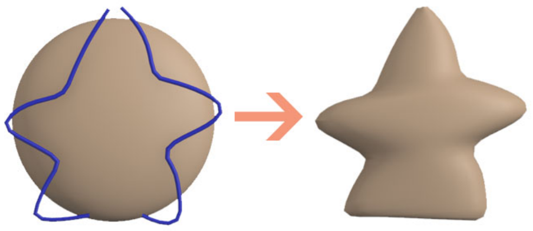
When we write the finite difference equations for the nodes $1, 2, 2N-2$ and $2N-1$, the node 1 will be involved, and we have $\mathbf{d}_0 = \mathbf{c}'_{13} - \mathbf{c}_{13}$. The finite difference equations at these points can be derived from Equation (5.7). Substituting $\mathbf{d}_0 = \mathbf{c}'_{13} - \mathbf{c}_{13}$ into these equations, we obtain the finite difference equations for the nodes 1, 2, $2N-2$ and $2N-1$, and present them in Appendix A.

When we write the finite difference equations for the nodes $N-2$,$N-1$, $N+1$ and ,$N+2$ the node N will be involved, and we have $\mathbf{d_N} = \mathbf{c'_{23}} - \mathbf{c_{23}}$. Once again, the finite difference equations at these points can be derived from Equation (5.7). Substituting $\mathbf{d}_N = \mathbf{c}'_{23} - \mathbf{c}_{23}$ into these equations, we obtain the finite difference equations for the nodes $N-2$, $N-1$, $N+1$ and $N+2$, and present them in Appendix A as well.

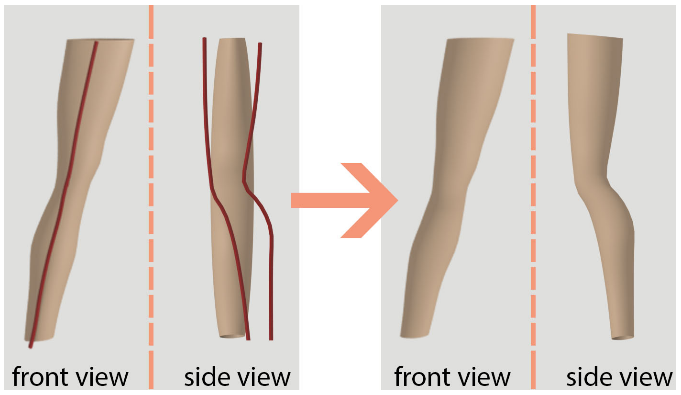
For all other nodes 3, 4, 5,..., $N-4$, $N-3$ and $N+3$,$N+4$,..., $2N-4$, $2N-3$, the finite difference equations are the same as Equation (5.7). For these nodes, there are $2N-8$ finite difference equations. Plus the 8 finite difference equations at node 1, 2, $N-2$, $N-1$, $N+1$, $N+2$, $2N-2$ and $2N-1$, we get $2N$ linear algebra equations which can be solved to determine the $2N$ unknown constants f_0, $\mathbf{d}_1$, $\mathbf{d}_2$, ...,$\mathbf{d}_{N-2}$, $\mathbf{d}_{N-1}$, f_N, $\mathbf{d}_{N+1}$, $\mathbf{d}_{N+2}$,..., $\mathbf{d}_{2N-1}$, and $\mathbf{d}_{2N}$. Adding $\mathbf{d}_i$ ($i = 0, 1, \ldots, 2N-2, 2N-1$) to the original curve $\mathbf{c}(u)$, we obtain the deformed curve $\mathbf{c}'(u)$, and depict it as a solid curve in Figure 5.5b. Repeating the above operations for all other points on the two silhouette contours, we obtain all deformed curves. These curves together describe a new 3D deformed shape.

Finally, we need to reconstruct the 3D model from these deformed curves. With the curve fitting method introduced in Chapter. 3, these deformed curves are now represented with trigonometric series and can be seen as cross-sectional curves. Using the algorithm in Chapter. 4, we create a new 3D deformed shape from these deformed curves.

Other applications of this method other than human character modeling include more generic organic objects, and deformations responding to free form curves. Fig 5.6a has shown this ODE-driven deforming method has good performance when it is used to create a sea-star shaped organic object. One can also use this deformer for multiple times to obtain the ideal shape if once is not enough. One such example is shown in Fig 5.6b where the leg has been deformed responding to the free form curves after initially deformed by single-view sketch strokes. A few more words for why not let one curve cover the whole sketch instead of splitting the whole surface into multiple sections: because the demand for 3D branching structures in human body(shoulder vs. arms, torso vs. legs), in this research multiple sections are needed.



(a) *the deformation process of an organic shape represented by an ellipsoid and its 2D silhouette contour, and the deformed shape of the ellipsoid*



(b) *a leg that has been deformed by single-view sketch strokes before, now get further deformed in accordance with the free form red-colored curves*

Figure 5.6: *Additional applications of ODE-driven Sketch-based deformations*

However, since the cross-section shape of a superellipsoid lacks variations and cannot capture the complexity of actual objects, deforming from a superellipsoid primitive alone is not ideal. In such cases, we edit the cross-section first (Fig.5.7b), and then deform the shape again with the following treatment.

When a cross-sectional contour is used to modify the cross-section shape, the influence range of the cross-sectional contour is first specified to generate two boundary curves. The cross-sectional contour and the two boundary curves are parameterized to find the corresponding points. If only the position continuity is required, the three corresponding points: two are on each of the two boundary curves and the third is on the cross-sectional contour, are taken to be the nodes of the finite difference calculations and introduced into Equation (5.7) to reduce three unknown constants. If both position and tangent continuities are required, the first partial derivatives at the boundary curves are obtained from the original model and introduced into Equation (5.7) to reduce two more unknown constants.

With the above ODE-driven deformations, realistic cross-section shapes are created. Taking the vest created in Chapter 6.3 and shown in Figure 5.7a as an example, one original cross-section curve of the vest is an ellipse highlighted in blue in Figure 5.7b, and the real cross-section curve of the vest on a female body should be the red one shown in the same figure. The real cross-section curve in red and the two boundary curves highlighted in blue are depicted in Figure 5.7c. The cross-section shape within the boundary curves is modified and shown in Figure 5.7d.

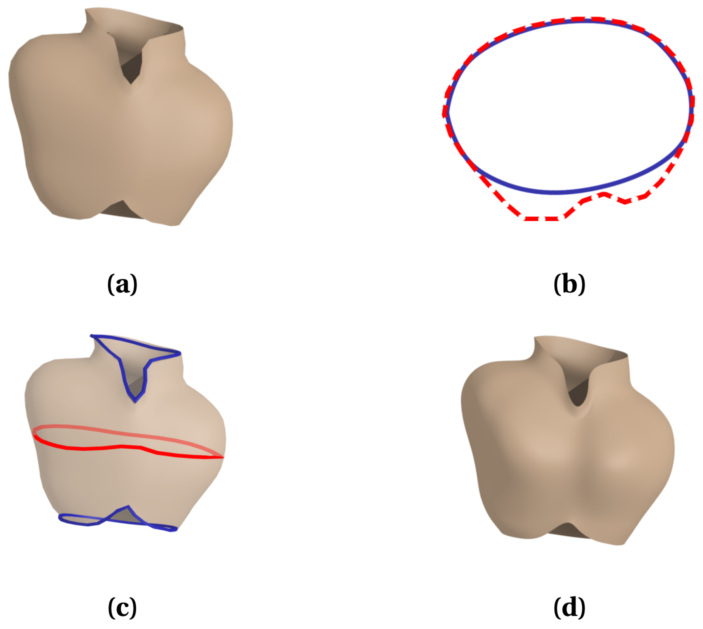


Figure 5.7: *Cross-section shape modifications by cross-section contours: a) vest, b) the origin cross section contour (in blue) and a modified cross section contour (in red), c) the modified cross section is in place, d) 3D shape after cross section contour modifications.*

5.6 Conclusion

By now, we've achieved the goal of generate primitives from silhouette sketches in two orthogonal view-planes by solving an ODE equation with a finite different method. As shown in Fig. 5.8, the deformed primitives have matched the generated 2D silhouette contours exactly.

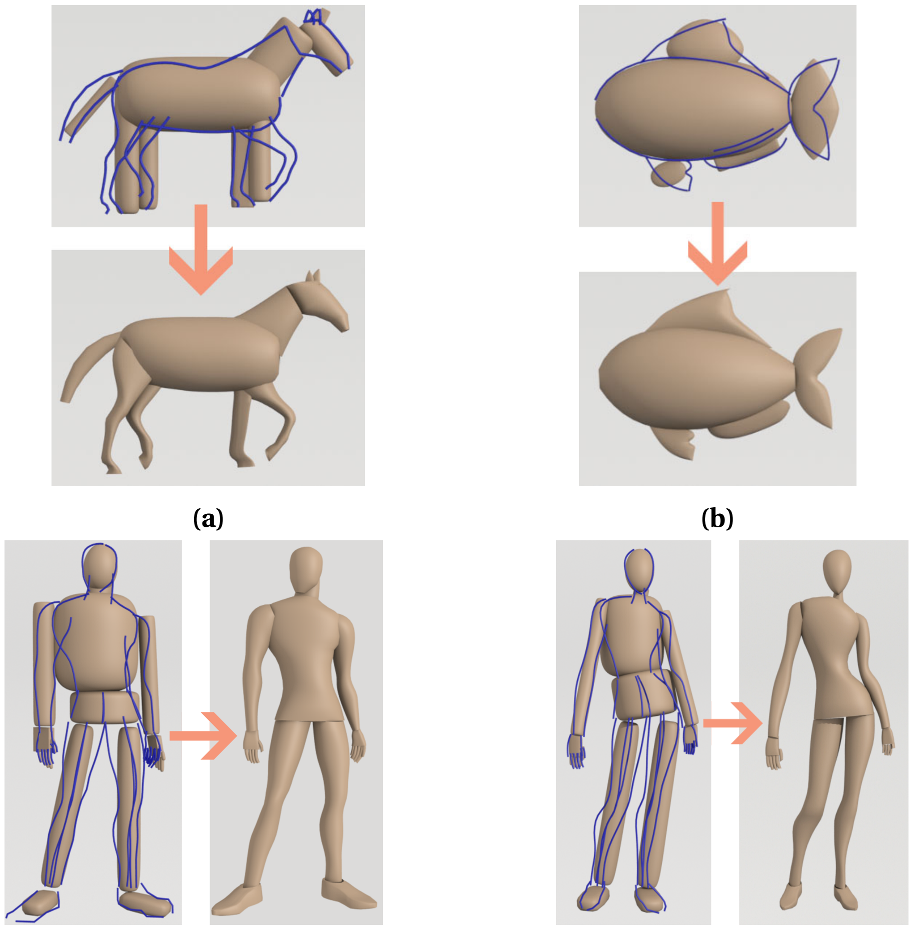


Figure 5.8: *Examples of the primitive deformation method*

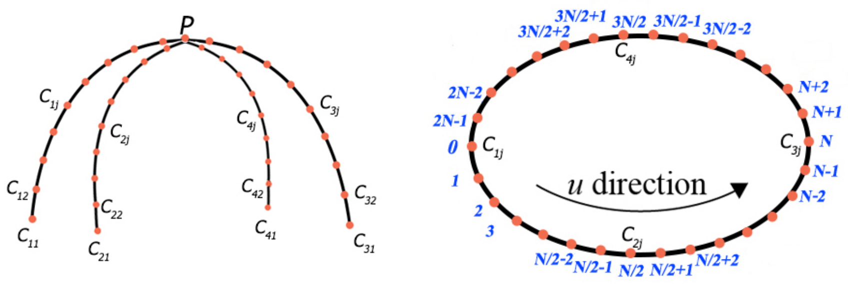
Chapter 6

Free-form Patch Generator

The mesh generated from the deformed primitives described in Chapter 5 only represents a base mesh. Therefore, we need to add 3D details to the base mesh to generate a detailed 3D model. In this section, I will develop ODE-driven shape generators to create various local shapes from the user's drawn sketches in different view planes. These sketches can be divided into the following cases: (i) two open sketches in two different view planes, (ii) one open and one closed sketches in two different view planes, and (iii) two open and one closed sketches in three different view planes. In the following, I will describe the algorithms to create 3D shapes from the three different cases of sketches.

6.1 Two Open Silhouette Contours in Two Different View Planes

For the creation of a local 3D model from two open silhouette contours in two different view planes, we use the intersecting point **P** to segment each of the two sketched 2D silhouette contours obtained from the front view and the side view into two curves and denote all the four segmented curves with $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ and $\mathbf{c}_4$. Then, we find the four corresponding points on the four curves, and denote them with $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, $\mathbf{c}_{3j}$ and $\mathbf{c}_{4j}$, respectively as shown in Figure 6.1. Since the four points are on the two sketched 2D silhouette contours in the front view and the side view, their coordinate values are known.



(a) *sketched contours* **(b)** *generator to be created*

Figure 6.1: *The finite difference nodes used in the algorithm in the case of two open silhouette contours in two different view planes.*

The 3D model to be created from the two sketched 2D silhouette contours can be regarded as a sweeping surface whose generator is a closed curve $\mathbf{c}(u)$, shown in Figure 6.1b, passing through the four corresponding points $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, $\mathbf{c}_{3j}$ and $\mathbf{c}_{4j}$ (shown in Figure 6.1a and b). We uniformly divide the domain of the parametric variable u corresponding to the closed curve (generator) $\mathbf{c}(u)$ into $2N$ equal intervals. The nodes corresponding to $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, $\mathbf{c}_{3j}$ and $\mathbf{c}_{4j}$ are 0, $N/2$, N, and $3N/2$, respectively, as illustrated in Figure 6.1b. Here $N \geq 4$ is an even number.

At the nodes 0, $N/2$, N, and $3N/2$, the coordinate values are known, i.e., $\mathbf{d}_0 = \mathbf{c}_{1j}$, $\mathbf{d}_{N/2} = \mathbf{c}_{2j}$, $\mathbf{d}_N = \mathbf{c}_{3j}$, and $\mathbf{d}_{3N/2} = \mathbf{c}_{4j}$, but the sculpting forces $\mathbf{f}_0$, $\mathbf{f}_{N/2}$, $\mathbf{f}_N$, and $\mathbf{f}_{3N/2}$ are unknown. At all of the other nodes shown in Figure 6.1b, there are no sculpting forces, but the displacements at these nodes are unknown.

Writing the finite difference equations for all the nodes from 0 to $2N-1$ and considering the known displacements at nodes 0, $N/2$, N, and $3N/2$, we obtain $2N$ linear equations. Solving the $2N$ linear equations, we can obtain the generator consisting of the points $\mathbf{d}_i (i = 0, 2, 3, \ldots, 2N-1)$.

Repeating the above process for all other points on the four curves $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ and $\mathbf{c}_4$, we can obtain the generators at the other positions. With these generators and the method described in Chapter 4, we can create local 3D shapes.

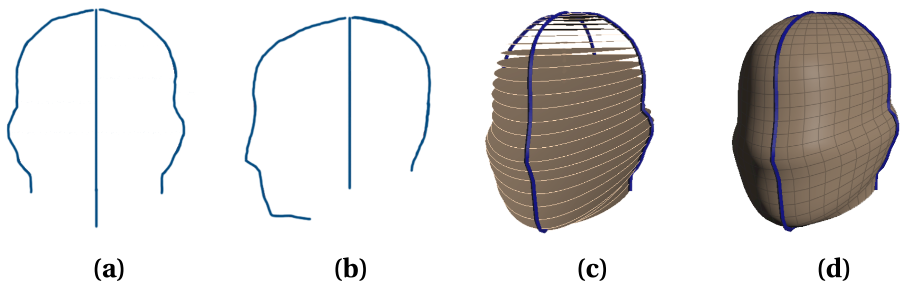


Figure 6.2: *Shape generators: a) 2D silhouette contours of a male head in front view, b) 2D silhouette contours of the male head in side view, c) cross-sections of the male head, d) creating the male head model from the cross-section curves.*

A head model shown in Figure 6.2 is used to demonstrate the above method. Figure 6.2a and Figure 6.2b show two open head silhouette contours in two different view plans. Figure 6.2c shows the cross section for each discrete parameter u_i. Figure 6.2d depicts the created 3D head model.

The above method can be directly extended to deal with local shape creation from four disconnected silhouette contours. The silhouette contours of a leg model in the front and side views are shown in Figure 6.3a and Figure 6.3b, respectively. The corresponding four points on the four silhouette contours are used to define a cross-section curve. All of these cross-section curves are used to generate a 3D leg model with the front and side views in Figure 6.3c and Figure 6.3d, respectively.

Another example given in Figure 6.3 is a neck model. Figure 6.3e shows the 4 disconnected contours, Figure 6.3f shows the cross-section curves, and the final mesh is shown in Figure 6.3g.

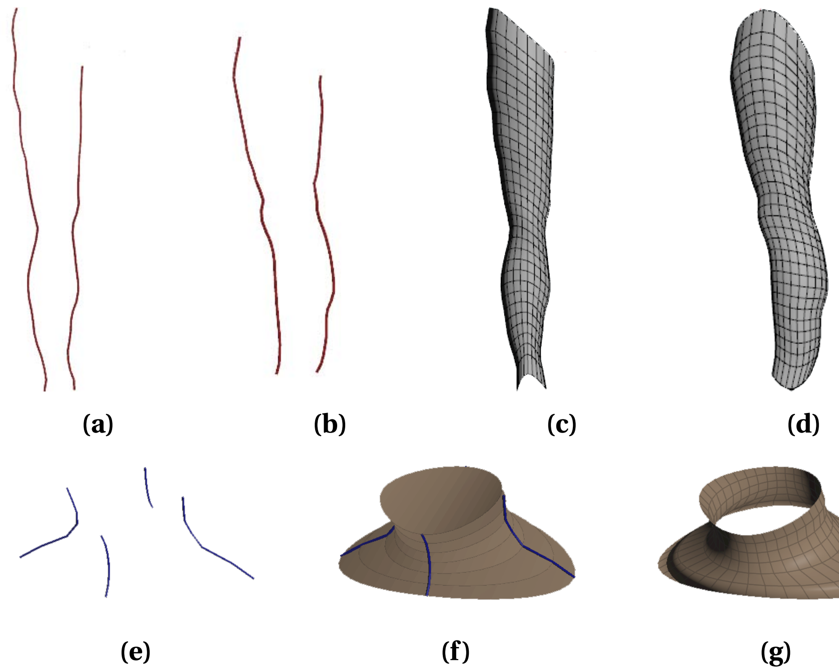


Figure 6.3: *Generation of 3D shapes from two open silhouette contours in two different view planes. a) contours in front view, b) contours in side view, c) leg mesh by using my method in front view, d) leg mesh in side view, e) contours of a neck, f) cross sections of a neck generated by my method, g) final neck mesh.*

6.2 One Open and One Closed Silhouette Contours in Two Different View Planes

For the creation of a local 3D model from one open and one closed silhouette contours in two different view plans as shown in Figure 6.4, the intersecting points $\mathbf{P}_1$ and $\mathbf{P}_3$ between the open silhouette contour and the closed silhouette contour divide the closed silhouette contour into two curves. Then we find the middle points $\mathbf{P}_2$ and $\mathbf{P}_4$ of the two curves. These four points $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{P}_4$ divide the closed silhouette contour into four curves $\mathbf{c}_1$, $\mathbf{c}_3$, $\mathbf{c}_4$, and $\mathbf{c}_6$. Next, we find the middle point $\mathbf{P}$ of the open silhouette contour, which divides the open silhouette contour into two curves $\mathbf{c}_2$ and $\mathbf{c}_5$. With this treatment, the creation of the local 3D detail model is changed into creation of two sweeping surfaces: one is defined by $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$, and the other is defined by $\mathbf{c}_4$, $\mathbf{c}_5$, and $\mathbf{c}_6$.

Since the creation processes of the two sweeping surfaces are the same, we take the creation of the sweeping surface defined by $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ to illustrates the process. For each of $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$, we uniformly divide it into $2N$ equal intervals, and use $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, and $\mathbf{c}_{3j}$ $(j = 1, 2, \ldots, J)$ to respectively indicate the nodes on the three curves.

The sweeping surface can be generated by sweeping a generator $\mathbf{c}(u)$ passing through the points $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, and $\mathbf{c}_{3j}$. Based on this consideration, the creation of the sweeping surface is transformed into the determination of the generator at different positions along the curve.

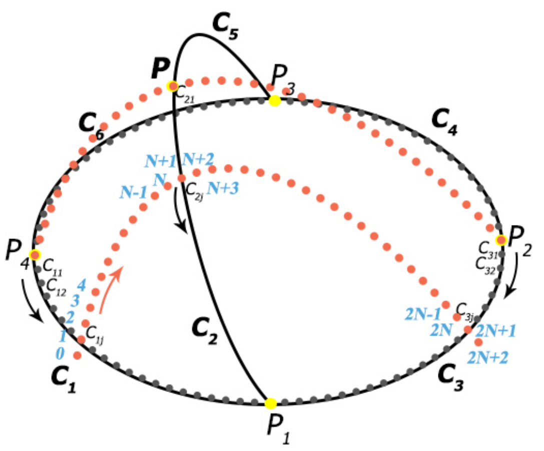


Figure 6.4: *Finite difference nodes for one open and one closed silhouette contours.*

In order to create the generator $\mathbf{c}(u)$ passing the points $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$ and $\mathbf{c}_{3j}$, we uniformly divide the domain of the parametric variable u corresponding to the generator $\mathbf{c}(u)$ into $2N$ equal intervals, and obtain the nodes 1, 2, ..., $2N-1$, $2N$ and $2N+1$. Since the nodes 1, $N+1$, and $2N+1$ are on the curves $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$, respectively, their values are known, i. e., $\mathbf{d}_1 = \mathbf{c}_{1j}$, $\mathbf{d}_{N+1} = \mathbf{c}_{2j}$, and $\mathbf{d}_{2N+1} = \mathbf{c}_{3j}$. When writing the finite difference equations for the node 2 and the node $2N$, the node 0 beyond the boundary node 1 and the node $2N+2$ beyond the boundary node $2N+1$ will be involved. They can be determined below:

If the created local 3D model is to be smoothly connected to another 3D model, we can obtain a closed curve close to the closed silhouette contour from another 3D model. Then, the vector-valued first derivative $\mathbf{T}_{1j}$ on the curve $\mathbf{c}_1$ and $\mathbf{T}_{3j}$ on the curve $\mathbf{c}_3$ can be calculated from the closed curve and the closed silhouette contour.

Otherwise, if the created local 3D model is to be connected to another 3D model with positional continuity only, the vector-valued first derivative $\mathbf{T}_{1j}$ on the curve $\mathbf{c}_1$ and $\mathbf{T}_{3j}$ on the curve $\mathbf{c}_3$ can be estimated from the three points $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, and $\mathbf{c}_{3j}$ by first constructing a curve passing through the three points and then calculating the vector-valued first derivative of the constructed curve at the points $\mathbf{c}_{1j}$ and $\mathbf{c}_{3j}$. Once the vector-valued first derivatives $\mathbf{T}_{1j}$ and $\mathbf{T}_{3j}$ are obtained, the nodes 0 and $2N+2$ can be determined by the following finite difference approximation:

$$\begin{aligned}\mathbf{T}_{1j} &= \frac{\mathbf{d}_2 - \mathbf{d}_0}{2\triangle u},\\ \mathbf{T}_{3j} &= \frac{\mathbf{d}_{2N+2} - \mathbf{d}_{2N}}{2\triangle u}.\end{aligned}$$

By solving the above equation, we can obtain the nodes 0 and $2N+2$ as follows:

$$\mathbf{d}_0 = \mathbf{d}_2 - 2\mathbf{T}_{1j} \triangle u,$$

$$\mathbf{d}_{2N+2} = \mathbf{d}_{2N} + 2\mathbf{T}_{3j} \triangle u.$$

At the nodes 1, $N+1$, and $2N+1$, the coordinate values are known, i. e., $\mathbf{d}_1 = \mathbf{c}_{1j}$, $\mathbf{d}_{N+1} = \mathbf{c}_{2j}$ and $\mathbf{d}_{2N+1} = \mathbf{c}_{3j}$. The sculpting forces $\mathbf{f}$ at nodes 0, 2, 3, ..., N, $N+2$, ..., $2N$, $2N+2$ equal zero, but the sculpting forces $\mathbf{f}_1$, $\mathbf{f}_{N+1}$ and $\mathbf{f}_{2N+1}$ are unknown.

Writing the finite difference equations for all the nodes from 2 to $2N$, we obtain $2N-1$ linear equations given in Appendix C. Taking the four boundary conditions $\mathbf{d}_1 = \mathbf{c}_{1j}$, $\mathbf{d}_{2N+1} = \mathbf{c}_{3j}$, $\mathbf{d}_0 = \mathbf{d}_2 - 2\mathbf{T}_{1j} \triangle u$, and $\mathbf{d}_{2N+2} = \mathbf{d}_{2N} + 2\mathbf{T}_{3j} \triangle u$ into consideration, we can obtain $2N+3$ linear equations.

Solving the $2N+3$ linear equations, we can determine the $2N+3$ unknown constants $\mathbf{d}_0$, $\mathbf{f}_1$, ..., $\mathbf{d}_N$, $\mathbf{f}_{N+1}$, $\mathbf{d}_{N+2}$, ..., $\mathbf{d}_{2N}$, $\mathbf{f}_{2N+1}$, and $\mathbf{d}_{2N+2}$. Using $\mathbf{d}_1 = \mathbf{c}_{1j}$, $\mathbf{d}_{N+1} = \mathbf{c}_{2j}$, and $\mathbf{d}_{2N+1} = \mathbf{c}_{3j}$ to replace $\mathbf{f}_1$, $\mathbf{f}_{N+1}$, and $\mathbf{f}_{2N+1}$, respectively, we obtain the generator consisting of the points $\mathbf{d}_i$ $(i = 1, 2, 3, \ldots,$ $2N+1)$.

Using the same process, we can obtain the generator at the other positions. From these obtained generators, a detailed 3D model can be created (an example is shown in Figure 6.5).

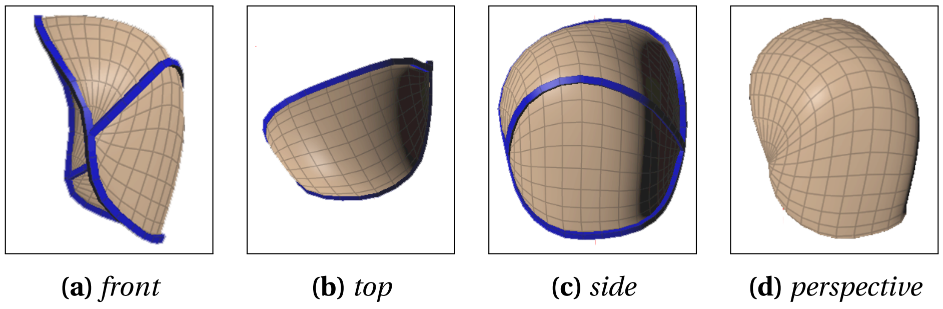


Figure 6.5: *The shape creation from one open and one closed silhouette contours in two different view planes.*

6.3 Two Open and One Closed Silhouette Contours in Three Different View Planes

The algorithms given in Subsections 6.1 and 6.2 enable users to create local 3D shapes from the sketches in two orthographic view planes. By adding one or more silhouette contours in the third orthographic view plane, users can further control the shape of 3D models. In this subsection, I'll describe how to construct a 3D detailed model from two open and one closed silhouette contours in three orthographic view planes.

As shown in Figure 6.6, the task of creating a local 3D shape passing through two open and one closed silhouette contours can be transformed into constructing four sweeping surfaces encircled by the curves $\mathbf{c}_1\mathbf{c}_6\mathbf{c}_5$, $\mathbf{c}_2\mathbf{c}_7\mathbf{c}c_6$, $\mathbf{c}_3\mathbf{c}_8\mathbf{c}_7$, and $\mathbf{c}_4\mathbf{c}_5\mathbf{c}_8$, respectively.

Since the construction algorithms for the four sweeping surfaces are the same, without the loss of generality, we take the sweeping surface encircled by the three curves $\mathbf{c}_1$, $\mathbf{c}_6$ and $\mathbf{c}_5$ as an example to demonstrate the construction algorithm.

The construction algorithm for the curve $\mathbf{c}_6$ is exactly the same as that of the curve $\mathbf{c}_5$. Here we take the reconstruction of the curve $\mathbf{c}_5$ as an example

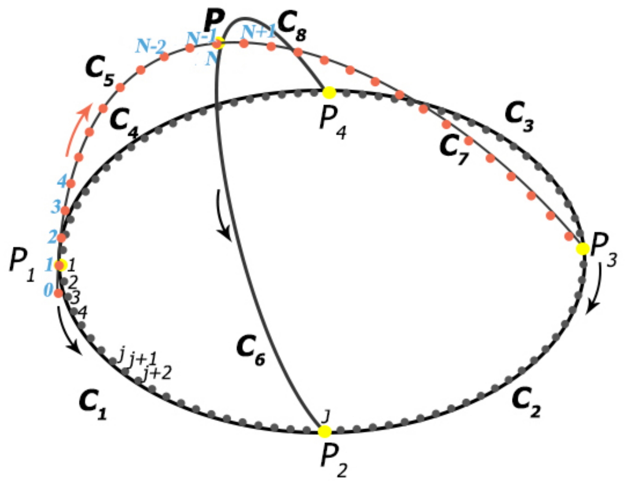


Figure 6.6: *Finite difference nodes in the algorithm for the case of two open and one closed silhouette contours.*

to demonstrate the algorithm.

Similar to the previous treatment, we uniformly divide the domain of the parametric variable u for the curve $\mathbf{c}_5(u)$ into $N-1$ equal intervals. The vector-valued first derivatives of the curves $\mathbf{c}_5(u)$ at the node 1 and N can be determined from the curve $\mathbf{c}_5(u)$ and indicated by $\mathbf{T}_{5,0}$ and $\mathbf{T}_{5,1}$.

When writing the finite difference equations for the inner nodes 2, the node 0 beyond the boundary node 1 will be involved. We can use the method given in Subsection 6.2 to determine the nodes 0 from the vector-valued first derivatives $\mathbf{T}_{5,0}$ through

$$\mathbf{d}_0 = \mathbf{d}_2 - 2\mathbf{T}_{5,0}\triangle u. \tag{6.1}$$

For each of the inner nodes 2, 3, ..., $N-1$, N, we can write a finite difference equation. Since the coordinate values for all the nodes on the curve $\mathbf{c_5}$ are known, we can calculate the sculpting force $\mathbf{f}_i$ from Equation (5.8). We denote these sculpting forces as $\mathbf{f}_{5,i} = \mathbf{f}_i (i = 2,3,\ldots,N-1,N)$. With the same method, we can obtain the sculpting forces $\mathbf{f}_{6,i} = \mathbf{f}_i (i = 2, 3, \ldots, N-1, N)$ acting on the curve $\mathbf{c}_6$.

The curves $\mathbf{c}_5$ and $\mathbf{c}_6$ can be regarded as generators. The generation of the sweeping surface defined by $\mathbf{c}_1$, $\mathbf{c}_6$ and $\mathbf{c}_5$ is to sweep the generator from $\mathbf{c}_5$ to $\mathbf{c}_6$ along $\mathbf{c}_1$ and the point $\mathbf{P}$. In order to determine the shape of the generator at different positions along $\mathbf{c}_1$, we uniformly divide $\mathbf{c}_1$ into J equal intervals and obtain the nodes $j = 1, 2, 3, \ldots, J-2, J-1, J$, where $j = 1$ and $j = J$ are the intersecting points between $\mathbf{c}_5$ and $\mathbf{c}_1$ and between $\mathbf{c}_6$ and $\mathbf{c}_1$. Then we determine the shape of the generator between the node $j (j = 2, 3, \ldots, J-2, J-1)$ and the point $\mathbf{P}$.

With the same treatment, we divide the domain of the parametric variable u for the generator between the node j and the point $\mathbf{P}$ into N equal intervals (the node j on $\mathbf{c}_1$ is the node 0 on the generator between the node j and the point $\mathbf{P}$). When sweeping $\mathbf{c}_5$ along $\mathbf{c}_1$ to $\mathbf{c}_6$, the sculpting force $\mathbf{f}_{5,i}$ acting at the node i of $\mathbf{c}_5$ is gradually changed to the sculpting force $\mathbf{f}_{6,i}$ acting at the node i of $\mathbf{c}_6$. Here we use a linear interpolation to describe the gradual change and obtain the sculpting force $\mathbf{f}_i$ below acting at the node i of the generator between the node j and the point $\mathbf{P}$:

$$\mathbf{f}_i = \mathbf{f}_{5,i} + \frac{L_j}{L}(\mathbf{f}_{6,i} - \mathbf{f}_{5,i}), \tag{6.2}$$

where L_j is the length from the point $\mathbf{P}_1$ to the node j and L is the length from the point $\mathbf{P}_1$ to the point $\mathbf{P}_2$.

When sweeping $\mathbf{c}_5$ along $\mathbf{c}_1$ to $\mathbf{c}_6$, the vector-valued first derivatives of $\mathbf{c}_5$ at the points $\mathbf{P}$ and $\mathbf{P}_1$ are also gradually changed to the vector-valued first derivatives of $\mathbf{c}_6$ at the points $\mathbf{P}$ and $\mathbf{P}_2$. The same linear interpolation is used to describe such a gradual change and determine the vector-valued first derivatives $\mathbf{T}_0$ and $\mathbf{T}_1$ of the generator at the node j and the point $\mathbf{P}$:

$$\begin{aligned}\mathbf{T}_0 &= \mathbf{T}_{5,0} + \frac{L_j}{L}(\mathbf{T}_{6,0} - \mathbf{T}_{5,0}),\\ \mathbf{T}_1 &= \mathbf{T}_{5,1} + \frac{L_j}{L}(\mathbf{T}_{6,1} - \mathbf{T}_{5,1}).\end{aligned} \tag{6.3}$$

Having known the sculpting forces at all the inner nodes 2, 3, ..., $N-1$, N, and the coordinate values at the boundary nodes 1 and $2N+1$ and the nodes 0 and $N+1$ beyond the boundary nodes by Equation (6.1), we can write the finite difference equations for all the inner nodes where the finite difference equations for the nodes 4, 5, ..., $N-3$, $N-2$ are the same as Equation (5.8) with the sculpting force being calculated by Equation (6.2), and the finite difference equations for the nodes 2, 3, $N-2$, and $N-1$ are given in **Appendix D**

Solving all the finite difference equations, we can obtain all the unknown constants $\mathbf{d}_2$, $\mathbf{d}_3$, ..., $\mathbf{d}_{2N-1}$, and $\mathbf{d}_{2N}$. They together with the two known points $\mathbf{d}_1$ and $\mathbf{P}$ are used to create the generator.

Using the same treatment, we can obtain all the curves of the generator at the positions $j = 1, 2, 3, \ldots, J-2, J-1, J$. They are used to create the sweeping surface. With the above method, we draw a closed lip contour and two open curves, as shown in Figure 6.7a and 6.7b. The two open curves divide the mouth into four regions. One surface is created for each of the four regions. Figure 6.7c depicts the mouth model consisting of the four surfaces.

The above method can also be extended to deal with the situations where the two open curves do not intersect. For such situations, the intersecting point of the two open curves becomes the upper closed curve as in-

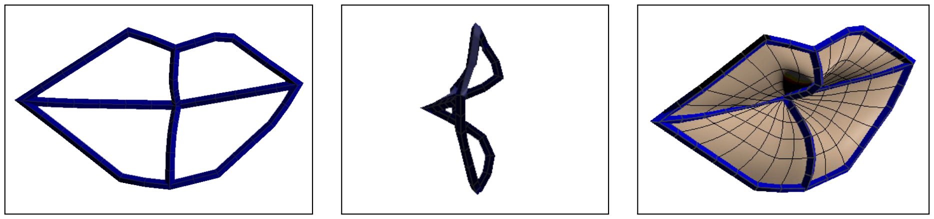

(a) *front* **(b)** *side* **(c)** *perspective*

Figure 6.7: *A mouth model creation from two open and one closed silhouette contours in three different view planes.*

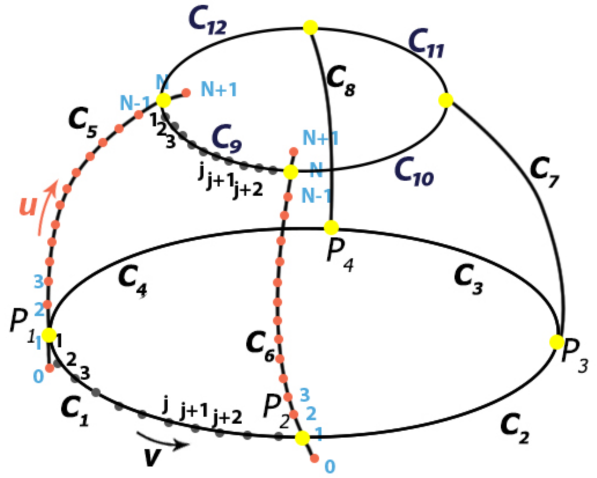


Figure 6.8: *Finite difference nodes for the algorithm for four open and two closed silhouette contours.*

dicated in Figure 6.8, and a sweeping surface is encircled by four curves. When writing the finite difference equations for the inner nodes $N-1$, the node $N+1$ beyond the boundary node N will be involved. We can use the method given in Subsection 6.2 to determine the nodes $N+1$ from the vector-valued first derivatives $\mathbf{T}_{5,1}$ through

$$\mathbf{d}_{N+1} = \mathbf{d}_{N-1} + 2\mathbf{T}_{5,1}\triangle u. \tag{6.4}$$

With this extension, we can create a 3D vest model and an eye socket model, as shown in Figure 6.9a and Figure 6.9b, respectively.

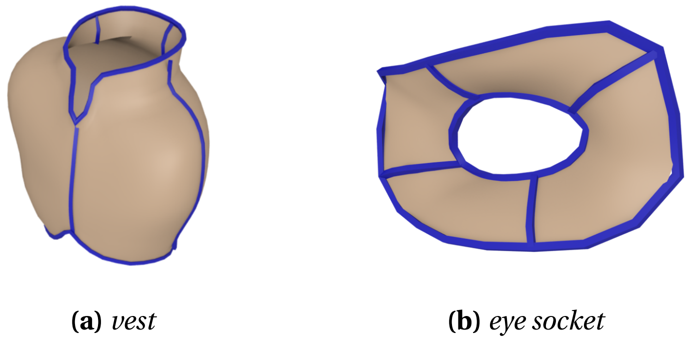

(a) *vest* **(b)** *eye socket*

Figure 6.9: *Vest and eye socket creation from four open and two closed silhouette contours.*

The extended algorithm is not only applicable to top and bottom closed curves, but also top and bottom open curves. Figure 6.10 gives such an example where the three curves in red are open as shown in Figure 6.10a. The generated 3D model is depicted in Figure 6.10b.

6.4 User Interface for the Detail Generator

Fig.6.11 shows the user-interface of the proposed sketch-based modelling system, where four split viewports are on Fig.6.11a with the front view dis-

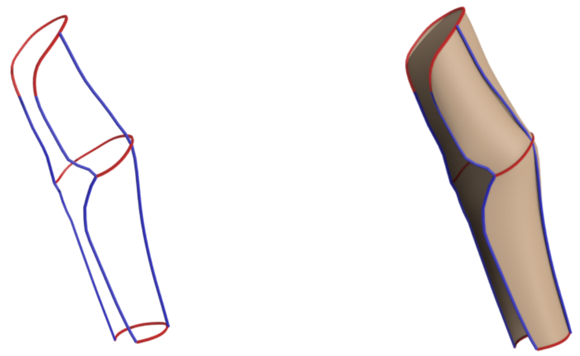
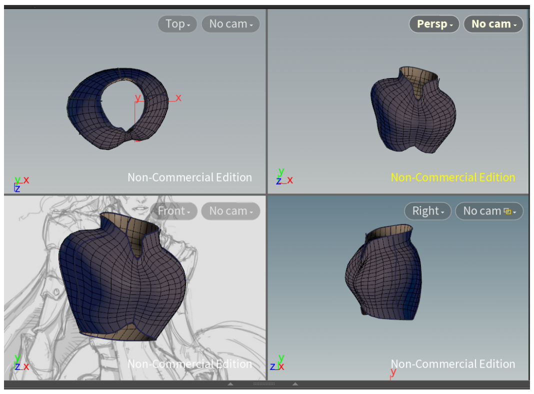

(a) *contours* **(b)** *mesh*

Figure 6.10: *Creation of a thin sheet on the left forearm.*

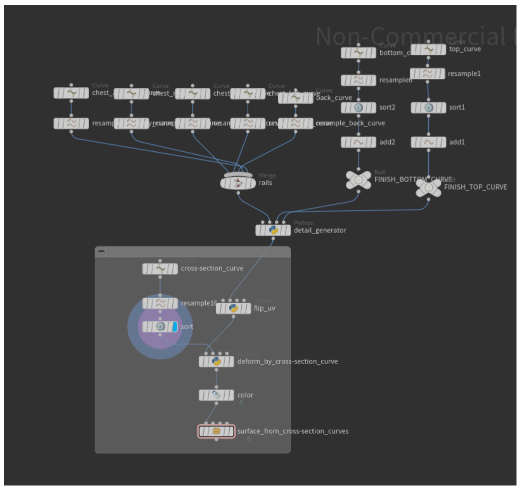
plays the reference image on the background, while the node-based pipeline is displayed on Fig.6.11b. As shown on Fig.6.11b, this system firstly let the user to draw sketched curves and then *resample* node samples the input sketched stroke to find the discrete points to feed the algorithm of detail generators. Then *sort* node will appoint the points with correct index. Later, the *detail generator* node is responsible for creating the surface, in this case, it's a vest from five rails, one top curve, and one bottom curve. If later, another cross-sectional curve is provided to guide the deforming step, *deform by cross-section curve* node will deform the whole surface based on the curve and the deforming method described in Chapter.5. Finally, *surface from cross-section curves* node will take the cross-sectional curves calculated from the previous steps and turn them into a surface with the method in Chapter.4.

6.5 Experiments and Comparisons

This section presents the experiments to validate that my approach can create more realistic shapes and use the creation of a 3D facial model to



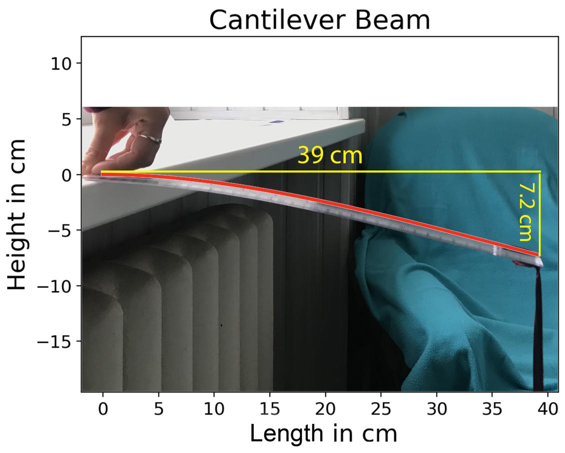
(a)



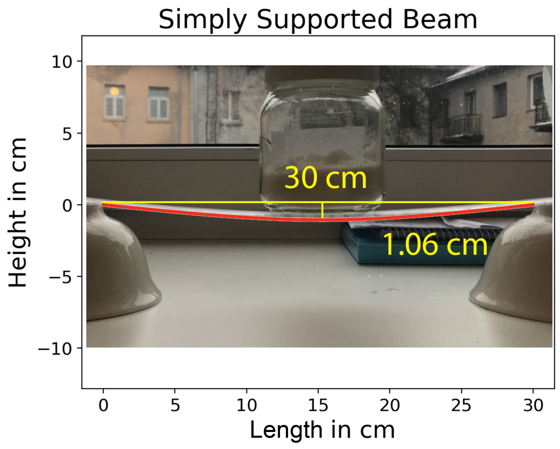
(b)

Figure 6.11: *The user-interface of the sketch-based modelling system*

demonstrate that my approach can create detailed 3D models more efficiently. The experiment is to deform a plastic ruler and compare its ground-truth deformations with those determined by my ODE-driven deformation approach.



(a)



(b)

Figure 6.12: *Deformation comparisons: (a) deform a plastic ruler by fixing one end of the ruler and putting a heavy weight on the other free end of the ruler, (b) deform a plastic ruler by placing a load at the midpoint of a plastic ruler, with the two ends of the ruler simply supported by two identical objects. Red curves are the simulated results and the gray rulers are the ground-truths.*

As shown in Figure 6.12a, the plastic ruler is fixed at one end and its other end is tied with a heavy weight, like a cantilever beam. Therefore, the ruler is bent towards the ground by $7.2cm$ at its free end with a force equal to the gravity of the heavy weight acting on it. The setup of my ODE-driven deformations is: the displacement differences $d_0 = 0$ and $d_{N+1} = 7.2cm$, the rotation of the fixed end $\frac{\partial d_0}{\partial u} = 0$, and the third derivative of displacement at the free end is proportional to the force applied to it $\frac{\partial^3 d_{N+1}}{\partial u^3} = \frac{f}{k_b}$. I use the bend modulus of PP plastics to approximate the bending stiffness $k_b = 0.5$, and set the stretching stiffness $k_s = 1.4$. The red curve in Figure 6.12a shows its simulation, which is very close the ground-truth (the ruler itself).

In another experiment shown in Figure6.12b, two identical objects simply supported a ruler at its two ends, and a load was put at the midpoint of the ruler. Since we did not fix the two ends of the ruler, no bending moment M results in zero curvature at the two ends $\frac{\partial^2 d_0}{\partial u^2} = \frac{M}{EI} = \frac{0}{EI} = 0$, $\frac{\partial^2 d_{2N-1}}{\partial u^2} = \frac{M}{EI} = \frac{0}{EI} = 0$. Together with the displacement differences at the two ends $d_0 = d_{2N-1} = 0$, and at the midpoint $d_{N-1} = 1.06cm$, the simulation by my ODE-driven deformation technique is shown as the red curve in Figure6.12b, which is also very close to the ground-truth (the gray curve).

By comparing the red curves and the real deformed rulers in (a) and (b) of Figure 6.12, the red curves are very close to the real deformed rulers. It indicates that ODE-driven deformations are able to depict real shape changes.

Human faces have the most details among various virtual characters. With the algorithms developed in this paper, a detailed human face can be easily and efficiently created.

Next we use 3D face creation as a specific application to demonstrate that my approach can efficiently add 3D details to the created 3D models. Taking the female warrior face shown in Figure 6.13a as an example, I compare 3D face generation between the traditional polygon modelling and my sketch-based, ODE-driven shape modelling. With the Maya-provided polygon modelling, it took an invited user about 40 hours to create a detailed facial model in Figure 6.13b. With my sketch-based, ODE-driven shape modelling approach, a curve network shown in Figure 6.13c which defines the female warrior face is first generated. Then, the curve network is decomposed into different groups of sketches and my ODE-driven deformations are used to automatically create surface patches from different groups of sketches. The created surface patches are smoothly connected to generate the detailed facial model shown in Figure 6.13d. The total time of obtaining the detailed facial model in Figure 6.13d including generating the facial curve network in Figure 6.13c was about 5 hours by the same user.

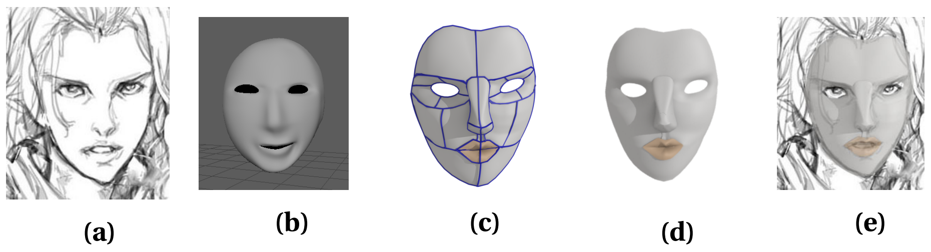


Figure 6.13: *Comparison of creating 3D facial details with the polygon modelling and sketch-based and ODE-driven shape generator. a) sketch of a female character, b) traditional polygon modelling result, c) input contours for face reconstruction, d) result of the proposed sketch-based modelling e) result of the proposed sketch-based modelling superimposed on the input sketch.*

My approach was developed in python on the Houdini FX Education Edition 16.5.323 package, and ran on a dual boot Linux PC with 23GB memory and 64 bits Intel(R) Xeon(R) CPU E5-1650 0 @ 3.20GHz CPU. The average time for deforming a primitive is 0.17 seconds and the average time for detail generators is 0.09 seconds, which ensures a smooth real-time modelling user experience.

6.6 Conclusion

In this chapter, a free-form patch generator is developed. It involves a force field into the ODE equation in Chapter 5 and utilises the boundary tangent as boundary constraints to solve the equations. The example shown in Fig. 6.14 to demonstrate the outcome of the proposed free-form patch generator.

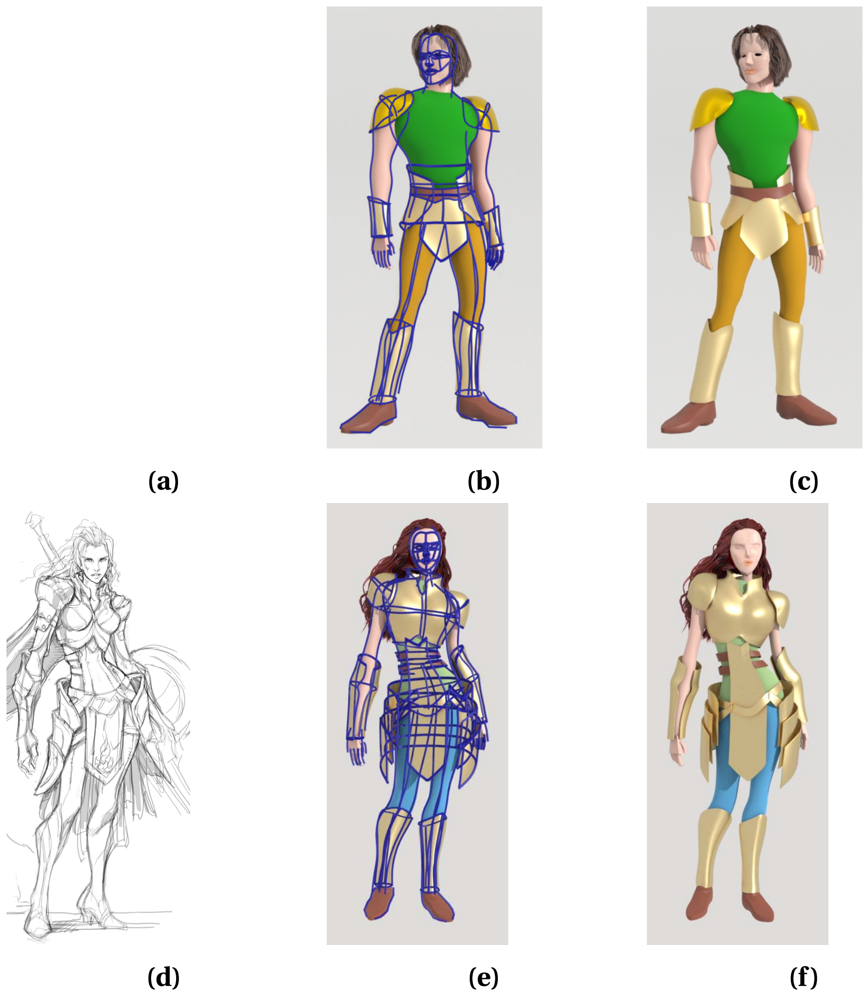


Figure 6.14: *a) Creation of character models with my proposed approach a) sketch of a male character. Image retrieved from: www.wolfiesden.com, b) 3d model of a male character by using my method with the sketched strokes that define the shape, c) 3d model of a male character by using my method. d) sketch of a female character. Sketch by ©EngKit Leong, e) 3d model of a female character by using my method with the sketched strokes that define the shape, f) 3d model of a female character by using my method.*

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Virtual characters are widely applied in various situations such as computer games, virtual reality and digital films. As pointed out by Magnenat-Thalmann & Thalmann [2005], in order to create realistic and believable virtual humans, three techniques should be developed, they are realistic, smooth and flexible motion modelling and realistic high-level behaviour modelling. Among them, realistic appearance modelling consisting of human modelling and deformation has attracted a lot of research attention. Many modeling technologies are proposed during the past two decades, and sketch-based modeling (SBM) in particular is popular with the modeling community by its user-friendly and intuitive interface. However, existing sketch-based modelling approaches are incapable in creating detailed and realistic 3D character models. In order to tackle this problem, I have added cross-section curves to define local shapes which cannot be defined by the sketches in front and side views, formulated analytical mathematical representations of cross-section curves and surfaces interpolating cross-section curves, developed an ODE-defined primitive generator from the sketches in front and side views and cross-section curves to generate more detailed and realistic base models, and invented an ODE-driven free-form patch generator to add local shapes to base models.

Firstly this research introduced generalized elliptic curves to approximate the cross section curves of human body in Chapter 3. This curve-fitting method achieves very good accuracy and the function is continuous everywhere. This generalized elliptic curves can greatly improve the accuracy of cross-sectional contours of human body compared with simply represent the cross-sectional contours with standard ellipses without trading off the computational cost and storage.

Secondly this thesis proposed a surface modeling technique customized for the cross-section contours represented by trigonometric series, so as to construct human body parts from the cross sectional curves, presented in Chapter 4. This simple and efficient method to build human models from the cross-section curves obtained in Chapter 3 is very light in data storage which overcomes the limitation of the work of Hyun et al. [2003, 2005] where sweeping standard ellipses is used to approximate the base shape and a displacement map is used to restore the original shape's details. My proposed surface modeling technique also supports LOD(level of detail) and is efficient in computational cost, as such, this technique is suitable for games where the characters far from viewer can be modeled with less details and accuracy.

In Chapter 5, a new ODE-based modeling method to deform the base mesh to exactly match the input complex sketch strokes is developed. Therefore, the result mesh of my method is more detailed and suitable for generating complex organic character models than existing sketch-based modeling method such as Igarashi et al. [1999] which generates simple toy-like shapes. This thesis also provide a novel pipeline consisting of the following steps. Step 1. To take the easiness and efficiency advantages of primitives in representing rough 3D models, this research choose superellipsoid primitives to quickly obtain an initial 3D mesh. Step 2. All the cross-section contours of this superellipsoid primitive are deformed such that the new surface's projected silhouette fits the input sketched stroke exactly. Step 3. Using trigonometric series algorithm proposed in Chapter 3 to fit the new cross-section contours, we got the parametric representation of the cross-sectional curves. Step 4. Obtain the final shape with the surface reconstruction method in Chapter 4. This method not only can deform the base mesh longitudinally, but also works in an attitudinal fashion: if user sketches a new cross-sectional curve, my method can deform the mesh to follow the sketched cross-sectional curve exactly and this mesh editing method is more accurate than Gingold et al. [2009] where the later only allows scale and rotate of the cross-sectional curve or the simple arbitrary cross-sectional curve.

Finally in Chapter 6, I invented a new free-form patch generating method to take a network of sketched curves as the input and then solve an ODE to get each of the vertex's position information within the span. This method gives the users enough control over the shape. In addition, to our best knowledge, it is the first free-form modeling tool that can create a dome-shaped surface without revolting a profile curve along an axis or a rail, furthermore, it supports multiple profile curves so that user can obtain the idea shape on one go, rather than using basic modeling tools to get an initial base mesh and editing handles on base mesh little by little.

7.2 Future Work

Though the curve-fitting method in Chapter 3 fulfills the demand of fitting the curve with generalized ellipse accurately and efficiently, when it comes to splitting the curve into composite generalized elliptic segments, depend on the SLSQP optimiser's performace, the computational time grows as the complexity of the segment grows. Hence, this technique can be further improved if a more suitable optimiser is employed.

As for the surface reconstruction method in Chapter 4, more auxiliary tools should be developed and more geometrical features of the method should be studied, if this surface representation wants to be added as a new standard mathematical model like NURBS modeling.

One possible future work may investigate the potential of the proposed the sketch-based and ODE-driven deformation technology from Chapter 5 in terms of changing the way of animating characters by integrating ODE-driven deformations and keyframes. With this new technique, animators don't have to manipulate the bones and repaint the weight or fine-tune the mesh little by little any more, all they need to do is generating shapes at keyframe poses. Then ODE-driven deformations are used to determine the shapes between two keyframe poses.

Another topic to be investigated in the future is cross-sectional curve-based local shape retrieval. This PhD thesis has proposed cross-sectional curve-based local shape creation. In order to generate more realistic local shape from cross-sectional curves, it is necessary to retrieve a local shape from user-drawn cross-sectional curves with machine learning technology, and add the retrieved local shape to base meshes to create more realistic 3D models.

Appendix A: Finite difference equations for primitive deformer

The finite difference equations at the nodes 0 and N can be formulated as below.

For node 0,

$$-f_0 + k_b d_{2N-2} + k_b d_2 - (4k_b + k_s \triangle u^2) d_{2N-1}$$
$$-(4k_b + k_s \triangle u^2) d_1 = -(6k_b + 2k_s \triangle u^2)(C'_{13} - C_{13})$$

For node N,

$$-f_N + k_b d_{N-2} + k_b d_{N+2} - (4k_b + k_s \triangle u^2) d_{N-1}$$
$$-(4k_b + k_s \triangle u^2) d_{N+1} = -(6k_b + 2k_s \triangle u^2)(C'_{23} - C_{23})$$

(A1)

The finite difference equations at the nodes 1, 2, $2N-2$ and $2N-1$ can be formulated as below.

For node 1,

$$(6k_b + 2k_s \triangle u^2) d_1 + k_b d_{2N-1} + k_b d_3$$
$$-(4k_b + k_s \triangle u^2) d_2 = (4k_b + k_s \triangle u^2)(C'_{13} - C_{13})$$

For node 2,

$$(6k_b + 2k_s \triangle u^2) d_2 + k_b d_4 - (4k_b + k_s \triangle u^2) d_1$$

$$-(4k_b + k_s\triangle u^2)d_3 = -k_b(C'_{13} - C_{13})$$

For node $2N-2$,

$$(6k_b + 2k_s\triangle u^2)d_{2N-2} + k_b d_{2N-4} - (4k_b + k_s\triangle u^2)d_{2N-3}$$
$$-(4k_b + k_s\triangle u^2)d_{2N-1} = -k_b(C'_{13} - C_{13})$$

For node $2N-1$,

$$(6k_b + 2k_s\triangle u^2)d_{2N-1} + k_b d_{2N-3} + k_b d_1$$
$$-(4k_b + k_s\triangle u^2)d_{2N-2} = (4k_b + k_s\triangle u^2)(C'_{13} - C_{13})$$

(A2)

The finite difference equations at the nodes $N-2$, $N-1$, $N+1$ and $N+2$ are

For node $N-2$,

$$(6k_b + 2k_s\triangle u^2)d_{N-2} + k_b d_{N-4} - (4k_b + k_s\triangle u^2)d_{N-3}$$
$$-(4k_b + k_s\triangle u^2)d_{N-1} = -k_b(C'_{23} - C_{23})$$

For node $N-1$,

$$(6k_b + 2k_s\triangle u^2)d_{N-1} + k_b d_{N-3} + k_b d_{N+1}$$
$$-(4k_b + k_s\triangle u^2)d_{N-2} = (4k_b + k_s\triangle u^2)(C'_{23} - C_{23})$$

For node $N+1$,

$$(6k_b+2k_s\triangle u^2)d_{N+1}+k_bd_{N+3}+k_bd_{N-1}$$
$$-(4k_b+k_s\triangle u^2)d_{N+2}=(4k_b+k_s\triangle u^2)(C'_{23}-C_{23})$$

For node $N+2$,

$$(6k_b+2k_s\triangle u^2)d_{N+2}+k_bd_{N+4}-(4k_b+k_s\triangle u^2)d_{N+3}$$
$$-(4k_b+k_s\triangle u^2)d_{N+1}=-k_b(C'_{23}-C_{23})$$

(A3)

The finite difference equation for the rest of the nodes i (i=3, 4, ..., $N-3$, $N+3$, $N+4$, ..., $2N-3$) is

For nodes i in 3, 4, ..., $N-3$, $N+3$, $N+4$, ..., $2N-3$,

$$(6k_b+2k_s\triangle u^2)d_i+k_bd_{i-2}+k_bd_{i+2}$$
$$-(4k_b+k_s\triangle u^2)d_{i-1}-(4k_b+k_s\triangle u^2)d_{i+1}=0$$

(A4)

Appendix B: Finite difference equations for two open silhouette contours in two different view planes

The finite difference equations at the nodes 0, $N/2$, N and $3N/2$ can be formulated as below. For node 0,

$$\begin{aligned}-f_0 + k_b d_{2N-2} + k_b d_2 - (4k_b + k_s \triangle u^2) d_{2N-1} \\ -(4k_b + k_s \triangle u^2) d_1 = -(6k_b + 2k_s \triangle u^2) C_{1j}\end{aligned}$$

For node $\frac{N}{2}$,

$$\begin{aligned}-f_{N/2} + k_b d_{N/2-2} + k_b d_{N/2+2} - (4k_b + k_s \triangle u^2) d_{N/2-1} \\ -(4k_b + k_s \triangle u^2) d_{N/2+1} = -(6k_b + 2k_s \triangle u^2) C_{2j}\end{aligned}$$

For node N,

$$\begin{aligned}-f_N + k_b d_{N-2} + k_b d_{N+2} - (4k_b + k_s \triangle u^2) d_{N-1} \\ -(4k_b + k_s \triangle u^2) d_{N+1} = -(6k_b + 2k_s \triangle u^2) C_{3j}\end{aligned}$$

For node $\frac{3N}{2}$,

$$\begin{aligned}-f_{3N/2} + k_b d_{3N/2-2} + k_b d_{3N/2+2} - (4k_b + k_s \triangle u^2) d_{3N/2-1} \\ -(4k_b + k_s \triangle u^2) d_{3N/2+1} = -(6k_b + 2k_s \triangle u^2) C_{4j}\end{aligned}$$

(B1)

The finite difference equations at the nodes 1, 2, $2N-1$, and $2N-2$ can be obtained by replacing $c'_{13}-c_{13}$ in Equation(A2) with c_{1j} and those at the nodes $N-2$, $N-1$, $N+1$ and $N+2$ can be achieved by replacing $c'_{23}-c_{23}$ in Equation(A3) with c_{3j}.

The finite difference equations at the nodes $N/2-2$, $N/2-1$, $N/2+1$, and $N/2+2$ can be written as

For node $\frac{N}{2}-2$,

$$(6k_b-2k_s\triangle u^2)d_{N/2-2}+k_bd_{N/2-4}-(4k_b+k_s\triangle u^2)d_{N/2-3}$$
$$-(4k_b+k_s\triangle u^2)d_{N/2-1}=-k_bC_{2j}$$

For node $\frac{N}{2}-1$,

$$(6k_b+2k_s\triangle u^2)d_{N/2-1}+k_bd_{N/2-3}+k_bd_{N/2+1}$$
$$-(4k_b+k_s\triangle u^2)d_{N/2-2}=(4k_b+k_s\triangle u^2)C_{2j}$$

For node $\frac{N}{2}+1$,

$$(6k_b+2k_s\triangle u^2)d_{N/2+2}+k_bd_{N/2}+k_bd_{N/2+4}$$
$$-(4k_b+k_s\triangle u^2)d_{N/2+3}=(4k_b+k_s\triangle u^2)C_{2j}$$

For node $\frac{N}{2}+2$,

$$(6k_b+2k_s\triangle u^2)d_{N/2+2}+k_bd_{N/2+4}-(4k_b+k_s\triangle u^2)d_{N/2+1}$$

$$-(4k_b + k_s\triangle u^2)d_{N/2+3} = -k_bC_{2j}$$

The finite difference equations at the nodes $3N/2-2$, $3N/2-1$, $3N/2+1$ and $3N/2+2$ are

For node $\frac{3N}{2}-2$

$$(6k_b + 2k_s\triangle u^2)d_{3N/2-2} + k_bd_{3N/2-4}$$
$$-(4k_b + k_s\triangle u^2)d_{3N/2-3} - (4k_b + k_s\triangle u^2)d_{3N/2-1} = -k_bC_{4j}$$

For node $\frac{3N}{2}-1$

$$(6k_b + 2k_s\triangle u^2)d_{3N/2-1} + k_bd_{3N/2-3} + k_bd_{3N/2+1}$$
$$-(4k_b + k_s\triangle u^2)d_(3N/2-2) = (4k_b + k_s\triangle u^2)C_{4j}$$

For node $\frac{3N}{2}+1$

$$(6k_b + 2k_s\triangle u^2)d_{3N/2+1} + k_bd_{3N/2-1} + k_bd_{3N/2+3}$$
$$-(4k_b + k_s\triangle u^2)d_{3N/2+2} = (4k_b + k_s\triangle u^2)C_{4j}$$

For node $\frac{3N}{2}+2$

$$(6k_b + 2k_s\triangle u^2)d_{3N/2+2} + k_bd_{3N/2+4}$$
$$-(4k_b + k_s\triangle u^2)d_{3N/2+1} - (4k_b + k_s\triangle u^2)d_{3N/2+3}$$
$$= -k_bC_{4j}$$

(B2)

For nodes in 3, 4, … , $N/2-3$, $N/2+3$, $N/2+4$, … , $N-4$, $N-3$, $N+3$, $N+4$, …, $3N/2-4$, $3N/2-3$, $3N/2+3$, $3N/2+4$, …, $2N-4$, $2N-3$, same as that in Equation(A4).

Appendix C: Finite difference equations for one open and one closed silhouette contours in two different view planes

The finite difference equations at the nodes $N-2$, $N-1$, $N+1$ and $N+2$ can be obtained by replacing $c'_{23}-c_{23}$ in Equation (A2) with c_{2j}.
The finite difference equations at the nodes 2 and 3 can be written as
For node 2,

$$\begin{aligned}(7k_b+2k_s\triangle u^2)d_2-(4k_b+k_s\triangle u^2)d_3+k_bd_4\\ =2k_bT_{1j}\triangle u+(4k_b+k_s\triangle u^2)C_{1j}\end{aligned}$$

For node 3,

$$\begin{aligned}(6k_b+2k_s\triangle u^2)d_3-(4k_b+k_s\triangle u^2)d_2+k_bd_5\\ -(4k_b+k_s\triangle u^2)d_4=-k_bC_{1j}\end{aligned}$$

(C1)

The finite difference equations at the nodes $N-1$, N, $N+2$ and $N+3$ can be written as

For node $N-1$,

$$(6k_b + 2k_s\triangle u^2)d_{N-1} + k_b d_{N-3} - (4k_b + k_s\triangle u^2)d_{N-2}$$
$$-(4k_b + k_s\triangle u^2)d_N = -k_b C_{2j}$$

For node N,

$$(6k_b + 2k_s\triangle u^2)d_N + k_b d_{N+2} - (4k_b + k_s\triangle u^2)d_{N-1}$$
$$= (4k_b + k_s\triangle u^2)C_{2j}$$

For node $N+2$,

$$(6k_b + 2k_s\triangle u^2)d_{N+2} + k_b d_N + k_b d_{N+4}$$
$$-(4k_b + k_s\triangle u^2)d_{N+3} = (4k_b + k_s\triangle u^2)C_{2j}$$

For node $N+3$,

$$(6k_b + 2k_s\triangle u^2)d_{N+3} + k_b d_{N+5} - (4k_b + k_s\triangle u^2)d_{N+2}$$
$$-(4k_b + k_s\triangle u^2)d_{N+4} = -k_b C_{2j}$$

(C2)

The finite difference equations at the nodes $2N-1$ and $2N$ can be written as

For node $2N-1$,

$$(6k_b + 2k_s \triangle u^2)d_{2N-1} + k_b d_{2N-3} - (4k_b + k_s \triangle u^2)d_{2N-2}$$
$$-(4k_b + k_s \triangle u^2)d_{2N} = -k_b C_{3j}$$

For node $2N$,

$$(7k_b + 2k_s \triangle u^2)d_{2N} + k_b d_{2N-2} - (4k_b + k_s \triangle u^2)d_{2N-1}$$
$$= (4k_b + k_s \triangle u^2)C_{3j} - 2k_b T_{5j} \triangle u$$

(C3)

For nodes in 4, 5, ..., $N-2$, $N+4$, $N+5$, ..., $2N-2$, the formula is the same as that in Equation(A4).

Appendix D: Finite difference equations for two open and one closed silhouette contours in three different view planes

Sculpting force at node 2 on curve C_5

$$f_{5,2} = \frac{1}{\triangle u^4}[(6k_b + 2k_s \triangle u^2)C_{5,2} + k_b(C_{5,2} - 2T_{5,0}\triangle u)$$
$$+k_b C_{5,4} - (4k_b + k_s \triangle u^2)C_{5,1} - (4k_b + k_s \triangle u^2)C_{5,3}]$$

Sculpting force at node 2 on curve C_6

$$f_{6,2} = \frac{1}{\triangle u^4}[(6k_b + 2k_s\triangle u^2)C_{6,2} + k_b(C_{6,2} - 2T_{6,0}\triangle u)$$
$$+k_bC_{6,4} - (4k_b + k_s\triangle u^2)C_{6,1} - (4k_b + k_s\triangle u^2)C_{6,3}]$$

The finite difference equations at the nodes 2 and 3 can be written as

For node 2,

$$(7k_b + 2k_s\triangle u^2)d_2 + k_bd_4 - (4k_b + k_s\triangle u^2)d_3$$
$$= \triangle u^4[f_{5,2} + \frac{L_j}{L}(f_{6,2} - f_{5,2})]$$
$$+2k_b\triangle u[T_{5,0} + \frac{L_j}{L}(T_{6,0} - T_{5,0})] + (4k_b + k_s\triangle u^2)P_1$$

For node 3,

$$(6k_b + 2k_s\triangle u^2)d_3 + k_bd_5 - (4k_b + k_s\triangle u^2)d_2$$
$$-(4k_b + k_s\triangle u^2)d_4 = \triangle u^4[f_{5,3} + \frac{L_j}{L}(f_{6,3} - f_{5,3})] - k_bP_1$$

(D1)

The finite difference equations at the nodes $N-1$ and N are

For node $N-1$,

$$(6k_b + 2k_s\triangle u^2)d_{N-1} + k_bd_{N-3} + k_bd_{N+1}$$
$$-(4k_b + k_s\triangle u^2)d_{N-2} =$$
$$\triangle u^4[f_{5,N-1} + \frac{L_j}{L}(f_{6,N-1} - f_{5,N-1})] + (4k_b + k_s\triangle u^2)P$$

For node N,

$$(6k_b - 2k_s\triangle u^2 + k_b)d_N + k_b d_{N-2} + k_b d_{N+2}$$
$$-(4k_b + 2k_s\triangle u^2)d_{2N-1} = \triangle u^4[f_{5,N} + \frac{L_j}{L}(f_{6,N} - f_{5,N})]$$
$$+(4k_b + k_s\triangle u^2)P$$

(D2)

For nodes in 4, 5, ..., $N-3$, $N-2$, the formula is as follow

$$(6k_b + 2k_s\triangle u^2)d_i + k_b d_{i-2} + k_b d_{i+2}$$
$$-(4k_b + k_s\triangle u^2)d_{i-1} - (4k_b + k_s\triangle u^2)d_{i+1}$$
$$= \frac{1}{\triangle u^4}[f_{5,i} + \frac{L_j}{L}(f_{6,i} - f_{5,i})]$$

Appendix E: Matrices for generalized ellipses

Each of the equation sets (3.8) and (3.9) has $2J+1$ equations, corresponding to $2J+1$ unknown parameters $a_0, a_1, \ldots, a_{2J}$ and $b_0, b_1, \ldots, b_{2J}$, respectively. Reformat them into two matrices of linear equations, we can get the following coefficient matrices and ordinate vectors for solution vector

$[a_0, a_1, \ldots, a_{2J}]^T$ and $[b_0, b_1, \ldots, b_{2J}]^T$, respectively.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 \\ \cos(1v_1) & \cos(1v_2) & \ldots & \cos(1v_{I-1}) & \cos(1v_I) \\ \sin(1v_1) & \sin(1v_2) & \ldots & \sin(1v_{I-1}) & \sin(1v_I) \\ \cos(2v_1) & \cos(2v_2) & \ldots & \cos(2v_{I-1}) & \cos(2v_I) \\ \sin(2v_1) & \sin(2v_2) & \ldots & \sin(2v_{I-1}) & \sin(2v_I) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \cos\left((J-1)v_1\right) & \cos\left((J-1)v_2\right) & \ldots & \cos\left((J-1)v_{I-1}\right) & \cos\left((J-1)v_I\right) \\ \sin\left((J-1)v_1\right) & \sin\left((J-1)v_2\right) & \ldots & \sin\left((J-1)v_{I-1}\right) & \sin\left((J-1)v_I\right) \\ \cos(Jv_1) & \cos(Jv_2) & \ldots & \cos(Jv_{I-1}) & \cos(Jv_I) \\ \sin(Jv_1) & \sin(Jv_2) & \ldots & \sin(Jv_{I-1}) & \sin(Jv_I) \end{bmatrix}$$

$$\mathbf{A} \times \mathbf{A}^T \times \begin{bmatrix} a_0 & a_1 & a_2 & \ldots & a_{2J-1} & a_{2J} \end{bmatrix}^T = \begin{bmatrix} \sum_{i=1}^{I}(x_i - x_c) \\ \sum_{i=1}^{I}(x_i - x_c)\cos(1v_i) \\ \sum_{i=1}^{I}(x_i - x_c)\sin(1v_i) \\ \sum_{i=2J-1}^{I}(x_i - x_c)\cos(2v_i) \\ \sum_{i=2J-1}^{I}(x_i - x_c)\sin(2v_i) \\ \ldots \\ \sum_{i=2J}^{I}(x_i - x_c)\cos(Jv_i) \\ \sum_{i=2J}^{I}(x_i - x_c)\sin(Jv_i) \end{bmatrix} \tag{7.1}$$

and

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 \\ \sin(v_1) & \sin(v_2) & \dots & \sin(v_{I-1}) & \sin(v_I) \\ \cos(v_1) & \cos(v_2) & \dots & \cos(v_{I-1}) & \cos(v_I) \\ \sin(2v_1) & \sin(2v_2) & \dots & \sin(2v_{I-1}) & \sin(2v_I) \\ \cos(2v_1) & \cos(2v_2) & \dots & \cos(2v_{I-1}) & \cos(2v_I) \\ \dots & \dots & \dots & \dots & \dots \\ \sin(Jv_1) & \sin(Jv_2) & \dots & \sin(Jv_{I-1}) & \sin(Jv_I) \\ \cos(Jv_1) & \cos(Jv_2) & \dots & \cos(Jv_{I-1}) & \cos(Jv_I), \end{bmatrix}$$

$$\mathbf{B} \times \mathbf{B}^T \times \begin{bmatrix} b_0 & b_1 & b_2 & \dots & b_{2J-1} & b_{2J} \end{bmatrix}^T = \begin{bmatrix} \sum_{i=1}^{I}(y_i - y_c) \\ \sum_{i=1}^{I}(y_i - y_c)\sin(v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\cos(v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\sin(2v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\cos(2v_i) \\ \dots \\ \sum_{i=2J}^{I}(y_i - y_c)\sin(Jv_i) \\ \sum_{i=2J}^{I}(y_i - y_c)\cos(Jv_i) \end{bmatrix} \tag{7.2}$$

Solving equations (7.1) and (7.2), respectively, we determine all unknown constants and obtain the mathematical representation of the generalized ellipse.

Appendix F: Matrices for composite generalized elliptic segment that links two other segments

Eqs. (7.3) and Eqs. (7.4).

$$\mathbf{M}_a = \begin{bmatrix} 1-\cos 2v_1 & 1-\cos 2v_2 & \dots & 1-\cos 2v_I \\ \cos 3v_1 - \frac{1}{2}D_1(3,v_1) & \cos 3v_2 - \frac{1}{2}D_1(3,v_2) & \dots & \cos 3v_I - \frac{1}{2}D_1(3,v_I) \\ \sin 3v_1 - \frac{3}{2}D_2(3,v_1) & \sin 3v_2 - \frac{3}{2}D_2(3,v_2) & \dots & \sin 3v_I - \frac{3}{2}D_2(3,v_I) \\ \cos 4v_1 - \frac{1}{2}D_1(4,v_1) & \cos 4v_2 - \frac{1}{2}D_1(4,v_2) & \dots & \cos 4v_I - \frac{1}{2}D_1(4,v_I) \\ \sin 4v_1 - \frac{4}{2}D_2(4,v_1) & \sin 4v_2 - \frac{4}{2}D_2(4,v_2) & \dots & \sin 4v_I - \frac{4}{2}D_2(4,v_I) \\ \cos 5v_1 - \frac{1}{2}D_1(5,v_1) & \cos 5v_2 - \frac{1}{2}D_1(5,v_2) & \dots & \cos 5v_I - \frac{1}{2}D_1(5,v_I) \\ \sin 5v_1 - \frac{5}{2}D_2(5,v_1) & \sin 5v_2 - \frac{5}{2}D_2(5,v_2) & \dots & \sin 5v_I - \frac{5}{2}D_2(5,v_I) \\ \dots & \dots & \dots & \dots \\ \cos Jv_1 - \frac{1}{2}D_1(J,v_1) & \cos Jv_2 - \frac{1}{2}D_1(J,v_2) & \dots & \cos Jv_I - \frac{1}{2}D_1(J,v_I) \\ \sin Jv_1 - \frac{J}{2}D_2(J,v_1) & \sin Jv_2 - \frac{J}{2}D_2(J,v_2) & \dots & \sin Jv_I - \frac{J}{2}D_2(J,v_I), \end{bmatrix}$$

$$\mathbf{M}_a \times \mathbf{M}_a^T \times \begin{bmatrix} \bar{\bar{a}}_0 \\ \bar{\bar{a}}_5 \\ \bar{\bar{a}}_6 \\ \bar{\bar{a}}_7 \\ \dots \\ \bar{\bar{a}}_{2J-1} \\ \bar{\bar{a}}_{2J} \end{bmatrix} = \mathbf{M}_a \times \begin{bmatrix} C_a(1) \\ C_a(2) \\ C_a(3) \\ C_a(4) \\ \dots \\ C_a(I-1) \\ C_a(I) \end{bmatrix} \tag{7.3}$$

and

$$\mathbf{M}_b = \begin{bmatrix} 1-\cos 2v_1 & 1-\cos 2v_2 & \dots & 1-\cos 2v_I \\ \sin 3v_1 - \frac{3}{2}D_2(3,v_1) & \sin 3v_2 - \frac{3}{2}D_2(3,v_2) & \dots & \sin 3v_I - \frac{3}{2}D_1(3,v_I) \\ \cos 3v_1 - \frac{1}{2}D_1(3,v_1) & \cos 3v_2 - \frac{1}{2}D_1(3,v_2) & \dots & \cos 3v_I - \frac{1}{2}D_1(3,v_I) \\ \sin 4v_1 - \frac{4}{2}D_2(4,v_1) & \sin 4v_2 - \frac{4}{2}D_2(4,v_2) & \dots & \sin 4v_I - \frac{4}{2}D_2(4,v_I) \\ \cos 4v_1 - \frac{1}{2}D_1(4,v_1) & \cos 4v_2 - \frac{1}{2}D_1(4,v_2) & \dots & \cos 4v_I - \frac{1}{2}D_1(4,v_I) \\ \sin 5v_1 - \frac{5}{2}D_2(5,v_1) & \sin 5v_2 - \frac{5}{2}D_2(5,v_2) & \dots & \sin 5v_I - \frac{5}{2}D_2(5,v_I) \\ \cos 5v_1 - \frac{1}{2}D_1(5,v_1) & \cos 5v_2 - \frac{1}{2}D_1(5,v_2) & \dots & \cos 5v_I - \frac{1}{2}D_1(5,v_I) \\ \dots & \dots & \dots & \dots \\ \sin Jv_1 - \frac{J}{2}D_2(J,v_1) & \sin Jv_2 - \frac{J}{2}D_2(J,v_2) & \dots & \sin Jv_I - \frac{J}{2}D_2(J,v_I) \\ \cos Jv_1 - \frac{1}{2}D_1(J,v_1) & \cos Jv_2 - \frac{1}{2}D_1(J,v_2) & \dots & \cos Jv_I - \frac{1}{2}D_1(J,v_I) \end{bmatrix}$$

$$\mathbf{M}_b \times \mathbf{M}_b^T \times \begin{bmatrix} \bar{\bar{b}}_0 \\ \bar{\bar{b}}_5 \\ \bar{\bar{b}}_6 \\ \bar{\bar{b}}_7 \\ \dots \\ \bar{\bar{b}}_{2J-1} \\ \bar{\bar{b}}_{2J} \end{bmatrix} = \mathbf{M}_b \times \begin{bmatrix} C_b(1) \\ C_b(2) \\ C_b(3) \\ C_b(4) \\ \dots \\ C_b(I-1) \\ C_b(I) \end{bmatrix} \tag{7.4}$$

Bibliography

D. Ali-Hamadi, et al. (2013). 'Anatomy transfer'. *ACM Transactions on Graphics (TOG)* **32**(6):188.

R. Arora, et al. (2017). 'Experimental Evaluation of Sketching on Surfaces in VR.'. In *CHI*, vol. 17, pp. 5643–5654.

A. Aubel & D. Thalmann (2001). 'Interactive modeling of the human musculature'. In *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No. 01TH8596)*, pp. 167–255. IEEE.

I. Autodesk (2015). 'Character Generator'. `https://charactergenerator.autodesk.com/`.

S.-H. Bae, et al. (2008). 'ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models'. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pp. 151–160. ACM.

P. Beylot, et al. (1996). '3D interactive topological modeling using visible human dataset'. In *Computer Graphics Forum*, vol. 15, pp. 33–44. Wiley Online Library.

S. Bian, et al. (2019). 'Efficient and realistic character animation through analytical physics-based skin deformation'. *Graphical Models* **104**:101035.

S. Bian, et al. (2018). 'Automatic generation of dynamic skin deformation for animated characters'. *Symmetry* **10**(4):89.

J.-D. Boissonnat & P. Memari (2007). 'Shape reconstruction from unorganized cross-sections'. In *Symposium on geometry processing*, pp. 89–98.

M. Botsch & L. Kobbelt (2004). 'An intuitive framework for real-time freeform modeling'. In *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 630–634. ACM.

M. Botsch & O. Sorkine (2008). 'On linear variational surface deformation methods'. *IEEE Transactions on Visualization and Computer Graphics* **14**(1):213–230.

P. Buchanan, et al. (2013). 'Automatic single-view character model reconstruction'. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, pp. 5–14. ACM.

G. Celniker & D. Gossard (1991). 'Deformable curve and surface finite-elements for free-form shape design'. *ACM SIGGRAPH computer graphics* **25**(4):257–266.

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

E. Chaudhry, et al. (2013). 'Shape modeling for animated characters using ordinary differential equations'. *Computers & Graphics* **37**(6):638–644.

T. Chen, et al. (2013). '3-sweep: Extracting editable objects from a single photo'. *ACM Transactions on Graphics (TOG)* **32**(6):195.

F. Cole, et al. (2008). 'Where do people draw lines?'. In *ACM Transactions on Graphics (TOG)*, vol. 27, p. 88. ACM.

S. A. Coons (1967). 'Surfaces for computer-aided design of space forms'. Tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

C. De Paoli & K. Singh (2015). 'SecondSkin: sketch-based construction of layered 3D models'. *ACM Transactions on Graphics (TOG)* **34**(4):126.

D. DeCarlo, et al. (2003). 'Suggestive contours for conveying shape'. *ACM Transactions on Graphics (TOG)* **22**(3):848–855.

M. P. Do Carmo, et al. (2017). 'Differential Geometry'. In *Mathematical Models*, pp. 155–180. Springer.

D. H. Douglas & T. K. Peucker (1973). 'Algorithms for the reduction of the number of points required to represent a digitized line or its caricature'. *Cartographica: the international journal for geographic information and geovisualization* **10**(2):112–122.

A. Dürer (1534). *Four Books on Human Proportion.* Hieronymus Andreae.

F. Ebrahimi (2011). *Advances in vibration analysis research.* BoD–Books on Demand.

Facebook (2019). 'Quill'. `https://quill.fb.com/`.

T. Fleisch, et al. (2004). 'Constraint Stroke-Based Oversketching for 3D Curves.'. In *SBM*, pp. 161–165.

S. Gibson, et al. (1998). 'Volumetric object modeling for surgical simulation'. *Medical Image Analysis* **2**(2):121–132.

Y. Gingold, et al. (2009). ‘Structured annotations for 2D-to-3D modeling’. In *ACM Transactions on Graphics (TOG)*, vol. 28, p. 148. ACM.

Google Inc (2017). ‘Tilt Brush’. `https://www.tiltbrush.com/`.

Gravity Sketch (2017). ‘Gravity Sketch’. `https://www.gravitysketch.com/`.

D. D. Hoffman & M. Singh (1997). ‘Salience of visual parts’. *Cognition* **63**(1):29–78.

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

D.-E. Hyun, et al. (2003). ‘Modeling and deformation of arms and legs based on ellipsoidal sweeping’. In *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, pp. 204–212. IEEE.

T. Igarashi, et al. (1999). ‘Teddy: a sketching interface for 3D freeform design’. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 409–416. ACM Press/Addison-Wesley Publishing Co.

W. Jane & V. Allen (1997). ‘Anatomically based modelling’. In *Proceedings of the 1997 Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 97), ACM Press*, pp. 173–180.

L. B. Kara & K. Shimada (2007). ‘Sketch-based 3d-shape creation for industrial styling design’. *IEEE Computer Graphics and Applications* **27**(1):60–71.

O. A. Karpenko & J. F. Hughes (2006). ‘SmoothSketch: 3D free-form shapes from complex sketches'. In *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 589–598. ACM.

I. K. Kazmi, et al. (2015). 'Efficient sketch-based creation of detailed character models through data-driven mesh deformations'. *Computer Animation and Virtual Worlds* **26**(3-4):469–481.

I. K. Kazmi, et al. (2014). 'A survey of sketch based modeling systems'. In *Computer Graphics, Imaging and Visualization (CGIV), 2014 11th International Conference on*, pp. 27–36. IEEE.

V. Kraevoy, et al. (2009). 'Modeling from contour drawings'. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based interfaces and Modeling*, pp. 37–44. ACM.

D. H. Kraft (1988). 'A software package for sequential quadratic programming'.

V. Krs, et al. (2017). 'Skippy: single view 3D curve interactive modeling'. *ACM Transactions on Graphics (TOG)* **36**(4):128.

C. Li, et al. (2017). 'BendSketch: modeling freeform surfaces through 2D sketching'. *ACM Transactions on Graphics (TOG)* **36**(4):125.

H. Liang, et al. (2015). 'Advanced ordinary differential equation based head modelling for Chinese marionette art preservation'. *Computer Animation and Virtual Worlds* **26**(3-4):207–216.

N. Magnenat-Thalmann & D. Thalmann (2005). 'Virtual humans: thirty years of research, what next?'. *The Visual Computer* **21**(12):997–1015.

Mozilla (2019). 'Mozilla A-Painter'. `https://aframe.io/a-painter/`.

A. Nealen, et al. (2007a). ‘FiberMesh: designing freeform surfaces with 3D curves’. In *ACM transactions on graphics (TOG)*, vol. 26, p. 41. ACM.

A. Nealen, et al. (2005). ‘A sketch-based interface for detail-preserving mesh editing’. In *ACM SIGGRAPH 2005 Papers*, pp. 1142–1147.

A. Nealen, et al. (2007b). ‘A sketch-based interface for detail-preserving mesh editing’. In *ACM SIGGRAPH 2007 courses*, p. 42. ACM.

L. P. Nedel & D. Thalmann (1998). ‘Modeling and deformation of the human body using an anatomically-based approach’. In *Proceedings Computer Animation'98 (Cat. No. 98EX169)*, pp. 34–40. IEEE.

Oculus (2019). ‘Oculus Medium’. `https://www.oculus.com/medium/?locale=en_GB`.

L. Olsen, et al. (2009). ‘Sketch-based modeling: A survey’. *Computers & Graphics* **33**(1):85–103.

R. Parent (2012). *Computer animation: algorithms and techniques*. Newnes.

L. Piegl & W. Tiller (2012). *The NURBS book*. Springer Science & Business Media.

L. Porcher Nedel & D. Thalmann (2000). ‘Anatomic modeling of deformable human bodies’. *The Visual Computer* **16**(6):306–321.

S. Saito, et al. (2015). ‘Computational bodybuilding: Anatomically-based modeling of human bodies’. *ACM Transactions on Graphics (TOG)* **34**(4):1–12.

F. Scheepers, et al. (1997). ‘Anatomy-based modeling of the human musculature’. In *Proceedings of the 24th annual conference on Computer graph-*

ics and interactive techniques, pp. 163–172. ACM Press/Addison-Wesley Publishing Co.

A. Shtof, et al. (2013). ‘Geosemantic snapping for sketch-based modeling’. In *Computer graphics forum*, vol. 32, pp. 245–253. Wiley Online Library.

K. Takayama, et al. (2013). ‘Sketch-based generation and editing of quad meshes’. *ACM Transactions on Graphics (TOG)* **32**(4):97.

D. Terzopoulos, et al. (1987). ‘Elastically deformable models’. In *ACM Siggraph Computer Graphics*, vol. 21, pp. 205–214. ACM.

Y. Tokuyama (2000). ‘Skinning-surface generation based on spine-curve control’. *The Visual Computer* **16**(2):134–140.

Trimble Inc. (2019). ‘SketchUp’. `https://www.sketchup.com/`.

W. Welch & A. Witkin (1992). ‘Variational surface modeling’. In *ACM SIGGRAPH computer graphics*, vol. 26, pp. 157–166. ACM.

M. Xu, et al. (2016). ‘Interactive mechanism modeling from multi-view images’. *ACM Transactions on Graphics (TOG)* **35**(6):236.

R. Yang & B. C. Wünsche (2010). ‘Life-sketch: a framework for sketch-based modelling and animation of 3D objects’. In *Proceedings of the Eleventh Australasian Conference on User Interface-Volume 106*, pp. 61–70.

X. Yang & J. J. Zhang (2006). ‘Automatic muscle generation for character skin deformation’. *Computer Animation and Virtual Worlds* **17**(3-4):293–303.

L. You, et al. (2004). ‘PDE blending surfaces with C2 continuity’. *Computers & Graphics* **28**(6):895–906.

L. You, et al. (2014). ‘Blending using ODE swept surfaces with shape control and C^1 continuity’. *The Visual Computer* **30**(6-8):625–636.

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. (2010). ‘Shape manipulation using physically based wire deformations’. *Computer Animation and Virtual Worlds* **21**(3-4):297–309.

L. You, et al. (2008). ‘Dynamic skin deformation with characteristic curves’. *Computer Animation and Virtual Worlds* **19**(3-4):433–444.

M. Zou, et al. (2015). ‘Topology-constrained surface reconstruction from cross-sections’. *ACM Transactions on Graphics (TOG)* **34**(4):128.