Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.

Vol. **xx**, No. x, pp. 1-32
x 2023

Character Modelling with Sketches and ODE-Based Shape Creation

Ouwen Li[1], Haibin Fu[1,*], Shaojun Bian[1], Xiaosong Yang[1], Xiaogang Jin[2], Andres Iglesias[3,4], Algirdas Noreika[5], Lihua You[1] and Jian Jun Zhang[1]

[1] *National Centre for Computer Animation, Bournemouth University, United Kingdom*
[2] *State Key Lab of CAD&CG, Zhejiang University, China*
[3] *Department of Applied Mathematics and Computational Sciences, University of Cantabria, Spain*
[4] *Department of Information Science, Toho University, Japan*
[5] *Indeform Ltd, Lithuania*

Received 9 November 2022; Accepted (in revised version) 2 March 2023

Abstract. Character models have enormous applications in industry. Efficient creation of detailed character models is an important topic. This paper proposes a new and easy-to-use technique to quickly create detailed character models from sketches. The proposed technique consists of two main components: primitive deformer and shape generators. With this technique, 2D silhouette contours of a character model are drawn or extracted from an image or sketch. Then, proper geometric primitives are selected and aligned with the corresponding 2D silhouette contours. After that, a primitive deformer is used to create a base mesh and three shape generators are used to add 3D details to the base mesh. The primitive deformer and three shape generators are developed from ODE-driven deformations. The primitive deformer deforms the aligned geometric primitives to exactly match the 2D silhouette contours in one view plane and obtains a base mesh of a character model consisting of deformed primitives. The shape generators are used to add 3D details to the base mesh by creating local 3D models. The experimental results demonstrate that the new technique can quickly create detailed 3D character models from sketches with few manual operations. The new technique is physics-based and easy to learn and use.

AMS subject classifications: 15A12, 65F10, 65F15

Key words: Sketch-based character modelling, shape generators, ordinary differential equations, finite difference solution.

*Corresponding author. *Email address:* hfu@bournemouth.ac.uk (H. Fu),

1. Introduction

Character models are widely applied in virtual reality, computer animation, and video games etc. How to create detailed and realistic character models efficiently with fewer manual operations is of practical significance.

Character modelling can be divided into geometric modelling, physics-based modelling, and sketch-based modelling. Geometric modelling studies methods and algorithms without involving physics for the mathematical description of shapes. Physics-based modelling investigates how to create motions and deformations of objects with the laws of physics. Sketch-based modelling explores techniques and interfaces of drawing two-dimensional (2D) strokes of three-dimensional (3D) objects and converting the drawn 2D strokes into 3D models automatically. It also includes drawing 3D curve networks and surfacing 3D curve networks to obtain 3D models.

Geometric modelling can be further divided into polygon [36], subdivision [32], and Non-uniform rational basis spline (NURBS) [40] modelling. They are computationally more efficient than physics-based modelling and can create more detailed shapes than sketch-based modelling. However, they create less realistic models, are difficult to learn and use, and require a lot of time and manual operations.

Physics-based modelling approaches [35] such as the finite element method [26] and mass-spring systems [20] consider underlying physics and can create more realistic shapes of 3D models. However, they involve heavy numerical calculations. Ordinary differential equation (ODE) based modelling [47] is physics-based and has a potential to create more realistic shapes. It avoids heavy calculations of existing physics-based modelling approaches.

In comparison with geometric modelling and physics-based modelling, sketch-based modelling [37] is more efficient, straightforward and involves much fewer manual operations. Sketch-based modelling method is a way that enables the user to construct 3D models using a sketch interface. It can be divided into template-based one, direct creation, and primitive-based generation. The biggest problem with sketch-based modelling is its incapability in creating detailed character models.

As discussed above, all state-of-the-art existing methods have their advantages and disadvantages. Sketch-based modelling can significantly reduce manual operations in geometric modelling, ODE-based modelling can avoid heavy numerical calculations in physics-based modelling and tackle the incapacity of sketch-based modelling in creating detailed 3D models. Based on these considerations, this paper will combine ODE with sketch-based modelling to develop efficient shape generators, and use them to create local shapes from user's drawn sketches for adding 3D details to coarse base models obtained from the primitive deformer proposed in [30, 31]

With this method, we use primitives to quickly obtain initial 3D meshes. Then, we use the primitive deformer [30, 31] to deform these primitives to the required shapes by fitting their silhouette contours to 2D sketches. Next, these deformed primitives are smoothly connected together to create a coarse base model. Finally, shape generators proposed in this paper are used to add 3D details to the coarse base model.

By integrating sketch-based modelling, primitive deformer, and shape generators, the proposed approach provides a simple, easy to learn and use, and efficient modelling solution to quickly create detailed and realistic 3D models from sketches and primitives. The contributions made in this paper are:

1) We combine ODE-driven shape deformations with sketch-based modelling to create detailed character models easily and quickly from sketches.

2) We develop three different shape generators to add 3D details to coarse models. Our proposed sketch-guided and ODE-driven shape generators can create a new local shape to match user's drawn sketches in different views.

3) We implement a modelling system from our previously developed primitive deformer and the shape generators developed in this paper. With our developed system, detailed character models can be created interactively in real time.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. An overview of the proposed approach is presented in Section 3. For the sake of completeness, our previously developed primitive deformer is briefly introduced in Section 4, the shape generators are investigated in Section 5, and the experiments and comparisons are given in Section 6. Finally, conclusions and future work are presented in Section 7.

2. Related work

Since this paper aims to integrate ODE and sketch-based modelling to develop a new technique of quickly creating detailed character models from sketches, we briefly review existing work in sketch-based modelling and ODE-based geometry processing in this section.

2.1. Sketch-based modelling

Due to its simplicity, easiness, and few manual operations, sketch-based modelling (SBM) has attracted the attention of many researchers, and various sketch-based modelling approaches have been developed. Comprehensive surveys of these methods can be found in [5, 12, 13, 16, 25, 37]. According to [25], sketch-based modelling systems can be divided into single-view systems, multi-view systems, and curve network-based systems.

2.1.1. Single-view systems

Single-view systems use 2D sketches from a single viewpoint to create 3D models. According to whether template models and primitives are used, they can be classified into template-based deformation, direct creation, and primitive-based generation.

Template-based deformation method is proposed in [27] to find precise correspondences and deform 3D models to fit user's drawn sketches. By combining skeleton-based skinning with mesh editing, an efficient approach is presented in [24] to quickly deform a 3D template model to fit the user's drawn sketches. The template-based global modelling investigated in [48] first deforms a template model to fit user's drawn silhouette contour and then adds more sketches to change the template model into a new model.

Besides template-based modelling, most of sketch-based modelling approaches create new 3D models directly from 2D sketches. Among them, many modelling approaches such as Teddy [22] and its various extensions [34] follow a sketch-rotate-sketch workflow, which requires users to sketch from a large number of different views. Besides the limitation that only coarse shapes can be created, the following limitations of these approaches have been documented in the literature [19]:

1) Users cannot match their input strokes to a guidance image due to view changes.

2) How to find a good view for a stroke is often difficult and time-consuming.

Direct creation method is investigated by most research activities [2, 8, 11, 15, 17, 19, 22, 23, 29, 34, 39, 43–45, 50] in comparison with template-based deformation and primitive-based generation. According to [5], surfaces can be created from sketches by extruding, inflating, and revolving strokes.

The method of extruding strokes uses one closed stroke and another silhouette stroke depicting the silhouette of an extruded surface. The closed stroke is translated along the silhouette stroke to create an extruded surface [22]. The method of inflating strokes was investigated in [22, 23, 29, 34]. Igarashi *et al.* [22] developed a system to interactively draw 2D strokes and inflate the 2D strokes to create rotund 3D models. Nealen *et al.* [34] introduced smooth and sharp curves to enrich editing operations for inflated models. Karpenko and Hughes [23] proposed a SmoothSketch system to address the problem of T-junctions and cusps. Li *et al.* [29] investigated the method of creating a freeform surface from sketched patch boundary and a small number of strokes representing the major bending directions of the shape. The method of revolving strokes creates a surface by moving a stroke around an axis in a circular order [8].

Besides the methods of extruding, inflating, and revolving strokes, some other methods have been developed. The interface for clothing virtual characters developed in [44] identifies two types of contour lines: silhouettes that do not cross a character body and borderlines that cross the body, converts the 2D sketches into 3D curves through calculating the distance of the points on 2D sketches to the body, and creates garment surfaces from converted 3D curves. The single-view sketch-based modelling system developed in [2] sweeps one sketched construction line along another construction line to create a surface. The method proposed in [4] constructs a 3D character model from a 2D cartoon drawing and a user's provided 3D skeleton. And SecondSkin proposed in [39] classifies 2D sketches into different 3D curve-types and constructs layered 3D models around a character body from 3D curve-types.

In recent years, machine learning has been introduced into sketch-based modelling. Xiang *et al.* [45] proposed an end-to-end learning framework with a differentiable renderer to construct 3D surfaces from sketches. Zhang *et al.* [50] presented a view-aware architecture for 3D modelling from single free-hand sketches, which decomposes sketch image features into a shape space and a view space, involves view-related information in the view space by training a view encoder and a view decoder, and preserves shape quality with a shape discriminator. Deng *et al.* [15] developed a sketch-based system to draw surface boundary and contour lines and infer the underlying freeform roof-like shape represented as a planar quadrilateral mesh in real-time with a deep neural network.

Primitive-based generation methods use primitives and edit them to fit sketches. In order to overcome the limitations caused by the sketch-rotate-sketch workflow, Gingold *et al.* [19] proposed a new approach, which first places geometric primitives on a 2D sketch and then applies some annotations to the geometric primitives to create 3D models. Since all manual operations are applied in a fixed view, this approach also creates coarse models only. Shtof *et al.* [43] investigated a snapping approach to determine the positions and core parameters of several geometric primitives. Chen *et al.* [11] developed a tool to create a cylinder from three strokes. The 2D profile is described by the first two strokes. The axis along which the 2D profile sweeps is defined by the last stroke. The work is applicable to man-made objects only.

2.1.2. Multi-view systems

Multi-view systems are popularly applied in engineering design. They use a standard 3-view orthographic drawing to create a 3D model, which avoids the difficulty of finding good views [33]. Besides their applications in engineering design, multi-view systems have also been used to create 3D man-made models with two or three silhouettes from front, side, or top views [41] and organic models [28,48]. Levi and Gotsman [28] developed an ArtiSketch system, which uses one silhouette contour sketch for each part of the 3D models in one of animation frames, finds the correspondence between silhouette contour sketches from different frames, and automatically converts the silhouette contour sketches to articulated 3D models. You *et al.* [48] used three orthotropic silhouettes of the whole model and its part models to create 3D character models. The main limitation for 3-view orthographic drawing is its unsuitability in creating 3D organic models where many polygon facets do not share common axes. In addition, multi-view systems do not follow a coarse-to-fine workflow of organic modelling.

2.1.3. Curve network-based systems

Curve network-based systems allow designers to draw a network of 3D curves, and create 3D models from the 3D curves. The system ILoveSketch [3] provides a virtual sketchbook to support tearing, peeling, panning, zooming, and rotation interactions and five different 3D curve sketching methods. The 3D curves drawn by ILoveSketch

are automatically converted into an acceptable, unambiguous patch representation of 3D models in [1]. JustDrawIt provides an interface consisting of 2D drawing, 3D curve drawing, and 3D surfacing where 3D surfacing creates 3D models from sketches [21]. In order to construct 3D models from curve networks, a new algorithm of finding cycles to bound surface patches is proposed in [51], a flow complex of capturing input topology and geometry is employed in [42] to develop a new method, which interpolates the input 3D curves to create intersection-free 3D triangular shapes, and a new approach of surfacing 3D curve networks is developed in [38] by automatically aligning curvature lines of reconstructed surface with network flow lines.

Multi-view systems require more manual operations than single-view systems to draw sketches in three orthotropic views or a series of different views and additional algorithms to find the correspondence among sketches. In contrast, single-view systems provide less shape information but have advantages of simplicity, ease-to-use, and fewest manual operations. Among template-based deformation, direct creation, and primitive-based generation, template-based deformation can create detailed 3D models, but requires a good template model. In comparison with surface creation by extruding, inflating, and revolving strokes, primitive-based generation is more straightforward and efficient in creating coarse 3D models. However, how to deform 3D primitives and add 3D details to coarse models to obtain more detailed shapes is an unsolved issue. This paper will address this issue.

2.2. ODE-based geometric processing

The solution to a vector-valued ordinary differential equation represents a 3D curve. Sweeping the curve along one or two boundaries plus other constraints such as tangent and curvature continuities leads to a 3D surface, which can be used to define 3D models [47] or describe their shape changes [6]. Such a method of creating, manipulating, and animating 3D models is called ODE-based geometry processing. Since ODEs, such as the governing equation for bending of elastic beams, describe underlying physics, ODE-based geometry processing is physics-based. In existing literature, various ODE-based geometry processing methods have been developed.

A fourth-order ordinary differential equation was introduced in [47] to develop an analytical solution, which is combined with C^1 continuous boundary constraints to develop C^1 continuous sweeping surfaces. The work was extended to create C^2 continuous sweeping surfaces by using C^2 continuous boundary constraints to determine the unknown constants involved in the analytical solution to a sixth-order ordinary differential equation [7]. Here, C^1 and C^2 continuous boundary constraints mean the first and second partial derivatives at the interface between two adjacent sweeping surfaces are identical, respectively. In order to describe different types of deformations, a first-order ordinary differential equation was introduced to create axial tensile and compressive deformations and a fourth-order ordinary differential equation was used to create lateral bending deformations [49]. A dynamic fourth-order ordinary differential equation was used to simulate dynamic skin deformations for character an-

imation [10]. Recently, a fourth-order ordinary differential equation for beam bending and a second-order ordinary differential equation representing Newton's second law of motion were combined together to reconstruct dynamic 3D models deformed with position based dynamics [18].

As discussed above, ODE-based geometry processing has been well investigated in the existing literature. Except for the work given in [30, 31] that integrates ODE-based geometry processing and sketch-based modelling to create coarse base models, all other ODE-based geometry processing methods do not involve sketch-based modelling. Therefore, how to combine ODE-based geometry processing with sketch-based modelling to create more detailed 3D models from primitives easily and efficiently has not been investigated. This paper will tackle this problem.

3. Overview of proposed approach

The proposed approach consists of two main parts: ODE-driven primitive deformer and shape generators. ODE-driven primitive deformer aligns primitives with the user's generated 2D silhouette contours to obtain an initial 3D mesh, deforms the primitives to exactly match the contours, and blends the deformed primitives to create a coarse base mesh, which has been investigated in [30, 31]. After that, shape generators add new local shapes to the coarse base mesh to obtain a final detailed model using three different algorithms from: 1) two open silhouette contours in two different view planes, 2) one open and one closed silhouette contour in two different view planes, and 3) two open and one closed silhouette contours in three different view planes.

Taking the creation of a 3D female model as an example, the modelling process of the proposed approach is shown in Fig. 1. First, the silhouette contours of the 2D female

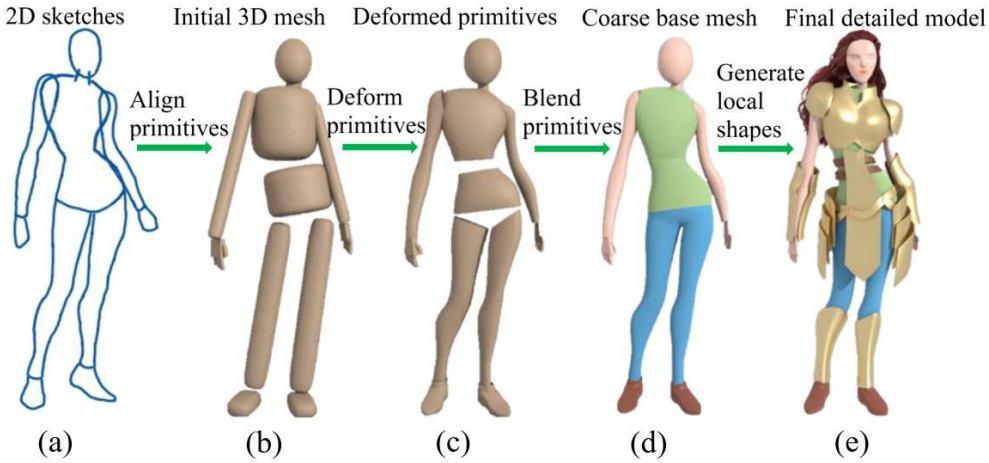


Figure 1: Creation of a 3D female model using the proposed approach, (a) 2D sketches (silhouette contours), (b) initial 3D mesh without primitive deformations, (c) deforming the primitives to match the generated 2D silhouette contours, (d) coarse base mesh by adding blending surfaces to smoothly connect the deformed primitives, (e) final detailed model through local shape creation with the developed shape generators.

model shown in Fig. 1(a) are generated by drawing silhouette contours directly or inputting selected sketches. Then, an initial 3D mesh shown in Fig. 1(b) is obtained by selecting proper primitives (ellipsoid for head, cylinders for neck, arms, torso, and legs, and boxes for feet) and automatically placing them to align with the corresponding silhouette contours through purely geometric transformations, i.e., translation, scale, and rotation. Since the silhouette contours of the primitives do not match the generated 2D silhouette contours, the primitive deformer developed from sketch-guided and ODE-driven deformations described in [30, 31] is used to deform the primitives to exactly match the corresponding 2D silhouette contours as depicted in Fig. 1(c). After that, these deformed primitives are smoothly connected with the aid of ODE-based surface blending [46] to create a coarse 3D base mesh, as shown in Fig. 1(d).

Once a 3D base mesh model is obtained with the method proposed in [30, 31], the shape generators described in Section 5 are employed to add 3D details to the coarse base mesh. These 3D details include the detailed face shape, armors such as chest vest, barcer, leg guard, thin sheet armors, and hair whose creation will be explained below. The final detailed 3D female model is shown in Fig. 1(e).

In order to create the female face model, users first draw sketches to define the female face shown in Fig. 19(b). The creation of the 3D mouth model from the mouth sketch in Fig. 19(b) is shown in Fig. 14. The creation of the eye socket from the sketch in Fig. 19(b) is shown in Fig. 16(b). All other parts of the female face can be created with the method of creating the surface defined by four curves $\boldsymbol{C_1}, \boldsymbol{C_6}, \boldsymbol{C_9}$, and $\boldsymbol{C_5}$ shown in Fig. 15. The male face model shown in Fig. 20(b) can be created with the same methods.

The chest vest, the male and female barcers and leg guards, and the armors on the male and female shoulder are created with the methods shown in Figs. 5, 17, and 12, respectively. All other thin sheet armors are created with the method shown in Fig. 15. Each of the hairs on the male and female head shown in Fig. 20 is represented with an ODE-based curve.

4. Primitive deformer

As shown in Fig. 1(b), the placed primitives cannot match the generated 2D silhouette contours. To tackle this problem, a primitive deformer has been developed in [30, 31]. For the sake of completeness, this section briefly introduces the primitive deformer developed in [30, 31].

After 3D primitives have been placed and aligned with the user's 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 Fig. 2 to demonstrate this. Fig. 2(a) depicts the torso part of the female model shown in Fig. 1, which is represented by a cylinder. The blue 2D silhouette contour to be matched is also shown in Fig. 2(a). Fig. 2(b) shows the cylinder is deformed with the algorithm proposed below to match the generated 2D silhouette contour exactly.

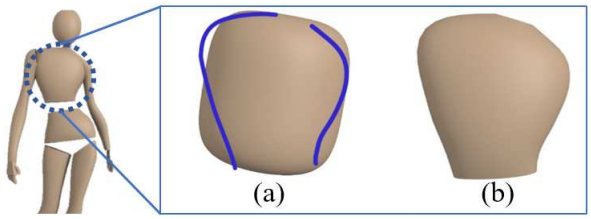


Figure 2: Primitive deformer: (a) the female torso represented by a cylinder and the user's generated 2D silhouette contour in blue, (b) the deformed shape of the cylinder.

Physics-based modelling can create more realistic deformations than geometric modelling. As discussed in [30, 31], the elastic energy considering local stretching and bending of two-manifold surfaces [9] can be used to create physics-based deformations. From the elastic energy, a fourth-order ordinary differential equation derived in [30, 31], which has the form below, is used to develop the primitive deformer

$$k_b \frac{d^4 \boldsymbol{X}}{du^4} - k_s \frac{d^2 \boldsymbol{X}}{du^2} = 0, \tag{4.1}$$

where k_b and k_s are the stiffness parameters controlling the resistance to stretching and bending, u is a parametric variable, and $\boldsymbol{X}$ is a vector-valued function used to describe the deformation of a curve which has three components x, y and z.

The closed form solution to Eq. (4.1) is obtainable. Since the closed form solution is a vector-valued function involving four vector-valued unknown constants only, it cannot represent many user's drawn sketches with an acceptable accuracy. In contrast, numerical methods can represent user's drawn sketches with user's required accuracy by increasing the number of discrete nodes. Due to this advantage of the numerical methods over the closed form solution, this paper uses the finite difference method to solve Eq. (4.1).

For a typical node i shown in Fig. 3 and using the central finite difference approximations of the second- and fourth-order derivatives in [10], the following finite difference equation at a typical node i can be obtained as:

$$\begin{aligned} &\left(6k_b - 2k_s\Delta u^2\right)\boldsymbol{X}_i + k_b\boldsymbol{X}_{i-2} + k_b\boldsymbol{X}_{i+2} \\ &\quad - \left(4k_b + k_s\Delta u^2\right)\boldsymbol{X}_{i-1} - \left(4k_b + k_s\Delta u^2\right)\boldsymbol{X}_{i+1} = 0, \end{aligned} \tag{4.2}$$

where Δu is the parametric distance between two adjacent nodes, and $\boldsymbol{X}_i$ is the displacement of the node i, which has three components x_i, y_i, and z_i.

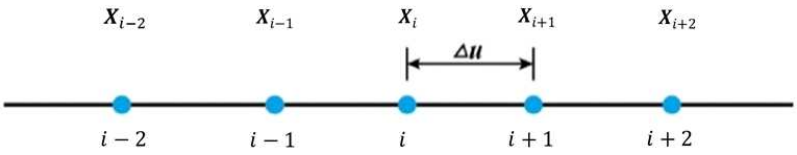


Figure 3: Typical node i for the finite difference approximations of derivatives.

As shown in Fig. 4, the silhouette contours of a 3D primitive are $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ and the user's drawn sketches are $\boldsymbol{c}'_1$ and $\boldsymbol{c}'_2$, which are defined by the points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}'_{1j}$ and $\boldsymbol{c}'_{2j}, j = 1, 2, \ldots, 6$, respectively. The question is to deform the primitive with the silhouette contours $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ into a shape with its silhouette contours coinciding with the user's drawn sketches $\boldsymbol{c}'_1$ and $\boldsymbol{c}'_2$. This can be achieved by deforming the cross-section curve $\boldsymbol{c}$ passing through the points $\boldsymbol{c}_{1j}$ and $\boldsymbol{c}_{2j}$ into a new cross-section curve $\overline{\boldsymbol{c}}$ passing through the points $\boldsymbol{c}'_{1j}$, and $\boldsymbol{c}'_{2j}$ and then obtaining the deformed primitive from the new cross-section curves. In what follows, we take the cross-section curves passing through the points $\boldsymbol{c}_{13}$ and $\boldsymbol{c}_{23}$ shown in Fig. 4(b) as an example to demonstrate this.

For the dotted curve shown in Fig. 4(b), its parametric domain is between 0 and 1. We uniformly discretized this domain into $2N$ intervals and get $\Delta u = 1/(2N)$ and node $i,\ i = 0, 1, 2, \ldots, 2N-1, 2N$, with $\boldsymbol{X}_{2N} = \boldsymbol{X}_0$.

With the above discretization, the finite difference equation for the inner nodes $i = 1, 2, \ldots, N-1, N+1, \ldots, 2N-1$ can be formulated with Eq. (4.2). In order to solve these finite difference equations, the boundary conditions at the finite difference nodes 0 and N must be applied. These boundary conditions can be obtained below.

From the four points $\boldsymbol{c}'_{13}, \boldsymbol{c}'_{23}, \boldsymbol{c}_{13}$ and $\boldsymbol{c}_{23}$ on the two user's generated silhouettes $\boldsymbol{c}'_1$ and $\boldsymbol{c}'_2$ and the two silhouettes $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ of a primitive shown in Fig. 5(a), we can obtain $\boldsymbol{X}_0 = \boldsymbol{c}'_{13} - \boldsymbol{c}_{13}$ and $\boldsymbol{X}_N = \boldsymbol{c}'_{23} - \boldsymbol{c}_{23}$ as shown in Fig. 4(b).

When we write the finite difference equations for the nodes $1, 2, 2N-2$ and $2N-1$, the node 0 will be involved. Since we have the boundary condition $\boldsymbol{X}_0 = \boldsymbol{c}'_{13} - \boldsymbol{c}_{13}$, the

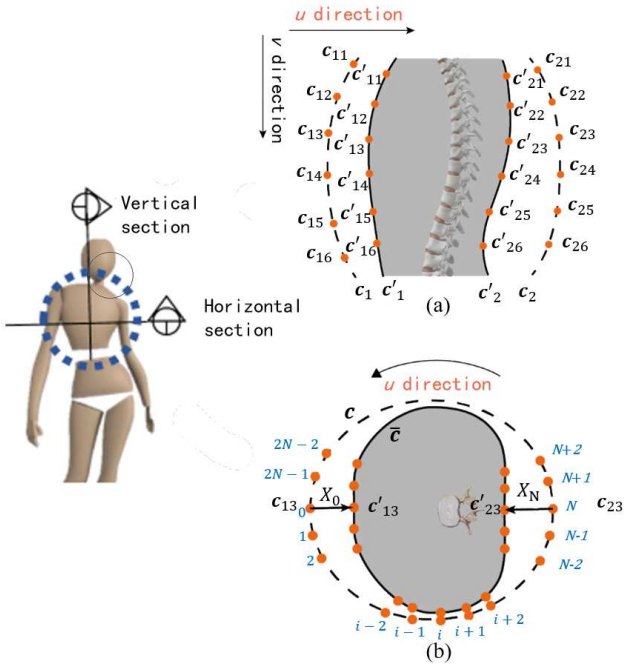


Figure 4: Silhouettes $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ of the primitive, sketched silhouettes $\boldsymbol{c}'_1$ and $\boldsymbol{c}'_2$, and finite difference nodes $0, 1, 2, \ldots, 2N-2, 2N-1$.

finite difference equations at these points can be derived from Eq. (4.2). Substituting the boundary condition $\boldsymbol{X}_0 = \boldsymbol{c}'_{13} - \boldsymbol{c}_{13}$ into these equations, we obtain the finite difference equations for the nodes $1, 2, 2N-2$ and $2N-1$, and present them in Appendix A.

When we write the finite difference equations for the nodes $N-2, N-1, N+1$ and $N+2$, the node N will be involved. Since we have $\boldsymbol{X}_N = \boldsymbol{c}'_{23} - \boldsymbol{c}_{23}$. Similarly, the finite difference equations at these points can be derived from Eq. (4.2). Substituting $\boldsymbol{X}_N = \boldsymbol{c}'_{23} - \boldsymbol{c}_{23}$ into these equations, we obtain the finite difference equations for the nodes $N-2$, $N-1$, $N+1$ and $N+2$, and present them in Appendix A as well.

For all other nodes $i = 3, 4, 5, \ldots, N-3$, and $N+3, N+4, \ldots, 2N-3$, the finite difference equations are the same as Eq. (4.2). For these nodes, there are $2N-10$ finite difference equations. Plus the 8 finite difference equations at node $1, 2, 2N-2$ and $2N-1, N-2, N-1, N+1$ and $N+2$ presented in Appendix A, we get $2N-2$ linear algebra equations, which can be solved to determine the $2N-2$ unknown constants $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{N-1}, \boldsymbol{X}_{N+1}, \boldsymbol{X}_{N+2}, \boldsymbol{X}_{N+3}, \ldots, \boldsymbol{X}_{2N-2}$, and $\boldsymbol{X}_{2N-1}$. Adding the $\boldsymbol{X}_i$, $i = 1, 2, 3, \ldots, 2N-1$, to the original curve $\boldsymbol{c}$, we obtain the deformed curve $\overline{\boldsymbol{c}}$, and depict it as a solid curve in Fig. 4(b). Repeating the above operations for all other points on the two silhouette contours, we obtain all deformed curves. These curves describe a new 3D deformed shape.

With the above primitive deformer, we deform the primitives of the 3D base mesh shown in Fig. 1(b) to obtain the deformed primitives shown in Fig. 1(c). It can be seen from Fig. 1(c) that the deformed primitives match the generated 2D silhouette contours exactly. Besides using the developed primitive deformer to deform primitives to match user's generated sketches, it can be used to modify cross-section shapes from generated cross-section contours. With this method, the influence range of a 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 Eq. (4.2) 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 Eq. (4.2) to reduce two more unknown constants.

Taking the vest created in Section 5.3 and shown in Fig. 5(a) as an example, one original cross-section curve of the vest is an ellipse highlighted in blue in Fig. 5(b), 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 Fig. 5(c). The cross-section shape within the boundary curves is modified and shown in Fig. 5(d). It can be observed that a more realistic shape is created.

The disconnected primitives shown in Fig. 1(c) can be smoothly connected by using the surface blending method proposed in [46] to generate a smooth transition surface between two adjacent deformed primitives as detailed below.

For a smooth transition surface generation, boundary curves and boundary tangents are first determined. Boundary curves can be obtained by interactively drawing one curve on each of two disconnected primitives or using one plane to intersect each of two disconnected primitives to obtain two boundary curves of the transition surface. Boundary tangents can be obtained by calculating the first derivatives of the disconnected primitives on the boundary curves in the direction perpendicular to the boundary curves. With the two boundary curves and the tangents at the two boundary curves as blending boundary constraints, the ODE-based surface blending method proposed in [46] is used to create the smooth transition surface that smoothly connects two primitives together. Fig. 6 gives such an example where the two disconnected primitives shown in Fig. 6(a) are smoothly connected with a smooth transition surface as shown in Fig. 6(b).

With the above surface blending method, the transition surfaces between the deformed primitives shown in Fig. 1(c) are created. A smooth coarse 3D base mesh is obtained in Fig. 1(d).

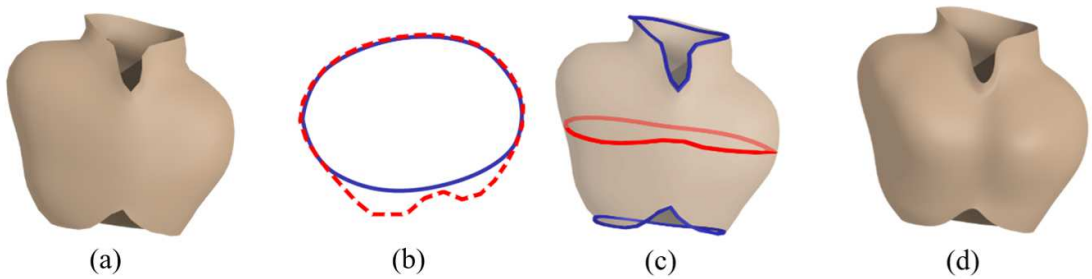


Figure 5: Cross-section shape modifications by cross-section contours: (a) vest, (b) the origin cross-section contour (in blue) and a modified cross-section contour (in red), (c) the modified cross-section is in place, (d) 3D shape after cross-section contour modifications.

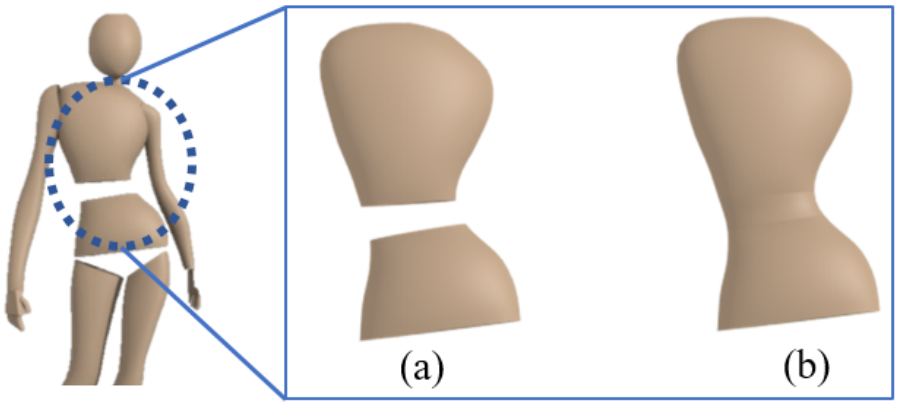


Figure 6: Creation of the smooth transition surface between primitives where (a) shows disconnected primitives and (b) shows transition surface.

5. Shape generators

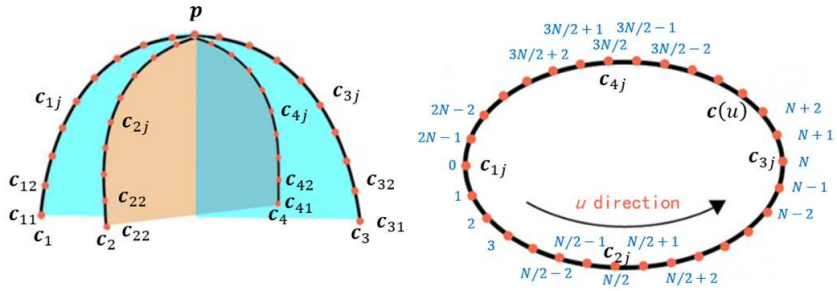
The base mesh generated from the deformed primitives only represents a coarse 3D model. Therefore, we need to add 3D details to the coarse base model to generate a fine detailed 3D model. When using polygon modelling method to manipulate 3D models, the vertices in deformation regions are pulled or pushed to create local shapes. Such local shapes can be also created by drawing their silhouette contour sketches and generating 3D local shapes from the drawn sketches. In this section, we will develop ODE-driven shape generators to create such local shapes from the user's drawn silhouette contour sketches. These sketches can be divided into the following three types: (1) two open sketches, (2) one open and one closed sketch, and (3) two open and one closed sketches. Among the three types of sketches, the third one provides the most complete silhouette contour information but requires more manual operations to draw the two open sketches and the closed sketch. When the shape of a local 3D model can be defined by two open silhouette contours, the third type is degenerated to the first type to reduce manual operations in drawing the closed sketch. When the shape of a local 3D model can be defined by an open sketch and a closed sketch, the third type is degenerated to the second type to reduce the manual operations in drawing one of the two open sketches. In the following subsections, we will describe the algorithms to create 3D shapes from the three different types of sketches.

5.1. Two open silhouette contours in two different view planes

For creation of a local 3D shape from two open silhouette contours in two different view planes, we use the intersecting point p to segment each of the two sketched 2D silhouette contours obtained from the front view and the side view into two curves and denote all the four segmented curves with $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ and $\boldsymbol{c}_4$, which are defined by the points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$, $j = 1, 2, 3, \ldots, J$, respectively. Then we find the four corresponding points from the four curves, and denote them with $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$, $j = 1, 2, 3, \ldots, J$, respectively as shown in Fig. 8(a). 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.

The 3D model to be created from the two sketched 2D silhouette contours can be regarded as a sweeping surfaces whose generator is a closed curve $\boldsymbol{c}(u)$ shown in Fig. 7(b) passing through the four corresponding points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$ as shown in Figs. 7(a) and 7(b). We uniformly divide the domain of the parametric variable u corresponding to the closed curve (generator) $\boldsymbol{c}(u)$ into $2N$ equal intervals and obtain $\Delta u = 1/(2N)$ and nodes $i = 0, 1, 2, \ldots, 2N-1, 2N$ with $\boldsymbol{X}_{2N} = \boldsymbol{X}_0$. The nodes corresponding to $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$ are $0, N/2, N$ and $3N/2$ respectively as demonstrated in Fig. 7(b). Here, the selection of N should ensure that N is an even number and $N \geq 4$.

For the problem shown in Fig. 7(b), the finite difference equation for the inner nodes $i = 1, 2, \ldots, N-1, N+1, \ldots, 2N-1$ can be formulated with Eq. (4.2). In



(a) Sketched contours (b) Generator to be created

Figure 7: The finite difference nodes used in the algorithm in the case of two open silhouette contours in two different view planes.

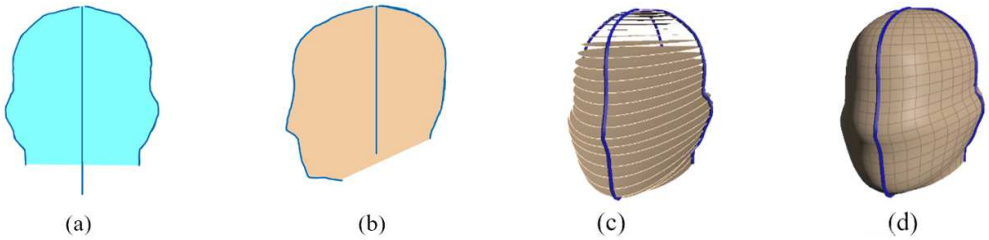
order to solve these finite difference equations, the boundary conditions at the finite difference nodes $0, N/2, N$, and $3N/2$ must be applied. These boundary conditions are: $\boldsymbol{X}_0 = \boldsymbol{c}_{1j}, \boldsymbol{X}_{N/2} = \boldsymbol{c}_{2j}, \boldsymbol{X}_N = \boldsymbol{c}_{3j}$, and $\boldsymbol{X}_{3N/2} = \boldsymbol{c}_{4j}$.

Among the inner nodes, the finite difference equations for the nodes $3, 4, \ldots, N/2-4$, $N/2-3$; $N/2+3, N/2+4, \ldots, N-4, N-3$; $N+3, N+4, \ldots, 3N/2-4$, $3N/2-3$; $3N/2+3, 3N/2+4, \ldots, 2N-4$, and $2N-3$ are the same as Eq. (4.2).

The finite difference equations for the nodes $1, 2, N/2-2, N/2-1, N/2+1, N/2+2$, $N-2, N-1, N+1, N+2, 3N/2-2, 3N/2-1, 3N/2+1, 3N/2+2, 2N-2$, and $2N-1$ can be obtained by introducing the corresponding one of $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$ into Eq. (4.2) to replace $\boldsymbol{X}_1, \boldsymbol{X}_{N/2}, \boldsymbol{X}_N$ and $\boldsymbol{X}_{3N/2}$ accordingly. The finite difference equations for these nodes are given in Appendix B.

Putting all obtained finite difference equations together, we obtain all the unknown constants $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{N/2-2}, \boldsymbol{X}_{N/2-1}$; $\boldsymbol{X}_{N/2+1}$, $\boldsymbol{X}_{N/2+2}$, $\ldots$, $\boldsymbol{X}_{N-2}$, $\boldsymbol{X}_{N-1}$; $\boldsymbol{X}_{N+1}$, $\boldsymbol{X}_{N+2}, \ldots, \boldsymbol{X}_{3N/2-2}, \boldsymbol{X}_{3N/2-1}$; $\boldsymbol{X}_{3N/2+1}, \boldsymbol{X}_{3N/2+2}, \ldots, \boldsymbol{X}_{2N-2}$, and $\boldsymbol{X}_{2N-1}$. They together with the known four points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}, \boldsymbol{c}_{3j}$ and $\boldsymbol{c}_{4j}$ are used to define the generator.

Repeating the above treatment for all other points on the four curves $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ and $\boldsymbol{c}_4$, we obtain the generators at the other positions. With these generators, we create local 3D shapes. Fig. 8 shows an example of using the shape generator to create a head model.



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

Figure 8: The Shape creation with the first shape generator: (a) 2D silhouette contours of a male head in the front view, (b) 2D silhouette contours of the male head in the side view, (c) the cross sections of the male head, (d) lofting the cross-sections into the male head mesh.

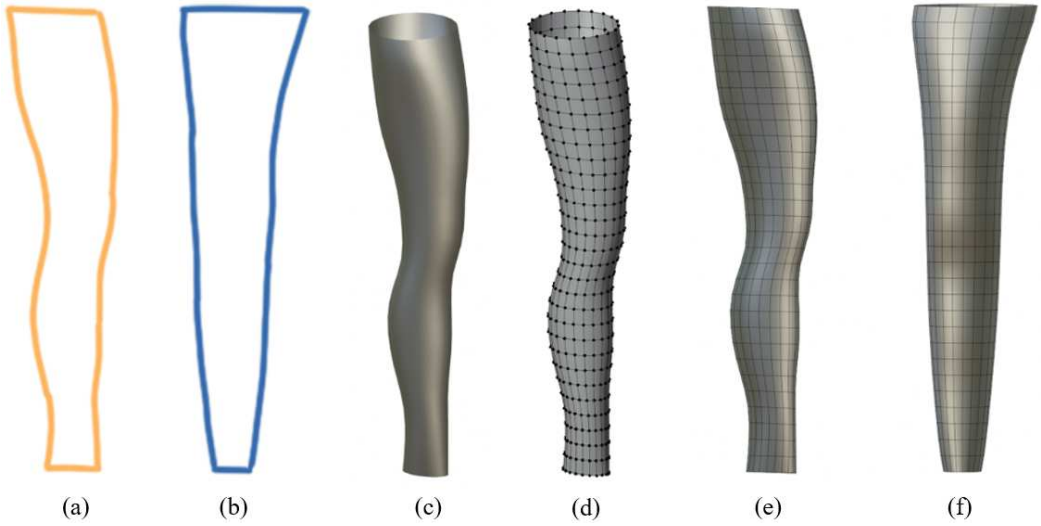


Figure 9: Generation of a leg shape from two open silhouette contours in two different view planes: (a) contours in the side view, (b) contours in the front view, (c) leg mesh by using our method in the perspective view, (d) leg mesh in the point view, (e) leg mesh in the side view, (f) leg mesh in the front view.

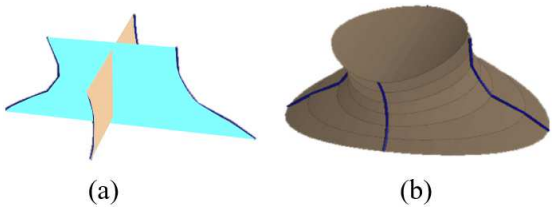


Figure 10: Generation of a neck shape from two open silhouette contours in two different view planes: (a) contours of a neck, (b) cross sections of a neck generated by our method.

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 Figs. 9(a) and 9(b), respectively. The corresponding four points on the four silhouette contours are used to define a cross section curve. All these cross-section curves are used to generate a 3D leg model shown in Figs. 9(c) and 9(d) with the front and side views in Figs. 9(e) and 9(f), respectively.

Another example is to create a neck shape shown in Fig. 10. Fig. 10(a) shows the four disconnected contours, Fig. 10(b) shows the cross-sections.

5.2. One open and one closed silhouette contours in two different view planess

For creation of a local 3D shape from one open and one closed silhouette contours in two different view planes shown in Fig. 11, the intersecting points $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ between the open silhouette contour and the closed silhouette contour divide the closed

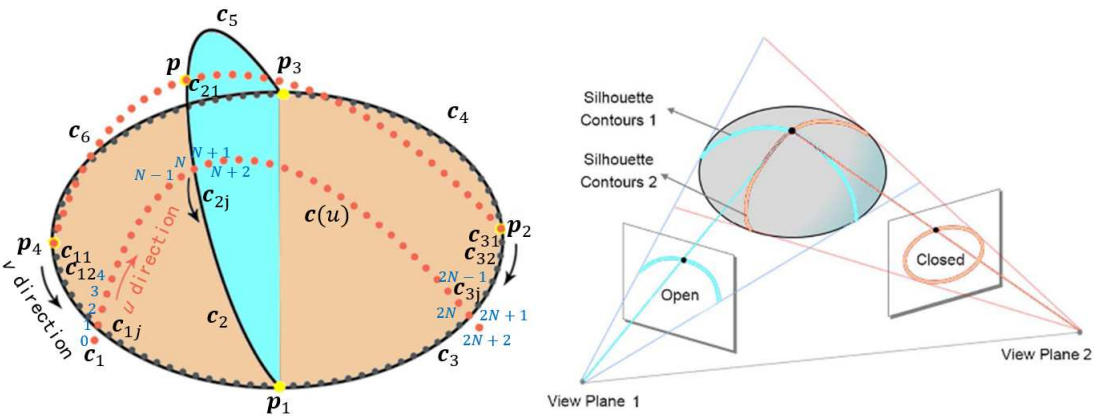


Figure 11: Finite difference nodes for one open and one closed silhouette contour in two different view planes.

silhouette contour into two curves. Then we find the middle points $\boldsymbol{p}_2$ and $\boldsymbol{p}_4$ of the two curves. These four points $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ and $\boldsymbol{p}_4$ divide the closed silhouette contour into four curves $\boldsymbol{c}_1, \boldsymbol{c}_3, \boldsymbol{c}_4$ and $\boldsymbol{c}_6$. Next, we find the middle point $\boldsymbol{p}$ of the open silhouette contour which divides the open silhouette contour into two curves $\boldsymbol{c}_2$ and $\boldsymbol{c}_5$. With this treatment, the creation of the local 3D detail model is changed into creation of two sweeping surfaces: one is defined by the three curves $\boldsymbol{c}_1, \boldsymbol{c}_2$ and $\boldsymbol{c}_3$, and the other is defined by the three curves $\boldsymbol{c}_4, \boldsymbol{c}_5$ and $\boldsymbol{c}_6$.

Since the creation process of the two sweeping surfaces is the same, we take the creation of the sweeping surface defined by the three curves $\boldsymbol{c}_1, \boldsymbol{c}_2$ and $\boldsymbol{c}_3$ to demonstrate the process. For each of the three curves c_1, c_2 and $\boldsymbol{c}_3$, we uniformly divide it into $2N$ equal intervals, and use $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}$ and $\boldsymbol{c}_{3j}$, $j = 1, 2, 3, \ldots, J$, to respectively indicate the nodes on the three curves.

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

To create the generator $\boldsymbol{c}(u)$ passing the points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}$, and $\boldsymbol{c}_{3j}$, we uniformly divide the domain of the parametric variable u corresponding to the generator $\boldsymbol{c}(u)$ into $2N$ equal intervals, and obtain $\Delta u = 1/(2N)$ and the nodes $1, 2, \ldots, 2N-1, 2N$ and $2N+1$. The finite difference equation for the inner nodes $i = 2, 3, \ldots, N-1, N, N+2, \ldots, 2N$ can be formulated with Eq. (4.2). In order to solve these finite difference equations, the boundary conditions (i.e. the values) at the finite difference nodes $0, 1, N+1, 2N+1$ and $2N+2$ must be applied. These boundary conditions can be obtained below.

Since the nodes $1, N+1$ and $2N+1$ are on the curves $\boldsymbol{c}_1, \boldsymbol{c}_2$, and $\boldsymbol{c}_3$, respectively, their values are known, i.e., $\boldsymbol{X}_1 = \boldsymbol{c}_{1j}, \boldsymbol{X}_{N+1} = \boldsymbol{c}_{2j}$ and $\boldsymbol{X}_{2N+1} = \boldsymbol{c}_{3j}$. When writing the 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. Their value can be determined with the following method.

If the created local 3D shape is to be smoothly connected to another 3D shape with positional and tangential continuities, the vector-valued first derivative $\boldsymbol{T}_{1j}$ at the node 1 and $\boldsymbol{T}_{3j}$ at the node $2N+1$ can be obtained from another 3D shape. Then, the nodes 0 and $2N+2$ can be determined by the following finite difference approximation:

$$\boldsymbol{T}_{1j} = \frac{\boldsymbol{X}_2 - \boldsymbol{X}_0}{2\Delta u}, \quad \boldsymbol{T}_{3j} = \frac{\boldsymbol{X}_{2N+2} - \boldsymbol{X}_{2N}}{2\Delta u}. \tag{5.1}$$

Solving the above equation, we obtain the nodes 0 and $2N+2$ by

$$\boldsymbol{X}_0 = \boldsymbol{X}_2 - 2\boldsymbol{T}_{1j}\Delta u, \quad \boldsymbol{X}_{2N+2} = \boldsymbol{X}_{2N} + 2\boldsymbol{T}_{3j}\Delta u. \tag{5.2}$$

Writing the finite difference equations for all the nodes from 2 to $2N$, we obtain $2N-1$ linear equations. Among them, the finite difference equations for the nodes $4, 5, \ldots, N-3, N-2$ and $N+4, N+5, \ldots, 2N-3, 2N-2$ are the same as Eq. (4.2), and the finite difference equation involves $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$ at the node 2, $\boldsymbol{X}_0$ at node 3, $\boldsymbol{X}_{N+1}$ at the nodes $N-1, N, N+2, N+3$, $\boldsymbol{X}_{2N+1}$ at the node $2N-1$, and $\boldsymbol{X}_{2N+1}$ and $\boldsymbol{X}_{2N+2}$ at the node $2N$, which are determined by considering the five boundary conditions $\boldsymbol{X}_0 = \boldsymbol{X}_2 - 2\boldsymbol{T}_{1j}\Delta u$, $\boldsymbol{X}_1 = \boldsymbol{c}_{1j}$, $\boldsymbol{X}_{N+1} = \boldsymbol{c}_{2j}$, $\boldsymbol{X}_{2N+1} = \boldsymbol{c}_{3j}$ and $\boldsymbol{X}_{2N+2} = \boldsymbol{X}_{2N} + 2\boldsymbol{T}_{3j}\Delta u$. The obtained finite difference equations for the nodes $2, 3, N-1, N, N+2, N+3, 2N-1$ and $2N$ are given in Appendix C.

Solving the finite difference equations for all the inner nodes $2, 3, \ldots, N-1, N$, $N+2, N+3, \ldots, 2N-1$ and $2N$, we obtain all the unknown constants $\boldsymbol{X}_2, \boldsymbol{X}_3, \ldots$, $\boldsymbol{X}_{N-1}, \boldsymbol{X}_N, \boldsymbol{X}_{N+2}, \boldsymbol{X}_{N+3}, \ldots, \boldsymbol{X}_{2N-1}$, and $\boldsymbol{X}_{2N}$. They together with the three known points $\boldsymbol{c}_{1j}, \boldsymbol{c}_{2j}$, and $\boldsymbol{c}_{3j}$ are used to create the generator.

Using the same treatment, we can obtain the generator at the other positions. From these obtained generators, a detail 3D shape can be created. Fig. 12 gives a buttock shape created with the above method.

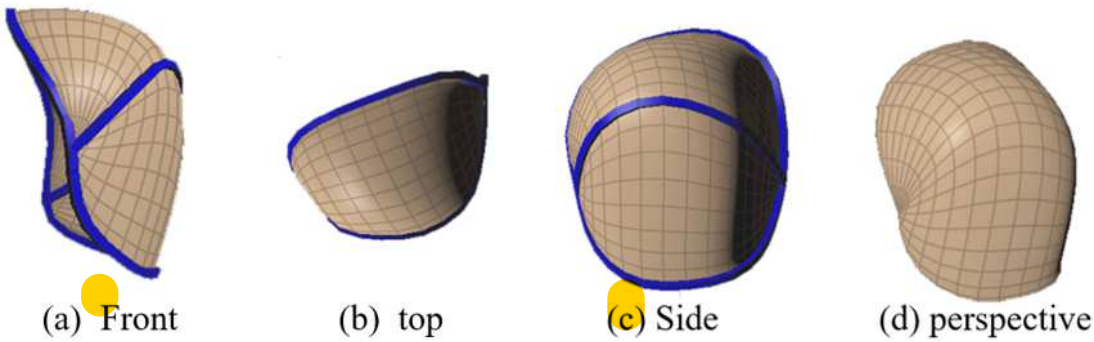


Figure 12: The shape creation from one open and one closed silhouette contours in two different view planes.

5.3. Two open and one closed silhouette contours in three different view planes

As shown in Fig. 13, the task of creating a local 3D shape passing through two open and one closed silhouette contours can be transformed into creating 4 sweeping surfaces encircled by the curves $\boldsymbol{c}_1\boldsymbol{c}_6\boldsymbol{c}_5$, $\boldsymbol{c}_2\boldsymbol{c}_7\boldsymbol{c}_6$, $\boldsymbol{c}_3\boldsymbol{c}_8\boldsymbol{c}_7$, and $\boldsymbol{c}_4\boldsymbol{c}_5\boldsymbol{c}_8$, respectively.

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

The sweeping surface can be regarded as sweeping the curve $\boldsymbol{c}_5$ to the curve $\boldsymbol{c}_6$ along the curve $\boldsymbol{c}_1$ and the point $\boldsymbol{p}$. In order to obtain the generator of the sweeping surface, we uniformly divide the curve $\boldsymbol{c}_1$ into $J-1$ 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 the curve $\boldsymbol{c}_5$ and $\boldsymbol{c}_1$ and between $\boldsymbol{c}_6$ and $\boldsymbol{c}_1$. The two ends of the generator j are at the point j and the point $\boldsymbol{p}$, respectively. We uniformly divide the domain of the parametric variable u of the generator j between the point j and the point $\boldsymbol{p}$ into $2N$ equal intervals, and obtain $\Delta u = 1/(2N)$ and the nodes $1, 2, \ldots, 2N$, and $2N+1$. Then, we use $\boldsymbol{X}_{j,i},\ j = 1, 2, 3, \ldots, J-2, J-1, J,\ i = 1, 2, \ldots, 2N, 2N+1$, to indicate the vector-valued coordinate of the node i on the generator j. When $j = 1$, $\boldsymbol{X}_{1,i}, i = 1, 2, \ldots, 2N, 2N+1$, indicate the vector-valued coordinates of the nodes on the curve $\boldsymbol{c}_5$. When $j = J$, $\boldsymbol{X}_{J,i}, i = 1, 2, \ldots, 2N, 2N+1$, indicate the vector-valued coordinates of the nodes on the curve $\boldsymbol{c}_6$.

In order to create the sweeping surface encircled by the three curves $\boldsymbol{c}_1, \boldsymbol{c}_6$ and $\boldsymbol{c}_5$, we require the additional information called sculpting forces acting at all the inner notes. To this aim, we modify the vector-valued ordinary differential equation (4.2) by introducing a sculpting force $\boldsymbol{f}$ and obtain

$$k_b \frac{d^4 \boldsymbol{X}}{du^4} - k_s \frac{d^2 \boldsymbol{X}}{du^2} = \boldsymbol{f}, \tag{5.3}$$

$\boldsymbol{f}$ is a vector-valued force, which has three components f_x, f_y, and f_z.

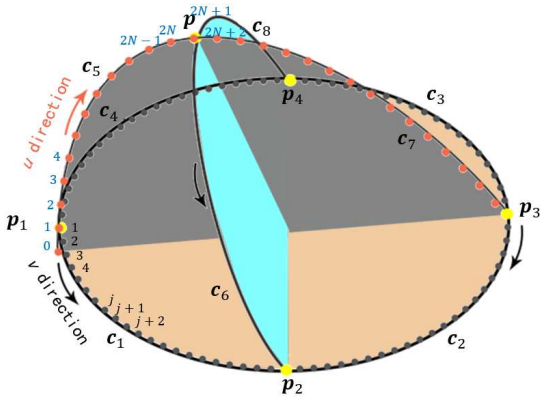


Figure 13: Creation of a 3D local shape from two open and one closed silhouette contours.

Accordingly, the finite difference equation at the typical node i of the generator j for the above ordinary differential equation is

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{j,i} + k_b\boldsymbol{X}_{j,i-2} + k_b\boldsymbol{X}_{j,i+2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,i-1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,i+1} = \Delta u^4\boldsymbol{f}_{j,i}. \tag{5.4}$$

For the example shown in Fig. 13, the finite difference equation for the inner nodes $i = 2, 3, \ldots, 2N-1, 2N$ can be formulated with Eq. (4.2). In order to solve these finite difference equations, the boundary conditions (i.e. the values) at the finite difference nodes $0, 1, 2N+1$, and $2N+2$ must be applied and the forces $\boldsymbol{f}_{j,i}$ must be known. They can be obtained below.

When using Eq. (5.4) to write the finite difference equation at the inner nodes $i = 2$ and $i = 2N$, the boundary nodes 1 and $2N+1$ and the node 0 beyond the boundary node 1 and the node $2N+2$ beyond the boundary node $2N+1$ are involved. When using Eq. (5.4) to write the finite difference equation at the inner nodes $i = 3$ and $i = 2N-1$, the boundary nodes 1 and $2N+1$ are involved. Therefore, we first discuss how to determine the boundary conditions (i.e., the values $\boldsymbol{X}_{j,0}, \boldsymbol{X}_{j,1}, \boldsymbol{X}_{j,2N+1}$ and $\boldsymbol{X}_{j,2N+2}$) at the node $0, 1, 2N+1$, and $2N+2$.

Since the curve $\boldsymbol{c}_1$ and the point $\boldsymbol{p}$ are known, we obtain $\boldsymbol{X}_{j,1} = \boldsymbol{c}_{1,j}$ and $\boldsymbol{X}_{j,2N+1} = \boldsymbol{c}_{5,2N+1}$. Since the curves $\boldsymbol{c}_5$ and $\boldsymbol{c}_6$ are known, we can calculate the vector-valued first derivatives of the curves $\boldsymbol{c}_5(u)$ and $\boldsymbol{c}_6(u)$ at the node 1 and $2N+1$, and indicate them with $\boldsymbol{T}_{1,1}$, $\boldsymbol{T}_{1,2N+1}$, $\boldsymbol{T}_{J,1}$, and $\boldsymbol{T}_{J,2N+1}$. Using the central finite difference approximation and Fig. 13, we have

$$\boldsymbol{T}_{1,1} = \frac{\boldsymbol{X}_{1,2} - \boldsymbol{X}_{1,0}}{2\Delta u}, \quad \boldsymbol{T}_{1,2N+1} = \frac{\boldsymbol{X}_{1,2N+2} - \boldsymbol{X}_{1,2N}}{2\Delta u},$$
$$\boldsymbol{T}_{J,1} = \frac{\boldsymbol{X}_{J,2} - \boldsymbol{X}_{J,0}}{2\Delta u}, \quad \boldsymbol{T}_{J,2N+1} = \frac{\boldsymbol{X}_{J,2N+2} - \boldsymbol{X}_{J,2N}}{2\Delta u}.$$

When sweeping the curve $\boldsymbol{c}_5$ along the curve $\boldsymbol{c}_1$ and the point $\boldsymbol{p}$ to the curve $\boldsymbol{c}_6$, the vector-valued first derivatives $\boldsymbol{T}_{1,1}$ and $\boldsymbol{T}_{1,2N+1}$ gradually change to $\boldsymbol{T}_{J,1}$ and $\boldsymbol{T}_{J,2N+1}$. Here we use a linear interpolation to describe the gradual change and obtain the vector-valued first derivatives $\boldsymbol{T}_{j,1}$ and $\boldsymbol{T}_{j,2N+1}$, $j = 2, 3, \ldots, J-2, J-1$, with the following equations:

$$\begin{aligned} \boldsymbol{T}_{j,1} &= \boldsymbol{T}_{1,1} + \frac{L_j}{L}(\boldsymbol{T}_{J,1} - \boldsymbol{T}_{1,1}), \\ \boldsymbol{T}_{j,2N+1} &= \boldsymbol{T}_{1,2N+1} + \frac{L_j}{L}(\boldsymbol{T}_{J,2N+1} - \boldsymbol{T}_{1,2N+1}), \end{aligned} \tag{5.5}$$

where L_j is the arc length from the point $\boldsymbol{p}_1$ to the node j on the curve $\boldsymbol{c}_1$ and L is the arc length from the point $\boldsymbol{p}_1$ to the point $\boldsymbol{p}_2$. Since

$$\boldsymbol{T}_{j,1} = \frac{\boldsymbol{X}_{j,2} - \boldsymbol{X}_{j,0}}{2\Delta u}, \quad \boldsymbol{T}_{j,2N+1} = \frac{\boldsymbol{X}_{j,2N+2} - \boldsymbol{X}_{j,2N}}{2\Delta u},$$

we obtain

$$\boldsymbol{X}_{j,0} = \boldsymbol{X}_{j,2} - 2\boldsymbol{T}_{j,1}\Delta u, \quad \boldsymbol{X}_{j,2N+2} = \boldsymbol{X}_{j,2N} + 2\boldsymbol{T}_{j,2N+1}\Delta u. \tag{5.6}$$

In order to obtain $X_{j,i}$, $j = 2,3,\ldots,J-2,J-1$, $i = 2,3,\ldots,2N-1,2N$ from Eq. (5.4), we need to determine $\boldsymbol{f}_{j,i}$. For each of the inner nodes $2,3,\ldots,2N-1,2N$, we can write a finite difference equation. Since the coordinate values for all the nodes on the curve $\boldsymbol{c}_5$ are known, we can calculate the sculpting force $\boldsymbol{f}_i$, $i = 2,3,\ldots,2N-1,2N$ from the finite difference equation (5.4). We denote these sculpting forces acting at the nodes of the curve $\boldsymbol{c}_5$ as $\boldsymbol{f}_{1,i} = \boldsymbol{f}_i$, $i = 2,3,\ldots,2N-1,2N$. With the same method, we can obtain the sculpting forces $\boldsymbol{f}_{J,i}$, $i = 2,3,\ldots,2N-1,2N$, acting at the nodes of the curve $\boldsymbol{c}_6$.

When sweeping the curve $\boldsymbol{c}_5$ along the curve $\boldsymbol{c}_1$ to the curve $\boldsymbol{c}_6$, the sculpting force $\boldsymbol{f}_{1,i}$ acting at the node i of the curve $\boldsymbol{c}_5$ is gradually changed to the sculpting force $\boldsymbol{f}_{J,i}$ acting at the node i of the curve $\boldsymbol{c}_6$. Same as above, we use a linear interpolation to describe the gradual change and obtain the sculpting force $\boldsymbol{f}_{j,i}$ below acting at the inner node i of the generator j

$$\boldsymbol{f}_{j,i} = \boldsymbol{f}_{1,i} + \frac{L_j}{L}(\boldsymbol{f}_{J,i} - \boldsymbol{f}_{1,i}), \quad j = 2,3,\ldots,J-2,J-1, \quad i = 2,3,\ldots,2N-1,2N. \quad (5.7)$$

Having known the sculpting forces $\boldsymbol{f}_{j,i}$, $j = 2,3,\ldots,J-2,J-1$, $i = 2,3,\ldots,2N-1$, $2N$, boundary nodes $\boldsymbol{X}_{j,1}$ and $\boldsymbol{X}_{j,2N+1}$, and the nodes $\boldsymbol{X}_{j,0}$ and $\boldsymbol{X}_{j,2N+2}$ beyond the boundary nodes $\boldsymbol{X}_{j,1}$ and $\boldsymbol{X}_{j,2N+1}$, we can write the finite difference equations for all the inner nodes on the generator j. The finite difference equations at the inner nodes ($j = 2,3,\ldots,J-2,J-1, i = 4,5,\ldots,2N-3,2N-2$) are the same as Eq. (5.4). The finite difference equations at the inner nodes $2,3,2N-1$ and $2N$ on the generator j are given in Appendix D.

Solving all the finite difference equations, we obtain all the unknown constants $\boldsymbol{X}_{j,2}, \boldsymbol{X}_{j,3},\ldots,\boldsymbol{X}_{j,2N-1}$, and $\boldsymbol{X}_{j,2N}$. They together with the two known points $\boldsymbol{X}_{j,1}$ and $\boldsymbol{X}_{j,2N+1}$ are used to create the generator j of the sweeping surface.

With the above method, we draw a closed lip contour and two open curves, as shown in Figs. 14(a) and 14(b). The two open curves divide the mouth into four regions. One surface is created for each of the four regions. Fig. 14(c) depicts the mouth shape, which consists of the four surfaces.

The above method can also be extended to deal with the situations where the two open curves do not intersect. For such situations, the intersecting point of the two open curves becomes the upper closed curve as indicated in Fig. 15, and a sweeping surface

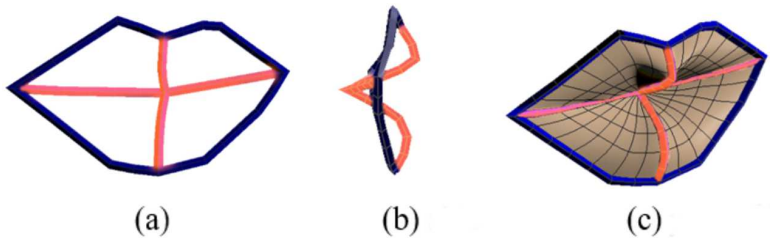


Figure 14: A mouth shape creation from two open and one closed silhouette contours in three different view planes.

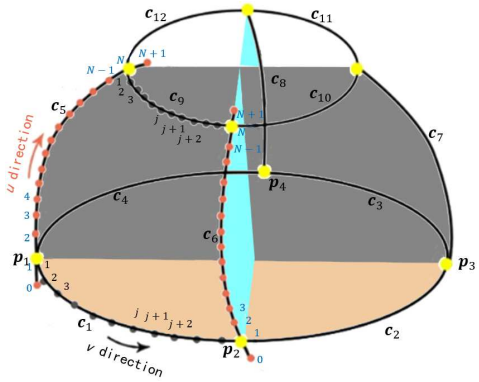


Figure 15: Finite difference nodes for four open and two closed silhouette contours.

Figure 16: Vest and eye socket creation from four open and two closed silhouette contours.

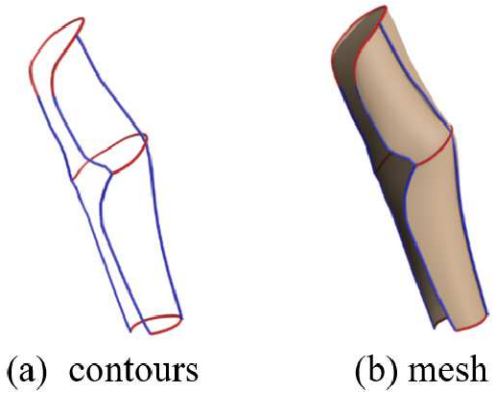


Figure 17: Creation of a thin sheet on the left forearm.

is encircled by four curves. With this extension, we can create a 3D vest model and an eye socket model, as shown in Figs. 16(a) and 16(b), respectively.

The extended algorithm is not only applicable to top and bottom closed curves, but also top and bottom open curves. Fig. 17 gives such an example where the three curves in red are open as shown in Fig. 17(a). The generated 3D model is depicted in Fig. 17(b).

6. Experiments and comparisons

In this section, we present two experiments to validate that the proposed approach can create more realistic shapes and use the creation of a 3D facial model to demonstrate that the proposed approach can create detailed 3D models more efficiently.

The first experiment is to deform a plastic ruler and compare its ground-truth deformations with those determined by our ODE-driven deformation approach. As shown in Fig. 18(a), the plastic ruler is fixed at one end and its other end is tied with a heavy weight, like a cantilever beam. The ruler is bent towards the ground by $7.2\ cm$ at its free end with a force equal to the gravity of the heavy weight acting on it. The setup of our ODE-driven deformations is: the displacement differences $y_0 = 0$ and $y_{N+1} = 7.2\ cm$, the rotation of the fixed end $dy_0/du = 0$, and the third derivative of displacement at the free end is proportional to the force applied to it $d^3y_{N+1}/du^3 = f_y/k_b$. The ruler material is PP plastics. We use its bend modulus to determine the bending stiffness $k_b = 5\ GPa = 0.5\ N/cm^2$. The red curve in Fig. 18(a) shows the simulation result, which is very close to the ground-truth (the grey curve).

In the second experiment shown in Fig. 18(b), two identical objects simply support a ruler at its two ends, and a load is put at the midpoint of the ruler. Since we do not fix the two ends of the ruler, no bending moment M results in zero curvature at the two ends $d^2y_0/du^2 = d^2y_{2N}/du^2 = M/EI = 0/EI = 0$. Together with the displacement $y_0 = y_{2N} = 0$ at the two ends and $y_N = 1.06\ cm$ at the midpoint, the simulation result by our ODE-driven deformation technique is shown as the red curve in Fig. 18(b), which is also very close to the ground-truth (the grey curve). These two experiments indicate that the OED-driven deformations produce realistic appearances.

Next we use 3D face creation as a specific application to demonstrate that the proposed approach can efficiently add 3D details to create detailed 3D models. With our sketch-based and ODE-driven shape modelling approach, a curve network shown in

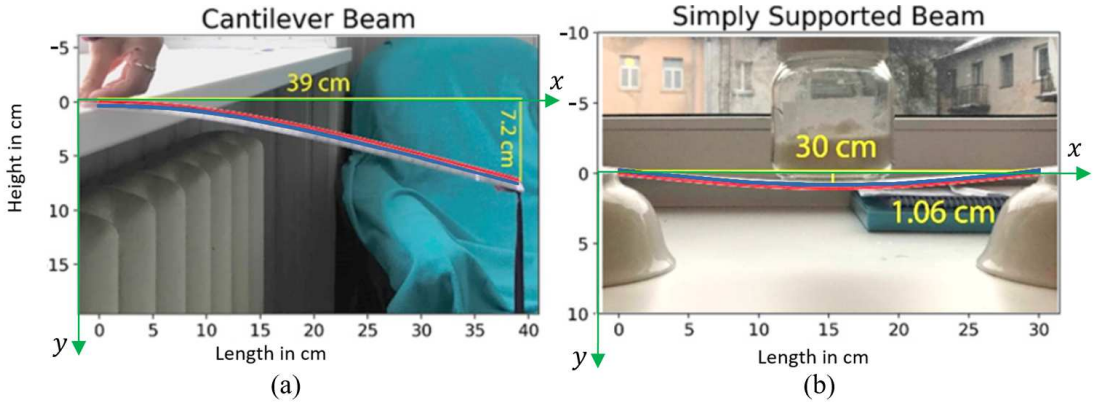


Figure 18: Deformation comparison: (a) deform a plastic ruler by fixing one end of the ruler and putting a heavy weight on the other free end of the ruler, (b) deform a plastic ruler by placing a load at the midpoint of a plastic ruler, with the two ends of the ruler simply supported by two identical objects. Red curves are the simulated results and the grey curves are the ground-truths.

Fig. 19(b), which defines the female 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 Fig. 19(c). Compared to the approach of manipulating primitives in one view plane proposed in [19], our approach adds 3D details to eyes, nose, mouth, etc. and creates more detailed 3D shapes. With Maya polygon modelling, we take about 40 hours as a less experienced person to create the facial model in Fig. 19(a). The total time of obtaining the detailed facial model in Fig. 19(c) including generating the facial curve network in Fig. 19(b) is about 5 hours. Clearly, our proposed sketch-based and ODE-driven shape generator is more efficient in creating detailed facial models than the polygon modelling.

The computer language used to implement our proposed approach is Python scripting language. It was used to get information from and control Houdini via the Houdini FX Education Edition $16.5.323$ package. The implemented approach ran on a dual boot Linux PC with 23 GB memory and 64 bits Intel(R) Xeon(R) CPU E5-1650 0 @ 3.20 GHz CPU. The average time for deforming a primitive is 0.17 seconds and the average time for detail generation is 0.09 seconds, which ensures a smooth real-time modelling user experience. The average time is the total time divided by the times of using the primitive deformer and shape generators to obtain shape changes.

After using our developed shape generators to add 3D details to the base model, we can obtain a detailed model shown in Fig. 20(a). We also use the approach developed in this paper to create another detailed character model shown in Fig. 20(b). Comparing the 3D models obtained in Fig. 21 with the 3D models obtained in [19], it is clear that our proposed approach creates more detailed models than the existing approach.

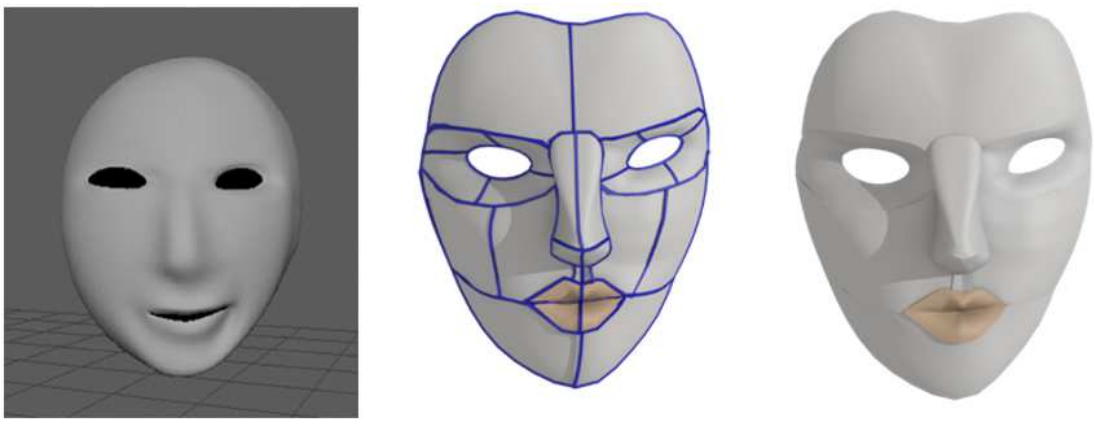


Figure 19: Comparison of creating 3D facial details with the polygon modelling and sketch-based and ODE-driven shape generators: (a) traditional polygon modelling result, (b) generated contours for face reconstruction, (c) result of our sketch-based modelling.

Figure 20: 3D character models created using our approach.

7. Conclusions and future work

A new modelling method has been presented in this paper to quickly create detailed character models from sketches. By proposing an efficient finite difference solution to a fourth-order ordinary differential equation, we have developed an ODE-driven primitive deformer, which deforms primitives to fit 2D sketches exactly and ODE-driven shape generators in creating new 3D local shapes from the user's drawn sketches. We have also demonstrated the efficiency of our approach to create detailed 3D models with physical realism.

Compared to existing sketch-based modelling approaches, our approach is physics-based and able to create physically realistic and detailed 3D models from 2D sketches efficiently. It is particularly powerful in deforming and manipulating primitives to obtain more detailed shapes. Compared to the traditional polygon modelling technology, the approach developed in this paper has the advantages of: 1) easier for beginners to create detailed character models, 2) greatly reducing manual operations, and 3) more efficient in modelling character models.

Our current approach has several limitations. First, the input stroke coordinates of sketches may have certain ambiguities, since in essence they are 2D sketches instead of 3D sketches. Although our approach could create 3D shapes from 2D sketches in two or three different views, including generating various 3D details, having the capability to directly handle 3D sketches can make this approach more robust. Second, our current approach does not support suggestive contours [14], which makes the face creation

more complicated compared to other details such as shoulder, armors, etc., as the user has to provide more strokes to assist the generation of face parts. We plan to address these limitations in our future work.

Although the modelling examples presented in this paper are character models including human part models, our approach is potentially applicable to other organic models and man-made models. In the future, we will extend its applications to geometric modelling of various organic and man-made models.

Appendix A. Finite difference equations at the unknown nodes adjacent to the known nodes 0 and N

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

At node 1,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_1 + k_b\boldsymbol{X}_{2N-1} + k_b\boldsymbol{X}_3 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_2 = (4k_b + k_s\Delta u^2)(\boldsymbol{C}'_{13} - \boldsymbol{C}_{13}). \tag{A1}$$

At node 2,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_2 + k_b\boldsymbol{X}_4 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_1 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_3 = -k_b(\boldsymbol{C}'_{13} - \boldsymbol{C}_{13}). \tag{A2}$$

At node $N-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N-2} + k_b\boldsymbol{X}_{N-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-1} = -k_b(\boldsymbol{C}'_{23} - \boldsymbol{C}_{23}). \tag{A3}$$

At node $N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N-1} + k_b\boldsymbol{X}_{N-2} + k_b\boldsymbol{X}_{N+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-2} = (4k_b + k_s\Delta u^2)(\boldsymbol{C}'_{23} - \boldsymbol{C}_{23}). \tag{A4}$$

At node $N+1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N+1} + k_b\boldsymbol{X}_{N-1} + k_b\boldsymbol{X}_{N+3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+2} = (4k_b + k_s\Delta u^2)(\boldsymbol{C}'_{23} - \boldsymbol{C}_{23}). \tag{A5}$$

At node $N+2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N+2} + k_b\boldsymbol{X}_{N+4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+3} = -k_b(\boldsymbol{C}'_{23} - \boldsymbol{C}_{23}). \tag{A6}$$

At node $2N-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{2N-2} + k_b\boldsymbol{X}_{2N-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-1} = -k_b(\boldsymbol{C}'_{13} - \boldsymbol{C}_{13}). \tag{A7}$$

At node $2N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{2N-1} + k_b\boldsymbol{X}_{2N-3} + k_b\boldsymbol{X}_1 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-2} = (4k_b + k_s\Delta u^2)(\boldsymbol{C}'_{13} - \boldsymbol{C}_{13}). \tag{A8}$$

Appendix B. Finite difference equations at the unknown nodes adjacent to the known nodes $0, N/2, N$, and $3N/2$

The finite difference equations at the unknown nodes $1, 2, N/2-2, N/2-1, N/2+1, N/2+2, N-2, N-1, N+1, N+2, 3N/2-2, 3N/2-1, 3N/2+1, 3N/2+2, 2N-2$, and $2N-1$ can be formulated below.

At node 1,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_1 + k_b\boldsymbol{X}_{2N-1} + k_b\boldsymbol{X}_3 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_2 = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{1j}. \tag{B1}$$

At node 2,

$$(6k_b - 2k_s\Delta u^2)_2 + k_b\boldsymbol{X}_4 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_1 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_3 = -k_b\boldsymbol{C}_{1j}. \tag{B2}$$

At node $N/2-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N/2-2} + k_b\boldsymbol{X}_{N/2-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2-1} = -k_b\boldsymbol{C}_{2j}. \tag{B3}$$

At node $N/2-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N/2-1} + k_b\boldsymbol{X}_{N/2-3} + k_b\boldsymbol{X}_{N/2+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2-2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{2j}. \tag{B4}$$

At node $N/2+1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N/2+1} + k_b\boldsymbol{X}_{N/2-1} + k_b\boldsymbol{X}_{N/2+3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2+2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{2j}. \tag{B5}$$

At node $N/2+2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N/2+2} + k_b\boldsymbol{X}_{N/2+4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N/2+3} = -k_b\boldsymbol{C}_{2j}. \tag{B6}$$

At node $N-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N-2} + k_b\boldsymbol{X}_{N-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-1} = -k_b\boldsymbol{C}_{3j}. \tag{B7}$$

At node $N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N-1} + k_b\boldsymbol{X}_{N-3} + k_b\boldsymbol{X}_{N+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{3j}. \tag{B8}$$

At node $N+1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N+1} + k_b\boldsymbol{X}_{N-1} + k_b\boldsymbol{X}_{N+3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{3j}. \tag{B9}$$

At node $N+2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N+2} + k_b\boldsymbol{X}_{N+4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+3} = -k_b\boldsymbol{C}_{3j}. \tag{B10}$$

At node $3N/2-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{3N/2-2} + k_b\boldsymbol{X}_{3N/2-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2-1} = -k_b\boldsymbol{C}_{4j}. \tag{B11}$$

At node $3N/2-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{3N/2-1} + k_b\boldsymbol{X}_{3N/2-3} + k_b\boldsymbol{X}_{3N/2+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2-2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{4j}. \tag{B12}$$

At node $3N/2+1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{3N/2+1} + k_b\boldsymbol{X}_{3N/2-1} + k_b\boldsymbol{X}_{3N/2+3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2+2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{4j}. \tag{B13}$$

At node $3N/2+2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{3N/2+2} + k_b\boldsymbol{X}_{3N/2+4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2+1} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{3N/2+3} = -k_b\boldsymbol{C}_{4j}. \tag{B14}$$

At node $2N-2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{2N-2} + k_b\boldsymbol{X}_{2N-4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-1} = -k_b\boldsymbol{C}_{1j}. \tag{B15}$$

At node $2N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{2N-1} + k_b\boldsymbol{X}_{2N-3} + k_b\boldsymbol{X}_1 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-2} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{1j}. \tag{B16}$$

Appendix C. Finite difference equations at the unknown nodes adjacent to the known nodes $1, N+1$, and $2N+1$

The finite difference equations at the unknown nodes $2, 3, N-1, N, N+2, N+3$, $2N-1$ and $2N$ can be formulated below.

At node 2,

$$(6k_b - 2k_s\Delta u^2 - k_b)\boldsymbol{X}_2 + k_b\boldsymbol{X}_4 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_3 = 2k_b\boldsymbol{T}_{1j}\Delta u + (4k_b + k_s\Delta u^2)\boldsymbol{C}_{1j}. \tag{C1}$$

At node 3,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_3 + k_b\boldsymbol{X}_5 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_2 - (4k_b + k_s\Delta u^2)\boldsymbol{X}_4 = -k_b\boldsymbol{C}_{1j}. \tag{C2}$$

At node $N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N-1} + k_b\boldsymbol{X}_{N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_N = -k_b\boldsymbol{C}_{2j}. \tag{C3}$$

At node N,

$$(6k_b - 2k_s\Delta u^2 + k_b)\boldsymbol{X}_N + k_b\boldsymbol{X}_{N-2} + k_b\boldsymbol{X}_{N+2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N-1} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{2j}. \tag{C4}$$

At node $N+2$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{N+2} + k_b\boldsymbol{X}_N + k_b\boldsymbol{X}_{N+4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+3} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{2j}. \tag{C5}$$

At node $N+3$,

$$(6k_b - 2k_s\Delta u^2 + k_b)\boldsymbol{X}_{N+3} + k_b\boldsymbol{X}_{N+5} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{N+4} = -k_b\boldsymbol{C}_{2j}. \tag{C6}$$

At node $2N-1$,

$$(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{2N-1} + k_b\boldsymbol{X}_{2N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N} = -k_b\boldsymbol{C}_{3j}. \tag{C7}$$

At node $2N$,

$$(6k_b - 2k_s\Delta u^2 + k_b)\boldsymbol{X}_{2N} + k_b\boldsymbol{X}_{2N-2} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{2N-1} = (4k_b + k_s\Delta u^2)\boldsymbol{C}_{3j} - 2k_b\boldsymbol{T}_{3j}\Delta u. \tag{C8}$$

Appendix D. Finite difference equations at the unknown nodes adjacent to the known nodes 1 and $2N+1$

The finite difference equations at the unknown nodes $2, 3, 2N-1$ and $2N$ of the generator j can be formulated below.

At node 2,

$$\begin{aligned}&(6k_b - 2k_s\Delta u^2 + k_b)\boldsymbol{X}_{j,2} + k_b\boldsymbol{X}_{j,4} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,3}\\ &= \Delta u^4\boldsymbol{f}_{j,2} + 2k_b\Delta u\boldsymbol{T}_{j,1} + (4k_b + k_s\Delta u^2)\boldsymbol{X}_{1,j}.\end{aligned} \tag{D1}$$

At node 3,

$$\begin{aligned}&(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{j,3} + k_b\boldsymbol{X}_{j,5} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,2}\\ &\quad - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,4} = \Delta u^4\boldsymbol{f}_{j,3} - k_b\boldsymbol{C}_{1,j}.\end{aligned} \tag{D2}$$

At node $2N-1$,

$$\begin{aligned}&(6k_b - 2k_s\Delta u^2)\boldsymbol{X}_{j,2N-1} + k_b\boldsymbol{X}_{j,2N-3} - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,2N-2}\\ &\quad - (4k_b + k_s\Delta u^2)\boldsymbol{X}_{j,2N} = \Delta u^4\boldsymbol{f}_{j,2N-1} - k_b\boldsymbol{C}_{5,2N+1}.\end{aligned} \tag{D3}$$

At node $2N$,

$$\begin{aligned}&(6k_b - 2k_s\Delta u^2 + k_b)\boldsymbol{X}_{2N} + k_b\boldsymbol{X}_{2N-2} - (4k_b + 2k_s\Delta u^2)\boldsymbol{X}_{2N-1}\\ &= \Delta u^4\boldsymbol{f}_{j,2N} - 2k_b\Delta u\boldsymbol{T}_{j,2N+1} + (4k_b + k_s\Delta u^2)\boldsymbol{C}_{5,2N+1}.\end{aligned} \tag{D4}$$

Acknowledgments

This research is supported by the PDE-GIR project, which has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 778035, by the project PID2021-127073OB-I00 of the MCIN/AEI/10.13039/501100011033/FEDER, EU, and by the Santander PGR Grants. X. Jin was supported by the Ningbo Major Special Projects of the "Science and Technology Innovation 2025", grant No. 2020Z007.

References

[1] F. ABBASINEJAD, P. JOSHI, AND N. AMENTA, *Surface patches from unorganized space curves*, in: Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry, Association for Computing Machinery (2012), 417–418.

[2] A. ANDRE AND S. SAITO, *Single-view sketch based modeling*, in: Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM '11), Association for Computing Machinery, (2011), 133–140.

[3] S.-H. Bae, R. Balakrishnan, and K. Singh, *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, Association for Computing Machinery, (2008), 151–160.
[4] M. Bessmeltsev, W. Chang, N. Vining, A. Sheffer, and K. Singh, *Modeling character canvases from cartoon drawings*, ACM Trans. Graph. 34(5) (2015), 1–16.
[5] S. Bhattacharjee and P. Chaudhuri, *A survey on sketch based content creation: From the desktop to virtual and augmented reality*, Comput. Graph. Forum 39 (2020), 757–780.
[6] S. Bian, Z. Deng, E. Chaudhry, L. H. You, X. S. Yang, L. Guo, H. Ugail, X. Jin, Z. Xiao, and J. J. Zhang, *Efficient and realistic character animation through analytical physics-based skin deformation*, Graph. Models 104 (2019), 101035.
[7] S. Bian, G. Maguire, W. Kokke, L. H. You, and J. J. Zhang, *Efficient C2 continuous surface creation technique based on ordinary differential equation*, Symmetry 12(1) (2019), 38.
[8] P. Borosán, M. Jin, D. Decarlo, Y. Gingold, and A. Nealen, *Rigmesh: Automatic rigging for part-based shape modeling and deformation*, ACM Trans. Graph. 31(6) (2012), 1–9.
[9] M. Botsch and O. Sorkine, *On linear variational surface deformation methods*, IEEE Trans. Vis. Comput. Graph. 14(1) (2008), 213–230.
[10] E. Chaudhry, S. Bian, H. Ugail, X. Jin, L. H. You, and J. J. Zhang, *Dynamic skin deformation using finite difference solutions for character animation*, Comput. Graph. 46 (2015), 294–305.
[11] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, *3-Sweep: Extracting editable objects from a single photo*, ACM Trans. Graph. 32(6) (2013), 1–10.
[12] P. Company, A. Piquer, M. Contero, and F. Naya, *A survey on geometrical reconstruction as a core technology to sketch-based modeling*, Comput. Graph. 29(6) (2005), 892–904.
[13] M. T. Cook and A. Agah, *A survey of sketch-based 3-D modeling techniques*, Interact. Comput. 21(3) (2009), 201–211.
[14] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, *Suggestive contours for conveying shape*, ACM Trans. Graph. 22(3) (2003), 848–855.
[15] Z. Deng, Y. Liu, H. Pan, W. Jabi, J. Zhang, and B. Deng, *Sketch2PQ: Freeform planar quadrilateral mesh design via a single sketch*, IEEE Trans. Vis. Comput. Graph. (2022), doi: 10.1109/TVCG.2022.3170853.
[16] C. Ding and L. Liu, *A survey of sketch based modeling systems*, Front. Comput. Sci. 10 (2016), 985–999.
[17] E. Entem, L. Barthe, M.-P. Cani, F. Cordier, and M. van de Panne, *Modeling 3D animals from a side-view sketch*, Comput. Graph. 46 (2015), 221–230.
[18] J. H. Fang, E. Chaudhry, A. Iglesias, J. Macey, L. H. You, and J. J. Zhang, *Reconstructing dynamic 3D models with small data by integrating position-based dynamics and PDE-based modelling*, Mathematics 10(5) (2022), 821.
[19] Y. Gingold, T. Igarashi, and D. Zorin, *Structured annotations for 2D-to-3D modeling*, ACM Trans. Graph. 28(5) (2009), 1–9.
[20] K. Golec, J.-F. Palierne, F. Zara, S. Nicolle, and G. Damiand, *Hybrid 3D mass-spring system for simulation of isotropic materials with any Poisson's ratio*, Vis. Comput. 36 (2020), 809–825.
[21] C. Grimm and P. Joshi, *JustDrawIt: A 3D sketching system*, in: Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, Association for Comput-

ing Machinery (2012), 121–130.
[22] T. IGARASHI, S. MATSUOKA, AND H. TANAKA, *Teddy: A sketching interface for 3D freeform design*, in: Proceedings of ACM SIGGRAPH 2006 Courses, Association for Computing Machinery (2006), 1–11.
[23] O. A. KARPENKO AND J. F. HUGHES, *SmoothSketch: 3D free-form shapes from complex sketches*, ACM Trans. Graph. 25(3) (2006), 589–598.
[24] I. K. KAZMI, L. H. YOU, X. S. YANG, X. G. JIN, AND J. J. ZHANG, *Efficient sketch-based creation of detailed character models through data-driven mesh deformations*, Comput. Animat. Virtual Worlds 26 (2015), 469–481.
[25] I. K. KAZMI, L. H. YOU, AND J. J. ZHANG, *A survey of sketch based modeling systems*, in: Proceedings of the 11th International Conference on Computer Graphics, Imaging and Visualization, IEEE (2014), 27–36.
[26] D. KIM, T. KUANG, Y. L. RODRIGUES, E. BOXERMAN, J. GATENO, S. G. F. SHEN, X. WANG, K. STEIN, H. H. DENG, M. A. K. LIEBSCHNER, AND J. J. XIA, *A novel incremental simulation of facial changes following orthognathic surgery using FEM with realistic lip sliding effect*, Med. Image Anal. 72 (2021), 102095.
[27] V. KRAEVOY, A. SHEFFER, AND M. VAN DE PANNE, *Modeling from contour drawings*, in: Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling, Association for Computing Machinery (2009), 37–44.
[28] Z. LEVI AND C. GOTSMAN, *ArtiSketch: A system for articulated sketch modeling*, Comput. Graph. Forum 32 (2013), 235–244.
[29] C. LI, H. PAN, Y. LIU, X. TONG, A. SHEFFER, AND W. WANG, *BendSketch: Modeling freeform surfaces through 2D sketching*, ACM Trans. Graph. 36(4) (2017), 1–14.
[30] O. W. LI, *Efficient and Detailed Sketch-Based Character Modelling with Composite Generalized Elliptic Curves and ODE Surface Creators*, PhD Thesis, Bournemouth University, 2020.
[31] O. W. LI, S. J. BIAN, A. NOREIKA, I. K. KAZMI, L. H. YOU, AND J. J. ZHANG, *Efficient creation of 3D organic models from sketches and ODE-based deformations*, in: Transactions on Computational Science XXXVIII, Lecture Notes in Computer Science, Springer, (2021), 1–16.
[32] W. MA, *Subdivision surfaces for CAD - an overview*, Comput. Aided Des. 37(7) (2005), 693–709.
[33] D. P. MADSEN, *Engineering Drawing and Design*, Cengage Learning, 2016.
[34] A. NEALEN, T. IGARASHI, O. SORKINE, AND M. ALEXA, *FiberMesh: Designing freeform surfaces with 3D curves*, ACM Trans. Graph. 26(3) (2007), 41.
[35] A. NEALEN, M. MÜLLER, R. KEISER, E. BOXERMAN, AND M. CARLSON, *Physically based deformable models in computer graphics*, Comput. Graph. Forum 25 (2006), 809–836.
[36] F. NOORUDDIN AND G. TURK, *Simplification and repair of polygonal models using volumetric techniques*, IEEE Trans. Vis. Comput. Graph. 9(2) (2003), 191–205.
[37] L. OLSEN, F. F. SAMAVATI, M. C. SOUSA, AND J. A. JORGE, *Sketch-based modeling: A survey*, Comput. Graph. 33(1) (2009), 85–103.
[38] H. PAN, Y. LIU, A. SHEFFER, N. VINING, C. LI, AND W. WANG, *Flow aligned surfacing of curve networks*, ACM Trans Graph 34(4) (2015), 1–10.
[39] C. D. PAOLI AND K. SINGH, *SecondSkin: Sketch-based construction of layered 3D models*, ACM Trans. Graph. 34(4) (2015), 126.
[40] L. PIEGL AND W. TILLER, *The NURBS Book*, Springer, 2012.
[41] A. RIVERS, F. DURAND, AND T. IGARASHI, *3D modeling with silhouettes*, ACM Trans. Graph. 29(4) (2010), 1–8.
[42] B. SADRI AND K. SINGH, *Flow-complex-based shape reconstruction from 3D curves*, ACM

Trans. Graph. 33(2) (2014), 1–15.
[43] A. SHTOF, A. AGATHOS, Y. GINGOLD, A. SHAMIR, AND D. COHEN-OR, *Geosemantic snapping for sketch-based modeling*, Comput. Graph. Forum 32(2) (2013), 245–253.
[44] E. TURQUIN, J. WITHER, L. BOISSIEUX, M. CANI, AND J. F. HUGHES, *A sketch-based interface for clothing virtual characters*, IEEE Comput. Graph. Appl. 27(1) (2007), 72–81.
[45] N. XIANG, R. WANG, T. JIANG, L. WANG, Y. LI, X. S. YANG, AND J. J. ZHANG, *Sketch-based modeling with a differentiable renderer*, Comput. Animat. Virtual Worlds 31(4-5) (2020), 1–12.
[46] L. H. YOU, H. UGAIL, B. P. TANG, X. JIN, X. Y. YOU, AND J. J. ZHANG, *Blending using ODE swept surfaces with shape control and C1 continuity*, Vis. Comput. 30 (2014), 625–636.
[47] L. H. YOU, X. S. YANG, M. PACHULSKI, AND J. J. ZHANG, *Boundary constrained swept surfaces for modelling and animation*, Comput. Graph. Forum 26(3) (2007), 313–322.
[48] L. H. YOU, X. S. YANG, J. PAN, T.-Y. LEE, S. J. BIAN, K. QIAN, Z. HABIB, A. B. SARGANO, I. K. KAZMI, AND J. J. ZHANG, *Fast character modeling with sketch-based PDE surfaces*, Multimed. Tools Appl. 79 (2020), 23161–23187.
[49] L. H. YOU, X. S. YANG, X. Y. YOU, X. JIN, AND J. J. ZHANG, *Shape manipulation using physically based wire deformations*, Comput. Animat. Virtual Worlds 21 (2010), 297–309.
[50] S.-H. ZHANG, Y. GUO, AND Q.-W. GU, *Sketch2Model: View-aware 3D modeling from single free-hand sketches*, in: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2021), 6000–6017.
[51] Y. ZHUANG, M. ZOU, N. CARR, AND T. JU, *A general and efficient method for finding cycles in 3D curve networks*, ACM Trans. Graph. 32(6) (2013), 1–10.