Efficient sketch-based character modelling with primitive deformer and shape generator

Anonymous cvm submission

Paper ID ****

Abstract

This paper proposes a new sketch-guided and ODE-driven character modelling technique. Our system 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.

1. Introduction

- We develop an efficient sketch-guided and ODE-driven primitive deformer to create a base mesh. It can deform primitives to exactly match the generated silhouette contours. Compared to the existing methods, it automates shape manipulation, avoids tedious manual operations, can deform primitives to match the generated silhouette contours quickly, and is powerful in achieving different shapes of a same primitive.

- We develop a detail generator to add 3D details to the base mesh. Our proposed sketch-guided and ODE-driven local shape creator can create a new local shape to match user's drawn sketches in different views quickly. The image-based detail generator can automatically generate fine details from 2D images.

- Our character modelling system provide editing operations, support users to manipulate the control curves the draw in the creation stage and adding more control curves after the surface has been created. With our developed system, 3D character models with 3D details can be created easily and efficiently.

The rest of the paper is organised as follows. The previous related work is briefly reviewed in Section 2. The system overview of our proposed approach is presented in Section 3. Primitive deformer is examined in Section 4, and Detail generator is investigated in Section 5. Finally, the conclusions and future work are discussed in Section 6.

2. Related Work

2.1. Inflation Technology

Over the past two decades, sketch-based-modelling (SBM) has been widely studied in the computer graphic research community. Several research based systems have been proposed to generate organic models. The surface inflation technique extrudes the polygonal mesh from the skeleton outwards do a good job in modelling stuffed toys. One trend is to inflate freeform surfaces to create simple stuffed animals and other rotund objects in a sketch-based modelling fashion, such as [8, 9, 12]. The Teddy system [8] is the pioneer, it takes closed curves as inputs and find their cordial axes as spline, then wrap the splines with the polygonal mesh. Later, FiberMesh[12] enriched the editing operations for the inflating base mesh. FiberMesh 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 C0 continuity. Sharp control curves appears when operations like cutting, extrusion and tunnel take place. Sharp control curves also serves the creation of creases on surface. However, FiberMesh [12] doesn't allow users to specify the ROI.Based on the study from William[16], the SmoothSketch system addressed the problem of T-junction and cusp, which Teddy fails to solve.

2.2. Primitive Technology

Unlike the inflating systems, primitives-based systems deconstruct the modelling task as a process of creating

a certain set of geometry primitives and further editing on the primitives. The idea of assembling simple geometry primitives to form 3D models is very common in CSG(constructive solid geometry) modelling, related researches including [14, 4]. Shtof et.al [14] introduces a snapping method which helps determining the position and core parameters of several simple geometry primitives. In [4], the authors provides tools for generating 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 will sweep. 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 for man-made objects which simple sweeping surface can meet the quality requirements of the shapes. Structured Annotations for 2D-to-3D Modelling [7], on the other hand, focus on organic modelling. It is a system using two sets of the primitives, one is generalized cylinders, created by the input of a single open sketch stroke representing the spline, and then modified by using simple gestures such as tilt, scale local radius, rotate symmetrical plane, and change cap size; the other primitive is ellipsoid, generated according to the drawn closed ellipse sketch stroke. As the system's name indicates, there are a set pf annotation tools to further editing the surface shape using the annotations such as same-lengths, same angles, alignment and mirror symmetry.

2.3. Mesh editing

A universal disadvantage of above systems is features provided are limited it was not suited to modelling complex and production ready models. This is partly due to their limitations in mesh editing operations, so several sketch-based editing methods have been development to help to improve the performance of sketch-based modelling system. One group of sketch-based editing methods [6, 10] treats individual sketch curve as the reference for deformations like bending. Take the [10] for example, user will draw a first curve as a reference curve to both set the ROI (region of interest) and be a controlling 'skeleton', and a new sketch curve indicating the desired deformation of the reference curve. Then mapping the reference curve to the second curve, the ROI will be deformed. These methods are very useful for simple keyframe animation, but for more small-scaled shape's editing task, studies like [11, 13, 17] are more suitable. These studies take mesh's contours (depend on the viewplan at the time of editing operation is happening) as reference curves, and the newly sketched curves indicating the new positions that the ROI of mesh should be reconstructed to. As they aiming at maintaining the local geometrical details, Laplacian coordinates and correspondingly laplacian surface editing technology are implemented to preserving the features.

3. System Overview

Based on our proposed approach, we have developed a modelling system which is composed of two main components: primitive deformer and detail generator. The primitive deformer is used to deform primitives to exactly match user's generated 2D silhouette contours and create a rough base mesh. The detail generator consists of local shape creator and image-based detail generator. The local shape creator creates a new local shape from four different algorithms. They are local shape creation from: 1) two open silhouette contours in two different view planes, 2) one open and one closed silhouette contour in two different view planes, 3) two open and one closed silhouette contours in three different view planes, and 4) two closed curves. The image-based detail generator generates fine 3D details from 2D images automatically through the SFS algorithm.
The modeling process using our developed system is demonstrated in Figure 1 where a 3D female warrior model is created.

First, 2D character silhouette contours are generated. Users can draw their own silhouette contours directly or input their selected sketches into our developed system. If the selected sketches are input into our developed system, a further process may be required to extract the 2D silhouette contours from the input sketches. For the example demonstrated in Figure 1, a 2D female warrior sketch shown in Figure 1a is input. Then, users can extract 2D silhouette contours from the input sketch as shown in Figure 1b. After that, proper primitives are selected and placed to align with the corresponding silhouette contours through purely geometric transformations as shown in Figure 1c. Since the silhouette contours of the primitives do not match the generated 2D silhouette contours of the 2D female warrior, the primitive deformer developed from sketch-guided and ODE driven deformations described in Section 4 is applied to deform the primitives to exactly match the corresponding 2D silhouette contours and create a rough 3D base mesh depicted in Figure 1d.
Once a 3D base mesh model is obtained, the detail generator described in Section 5 is employed to add 3D details to the 3D base mesh. First, the local shape creator developed in Subsection 5.2 is used to add local shapes and smoothly connect primitives together as demonstrated in Figure 1e. After that, we want to add fine details such as a dragon to the female warrior model. Since creating fine 3D details is not an easy task, we apply the image-based detail generator described in Subsection 5.3 to achieve the complicated 3D details shown in Figure 1f.
In the following two sections, we will introduce in detail the primitive deformer and the detail generator, respectively. Some examples will be presented to demonstrate their applications.

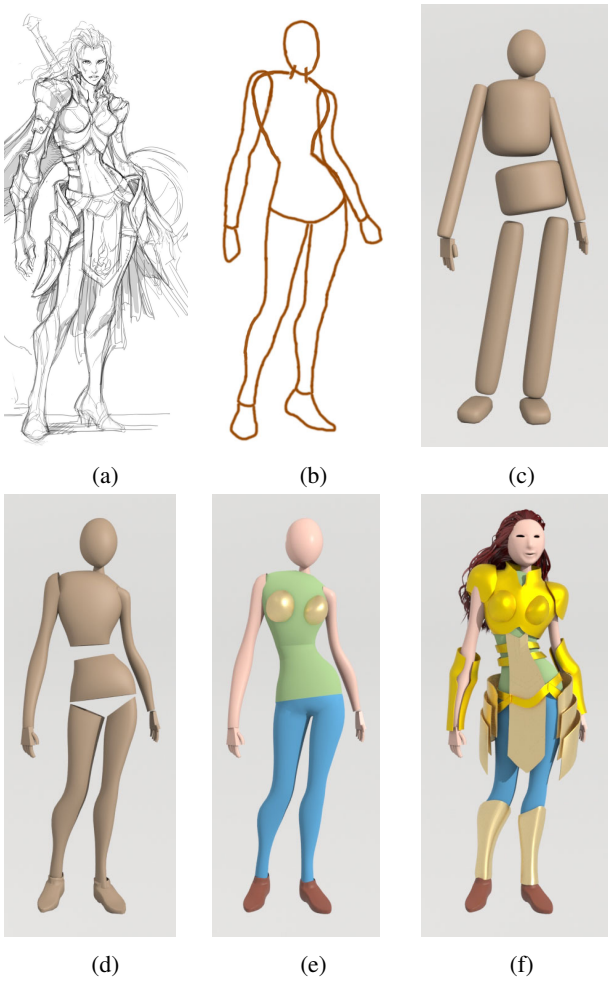


Figure 1: Quick creation of a 3D female warrior model: (a) 2D female warrior sketch, (b) 2D silhouette contours, (c) base mesh without primitive deformations, (d) base mesh by deforming primitives to match generated 2D silhouette contours or adding local shapes created from 2D silhouette contours, (e) base mesh by adding blending surfaces to smoothly connect deformed primitives, (f) detail generation through local shape creation and shape from shading (Sketch by EngKit Leong)

4. Primitive deformer

As shown in Figure 1c, the base mesh of the female warrior without primitive deformations cannot exactly match the generated 2D silhouette contours of the female warrior. In order to tackle this problem, in this section, we develop a primitive deformer. In the subsections below, we first introduce the interface of the primitive deformer in Subsection 4.1. Then, we discuss the algorithm of the primitive deformer in Subsection .

4.1. User interface of primitive deformer

The user interface of our developed primitive deformer uses four windows. 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 display primitives and the user-drawn 2D silhouette contours for these primitives. If required, the user can edit the generated 2D silhouette contours in the upper right window. The deformed primitives are shown in the bottom windows where the left is from the front view and the right is from the side view.

Taking the left leg of the female warrior shown in Figure 2 as an example, the primitive of the left leg is shown in the upper left window. It and the user-drawn 2D silhouette contours are depicted in the upper right window. Our proposed primitive deformer described in Subsection 4.2 deforms the primitives of the 3D base mesh to exactly match the generated 2D silhouette contours and create a deformed 3D base mesh with primitive deformations, and depicted the deformed 3D base mesh in bottom windows.

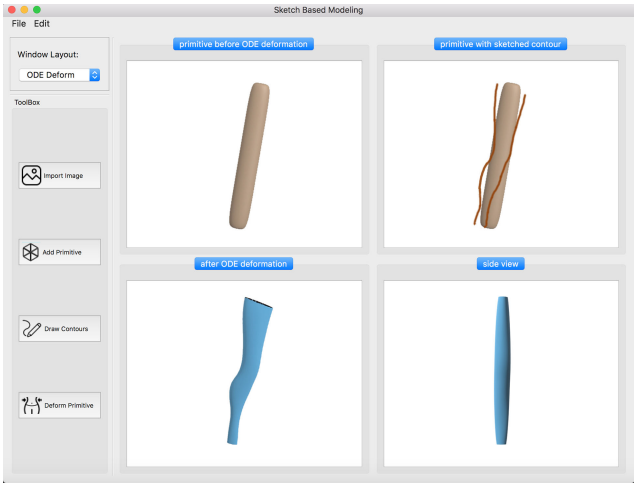
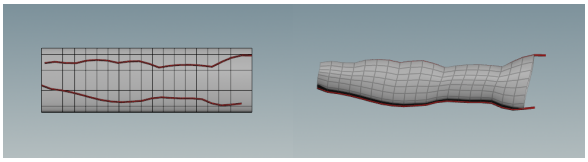


Figure 2: Interface for primitive deformer: (a) 3D base mesh of the left leg of the female warrior without primitive deformations, (b) 2D silhouette contours of the female warrior leg sketch and primitive, (c) and (d) front view and side view of the 3D left leg base mesh after primitive deformations

4.2. Algorithm of primitive deformer

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 3 to demonstrate the algorithm of our proposed primitive deformer and

how it deforms a 3D primitive to match the 2D silhouette contours.



(a) (b)

Figure 3: Primitive deformer: a) female warrior arm represented with a cylinder and its 2D silhouette contour, b) deformed shape of the cylinder

Figure 3a depicts an arm 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 3b shows how the cylinder is deformed with the algorithm developed below to match the 2D silhouette contour exactly. This is achieved by a sketch-guided and ODE-drive primitive deformer. It is developed from a simplified version of the Euler-Lagrange PDE (partial differential equation) which is widely used in physically-based surface deformations and briefly introduced below.

As discussed in [1], the main requirement for physically-based surface deformations is an elastic energy which considers locally stretching for solid objects plus bending for two-manifold surfaces called thin-shells. When a surface $\mathbb{S} \subset \mathbb{R}^3$ parameterized by a function $\mathbf{P}(\mathbf{u},\mathbf{v}) : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{S} \subset \mathbb{R}^3$is deformed to a new shape $\mathbb{S}'$ through adding a displacement vector $\mathbf{d}(\mathbf{u},\mathbf{v})$ to each point $\mathbf{P}(\mathbf{u},\mathbf{v})$, the change of the first and second fundamental $I(u,v), \Pi(u,v) \in \mathbb{R}^{2\times 2}$,forms in differential geometry [5] yields a measure of stretching and bending descried by [15]

$$E_{shell}(S') = \int_\Omega k_s \|I' - I\|_F^2 + k_b \|\Pi' - \Pi\|_F^2 dudv \quad (1)$$

Where ,I',Π' are the first and second fundamental forms of the surface $\mathbf{S}'$, $\|.\|$ 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 which is non-linear and computationally too 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 $\mathbf{d}(\mathbf{u},\mathbf{v})$ (Celniker, 1991), (Welch, 1992) , i. e.,

$$\begin{aligned}\tilde{E}_{shell}(d) = \int_\Omega & k_s(\|d_u\|^2 + \|d_v\|^2) \\ & + k_b(\|d_{uu}\|^2 + 2\|d_{uv}\| + \|d_{vv}\|^2)dudv\end{aligned} \quad (2)$$

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

$$-k_s \triangle d + k_b \triangle^2 d = 0 \quad (3)$$

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

$$\begin{aligned}\triangle d &= div \nabla d = d_{uu} + d_{vv} \\ \triangle^2 d &= \triangle(\triangle d) = d_{uuuu} + 2d_{uuvv} + d_{vvvv}\end{aligned} \quad (4)$$

Using the sketched 2D silhouette contours shown in Figure 3a to change the shape of the primitive can be transformed into generation of a sweeping surface which passes through the two sketched 2D silhouette contours. The generator creating the sweeping surface is a curve of the parametric variable u only, and the two silhouette contours are trajectories. If Equation (3) is used to describe the generator, the parametric variable in Equation (3) drops, and we have $d_{vv} = 0$ and $d_{vvvv} = 0$. Substituting and into Equation (3), we obtain the following simplified version of the Euler-Lagrange PDE (3)which is actually a vector-valued ordinary differential equation

$$k_b \frac{\partial^4 d}{\partial u^4} - k_s \frac{\partial^2 d}{\partial u^2} = 0 \quad (5)$$

As pointed out in [3] and [2] , the finite difference solution of ordinary differential equations is very efficient, we here investigate such a numerical solution of Equation (4).

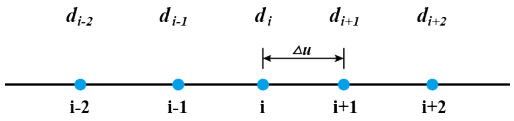


Figure 4: Typical node i for the finite difference approximations of derivatives

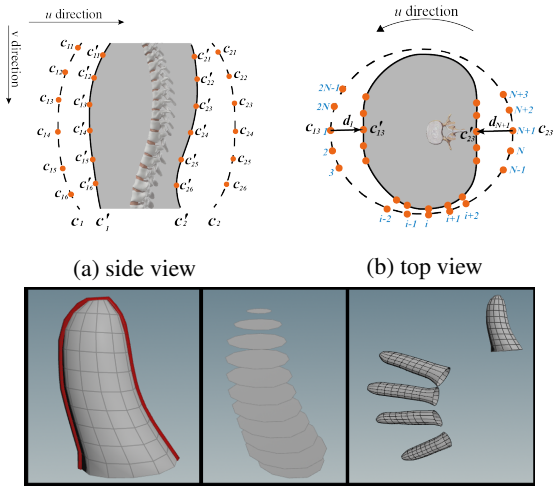
For a typical node shown in Figure 4, the central finite difference approximations of the second and fourth order derivatives can be written as [3]

$$\begin{aligned}\frac{\partial^2 d}{\partial u^2}|_i &= \frac{1}{\triangle u^2}(d_{i+1} - 2d_i + d_{i+1}) \\ \frac{\partial^4 d}{\partial u^4}|_i &= \frac{1}{\triangle u^4}[6d_i - 4(d_{i-1} + d_{i+1}) + d_{i-2} + d_{i+2}]\end{aligned} \quad (6)$$

Introducing Equation (6) into Equation (5), the following finite difference equation at the representative node i can be written as:

$$\begin{aligned}(6k_b + 2k_s h^2)d_i + k_b d_{i-2} + k_b d_{i+2} \\ -(4k_b + k_s h^2)d_{i-1} - (4k_b + k_s h^2)d_{i+1} = 0\end{aligned} \quad (7)$$

For character models, the 3D shape defined by two silhouette contours is closed in the parametric direction u as indicated in Figure 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 5b 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 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 5a and depicted it as a dashed curve in Figure 5(b). 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 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 $\mathbf{c'(u)} = \mathbf{d(u)} + \mathbf{c(u)}$.



(a) side view (b) top view

(c) the making of the fingers

Figure 5: Finite difference nodes for local shape manipulation from sketches in different view planes and the deformed 3D finger models

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 interval as indicated in Figure 5. The displacement difference at node 1 and node $\mathbf{N}$ is known, i. e. $\mathbf{d_1} = \mathbf{c'_{13}} - \mathbf{c_{13}}$ and $\mathbf{d_{N+1}} = \mathbf{c'_{23}} - \mathbf{c_{23}}$.
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 (7). Substituting $\mathbf{d_1} = \mathbf{c'_{13}} - \mathbf{c_{13}}$ into these equations, we obtain the finite difference equations for the nodes 2, 3, $2N-1$ and $2N$
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 (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
For all other nodes 3, 4, 5,..., $N-3$, $N-2$ and $N+4$,$N+5$,..., $2N-3$, $2N-2$, the finite difference equations are the same as Equation (7). For these nodes, there are $2N-5$ finite difference equations. Plus the 8 finite difference equations at node 2, 3, $N-1$, N , $N+2$, $N+3$, $2N-1$ and $2N$, we get $2N-2$linear algebra equations which can be solved to determine the unknown constants $\mathbf{d_2}$, $\mathbf{d_3}$, ...,$\mathbf{d_{N-1}}$, $\mathbf{d_N}$, $\mathbf{d_{N+2}}$, $\mathbf{d_{N+3}}$,..., $\mathbf{d_{2N-1}}$, and $\mathbf{d_{2N}}$. Adding the $\mathbf{d_i}$ ($i = 1, 2, \ldots, 2N-1, 2N$) to the original curve $\mathbf{c(u)}$, we obtain the deformed curve $\mathbf{c'(u)}$, and depict it as a solid curve in Figure 5b. 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.
Taking a finger shown in Figure 5c as an example, the left image shows the silhouette contours. It is used deform a cylinder into the middle 3D model (the second image from the left). The cross-section shapes of the finger model are depicted in the third image from the left. The rightmost image shows the five finger models created with the above method.
With the primitive deformer developed above, we deform the primitive of the 3D base mesh shown in Figure 1c, obtain a deformed 3D base mesh, and depicted it in Figure 1d. It is clear that deformed primitives have matched the generated 2D silhouette contours exactly.

5. Detail generator

In order to add 3D details to 3D base mesh created from the primitive generator, we develop a detail generator consisting of a local shape creator and an image-based detail generator. In what follows, we first introduce the user interface of our developed detail generator in Subsection5.1. Then we discuss the local shape creator in Subsection5.2.

5.1. User interface of detail generator

The user interface of our developed detail generator uses different windows. They are used to respectively treat lo-

cal model creation from sketches in different view planes, and 2D image-based detail generation. For local model creation, four windows are used to help provide depth information and control the shape of 3D local details to be created. As shown in Figure 7, the two top windows and left bottom window allow users to sketch 2D silhouette contours from two or all of three orthotropic views: front view, side view, and top view in the local coordinates of a local shape, or project a 3D model to the three orthotropic views to modify them. The right bottom window is used to display the created 3D models.

Taking the upper body model as an example, we first sketch

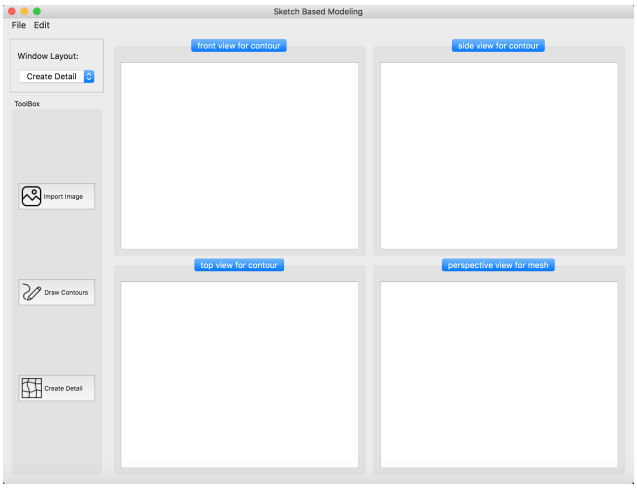


Figure 6: Interface for sketching three orthotropic silhouette contours and viewing generated 3D models

a 2D silhouette contour in the front view window as indicted in the upper left part in Figure 6. The sketched silhouette contour is also displayed in the side view window, which locates at the upper right part of the Figure 6. Then we sketch the second 2D silhouette contour in the side view window which intersects with the 2D silhouette contour in the front view window. Both the 2D silhouette contours sketched in the front and side view windows are displayed in the top view window in the lower left part of Figure 6. In the top view window, we sketch the third 2D silhouette contour shown which intersects with the two sketched silhouette contours respectively obtained from the front view and the side view. Once a 3D model is created from the two or three sketched silhouette contours, it is displayed in the right bottom window.

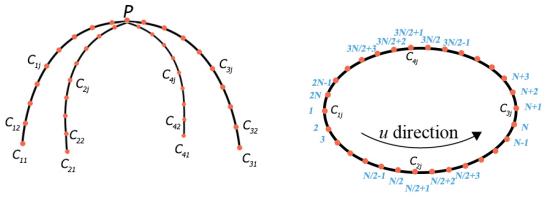
For 2D image-based detail generation, 2D images are input into the left window. After carrying out the calculations of shape-from-shading, the generated 3D models are displayed in the right windows. Users can input a whole character model into the window and perform the operations of adding the generated models to the whole character model.

5.2. local shape creator

Although a 3D regional model can be created from two sketched silhouette contours in the same view plane as discussed in Subsection 4.2, the shape changes in the depth direction may not be our expected ones. In order to allow users to control shape changes in the depth direction and smoothly connect primitives together, we develop a new local shape creator to create a 3D model from two or three silhouette contours in different view planes. It is developed from the four algorithms which can create a model from two open sketches, one open and one closed sketches, two open and one closed sketches, and two closed curves, respectively. In the following four subsections, we will investigate these four algorithms.

5.2.1 Algorithm for two open silhouette contours in two different view planes

For 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 from 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 7. 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) side view (b) top view

Figure 7: Finite difference nodes for the algorithm for 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 $\mathbf{c}(\mathbf{u})$ shown in Figure 7b passing through the four corresponding points $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$, $\mathbf{c_{3j}}$ and $\mathbf{c_{4j}}$, and shown in Figure 7(a) and (b). We uniformly divide the domain of the parametric variable u corresponding to the closed curve (generator) $\mathbf{c}(\mathbf{u})$ into $2N$ equal intervals. The nodes corresponding to $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$, $\mathbf{c_{3j}}$ and $\mathbf{c_{4j}}$ are 1, $N/2+1$, $N+1$ and $3/2N+1$, respectively

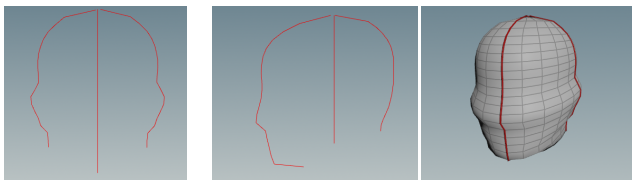
as demonstrated in Figure 7(b). Here, the selection of N should ensure that is an even number and $N \geqslant 4$.

The finite difference equations for the nodes 4, 5, ..., $N/2-3$, $N/2-2$; $N/2+4$, $N/2+5$, ..., $N-3$, $N-2$; $N+4$, $N+5$, ..., $3N/2-3$, $3N/2-2$; $3N/2+2$, $3N/2+3$, ..., $2N-3$, and $2N-2$ are the same as Equation (7).

The finite difference equations for the nodes 2, 3, $N/2-1$, $N/2$, $N/2+2$, $N/2+3$, $N-1$, N, $N+2$, $N+3$, $3N/2-1$, $3N/2$, $3N/2+2$, $3N/2+3$, $2N-1$, and $2N$ can be obtained by introducing the corresponding one of $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$, $\mathbf{c_{3j}}$ and $\mathbf{c_{4j}}$ into Equation (7) to replace d_1 , $d_{N/2+1}$, d_{N+1} and $d_{3N/2+1}$ accordingly.

Putting all obtained finite difference equations together, we obtain all the unknown constants $\mathbf{d_2}$, $\mathbf{d_3}$, ..., $\mathbf{d_{N/2-1}}$, $\mathbf{d_{N/2}}$, $\mathbf{d_{N/2+2}}$, $\mathbf{d_{N/2+3}}$, ..., $\mathbf{d_{N-1}}$, $\mathbf{d_N}$, $\mathbf{d_{N+2}}$, $\mathbf{d_{N+3}}$,..., $\mathbf{d_{3N/2-1}}$, $\mathbf{d_{3N/2}}$, $\mathbf{d_{3N/2+2}}$, $\mathbf{d_{3N/2+3}}$, ..., $\mathbf{d_{2N-1}}$ and d_{2N} . They together with the known four points $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$, $\mathbf{c_{3j}}$ and $\mathbf{c_{4j}}$ are used to define the generator.

Repeating the above treatment for all other points on the four curves $\mathbf{c_1}$, $\mathbf{c_2}$, $\mathbf{c_3}$ and $\mathbf{c_4}$, we obtain the generators at the other positions. With these generators, we create local 3D shapes.



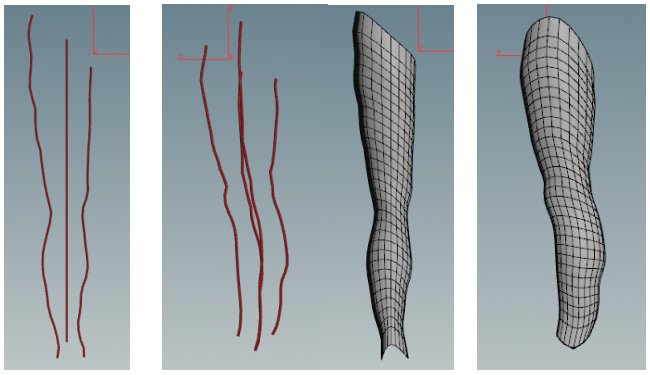
(a) (b) (c)

Figure 8: Primitive generator: a) 2D silhouette contours of the male head in front view, b)-2D silhouette contours of the male head in front view, c) generated male head

A head model is first used to demonstrate the above method Figure 8. Figure 8a and Figure 8b show two open head silhouette contours in two different view plans. Figure 8c depicted the created 3D head model.

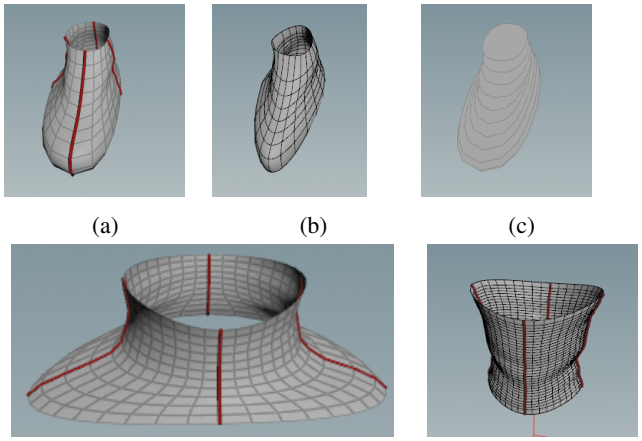
The above method can be directly extended to deal with local shape creation from four disconnected silhouette contours. The silhouette contours of the leg model in the front and side views are shown in Figure 9a and Figure 9b, respectively. The corresponding four points on the four silhouette contours are used to define a cross-section curve. All of these cross section curves are used to generate the 3D leg model with the front and side views in Figure 9c and Figure 9d, respectively.

More modeling examples are given in Figures 10. Figure 10a shows the four silhouette contours of a foot model in the front and side views. Figure 10b depicts the 3D foot model created from the four silhouette contours. Figure 10c shows the cross-section curves. Figures 10d and 10e



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

Figure 9: leg model creation from two open silhouette contours in two different view planes



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

Figure 10: Creation of 3D foot, neck, and torso models from two open silhouette contours in two different view planes

give 3D neck model and torso model generated from the above method.

5.2.2 Algorithm for one open and one closed silhouette contours in two different view planes

For creation of a local 3D model from one open and one closed silhouette contours in two different view plans as shown in Figure 11, the intersecting points $\mathbf{p_1}$ and $\mathbf{p_3}$ between the open silhouette contour and the closed silhouette contour divide the closed silhouette contour into two curves. Then we find the middle points $\mathbf{p_2}$ and $\mathbf{p_4}$ of the two curves. These four points $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$ and $\mathbf{p_4}$ divide the closed silhouette contour into four curves $\mathbf{c_1}$, $\mathbf{c_3}$, $\mathbf{c_4}$ and $\mathbf{c_6}$. Next, we find the middle point $\mathbf{p}$ of the

open silhouette contour which divides the open silhouette contour into two curves $\mathbf{c_2}$ and $\mathbf{c_5}$. With this treatment, the creation of the local 3D detail model is changed into creation of two sweeping surfaces: one is defined by the three curves $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$, and the other is defined by the three curves $\mathbf{c_4}$, $\mathbf{c_5}$ and $\mathbf{c_6}$.

Since the creation process of the two weeping surfaces is the same, we take the creation of the sweeping surface defined by the three curves $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$ to demonstrate the process.

For each of the three curves $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$ we uniformly divide it into $2N$ equal intervals, and use $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$ and $\mathbf{c_{3j}}(\mathbf{j} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{J})$ to respectively indicate the nodes on the three curves.

The sweeping surface can be generated by sweeping a generator $\mathbf{c}(\mathbf{u})$ 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 determination of the generator at different positions along the curve.

In order to create the generator $\mathbf{c}(\mathbf{u})$ 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 $\mathbf{c}(\mathbf{u})$ into 2N equal intervals, and get the nodes $1, 2, \ldots, 2N-1$, $2N$ and $2N+1$. Since the nodes 1, $N+1$, and $2N+1$ are on the curves $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$, 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 positional continuation only, the vector-valued first derivative $\mathbf{T_{1j}}$ on the curve $\mathbf{c_1}$ and $\mathbf{T_{3j}}$ on the curve $\mathbf{c_3}$ can be estimated from the three points $\mathbf{c_{1j}}$, $\mathbf{c_{2j}}$ and $\mathbf{c_{3j}}$ by first constructing a curve passing through the three points and then calculating the vector-valued first derivative of the constructed curve at the points $\mathbf{c_{1j}}$ and $\mathbf{c_{3j}}$. Once the vector-valued first derivatives $\mathbf{T_{1j}}$ and $\mathbf{T_{3j}}$ are obtained. The nodes 0 and $2N+2$ can be determined by the following finite difference approximation.

$$\mathbf{T_{1j}} = \frac{\mathbf{d_2} - \mathbf{d_0}}{2\triangle u}, \mathbf{T_{3j}} = \frac{\mathbf{d_{2N+2}} - \mathbf{d_{2N}}}{2\triangle u} \tag{8}$$

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

$$\begin{aligned} \mathbf{d_0} &= \mathbf{d_2} - 2\mathbf{T_{1j}}\triangle u \\ \mathbf{d_{2N+2}} &= \mathbf{d_{2N}} + 2\mathbf{T_{3j}}\triangle u \end{aligned} \tag{9}$$

The finite difference equations for nodes 4, 5, ..., $N-3$, $N-2$ and $N+4, N+5, \ldots, 2N-3, 2N-2$ are the same as Equation (7).

The finite difference equations for the nodes 2, 3, $N-1$, N , $N+2$, $N+3$, $2N-1$ and $2N$ can be obtained by introducing the above equation and $\mathbf{d_1} = \mathbf{c_{1j}}$, $\mathbf{d_2} = \mathbf{c_{2j}}$ and $\mathbf{d_3} = \mathbf{c_{3j}}$ into Equation (7).

Solving the finite difference equations for all the inner nodes $2, 3, \ldots, 2N-1$ and $2N$, we obtain all the unknown constants d_2 , d_3 , ..., d_{2N-1} and d_{2N}. They together with the three known points c_{1j} , c_{2j} and 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 local detail 3D model is created and depicted in (b) of Figure 12.

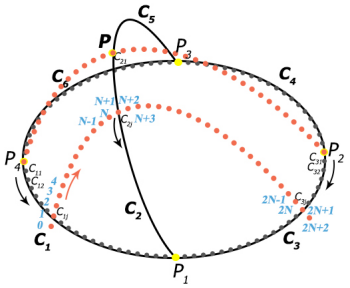


Figure 11: Finite difference nodes for Algorithm for one open and one closed silhouette contours in two different view planes

5.2.3 Algorithm for two open and one closed silhouette contours in three different view planes

By introducing depth information, users can have more power to control the shape of the 3D model to be created as demonstrated by the algorithms given in Subsections 5.2.1 and 5.2.2. Users can further control the shape of 3D models by using one or more silhouette contours in other view planes. In this subsection, we discuss construct a 3D detail model from two open and one closed silhouette contours in three orthotropic view planes.

As shown in Figure 13, the task of creating a local 3D detail model passing through two open and one closed silhouette contours can be transformed into constructing 4 sweeping surfaces encircled by the curves , , , and , respectively.

Figure 13: Creation of a 3D local detail model from two open and one closed silhouette contours Since the construction algorithm for the 4 sweeping surfaces is the same, we take the sweeping surface encircled by the three curves , and

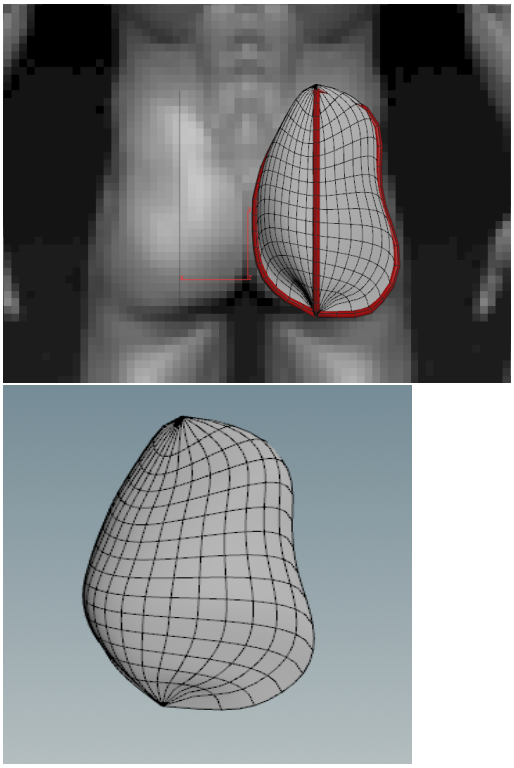


Figure 12: Algorithm for two open and one closed silhouette contours in three different view planes

to demonstrate the construction algorithm. This algorithm will require reconstruction of the curves and to get additional information called sculpting forces for generating the sweeping surface. In order to reconstruct the curves and accurately, we modify the vector-valued ordinary differential equation (e) by introducing a sculpting force and obtain Accordingly, the finite difference equation for the above ordinary differential equation becomes The reconstruction algorithm for the curve is exactly same as that of the curve . Here we take the reconstruction of the curve to demonstrate the algorithm. Similar to the previous treatment, we uniformly divide the domain of the parametric variable for the curve into 2N equal intervals. The vector-valued first derivatives of the curves at the node 1 and 2N+1 can be determined from the curve and indicated by and . When writing the finite difference equations for the inner nodes 2, and 2N, the node 0 beyond the boundary node 1 and the node 2N+2 beyond the boundary node 2N+1 will be involved. We can use the method given in Subsection 5.2.2 to determine the nodes 0 and 2N+2 from the vector-valued first derivatives and through For each of the inner nodes 2, 3, ?, 2N-1, 2N, we can write a finite difference equation. Since the coordinate values for all the nodes on the curve are known, we can calculate the sculpting force from these equations,. We denote these sculpting forces as . With the same method, we can obtain the sculpting forces acting on the curve . The curves and can be regarded as generators. The generation of the sweeping surface defined by the three curves , and is to sweep the generator from the curve to the curve along the curve and the point . In order to determine the shape of the generator at different positions along the curve , we uniformly divide the curve into equal intervals and obtain the nodes where and are the intersecting points between the curves and and between and . Then we determine the shape of the generator between the node and the point . With the same treatment, we divide the domain of the parametric variable for the generator between the node and the point into 2N equal intervals (the node on the curve is the node 0 on the generator between the node and the point). When sweeping the curve along the curve to the curve , the sculpting force acting at the node of the curve is gradually changed to the sculpting force acting at the node of the curve . Here we use a linear interpolation to describe the gradual change and obtain the sculpting force below acting at the node of the generator between the node and the point where is the length from the point to the node and is the length from the point to the point . When sweeping the curve along the curve to the curve , the vector-valued first derivatives of the curve at the points and are also gradually changed to the vector-valued first derivatives of the curve at the points and . The same linear interpolation is used to describe such a gradual change and determine the vector-valued first derivatives and of the generator at the node and the point Having known the sculpting forces at all the inner nodes 2, 3, ?, 2N-1, 2N, and the coordinate values at the boundary nodes 0 and 2N+1 and the nodes 0 and 2N+2 beyond the boundary nodes, we can write the finite difference equations for all the inner nodes where the finite difference equations for the nodes 4, 5,?, 2N-3, 2N-2 are the same as Equation (11) with the sculpting force being calculated by Equation (13), and the finite difference equations for the nodes 2, 3, 2N-1, and 2N are given in Appendix D. Solving all the finite difference equations, we obtain all the unknown constants , , ?, , and . They together with the two known points and are used to create the generator. With the same method, we obtain all the curves of the generator at the positions. They are used to create the sweeping surface. Figure 14: palm model creation from two open and one closed silhouette contours in three different view planes

With the above method, we draw a closed palm contour and two open curves shown in Figure 14(a). They divide the palm into four regions. One surface is created each of the four regions. Figure 14(b) depicts the palm model consisting of the four surfaces. The above method can be also 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 Figures 15(a) and (b), and a sweeping surface is encircled by four curves. With this extension, we created 3D shoulder and hip models and depicted them in Figures 15(a) and (b), respectively. Figure 15: palm model creation from two open and one closed silhouette contours in three different view planes

5.2.4 Algorithm for two closed curves

This algorithm is to create smooth transition surfaces between two disconnected primitives. Users can interactively draw two closed curves on two disconnect primitives or use two planes to intersect the two primitives to obtain two boundary curves of the transition surface. In order to obtain smooth transition between the two primitives and the transition surface, the tangents at the two boundary curves are obtained from the two primitives. With the two boundary curves and the tangents at the two boundary curves as the boundary conditions, the ODE-based surface blending method introduced in [34] is used to create the smooth transition surface which smoothly connects two primitives together. Figure 17 gives such an example. Figure 17: Creation of smooth transition surfaces between primitives

By adding local shapes and smooth transition surfaces, the base mesh shown in Figure 1(d) is changed into that depicted in Figure 1(e).

Compare the following:

```
$conf_a$                $conf_a$
$\mathit{conf}_a$       $\mathit{conf}_a$
```

See The TEXbook, p165.

The space after *e.g.*, meaning "for example", should not be a sentence-ending space. So *e.g.* is correct, *e.g.* is not. The provided `\eg` macro takes care of this.

When citing a multi-author paper, you may save space by using "et alia", shortened to "*et al.*" (not "*et. al.*" as "*et*" is a complete word.) However, use it only when there are three or more authors. Thus, the following is correct: " Frobnication has been trendy lately. It was introduced by Alpher [**?**], and subsequently developed by Alpher and Fotheringham-Smythe [**?**], and Alpher *et al.* [**?**]."

This is incorrect: "... subsequently developed by Alpher *et al.* [**?**] ..." because reference [**?**] has just two authors. If you use the `\etal` macro provided, then you need not worry about double periods when used at the end of a sentence as in Alpher *et al.*

For this citation style, keep multiple citations in numerical (not chronological) order, so prefer [**?**, **?**, **?**] to [**?**, **?**, **?**].

6. Conclusions and Future work

For now in the modelling stage of our system, all our control curves are creating smooth surfaces with C2 continuity, and the result template mesh looks pleasing as it is. In our future work, we will investigate the potentials of creases in more detailed character sculpting.

Name	Performance
A	OK
B	Bad
Ours	Great

Table 1: An example for using tables.

Acknowledgement

This research is supported the PDE-GIR project which has received funding from the European Union?s Horizon 2020 research and innovation programme under the Marie Sk?odowska-Curie grant agreement No 778035.

6.1. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [**?**]. Where appropriate, include the name(s) of editors of referenced books.

6.2. Illustrations, graphs, and photographs

When placing figures in LaTeX, it's almost always best to use `\includegraphics`, and to specify the figure width as a multiple of the line width as in the example below

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]
                {myfile.eps}
```

References

[1] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1):213–230, 2008. 4

[2] E. Chaudhry, S. Bian, H. Ugail, X. Jin, L. You, and J. J. Zhang. Dynamic skin deformation using finite difference solutions for character animation. *Computers & Graphics*, 46:294–305, 2015. 4

[3] E. Chaudhry, L. You, X. Jin, X. Yang, and J. J. Zhang. Shape modeling for animated characters using ordinary differential equations. *Computers & Graphics*, 37(6):638–644, 2013. 4

[4] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6):195, 2013. 2

[5] M. P. Do Carmo, G. Fischer, U. Pinkall, and H. Reckziegel. Differential geometry. In *Mathematical Models*, pages 155–180. Springer, 2017. 4

[6] G. Draper and P. K. Egbert. A gestural interface to free-form deformation. In *graphics interface*, volume 2003, pages 113–120, 2003. 2

[7] Y. Gingold, T. Igarashi, and D. Zorin. Structured annotations for 2d-to-3d modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, page 148. ACM, 2009. 2

[8] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416. ACM Press/Addison-Wesley Publishing Co., 1999. 1

[9] O. A. Karpenko and J. F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph.*, 25(3):589–598, July 2006. 1

[10] Y. Kho and M. Garland. Sketching mesh deformations. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 147–154. ACM, 2005. 2

[11] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Shape Modeling Applications, 2004. Proceedings*, pages 181–190. IEEE, 2004. 2

[12] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM transactions on graphics (TOG)*, 26(3):41, 2007. 1

[13] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2007 courses*, page 42. ACM, 2007. 2

[14] A. Shtof, A. Agathos, Y. Gingold, A. Shamir, and D. Cohen-Or. Geosemantic snapping for sketch-based modeling. In *Computer graphics forum*, volume 32, pages 245–253. Wiley Online Library, 2013. 2

[15] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *ACM Siggraph Computer Graphics*, volume 21, pages 205–214. ACM, 1987. 4

[16] L. R. Williams and D. W. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural computation*, 9(4):837–858, 1997. 1

[17] J. Zimmermann, A. Nealen, and M. Alexa. SilSketch: Automated Sketch-Based Editing of Surface Meshes. In M. van de Panne and E. Saund, editors, *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association, 2007. 2