Efficient and detailed character modelling by integrating sketches, ODE-driven deformations, geometric primitives, and composite generalized elliptic curves

OUWEN LI

Thesis of Doctor of Philosophy

Bournemouth University

Copyright statement

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognize that its copyright rests with its author and due acknowledgment must always be made of any material contained in, or derived from, this thesis.

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.

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.

First, a new sketch-guided and ODE-driven character modelling technique is proposed. It consists of two main components: primitive deformer and detail generator. With such a technique, we first draw 2D silhouette contours of a character model. Then, we select proper primitives and align them with the corresponding silhouette contours. After that, we develop a sketch-guided and ODE-driven primitive deformer. It uses ODE-based deformations to deform the primitives to exactly match the generated 2D silhouette contours in one view plane and obtain a base mesh of a character model consisting of deformed primitives. In order to add various 3D details, we 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 our 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 tem-

plate models and use the retrieved 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 sketch-based system, which is one of the contributions of our project.

Contents

Abstract iii

Table of contents v

1 Introduction **1**

1.1 Background 1

1.2 Main challenges 2

1.3 Research Objectives 3

1.4 Contribution 3

2 Literature Review **5**

2.1 Human Character Modeling 5

2.1.1 Shell surface modelling vs Volumetric modelling . . . 5

2.1.2 Anatomy-based Human Character Modeling 6

2.2 Sketch-Based Modelling Systems Overview 9

2.2.1 Theoretical Foundation As for Why Sketch Is a Reliable Source to Deduce 3D Shapes 9

2.2.2 Direct shape generation 10

2.2.3 template-based mesh creation 13

2.2.4 Layered Modeling 13

2.3 Sketch-based Deformation 15

3 Curve Fitting for Cross-Section **18**

3.1 Generalized ellipses 18

3.2 Bridging multiple curve segments 28

4 Primitive Generator **38**

4.1 Pipeline Overview 38

4.2 User Interface of Primitive Deformer 39

4.3 Thin Shell Deform Energy 40
4.4 Finite Difference Method for Geometric Processing 42
4.5 Algorithm for Primitive Deformer 44
4.6 Conclusion 47

5 Free-form Patch Generator 50
5.1 Two Open Silhouette Contours in Two Different View Planes 50
5.2 One Open and One Closed Silhouette Contours in Two Different View Planes 52
5.3 Two Open and One Closed Silhouette Contours in Three Different View Planes 56
5.4 Experiments and Comparisons 60
5.5 Conclusion 63

6 Cross-Section Defined Surface Modeling 65
6.1 Introduction 65
6.2 3D models defined with generalized ellipses 66
6.3 Surfaces defined with composite generalized elliptic segments 69
6.4 Assembly of human parts by surface blending 73
6.5 Conclusions 75

7 Conclusion and Future Work 78

Appendices 80

References 88

List of Figures

3.1 Comparison between real human cross sections and approximated ellipses 19

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: subfigure a,b,c shows Cross section curve of human leg with $J = 2, 3, 4$, respectively; subfigure d,e,f shows Cross section curve of human torso with $J = 3, 7, 10$, respectively 25

3.3 Program flowchart of finding a proper J 27

3.4 head,neck and arm cross-section curves in red, and their correspondent generalized ellipses produced by Eq.(3.3) with optimised J in green 28

3.5 open curves in red, and their correspondent generalised curves produced by Eq.(3.3) with optimised J in green 28

3.6 two segments of a hip cross-section curve 29

3.7 composite generalized elliptic segments in green and purple; one-piece generalized ellipse in green; original belly, epigastrium, chest and hip contour are in red 37

4.1 pipeline overview. 39

4.2 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 40

4.3 Primitive deformer: a) female warrior torso represented with a cylinder and its 2D silhouette contour, b) deformed shape of the cylinder 41
4.4 Typical node i for the finite difference approximations of derivatives 43
4.5 Finite difference nodes for local shape manipulation from sketches in different view planes. 45
4.6 additional applications of ODE-driven Sketch-based deformations 46
4.7 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. 48
4.8 Examples of the primitive deformation method 49

5.1 The finite difference nodes used in the algorithm in the case of two open silhouette contours in two different view planes. 51
5.2 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) the cross sections of the male head, d) lofting the cross-sections into the male head mesh. 52
5.3 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 our method in front view, d) leg mesh in side view, e) contours of a neck, f) cross sections of a neck generated by our method, g) final neck mesh. 53
5.4 Finite difference nodes for one open and one closed silhouette contours. 54
5.5 The shape creation from one open and one closed silhouette contours in two different view planes. 56
5.6 Finite difference nodes in the algorithm for the case of two open and one closed silhouette contours. 57
5.7 A mouth model creation from two open and one closed silhouette contours in three different view planes. 59

5.8 Finite difference nodes for the algorithm for four open and two closed silhouette contours. 59
5.9 vest and eye socket creation from four open and two closed silhouette contours. 60
5.10 Creation of a thin sheet on the left forearm. 60
5.11 Deformation comparisons: (a) deform a plastic ruler by fix one end of the ruler and put a heavy weight on the other free end of the ruler, (b) deform a plastic ruler by place 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 underneath gray curves are the ground-truths. 61
5.12 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 our sketch-based modelling. 63
5.13 a) sketch of a male character.*Image retrieved from: www.wolfiesden.com*, b) 3d model by using our method. 64

6.1 Human torso, legs and arm created from generalized ellipses 69
6.2 Connection of different human parts 75
6.3 Human model assembled from different human parts 76

List of Tables

2.1 Selected Terms for anatomyParent [2012] 9

Thesis Outline

This thesis is structured by eight chapters.

- **Chapter 1 - Introduction:** In this chapter, a brief outlook 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, surface blending, modeling from geometric primitives and shape construction by cross-section curves that are related with this research.
- **Chapter 3 - Primitive Generator:** In this chapter, an ODE-driven sketch-based modeling method for generating a primitive and deforming it afterward is presented. This deformation approach 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.
- **Chapter 4 - 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, more details such as accessories and clothes can be added on human character models.
- **Chapter 5 - Composite Cross-Section Contour by generalized ellipse segments:** In this chapter presented a method to approximate a cross-section curve as a generalized ellipse or a group of end-to-end generalized ellipse segments with a parametric formula.
- **Chapter 6 - Surface Reconstruction by Cross-section Contours** 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-section contours created by the method introduced in Chapter 5.
- **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 Background

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 and Singh [Hoffman & Singh 1997], for human vision systems, silhouettes alone are efficient for recognising everyday objects, by indexing their shapes in human memory. Also, we noticed that the most natural way for both novice modellers and veteran modellers 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 Gravity Sketch [2017], Tilt Brush Google Inc [2017], Quill Facebook [2019], Medium Oculus [2019] and Mozilla A-Painter Mozilla [2019] also show artists have no problems

in sketching in 3D canvas.

1.2 Main challenges

Surface reconstruction from cross-section contours

If we merely rely on sketches, there isn't much about the 3D depths information, so in this research we use cross-sections contours extracted from template models to help in defining the shapes.

- **Find the right cross-section contours from template models** As the blooming of high-quality 3d models for human character online, we can get the cross section contours extracted from those template models
- **Surface reconstruction and editing from the cross-section contours** Some generated piece-wise linear surface from cross section contours by marching tetrahedra, an iso-surfacing algorithm on tetrahedron surface.[Zou et al. 2015] We adopt the idea of dividing the mesh into pice-wise linear surface but come up with a Finite Difference Method of solving a partial derivative equation which minimises the elastic energy of the mesh deformation.
- **Surface reconstruction and editing from the input sketches** In film/animation industry quadrilateral meshes are heavily used whereas there are very few state of art mesh editing technologies on quadrilateral meshes which meet the computational efficiency requirement and the physical as well as aesthetic fidelity requirements.

sketch-based modeling system

The user interface of sketch-based modeling system should be intuitive enough to use, and provides the enough degree of freedom to meet users' editing demands.

1.3 Research Objectives

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

- **Literature Review**: Take a look at precedent researches on sketch-based modeling system, 3D shape retrieval technology and shape editing technology.
- **Reconstruct the 3D shape by interpolating cross-section contours that are extracted from 3D template models**: Extract cross-section contours at the standard benchmark feature positions and snap the cross-section contours onto the target mesh. Afterward, reconstruct 3D shapes based on the cross-section contours.
- **shape deforming technology**: Find an efficient physical-based mesh editing algorithm which works well on quadrilateral meshes. Deform the base mesh from the input sketch silhouette and editing the base mesh by the cross-section contours extracted by the method mentioned above. Finally based on other feature lines from the input sketches, generate details like nose eyes and ears on the base mesh.
- **sketch-based modeling system**: Equipped with the shape construction methods, develop a user-friendly human character sketch based modelling system.

1.4 Contribution

We propose a sketch-based modeling system. To ease the complexity, we ask users only draw the nude character without clothes or accessories. In this research, we not only focus on the base mesh modelling phase, but also the sculpting phase (for example, refining the subtle details like ridges and valleys), by taking cross-section contours and feature curves into accounts. By inventing an ordinary differential equation (ODE) driven surface deformation technology, we reduce the computational cost compared with the state-of-art physics-based surface deformation methods, while

achieve a better physical plausibility than purely geometric shape deformation methods. Our main contributions to the sketch-based modeling technique are listed as follows:

- Introducing a new modelling technology based on cross-section contours to control the overall shape and add details to the mesh.
- Introducing an ODE driven mesh editing technology which generates plausible skin deformations with tolerable computation costs.
- Integrate primitive-based base mesh generation, cross-section contour-based shape modifications, and ODE-driven mesh editing into a modelling framework to achieve efficient and more detailed sketch-based character creation.
- Interactive system for users to draw sketches on the canvas and rotate the view.

Chapter 2

Literature Review

In this project, an ODE-driven sketch-based modeling system specialized in human character models is proposed. To be more exact, the data extracted from matching template 3D model are cross-section 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 modeling related basic knowledge, overview of sketch-based modelling systems and shape deformation technologies.

2.1 Human Character Modeling

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 three types of shell surface presentations: polygon mesh, free-form surface (eg. nonuniform rational B-splinesPiegl & Tiller [2012], Coons patchCoons [1967], etc), implicit surface (for example, metaball), and subdivision surfaces.

Volumetric modeling is very computational expensive, because unlike

shell surface which only stores surface data, it also stores interior space data under the surface. However, this meets the requirement of medical research where underlying anatomy are important and hence worth keeping.

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].

2.1.2 Anatomy-based Human Character Modeling

Good human character models should meet two requirement: 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 modeling encompassing skeleton, muscles, viscera, fat tissues, and skins has emerged. Among them, realistic appearance modelling consisting of human modelling and deformation has attracted a lot of research attention.

In the work of Beylot et al., the issues of image data for extraction of 3D shapes, surface reconstruction, topological modelling of different anatomical elements and potential applications of topological data base were addressed Beylot et al. [1996]. Scheepers et al. considered the influence of the musculature on 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 Scheepers et al. [1997]. Modeling muscles, bones, and generalized tissue as triangle meshes or ellipsoids, treating muscles as deformable discretized cylinders whose shapes change as the joints move, creating skin by voxelizing the underlying components, filtering, and extracting a polygonal isosurface, Jane and Allen proposed an improved, anatomically based approach to modeling and animating animals Jane & Allen [1997]. Allen et al. introduced an example-based method to capture human body scans, esti-

mate poses and kinematics, reconstruct a complete displaced subdivision surface in each pose and combine the surfaces using k-nearest-neighbors scattered data Allen et al. [2002]. Based on anatomy concepts, Nedel and Thalmann presented a method which divides a human representation into three different layers: the rigid body from a real skeleton, the muscle design and deformation, and the skin generation Nedel & Thalmann [1998]. Still using the anatomic model of deformable human bodies consisting of skeleton, muscles and skin, and presenting muscles by the action lines and the muscle shape, they introduced a mass-spring system with angular springs to physically simulate muscle deformations Porcher Nedel & Thalmann [2000]. Defining the space deformed by the control surface by a distance function around the surface, Singh and Kokkevis proposed a surface-oriented FFD which is more suitable for the automated skinning of characters Singh & Kokkevis [2000]. Deriving muscle motion and deformation from one or several action lines and deforming each action line by a 1D mass-spring system, Aubel and Thalmann proposed a muscle model based on physiological and anatomical considerations Aubel & Thalmann [2001]. Using a layered canonical model to represent the animal's skeleton, muscles, and skin, generating feature points to deform the attached mesh skin representation, Maryann et al. studied a semi-automatic technique for creating 3D models of creatures Simmons et al. [2002]. Mohr and Gleicher presented a framework for extending linear blending skinning which can capture detailed skin deformations and developed an automated method to build efficient and accurate character skins from a set of examples Mohr & Gleicher [2003]. Using motion capture and video cameras and providing a reconstruction algorithm to solve the problems of occlusion, hole-filling, deformation and noise-removal, Sand et al. examined how to acquire deformable human geometry from silhouettes Sand et al. [2003]. Taking the objective function to be proximity of transformed vertices to the range data, similarity between neighbouring transformations, and proximity of sparse markers at corresponding locations on the template and target surface, Allen et al. presented a new method by solving an optimisation problem and explored its applications in human body modelling Allen et al. [2003]. Hyun et al. represented each limb as a set of ellipsoids of vary-

ing size, approximated these ellipsoids with a swept ellipse, determined the difference between the original and approximated limbs using a displacement map, and proposed a new approach to model and deform a human or virtual character's arms and legs Hyun et al. [2003]. This work was extended to the modelling and deformation of a whole human body, and anatomical features are realised by a GPU-based collision-detection procedure Hyun et al. [2005]. Seo and Thalmann presented a set of techniques to automatically generate a new human body or modify an existing one by manipulating the parameters provided Seo & Magnenat-Thalmann [2004]. Using quasi-static linear deformation model and finite element method to calculate the deformation of chunks which represent the internal structures of a virtual character, Guo and Wong gave an approach to create skin deformations Guo & Cheong Wong [2005]. Venkataraman et al. introduced a combination of a kinematic and a variational model to deal with the wrinkling of skin by minimising a functional including energies for stretching, bending and self-intersection Venkataraman et al. [2005]. By decomposing the facial meshes into the global shape and 3D skin detail and recomposing the shape and 3D skin, Lee and Soon proposed a method to reproduce the scanned model which allows to simulate the exaggeration of the facial global shape, retain the original skin detail and transfer 3D skin from one to another Lee & Soon [2006]. Using sweeps following a simplified skeleton, Lee et al. proposed a new method to carry out realistic human hand modelling and deformation which can achieve real-time performance Lee et al. [2006]. Yang and Zhang presented a new anatomy-based skin deformation method which extracts major muscles automatically, formulates muscle sliding around a joint and sliding around a bone, develops a hybrid skin deformation to combine the strengths of anatomy based and smooth skinning, and is compatible with the current animation workflow Yang & Zhang [2006]. By using the so-called curve skeletons along with the joint-based skeletons, Yang et al. investigated an approach to deal with the inherent non-linear relations between the movement of skeletons and the caused skin shapes Yang et al. [2006].

Table 2.1: *Selected Terms for anatomyParent [2012]*

Sagittal plane	Perpendicular to the ground and divides the body into right and left halves
Coronal plane	Perpendicular to the ground and divides the body into front and back halves
Transverse plane	Parallel to the ground and divides the body into top and bottom halves
Distal	Away from the attachment on the limb
Proximal	Toward the attachment of the limb
Flexion	Movement of the joint that decreases the angle between two bones
Extension	Movement of the joint that increases the angle between two bones

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 validate sketches, as a viable source containing a good amount of 3D shapes' geometrical properties, can be used reconstruct 3D shapes sufficiently. Then this section will dive into some typical and representative state-of-art Sketch-Based Modeling systems that can be broadly divided into Direct shape generation and template-based mesh creation.

2.2.1 Theoretical Foundation As for Why 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] et al., human line drawings are quite consistent with each other. Lines drawn by artists largely overlapped one another, par-

ticularly 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 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 algorithm. 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 modeling systems' building bricks are geometrical primitives of all sorts.

To understand why geometrical primitives is chosen to be the starting points to modeling 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 evidence shows that these factors influence visual processes which determine the choice of figure and ground. Hence, we 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. 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 determine the core parameters of these simple geometric primitives so as to fit the 2D sketches, and then improves the model globally by inferring geosemantic constraints that define the relationship between different parts, the relations including: parallelism, orthogonality, of collinear centers, concentric, coplanar. In the work of Chen et al. [2013], 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 modeling 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. It's 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 single view modelling system with only one arbitrary angled guide image. But these manual workloads somehow impede the user-experience because for users to figure out all these semantic information is an anxious job. Given that those geometric semantic information can be represented by skeleton and computer is quite good at enumerating all the possible semantic constrains on each bone without omissions. In our research, we will address the problem by implementing a skeletal layer, such that the semantic constraints like alignment, same length and angle are automatically pre-set.

Other organic modelling system examples whose modelling procedure starts with the ellipsoids as geometry primitives are: Matisse [Bernhardt et al. 2008] and RÃľpousse[Joshi & Carr 2008].

Surface Inflation

Another way to generate shape is inflating a surface represent by closed 2D region/sketch to give it a volume [Kazmi et al. 2014]. 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 [2006a]. 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 2006b] support 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 surface inflation approach is good at creating stuffed animals and simple toys, they are not versatile enough to express geometrical details on the surface. BenSketchLi 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 BendSketch system has specified, which mimics the hatching technique artists often utilise to express the sense of volume and curvature information on the surfaces.

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 modeling has been proposed in Kraevoy et al. [2009] to find precise correspondences and determine mesh deformations. By combining skeleton-based deformation and mesh editing, an efficient approach was proposed in Kazmi et al. [2015] to quickly deform 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].

2.2.4 Layered Modeling

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 interprets 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 will be created as well. This work assumed input sketched strokes are around the underlying object, rather than curves lying on the objects. Treating inputted sketches as curves on base mesh are well investigated in works like Takayama et al. [2013a]; Nealen et al. [2007a]; Kara & Shimada [2007].

Skippy Krs et al. [2017] is another sketch-based curve modeling system that predicting 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 that occluded by the existing geometries.

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, for example the time-out based curve finalization, strokes drawn over the overlapped part of two tangent curves make them connected smoothly, etc.

Buchanan et al. 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 [Buchanan et al. 2013]. 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. This algorithm will benefit our research regarding to building triangulated meshes from cross-sections. Based on their observation, concept artists draw a lot of pictures in three quarters view, so their orientation establishing has a promising applications.

2.3 Sketch-based Deformation

Deformation tools provide the interface for users to interact and modify the mesh surface. 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 happen 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 2006b; Fleisch et al. 2004], [Terzopoulos et al. 1987] featured inflation approaches use smooth silhouettes, thus they create only smooth shapes. In order to create aesthetical 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 relied 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 depend 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. 2007a]. 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, Andrew et al. [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 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 computational-expensive, at least they are not as efficient as geometry based approaches. Hence, our 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 are used to solve the partial derivative equation in this scenario. For the concern of computation efficiency, we 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-section 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 we proposed impact the whole cross-section contour.

Nealen et al. [2007b] proposed a sketch-based mesh editing system that can deform a user selected ROI(region of interest) to fit the input sketch. Zimmermann et al. [2007] is able to compute from the oversketched feature line alone. Takayama et al. [2013b]

Chapter 3

Curve Fitting for Cross-Section

The study of how cross-section 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-section contours of human body to study human figures in his publications Four Books on Human Proportion (Vier Bücher von Menschlicher Proportion) of 1528.

3.1 Generalized ellipses

In the work of Hyun et al. 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. The mathematical description of sweep surfaces has the form of Hyun et al. [2005]

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

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.

(a) *Cross section curves of right leg*

(b) *Cross section curves of left arm*

(c) *Cross section curves of torso*

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

Using standard ellipses to approximate these cross sections will bring in large differences leading to unrealistic human modelling.

As indicated in Fig. 3.1, we give some cross-section curves of human legs, arms, and torso from a human model built with the polygon modelling approach and those approximated by elliptic cross sections where Fig. 3.1a was taken from human right leg, Fig. 3.1b was from human left arm, and Fig. 3.1c was 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

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

The above equation indicates that at the plane determined by a point $C(u_0)$ 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, we introduce the conception of generalized elliptic curves which consist of a number of trigonometric functions and $(j = 1, 2, \ldots)$. With the application of more trigonometric functions, the cross sections of human body can be approximated very accurately.

We divide generalized elliptic curves 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, we 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 perpendicular to 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 perpendicular to 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, we use curve fitting and the least square algorithm to determine the unknown constants in Eq. (3.3). To this aim, we 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.

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

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

$$
\begin{aligned}
\frac{\partial E_x}{\partial a_j} &= 0 \\
\frac{\partial E_y}{\partial b_j} &= 0 \\
(j = 0, 1, 2, 3, \ldots, 2J)
\end{aligned}
\tag{3.6}
$$

which leads to the following linear algebraic equations

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

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

$$\sum_{i=1}^{I}(x_i - x_c)\sin(k \times v_i) - a_0\sum_{i=1}^{I}\sin(k \times v_i) - \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j-1}\cos(j \times v_i)\sin(k \times v_i)$$
$$- \sum_{j=1}^{J}\sum_{i=1}^{I} a_{2j} sin(j \times v_i)\sin(k \times v_i) = 0$$
$$(k = 1, 2, 3, \ldots, J) \tag{3.7}$$

and

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

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

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

$(k = 1, 2, 3, \ldots, J)$

$$(3.8)$$

Each of the equations set (3.7) and (3.8) 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.

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

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

$$(3.9)$$

and

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

$$\mathbf{B} \times \mathbf{B}^T \times \begin{bmatrix} b_0 & b_1 & b_2 & \dots & b_{2J-1} & b_{2J} \end{bmatrix}^T = \begin{bmatrix} \sum_{i=1}^{I}(y_i - y_c) \\ \sum_{i=1}^{I}(y_i - y_c)\sin(v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\cos(v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\sin(2v_i) \\ \sum_{i=1}^{I}(y_i - y_c)\cos(2v_i) \\ \dots \\ \sum_{i=2J}^{I}(y_i - y_c)\sin(Jv_i) \\ \sum_{i=2J}^{I}(y_i - y_c)\cos(Jv_i) \end{bmatrix} \tag{3.10}$$

Solving equations (3.9) and (3.10), respectively, 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: subfigure a,b,c shows Cross section curve of human leg with* $J = 2,3,4$*, respectively; subfigure d,e,f shows Cross section curve of human torso 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 to achieve high accuracy of curve modelling.

Due to different shapes of cross section 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, we here propose the following strategy, and the results were shown in Fig.3.4.

First, we define an average error E_a and a maximum error E_m below. 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.

$$E_a = \frac{1}{I}\sum_{i=1}^{I}\frac{d_i}{\bar{d}_i} \tag{3.11}$$

where

$$\begin{aligned}
X &= x_i - x_c - a_0 - \sum_{j=1}^{J}\left(a_{2j-1}\cos(j\times v_i) + a_{2j}\sin(j\times v_i)\right)\\
Y &= y_i - y_c - b_0 - \sum_{j=1}^{J}\left(b_{2j-1}\sin(j\times v_i) + b_{2j}\cos(j\times v_i)\right)\\
d_i &= \sqrt{X^2+Y^2}\\
\bar{d}_i &= \sqrt{(x_i-x_c)^2+(y_i-y_c)^2}
\end{aligned} \tag{3.12}$$

and

$$E_m = max\{\frac{d_1}{\bar{d}_1}, \frac{d_2}{\bar{d}_2}, \frac{d_3}{\bar{d}_3}, \ldots, \frac{d_J}{\bar{d}_J}\} \tag{3.13}$$

Then, we can set different errors to determine the required terms in Eq.(3.3). For example, we take the average error and maximum error not more than 1% and 5%, respectively. That is

$$\begin{aligned}
E_a &\le 1\%\\
E_m &\le 2.5\%
\end{aligned} \tag{3.14}$$

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, we 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. (3.14), or one of E_{a10} and E_{m10} does not satisfy Eq. (3.3), 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. We round off it to

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

the nearest integer. With J_1, we calculate the average errors E_{aJ_1} and the maximum errors E_{mJ_1} and examine whether they have met Eq. (3.3). If both E_{aJ_1} and E_{mJ_1} have met Eq. (3.14) and $J_1 < 3$, the linear extrapolation operation occurs with J_1 and $J = 3$. If one of E_{aJ_1} and E_{mJ_1} does not satisfy Eq. (3.3) and $J_1 > 10$, the linear extrapolation operation is performed with $J = 10$ and J_1. If both E_{aJ_1} and E_{mJ_1} have met Eq. (3.14) and $3 < J_1 < 10$, the linear interpolation operation is conducted between $J = 3$ and J_1. If one of E_{aJ_1} and E_{mJ_1} does not satisfy Eq. (3.3) and $3 < J_1 < 10$, the linear interpolation operation is carried out between J_1 and $J = 10$.

Since there are two quantities E_a and E_m which can be employed for the interpolation, we must determine which one should be used. Obviously, if only one of E_a and E_m does not meet Eq. (3.3), this error is used for the interpolation. If both of them do not satisfy Eq. (3.3), we 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-section 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 Bridging multiple curve segments

In this section, we will discuss the possibility to approximate and bridge multiple curve segments with the aforementioned trigonometric series representation3.1 with C1 continuity between adjacent segments.

One way is in piece-wise fashion: step 1. approximate the first segment as a new curve segment and is called a generalized elliptic segment; step 2. compositing its next adjacent segment by adding the C1 continuity to the

first generated generalized elliptic segment as a constraint; step 3. repeating step 2 until each of the segments is reformatted in trigonometric series representation and is linked to the last generalized elliptic segment. The curve consisting of these generalized elliptic segments is named a composite generalized elliptic curve.

The other way is to rebuild all of these curve segments together into one integrated generalized elliptic curve , once and for all. Since the trigonometric series are continuous derivative functions, the C1 continuity requirement is achieved across the new generalized elliptic curve.

Finally we will compare the results of these two methods in terms of computational efficiency.

In order to guarantee two adjacent generalized elliptic segments are smoothly connected together, we investigate the continuity between these two segments.

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

Figure 3.6: *two segments of a hip cross-section curve*

Firstly, we divide a cross section curve from human hip into two separate curve segments $\widehat{ABC}$ and $\widehat{CDA}$ as indicated in Fig. 3.6. With Eq. (3.3)

and curve fitting algorithm Eq. (3.9) and Eq. (3.10), we can determine the first generalized elliptic segment corresponding to the curve segment $\widehat{ABC}$ which is defined within the region of the parametric variable υ from $\upsilon = 0$ to $\upsilon = \pi$. The next task is to determine the second generalized elliptic segment corresponding to the curve segment $\widehat{CDA}$ which is defined in the region of the parametric variable υ from $\upsilon = \pi$ to $\upsilon = 2\pi$. 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 $\upsilon = 0$ and $\upsilon = \pi$ should be equal to those of the second generalized elliptic segment at $\upsilon = 2\pi$ and $\upsilon = \pi$, respectively. Therefore, we have the following positional continuity conditions for the x, y position function

$$\begin{aligned}
&\bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} (-1)^j \bar{\bar{a}}_{2j-1} = \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{J} (-1)^j \bar{a}_{2j-1} \\
&\bar{\bar{x}}_c + \bar{\bar{a}}_0 + \sum_{j=1}^{J} \bar{\bar{a}}_{2j-1} = \bar{x}_c + \bar{a}_0 + \sum_{j=1}^{J} \bar{a}_{2j-1} \\
&\bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} (-1)^j \bar{\bar{b}}_{2j} = \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{J} (-1)^j \bar{b}_{2j} \\
&\bar{\bar{y}}_c + \bar{\bar{b}}_0 + \sum_{j=1}^{J} \bar{\bar{b}}_{2j} = \bar{y}_c + \bar{b}_0 + \sum_{j=1}^{J} \bar{b}_{2j}
\end{aligned} \tag{3.15}$$

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

$$\begin{aligned}
&\sum_{j=1}^{J} (-1)^j j \bar{\bar{a}}_{2j} = \sum_{j=1}^{J} (-1)^j j \bar{a}_{2j} \\
&\sum_{j=1}^{J} j \bar{\bar{a}}_{2j} = \sum_{j=1}^{J} j \bar{a}_{2j} \\
&\sum_{j=1}^{J} (-1)^j j \bar{\bar{b}}_{2j-1} = \sum_{j=1}^{J} (-1)^j j \bar{b}_{2j-1} \\
&\sum_{j=1}^{J} j \bar{\bar{b}}_{2j-1} = \sum_{j=1}^{J} j \bar{b}_{2j-1}
\end{aligned} \tag{3.16}$$

From the first two of Eqs. (3.15) and (3.16), 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.(3.17). To be more specifically, if we subtract first two of Eqs. (3.15), we can get the formula for $\bar{\bar{a}}_1$. By subtracting the first two of Eqs. (3.16), we can get the formula for $\bar{\bar{a}}_2$. Substituting $\bar{\bar{a}}_1$ from Eqs.(3.17) into the first equation in Eqs.(3.15) and we get the formula for $\bar{\bar{a}}_3$. Adding the first two of Eqs.(3.16), we can get the formula for $\bar{\bar{a}}_4$.

$$
\begin{aligned}
\bar{\bar{a}}_1 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,\bar{a}_{2j-1} - \sum_{j=3}^{J}[1-(-1)^j]\,\bar{\bar{a}}_{2j-1}\Big\} \\
\bar{\bar{a}}_2 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,j\,\bar{a}_{2j} - \sum_{j=3}^{J}[1-(-1)^j]\,j\,\bar{\bar{a}}_{2j}\Big\} \\
\bar{\bar{a}}_3 &= \bar{x}_c - \bar{\bar{x}}_c + \bar{a}_0 - \bar{\bar{a}}_0 + \frac{1}{2}\Big\{\sum_{j=1}^{J}[1+(-1)^j]\,\bar{a}_{2j-1} - \sum_{j=3}^{J}[1+(-1)^j]\,\bar{\bar{a}}_{2j-1}\Big\} \\
\bar{\bar{a}}_4 &= \frac{1}{4}\Big\{\sum_{j=1}^{J}[1+(-1)^j]\,j\,\bar{a}_{2j} - \sum_{j=3}^{J}[1+(-1)^j]\,j\,\bar{\bar{a}}_{2j}\Big\}
\end{aligned}
\tag{3.17}
$$

From the last two of Eqs.(3.15) and (3.16) , 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.(3.18). More specifically, subtracting the last two of Eqs.(3.15), we can get the $\bar{\bar{b}}_2$; subtracting the last two of Eqs.(3.16), we can get the $\bar{\bar{b}}_1$; adding the last two of Eqs.(3.16), we can get the $\bar{\bar{b}}_3$; adding the last two of Eqs. (3.15), we can get $\bar{\bar{b}}_4$.

$$
\begin{aligned}
\bar{\bar{b}}_1 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,j\,\bar{b}_{2j-1} - \sum_{j=3}^{J}[1-(-1)^j]\,j\,\bar{\bar{b}}_{2j-1}\Big\} \\
\bar{\bar{b}}_2 &= \frac{1}{2}\Big\{\sum_{j=1}^{J}[1-(-1)^j]\,\bar{b}_{2j} - \sum_{j=3}^{J}[1-(-1)^j]\,\bar{\bar{b}}_{2j}\Big\} \\
\bar{\bar{b}}_3 &= \frac{1}{4}\Big\{\sum_{j=1}^{J}[1+(-1)^j]\,j\,\bar{b}_{2j-1} - \sum_{j=3}^{J}[1+(-1)^j]\,j\,\bar{\bar{b}}_{2j-1}\Big\} \\
\bar{\bar{b}}_4 &= \bar{y}_c - \bar{\bar{y}}_c + \bar{b}_0 - \bar{\bar{b}}_0 + \frac{1}{2}\Big\{\sum_{j=1}^{\bar{J}}[1+(-1)^j]\,\bar{b}_{2j} - \sum_{j=3}^{J}[1+(-1)^j]\,\bar{\bar{b}}_{2j}\Big\}
\end{aligned}
\tag{3.18}
$$

Substituting Eqs. (3.17) and (3.18) into Eq. (3.3), the x and y compo-

nents of the positional function for the second generalized elliptic segment are changed into

$$
\begin{aligned}
\bar{\bar{x}}(v) &= (\bar{x}_c + \bar{a}_0)\cos 2v + \frac{1}{2}\sum_{j=1}^{J}\left[D_1(j,v)\bar{a}_{2j-1} + jD_2(j,v)\bar{a}_{2j}\right] \\
&\quad + (\bar{\bar{x}}_c + \bar{\bar{a}}_0)(1-\cos 2v) \\
&\quad + \frac{1}{2}\sum_{j=3}^{J}\left\{\left[2\cos jv - D_1(j,v)\right]\bar{\bar{a}}_{2j-1} + \left[2\sin jv - jD_2(j,v)\right]\bar{\bar{a}}_{2j}\right\} \\
\bar{\bar{y}}(v) &= (\bar{y}_c + \bar{b}_0)\cos 2v + \frac{1}{2}\sum_{j=1}^{J}\left[jD_2(j,v)\bar{b}_{2j-1} + D_1(j,v)\bar{b}_{2j}\right] \\
&\quad + (\bar{\bar{y}}_c + \bar{\bar{b}}_0)(1-\cos 2v) \\
&\quad + \frac{1}{2}\sum_{j=3}^{J}\left\{\left[2\sin jv - jD_2(j,v)\right]\bar{\bar{b}}_{2j-1} + \left[2\cos jv - D_1(j,v)\right]\bar{\bar{b}}_{2j}\right\}
\end{aligned}
\tag{3.19}
$$

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. (3.19) we can perform curve fitting for the curve segment $\widehat{CDA}$. Firstly, constructing similar Errors for $\bar{\bar{x}}$ and $\bar{\bar{y}}$ as that in Eqs.(3.4), (3.5) and (3.6). Then by setting the derivatives of the Errors equal to zero, we can get the following equations involving $2J-3$ items $\bar{\bar{a}}_0, \bar{\bar{a}}_5, \bar{\bar{a}}_6, \ldots, \bar{\bar{a}}_{2J-1}, \bar{\bar{a}}_{2J}$ as unknown variables for x position, and another $2J-3$ items $\bar{\bar{b}}_0, \bar{\bar{b}}_5, \bar{\bar{b}}_6, \ldots, \bar{\bar{b}}_{2J-1}$, $\bar{\bar{b}}_{2J}$ for y position

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

$$
\begin{aligned}
&\sum_{i=1}^{I}(1-\cos 2v_i)\times A(i)=\sum_{i=1}^{I}(1-\cos 2v_i)\times C_a(i)\\
&\sum_{i=1}^{I}P_1(k,i)\times A(i)=\sum_{i=1}^{I}P_1(k,i)\times C_a(i),(k=5,7,9,\ldots,2J-1)\\
&\sum_{i=1}^{I}P_2(k,i)\times A(i)=\sum_{i=1}^{I}P_2(k,i)\times C_a(i),(k=6,8,10,\ldots,2J)
\end{aligned}
\tag{3.20}
$$

and

$$
\begin{aligned}
&C_b(i)=y_i-(\bar{y}_c+\bar{b}_0)\cos 2v_i-\bar{\bar{y}}_c(1-\cos 2v_i)\\
&-\frac{1}{2}\sum_{j=1}^{J}[jD_2(j,v_i)\bar{b}_{2j-1}+D_1(j,v_i)\bar{b}_{2j}]\\
&B(i)=\bar{\bar{b}}_0(1-\cos 2v_i)+\sum_{j=3}^{J}[\sin jv_i-\frac{j}{2}D_2(j,v_i)]\bar{\bar{b}}_{2j-1}\\
&+\sum_{j=3}^{J}[\cos jv_i-\frac{1}{2}D_1(j,v_i)]\bar{\bar{b}}_{2j}\\
&P_3(k,i)=\sin(\frac{k+1}{2}\times v_i)-\frac{k+1}{4}D_2(\frac{k+1}{2},v_i),(k=5,7,9,\ldots,2J-1)\\
&P_4(k,i)=\cos(\frac{k}{2}\times v_i)-\frac{1}{2}D_1(\frac{k}{2},v_i),(k=6,8,10,\ldots,2J)\\
&\sum_{i=1}^{I}(1-\cos 2v_i)\times B(i)=\sum_{i=1}^{I}(1-\cos 2v_i)\times C_b(i)\\
&\sum_{i=1}^{I}P_3(k,i)\times B(i)=\sum_{i=1}^{I}P_3(k,i)\times C_b(i),(k=5,7,9,\ldots,2J-1)\\
&\sum_{i=1}^{I}P_4(k,i)\times B(i)=\sum_{i=1}^{I}P_4(k,i)\times C_b(i),(k=6,8,10,\ldots,2J)
\end{aligned}
\tag{3.21}
$$

These two systems of linear equations .(3.20) and (3.21) can be represented in matrix form, as in Eqs. (3.22) and Eqs. (3.23). Note that since either $\mathbf{M}_a$ or $\mathbf{M}_b$ is square matrix, we can't eliminate $\mathbf{M}_a$ or $\mathbf{M}_b$ in the equa-

tions with their respective inverse $\mathbf{M}_a^{-1}$ or $\mathbf{M}_b^{-1}$.

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

$$\mathbf{M}_a \times \mathbf{M}_a^T \times \begin{bmatrix} \bar{\bar{a}}_0 \\ \bar{\bar{a}}_5 \\ \bar{\bar{a}}_6 \\ \bar{\bar{a}}_7 \\ \dots \\ \bar{\bar{a}}_{2J-1} \\ \bar{\bar{a}}_{2J} \end{bmatrix} = \mathbf{M}_a \times \begin{bmatrix} C_a(1) \\ C_a(2) \\ C_a(3) \\ C_a(4) \\ \dots \\ C_a(I-1) \\ C_a(I) \end{bmatrix} \tag{3.22}$$

and

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

$$\mathbf{M}_b \times \mathbf{M}_b^T \times \begin{bmatrix} \bar{\bar{b}}_0 \\ \bar{\bar{b}}_5 \\ \bar{\bar{b}}_6 \\ \bar{\bar{b}}_7 \\ \dots \\ \bar{\bar{b}}_{2J-1} \\ \bar{\bar{b}}_{2J} \end{bmatrix} = \mathbf{M}_b \times \begin{bmatrix} C_b(1) \\ C_b(2) \\ C_b(3) \\ C_b(4) \\ \dots \\ C_b(I-1) \\ C_b(I) \end{bmatrix} \tag{3.23}$$

With the solution of these two systems of equations, we obtain all $2J-3$ unknown variables $\bar{\bar{a}}_0, \bar{\bar{a}}_5, \bar{\bar{a}}_6, \bar{\bar{a}}_7, \dots, \bar{\bar{a}}_{2J}$ 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 second generalized elliptic segment according to Eqs.(3.19), we generate the composite generalized elliptic segments depicted in the left column of Fig. 3.7 where the curves in red stand for the original ones (ground-truth), that in green and purple were produced from the composite generalized elliptic segments, whereas in the right column are their one-piece generalized ellipse counterparts which shows in green and the original ones are again depicted in red. From these images, we know both composite generalized elliptic segments and generalized ellipse approach the original curve very well.

However, when it comes to computational efficiency, the composite gen-

eralized elliptic segments require no less of unknown constants than that of the generalized ellipse. For example, as we can see from the first row of Fig.3.7, to compute the result in the left figure, we need 60 constants to get the composite generalized elliptic segments (in this case, two segments); meanwhile, with similar computational and aesthetics accuracy, to get the one-piece generalized ellipse in the right figure, we only need 38 constants. Same results occur in the other examples in Fig.3.7. Therefore, under the same level of computational accuracy, the algorithm of composite generalized elliptic segments needs more constants, and as such, it can't improve the computational efficiency, or raise human modelling performance under the same approximate accuracy, compared to one-piece generalised ellipse algorithm.

(a) *belly contour in two composite generalized elliptic segments number of total constants=60 Ea=0.24%, Em=1.16%*

(b) *belly contour in generalized ellipse number of total constants=38 Ea=0.50%, Em=1.71%*

(c) *epigastrium contour in two composite generalized elliptic segments number of total constants=60 Ea=0.46%, Em=2.19%*

(d) *epigastrium contour in generalized ellipse number of total constants=30 Ea=0.46%, Em=2.42%*

(e) *chest contour in two composite generalized elliptic segments number of total constants=44 Ea=0.74%,Em=2.25%*

(f) *chest contour in generalized ellipse number of total constants=38 Ea=0.79%,Em=1.93%*

(g) *hip contour in two composite generalized elliptic segments number of total constants=76 Ea=0.36%,Em=1.97%*

(h) *hip contour in generalized ellipse number of total constants=66 Ea=0.44%,Em=2.23%*

Figure 3.7: *composite generalized elliptic segments in green and purple; one-piece generalized ellipse in green; original belly, epigastrium, chest and hip contour are in red*

Chapter 4

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 Finite Difference Method as numeric solution.

4.1 Pipeline Overview

As shown in the Fig. 4.1, one can extract 2D silhouette contours from sketches, or to directly draw 2D silhouette contours.

After generating 2D silhouette contours, the next step is to set superellipsoid as 3D primitives. We sample the silhouette contours into point cloud and we use radial Euclidean distance to format the global distortionChevalier et al. [2003] to determine the parameters of the super-ellipse: ϵ_z, r_x, r_y. If one only feeds the system with the 2D sketch in one view not providing z-axis depth information, then the radius of z-axis r_z and ϵ_{xy} requires users to manually set, and any necessary translations in z-axis should be done manually. Since the parameter determination took place in object space, when we transform the superellipsoid primitives into the world space, they will align with its silhouette in world space.

Then the ODE-driven deformation method described in Section 4.3 is applied to deform the primitives to exactly match the corresponding 2D

silhouette contours.

Figure 4.1: *pipeline overview.*

4.2 User Interface of Primitive Deformer

The user interface of our developed primitive deformer uses four windows as shown in Figure 4.2. 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 these primitives, and the deformed primitives in the front view. The bottom left window is used to draw and edit 2D silhouette contours and deformed the primitives in the die view. The bottom right window is used to draw and edit cross section contours and deform 3D models to obtain realistic shapes. Taking the left leg of the female warrior as an example Figure 4.2, the primitive of the left leg is shown in the upper left window. The user-generated 2D silhouette contours in the front view and side view plus the cross section contours are depicted in the top right, bottom left and bottom right windows. Our proposed primitive deformer described in Subsection 4.5 deforms the primitive to exactly match the generated 2D silhouette contours and cross section contours, create a realistic 3D leg model shown in the bottom left window.

Figure 4.2: *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*

4.3 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 4.5 to demonstrate the algorithm of our proposed primitive deformer and how it deforms a 3D primitive to match the 2D silhouette contours.

Figure 4.3a depicts a torso model of the female warrior which is represented with a cylinder. The 2D silhouette contour to be matched is also shown in the image. Figure 4.3b 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

(a) **(b)**

Figure 4.3: *Primitive deformer: a) female warrior torso represented with a cylinder and its 2D silhouette contour, b) deformed shape of the cylinder*

Euler-Lagrange PDE (partial differential equation) which is widely used in physically-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]:

$$E_{shell}(\mathbb{S}') = \int_{\Omega} k_s \| I' - I \|_F^2 + k_b \| \Pi' - \Pi \|_F^2 \, \mathrm{d}u \, \mathrm{d}v \tag{4.1}$$

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 4.1, 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.,

$$\tilde{E}_{shell}(\mathbf{d}) = \int_{\Omega} k_s(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2) + k_b(\|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\| + \|\mathbf{d}_{vv}\|^2) \, \mathrm{d}u \, \mathrm{d}v, \tag{4.2}$$

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

$$-k_s \triangle \mathbf{d} + k_b \triangle^2 \mathbf{d} = 0, \tag{4.3}$$

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

$$\begin{aligned} \triangle \mathbf{d} &= div \nabla \mathbf{d} = \mathbf{d}_{uu} + \mathbf{d}_{vv}, \\ \triangle^2 \mathbf{d} &= \triangle(\triangle \mathbf{d}) = \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv}. \end{aligned} \tag{4.4}$$

Up till now, the minimization of the thin-shell deform energy is described in a continuous space, depicting the physical phenomenon happens in real world. In next section, we will discrete Eq. (4.3) for computers to simulate.

4.4 Finite Difference Method for Geometric Processing

In this section, a discretized ODE-driven and sketch-guided primitive deformation method is proposed. It is developed from a simplified version of the Euler-Lagrange PDE (Partial Differential Equation), which is widely used in physically-based surface deformations. 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. 4.5 to change the shape of the primitive can be transformed to 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 (4.3) is used to describe the generator, the parametric variable v in Equation (4.3) 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 (4.4), we obtain the following simplified version of the Euler-Lagrange PDE, seen as (4.5), which is actually a vector-valued ODE.

$$k_b \frac{\partial^4 \mathbf{d}}{\partial u^4} - k_s \frac{\partial^2 \mathbf{d}}{\partial u^2} = 0. \tag{4.5}$$

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 (4.5).

Figure 4.4: *Typical node i for the finite difference approximations of derivatives*

For a typical node shown in Figure 4.4, the central finite difference approximations of the second and fourth order derivatives can be written as

Chaudhry et al. [2013]:

$$\begin{aligned}\frac{\partial^2 \mathbf{d}}{\partial u^2}|_i &= \frac{1}{\triangle u^2}(\mathbf{d}_{i+1} - 2\mathbf{d}_i + \mathbf{d}_{i-1}),\\ \frac{\partial^4 \mathbf{d}}{\partial u^4}|_i &= \frac{1}{\triangle u^4}[6\mathbf{d}_i - 4(\mathbf{d}_{i-1} + \mathbf{d}_{i+1}) + \mathbf{d}_{i-2} + \mathbf{d}_{i+2}].\end{aligned} \tag{4.6}$$

Introducing Equation (4.6) into Equation (4.5), the following finite difference equation at a representative node i can be written as:

$$(6k_b + 2k_s h^2)\mathbf{d}_i + k_b\mathbf{d}_{i-2} + k_b\mathbf{d}_{i+2} - (4k_b + k_s h^2)\mathbf{d}_{i-1} - (4k_b + k_s h^2)\mathbf{d}_{i+1} = 0. \tag{4.7}$$

4.5 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 4.5b. 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 4.5a as an example, we find two corresponding points $\mathbf{c}_{13}$ and $\mathbf{c}_{23}$ 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 4.5b. 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 4.5a and depict it as a dashed curve in Figure 4.5b. 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 $\mathbf{d}(u) = \mathbf{c}'(u) - \mathbf{c}(u)$. Our task is to find the displacement difference $\mathbf{d}(u)$ and generate the deformed curve $\mathbf{c}'(u) = \mathbf{d}(u) + \mathbf{c}(u)$.

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 4.5b. The displacement difference at node 0 and node N is known, i. e. $\mathbf{d}_0 = \mathbf{c}'_{13} - \mathbf{c}_{13}$ and $\mathbf{d}_N = \mathbf{c}'_{23} - \mathbf{c}_{23}$.

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

Figure 4.5: *Finite difference nodes for local shape manipulation from sketches in different view planes.*

When we write the finite difference equations for the nodes 2, 3, $2N-1$ and $2N$, the node 1 will be involved, and we have $\mathbf{d}_1 = \mathbf{c}'_{13} - \mathbf{c}_{13}$. The finite difference equations at these points can be derived from Equation (4.7). Substituting $\mathbf{d}_1 = \mathbf{c}'_{13} - \mathbf{c}_{13}$ 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 $N-1, N, N+2$ and $,N+3$ the node $N+1$ will be involved, and we have $\mathbf{d_{N+1}} = \mathbf{c'_{23}} - \mathbf{c_{23}}$. Once again, the finite difference equations at these points can be derived from Equation (4.7). Substituting $\mathbf{d}_{N+1} = \mathbf{c}'_{23} - \mathbf{c}_{23}$ into these equations, we obtain the finite difference equations for the nodes $N-1$, N, $N+2$ and $N+3$, 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 (4.7). 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 $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{N-2}, \mathbf{d}_{N-1}, \mathbf{d}_{N+1}, \mathbf{d}_{N+2}, \ldots, \mathbf{d}_{2N-2}$, and $\mathbf{d}_{2N-1}$. Adding the $\mathbf{d}_i$ ($i = 0, 1, \ldots, 2N-2, 2N-1$) to the original curve $\mathbf{c}(u)$, we obtain the deformed curve $\mathbf{c}'(u)$, and depict it as a solid curve in Figure 4.5b. 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.

Other applications of this method other than human character modeling including more generic organic objects, and deformations responding to free form curves. Fig 4.6a has shown this ODE-driven deforming method has good performance when applying to 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, for example in Fig 4.6b, the leg has been deformed responding to free form curves after initally deformed by single-view sketch strokes.

(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 free form red-colored curves*

Figure 4.6: *additional applications of ODE-driven Sketch-based deformations*

However, since the shape of cross-section of a superellipsoid lacks variations and failed to capture the complexity of the actual form. Hence deforming from a superellipsoid primitive alone is not ideal. In such cases, we edit the cross-section first (Fig.4.7b), and then deform the shape again with the folowing treatment.

When a cross-section contour is used to modify the cross-section shape, the influence range of the cross-section contour is first specified to generate two boundary curves. The cross-section 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-section contour, are taken to be the nodes of the finite difference calculations and introduced into Equation (4.7) 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 (4.7) to reduce two more unknown constants.

With the above ODE-driven deformations, realistic cross-section shapes are created. Taking the vest created in Chapter 5 5.3 and shown in Figure 4.7a as an example, one original cross-section curve of the vest is an ellipse highlighted in blue in Figure 4.7b, 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 4.7c. The cross-section shape within the boundary curves is modified and shown in Figure 4.7d.

4.6 Conclusion

For now, we've achieved the goal of generate primitives from silhouette sketches of two orthogonal view-planes by implementing finite different method on solving an ODE equation. It is clear that deformed primitives have matched the generated 2D silhouette contours exactly Fig. 4.8.

Figure 4.7: *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.*

Figure 4.8: *Examples of the primitive deformation method*

Chapter 5

Free-form Patch Generator

The base mesh generated from the deformed primitives described in chapter4 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, we 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, (ii) one open and one closed sketches, and (iii) two open and one closed sketches. In the following, we will describe the algorithms to create 3D shapes from the three different cases of sketches.

5.1 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 $\mathbf{P}$ 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 5.1. 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 5.1: *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 5.1b, passing through the four corresponding points $\mathbf{c}_{1j}$, $\mathbf{c}_{2j}$, $\mathbf{c}_{3j}$ and $\mathbf{c}_{4j}$ (shown in Figure 5.1a 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 5.1b. 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 5.1b, 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 above $2N$ 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, we create local 3D shapes. Figure 5.2 shows an example of the shape generators.

A head model shown in Figure 5.2 is used to demonstrate the above

Figure 5.2: *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) the cross sections of the male head, d) lofting the cross-sections into the male head mesh.*

method. Figure 5.2a and Figure 5.2b show two open head silhouette contours in two different view plans. Figure 5.2c shows the cross section for each discrete parameter u_i. Figure 5.2d depicted 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 5.3a and Figure 5.3b, 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 5.3c and Figure 5.3d, respectively.

Another example given in Figure 5.3 is a neck model. Figure 5.3e shows the 4 disconnected contours, Figure 5.3f shows the cross-sections for each discrete u, and the final mesh is shown in Figure 5.3g.

5.2 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 5.4, 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

Figure 5.3: *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 our method in front view, d) leg mesh in side view, e) contours of a neck, f) cross sections of a neck generated by our method, g) final neck mesh.*

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 5.4: *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 but larger than 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. If the created local 3D model is to be connected to another 3D model with both positional and tangential continuities, 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:

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

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, $\ldots$, N, $N+2$, $\ldots$, $2N$, $2N+2$ equals to 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 $\mathbf{d}_0$, $\mathbf{f}_1$, $\ldots$, $\mathbf{d}_N$, $\mathbf{f}_{N+1}$, $\mathbf{d}_{N+2}$, $\ldots$, $\mathbf{d}_{2N}$, $\mathbf{f}_{2N+1}$, and $\mathbf{d}_{2N+2}$. Together with $\mathbf{d}_1 = \mathbf{c}_{1j}$, $\mathbf{d}_{N+1} = \mathbf{c}_{2j}$, and $\mathbf{d}_{2N+1} = \mathbf{c}_{3j}$ to replace $\mathbf{f}_1$, $\mathbf{f}_{N+1}$, and $\mathbf{f}_{2N+1}$, respectively, we can 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 5.5).

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

Figure 5.5: *The shape creation from one open and one closed silhouette contours in two different view planes.*

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

The algorithms given in Subsections 5.1 and 5.2 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, we 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 5.6, 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 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

Figure 5.6: *Finite difference nodes in the algorithm for the case of two open and one closed silhouette contours.*

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, and $N-1$, the node 0 beyond the boundary node 1 will be involved. We can use the method given in Subsection 5.2 to determine the nodes 0 from the vector-valued first derivatives $\mathbf{T}_{5,0}$ through

$$\mathbf{d}_0 = \mathbf{d}_2 - 2\mathbf{T}_{5,0}\triangle u. \tag{5.1}$$

For each of the inner nodes $2, 3, \ldots, N-1, N$, 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 (4.7). 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 $\mathbf{f}_{5,i} = \mathbf{f}_i\,(i = 2, 3, \ldots, N-1, N)$ 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 $\mathbf{c}_7$ 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 $\mathbf{P}$:

$$\mathbf{f}_i = \mathbf{f}_{5,i} + \frac{L_j}{L}(\mathbf{f}_{6,i} - \mathbf{f}_{5,i}), \tag{5.2}$$

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 $\mathbf{P}$:

$$\begin{aligned} \mathbf{T}_0 &= \mathbf{T}_{5,0} + \frac{L_j}{L}(\mathbf{T}_{6,0} - \mathbf{T}_{5,0}), \\ \mathbf{T}_1 &= \mathbf{T}_{5,1} + \frac{L_j}{L}(\mathbf{T}_{6,1} - \mathbf{T}_{5,1}). \end{aligned} \tag{5.3}$$

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 (5.1), 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 (4.5). with the sculpting force being calculated by Equation (5.2), 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 5.7a and 5.7b. The two open curves divide the mouth into four regions. One surface is created for each of the four regions. Figure 5.7c depicts the mouth model consisting of the four surfaces.

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

Figure 5.7: *A mouth model creation from two open and one closed silhouette contours in three different view planes.*

Figure 5.8: *Finite difference nodes for the algorithm for four open and two closed silhouette contours.*

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 indicated in Figure 5.8, and a sweeping surface is encircled by four curves. With

this extension, we can create a 3D vest model and an eye socket model, as shown in Figure 5.9a and Figure 5.9b, respectively.

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

Figure 5.9: *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 5.10 gives such an example where the three curves in red are open as shown in Figure 5.10a. The generated 3D model is depicted in Figure 5.10b.

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

Figure 5.10: *Creation of a thin sheet on the left forearm.*

5.4 Experiments and Comparisons

In this section, we present our experiments to validate that our approach can create more realistic shapes and use the creation of a 3D facial model to demonstrate that our 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 our ODE-driven deformation approach.

(a)

(b)

Figure 5.11: *Deformation comparisons: (a) deform a plastic ruler by fix one end of the ruler and put a heavy weight on the other free end of the ruler, (b) deform a plastic ruler by place 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 underneath gray curves are the ground-truths.*

As shown in Figure 5.11a, 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 to 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 our ODE-driven de-

formations 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 $\frac{\partial^3 d_{N+1}}{\partial u^3} = \frac{f}{k_b}$. We use the bend modulus of PP plastics to approximate the bending stiffness $k_b = 5$, $GPa = 0.5N/cm^2$. The red curve in Figure 5.11a shows its simulation, which is very close the ground-truth (the ruler itself).

In another experiment shown in Figure5.11b, two identical objects simply support 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 our ODE-driven deformation technique is shown as the red curve in Figure5.11b, which is also very close to the ground-truth (the gray curve). This indicates that the OED-driven deformations produce realistic appearances.

By comparing the red curves and the real rulers in Figure 5.11, the red curve is very close to the real ruler trajectories. It indicates that ODE-driven deformations give a realistic appearance.

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 our approach can efficiently add 3D details to the created 3D models. Taking the female warrior face shown in Figure 5.12a as an example, we compare 3D face generation between the traditional polygon modelling and our sketch-based, ODE-driven shape modelling. With the Maya-provided polygon modelling, about 40 hours are used to create a detailed facial model in Figure 5.12b. With our sketch-based, ODE-driven shape modelling approach, a curve network shown in Figure 5.12c which defines the female warrior face is first generated. Then, the curve network is decomposed into different groups of sketches and our ODE-driven deformations are used to automatically create surface patches from different

groups of sketches. All the created surface patches are naturally and smoothly connected to generate the detailed facial model shown in Figure 5.12d. The total time of obtaining the detailed facial model in Figure 5.12d including generating the facial curve network in Figure 5.12c was about 5 hours. Compared to the approach of manipulating primitives in one view plane proposed in Gingold et al. [2009], our approach can create more detailed 3D shapes.

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

Figure 5.12: *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 our sketch-based modelling.*

Our 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.

5.5 Conclusion

In this chapter, a free-form patch generator is developed. It invites a force field into the ODE equation in Chapter 4 and utilises the border tangent as border constraints to solve the equations. Here we use the example shown in Fig. 5.13 to demonstrate the outcome of of our proposed free-form patch generator.

(a) **(b)**

Figure 5.13: *a) sketch of a male character.Image retrieved from: www.wolfiesden.com, b) 3d model by using our method.*

Chapter 6

Cross-Section Defined Surface Modeling

In this chapter, a modelling method to 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 or composite generalized elliptic segments. The parts are constructed from these generalized elliptic curves and assembled together to create the 3D model. With the approach presented in this paper, 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, our approach can reduce data size of surface modeling.

6.1 Introduction

In order to facilitate a more accurate mesh editing, extracted cross-section contours from template models are used to guide the deformation of base mesh. First off, this research uses the 3D models from dataset Autodesk. 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, extract the cross-sections on the template model corresponding to the sketches users have inputted. Then, the extracted cross-sections are fitted onto the generated meshes. And finally, editing the meshes to meet the constraints of the newly fitted cross-sections while keep the local geometry property of its far away neighbouring area.

Activated by the work of Hyun et al. Hyun et al. [2003, 2005] but without using standard ellipses and displacement map, we will introduce generalized elliptic curves to approximate the cross section curves of human body, and present a simple and efficient method with small data storage to build human models in this chapter.

6.2 3D models defined with generalized ellipses

In order to build a 3D model, we draw some cross section curves which define the 3D model. Then we approximate the original cross sections with the generalized ellipses. After that, we can construct surface patches from these generalized ellipses with the following treatment.

As mentioned by Tokuyama, 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 Tokuyama [2000]. Here we use this skinning method to construct surface patches of human parts.

If a surface patch will be constructed from K generalized ellipses $\big(c_{xk}(v), c_{yk}(v), c_{zk}(v)\big)$ $(k = 0, 1, 2, 3, \ldots, K-1)$ where $c_{xk}(v)$, $c_{yk}(v)$ and $c_{zk}(v)$ are determined by Eq. (3.3), we use the following equation to describe the surface to be constructed

$$t(u, v) = \sum_{m=0}^{K-1} u^m f_{tm}(v) \qquad (t = x, y, z) \tag{6.1}$$

where $f_{tm}(v)(m = 0,1,2,3,\ldots,K-1)$ are unknown functions.

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 functions $f_{tm}(v)$ $(m = 0,1,2,3,\ldots,K-1)$ can be determined by solving the following linear algebraic equations

$$\begin{aligned} c_{tk}(v) &= \sum_{m=0}^{K-1} u_k^m f_{tm}(v) \\ &(k = 0,1,2,3,\ldots,K-1; t = x,y,z) \end{aligned} \tag{6.2}$$

Expanding Eq. (6.2) and rewriting it into the form of matrix, we obtain the following mathematical expression

$$\mathbf{R}_t(u_{km})\mathbf{F}_t(v) = \mathbf{C}_t(v) \tag{6.3}$$

where $\mathbf{R}_t(u_{km})$ is a $K \times K$ square matrix with the elements $u_{km} = u_k^m$, $\mathbf{F}_t(v) = \left[f_{t0}(v), f_{t1}(v), f_{t2}(v), \ldots, f_{tK-1}(v)\right]^T$ and $\mathbf{C}_t(v) = \left[c_{t0}(v), c_{t1}(v), c_{t2}(v), \ldots, c_{tK-1}(v)\right]^T$ are two vectors with K elements.

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. (6.2) by this inverse matrix, we obtain the unknown functions with the following equation

$$\mathbf{F}_t(v) = \mathbf{R}_t(u_{km})^{-1} \times \mathbf{C}_t(v) \tag{6.4}$$

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. 6.1, 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 tan-

gent continuity at $u = 1$ of the existing surface patch $\bar{t}(u,v)$, we obtain the mathematical expressions of the first partial derivative $\left[\frac{\partial \bar{t}(u,v)}{\partial u}\right]_{u=1} = \sum_{m=1}^{K-1} m\bar{f}_{tm}(v)$ 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. 6.1, respectively. Then the following tangential continuity constraint is added to Eq. 6.2.

$$\begin{aligned} & f_{t1}(v) = \sum_{m=1}^{K-1} m\bar{f}_{tm}(v) \\ & (t = x, y, z) \end{aligned} \tag{6.5}$$

Since one more linear algebraic equation is introduced, the unknown functions $f_{tm}(v)$ in Eq. 6.1 should be increased from $f_{tK-1}(v)$ to $f_{tK}(v)$ and all K in Eq. 6.1 and thereafter will be replaced by $(K+1)$.

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. 6.2

$$\begin{aligned} & \sum_{m=1}^{K-1} m f_{tm}(v) = \bar{\bar{f}}_{t1}(v) \\ & (t = x, y, z) \end{aligned} \tag{6.6}$$

and the unknown functions $f_{tm}(v)$ in Eq. 6.1 will be increased from $f_{tK}(v)$ to $f_{tK+1}(v)$ and all $(K+1)$ in Eq. 6.1 and thereafter will be replaced by $(K+2)$.

With the above constructed surface function, we create human torso, legs and arms from generalized ellipses as demonstrated in Fig. 6.1.

Figure 6.1: *Human torso, legs and arm created from generalized ellipses*

6.3 Surfaces defined with composite generalized elliptic segments

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.

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 sample 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 approximate 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 quite exactly, the accuracy improvement using the displacement map or other approaches

is not required. Therefore, the method proposed in this paper can significantly reduce the computer storage and obviously increase modelling efficiency.

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 $\upsilon = 0$, $\upsilon = \frac{2}{3}\pi$, $\upsilon = \frac{4}{3}\pi$ and $\upsilon = 2\pi$, respectively, we simply divide the generalized ellipse into $\upsilon = 0$ to $\upsilon = \frac{2}{3}\pi$, $\upsilon = \frac{2}{3}\pi$ to $\upsilon = \frac{4}{3}\pi$, and $\upsilon = \frac{4}{3}\pi$ to 2π.

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_{in}. 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 we only consider positional and tangen-

tial continuities. Of course, the treatment discussed here is also suitable for 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

$$\begin{aligned}
&\mathbf{X}_{ij}(v = v_{ij}) = \mathbf{X}_{ij+1}(v = v_{ij}) \\
&\left[\frac{\partial \mathbf{X}_{ij}}{\partial v}\right]_{v=v_{ij}} = \left[\frac{\partial \mathbf{X}_{ij+1}}{\partial v}\right]_{v=v_{ij}} \\
&(i = 0,1,2,3,\ldots,I; j = 0,1,2,3,\ldots,J-1)
\end{aligned} \tag{6.7}$$

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

$$\begin{aligned}
&\mathbf{X}_{iJ}(v = 2\pi) = \mathbf{X}_{i0}(v = 0) \\
&\left[\frac{\partial \mathbf{X}_{iJ}}{\partial v}\right]_{v=2\pi} = \left[\frac{\partial \mathbf{X}_{i0}}{\partial v}\right]_{v=0} \\
&(i = 0,1,2,3,\ldots,I)
\end{aligned} \tag{6.8}$$

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

$$\begin{aligned}
&\mathbf{S}_j(u,v) = \sum_{m=0}^{I} u^m \bar{\mathbf{X}}_{mj}(v) \\
&(j = 0,1,2,3,\ldots,I)
\end{aligned} \tag{6.9}$$

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.,

$$\begin{aligned}
&\sum_{m=0}^{I} u_i^m \bar{\mathbf{X}}_{mj}(v) = \mathbf{X}_{ij}(v) \\
&\sum_{m=0}^{I} u_i^m \bar{\mathbf{X}}_{mj+1}(v) = \mathbf{X}_{ij+1}(v) \\
&(i = 0, 1, 2, 3, \ldots, I)
\end{aligned} \tag{6.10}$$

Using the same treatment given in Section 6.2, we can determine $\bar{\mathbf{X}}_{mj}(v)$ and $\bar{\mathbf{X}}_{mj+1}(v)$ using the following matrix equations

$$\begin{aligned}
&\bar{\mathbf{X}}_j(v) = \mathbf{R}_j(u_{im})^{-1}\mathbf{X}_j(v) \\
&\bar{\mathbf{X}}_{j+1}(v) = \mathbf{R}_{j+1}(u_{im})^{-1}\mathbf{X}_{j+1}(v) \\
&(j = 0, 1, 2, 3, \ldots, J)
\end{aligned} \tag{6.11}$$

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

$$\mathbf{R} = \mathbf{R}_j\left(u_{im}\right)^{-1} = \mathbf{R}_{j+1}\left(u_{im}\right)^{-1} = R_{ij} \tag{6.12}$$

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

$$\begin{aligned}
&\bar{\mathbf{X}}_{lj}(v) = \sum_{i=0}^{I} R_{li}\mathbf{X}_{ij}(v) \\
&\bar{\mathbf{X}}_{lj+1}(v) = \sum_{i=0}^{I} R_{li}\mathbf{X}_{ij+1}(v) \\
&(l = 0, 1, 2, 3, \ldots, I)
\end{aligned} \tag{6.13}$$

Substituting Eq. (6.13) into (6.9), the position functions of the jth and

$(j+1)$th surface patches are found to be

$$\begin{aligned}\mathbf{S}_j(u,v) &= \sum_{m=0}^{I} u^m \sum_{i=0}^{I} R_{mi}\mathbf{X}_{ij}(v)\\ \mathbf{S}_{j+1}(u,v) &= \sum_{m=0}^{I} u^m \sum_{i=0}^{I} R_{mi}\mathbf{X}_{ij+1}(v)\end{aligned} \tag{6.14}$$

From Eq. (6.14), 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.(6.7) of $\mathbf{X}_{ij}(v)$ and $\mathbf{X}_{ij+1}(v)$ at their joint v_{ij}, we find

$$\begin{aligned}&\mathbf{S}_j(u,v_{ij}) = \mathbf{S}_{j+1}(u,v_{ij})\\ &\left[\frac{\partial \mathbf{S}_j(u,v)}{\partial v}\right]_{v=v_{ij}} = \left[\frac{\partial \mathbf{S}_{j+1}(u,v)}{\partial v}\right]_{v=v_{ij}}\end{aligned} \tag{6.15}$$

Equation (6.15) 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 6.2, we can also tackle the continuity of surface patches at $u=0$ and $u=1$ along u parameter direction.

Cross section curves of human hip are comparatively complicated. We use composite generalized elliptic segments to deal with them and build human hip as indicated in Fig. 5.

6.4 Assembly of human parts by surface blending

After successful construction of human parts including human torso, arms, legs and hip, the remaining task is how to assemble them together which maintains smooth transition between different human parts and ensures the surface in the transition region has the realistic appearance of the human part.

There are two approaches which can be used to achieve smoothing transiting body parts aim: one is surface blending Zhang & You [2004] and the other is local extrusion deformation You & Zhang [2003]. Another application of our algorithm in 6.2 and 6.3 lies in their ability to blending two surfaces with their surface boundary curves.

Here we take the connection between human torso and right arm as an example to explain the operation of surface blending. Firstly, we determine the boundary curves and boundary tangents of the surface which will smoothly connect human right arm and torso together. According to the position function of the surface patch at the leftmost end of the right arm, we can determine the right boundary curve and boundary tangent of the transition surface. In order to determine the left boundary curve and boundary tangent of the transition surface, we create a cross sectional curve on the human torso near the left end of the right arm and calculate the tangent of the human torso at this curve. Then, we construct the transition surface with these boundary curves and boundary tangents. We can change the size of the boundary tangents to change the shape of the transition surface and to make the transition surface look more realistically. If the manipulation of the boundary tangents does not achieve the goal of realism, we add one or more characteristic cross section curves between the two boundary curves and create the transition surface from two boundary curves, two boundary tangents and the characteristic curves in-between. With such a method, we connect two different parts together. The image on the left of Fig. 6.2 gives two human parts: arm and torso, and the image on the right shows these two human parts have been smoothly connected together. With the same treatment, we also connected the left arm to the torso.

For the connection between the human torso and legs, we first use a surface to close the bottom opening of the human torso. Then we determine boundary curves and tangents of the torso corresponding the right and left legs, respectively. And finally, the above method is employed to produce the transition surfaces between the three human parts. In Fig. 6, we gave a human model built with our proposed approach. The left image uses the grids to indicate different human parts and the connections between these

Figure 6.2: *Connection of different human parts*

human parts and the right image is the assembled human body. The approach presented in this paper greatly lowers the computer storage.

6.5 Conclusions

Human modelling can be performed with NURBS, polygon and subdivision. It is a complicated and time-consuming task. In order to simplify the modelling process and raise the modeling efficiency, we present a modelling method to build human models.

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

Figure 6.3: *Human model assembled from different human parts*

tic curves can maintain the required continuities between two adjacent patches. Finally, we discussed the assembly issue of human parts.

Unlike sweep-based human modelling, our method can represent cross sectional curves of human body very accurately. Due to this reason, the treatment to improve modelling realism using some approaches such as displacement map is not required.

Our proposed approach introduces a lot of mathematical operations for curve creation and surface generation which replace part of manual work in modelling process of human models. Therefore, it can obviously simplify the work of human modelling, reduce manual operations, lower the required computer storage and raise the efficiency of building human models.

Chapter 7

Conclusion and Future Work

Virtual humans are widely applied in various situations such as computer games, virtual reality and digital films. As pointed out by Thalmann NM and Thalmann D, in order to create realistic and believable virtual humans, three techniques should be developed. They are realistic appearance modelling, realistic, smooth and flexible motion modelling and realistic high-level behaviour modelling Magnenat-Thalmann & Thalmann [2005].

This research takes the easiness and efficiency advantages of primitives in representing rough 3D models. It uses primitives to quickly obtain an initial 3D mesh. Then, these primitives are deformed into required shapes by fitting their silhouette contours to 2D sketches. This research also invented a free-form patch generating method to take a network of sketched curved as the input and then solve an ODE to get each of the vertex's position information within the span.

Another contribution of this research is the proposition of a surface modeling technique customized for cross-section contours. Start from using trigonometry series to interpolate the cross-section contours, we got the parametric representation of the cross-section curves. Then, this research develops an algorithm to construct human body parts from the cross sectional curves. Finally, with a they are smoothly connected together to create a complete 3D model.

One possible future work may investigate the potential of the proposed

sketch based deformation technology in changing the way animators animating the characters altogether. In the future, 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 drawing a key frame and let the computer deform the shape accordingly.

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 $N+1$,

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

For node $N+2$,

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

(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 node i in 3, 4, ... , $N-3$, $N+3$, $N+4$, ... , $2N-3$,

$$(6k_b + 2k_s\triangle u^2)d_i + k_b d_{i-2} + k_b d_{i+2}$$
$$-(4k_b + k_s\triangle u^2)d_{i-1} - (4k_b + k_s\triangle u^2)d_{i+1} = 0 \tag{A4}$$

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,

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

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

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

For node N,

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

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

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

(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 $\frac{N}{2}-2$,

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

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

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

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

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

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

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

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

$$(6k_b+2k_s\triangle u^2)d_{3N/2+2}+k_bd_{3N/2+4}$$
$$-(4k_b+k_s\triangle u^2)d_{3N/2+1}-(4k_b+k_s\triangle u^2)d_{3N/2+3}$$
$$=-k_bC_{4j} \tag{B2}$$

For node in $3, 4, \ldots, N/2-3, N/2+3, N/2+4, \ldots, N-4, N-3, N+3, N+4, \ldots, 3N/2-4, 3N/2-3, 3N/2+3, 3N/2+4, \ldots, 2N-4, 2N-3$, 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,

$$\begin{aligned}(7k_b + 2k_s\triangle u^2)d_2 - (4k_b + k_s\triangle u^2)d_3 + k_b d_4 \\ = 2k_b T_{1j}\triangle u + (4k_b + k_s\triangle u^2)C_{1j}\end{aligned}$$

For node 3,

$$\begin{aligned}(6k_b + 2k_s\triangle u^2)d_3 - (4k_b + k_s\triangle u^2)d_2 + k_b d_5 \\ -(4k_b + k_s\triangle u^2)d_4 = -k_b C_{1j}\end{aligned} \tag{C1}$$

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

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

For node N,

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

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 node in $4, 5, \ldots, N-2, N+4, N+5, \ldots, 2N-2$, the formulates are 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

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

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,

$$(6k_b + 2k_s\triangle u^2)d_3 + k_bd_5 - (4k_b + k_s\triangle u^2)d_2$$
$$-(4k_b + k_s\triangle u^2)d_4 = \triangle u^4[f_{5,3} + \frac{L_j}{L}(f_{6,3} - f_{5,3})] - k_bP_1 \tag{D1}$$

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

For node $N-1$,

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

For node N,

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

(D2)

For node in 4, 5, … , $N-3$, $N-2$, the formulate is as follow

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

Bibliography

B. Allen, et al. (2002). 'Articulated body deformation from range scan data'. In *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 612–619. ACM.

B. Allen, et al. (2003). 'The space of human body shapes: reconstruction and parameterization from range scans'. In *ACM transactions on graphics (TOG)*, vol. 22, pp. 587–594. ACM.

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 (????). 'Autodesk Character Generator'.

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.

A. Bernhardt, et al. (2008). 'Matisse: Painting 2D regions for modeling free-form shapes'. In *SBM'08-Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pp. 57–64. Eurographics Association.

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.

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.

L. Chevalier, et al. (2003). 'Segmentation and superquadric modeling of 3D objects' .

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.

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

Facebook (2019). 'Quill'.

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

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'.

Gravity Sketch (2017). 'Gravity Sketch'.

Z. Guo & K. Cheong Wong (2005). 'Skinning with deformable chunks'. In *Computer Graphics Forum*, vol. 24, pp. 373–381. Wiley Online Library.

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.

P. Joshi & N. A. Carr (2008). 'Repoussé: Automatic Inflation of 2D Artwork.'. In *SBM*, pp. 49–55.

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 (2006a). 'SmoothSketch: 3D Free-form Shapes from Complex Sketches'. *ACM Trans. Graph.* **25**(3):589–598.

O. A. Karpenko & J. F. Hughes (2006b). '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.

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

J. Lee, et al. (2006). 'Realistic human hand deformation'. *Computer Animation and Virtual Worlds* **17**(3-4):479–489.

W.-S. Lee & A. Soon (2006). 'Facial shape and 3D skin'. *Computer Animation and Virtual Worlds* **17**(3-4):501–512.

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

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

A. Mohr & M. Gleicher (2003). 'Building efficient, accurate character skins from examples'. In *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 562–568. ACM.

Mozilla (2019). 'Mozilla 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. (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'.

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.

P. Sand, et al. (2003). 'Continuous capture of skin deformation'. In *ACM Transactions on Graphics (TOG)*, vol. 22, pp. 578–586. ACM.

F. Scheepers, et al. (1997). 'Anatomy-based modeling of the human musculature'. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 163–172. ACM Press/Addison-Wesley Publishing Co.

H. Seo & N. Magnenat-Thalmann (2004). 'An example-based approach to human body manipulation'. *Graphical Models* **66**(1):1–23.

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

M. Simmons, et al. (2002). 'Model-based reconstruction for creature animation'. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 139–146. ACM.

K. Singh & E. Kokkevis (2000). 'Skinning Characters using Surface Oriented Free-Form Deformations.'. In *Graphics interface*, vol. 2000, pp. 35–42.

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

K. Takayama, et al. (2013b). 'Sketch-based Generation and Editing of Quad Meshes'. *ACM Trans. Graph.* **32**(4):97:1–97:8.

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.

K. Venkataraman, et al. (2005). 'A kinematic-variational model for animating skin with wrinkles'. *Computers & Graphics* **29**(5):756–770.

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.

X. Yang, et al. (2006). 'Curve skeleton skinning for human and creature characters'. *Computer Animation and Virtual Worlds* **17**(3-4):281–292.

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 & J. J. Zhang (2003). 'Fast generation of 3-D deformable moving

surfaces'. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **33**(4):616–625.

J. J. Zhang & L. You (2004). 'Fast surface modelling using a 6th order PDE'. In *Computer Graphics Forum*, vol. 23, pp. 311–320. Wiley Online Library.

J. Zimmermann, et al. (2007). 'SilSketch: automated sketch-based editing of surface meshes'. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pp. 23–30. ACM.

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