공학박사학위논문

Sweep-based Approach to Three-Dimensional Shape Deformation

3차원 형상변형을 위한 스윕기반의 접근

2007년 2월

서울대학교 대학원
컴퓨터공학부
윤 승 현

Sweep-based Approach to Three-Dimensional Shape Deformation

3차원 형상변형을 위한 스윕기반의 접근

지도교수 김 명 수

이 논문을 공학박사 학위논문으로 제출함

2006 년 10월

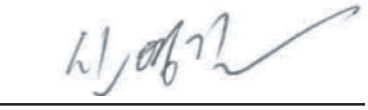
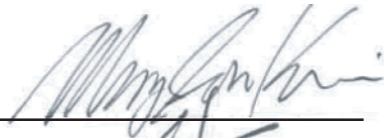
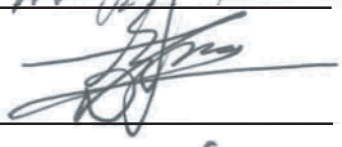
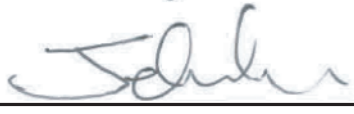
서울대학교 대학원

컴퓨터공학부

윤 승 현

윤승현의 공학박사 학위논문을 인준함

2006년 12월

위 원 장	신 영 길	
부위원장	김 명 수	
위 원	고 형 석	
위 원	이 제 희	
위 원	이 인 권	

Abstract

We present a sweep-based approach to modeling and deformation of three-dimensional objects. We approximate the deformable parts of an object using sweep surfaces. The vertices on the object boundary are bound to the sweep surfaces and then follow their deformation. For practical applications, we apply this approach to 3D human body modeling and deformation, and then extend it to the freeform deformation and to the elastic deformation of three-dimensional objects.

For human modeling and deformation, a simple skeleton structure is extracted from user-specified feature points; after that, control sweep surfaces that approximate human arms, legs and torso are automatically generated. The vertices on the model are bound to nearby sweep surfaces and follow the deformation of the sweep surfaces as the model bends and twists its arms, legs, spine and neck. Anatomical features including bone-protrusion, muscle-bulge, and skin-folding are also supported.

We then extend the sweep-based approach to the freeform deformation of three-dimensional objects. The user selects deformable parts of an object and the corresponding control sweep surfaces are generated by interpolating key cross-sections. The user can deform the shape of an object by controlling the underlying sweep surfaces. Several sweep surfaces of deformable parts can be organized into a hierarchy so that they interact with each other in a controlled manner. The sweep-based freeform deformation technique also provides various important advantages, including volume

preservation and shape transfer.

We further extend the sweep-based approach to the elastic deformation of three-dimensional objects by employing an elastic sweep surface. An elastic sweep surface is constructed by interpolating key cross-sections whose positions, orientations and boundary shapes are determined by physical simulations of simple mass-spring systems. As the user controls the underlying sweep surfaces or applies an external force, the corresponding parts of an object elastically change their shapes. An external force is decomposed into a rotational force and a radial one so that different deformation effects are achieved by applying each of them selectively.

We also apply our sweep-based approach to geometric models represented as point clouds. For this purpose, we approximate a point cloud using a surface displaced from a manifold. A control mesh is generated from a point cloud and a local patch for each vertex of the control mesh is constructed. The original points are then projected on to nearby local patches and their displacements are adjusted so that the final displaced surface approximates the point cloud with high precision. Sweep-based approach is then applied to the control mesh. As the control mesh deforms, the displaced surface is reconstructed and the corresponding smooth shape deformations follow. In experimental results, we demonstrate the effectiveness of the sweep-based approach and show that our approach provides an excellent control mechanism for deforming three-dimensional objects.

keywords: Sweep Surface, Human Body Deformation, Freeform Deformation, Elastic Deformation, Manifold, Displaced Surface.

Student Number: 2001-21510

Contents

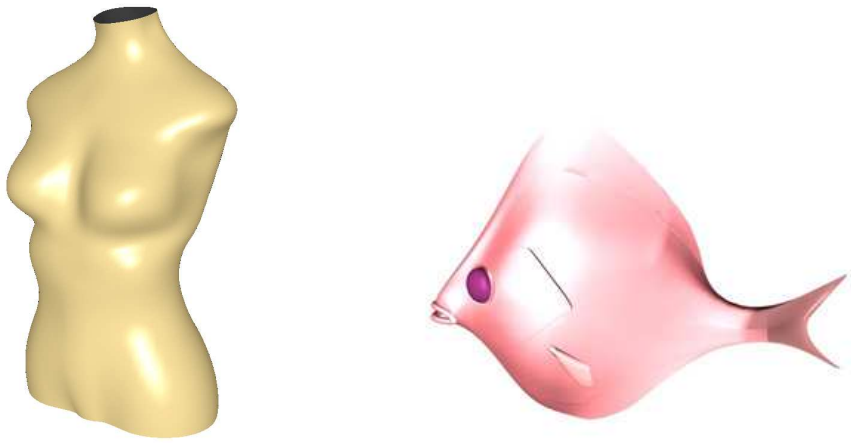
List of Figures

Chapter 1

Introduction

Sweeps are a procedural modeling technique for representing three-dimensional tubular objects [25]. When a 2D area or a 3D volume moves along a prescribed trajectory in the space, the swept volume of this moving object is generated and a sweep surface is then defined as its boundary. Such a procedural description of an object is efficient in the sense that it requires only the specification of the trajectory and that of the moving object. The surfaces of extrusion and revolution which are found in a surface of solid modeling systems can be considered as simple sweep surfaces. Extrusion is a translational sweep surface whose trajectory is a straight line and revolution is a rotational sweep surface along a circular trajectory.

Jüttler and Wagner [43] developed a motion interpolation technique and applied this technique to the design of sweeps with rational B-spline motions. They decomposed a motion into translational and rotational components, and combined them to form a matrix-valued rational spline curve. Under a rational motion, a 2D cross-section generats a rational sweep surface represented in a NURBS form. Figure 1.1



(a) (b)

Figure 1.1: Sweep surfaces generated by a rational motion: (a) female body shape, (b) fish shape composed of sweep surfaces [11]

shows various 3D shapes generated using this technique combined with a conventional trimming method.

In this thesis, we employ sweeps as a shape control tool and present a sweep-based approach to shape modeling and deformation of three-dimensional objects. Our sweep-based approach is distinguished from previous approaches [1, 18, 50, 56, 57, 64, 75] in the sense that it provides an intuitive control mechanism and supports various geometric constraints during a deformation. Moreover, our approach is independent of geometric representations and can easily be extended to a physical setting. A specialized technique is proposed for 3D human model and then extended to the freeform deformation and to the elastic deformation of arbitrary objects.

Virtual human models play an important role in computer animation, including real-time applications in virtual reality and computer games. Advanced anatomy-based simulation techniques produce very realistic modeling and deformation of vir-

tual humans [5, 62, 78]. However, they do not support real-time applications, for which vertex blending is now widely accepted as the method of choice [1, 50, 57, 75]. We propose a sweep-based approach as an alternative that combines the advantages of these existing techniques. Important anatomical features are supported, while computational cost remains about the same as vertex blending.

This work was motivated by the recent results of Allen et al. [2, 3] in which 3D human models are reconstructed from range scan data and then animated, parameterized and processed further for various applications. We have also adapted the extended linear blending scheme of Mohr and Gleicher [57] to become a sweep-based shape blending scheme. In [2, 57], the deformation of the human models is based on example data acquired from many different poses; our approach requires only a single pose. Additional poses can then be generated automatically. The user can also modify the shape of the model in a particular pose or during motion by editing and interpolating the underlying sweep surfaces.

Our sweep-based approach is similar to the human limb modeling and deformation technique proposed by Hyun et al. [39] and Kalra et al. [44]. The main difference is that we represent a whole body using sweep surfaces with star-shaped cross-sections. The body shape is then precisely reconstructed from these sweep surfaces using displacement maps. One technical challenge is how to combine displacement surfaces around the shoulder and the hip, where different surfaces meet. We address this issue by applying a smooth shape blending scheme to the displacements from different surfaces. Figure 1.2 shows a deformation result of a female model using our technique.

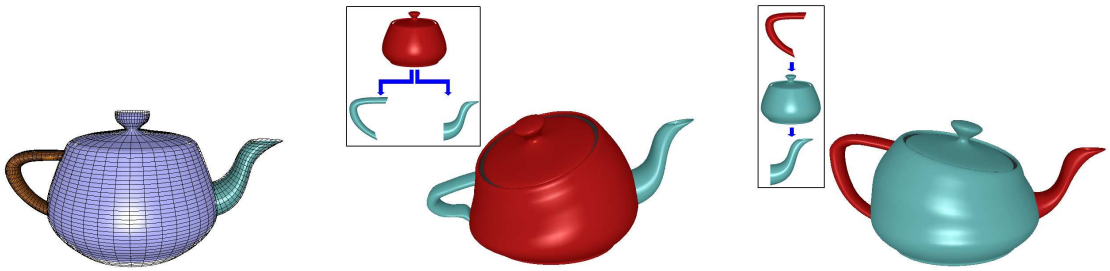
We also support certain features of the anatomy-based approach, such as elbow-

Figure 1.2: 'Victoria' in a ballet motion.

protrusion, muscle-bulge and skin-folding as an arm bends. These effects are realized using a GPU-based collision-detection procedure [28].

We then extend this technique to the freeform deformation of three-dimensional objects. Since its introduction by Sederberg and Parry [64], the freeform deformation (FFD) has established itself as one of the most powerful shape design methods for freeform objects. A user of FFD starts with an existing object and changes its shape. This contrasts with the use of sweeps, which allow a designer to create three-dimensional objects from scratch instead of modifying existing shapes. In this thesis, we combine these two well-known shape design tools, and propose a new technique for the sweep-based freeform deformation of existing three-dimensional objects.

In conventional FFD [18, 56, 64], the user deforms the shape of an object using control lattices which define trivariate volumes enclosing the parts of an object to be



(a) (b) (c)

Figure 1.3: Interactions among deformations: (a) teapot model and control sweep surfaces, (b) deformation of the body, and (c) deformation of the spout and handle.

deformed. These methods require a hierarchy of multiple lattices when applied to complex geometric objects with multiple control handles. The hierarchy of multiple control lattices is rather difficult to specify because of its three-dimensional structure. This difficulty motivates us to consider variants of FFD which are based on a one-parameter family of affine transformations [7, 12] or on a coordinate frame that moves along the axis of an object [51].

Figure 1.3 explains the basic idea of our approach. It shows the Utah teapot represented as a union of three deformable parts: the body, the spout and the handle. Each part is approximated by a sweep surface. The boundary vertices of each deformable part are then bound to appropriate cross-sections of the sweep surfaces. By deforming these surfaces, the corresponding parts of the teapot change their shapes. An interesting feature of the Utah teapot is its multiple control handles, and the deformation of one handle influences the others to maintain the consistency of the teapot's topology. There certainly need to be constraints on the allowable interactions between the three different parts of the Utah teapot. Figure 1.3(b) shows a deformation of the teapot body and the hierarchy of different sweep surfaces, which

automatically changes the shape of the handle and the spout so as to maintain the topology of the teapot model as the body changes its shape. Figure 1.3(c) shows the result of a deformation of the handle, and the hierarchy of sweep surfaces which makes the body and spout change their shapes as the user bends the handle. In this thesis, we will address these interaction problems and show how to set up the necessarily complicated interaction rules between different deformations.

Using existing FFD methods, it is difficult, or at best tedious, to support interactions among multiple deformations. Even commercial modeling packages may only support a few limited symmetric interactions. Some FFD methods [7, 51] provide insufficient degrees of freedom for specifing arbitrary interactions among multiple deformations. Our new FFD approach uses sweeps to support interactions among different deformable parts in a natural way.

Our technique works by approximating the deformable parts of an object using control sweep surfaces which are constructed by interpolating some key cross-sections. An object is represented using multiple deformable sweep surfaces which can interact because they are organized in a hierarchy. An effective way of binding a sweep surface to other sweep surfaces within such a hierarchy is to bind key cross-sections to cross-sections of the parent part. Using this simple binding mechanism, we can construct a hierarchy between a number of control sweep surfaces and then deform them simultaneously.

We further extend the sweep-based approach to the elastic deformation of three-dimensional objects. We enhance a sweep surface to an elastic sweep surface. After binding the vertices of deformable parts to the elastic sweep surfaces, the user changes

the shapes of the underlying control sweep surfaces or applies external forces. By physical simulations of the sweep surfaces, the deformable part changes their shapes dynamically and elastic deformations are obtained.

An elastic sweep surface is constructed by interpolating key cross-sections that change their positions, orientations and boundary shapes elastically. For this purpose, we employ three different types of mass-spring systems, each of which simulates the elastic changes of the position, orientation and radii of a key cross-section respectively. Note that each of these changes are limited to key cross-sections and we consider only one-dimensional spaces such as angles and distances. To generate elastic deformation effects over the whole object, we combine the changes at key cross-sections by interpolating them simultaneously, which produces a plausible deformations of the elastic sweep surface. In effect, we decompose the deformation in three-dimensional space into simple one-dimensional components and reconstruct the overall deformation from the component deformations, which is the basic approach taken in this thesis.

An elastic potential energy or external force initiates the elastic deformation of a sweep surface. The elastic potential energy is generated by editing key cross-sections or directly controlling a sweep surface. The external force is applied to a sweep surface by a user interaction or as a result of collision with other objects. It is decomposed into a rotational force and a radial one for a proper key cross-section and different deformation effects are achieved by applying each of them selectively. Figure 1.4 shows the results of sequential deformations of an elastic sweep surface in which an external force is generated by the collision of a sphere (in red) with the sweep surface.

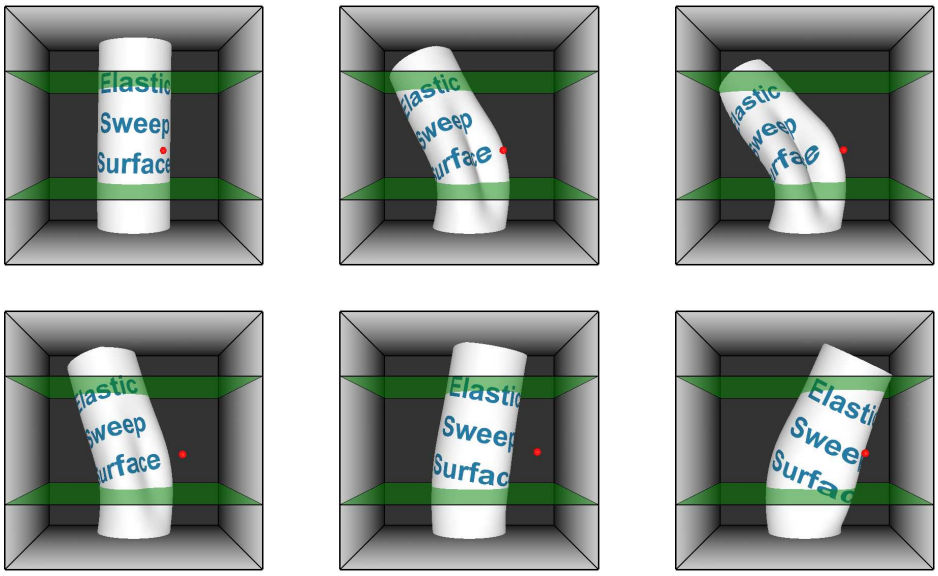


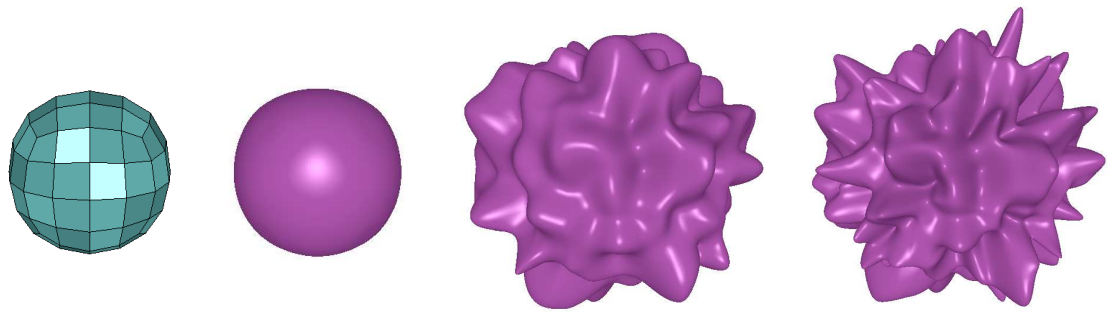
Figure 1.4: Deformation results of applying external forces.

Finally, we apply our sweep-based approach to a detailed geometric model represented as a point cloud. Thanks to recent advances in scanning technologies, it has become easy to obtain accurate and detailed geometric data from real-world objects. However, contemporary scanning systems produce a discrete representation, in the form of unorganized point clouds or polygonal meshes. Such models can have serious problems, including irregularities, discontinuities, huge size and missing areas, making it difficult to use this sort of data for practical applications.

Many mesh processing techniques have been developed to resolve these problems. Representative methods include simplification, smoothing, re-meshing and mesh optimization [22, 33, 34]. However, most of these techniques operate entirely in the domain of discrete geometry, and do not offer analytic representations which are useful in geometric modeling and processing.

Cook [17] introduced the displaced surface as an ingenious way of creating a

detailed geometric model by applying a displacement map to a smooth surface. Krishnamurthy and Levoy [49] used tensor product B-spline patches to represent the underlying domain surface and created fine details with displacement vectors; but this leads to a discontinuity problem across patch and displacement map boundaries. Recently, Lee et al. [52] have proposed a framework to unify subdivision surfaces and scalar displacements. Thanks to the properties of subdivision surfaces, there are no discontinuity problems in this approach. However, subdivision surfaces do not offer analytic representations in general.



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

Figure 1.5: Displaced surface: (a) control mesh, (b) C^2-continuous domain surface, (c) and (d) displaced surfaces at different resolutions.

In this thesis, we introduce a new representation scheme for displaced surfaces, based on the manifold technique proposed by Ying and Zorin [79]. We first construct a simple C^2-continuous smooth surface which interpolates the vertices of a given control mesh. Then, we define a scalar displacement function on each chart of an atlas of this domain. A detailed surface can now be constructed by applying scalar displacement functions to the local patches and blending them using the manifold structure of a control mesh. Figure 1.5 shows the displaced surfaces generated by our

technique.

In order to be able to address practical applications with our technique, we also propose an algorithm to approximate a detailed geometric model given as a point cloud. This approximation is based on the analytic properties of our representation scheme and the optimization of scalar displacement functions on the local charts of the domain atlas. Once the point cloud has been approximated using a displaced surface, sweep-based approach is then applied to a control mesh of the displaced surface. As the control mesh deforms, the displaced surface is reconstructed and smooth shape deformations are achieved.

The main contributions of this thesis are summarized as follows:

- Sweep-based human deformation technique provides highly effective control handles for animating and designing human motion. Our approach demonstrates real-time performance for deforming human models of reasonable complexity.

- Sweep-based human deformation technique also supports some anatomical features such as elbow and knee protrusions as well as non-penetrating skin folding using a GPU-based collision detection procedure.

- Sweep-based freeform deformation technique provides an intuitive and effective control mechanism for modifying and editing three-dimensional shapes and various direct control techniques are proposed to enhance the efficiency of our approach.

- Volume-preserving interactions among different deformable parts of an object

are fully supported in a hierarchical manner.

- Our elastic sweep surface is constructed by interpolating the elastic changes of the positions, orientations and radii of a few key cross-sections. This approach greatly simplifies the elastic deformation of the whole sweep surface and provides a real-time performance.

- Compound elastic deformation effects can easily be achieved by integrating simple component deformations from key cross-sections. Moreover, an external force can be decomposed into a rotational force and a radial one for a proper key cross-section and various different types of deformation effects can be achieved by applying a few of them selectively.

- Our sweep-based elastic deformation technique provides an effective control mechanism for the elastic deformation of three-dimensional objects by specifying constraints of the underlying control sweep surfaces. For example, the user can specify which parts an object would undergo elastic bending or twisting.

- We propose a new displaced surface representation using a manifold structure and displacement functions. Using this representation, a point cloud is approximated with high precision.

- We apply our sweep-based approach to the control mesh of our displaced surface. As the control mesh deforms, the displaced surface is reconstructed and smooth shape deformations are generated.

The rest of this thesis is organized as follows. In Chapter 2, we briefly review some mathematical preliminaries related to non-uniform rational B-spline curves/surfaces

and rational B-spline motions. A sweep-based approach to human body modeling and deformation is presented in Chapter 3. In Chapter 4, we extend this sweep-based approach to the freeform deformation of three-dimensional objects. In Chapter 5, we further extend the sweep-based approach to the elastic deformation of three-dimensional objects. In Chapter 6, we approximate detailed geometric models represented as point clouds using a displaced surface from a manifold, and apply our sweep-based approach to the control mesh of the displaced surface. Finally, we conclude this thesis and suggest future works in Chapter 7.

Chapter 2

Preliminaries

This chapter summarizes some fundamentals of rational curves and surfaces (see [23, 59]), rational motions (see [8, 43, 74]).

2.1 Spatial Kinematics

In order to study motions, we consider two copies of the Euclidean 3-space. The first one is called the fixed space E^3, with the points p. The second copy is called the moving space $\hat{E}^3$, with the points $\hat{p}$. Both spaces are associated with right-handed Cartesian coordinate frames. The moving space $\hat{E}^3$ can be identified with a moving Cartesian coordinate frame $\hat{F}$ which is moved along a certain curve of the fixed space E^3. The position $\mathbf{p} \in E^3$ of a point $\hat{\mathbf{p}}$ of the moving space $\hat{E}^3$ results from a linear transformation

$$\hat{\mathbf{p}} \mapsto \mathbf{p} = R\hat{\mathbf{p}} + \mathbf{v}, \tag{2.1}$$

where R is a 3×3 rotation matrix and $\mathbf{v} = (v_1, v_2, v_3)$ is the translation vector. This transformation can be represented using a 4×4 homogeneous coordinate transformation matrix M:

$$\begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = \underbrace{\left(\begin{array}{ccc|c} & R & & \mathbf{v} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)}_{M} \begin{pmatrix} \hat{\mathbf{p}} \\ 1 \end{pmatrix}.$$

This transformation preserves the distances measured between points in $\hat{F}$ and is therefore a rigid transformation which we term a *spatial displacement*.

2.1.1 Motion

The *motion* of a rigid body is represented as a continuous one parameter family of spatial displacements $M(t)$ where the parameter t is assumed to be the time. Given a point $\hat{\mathbf{p}}$, the matrix curve $M(t)$ generates a continuous set of points $\mathbf{p}(t) = M(t)\hat{\mathbf{p}}$ called the *trajectory* of the point $\hat{\mathbf{p}}$. The direction of the tangent to the trajectory $\mathbf{p}(t)$ at $t = t_0$ is the derivative

$$\dot{\mathbf{p}}(t_0) = \dot{M}(t_0)\hat{\mathbf{p}} = \dot{M}(t_0)M^{-1}(t_0)\mathbf{p}(t_0) \tag{2.2}$$

From this equation we observe that the matrix $\dot{M}M^{-1}$ computes the derivative $\dot{\mathbf{p}}$ by operating on the trajectory $\mathbf{p}(t)$. We call this matrix the *tangent operator*, because it computes the direction tangent to a motion.

Based on the spatial displacement described previously, the motion $M(t)$ is represented as:

$$M(t) = \left(\begin{array}{ccc|c} & R(t) & & \mathbf{v}(t) \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{2.3}$$

and its tangent operator is obtained by

$$\begin{aligned}\dot{M}M^{-1} &= \left(\begin{array}{ccc|c} & \dot{R} & & \dot{\mathbf{v}} \\ \hline 0 & 0 & 0 & 0 \end{array}\right)\left(\begin{array}{ccc|c} & R^{\mathrm{T}} & & -R^{\mathrm{T}}\mathbf{v} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\ &= \left(\begin{array}{ccc|c} & \dot{R}R^{\mathrm{T}} & & -\dot{R}R^{\mathrm{T}}\mathbf{v}+\dot{\mathbf{v}} \\ \hline 0 & 0 & 0 & 0 \end{array}\right),\end{aligned}$$

where $\dot{R}R^{\mathrm{T}}$ is the 3×3 skew-symmetric matrix.

2.2 Rational Curves and Surfaces

We briefly review some basic definitions and properties for polynomial and rational curves and surfaces.

2.2.1 Bézier curves and surfaces

The nth-degree *Bézier curve* is defined by

$$\mathbf{c}(u) = \sum_{i=0}^{n} B_i^n(u)\mathbf{p}_i, \qquad 0 \le u \le 1 \tag{2.4}$$

The basis(blending) function, $\{B_i^n(u)\}$, is the n-th-degree *Bernstein polynomial* given by

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}. \tag{2.5}$$

The geometric coefficients of this form, $\{\mathbf{p}_i\}$, are called *control points* in Euclidean 3-space E^3.

It is easy to derive the following properties of Bernstein polynomials $\{B_i^n(u)\}$.

- Subdivision

$$B_i^n(cu) = \sum_{j=0}^{n} B_i^j(c) B_j^n(u).$$

- Derivative:

$$\frac{d}{dt} B_i^n(u) = n[B_{i-1}^{n-1}(u) - B_i^{n-1}(u)].$$

- Product:

$$B_i^m(u) B_j^n(u) = \frac{\binom{m+n}{i+j}}{\binom{m}{i}\binom{n}{j}} B_{i+j}^{m+n}(u).$$

Tensor product polynomial Bézier surfaces are obtained by taking a bidirectional net of control points and products of the univariate Bernstein polynomials:

$$\mathbf{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) \mathbf{p}_{i,j}, \qquad 0 \le u, v \le 1. \tag{2.6}$$

Rational version of Bézier curves and surfaces are obtained with weight factor $\{w_i\}$ and $\{w_{i,j}\}$ and control points $\{\overline{\mathbf{p}}_i\}$ and $\{\overline{\mathbf{p}}_{i,j}\}$ in E^3, respectively, by

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} B_i^n(u) w_i \overline{\mathbf{p}}_i}{\sum_{i=0}^{n} B_i^n(u) w_i}, \qquad 0 \le u \le 1 \tag{2.7}$$

and

$$\mathbf{s}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) w_{i,j} \overline{\mathbf{p}}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) w_{i,j}}, \qquad 0 \le u, v \le 1. \tag{2.8}$$

The rational Bézier curves/surfaces are frequently represented in homogeneous coordinates with control points of projective 3 space (P^3) as follows:

$$\mathbf{c}(u) = \sum_{i=0}^{n} B_i^n(u) \mathbf{p}_i \quad \text{where} \quad \mathbf{p}_i = (w_i \overline{\mathbf{p}}_i, w_i) \in P^3, \quad 0 \le u \le 1$$

and

$$\mathbf{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) \mathbf{p}_{i,j} \quad \text{where} \quad \mathbf{p}_{i,j} = (w_{ij} \overline{\mathbf{p}}_{i,j}, w_{ij}) \in P^3, \quad 0 \le u, v \le 1$$

2.2.2 B-Spline curves and surfaces

Let $U = \{u_0, \ldots, u_m\}$ be a nondecreasing sequence of real numbers, i.e., $u_i \leq u_{i+1}$, $i = 0, \ldots, m-1$. The u_i are called *knots*, and U is the *knot vector*. The i-th *B-spline basis function* of degree p(order p+1), denoted by $N_i^p(u)$, is defined as

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} N_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_i + 1} N_{i+1}^{p-1}(u) \tag{2.9}$$

A pth-degree *B-spline curve* is defined by

$$\mathbf{c}(u) = \sum_{i=0}^{n} N_i^p(u)\mathbf{p}_i, \qquad a \leq u \leq b \tag{2.10}$$

where the $\{\mathbf{p}_i\}$ are the control points, and the $\{N_i^p(u)\}$ are the pth-degree B-spline basis functions defined on the knot vector with (m + 1) knots.

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}, \underbrace{b, \ldots, b}_{p+1}\}$$

In this thesis, we assume that $a = 0$ and $b = 1$.

A B-spline curve can be changed into a Bézier one by the well known knot insertion algorithm [23, 59]. The product operation between two B-spline curves can be computed based on the Bézier curves after we change them into Bézier ones. For more details, see [43].

A *B-spline surface* is obtained by taking a bidirectional net of control points, two knot vectors, and the product of the univariate B-spline functions

$$\mathbf{s}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_i^p(u) N_j^q(v) \mathbf{p}_{i,j} \tag{2.11}$$

with

$$U = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_{r-p-1},\underbrace{1,\ldots,1}_{p+1}\}$$
$$V = \{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,u_{s-1-1},\underbrace{1,\ldots,1}_{q+1}\}$$

U has r+1 knots, and V has s+1.

A pth-degree *non uniform rational B-spline(NURBS) curve* is defined by

$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} N_i^p(u) w_i \mathbf{p}_i}{\sum_{i=0}^{n} N_i^p(u) w_i} \qquad 0 \le u \le 1 \tag{2.12}$$

with the knot vector U. The homogeneous form of the curve is represented as:

$$\mathbf{c}(u) = \sum_{i=0}^{n} N_i^p(u)\mathbf{p}_i,$$

where

$$\mathbf{p}_i = (w_i\overline{\mathbf{p}}_i, w_i)$$

is the homogenous coordinate of its control point.

A *NURBS surface* of degree p in the u direction and degree q in the v direction is defined by

$$\mathbf{s}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_i^p(u)N_j^q(v)w_{i,j}\overline{\mathbf{p}}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_i^p(u)N_j^q(v)w_{i,j}} \qquad 0 \le u,v \le 1 \tag{2.13}$$

with two knot vectors U and V and its homogeneous representation is represented as:

$$\mathbf{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_i^p(u)N_j^q(v)\mathbf{p}_{i,j}, \quad \text{where} \quad \mathbf{p}_{i,j} = (w_{i,j}\overline{\mathbf{p}}_{i,j}, w_{i,j}).$$

2.3 Rational Motions

If all elements of a motion $M(t)$ are polynomials of maximal degree k, $M(t)$ is called a *rational motion* of degree k. The following Lemma has been derived in [43].

Lemma 2.1 (Rational Motion)

A motion M(t) is piecewise rational motion of degree k if and only if there exists a representation

$$M(t) = \left(\begin{array}{ccc|c} & & & v_1 \\ & v_0^* D(t) & & v_2 \\ & & & v_3 \\ \hline 0 & 0 & 0 & v_0^*(d_0^2+d_1^2+d_2^2+d_3^2) \end{array}\right) \tag{2.14}$$

with

$$D(t) = \begin{pmatrix} d_0^2+d_1^2-d_2^2-d_3^2 & 2(d_1d_2-d_0d_3) & 2(d_1d_3+d_0d_2) \\ 2(d_1d_2+d_0d_3) & d_0^2-d_1^2+d_2^2-d_3^2 & 2(d_2d_3-d_0d_1) \\ 2(d_1d_3-d_0d_2) & 2(d_2d_3+d_0d_1) & d_0^2-d_1^2-d_2^2+d_3^2 \end{pmatrix} \tag{2.15}$$

where $v_0^*(t)$, $v_i(t)$, and $d_i(t)$ are piecewise polynomials of maximal degree $k-2l$, k and l, respectively, where the number l satisfies $0 \leq 2l \leq k$.

The four parameters $\tilde{d} = (d_0, d_1, d_2, d_3)$ are called *Euler's parameters* of the rotational part $D(t)$. The trajectory of the origin of the moving frame is

$$\mathbf{u} = (v_1, v_2, v_3, v_0^* \|\tilde{\mathbf{d}}\|^2)$$

in homogeneous coordinates. The Euler parameters have the same role in 3D Rotation as the unit quaternion $\mathbf{q}$ except that they do not have to be a unit 4D vector. The homogeneous factor of matrix $M(t)$ does normalize $D(t)$ to be $D(t)/\|\tilde{d}\|^2 \in SO(3)$.

2.3.1 Rational B-Spline motion

Consider a piecewise rational motion $M(t)$ of degree k. Since all elements of $M(t)$ have to be piecewise polynomials, there exists a B-spline representation

$$M(t) = \sum_{i=0}^{n} N_i^k(t) A_i \tag{2.16}$$

of degree k, where $N_i^k(t)$ is B-spline basis function defined on a knot vector $U = \{u_0, ..., u_m\}$. The motion of Equation (2.16) is called a *rational B-spline motion* of degree k. For every parameter t, the matrix $M(t)$ describes a Euclidean spatial displacement. Therefore we have the constant coefficient matrices

$$A_i = \left(\begin{array}{ccc|c} & & & w_{i,1} \\ & S_i & & w_{i,2} \\ & & & w_{i,3} \\ \hline 0 & 0 & 0 & w_{i,0} \end{array}\right).$$

In general, the submatrices S_i do not fulfill the orthogonality condition $S_i S_i^{\mathrm{T}} = I$. Thus the matrices A_i describe some affine transformations. The set of all control positions $\{A_i\}$ is called the *control structure* of rational B-spline motion (Equation (2.16)).

For arbitrary affine transformations A_i, the motion of Equation (2.16) will be an affine motion. A Euclidean rational B-spline motion $M(t)$ of degree k can be obtained from $v_0(t)^*$, $\mathbf{v(t)} = (v_1(t), v_2(t), v_3(t))$ and $\tilde{\mathbf{d}}(\mathbf{t})$ by **Lemma** 2.1. For more details, see [74].

B-spline motion of Equation (2.16) has the same form as conventional B-spline curves. Various algorithms and properties of B-spline curves, such as subdivision,

knot insertion/removal, and degree elevation/reduction, can also be applied to B-spline motion [43, 59, 74].

Note that the trace of an arbitrary point $\mathbf{p}$ under B-spline motion $M(t)$ generates a rational B-spline curve

$$\mathbf{p}(t) = M(t)\mathbf{p}$$

The sweep of a rational B-spline curve $C(u)$ under the B-spline motion $M(t)$ generates a tensor product rational B-spline surface:

$$\mathbf{S}(u,t) = M(t)\mathbf{c}(u) \tag{2.17}$$

This kind of matrix - vector product includes the same procedure with the B-spline multiplication algorithm applied to the generation of B-spline motion.

Chapter 3

Sweep-based Human Deformation

In this chapter, We present a sweep-based approach to human body modeling and deformation. In Section 3.1, we briefly review some representative human body modeling and deformation techniques. Section 3.2 presents our sweep-based human modeling technique. In Section 3.3, sweep-based human deformation technique is presented and some special features such as muscle-bulge and skin folding are supported. Finally, experimental results are presented in Section 3.4.

3.1 Related Work

Hyun et al. [39] and Kalra et al. [44] represent deformable arms and legs using sweep surfaces. However, they describe no specific method of connecting the limbs to the main body. In the current paper, we attack this more challenging problem by smoothly connecting arms and legs to shoulders and hip using a shape blending technique.

Mohr and Gleicher [57] proposed an extended linear blending scheme which adds joints to the interior of a link so as to make the link flexible. If we keep on adding such joints, we end up with infinitely flexible links, which are similar in effect to our sweep surfaces. There remains the issue of controlling infinitely many degrees of freedom. Our approach, based on B-spline interpolation, makes it considerably easier to modify the shape of the controlling sweep surfaces using a few 'key' cross-sections.

Allen et al. [2, 3] present algorithms that reconstruct, deform and parametrize human body shapes from range scan data. In [2], subdivision surfaces were used for the representation of skeleton-based global shape changes. Starck et al. [32, 70] control the deformation of a polygonal human model using a simplified mesh. In bending and twisting motions, sweep surfaces are easier to control than subdivision surfaces or polygon meshes. To demonstrate this, we present animation test clips in which virtual 3D human models walk and dance.

Singh and Kokkevis [68] propose a surface-oriented FFD for controlling the skin deformation of characters. They use a polygon mesh as a driver for the skin deformation. Our approach is closely related to theirs. Our improvement is in using control sweep surfaces which fit tightly to the surface of the model and thus control skin deformation more faithfully.

Many previous authors [5, 10, 62, 78] have presented anatomy-based physical simulation techniques that produce very realistic modeling and deformation of humans and animals. We do not propose full-scale anatomical simulation with our simplified representation. Nevertheless, we can support in real-time some important features such as muscle-bulge, elbow-protrusion, skin-folding, and volume preservation. The

real-time performance is the main advantage of our method over previous methods that rely on physical simulation.

Our algorithm works for arbitrary human models based on boundary representations such as polygonal meshes, scan data, point clouds, subdivision surfaces, B-spline surfaces, etc. Thus it is relatively easy to adapt our shape control scheme to conventional techniques such as those of Allen et al. [3] and Seo and Magnenat-Thalmann [65] for manipulating human models.

3.2 Sweep-based Human Modeling

We approximate human arms, legs, torso and neck using sweep surfaces, which are then used as control structures for human body deformation. The vertices of a human model are bound to the sweep surfaces and then follow the transformations of the sweep surfaces. We apply a shape blending scheme to the areas where different sweep surfaces meet.

3.2.1 Sweep surface

Sweeps are a powerful paradigm for representing 3D freeform objects based on simple procedural rules [25, 59]. Examples include translational sweeps and rotational sweeps. General sweeps of 2D cross-sections are known as generalized cylinders [6, 46]. In this thesis, we consider star-shaped 2D cross-sections that approximate the cross-sectional shapes of various human body parts.

When a star-shaped cross-sectional closed curve $O_t(\theta) = (r(\theta, t)\cos\theta, r(\theta, t)\sin\theta, 0)^T$ is moving under rotation $R(t)$ and translation $C(t)$, it generates a sweep surface

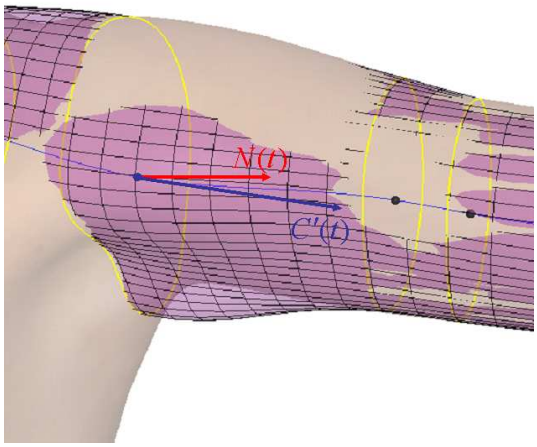


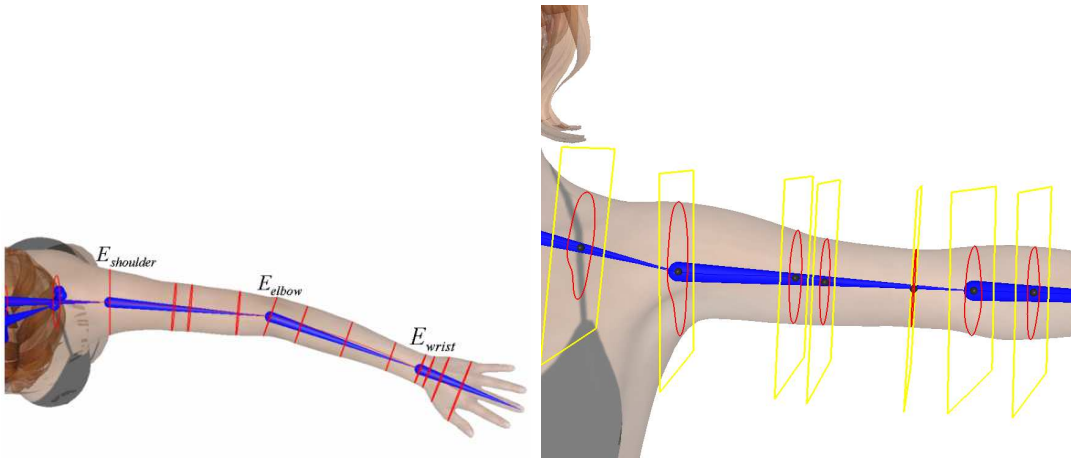
Figure 3.1: Sweep surface generated by a moving cross-section.

$S(\theta, t) = C(t) + R(t)O_t(\theta)$ which is precisely described as follows:

$$\begin{aligned} S(\theta,t) &= C(t) + R(t) \cdot O_t(\theta) \\ &= \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} r_{11}(t) & r_{12}(t) & r_{13}(t) \\ r_{21}(t) & r_{22}(t) & r_{23}(t) \\ r_{31}(t) & r_{32}(t) & r_{33}(t) \end{bmatrix} \cdot \begin{bmatrix} r(\theta,t)\cos\theta \\ r(\theta,t)\sin\theta \\ 0 \end{bmatrix}. \end{aligned} \tag{3.1}$$

The star-shaped cross-section $\hat{O}_t(\theta)$ lies on a moving plane $P(t)$ which is determined by a point $C(t)$ and a unit normal vector $N(t) = (r_{13}(t), r_{23}(t), r_{33}(t))^T$, which is the third column of $R(t)$.

Figure 3.1 shows how a moving cross-section generates a sweep surface. Note that the first two columns of $R(t)$ form the major and minor axis directions of the moving cross-section, and that the normal vector $N(t)$ is not exactly in the same direction as the tangent vector $C'(t)$ of the trajectory curve $C(t)$.



(a) (b)

Figure 3.2: Key cross-sections for the left arm.

3.2.2 Control sweep surface generation

Control sweep surfaces are generated by interpolating a set of 'key' cross-sections that tightly fit the cross-sections of human arms, legs, torso and neck. One key cross-section is assigned to each joint, and its center is fixed to that joint. The boundary shape and orientation of each key cross-section is determined by cutting the polygonal model through a plane orthogonal to the lower link of the joint. Additional key cross-sections are assigned to intermediate locations on a link (see Figure 3.2(a)). Each boundary shape of key cross-section is determined by cutting the model through a plane orthogonal to the link (see Figure 3.2(b)). Intermediate key cross-sections have centers at the centroids of their cross-sections, which may not lie on the link.

The center positions of key cross-sections are interpolated by a cubic B-spline curve $C(t)$ using a chord length parametrization for knot spacing. Their orientations are represented as unit quaternions, q_i, $i = 0, \ldots, n$, and linearly interpolated as

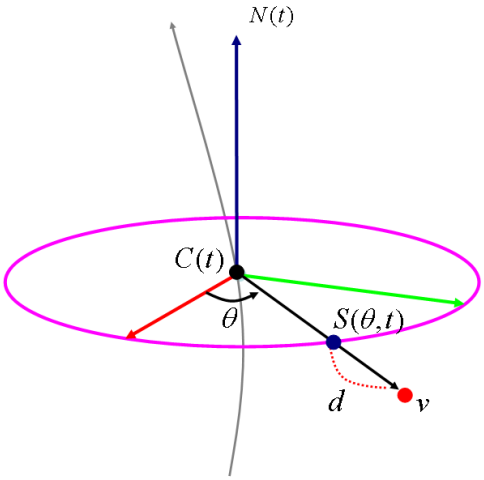


Figure 3.3: Binding a vertex to a control sweep surface.

follows:

$$Q_i(t) = (1-t)q_i + tq_{i+1}, \quad \text{for } t_i \leq t \leq t_{i+1}.$$

Note that the quaternion curve $Q_i(t)$ is unnormalized. We use a normalized curve $\frac{Q_i(t)}{\|Q_i(t)\|}$ to form a 3D rotation matrix $R(t)$, each term of which is a rational quadratic function of t. The sampled radii on each key cross-section are interpolated by the quadratic B-spline function $r(\theta, t)$. The sweep surface $S(\theta, t)$ is constructed from $C(t), R(t), r(\theta, t)$ using Equation (3.1).

3.2.3 Vertex binding and reconstruction

A sweep surface is a one-parameter family of cross-sections. Each vertex V on a human polygonal model is bound to an instance of a moving cross-section, and follows its transformation. A vertex V is bound to a cross-section $\hat{O}_t(\theta)$ by solving

$$< V - C(t), N(t) >= 0,$$

where $< \cdot, \cdot >$ signifies an inner product, and $N(t)$ represents a unit normal vector of the moving cross-sectional plane. The local coordinate of the vertex V in that plane is then determined from $\hat{V} = R(t)^T(V - C(t))$. The circular binding angle θ is $arctan(\hat{V}_y/\hat{V}_x)$. Finally, the signed distance d from the control sweep surface is computed as $d = \|\hat{V}\| - \|S(\theta, t) - C(t)\| = \|V - C(t)\| - \|S(\theta, t) - C(t)\|$. Figure 4.3 shows an example of this binding procedure in which a vertex V is bound to a cross-section of a control sweep surface using the binding parameters (θ, t, d). When a sweep surface deforms, the mesh vertex V is reconstructed from the sweep surface using the following equation:

$$V = C(t) + R(t) \begin{bmatrix} (r(\theta, t) + d) \cdot cos\theta \\ (r(\theta, t) + d) \cdot sin\theta \\ 0 \end{bmatrix},$$

where (θ, t, d) are the binding parameters of the vertex V.

Vertex binding is carried out at the 'dress' pose, the initial position of the human model, at which the sweep surfaces have no self-intersections. Thus each vertex around a sweep surface will be bound to a unique moving cross-section $\hat{O}_t$. This binding is fixed for the rest of the deformation. As the arms and legs deform, the sweep surfaces and the polygonal model may start developing self-intersections. The interpenetration of deformed vertices with other body parts is prevented by a GPU-based collision detection procedure, and the skin starts developing creases and then folds. The details of this will be discussed in Sections 3.3.3.

3.2.4 Blending sweep surfaces

Figure 3.4 shows a shoulder area, where vertices are bound to two different sweeps (one for the left arm and the other for the torso). When a vertex V is bound to the arm sweep, we have

$$V^a = S^a(\theta_V^a, t_V^a) + R^a(t_V^a) \begin{bmatrix} d_V^a cos\theta_V^a & d_V^a sin\theta_V^a & 0 \end{bmatrix}^T$$

The same vertex V is similarly bound to the torso sweep:

$$V^t = S^t(\theta_V^t, t_V^t) + R^t(t_V^t) \begin{bmatrix} d_V^t cos\theta_V^t & d_V^t sin\theta_V^t & 0 \end{bmatrix}^T$$

Initially, the two vertices V^a and V^t are located in the same position. As the shoulder angle changes, the two sweep surfaces deform quite differently and the two vertices V^a and V^t follow different paths. We compute the final transformation of V as a convex combination of V^a and V^t:

$$V = (1 - w_V)V^a + w_V V^t,$$

where w_V is a weighting factor assigned to V. This weighting factor changes smoothly in the area where the different sweep surfaces meet.

Figure 3.5 shows how the body is segmented into separate regions (torso, arms, legs). The progression from one gray level to another represents the smooth transition of weight assignments. In this example, the weights in the transition area are determined by the relative distances from two bounding planes. Figure 3.6 shows the deformation of a human model in the region of the shoulder and armpit as the model lifts its left arm. The displacement is relatively large in the armpit. Nevertheless, the result of blending two sweep-based representations V^a and V^t of the polygonal model V produces a natural body deformation.

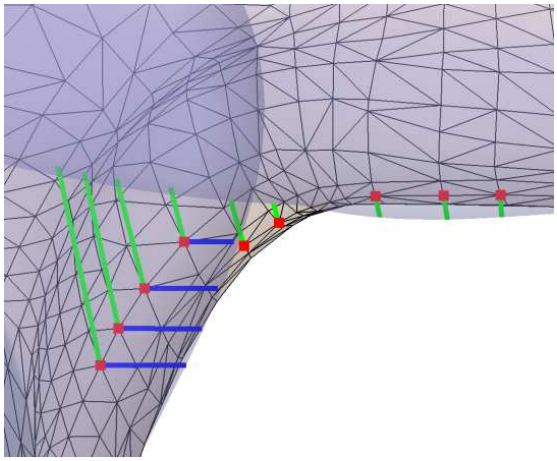


Figure 3.4: Vertices bound to two different sweep surfaces.

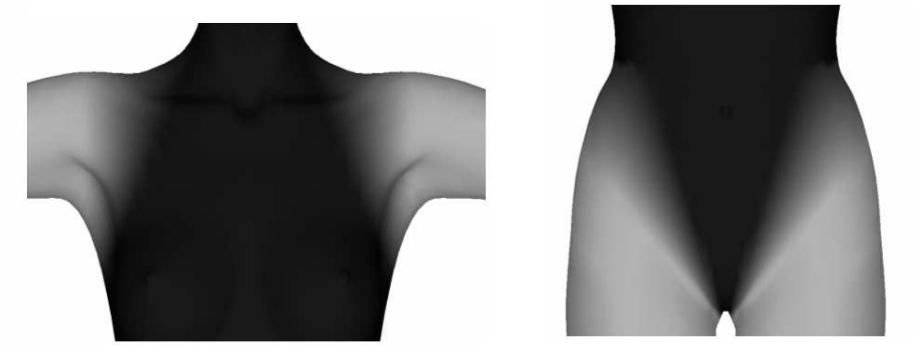


Figure 3.5: Smooth transition of weight assignments.

Figure 3.6: Deformation of shoulder.

3.3 Sweep-based Human Deformation

Human figures in different poses are automatically generated at an interactive speed as the user controls the skeleton of a human model using inverse kinematics or motion capture data. The sweep surfaces deform as the joint angles change. The user can further edit the sweep surfaces to improve the visual realism of the shape deformation. Anatomical features are also supported by a GPU-based collision detection procedure.

3.3.1 Sweep surface deformation

Sweep surfaces deform to follow changes to their key cross-sections. Each key cross-section assigned to a joint has its center fixed to that joint, and its orientation

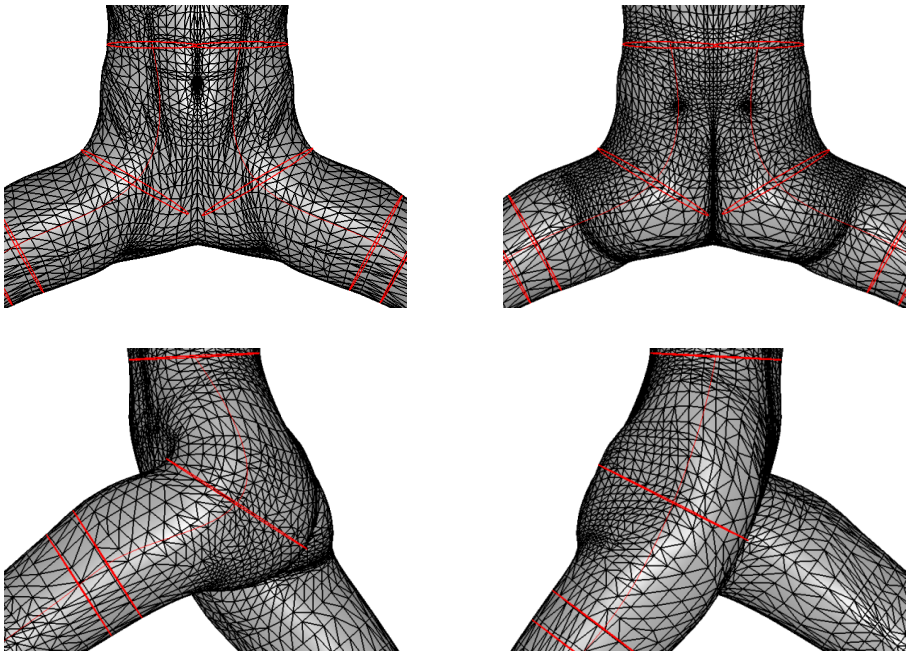


Figure 3.7: Deformation as the legs bend and spread out.

changes with the joint angle as a function of the relative orientation of the two links connected to the joint. Other intermediate key cross-sections follow the rigid motion of the link on which they are located.

The trajectory curve $C(t)$ and the orientation curve $Q(t)$ are recomputed in each frame to reflect the changing positions and orientations of key cross-sections. The scalar function $r(\theta, t)$ is not updated at this stage.

Figure 3.7 shows the deformation of a human model as it bends and spreads out its legs. This type of deformation is extremely difficult to carry out using conventional mesh processing techniques, because there will be many vertices clustered together in the crotch area of a human model.

3.3.2 Editing a sweep-based deformation

At extreme joint angles, a sweep-based deformation may not appear quite as natural as a real human body in the same pose because the sweep-based deformation does not consider the physical properties of skin and incorporates no anatomical knowledge. We allow the realism of such deformations to be enhanced by enabling the user to edit the control sweep surfaces.

The user can edit deformed human shapes by changing shape parameters for the key cross-sections that generate the control sweep surfaces. The position, orientation and radii of key cross-sections can all be changed and the human model will deform to follow these changes. Figure 3.8(b) shows the result of changing the orientation of only one key cross-section from Figure 3.8(a) to improve the deformation. The orientation of the key cross-section is a function of the shoulder joint angle. Thus the change of orientation at an extreme angle modifies this function smoothly using spline interpolation.

3.3.3 Anatomical features

Some anatomical features are very significant in the visual realism of body deformations: for example, elbow-protrusion, muscle-bulge, and skin-folding as an arm bends. We emulate these effects using a GPU-based collision-detection procedure.

Figure 3.9(a) shows the result of collision detection between the elbow-bone and the skin surface (shown in red), and also the result of detecting a self-intersection of the skin surface (shown in light green). Figure 3.9(b) shows similar results for knee bending. The polygon mesh around the collision region must deform according

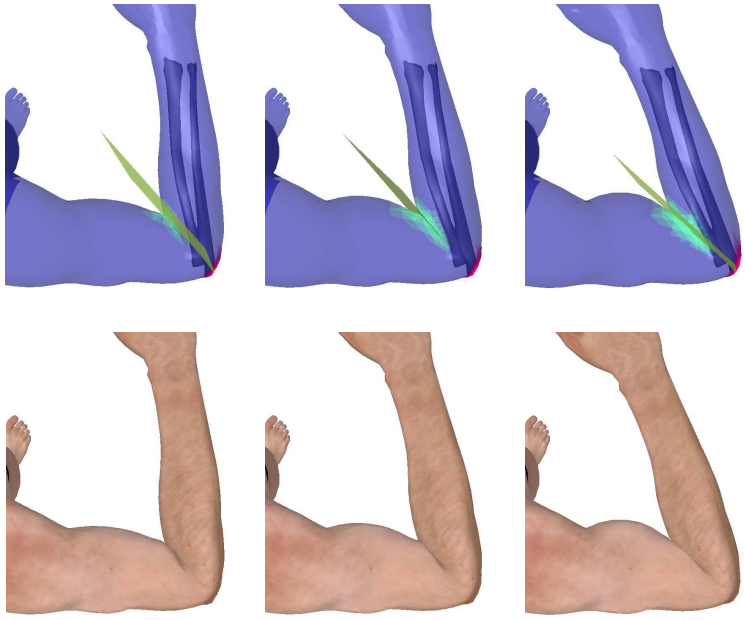
(a) (b)

Figure 3.8: Editing sweep deformations: (a) an automatically generated shoulder and (b) the result of editing one of its key cross-sections.

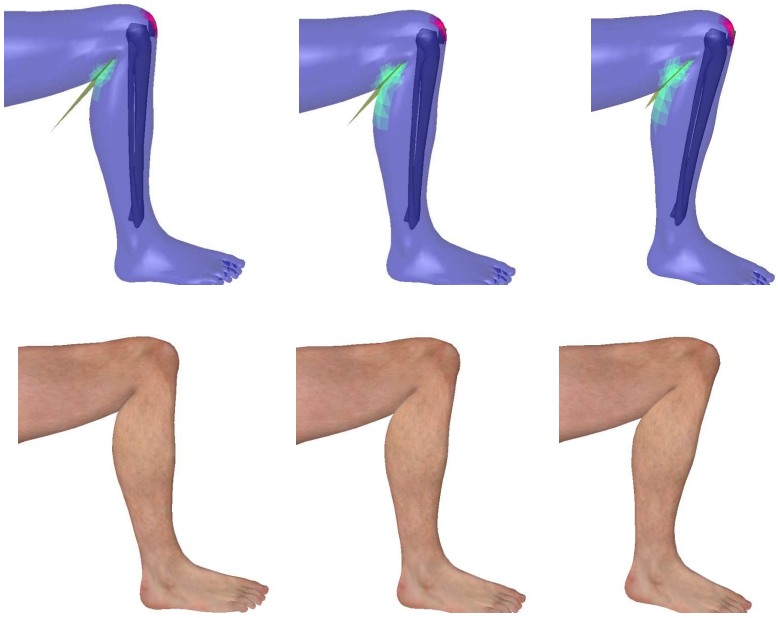
to the type of collision and its penetration depth. Our implementation of collision detection is based on the work of Govindaraju et al. [28], where GPU-based hardware programming was used to improve performance. For the computation of penetration depth, we use a software implementation.

A different approach is used to emulate muscle-bulge, an effect which is demonstrated in Figure 3.10. As the arm bends, the binding of skin vertices to underlying muscles is activated by increasing the weights influencing the muscles. The user can further edit the extent of muscle-bulge by changing the sweep parameters. Other anatomical features can also be emulated by adding sweep surfaces and binding skin vertices to these sweeps.

The sweep surfaces are generated so that they have no self-intersections when the body is in the initial 'dress' pose. Self-intersections start to develop in the region near a joint as the joint angle changes. A real human arm develops creases as it



(a)



(b)

Figure 3.9: Elbow and knee protrusions and skin-folding.

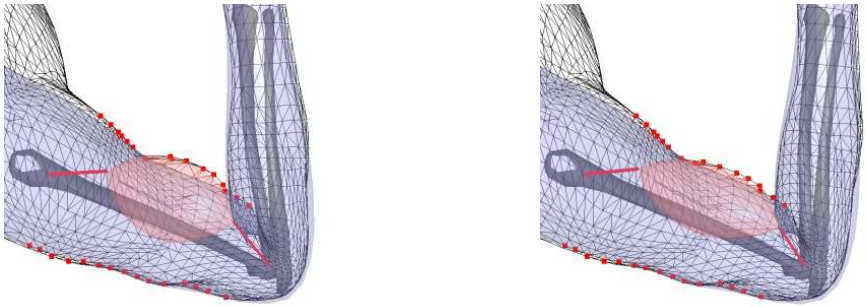


Figure 3.10: Muscle-bulge.

bends; and the skin folds as the arm or the leg bends further. We emulate these effects using a GPU-based collision detection procedure. Once all triangles in the region of self-intersection have been detected, we construct a bisecting plane for the self-intersection by fitting a plane (using least squares) to their vertices. The distance from this plane to each vertex gives the penetration depth of the vertex. By pulling the vertices back to the bisecting plane, we can generate skin-folding effects as an arm or leg bends. Figure 3.9 shows examples of skin-folding.

Elbow and knee protrusion effects are also realized using GPU-based collision detection. The penetration depth of each skin vertex in the collision region is computed by shooting a ray to the triangles of the bone in the collision region. The vertex is then pulled back to the exterior of the protruding bone.

3.4 Experimental Results

We have implemented our sweep-based human modeling and deformation algorithm in C++ on a P4-2GHz computer with a 1GB main memory and an NVIDIA

GeForce FX5700 GPU. Both the female model, 'Victoria', and the male model, 'Michael', were purchased from a commercial provider (http://www.daz3d.com). Each model has 72,712 vertices and 143,444 triangles. Motion capture data were re-targeted to these models using FilmBox [45].

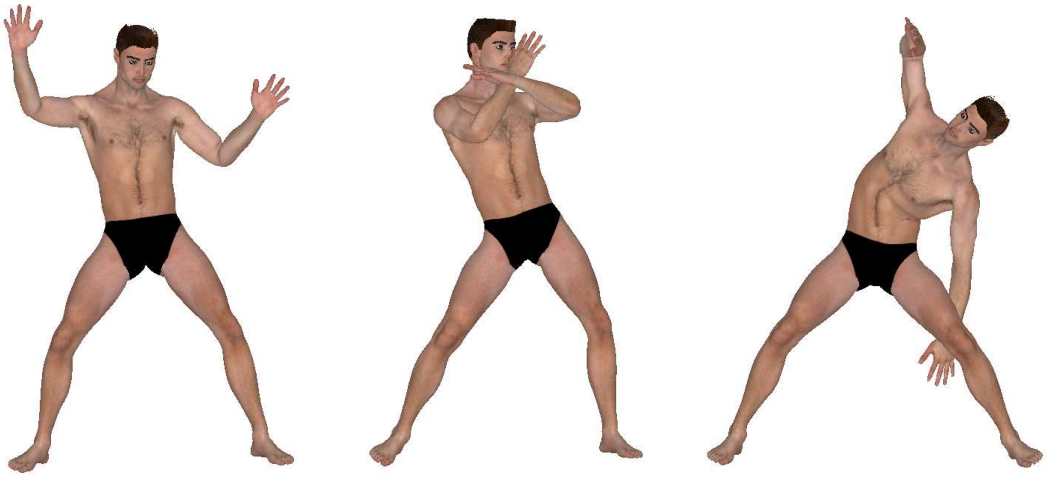
Figure 3.11 shows snapshots from animation clips we have generated to demonstrate the effectiveness of our approach. Each animation was generated by our system at about 7 frames per second, including all rendering processes. However, this frame rate does not include the time spent on supporting special features such as anatomy or volume preservation.

In emulating the special features of an arm, our system generates 11–12 frames per second, not including the rendering time. A sweep-based arm deformation, including muscle-bulge, is applied to approximately 20,000 triangles in the arm. This takes about 30–35% of the total processing time for a bending arm. After that, GPU-based collision detection is applied to a smaller set of 4400 triangles, which takes about 60–65% of the processing time. Elbow-protrusion and skin-folding effects are computed using these triangles in a software implementation. This process also includes the construction of the bisecting plane and computing the penetration depth of skin vertices. The performance for a leg model is slightly better since the muscle-bulge effect is not considered in this case.

3.5 Summary

We have shown that sweep surfaces provide an excellent control mechanism for deforming polygonal meshes which represent human models. Once a model has been

(a)



(b)

Figure 3.11: Snapshots from animation clips: (a) 'Victoria' in a ballet motion and (b) 'Michael' doing a 'techno' dance motion.

approximated with simple sweep surfaces, it is relatively easy to control the body shape using a small number of sweep parameters. Anatomical features are supported at an interactive speed using a GPU-based collision detection procedure. It is still difficult to apply all these features to a whole human model at an interactive speed.

Chapter 4

Sweep-based Freeform Deformation

In this chapter, we extend the sweep-based human deformation technique to the freeform deformation (FFD) of three-dimensional objects. In Section 4.1, we briefly review some representative FFD techniques. Section 4.2 introduces our sweep-based approach to freeform deformation. In Section 4.3, we propose three control techniques for sweep surfaces and in Section 4.4 we discuss interactions among deformations of different parts of an object. Finally, experimental results are presented in Section 4.6.

4.1 Related Work

FFD techniques employ different types of control lattices to construct three-dimensional volumes that surround the objects to be deformed. Sederberg and Parry [64] used parallelepiped control lattices. Coquillart [18] extended the types of control lattice to include cylindrical lattices and lattices located on surfaces. Using the Catmull-Clark subdivision scheme, MacCracken and Joy [56] further extended

the capability of FFD by introducing lattices of arbitrary topology. Recently, Song and Yang [69] extended the T-spline which was introduced by Sederberg et al. [63] and proposed a freeform deformation with weighted T-spline.

Barr [7] proposed a simple deformation technique for stretching, twisting, bending and tapering solid primitives. This method is essentially the application of a one-parameter family of affine transformations to the cross-sections of an object along its axis. Chang and Rockwood [12] clarified the underlying affine structure of this approach in a systematic way. Lazarus et al. [51] presented a general deformation scheme that is based on a coordinate frame that moves along the axis of an object. Singh et al. [67] proposed a deformation technique that uses a parametric curve and influence function. Hyun et al. [38] considered the deformation of a human body model using sweep surfaces.

These methods all provide some sort of intuitive handle on the deformation of an object. In this thesis, we further extend these results so as fully to utilize the underlying affine structures of the moving frames. Our approach offers effective ways of controlling the deformation of an object while preserving a visual resemblance to its original shape. An important advantage of this approach is that we can support various types of interaction among different parts of an object.

Recently, some new deformation techniques have been proposed. Hua and Qin [37] presented a modified FFD in which they employ a scalar field as the embedding space instead of a volume. The vertices of an object are parameterized by the level-set of a scalar field and follow its deformation. Angelidis et al. [4] introduced a volume-preserving operator as a shape deformation tool. Simplified polygonal meshes can also

be used as control structures [68, 16], as can continuous parametric surfaces [24, 13]. Yoshizawa et al. [80] presented a skeleton-driven mesh deformation technique in which they used a Voronoi-based skeletal mesh as a control structure for freeform models. Ju et al. [42] generalized mean-value coordinates from closed 2D polygons to closed triangular meshes and applied them to mesh deformation. Using a rigid motion-invariant mesh representation, Lipman et al. [55] proposed an interactive mesh editing and shape interpolation technique.

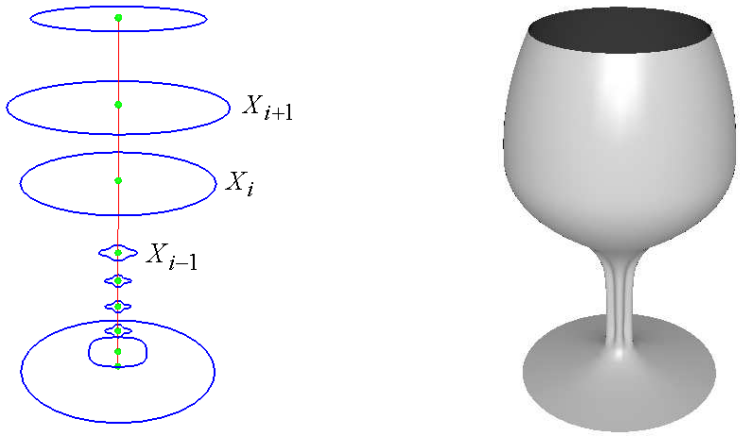
To improve the controllability of FFDs, a number of direct manipulation techniques have been proposed. Hsu and Kaufman [35] introduced the direct manipulation of FFDs, and later Hu et al. [36] presented an explicit solution to this problem using constrained optimization. Chang et al. [11] also proposed the direct manipulation of a generalized cylinder, which is somewhat similar to our sweep surface.

4.2 Sweep-based Freeform Deformation

Sweeps are a procedural modeling technique for representing three-dimensional freeform objects [25, 59]. In this section, we describe our sweep-based FFD.

4.2.1 Sweep surface from a continuous motion

A sweep surface generated by a continuous motion provides a nice control structure for FFDs. Let $\{X_i\}$ be a set of key cross-sections. Each key cross-section X_i is associated with a local transformation $T_{i-1,i}$ (represented by a 4×4 matrix) from the previous key cross-section X_{i-1}. Thus, when a key cross-section X_j changes its position and orientation, all the following key cross-sections $X_k (k > j)$ are automatically



(a) (b)

Figure 4.1: Sweep surface:(a) key cross-sections, (b) resulting sweep surface.

updated by a series of relative transformations. This arrangement facilitates intuitive deformations such as bending and twisting.

Our sweep surface is generated by interpolating these key cross-sections $\{X_i\}$:

$$\begin{aligned} S(\theta,t) &= C(t) + R(t)\cdot O_t(\theta) \\ &= \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} r_{11}(t) & r_{12}(t) & r_{13}(t) \\ r_{21}(t) & r_{22}(t) & r_{23}(t) \\ r_{31}(t) & r_{32}(t) & r_{33}(t) \end{bmatrix} \cdot \begin{bmatrix} r(\theta,t)\cos\theta \\ r(\theta,t)\sin\theta \\ 0 \end{bmatrix}, \end{aligned}$$

where $O_t(\theta)$ represents a star-shaped cross-sectional closed curve and $C(t)$ and $R(t)$ describe its position and orientation respectively.

Figure 5.1(a) shows a set of key cross-sections and Figure 5.1(b) shows the sweep surface generated by interpolating them. As shown in Figure 5.1, our sweep surface represents a time-variant star-shaped cross-section by a scalar radius function $r(\theta,t)$. This sweep surface can be used as a control structure for bending, twisting and

tapering. Moreover, to achieve local deformations, we can change the cross-sectional shape of a sweep surface by modifying the scalar radius function $r(\theta, t)$.

If the sweep surface $S(\theta, t)$ has no self-intersection, it bounds a volume of magnitude

$$\int A(t) \langle N(t), C'(t) \rangle \, dt, \tag{4.1}$$

where $A(t)$ is the area of the cross-section $O_t(\theta)$ (see [66]) and $N(t)$ is a unit normal vector of the moving cross-sectional plane, which appears as the third column vector of $R(t)$. In Section 4.4, we will show how to use this simple integral formula to solve an interaction problem while preserving (both absolute and relative) volumes among multiple deformations.

4.2.2 Control sweep surface construction

Before changing the shape of an object, its deformable parts must be approximated by sweep surfaces. We start by selecting the parts to be deformed and creating initial control sweep surfaces which enclose those parts. Figure 4.2(a) shows the leg of the well-known Armadillo model and an initial control sweep surface.

In the second step, key cross-sections are computed by cutting the polygonal model of the selected part with planes. The center of each key cross-section is computed, and then radii from the center to the boundary vertices of the cross-section are sampled. Figure 4.2(b) shows a key cross-section with sampled radii. A control sweep surface can then be constructed by interpolating these key cross-sections. The centers of the key cross-sections are interpolated by a cubic B-spline curve $C(t)$ using chord length parametrization, and their orientations (represented by unit quaternions) are linearly

interpolated and normalized to form a rotation matrix $R(t)$. The sampled radii on each key cross-section are interpolated by a B-spline function $r(\theta, t)$. Figure 4.2(c) shows the control sweep surface constructed by interpolating the computed key cross-sections.

In the third step, we can manually edit the positions and orientations of these key cross-sections, insert additional key cross-sections if necessary. Then the second and third steps are repeated until a tightly fitting surface has been generated.

Figure 4.2(d) shows a control sweep surface constructed in this iterative way including the insertion of some additional key cross-sections with manually adjusted orientations. Figures 4.4(b) and 4.5(b) show control sweep surfaces for all the deformable parts of a dinosaur and a bunny model.

4.2.3 Vertex binding and deformation

Once a control sweep surface has been constructed, the vertices in the part of the original model that is to be deformed are bound to that surface. To do this, we first compute the cross-sectional plane that contains a vertex v by solving

$$< v - C(t), N(t) >= 0,$$

where $< \cdot, \cdot >$ signifies an inner product, and $N(t)$ represents a unit normal vector of the moving cross-sectional plane. The local coordinate of the vertex v in that plane is then determined from $\hat{v} = R(t)^T(v - C(t))$. The circular binding angle θ is $arctan(\hat{v}_y/\hat{v}_x)$. Finally, the signed distance d from the control sweep surface is computed as $d = \|\hat{v}\| - \|S(\theta, t) - C(t)\| = \|v - C(t)\| - \|S(\theta, t) - C(t)\|$. Figure 4.3

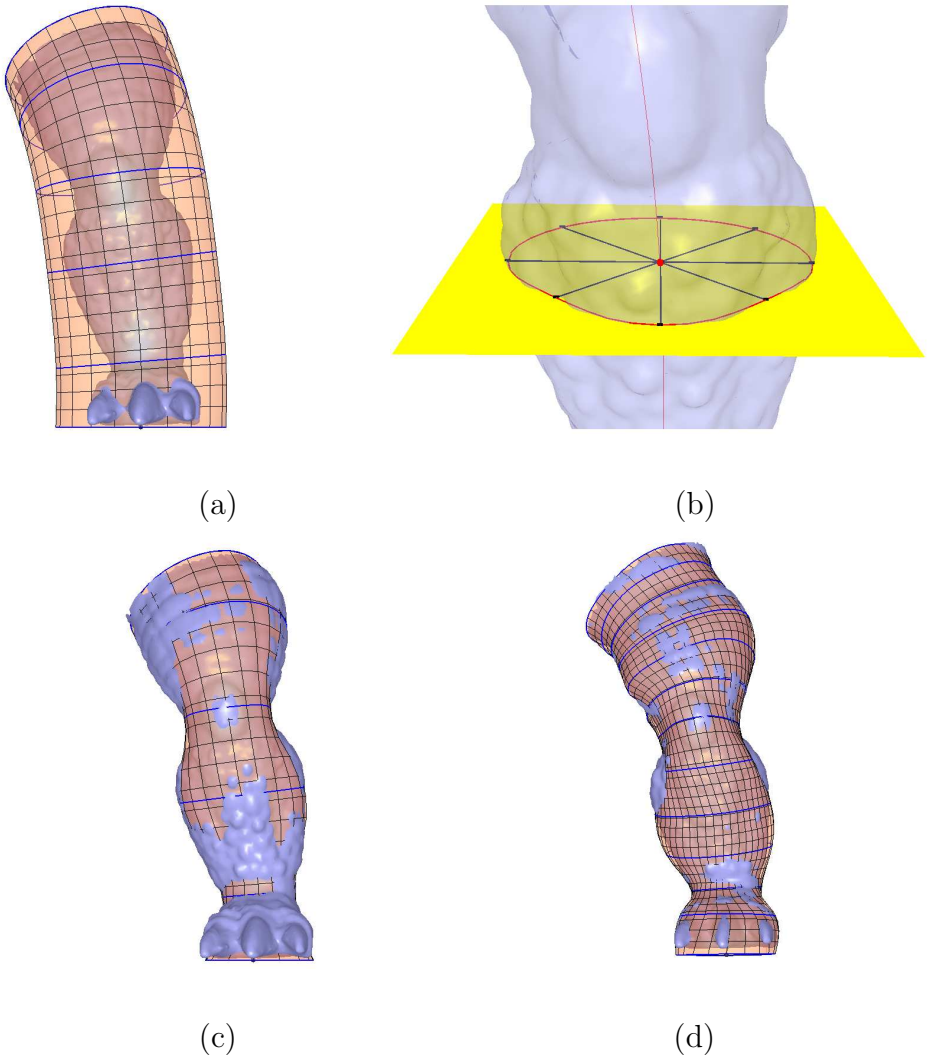


Figure 4.2: Control sweep surface construction:(a) the selected part of the Armadillo leg and the initial control sweep surface, (b) a key cross-section and sampled radii, (c) an intermediate control sweep surface, and (d) the final tightly fitting control sweep surface.

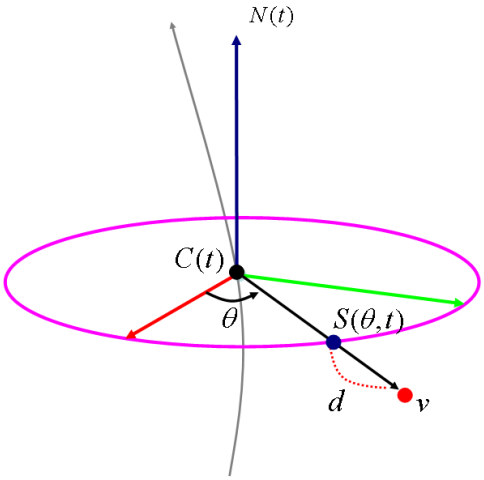


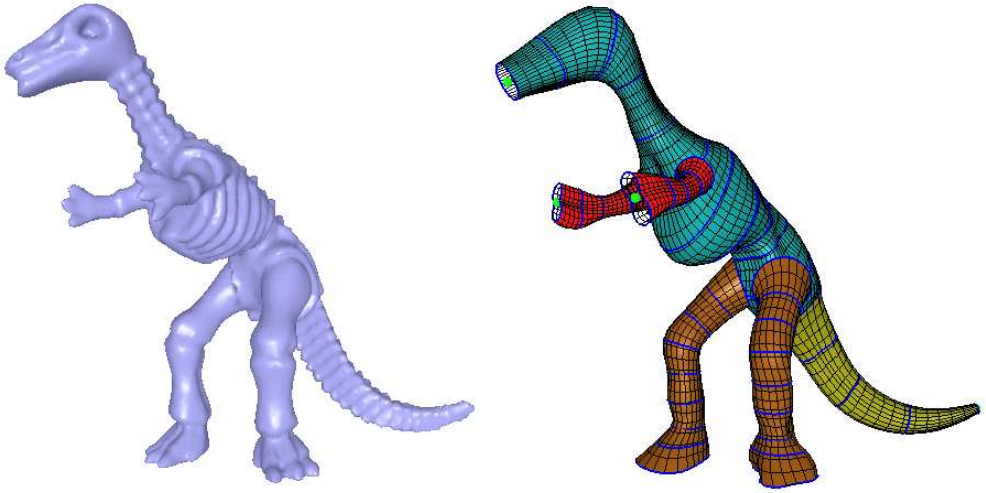
Figure 4.3: Binding a vertex to a control sweep surface.

shows an example of this binding procedure in which a vertex v is bound to a cross-section of a control sweep surface using the binding parameters (θ, t, d).

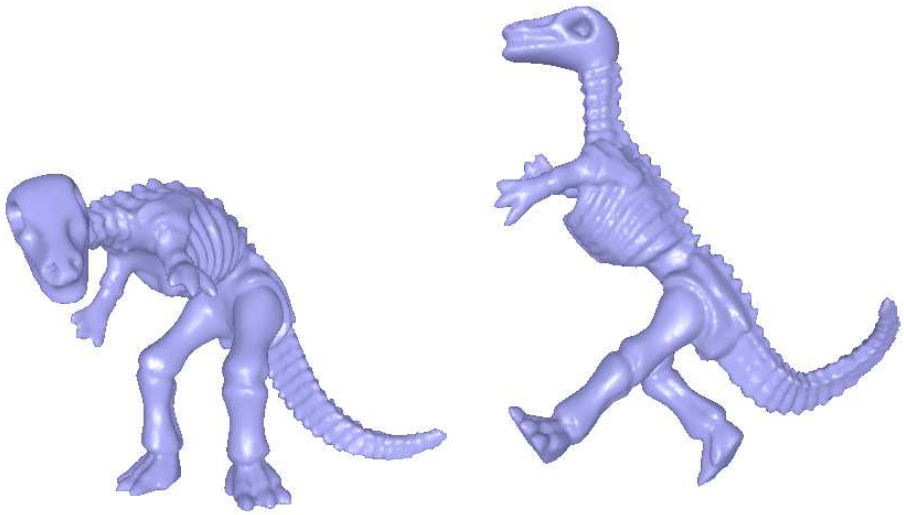
During a deformation, the user can change the shape of the control sweep surface to which all the vertices of the deformable part are bound. Then all the vertices of that part can be reconstructed using the following equation, and will then track the deformation of the control sweep surface.

$$v = C(t) + R(t) \begin{bmatrix} (r(\theta,t)+d) \cdot cos\theta \\ (r(\theta,t)+d) \cdot sin\theta \\ 0 \end{bmatrix}. \tag{4.2}$$

Figures 4.4(c) and 4.4(d) show the deformation of a dinosaur model and Figures 4.5(c) and 4.5(d) show the deformation of the neck and ears of a bunny model, all of which we achieved by editing the positions and orientations of a few key cross-sections.



(a) (b)



(c) (d)

Figure 4.4: Sweep-based deformations:(a) dinosaur model, (b) control sweep surfaces, (c) and (d) deformations of the dinosaur model.

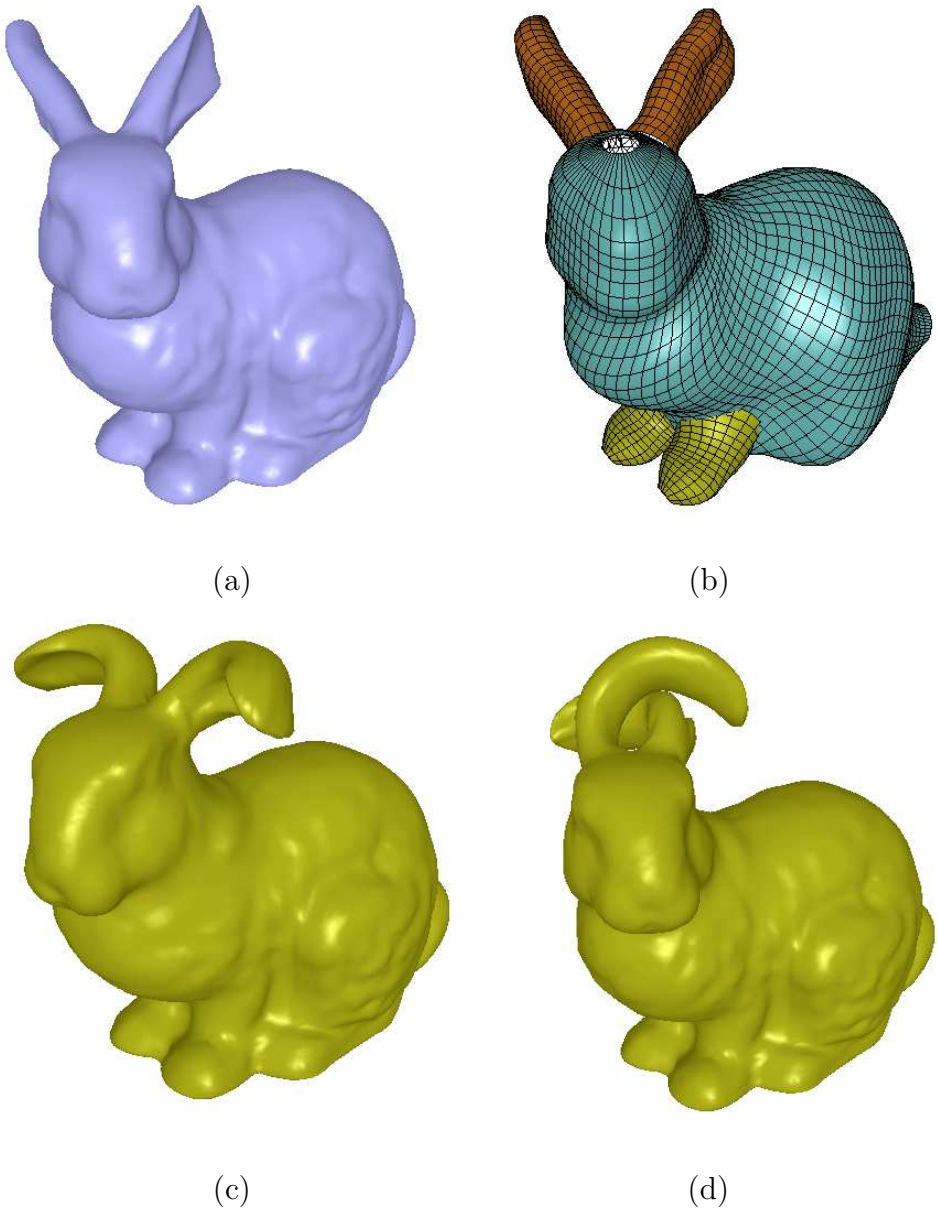


Figure 4.5: Sweep-based deformations:(a) bunny model, (b) control sweep surfaces, (c) and (d) deformation of the ears and neck.

4.3 Sweep Surface Control

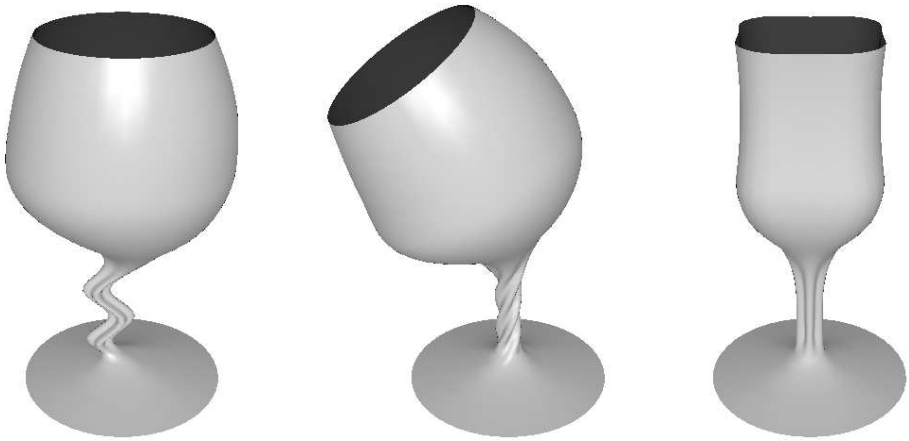
In this section, we will introduce three types of control technique for sweep surfaces: editing key cross-sections; directly controlling a point on a sweep surface; and controlling the position and orientation of an arbitrary cross-section. We start with the editing of key cross-sections and move on to the higher-level control techniques, which provide more intuitive and effective deformations.

4.3.1 Editing key cross-sections

We now consider how to apply various deformations such as bending, twisting, and tapering to sweep-based models by editing their key cross-sections. During a deformation, a user can select one key cross-section X_i and edit its local position and orientation $T_{i-1,i}$ with respect to the previous key cross-section X_{i-1}. These changes affect consecutive key cross-sections $\{X_k\}$, $k = i, i+1, \cdots, n$, and update their global positions and orientations $\{T_{0,k}\}$. Alternatively, we can edit a key cross-section X_i independently of other key cross-sections. We can also easily edit the radius function $r(\theta, t)$ of key cross-sections, in addition to transforming them. The edited key cross-sections are interpolated and the corresponding sweep surface is constructed. Figure 4.6 shows the results of these editing operations.

4.3.2 Direct control of a surface point

Sometimes a user wants to deform the shape of an object directly by editing points on its boundary. When the user picks and manipulates a vertex v with binding parameters (θ, t, d), the corresponding cross-section moves and rotates, resulting in



(a) (b) (c)

Figure 4.6: Editing key cross-sections by changing their: (a) position, (b) orientation and (c) radius.

a new configuration, and $S(\theta, t)$ must change to accommodate this offset. The basic idea of our technique is to edit appropriate key cross-sections so that the sweep surface passes through the changed target point. However, the relation between a point on a sweep surface and its key cross-sections is highly non-linear, and there are many possible ways to modify a sweep so that its surface contains a specified point.

Figure 4.7 shows a strategy based on an infinitesimal editing operation. The user picks a point v on a sweep surface and moves it to a point v', which generates a difference vector Δv at the vertex v. Our strategy is to divide Δv into a linear component Δt and an angular component Δq, and then to start a numerical target tracking process. We first select the lower key cross-section X_i, and consider a sphere centered at the origin p_i of X_i, with radius $\|v - p_i\|$. The vertex v' is then projected on to the sphere, where it generates a new vertex v''. The linear component Δt is the difference $v'' - v'$, and the rotational axis is computed as the cross product

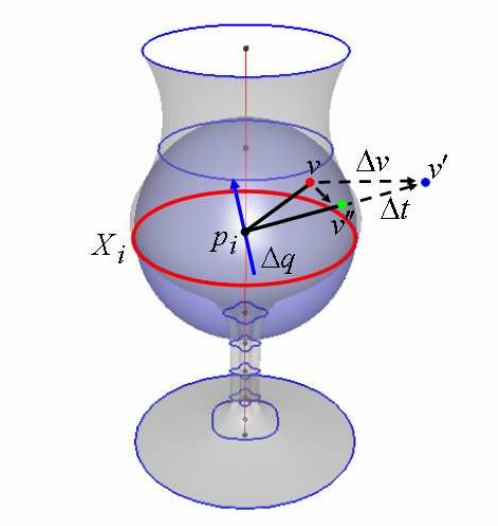
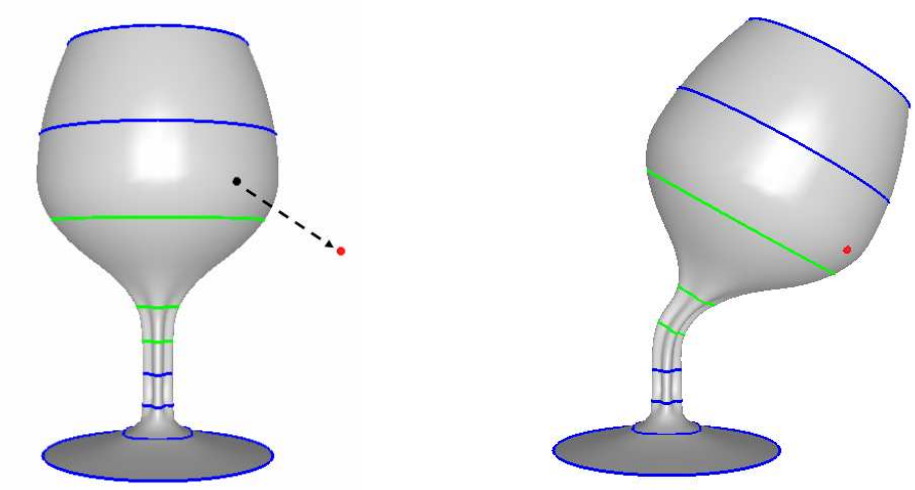


Figure 4.7: The separation of a difference vector Δv.

$(v'' - p_i) \times (v - p_i)$ and then normalized. The magnitude of Δq, which determines the rotation of the cross-section, is computed as follows:

$$\|\Delta q\| = arccos\left(\frac{< v - p_i, v'' - p_i >}{\|v - p_i\| \cdot \|v'' - p_i\|}\right).$$

By applying both the linear and angular difference vectors to the key cross-section X_i, the vertex v approaches the target position v' when a new sweep surface is constructed. To achieve convergence, we repeat this procedure until the distance from v to v' is reduced within a given tolerance. For stability, we propagate control to the k lower key cross-sections $X_i, \cdots, X_{i-k+1}$, starting from the lowest key cross-section X_{i-k+1} and ending at X_i. In general, each key cross-section has 6 degrees of freedom and there are a number of possible ways to bring the sweep on target, of which our technique is only one. Alternatively, we can use just one of the difference vectors to achieve other types of control. Figure 4.8(a) shows a point selected by the user v (in black), the target position v' (in red), and the affected key cross-sections (in green).



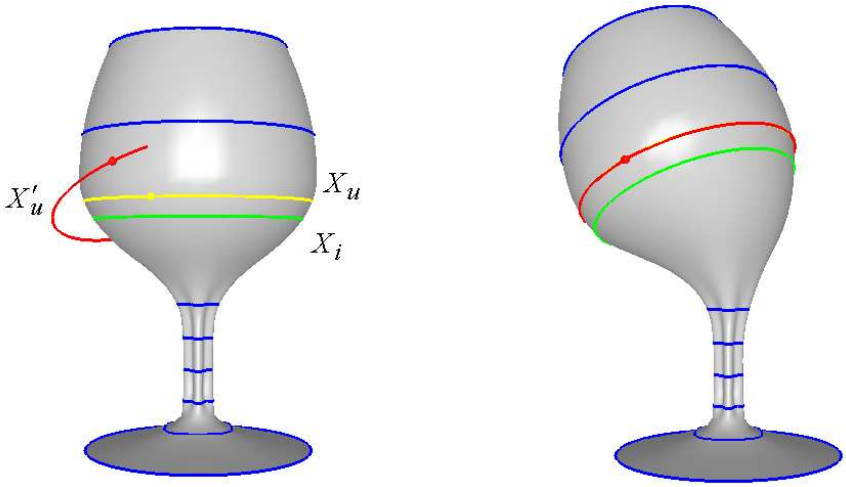
(a) (b)

Figure 4.8: Direct control of a sweep surface.

Figure 4.8(b) shows the result of the control process, in which both the linear and angular components of the change have been combined.

4.3.3 Direct control of an arbitrary cross-section

We now propose another direct control technique for a sweep surface. (Later, we will use this technique to support hierarchy-based interactions among different parts of an object.) During a deformation, the user selects a point on a sweep surface and changes the position and orientation of the cross-section that contains that point. In Figure 4.9(a), X_u is the selected cross-section and X'_u is the new cross-section. We edit the key cross-section X_i which is closest to X_u, assuming that X_u will follow X_i to a large extent, and repeat this process until the transformational distance between X_u and X'_u is reduced within a given tolerance. Figure 4.9(b) shows the direct control of the cross-section X_u using this method. Although we only edited X_i in Figure 4.9,



(a) (b)

Figure 4.9: Direct control of an arbitrary cross-section.

this method can be extended to the editing of multiple key cross-sections, as discussed above.

4.4 Interactions among Deformations

We will now consider the problem of simultaneously controlling the interaction between a number of different deformations of an object using multiple control handles. As a simple example, we will edit the Utah teapot, using separate handles to control the body, spout and handle. Each of these components is approximated by a sweep surface, and its shape is changed by controlling that surface. We need to maintain the consistency of the model's topology during deformations. For example, when the user bends the handle, the body should follow. When the body is modified, the handle and the spout need to track the deformation of the body. We require a

mechanism to specify the hierarchy of these deformations, and rules to control their interactions (see Figure 1.3).

4.4.1 Hierarchy of sweep surfaces

We now show how we specify a hierarchy of sweep surfaces. In the teapot example of Figure 1.3(b), the body is a root node, and the handle and the spout are child nodes. A hierarchy is then specified by binding key cross-sections of child nodes to cross-sections of their parent node. In the teapot example, there are two types of child node. One is a loop, for the handle, and the other is a branch, for the spout. As shown in Figure 4.10, both the first and last key cross-sections of the handle node (loop type), namely X^{handle}_{first} and X^{handle}_{last}, are bound to cross-sections of the body, namely $X^{body}_{handle\text{-}first}$ and $X^{body}_{handle\text{-}last}$. But for the spout node (branch type), only the first key cross-section of the spout, X^{spout}_{first}, is bound to cross-section $X^{body}_{spout\text{-}first}$ of the body. The binding of a key cross-section is a simple extension of the vertex binding technique introduced in Section 4.2, and can be represented by the binding parameters $(\theta, t, d, \hat{q})$. For example, the three binding parameters $(\theta^{handle}_{first}, t^{handle}_{first}, d^{handle}_{first})$ represent the displacement of the center of the key cross-section X^{handle}_{first} from the body sweep surface, as expressed in Equation (4.2), and the fourth binding parameter $\hat{q}^{handle}_{first}$ represents its relative orientation to the cross-section $X^{body}_{handle\text{-}first}$ of the body sweep surface. Using this binding information, the bound key cross-sections change their positions and orientations when the sweep surface of their parent node is deformed. In the teapot example of Figure 1.3(c), the handle is a root node, the body is a child node and the spout is a grandchild node. The rest of the interactions

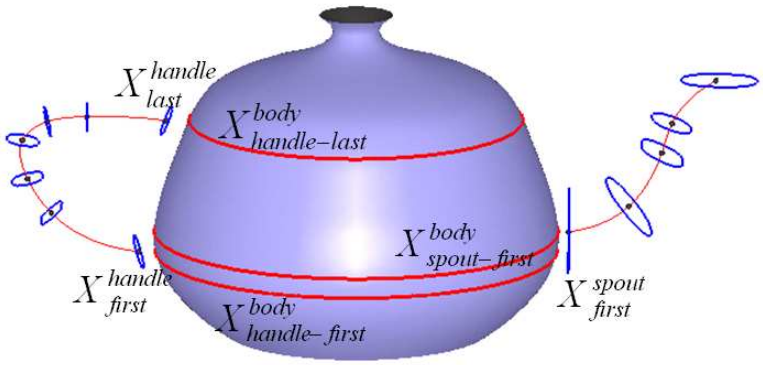


Figure 4.10: Binding of key cross-sections.

can be realized in a similar fashion.

4.4.2 Hierarchy-based interactions

Once the cross-sections of all child nodes have been bound to the sweep surface of their parent node, following the hierarchical structure, we need to solve the problem of interactions among multiple deformations.

In the teapot example, when the user changes the shape of the body, the handle and the spout should follow the deformation of the body. The first key cross-section X^{spout}_{first} of the spout is simply updated using its binding parameters $(\theta^{spout}_{first}, t^{spout}_{first}, d^{spout}_{first}, \hat{q}^{spout}_{first})$, and then the positions and orientations of the rest of the key cross-sections X_j, $j = 2, ..., n$, are automatically updated using the local transformations $T_{j-1,j}$ from the previous key cross-section X_{j-1}. The control sweep surface of the spout node is then reconstructed by interpolating the updated key cross-sections, and all vertices bound to that sweep surface are also reconstructed using their binding parameters. For the handle, the problem becomes more complicated. We can easily

compute the new positions and orientations of the two key cross-sections X_{first}^{handle} and X_{last}^{handle} from their binding parameters. But this may result in an undesirable deformation because the intermediate key cross-sections are not considered. So then we compute the difference between the new transformation of the last key cross-section X_{last}^{handle} and its old transformation. Finally, we apply our direct control technique to the last key cross-section X_{last}^{handle}, using the difference between these transformations. When the direct control technique is applied, all key cross-sections except the first one X_{first}^{handle} are updated iteratively, and the result is a natural deformation of the handle. Figure 1.3(b) shows a deformation of the teapot body in which the handle and the spout change their shapes automatically so as to maintain the topology of the teapot model as the body changes its shape.

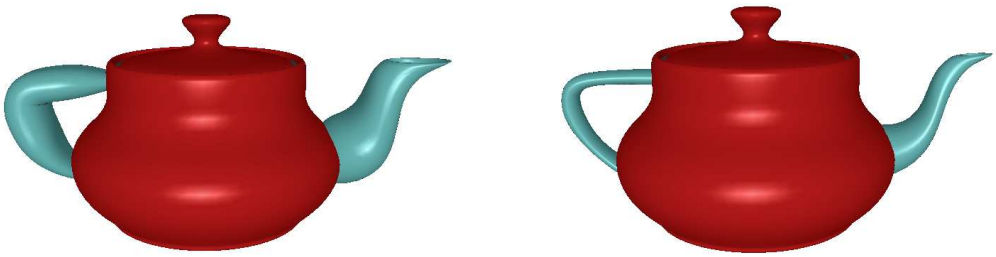
In the case of the deformation of the spout, we do not need to take any additional action, because there is no topological constraint on the spout and it can deform independently. But when we modify the handle, we need to deform the body as well, so that the loop-type topology of the handle and the body is maintained. Figure 4.11 shows the broken topology that occurs when we do not consider the simultaneous control of deformations. To maintain the teapot's topology, we need to apply our direct control technique to the cross-section $X_{handle\text{-}last}^{body}$ of the body to which the last key cross-section X_{last}^{handle} of the handle is bound (see Figure 4.10). When a user deforms the control sweep surface of the handle node, the last key cross-section X_{last}^{handle} changes its position and orientation. From the new values we can compute the target position and orientation of the cross-section $X_{handle\text{-}last}^{body}$ of the body node. The difference between the target and the previous transformation of the cross-section

Figure 4.11: Broken topology during a deformation.

$X^{body}_{handle\text{-}last}$ is then computed. Finally, using the direct control technique for an arbitrary cross-section which we introduced in Section 4.3, we can control the sweep surface of the body so that its cross-section $X^{body}_{handle\text{-}last}$ coincides with the target position and orientation, thus maintaining the topological constraints on the teapot model. Figure 1.3(c) shows the result of a deformation of the handle, in which the body and spout change their shapes automatically as the user bends the handle.

4.4.3 Volume-preserving interactions

The volume of a sweep surface is given in Equation (4.1). Using this integral formula, we have developed two simple interaction rules which can be applied to deformations of an object with multiple control handles. The first is a **total volume preservation** rule which maintains the overall volume of an object during a deformation. When the deformable part of an object changes its shape, it generates a change ΔV in the volume of this part. This difference ΔV is uniformly divided between each control handle, and each radius function $r(\theta, t)$ is updated to reflect



(a) (b)

Figure 4.12: Volume-based interactions:(a) total volume preservation, (b) volume ratio preservation.

to the proportion of the volume distributed to that handle. First, the target volume of each control handle is computed. We then increase or decrease the sampled radii of each key cross-section and repeat the process until the target volume is reached. Figure 4.12(a) shows the result of a deformation of the teapot using the volume-preservation rule, in which the spout and the handle increase their volumes as the body shrinks. The second interaction rule is a **volume ratio preservation** rule, which maintains the ratio of the volume of a child node to that of its parent node. This rule is implemented using a similar update of the radii of key cross-sections, embedded within an iteration process. Figure 4.12(b) shows the result of a deformation using the volume ratio-preservation rule, where the ratio between the volumes of the spout and handle, and that of the body, are preserved.

4.5 Local Deformation

In this section, we extend our sweep-based approach to local deformations. Although our technique supports local deformation by editing a radius function $r(\theta, t)$, this is limited to achieving a cross-sectional change in an object. We propose two other types of local deformations. One changes the local shape of an object boundary using a sweep surface and an influence function. The other captures the local shape of an object (for instance by means of a displacement map) and then transfers it to other objects.

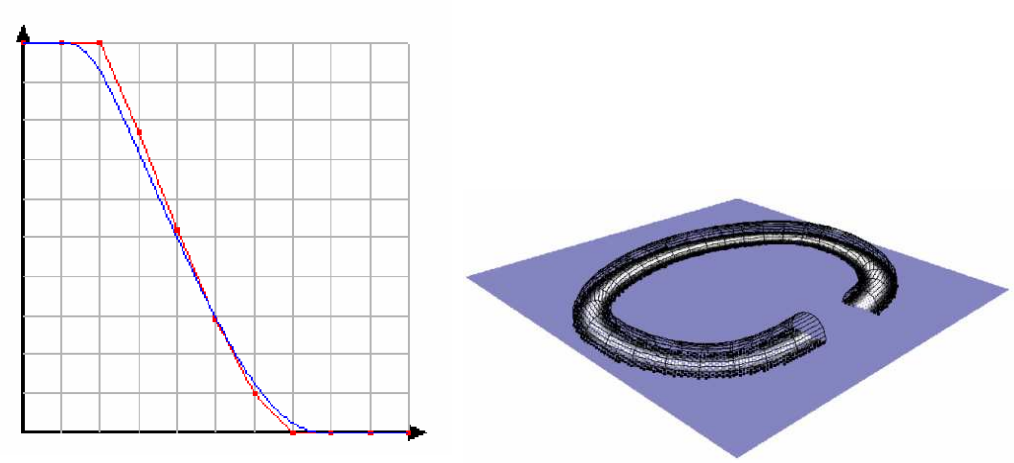
4.5.1 Local deformation by an influence function

We start by creating a control sweep surface on an object boundary. All vertices v of the object contained in the interior of the sweep are bound to the sweep surface and assigned their influence weights w as follows:

$$w = f(u), \quad 0 \leq u \left(= \frac{r(\theta, t) - d}{r(\theta, t)}\right) \leq 1,$$

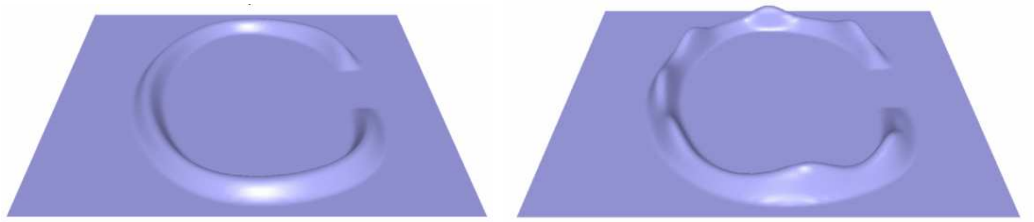
where $f(u)$ is a monotonically decreasing function such that $f(0) = 1$ and $f(1) = 0$, and d is the distance between v and $S(\theta, t)$. Figure 4.13(a) shows an influence function represented by a B-spline function, and Figure 4.13(b) shows a control sweep surface for a local deformation, with the weight assignments shown as grey levels. When the sweep surface deforms, the resulting positions v' of the vertices in the interior of it are linearly blended as follows:

$$v' = (1 - w) \cdot v_0 + w \cdot v,$$



(a) (b)

Figure 4.13: Weight function and control sweep surface.



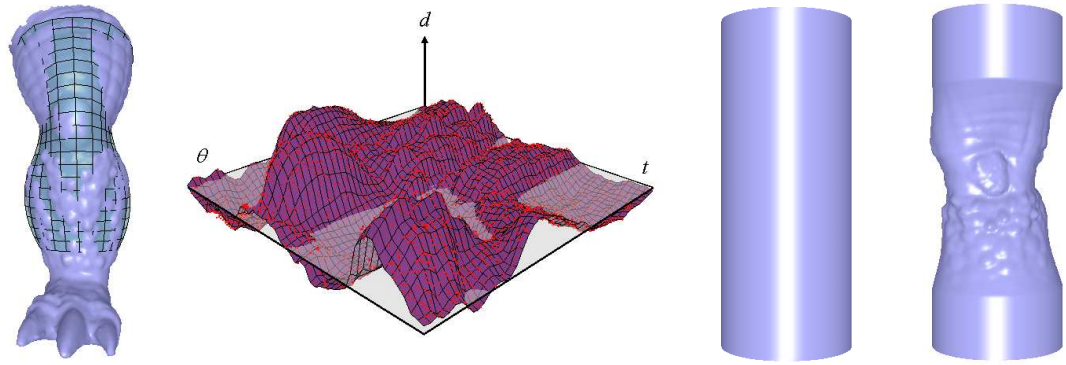
(a) (b)

Figure 4.14: Local deformations.

where v_0 are the initial positions of the vertices in the interior of sweep surface and v are their reconstructed positions using Equation (4.2). Figure 4.14 shows the result of local deformations used to engrave a character C on a plane. In Figure 4.14(b), the character is edited further to generate a wavy pattern. This shows how we can achieve a local deformation of an object boundary.

4.5.2 Shape transfer

We also propose a new technique for transferring shape information from one deformable part to other parts using our sweep-based approach. Once all vertices of a source part have been bound to the control sweep surface, we can transfer the local shape of the source part to a target part using the parameterizations of their control sweep surfaces. The shape of the source part is effectively captured by its control sweep surface and a displacement function that represents its detailed geometry. (Since the displacement values have irregular binding parameters, we use a the scattered data interpolation technique [54] to construct the displacement function.) The vertices of the source and target parts are then bound to their respective sweep surfaces, which have corresponding parameterizations. To achieve a shape transition, we replace the radius function $r(\theta, t)$ of the control sweep surface of the target part by that of the source part. And we also apply the displacement values d of the binding parameters (θ, t, d) of the source part to the vertices of the target part. This transfer the shape of the source part to the target part. Figure 4.15(a) shows the leg of an Armadillo model and a control sweep surface that approximates it. Figure 4.15(b) shows the irregular displacement values and a displacement function that interpolates them. Figure 4.15(c) shows the target part to which the shape of the source part is to be transferred. Figure 4.15(d) shows the resulting shape transition.



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

Figure 4.15: Shape transition.

4.6 Experimental Results

We have implemented our sweep-based freeform deformation technique in C++ on a P4-3.2GHz PC with a 2GB main memory. Our deformation technique works in real-time for all the test examples presented in this thesis; it is simple to implement and easy to use. The most time-consuming step is to construct control sweep surfaces for all the deformable parts of an object. This usually takes from five to ten minutes. To minimize this construction time, we have developed a user interface which includes a semi-automatic process to compute key cross-sections, and their positions and orientations can subsequently be edited manually. The processing time for vertex binding step depends on the number of vertices to be bound to a control sweep surface. Table 4.1 lists the number of vertices in each model and the time required for vertex binding.

Figure 4.16 shows the results of the direct deformation of a bunny model using five control sweep surfaces. In Figures 4.16(a) and 4.16(c), the user picks a vertex

Table 4.1: Number of vertex and binding time (sec).

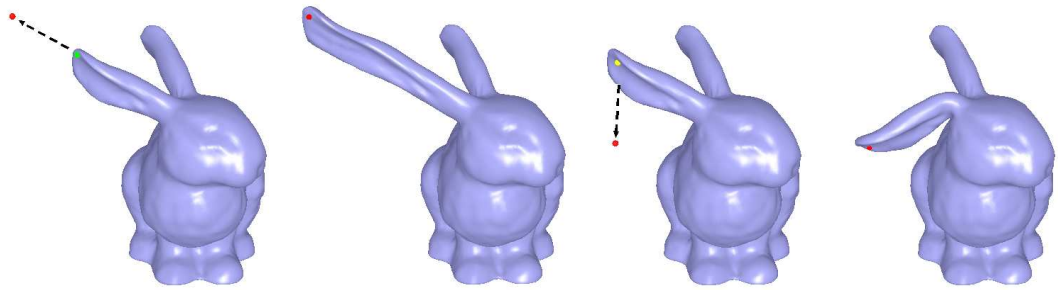
Models	Bunny	Dinosaur	Teapot
Number of vertices	19,226	14,048	12,882
Time for vertex binding	21.714	16.580	13.820

with the binding parameters (θ, t, d) and specifies its target position, which generates a displacement vector. Our direct control technique is then applied to the point $S(\theta, t)$ on the sweep surface. In Figure 4.16(b), only the translational component of the displacement vector is considered; while in Figure 4.16(d), only the rotational component is considered. Figure 4.17 shows the deformation of a teapot model, using the hierarchy-based interaction rule combined with editing of the radius function $r(\theta, t)$ to achieve a local deformation.

We have also extended our technique to support arbitrary interactions among deformable parts. In Figure 4.18(a), four legs of a chair model are approximated by control sweep surfaces. To facilitate interactions among them, they are virtually interconnected by three auxiliary sweep surfaces (shown in red and cyan). Figure 4.18(b) shows the deformation of a front leg with no interaction. In Figure 4.18(c) the user changes the shape of a single leg and both front legs change their shapes because of the presence of a virtual sweep that connects them. In Figure 4.18(d) three virtual sweeps are involved, and all four legs of the chair become elongated.

4.7 Summary

We have extended the sweep-based human deformation technique to a freeform deformation and also shown that this approach provides an excellent control mech-



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

Figure 4.16: Direct deformations of a bunny model.

Figure 4.17: Deformations of a teapot model.

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

Figure 4.18: Deformations of a chair model.

anism for deforming arbitrary three-dimensional objects. Once an object has been approximated with sweep surfaces, it is easy to control shape deformations using a small number of sweep parameters. We have also proposed various control techniques for sweep surfaces, which allow the user to change the shape of an object intuitively and effectively. When an object has multiple deformable parts, it is straightforward to build a hierarchy of deformations using sweeps, and it is also easy to describe the interactions among them. Moreover, our approach supports local deformations by the editing of a radius function $r(\theta, t)$. The main difficulty in our approach is the construction of the control sweep surfaces, which requires some user intervention. This can be ameliorated by incorporating convenient user interfaces or advanced surface fitting techniques.

Chapter 5

Sweep-based Elastic Deformation

In this chapter, we introduce an elastically deformable sweep surface and further extend a sweep-based approach to the elastic deformation of three-dimensional objects. In Section 5.1, we briefly review some representative techniques for modeling deformable objects. Section 5.2 introduces the details of our elastic sweep surface model. In Section 5.3, we describes how the elastic sweep surface responds to user interactions. Sweep-based elastic deformation technique is then presented in Section 5.4 and experimental results are presented in Section 5.5.

5.1 Related Work

Since the pioneering work of Terzopoulos et al. [71], modeling deformable objects has attracted considerable research attention in computer graphics. A wide variety of different approaches have been proposed during the last three decades. Gibson and Mirtich [26] provide a comprehensive survey on this topic ranging from purely

geometric approaches to physically-based ones.

Sederberg and Parry [64] introduced freeform deformation (FFD), which is a representative method for modeling deformable objects. They employed a parallelepiped control lattice to define a tri-variate volume enclosing a deformable object and changed the object shape by deforming the control lattice. Coquillart [18] extended the control lattice to a cylindrical lattice and also a lattice located on top of a surface. Using the Catmull-Clark subdivision scheme, MacCracken and Joy [56] further extended the capability of FFD to lattices of arbitrary topology. Recent work [38, 53] proposed different approaches for freeform shape deformation. In these approaches, a sweep surface is employed as a control structure which provides an excellent control mechanism for modifying existing shapes.

The conventional techniques [18, 53, 56, 64] are computationally efficient and provide a number of control parameters for details of shape deformations in a static sense, e.g. from one shape to a different shape. However, it is rather difficult to produce physically plausible continuous deformations. To overcome these limitations, many previous work explored physically-based methods for modeling deformable objects. By simulating physics rules, these methods produces realistic deformation results in a continuous animation.

Mass-spring systems are a well-established technique in physically-based deformation. Thanks to their simple structure and real-time simulation performance, the mass-spring systems have been widely used for modeling deformable objects. Platt and Badler [60] employed static mass-spring systems for animating facial expressions and Waters [76] extended this method to a more sophisticated model. Terzopoulos

and Waters [72] developed a 3D hierarchical model of a human face using dynamic mass-spring systems and extended this technique to facial models of particular individuals using laser-scanned image data [77]. Tu and Terzopoulos [73] proposed a mass-spring dynamic fish model and developed a physics-based virtual marine world to simulate their behaviors. Recently, Choi et Ko [14] proposed a semi-implicit cloth simulation technique using mass-spring systems, while achieving significant improvements in both stability and realism.

Chadwick et al. [10] simulated muscles in human character animation by combined mass-spring systems with FFD [64], whereas other techniques [14, 60, 72, 73, 76, 77] explicitly modeled deformable objects. The muscles were embedded in control lattices of mass-spring elements and deformed by forces applied to the control lattices. Our approach is similar to Chadwick et al. [10] by combining a geometric approach and physically-based one. The main difference is in the underlying control structure. We employ an elastic sweep surface instead of control lattices. The elastic sweep surface provides an intuitive and efficient control mechanism; moreover, we can demonstrate various other advantages such as real-time performance and numerical stability.

Modeling deformable objects as continuums and simulating their deformations using finite element methods (FEM) is also a very well-established technique in physically-based approaches. Celniker and Gossard [9] applied FEM to freeform shape design paradigm and Gouret et al. [27] used FEM to simulate interactions between a human hand and a deformable object in a grasping task. Although these FEM-based approaches using FEM reflect physics more faithfully than the mass-spring systems, they are computationally more expensive and thus they are not suitable to real-time

applications.

To reduce the computational complexity of FEM, Pentland and Williams [58] first introduced a modal analysis for simulating deformations in computer graphics. They transformed the standard FEM system to linearly independent equations, each of which describes a vibrational mode of an object. Although this approach significantly accelerates the simulation of a deformable object, the modal analysis still generates noticeable artifacts when applied to large deformations or rotational deformations. Recently, Choi and Ko [15] extended the modal analysis and proposed a modal warping technique with real-time performance for large rotational deformations. They identified the rotational components of an infinitesimal deformation and developed a procedure to integrate them to simulate large bending or twisting deformations.

5.2 Elastic Sweep Surface

We describe a sweep surface model with elastically deformable property.

5.2.1 Sweep surface

We employ the sweep surface introduced in Chapters 3 and 4. The sweep surface is generated by interpolating key cross-sections $\{X_i\}$ as follows:

$$\begin{aligned} S(\theta,t) &= C(t) + R(t)\cdot O_t(\theta) \\ &= \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} r_{11}(t) & r_{12}(t) & r_{13}(t) \\ r_{21}(t) & r_{22}(t) & r_{23}(t) \\ r_{31}(t) & r_{32}(t) & r_{33}(t) \end{bmatrix} \cdot \begin{bmatrix} r(\theta,t)\cos\theta \\ r(\theta,t)\sin\theta \\ 0 \end{bmatrix}, \end{aligned}$$

where $O_t(\theta)$ represents a star-shaped cross-sectional closed curve and $C(t)$ and $R(t)$ describe its position and orientation, respectively.

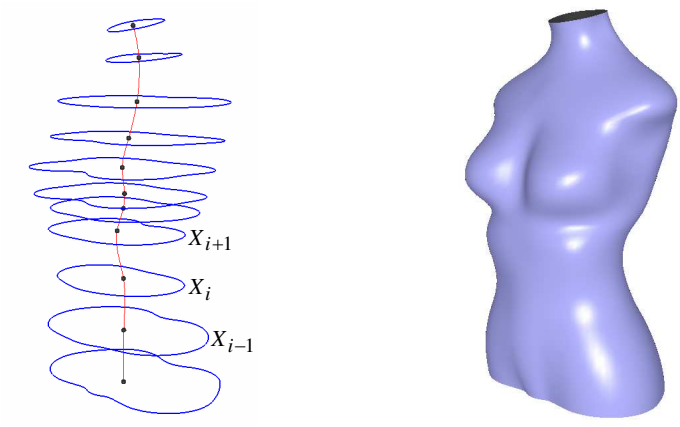
Each key cross-section X_i is associated with a local transformation $T_{i-1,i}$ (represented by a 4×4 matrix) from the previous key cross-section X_{i-1}. Thus, when a key cross-section X_j changes its position and orientation, all the following key cross-sections $X_k(k > j)$ are automatically updated by a series of relative transformations. This arrangement facilitates intuitive deformations such as bending and twisting.

The centers of the key cross-sections are interpolated by a cubic B-spline curve $C(t)$ using chord length parametrization, and their orientations (represented by unit quaternions) are linearly interpolated and normalized to form a rotation matrix $R(t)$. The sampled radii on each key cross-section are interpolated by a B-spline function $r(\theta, t)$. Figure 5.1(a) shows these key cross-sections and Figure 5.1(b) shows the sweep surface generated by interpolating them. As shown in Figure 5.1, our sweep surface represents a time-variant star-shaped cross-section by a scalar radius function $r(\theta, t)$.

5.2.2 Key cross-section with mass-spring systems

We enhance the capability of key cross-sections by including mass-spring systems. We devise three types of mass-spring systems for the elastic changes of the position, orientation and radii of a key cross-section respectively.

The elastic change of the position of a key cross-section is realized by a positional mass-spring system. Figure 5.2 shows a key cross-section X_i with a mass-spring system. The origin of X_i oscillates around the initial position of X_i in the simulation process. Although this positional mass-spring system results in the po-



(a) (b)

Figure 5.1: Sweep surface:(a) key cross-sections, (b) resulting sweep surface.

sition in a three-dimensional space, it is essentially represented by a distance in a one-dimensional space. Figure 5.3(a) shows the initial position of a cross-section (in red) and Figures 5.3(b)-(e) show its elastic deformation results using the positional mass-spring system.

We also develop a rotational mass-spring system for the elastic changes of the orientation of a key cross-section. All rotations in three-dimensional space can be represented by a rotational axis $\hat{\omega}$ and a rotational angle θ. Figure 5.4 shows an example of a rotational displacement of a key cross-section and the rotational mass-spring system. We apply a mass-spring system to the rotational angle θ to generate its elastic changes while fixing the rotational axis. This rotational mass-spring system is also represented by an angle in a one-dimensional space and generates deformation effects such as elastic bending and twisting of a sweep surface. Figure 5.5(a) shows the initial orientation of a cross-section (in red) and Figures 5.3(b)-(e) show its elastic

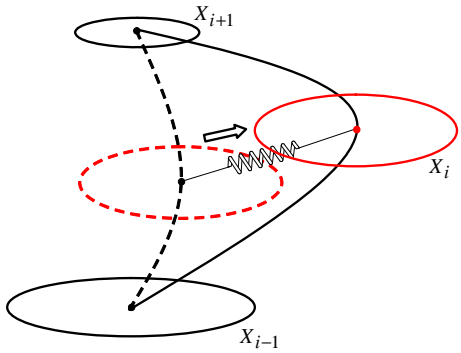


Figure 5.2: A key cross-section connected to its original position.

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

Figure 5.3: A sequence of elastic deformations using positional mass-spring system: (a) the initial position of a key cross-section (in red) and (b)-(e) elastic deformation results.

bending results using the radial mass-spring system.

Finally, we develop a radial mass-spring system for the elastic changes of the radii of a key cross-section. Each key cross-section has a prescribed n number of radial handles and they are circularly positioned on each key cross-section using a uniform angle $2\pi/n$. Figure 5.6 shows an example of a key cross-section where four radial handles are connected to the origin of the key cross-section using a radial mass-spring system. The scalar radius function $r(\theta, t)$ of a sweep surface is generated by interpolating the simulation results of these radial handles. Figure 5.7(a) shows the initial positions of two radial handles (in green) and Figures 5.7(b)-(e) show its elastic

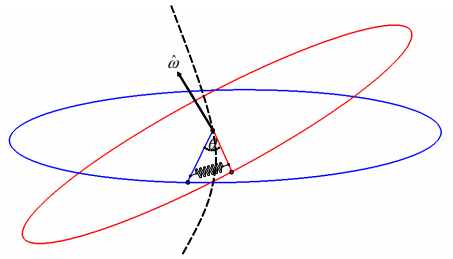


Figure 5.4: Axis-angle representation of a rotational displacement.

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

Figure 5.5: A sequence of elastic deformations using rotational mass-spring system: (a) the initial orientation of a key cross-section (in red) and (b)-(e) elastic deformation results.

deformation results using the radial mass-spring systems.

All of our mass-spring systems of a key cross-section are represented by scalars such as angle and distance, which greatly simplifies a numerical integration process as described in the following equation:

$$m_i \cdot \frac{d^2 X_i(t)}{dt^2} + c_i \cdot \frac{dX_i(t)}{dt} = F_i(t),$$

where m_i are the masses of key cross-sections or radial handles, c_i are the damping coefficients of the mass-spring system, and $X_i(t)$ are the simulation parameters such as angle and distance. The terms $F_i(t)$ are the total sums of spring forces and external forces. For a numerical integration, we simply employ an explicit Euler method with a fixed time step.

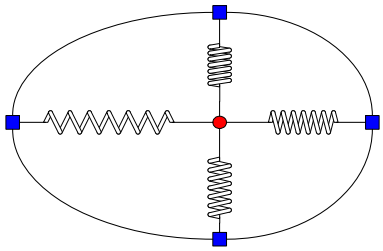
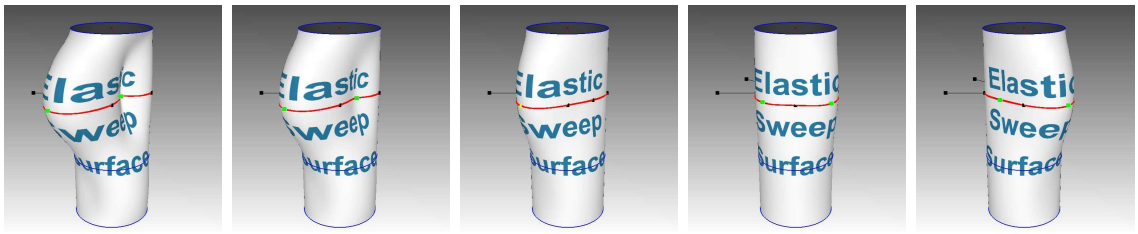


Figure 5.6: A key cross-section with four mass-spring systems.



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

Figure 5.7: A sequence of elastic deformations using radial mass-spring systems: (a) the initial positions of two radial handles (in green) and (b)-(e) elastic deformation results.

Although these mass-spring systems are quite simple and efficient, they have some limitations. Since each elastic change occurs in a key cross-section and it is also limited to a one-dimensional space such as angle and distance, it is rather difficult to generate a plausible shape deformation in a three-dimensional space. These limitations can easily be resolved by combining the simulation results from a sequence of key cross-sections and interpolating them simultaneously. In this way, we can generate plausible complex deformations of an elastic sweep surface that can be simulated in a three-dimensional space.

5.3 Response to User Interaction

In this section, we explain how an elastic sweep surface initiates an elastic deformation in response to the user interactions.

5.3.1 Elastic potential energy

The user can change the positions, orientations and the lengths of radial handles of key cross-sections. These operations generate elastic potential energies of the corresponding mass-spring systems and initiate the elastic deformation of a sweep surface. Figures 5.3, 5.5, and 5.7 show the results of sequential deformations where the user changes the position, orientation and the lengths of radial handles of a key cross-section respectively.

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

Figure 5.8: A sequence of elastic deformations using the direct control technique: (a) the control sweep surface edited directly, (b)-(e): simulation results.

These simulation results are quite simple and limited to one-dimensional spaces. Thus, it is rather difficult to produce plausible deformation results desired by the user. The direct control technique introduced in Chapter 4 for a sweep surface can be applied to generate a compound deformation effect in a considerably more intuitive way. The user picks a point on a surface and moves it to another position, which

generates a difference vector in the three-dimensional work space. The difference vector is then decomposed into rotational components, translational components, and radial offsets of proper key cross-sections. These components are added to appropriate key cross-sections until the distance between two positions are reduced within a given tolerance. These components produce simultaneous changes of the elastic potential energies in the mass-spring systems and initiate an elastic deformation of the sweep surface. Figure 5.8(a) shows the surface under a direct control and Figures 5.8(b)-(e) show the corresponding elastic deformation results in which the elastic potential energies are considered for rotational and radial mass-spring systems only.

5.3.2 External force

In addition to an elastic potential energy, an external force can be applied to initiate an elastic deformation of the sweep surface. In this section, we focus on the external force applied to a point on the sweep surface. Our strategy is to decompose the external force into a rotational force and a radial one for a proper key cross-section and then to apply them to each mass-spring system of the key cross-section. A translational force can be derived in a similar way, however, we do not take account this force. We replace the translational force with a radial one. Using this approach, we can generate compound deformation effects and provide an illusion that deformations are performed in a three-dimensional space.

Figure 5.9 shows an example of the force vector $\vec{f}_{ext}$ which is applied to a point p on the sweep surface. First we derive the rotational axis $\hat{\omega}$ by computing and normalizing the cross product of $\vec{a}$ and $\vec{f}_{ext}$. The rotational force is then computed

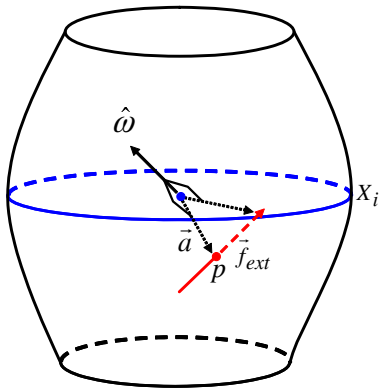


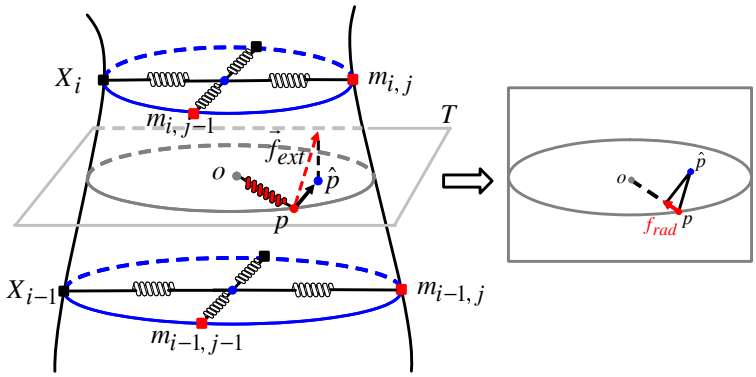
Figure 5.9: Rotational force.

as follows:

$$f_{rot} = C_{rot} \cdot \|\vec{f}_{ext}\|,$$

where C_{rot} is a scale factor for controlling the rotational force; this is necessary because the metric of a rotational force is different from that of a radial one. We apply the rotational axis and the force to the key cross-section X_i. Assuming p will follow X_i to a large extent, the sweep surface initiates the elastic deformations such as bending and twisting. The candidates of X_i can be either the closest key cross-section to the point p or the one determined by the user. When the user specifies the cross-section X_i, we can control the region where rotational deformation effects should occur, which provides an intuitive and flexible control to shape deformation.

Now we deal with how to compute the radial force from the external force $\vec{f}_{ext}$. Figure 5.10(a) shows the force vector $\vec{f}_{ext}$ applied to the point p on a sweep surface. We take the cross-sectional plane T on which the point p lies. The point $\hat{p}$ is obtained by projecting $\vec{f}_{ext}$ onto the plane T. Figure 5.10(b) shows the cross-sectional plane T



(a) (b)

Figure 5.10: Computing radial force

from the top view. The radial force f_{rad} is computed as follow:

$$f_{rad} = C_{rad} \cdot \frac{< \vec{op}, \vec{p\hat{p}} >}{\|\vec{op}\|},$$

where C_{rad} is also a scale factor for controlling the radial force and $< \cdot, \cdot >$ denotes the inner product.

5.3.3 Optimal distribution of an external force

The radial force f_{rad} should be applied to the virtual mass-spring system (in red) as shown in Figure 5.10(a), which connects the point p to the origin o of the cross-sectional plane T. However, we do not have such a mass-spring system since the cross-section T is not one of our key cross-sections X_i. We may add the cross-section T temporarily; however, it would make thins more difficult to deal with since forces can be applied to arbitrary points on the sweep surface. As a practical and reasonable approach, we distribute the radial force f_{rad} to the four nearby handles, $m_{i-1,j-1}, m_{i-1,j}, m_{i,j-1}$ and $m_{i,j}$ of the two bounding key cross-sections X_{i-1} and X_i.

First, initial forces for four handles are computed and optimal forces are then applied to these handles. With these optimal forces, we can imitate the simulation result of the virtual radial mass-spring system of the cross-section T.

Figure 5.11 shows the corresponding parameters of four bounding radial handles $m_{*,*}$ and the point p in the (θ, t)-space. For theses handles, their initial forces are computed as follows:

$$\begin{aligned} f_{rad_{i-1,j-1}} &= f_{rad} \cdot (1-s) \cdot (1-t), \\ f_{rad_{i-1,j}} &= f_{rad} \cdot s \cdot (1-t), \\ f_{rad_{i,j-1}} &= f_{rad} \cdot (1-s) \cdot t, \\ f_{rad_{i,j}} &= f_{rad} \cdot s \cdot t, \\ s &= \frac{(\theta_p - \theta_{i-1,j-1})}{(\theta_{i-1,j} - \theta_{i-1,j-1})}, \\ t &= \frac{(t_p - t_{i-1,j-1})}{(t_{i,j-1} - t_{i-1,j-1})}, \end{aligned}$$

where (s, t) represents a normalized coordinate of (θ_p, t_p) in a local domain bounded by a dashed rectangle as shown in Figure 5.11. Each force is inversely proportional to the area generated by (θ_p, t_p) and the four corners (θ_*, t_*) of the parameter domain. By applying the initial forces to four radial handles, the sweep surface initiates a physical simulation. The red points in Figure 5.12 represent the simulation results using these initial forces, whereas the green points are simulated by the virtual mass-spring (in red) of Figure 5.10(a). As shown in Figure 5.12, the simulation results of the initial forces are quite different from the one by the virtual mass-spring system. Therefore, we need to determine the optimal forces that generate similar results with the virtual mass-spring system.

The optimal forces are computed by minimizing the following functional:

$$\Psi(\lambda) = \|v_0 - v(\lambda \cdot \vec{f})\|^2,$$

where v_0 is the initial velocity of the point p when f_{rad} is directly applied to the virtual mass-spring system and $v(\cdot)$ represents the velocity of the point p when the force vector $\vec{f} = (f_{rad_{i-1,j}}, f_{rad_{i-1,j}}, f_{rad_{i-1,j}}, f_{rad_{i-1,j}})$, (scaled by λ), is applied to four handles. In effect, this is a simple optimization problem in one-dimensional space and a standard line minimization technique [61] can be employed for computing an optimal scalar value λ. The optimal force vector $\vec{f}_{opt}$ (scaled by λ) minimizes the squared difference between two velocities.

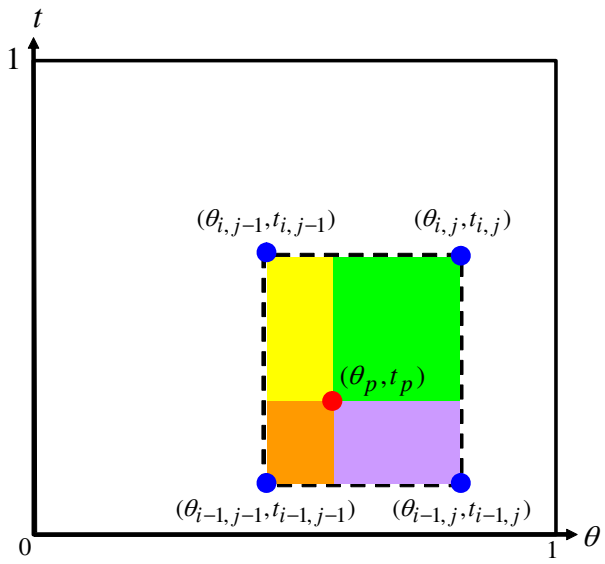


Figure 5.11: Force distribution

The black points in Figure 5.12 represent the simulation results when the optimal forces are applied to four handles. We can confirm that these simulation results are quite similar to the ones simulated by the virtual mass-spring system.

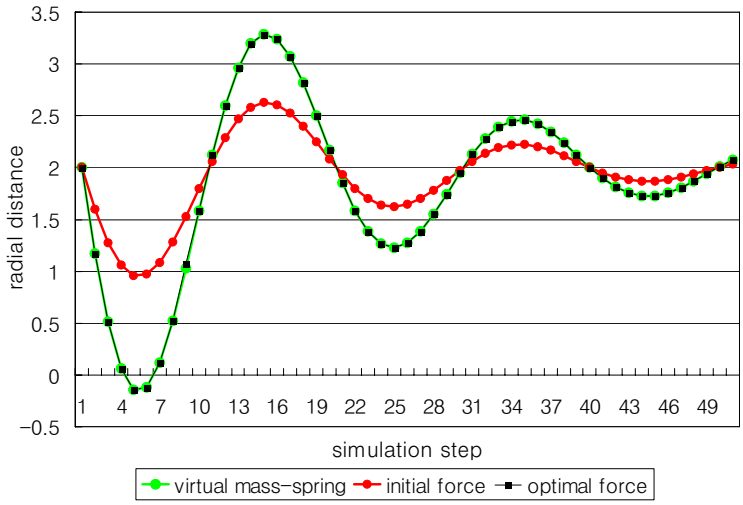


Figure 5.12: Force distribution results

During elastic deformations, successive external forces can be applied to the sweep surface. In this case, the elastic sweep surface should reflect them. The radial mass-spring systems can easily handle these forces by adding optimal forces to corresponding radial handles. However, a similar composition is impossible for the rotational mass-spring system. Even if we add additional axes and simulate their effects, it will be quite difficult to track all external forces. We do not know how many times the external force would be applied to the sweep surface. To avoid this problem, we separate the simulation procedure into two steps. The first step deals with elastic potential energies which are generated by changing the shape of a sweep surface. The second step deals with external forces applied to the sweep surface. Using this two-step procedure, we can easily deal with the successive forces applied to the rotational mass-spring systems. When a new external force is applied to the sweep surface that is under deformation, the current state of the sweep surface, which is generated by a

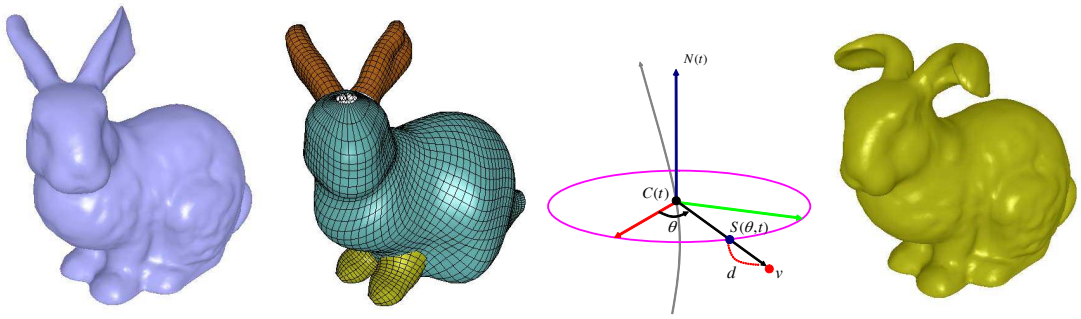
previous external force, is converted to elastic potential energies. By simulating both the elastic potential energy and the new external force simultaneously, we can obtain compound deformation results.

5.4 Sweep-based Elastic Deformation

Using the elastic sweep surface introduced in Sections 5.2 and 5.3, the sweep-based freeform deformation is further extended to the elastic deformation of three-dimensional object. The basic framework of the sweep-based approach can be summarized as follows:

- Given a three-dimensional object, we construct control sweep surfaces that approximate the deformable parts of the object (see Figure 5.13(a) and (b)).
- All vertices of the deformable parts are bound to the cross-sections of the control sweep surfaces using binding parameters (see Figure 5.13(c)).
- As the user controls the underlying control sweep surfaces, the corresponding deformable parts change their shapes (see Figure 5.13(d)).

Figure 5.13 shows the graphical illustrations of this framework. The main difference is that we enhance a control sweep surface to an elastic sweep surface and allow the user to apply external forces so as to control the underlying control sweep surfaces.



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

Figure 5.13: Framework of a sweep-based approach: (a) given three-dimensional object, (b) control sweep surfaces, (c) vertex binding and (d) deformation result.

5.5 Experimental Results

Our elastic deformation technique was implemented using C++ on a P4-3.2GHz PC with a 2GB main memory and an NVIDIA GeForce FX5700 Ultra 128MB graphic card. We used the fixed time step ($\triangle t = 0.1$) for the numerical integration. Our deformation technique works in real-time for all test examples presented in this thesis. Figure 5.14 shows the elastic deformation results of these examples. In Figure 5.14(a), the elastic deformation results of a bunny model are generated using the direct control technique. In Figure 5.14(b), the force applied to the bunny model is decomposed into a rotational force and a radial one, and the subsequent deformation results are generated by applying only the rotational force. On the other hand, Figure 5.14(c) shows the elastic deformation results of a dinosaur model where both rotational and radial forces are applied simultaneously and compound deformation effects are achieved. As demonstrated in these examples, our technique can generate various deformation effects of three-dimensional objects by decomposing an external force and applying them selectively.

Table 5.1: Number of vertex and frame rate (fps.).

Models	# of vert.	Control sweep surface # of keys	# of handles	Average frame rate (fps.) Our method	Modal warping
Bunny	19,226	21	8	38.7	-
Dinosaur	28,096	40	8	19.1	11.9
Head	7,769	5	4	77.5	-

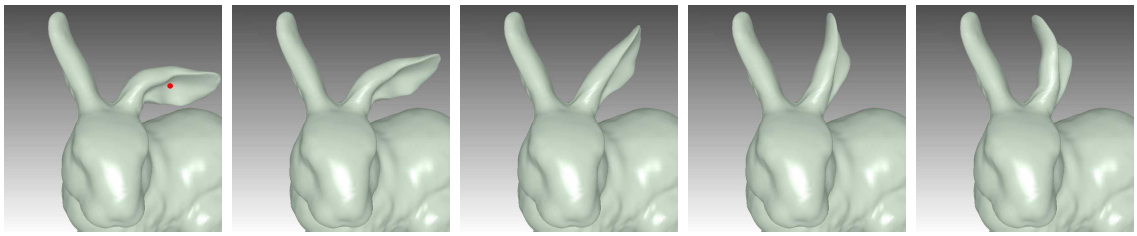
Moreover, our technique provides a flexible control mechanism by specifying the region where elastic bending or twisting effects should occur. Figure 5.14(d) shows elastic deformation results of a head model. In this example, we enforced the second key cross-section of the underlying sweep surface to handle the rotational force obtained by the decomposing technique. All bending effects are simulated in the neck region, whereas a force is applied to the middle position of the model. Figure 5.14(e) also shows the deformation results of the head model for the same external force. However, it shows stiffer deformation results at the same simulation steps compared with the result of Figure 5.14(d) since we assigned different damping constants to the mass-spring systems.

Table 5.1 lists the number of vertices, other data for the control sweep surfaces and the frame rate of deformation for each model. For the dinosaur model, we have compared the performance of our technique with the one of modal warping [15] which is the state-of-art technique in the simulation using a continuum model and FEM. Although we used the low-end graphic card, our technique works about 60% faster than modal warping technique in software implementation.

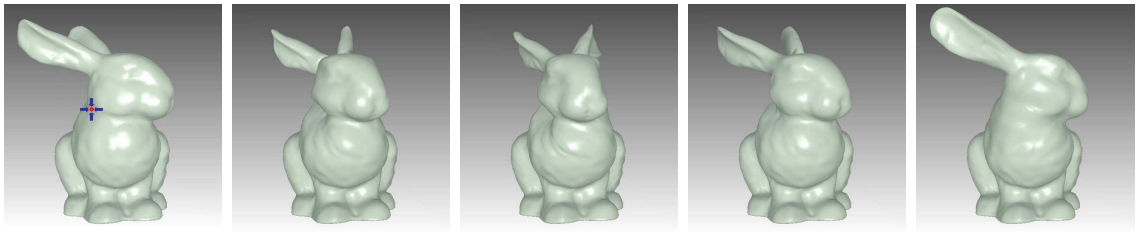
5.6 Summary

We have proposed an elastic sweep surface and extended the sweep-based approach to the elastic deformation of three-dimensional objects. Our technique provided an excellent control mechanism with real-time performance. Simple mass-spring systems were devised for the elastic changes of the position, orientation and radii of a key cross-section, and their simulation results were then interpolated to generate compound deformations of an elastic sweep surface.

Our technique is quite efficient and stable in the sense that the physical simulations are performed in one-dimensional spaces such as angle and distance. Our technique provided various deformation effects by decomposing an external force into a rotational force and a radial one, and applying them selectively. Moreover, it also provided a flexibility by specifying the region where elastic bending or twisting effects should occur.

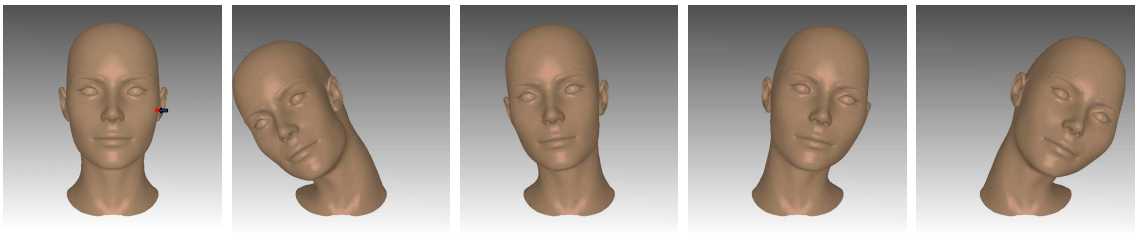


(a)

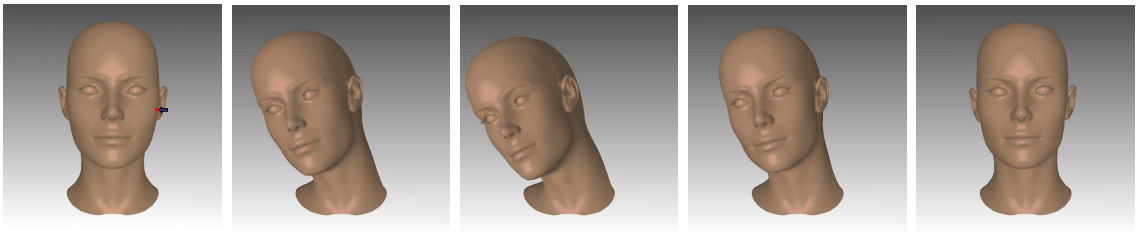


(b)

(c)



(d)



(e)

Figure 5.14: Deformation results of various models.

Chapter 6

Application to a Point Cloud

In this chapter, we introduce a new displaced surface representation using a manifold structure and displacement functions. Using this representation, a detailed geometric model, given as a point cloud is approximated with high precision. Our sweep-based approach is then applied to the control mesh of the displaced surface and smooth deformations are achieved. In Section 6.1, we review some approximation techniques using a displaced surface and also surface construction techniques using manifold theory. The manifold theory and structure proposed by Ying and Zorin [79] are briefly summarized in Section 6.2. In Section 6.3, we construct two geometric functions on each chart of a domain atlas. One is a local surface patch which interpolates the vertices of a control mesh, and the other is a scalar displacement function based on multi-level B-splines. Details of the final construction of a displaced surface are given in Section 6.4. We also propose an algorithm that can approximate a detailed geometric model in Section 6.5. Finally, experimental results are presented in Section 6.6.

6.1 Related Work

We begin with a brief review of some representative approximation schemes that use displaced surfaces and also techniques for surface modeling that are based on the manifold structure of a model.

6.1.1 Approximation with displaced surfaces

Krishnamurthy and Levoy [49] put forward a technique for approximating an arbitrary mesh using B-spline patches and displacement map, using vector displacements to approximate a target mesh. This technique produces a good surface fit, but there is a discontinuity problem across patch boundaries.

Lee et al. [52] proposed the displaced subdivision surface. They constructed a subdivision surface and a map of scalar displacements along its normals. The resulting detailed surface is represented using a unified subdivision framework. However, this approach does not provide an analytic representation for the displaced surface; thus masks must be used to compute differential properties of the surface.

Jeong and Kim [41] proposed a technique for constructing a displaced subdivision surface from an unorganized point cloud. They used the shrink-wrapping approach of Kobblet et al. [48] for the construction of a control mesh, but this method works for objects of genus 0. Recently, Kim and Kim [47] proposed a surface reconstruction technique that works for objects of arbitrary topology. They used a butterfly subdivision surface and a moving least-squares approach to approximate an unorganized point cloud.

6.1.2 Manifold surfaces

A differential manifold is a mathematical structure that defines the topology and geometry of a general space. In computer graphics, Grim and Hughes [31] used this concept in constructing surfaces of arbitrary topology. This technique has also been applied to the parametrization of a surface [29] and to fit a surface to point cloud data [30]. Cotrina and Pla [19, 20] proposed a C^k surface construction algorithm which uses a regular star-shaped configuration to represent an irregular vertex and a polynomial transition function with overlapping charts. The resulting surface is of B-spline form at a boundary and may be seen as a generalized B-spline surface. A more generic approach to freeform surface generation was proposed by Cotrina et al. [21] using manifold theory. They can construct three different types of surfaces, but their techniques require complicated transition functions.

Recently, a simpler surface construction technique has been proposed by Ying and Zorin [79]. They constructed charts in the complex plane with transition functions that have a simple analytic form. Their method provides C^∞ continuity and local support; moreover, the resulting surfaces are visually satisfactory.

Our displaced surface representation is based on the manifold structure of Ying and Zorin [79]. However, for the sake of efficiency, we employ bi-quadratic local patches to produce a domain surface that interpolates the mesh vertices. We also use scalar displacement functions to reconstruct the detailed geometry.

6.2 Manifold Theory

We will now define a general manifold structure, and summarize the 2-manifold structure of the control mesh used by Ying and Zorin [79]. A classical manifold is defined as follows:

Definition 6.1 (Manifold)

k-differentiable manifold of dimension n is a topological space W and a set of pairs $(U_i, \psi_i)_{i \in I}$*, where* U_i *is an open set of* R^n *and* ψ_i *is a k-diffeomorphism of* U_i *over an open subset of W, such that*

1. $W = \bigcup_{i \in I} \psi_i(U_i)$.

2. $\forall i, j \in I$ *such that* $\psi_i(U_i) \bigcap \psi_j(U_j) = X \neq \emptyset$*, the preimages* $\psi_i^{-1}(X)$ *and* $\psi_j^{-1}(X)$ *are open subsets of* R^n *and the map* $\theta_{i,j} = \psi_j^{-1} \circ \psi_i$ *is a k-diffeomorphism.*

□

The diffeomorphisms $\psi_i : U_i \rightarrow W$ are called parameterizations. The pairs (U_i, ψ_i) are the local charts and their collection $\{(U_i, \psi_i)\}_{i \in I}$ is called the atlas. And the maps $\theta_{i,j} : U_i \rightarrow U_j$ are called the transition functions.

Ying and Zorin [79] derived the 2-manifold structure of their control mesh. They defined the chart (U_i, ψ_i) for each vertex in the complex plane, and derived transition functions between the overlapping charts as follows:

$$\theta_{i,j}(z) = \psi_j^{-1} \circ \psi_i = z^{k_i/k_j},$$

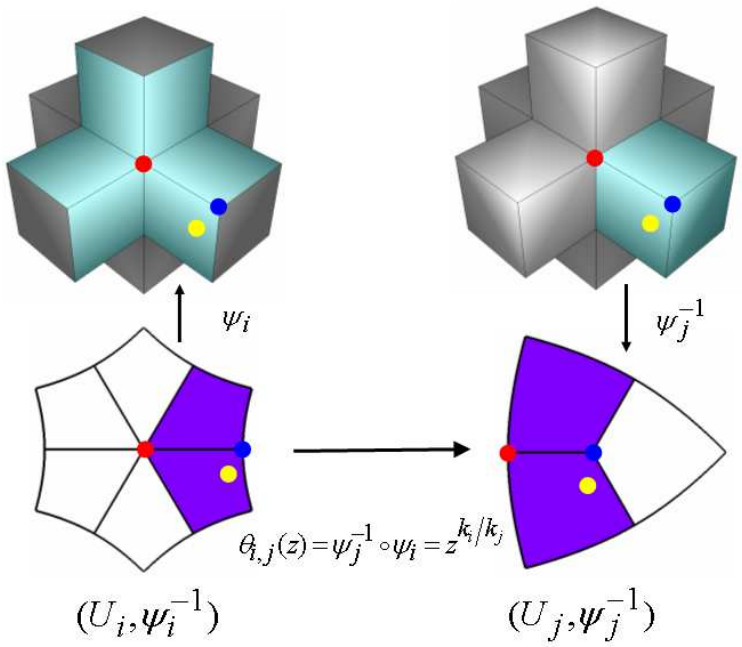


Figure 6.1: The manifold structure of a control mesh.

where k_i and k_j are the valencies of the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ respectively. If z is one parameter in a local chart U_i, then $\bar{z} = \theta_{i,j}(z)$ is the corresponding parameter of z in an overlapping chart U_j. Since the transition function is analytic, it represents a conformal mapping and is easy to compute. Figure 6.1 shows the manifold structure derived from a given control mesh, including the charts and a transition function between two overlapping charts. Ying and Zorin [79] also defined the geometric functions which approximate the subdivision surface of a control mesh, and the scalar blending functions which apply to each chart. Finally, they were able to construct a smooth blending surface using this manifold structure and the blending functions.

6.3 Manifold Structure of a Domain Surface

In this thesis, we employ the same manifold structure as Ying and Zorin [79]. They constructed a local patch on each chart which approximated the subdivision surface of a control mesh, and these patches were represented by polynomials of high degree, determined by the valency of each vertex. Instead, we employ local bi-quadratic patches, which are much simpler. This reduced structure cannot represent detailed geometries and requires a more refined control mesh, but it suffices for the construction of a smooth domain surface and has the great advantage of simplifying the approximation process and reducing the amount of computation. Moreover, our local patches have a useful interpolation property that relates to the vertex assigned to the origin of each chart.

In addition to these local patches, we also define scalar displacement functions on each chart. In the approximation process, these functions are constructed by interpolating the signed distances from the point cloud to nearby local patches using a scattered-data interpolation technique [54]. The detailed surface is then constructed by blending the local patches and the scalar displacement functions in a unified way.

6.3.1 A local patch

Let $\mathbf{v}_i$ be the vertex of a control mesh (Figure 6.2(a)) and let U_i be the corresponding chart. We construct a local patch $P_i(s,t)$ on each chart U_i using the following bi-quadratic polynomial representation, which is constructed by approximating the

positions of the neighboring vertices of $\mathbf{v}_i$ in the control mesh:

$$P_i(s,t) = \begin{bmatrix} s^2 & s & 1 \end{bmatrix} \begin{bmatrix} \vec{c}^{\,i}_{1,1} & \vec{c}^{\,i}_{1,2} & \vec{c}^{\,i}_{1,3} \\ \vec{c}^{\,i}_{2,1} & \vec{c}^{\,i}_{2,2} & \vec{c}^{\,i}_{2,3} \\ \vec{c}^{\,i}_{3,1} & \vec{c}^{\,i}_{3,2} & \vec{c}^{\,i}_{3,3} \end{bmatrix} \begin{bmatrix} t^2 \\ t \\ 1 \end{bmatrix}, \tag{6.1}$$

where $\vec{c}^{\,i}_{j,k} = \begin{bmatrix} x^i_{j,k} & y^i_{j,k} & z^i_{j,k} \end{bmatrix}^T$.

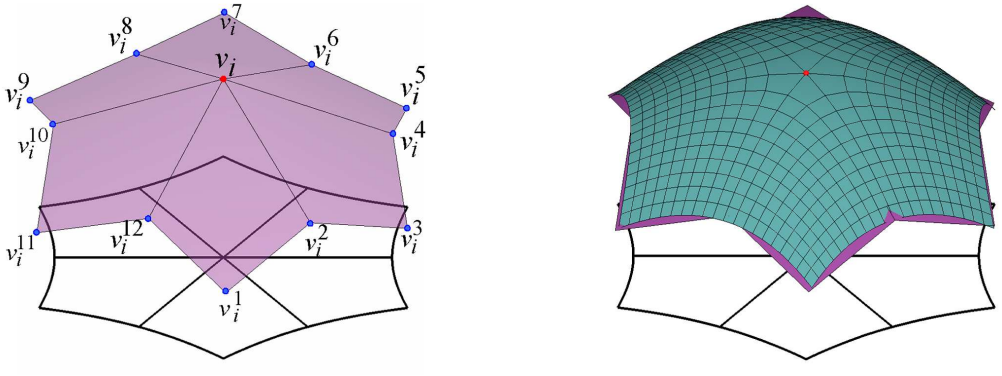
The coefficient vectors $\vec{c}^{\,i}_{j,k}$ of Equation (6.1) are determined by minimizing the following functional:

$$L(\vec{c}^{\,i}_{1,1}, \cdots, \vec{c}^{\,i}_{3,3}) = \sum_{k=1}^{2d} \|\mathbf{v}^k_i - P_i(s_k, t_k)\|^2, \tag{6.2}$$

subject to $P_i(0,0) = \mathbf{v}_i$, where d is the valency of a vertex $\mathbf{v}_i$ and $\mathbf{v}^k_i$ is the k^{th} neighboring vertex of $\mathbf{v}_i$. The constraint represented by this equation forces the coefficient $\vec{c}^{\,i}_{3,3}$ to be $\mathbf{v}_i$ in order to interpolate the position of the vertex $\mathbf{v}_i$, which is assigned to the origin of each chart. As a result, the local patch on each chart can interpolate the position of the center vertex. Figure 6.2(a) shows the vertices of a control mesh around the vertex $\mathbf{v}_i$ and Figure 6.2(b) shows the corresponding local patch. As shown in Figure 6.2(b), the position of the center vertex $\mathbf{v}_i$ is interpolated by the local patch.

6.3.2 Displacement function

Once local patches have been constructed for all the charts, scalar displacement functions $d_i(s,t)$ are created on each chart U_i by interpolating the signed distances



(a) (b)

Figure 6.2: Local patch: (a) the vertices of a control mesh and (b) the constructed local patch.

from each local patch to the point cloud data. For experimental purposes, we generate random scalar displacements. Since these are not regular data, we used a scattered-data interpolation technique based on multi-level B-splines [54]. Figure 6.3(a) shows random displacements on a chart U_i, and Figures 6.3(b) and 6.3(c) show the corresponding displacement functions $d_i(s,t)$ at different levels of detail. Figure 6.3(d) shows the displaced local patch $S_i(s,t)$ generated by applying a displacement function $d_i(s,t)$ to a local patch $P_i(s,t)$. Since it is represented in analytic form, differential properties such as partial derivatives, normal vectors and curvatures can be easily and exactly computed using chain rule and Leibniz rule. The position of a displaced point is computed as follows:

$$S_i(s,t) \;=\; P_i(s,t) + d_i(s,t) \cdot N_i(s,t), \tag{6.3}$$

where

$$N_i(s,t) = \frac{\frac{\partial P_i}{\partial s}(s,t) \times \frac{\partial P_i}{\partial t}(s,t)}{\left\| \frac{\partial P_i}{\partial s}(s,t) \times \frac{\partial P_i}{\partial t}(s,t) \right\|}.$$

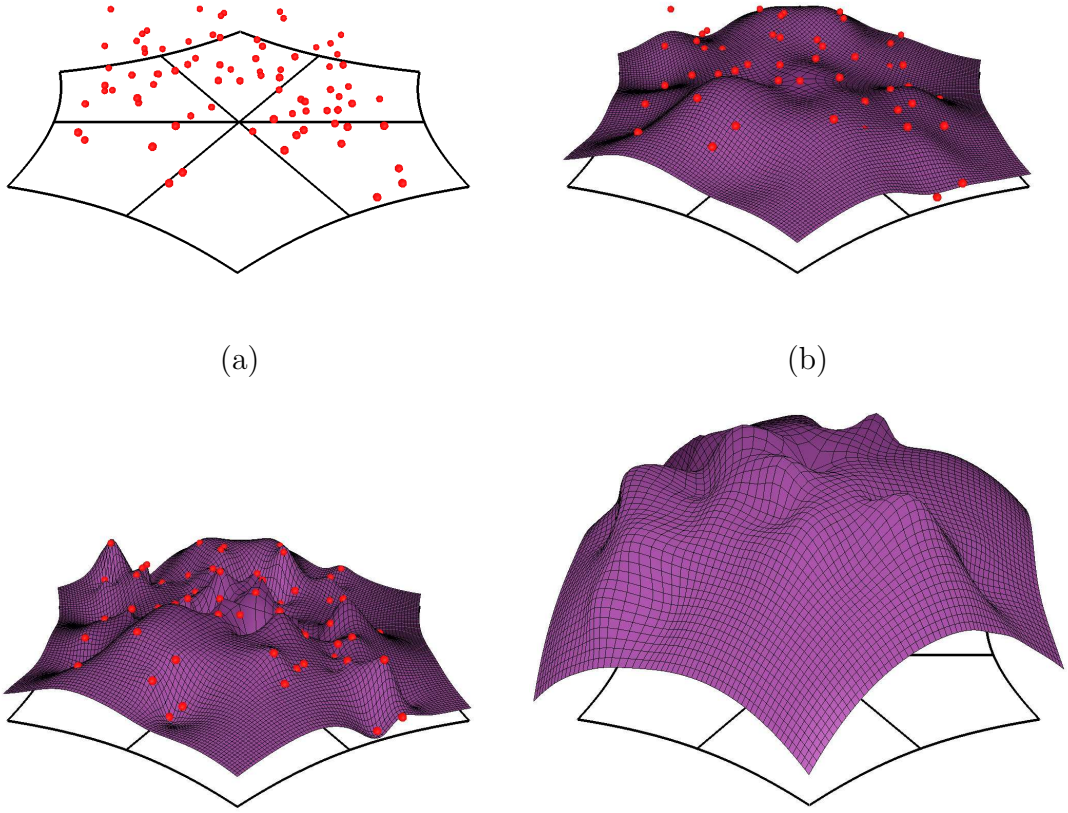


Figure 6.3: Displacement function: (a) randomly generated displacements, (b) displacement function at low level of detail, (c) displacement function at high level of detail and (d) displaced local patch $S_i(s,t)$.

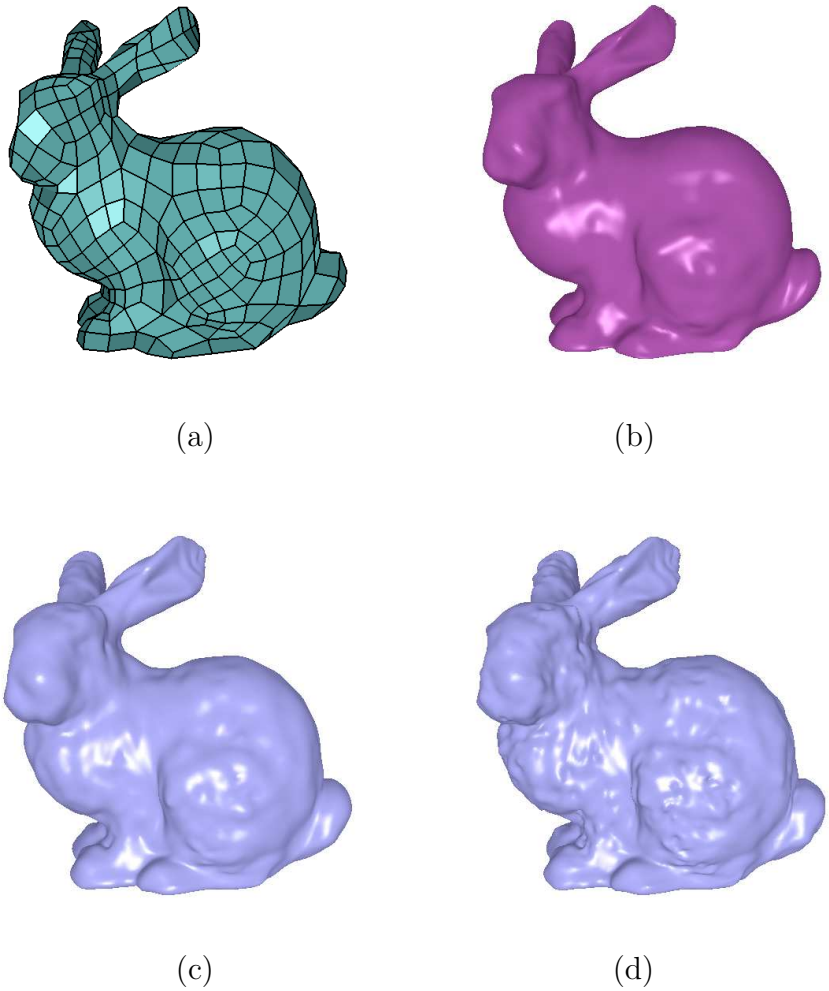
6.4 Displaced Surface Construction

We have now constructed a local patch and scalar displacement function on each chart, and each local patch can interpolate the position of a vertex on the control mesh. We now use the transition functions $\theta_{i,j}(z)$ to represent the correspondence of points and displacements to be blended between overlapping local patches. To achieve a smooth blend, smooth blending functions are necessary on each chart and they must also be normalized. We use the same blending functions as Ying and Zorin [79]. Since the blending function has unit weight at its origin and zero weight at its boundary, the interpolation property of the local patches are preserved in the final blended domain surface. Figure 6.4(a) shows a control mesh and Figure 6.4(b) shows a blended domain surface.

The final displaced surface is achieved by applying the scalar displacement functions to the local patches. These functions are smoothly blended together in the same way as the local patches. For a given parameter value z in a chart U_i, the displaced surface point $D_i(z)$ can be evaluated from Equation (6.4). First we need to find all the overlapping charts $\{U_j\}$ in the atlas which contain z, and then compute the corresponding transition parameters on each chart U_j, using the transition functions $\theta_{i,j}(z)$. Then we compute the displaced points $S_j(z)$ from Equation (6.3). Finally, all the displaced points are blended together smoothly using the weight functions $w_j(\theta_{i,j}(z))$ on each chart:

$$D_i(z) \quad = \quad \sum_{j \in J_z} w_j(\theta_{i,j}(z)) \cdot S_j((\theta_{i,j}(z)), \tag{6.4}$$

where z is the value of a parameter in a local chart U_i, $J_z = \{j | z \in U_j\}$ is an index



(a) (b)

(c) (d)

Figure 6.4: Displaced surface: (a) control mesh, (b) domain surface, (c) and (d) displaced surfaces with different resolutions.

set, and $w_j(\cdot)$ are the smooth blending functions that form a partition unity.

To achieve representations at different resolutions, we change the level of the scalar displacement functions in Equation (6.3). Figures 6.4(c) and 6.4(d) show the displaced surfaces generated by random displacements with different levels of displacement functions.

6.5 Approximation to a Point Cloud

Now we present a new algorithm that approximates a detailed geometric model with our displaced surface. The approximation proceeds in three steps. First we generate a control mesh and the corresponding domain surface from a point cloud. Each data point is then projected on to nearby local patches and initial projection parameters and corresponding displacements are computed. Finally, they are adjusted to minimize the approximation errors and optimal displacement functions are constructed. Our contribution consists of the second and third steps, while the first step is performed using existing methods.

6.5.1 Control mesh

Many advanced techniques [33, 41, 47, 48, 52] exist for the construction of a good control mesh and domain surface from a detailed geometric model. We use a commercial software package [40] which is based on a combination of existing methods. Figures 6.6(a) and 6.7(a) show typical point cloud data and Figures 6.6(b) and 6.4(a) show the resulting control meshes.

In our work with scanned human models, we often discover missing areas in the acquired models. To deal with this problem, we have developed a special user interface for constructing a control mesh from human body scans. The user picks a few feature points on the model, and these are then used to segment the body into torso, arms and legs, and to construct a skeleton structure for subsequent deformation. We then cut the segmented point cloud using planes orthogonal to the skeleton and project all the vertices on to nearby cutting planes. By sampling the vertices on each cut

and connecting those which appear in two consecutive cuts, we can generate a control mesh. Figures 6.8(a) and 6.8(b) show a scanned human model and the corresponding control mesh.

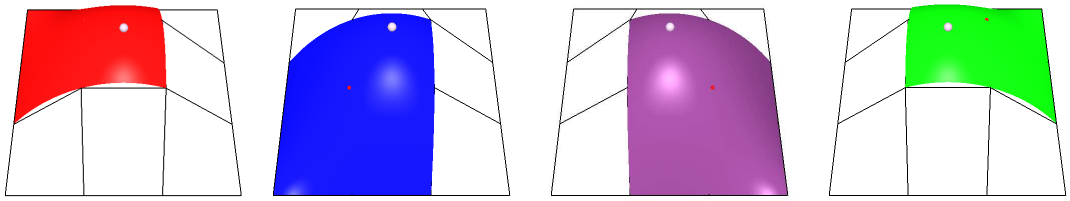
6.5.2 Initial displacement

Once the control mesh has been constructed, a smooth domain surface is generated by blending the local patches. Scalar displacement functions are then constructed on each chart. To construct the optimal scalar displacement functions, we compute initial displacements on each chart and adjust them to minimize the approximation errors. The initial displacements are computed in a different way from previous methods [41, 47, 52], which shoot rays from the domain surface to the detailed geometry. Some of these techniques need connectivity information for the target model. Others [41, 47] are based on guessing the local geometry. Our projection method proceeds in the opposite direction, from the point data to the domain surface, which is considerably easier to implement and produces a more precise result than shooting methods. The main computational expense of our approach is the solution of the following system of non-linear equations, which determine the projection of each point:

$$\begin{cases} F_1(s,t) &= \left\langle \mathbf{p}_i - P_j(s,t), \frac{\partial P_j}{\partial s}(s,t) \right\rangle = 0 \\ \\ F_2(s,t) &= \left\langle \mathbf{p}_i - P_j(s,t), \frac{\partial P_j}{\partial t}(s,t) \right\rangle = 0, \end{cases} \tag{6.5}$$

where $\mathbf{p}_i$ is a point to be projected on to a local patch $P_j(s,t)$.

Assuming that we are using a quadrilateral control mesh, $\mathbf{p}_i$ will be projected on to four local patches P_j, $j = 0, 1, 2, 3$, as shown in Figure 6.5. The use of a bi-



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

Figure 6.5: Projection on to local patches.

quadratic surface representation for local patches greatly simplifies the solution of Equation (6.5).

Let $\hat{\mathbf{p}}_i$ denote the projection of $\mathbf{p}_i$ on to the closest face. Then we find the vertex $\mathbf{v}_0$ of that face which is nearest to the projection point $\hat{\mathbf{p}}_i$. The local patch corresponding to $\mathbf{v}_0$ will be denoted as the patch $P_0(s,t)$. An initial guess of the parameters of the projected point, (s_0^0, t_0^0), is estimated from the coordinates of $\hat{\mathbf{p}}_i$ projected on to the corresponding face of the control mesh. From the initial solution (s_0^0, t_0^0), we compute more precise parameters $(\hat{s}_0, \hat{t}_0)$ using the following Newton iteration:

$$\begin{bmatrix} s^{k+1} \\ t_{k+1} \end{bmatrix} = \begin{bmatrix} s^k \\ t_k \end{bmatrix} - \begin{bmatrix} \frac{\partial F_1}{\partial s} & \frac{\partial F_1}{\partial t} \\ \frac{\partial F_2}{\partial s} & \frac{\partial F_2}{\partial t} \end{bmatrix}^{-1} \begin{bmatrix} F_1(s^k, t^k) \\ F_2(s^k, t^k) \end{bmatrix}.$$

Once the projection parameter $(\hat{s}_0, \hat{t}_0)$ on the first patch has been obtained, the other three projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 1, 2, 3$, are guessed using the transition functions $\theta_{i,j}$. More precise solutions can be computed using a few additional steps of the above iteration. For robustness, we ignore any projection parameters that end up outside their chart.

The displacements $\hat{d}_j, j = 0, 1, 2, 3$, are then computed as signed distances from $P_j(\hat{s}_j, \hat{t}_j)$ to $\mathbf{p}_i$. These become the initial scalar displacements that correspond to the projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$, on each chart. Using existing techniques [41, 47, 52], it is difficult to find the projected points since there are no analytic forms; thus they employed ray-shooting method. On the other hand, our analytic representation greatly simplifies the projection procedure and produces considerably more precise results using an optimization technique to be discussed below.

6.5.3 Optimal displacement function

In general, the four projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$, on each chart do not satisfy the transition relations. Each initial scalar displacement only represents the exact displacement of the point data from one local patch. When the four displacements are blended together, the resulting surface may not interpolate the original point data exactly. This approximation error is due to a mismatch in the transition relations among the projection parameters $(\hat{s}_j, \hat{t}_j)$, $j = 0, 1, 2, 3$. To minimize the approximation error, we need to compute the optimal parameters (s_j, t_j) which satisfy transition relations and their corresponding displacements d_j, $j = 0, 1, 2, 3$. These optimal parameters and displacements are obtained by minimizing the following error functional:

$$\Psi(s_0, t_0, d_0, d_1, d_2, d_3) = \|\mathbf{p}_i - \sum_{j=0}^{3} w_j(s_j, t_j) \cdot (S_j(s_j, t_j) + N_j(s_j, t_j) \cdot d_j)\|^2, \quad (6.6)$$

where the parameter (s_0, t_0) is initialized by the first projection parameter $(\hat{s}_0, \hat{t}_0)$ and $(s_j, t_j) = \theta_{i,j}(s_0, t_0)$ are its transition parameters and the displacements d_j are initialized by $\hat{d}_j$, $j = 0, 1, 2, 3$. From the initial projection parameters and displacements,

we then find the optimal ones which minimize Equation (6.6). This is a general non-linear optimization problem in 6-dimensional space, which is solved by a direction set method [61]. Although the optimal parameters may not be the exact projection parameters to local patches, they minimize the approximation error in a least square sense.

The optimal parameters and displacements are assigned to each chart, and optimal scalar displacement functions are then constructed by interpolating the assigned displacements. The assigned parameters are not distributed in a regular fashion; thus we use a scatter-data interpolation technique [54].

6.6 Experimental Results

We implemented our displaced surface algorithm in C++ on a Pentium-IV(3.2GHz) desktop PC with a 2GB main memory. Figures 7–9 illustrate the approximation process and its results on three different models. The processing time can be divided into two parts: in the vertex projection stage, all the vertices are projected on to nearby local patches, which involves the calculation of optimal projection parameters and displacements, and in the second stage, the optimized displacement functions are constructed on each chart. Table 6.1 itemizes the processing times for each stage, together with the number of points in the point cloud data and the control meshes.

To demonstrate the effectiveness of our approach in practical applications, we applied a sweep-based approach to the approximation of a human model. The vertices of a control mesh are bound to sweep surfaces which approximate the arms, legs and torso. These vertices follow the deformations of the sweep surfaces, and domain

Table 6.1: Approximation results.

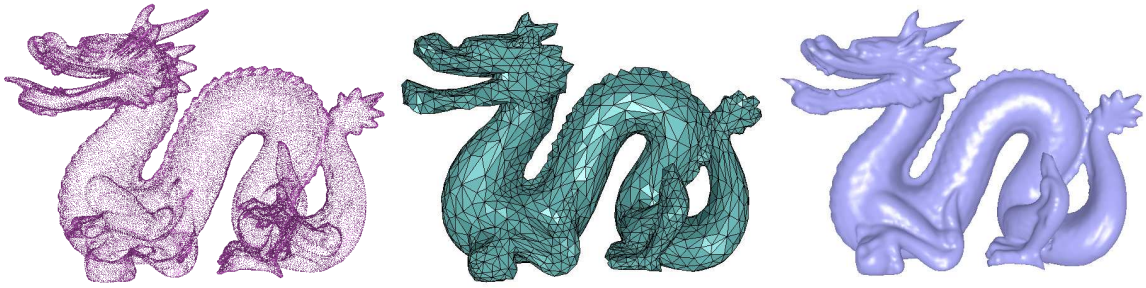
Models	Dragon	Bunny	Human	Armadillo
Number of data points	108,755	34,834	43,817	159,103
Vertices on control mesh	4,095	540	1,134	720
Time for proj. and opt. (sec)	552.53	221.07	318.72	813.72
Time for disp. func. (sec)	10.86	1.43	1.95	4.21
Approx. err. without opt.	$5.06 * 10^{-3}$	$1.78 * 10^{-3}$	$6.86 * 10^{-4}$	$7.84 * 10^{-3}$
Approx. err. with opt.	$3.41 * 10^{-4}$	$1.52 * 10^{-4}$	$1.23 * 10^{-4}$	$2.78 * 10^{-4}$

surfaces deform smoothly while approximating the body shape. Scalar displacement functions are then employed to produce detailed shapes. Figure 6.9 shows clips from an animation made using this technique. It shows that our displaced surface representation can produce continuous and natural deformations of a human model.

Our technique can also be used to construct multi-resolution representations. Our scalar displacement functions are in the form of a multi-level B-spline representation, allowing us to change their level of detail in runtime. Figure 6.10 shows Armadillo model represented by varying levels of the displacement functions. We also analyzed the approximation errors for each model (normalized to unit cube size). The approximation errors (average distances) for each model are listed in the last two rows of Table 6.1.

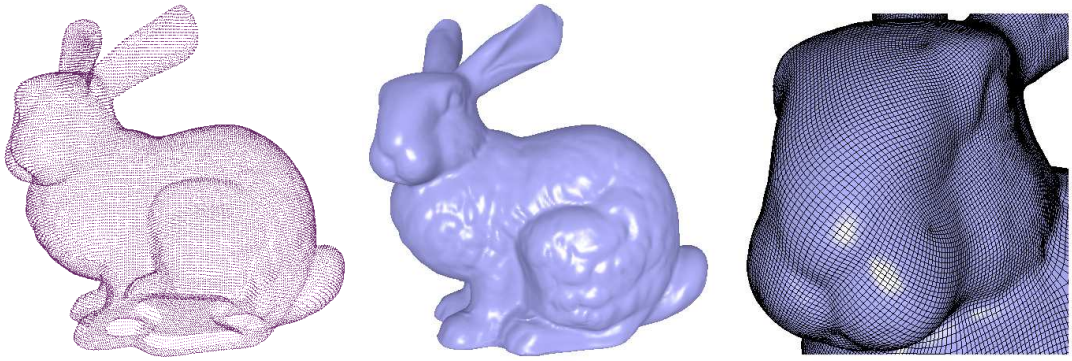
6.7 Summary

We have introduced a new displaced surface representation based on a manifold structure. Our representation consists of simple local patches and scalar displacement functions on each chart of a control mesh. These two geometries are smoothly blended to represent a detailed surface in a unified way. We have also presented an algorithm



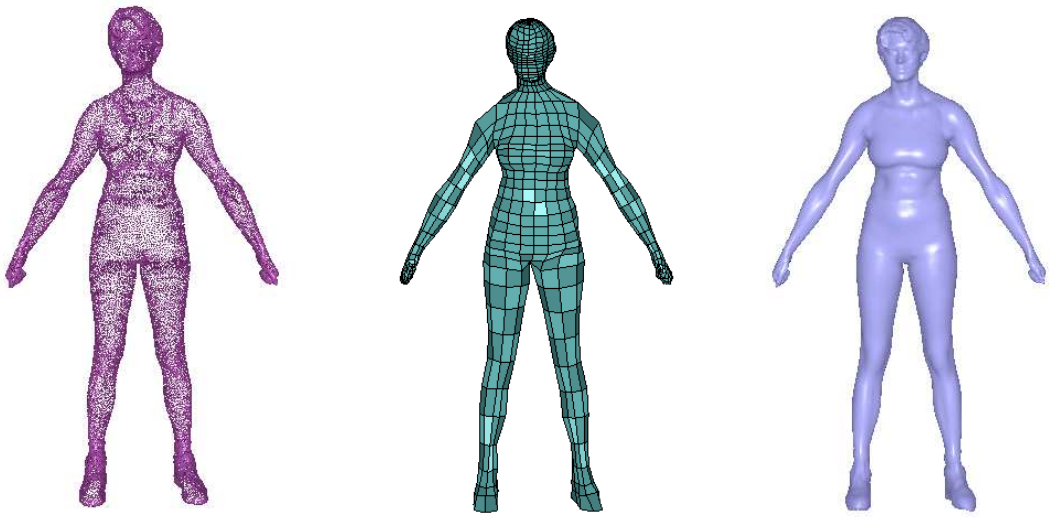
(a) (b) (c)

Figure 6.6: Approximation process: (a) point cloud, (b) control mesh and (c) result.



(a) (b) (c)

Figure 6.7: Approximation process: (a) points cloud, (b) result and (c) meshing of point cloud.



(a) (b) (c)

Figure 6.8: Approximation process: (a) points cloud, (b) control mesh and (c) result.

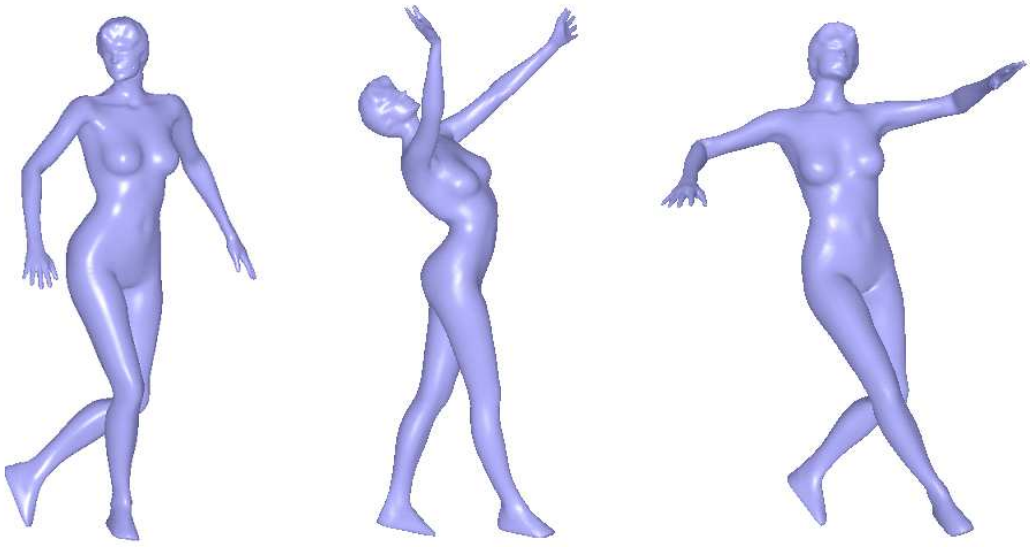


Figure 6.9: Sweep-based deformation results.

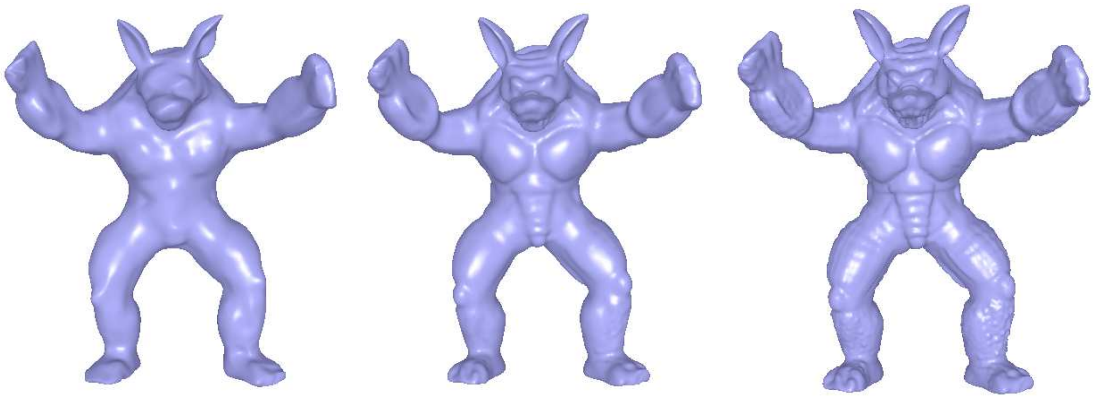


Figure 6.10: Multi-resolution representations.

for approximating a detailed geometric model given as a point cloud. This algorithm is different from existing techniques in the sense that our approach is based on vertex projection and the optimization of displacement functions. The vertex projection technique eliminates the requirement of the connectivity information in the model, and produces precise approximation results incorporating an optimization procedure. Using several experimental results, we have also demonstrated the effectiveness of our approach in various applications such as multi-resolution modeling and sweep-based shape deformation.

Chapter 7

Conclusions

In this thesis, we have presented a sweep-based approach to modeling and deformation of three-dimensional objects and then applied this approach to three practical applications; human deformation, freeform deformation and elastic deformation. In experimental results, we have shown that the sweep-based approach provides an excellent control mechanism for deforming three-dimensional objects and it can be easily extended to physically-based deformations.

Once the deformable parts of an object have been approximated with control sweep surfaces and all vertices of the parts have been bound to the sweep surfaces, it is relatively easy to change the shape of an object by controlling the underlying sweep surfaces. In the human deformation, we utilized the skeleton structure of a model to effectively control the underlying sweep surfaces and then addressed how to connect arms and legs to shoulders and hip using a shape blending technique. Some anatomical features such as elbow and knee protrusion, and skin folding were supported at an interactive speed using a GPU-based collision detection procedure

and muscle bulge effect was also realized using an additional sweep surface. However, it was still difficult to apply all these features to a whole human model at an interactive speed.

We have then extended the sweep-based approach to the freeform deformation of three-dimensional objects. We enhanced the sweep surface using the kinetic structure of key cross-sections and proposed various control techniques, which allow the user to change the shape of an object intuitively and effectively. When an object has multiple deformable parts, we built a hierarchy of deformations using sweeps and solved the problem of various interactions among them. The main difficulty in this technique is the construction of the control sweep surfaces, which requires some user intervention. This can be ameliorated by incorporating convenient user interfaces or advanced surface fitting techniques.

We have further extended the sweep-based approach to the elastic deformation of three-dimensional objects. For this, we enhanced the capability of key cross-sections by applying mass-spring systems and introduced an elastic sweep surface. The elastic sweep surface was generated by interpolating key cross-sections, while adapting dynamic changes in their positions, orientations and boundary shapes. Since the mass-spring systems are simulated in one-dimensional space, they were quite simple, efficient and stable in numerical integration stage. We have also developed a technique that decomposes external forces into different components of mass-spring system to reflect various deformation effects.

Finally, we have applied our sweep-based approach to geometric models represented as point clouds. To efficiently handle a point cloud, we approximated it using

a surface displaced from a manifold. An optimization technique was employed to minimize the approximation errors. Sweep-based approach was then applied to the control mesh of the point cloud. As the control mesh deforms, the displaced surface was reconstructed and the corresponding smooth shape deformations were achieved.

In the current implementation, we focused on applying our sweep-based approach to three-dimensional models represented as polygons or point clouds. In future work, we will investigate the feasibility of controlling the shape of freeform objects which have other representations, such as implicit surfaces or procedural models. And we will extend our freeform deformation technique to support more complex interaction rules and apply our sweep-based approach to other geometric problems such as shape morphing and compression. Furthermore, we will accelerate the performance of our our sweep-based approach using programable graphics hardware.

Bibliography

[1] Alias-Wavefront Technology. *Maya 5.0 Users Manual*, 2003. `http://www.alias.com`.

[2] B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*, 21(3):612–619, 2002.

[3] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parametrization from range scans. *ACM Transactions on Graphics*, 22(3):587–594, 2003.

[4] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. Swirling-sweepers: constant-volume modeling. In *Proceedings of Pacific Graphics*, pages 10–15, 2004.

[5] A. Aubel and D. Thalmann. Interactive modeling of the human musculature. In *Proceedings of Computer Animation*, pages 167–173, 2001.

[6] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall, 1982.

[7] A. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3):21–30, 1984.

[8] O. Bottema and B. Roth. *Theoretical Kinematics*. North-Holland Publishing Company, Amsterdam, 1979.

[9] G. Celniker and G. Gossard. Deformable curve and surface finite-elements for freeform shape design. *Computer Graphics*, 25(4):257 – 266, 1991.

[10] J. Chadwick, D. Haumann, and R. Parent. Layered construction for deformable animated characters. In *Proceedings of ACM SIGGRAPH*, pages 243–252, 1989.

[11] T.-I. Chang, J.-H. Lee, M.-S. Kim, and S.-J. Hong. Direct manipulation of generalized cylinders based on b-spline motion. *The Visual Computer*, 14(5):228–239, 1998.

[12] Y.-K. Chang and A. Rockwood. A generalized de casteljau approach to 3d free form deformation. In *Proceedings of ACM SIGGRAPH*, pages 257–260, 1994.

[13] B. Chen, Y. Ono, H. Johan, M. Ishii, T. Nisihita, and J. Feng. 3d model deformation along a parametric surface. In *Proceedings of International Conference on Visualization, Imaging and Image Processing*, pages 282–287, 2002.

[14] M.-G. Choi and H.-S Ko. Stable but responsive cloth. *ACM Transactions on Graphics*, 21(3):81–97, 2002.

[15] M.-G. Choi and H.-S Ko. Modal warping: real-time simulation of large rotatinal deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):91–101, 2005.

[16] K. Cobayashi and K. Ootusubo. t-ffd: free form deformation by using triangular mesh. In *Proceedings of ACM Symposium on Solid Modeling and Application*, pages 226–234, 2003.

[17] R. Cook. Shade trees. In *Proceedings of ACM SIGGRAPH*, pages 223–231, 1984.

[18] S. Coquillart. Extended free form deformation: a sculpturing tool for 3d geometric modeling. *Computer Graphics*, 24(4):187–196, 1990.

[19] J. Cotrina and N. Pla. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, 17(7):643–671, 2000.

[20] J. Cotrina and N. Pla. Modeling surfaces from planar irregular meshes. *Computer Aided Geometric Design*, 17(1):1–15, 2000.

[21] J. Cotrina, N. Pla, and M. Vingo. A generic approach to free form surface generation. In *Proceedings of the ACM Symposium on Solid Modeling and Applications*, pages 35–44, 2002.

[22] M. Eck, T. DeRose, H. Duchamp, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 303–312, 1995.

[23] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design (4th Ed.)*. Academic Press, Boston, 1997.

[24] J. Feng, L. Ma, and Q. Peng. A new free form deformation through the control of parametric surfaces. *Computers and Graphics*, 20(4):531–539, 1996.

[25] J. Foley, A. van Dam, S. Feiner, and J. Hughes. Computer graphics: Principles and practice, 2nd ed. Addison-Wesley, Reading, Mass.

[26] S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19.

[27] J. P. Gourret, N. Magnenat-Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. In *Proceedings of ACM SIGGRAPH*, pages 21–30, 1989.

[28] N. Govindaraju, S. Redon, M. Lin, and D. Manocha. Cullide: interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 25–32, 2003.

[29] C. Grim. Simple manifolds for surface modeling and parameterization. In *Proceedings of Shape Modeling International*, page 277, 2002.

[30] C. Grim, J. Crisco, and D. Laidlaw. Fitting manifold surfaces to 3d point clouds. *Journal of Biomechanical Engineering*, 124(1):136–140, 2002.

[31] C. Grim and J. Hughes. Modeling surfaces of arbitrary topology using manifolds. In *Proceedings of ACM SIGGRAPH*, pages 359–368, 1995.

[32] A. Hilton, J. Starck, and G. Collins. From 3d shape capture to animated models. In *Proceedings of 3D PVT*, pages 246–257, 2002.

[33] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of ACM SIGGRAPH*, pages 71–78, 1992.

[34] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of ACM SIGGRAPH*, pages 19–26, 1993.

[35] W. Hsu and H. Kaufman. Direct manipulation of free form deformation. *Computer Graphics*, 26(2):177–184, 1992.

[36] S. Hu, H. Zhang, C. Tai, and J. Sun. Direct manipulation of FFD. *The Visual Computer*, 17(6):370–379, 2000.

[37] J. Hua and H. Qin. Free form deformations via sketching and manipulating the scalar fields. In *Proceedings of ACM Symposium on Solid Modeling and Application*, pages 328–333, 2003.

[38] D.-E. Hyun, S.-H. Yoon, J.-W. Chang, J.-K. Seong, M.-S. Kim, and B. Jüttler. Sweep-based human deformation. *The Visual Computer*, 21(8–10):542–550, 2005.

[39] D.-E. Hyun, S.-H. Yoon, M.-S. Kim, and B. Jüttler. Modeling and deformation of arms and legs based on ellipsoidal sweeping. In *Proceedings of Pacific Graphics*, pages 204–212, 2003.

[40] INUS TECHNOLOGY. *RapidForm Users Manual*, 2006.

[41] W.-K. Jeong and C.-H. Kim. Direct reconstruction of displaced subdivision surface from unorganized points. *Graphical Models*, 64(2):78–93, 2002.

[42] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, 2005.

[43] B. Jüttler and M.G. Wagner. Computer aided design with spatial rational b–spline motions. *ASME Journal of Mechanical Design*, 118:193–201, 1996.

[44] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, and D. Thalmann. Real-time animation of realistic virtual humans. *IEEE Computer Graphics and Applications*, 18(5):42–57, 1998.

[45] Kaydara Inc. *FiLMBOX Reference Guide*, 2001. `http://www.kaydara.com`.

[46] M.-S. Kim, E.-J. Park, and H.-Y. Lee. Modeling and animation of generalized cylinders with variable radius offset space curves. *The Journal of Visualization and Computer Animation*, 5(4):189–207, 1994.

[47] S.-J. Kim and C.-H. Kim. Point cloud approximation by displaced butterfly subdivision surfaces. In *Proceedings of Israel-Korea Bi-National Conference*, pages 5–10, 2005.

[48] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. In *Proceeding of Eurographics*, pages 119–129, 1999.

[49] V. Krishnamurthy and M. Levoy. Fitting smooth surface to dense polygon meshes. In *Proceedings of ACM SIGGRAPH*, pages 313–324, 1996.

[50] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 153–160, 2002.

[51] F. Lazarus, S. Coquillart, and P. Jancène. Axial deformations: an intuitive deformation technique. *Computer-Aided Design*, 26(8):607–613, 1994.

[52] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proceedings of ACM SIGGRAPH*, pages 85–94, 2000.

[53] J. Lee, S.-H. Yoon, , and M.-S. Kim. Realistic human hand deformation. *Computer Animation and Virtual Worlds*, 17(3–4):479–489, 2006. (Proc. Computer Animation and Social Agents'06).

[54] S.-Y. Lee, G. Wolberg, and S.-Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.

[55] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH*, pages 479–487, 2005.

[56] R. MacCracken and K. Joy. Free form deformations with lattices of arbitrary topology. In *Proceedings of ACM SIGGRAPH*, pages 181–189, 1995.

[57] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3):562–568, 2003.

[58] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animatio. *Computer Graphics*, 23(3):215–222, 1989.

[59] L. Piegl and W. Tiller. *The NURBS Book, 2nd Ed.* Springer, 1997.

[60] S. Platt and N. Badler. Animating facial expressions. *Computer Graphics*, 15(3):245–252, 1981.

[61] W.-H. Press, S.-A. Teukolsky, W.-T. Vetterling, and B.-P. Flannery. *Numerical Recipes in C.* Cambridge University Press.

[62] F. Scheepers, R. Parent, W. Carlson, and S. May. Anatomy-based modeling of the human musculature. In *Proceedings of ACM SIGGRAPH*, pages 163–172, 1997.

[63] T. Sederberg, Zheng J., Bakenov A., and Nasri A. T-splines and T-NURCCs. In *Proceedings of ACM SIGGRAPH*, pages 477–484, 2003.

[64] T. Sederberg and S. Parry. Free form deformation of solid geometric models. In *Proceedings of ACM SIGGRAPH*, pages 151–160, 1986.

[65] H. Seo and N. Magnenat-Thalmann. An example-based approach to human body manipulation. *Graphical Models*, 66(1):1–23, 2004.

[66] S. Sheynin, A. Tuzikov, and P. Vasiliev. Volume computation from nonparallel cross-section measurements. *Pattern Recognition and Image Analysis*, 13(1):174–176, 2003.

[67] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of ACM SIGGRAPH*, pages 405–414, 1998.

[68] K. Singh and E. Kokkevis. Skinning characters using surface oriented free-form deformations. In *Proceedings of Graphics Interface*, pages 35–42, 2000.

[69] W. Song and X. Yang. Free-form deformation with weighted t-spline. *The Visual Computer*, 21(3):139–151, 2005.

[70] J. Starck, G. Collins, R. Smoth, A. Hilton, and J. Illingworth. Animated statues. *Machine Vision and Applications*, 14(4):248–259, 2002.

[71] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987. (Proc. ACM Siggraph'87).

[72] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis and animation. *Journal of Visualization and Computer Animation*, 1(2):73–80, 1990.

[73] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of ACM SIGGRAPH*, pages 43–50, 1994.

[74] M. G. Wagner. *A Geometric Approach to Motion Design*. PhD thesis, Technical University of Wien, 1994.

[75] X. C. Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 129–138, 2002.

[76] K. Waters. A muscle model for animating three dimensional facial expression. *Computer Graphics*, 21(4):17–24, 1987.

[77] K. Waters and D. Terzopolulos. Modeling and animating faces using scanned data. *Visualization and Computer Animation*, 2(4):123–128, 1991.

[78] J. Wilhelms and A.-V. Gelder. Anatomically based modeling. In *Proceedings of ACM SIGGRAPH*, pages 173–180, 1997.

[79] L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. In *Proceedings of ACM SIGGRAPH*, pages 271–275, 2004.

[80] S. Yoshizawa, A.-G. Belyaev, and H.-P. Seidel. Free form skeleton-driven mesh deformations. In *Proceedings of ACM Symposium on Solid Modeling and Application*, pages 247–253, 2003.

초 록

본 논문은 3차원 물체의 형상 모델링 및 변형을 위한 스윕기반의 접근법을 제시한다. 먼저, 스윕곡면을 이용하여 3차원 물체의 변형하고자 하는 부분을 근사한다. 물체 표면의 정점들은 스윕곡면의 단면 곡선에 바인딩 되어, 곡면의 변화를 따른다. 이러한 접근법을 활용하여, 3차원 인체형상에 대한 모델링 및 변형 기법이 제시되고, 이를 확장하여 임의의 3차원 물체에 대한 자유형상변형 기법과 탄력적 형상변형 기법이 소개된다.

인체형상 모델링 및 변형을 위해서, 사용자가 지정한 특징점들로 부터 단순화된 골격구조가 추출되고, 인체의 팔과 다리 그리고 몸통부분을 근사하는 스윕곡면들이 생성된다. 인체형상의 각 정점들은 가까운 스윕곡면들에 바인딩 되고, 인체형상의 팔, 다리, 척추 그리고 목등이 굽혀지고 뒤트러짐에 따라 스윕곡면들의 변형을 따른다. 뼈의 돌출, 근육의 팽창 그리고 피부의 접힘과 같은 해부학적 특징들도 함께 표현된다.

이러한 스윕기반의 접근법을 확장하여, 임의의 3차원 물체에 대한 자유형상변형 기법을 제시한다. 사용자가 3차원 물체의 변형할 부분을 선택하면 해당되는 스윕곡면이 키 단면들을 보간함으로써 생성된다. 사용자는 스윕곡면을 조정함으로써 물체의 형상을 변형하게 된다. 변형하고자 하는 부분을 근사하는 여러개의 스윕곡면들은 서로 상호작용을 할 수 있도록 계층구조로 구성된다. 스윕곡면들의 계층구조를 이용하여, 여러개의 제어핸들을 가진 복잡한 물체에 대한 제약조건을 만족

하는 형상변형을 지원한다. 스윕기반의 자유형상변형 기법은 또한 체적유지와 형상전이와 같은 중요한 장점들을 갖게된다.

스윕기반의 접근법을 더욱 확장하여, 3차원 물체의 탄력적인 형상변형 기법을 제시한다. 탄력적인 스윕곡면은 키 단면들을 보간하여 생성된다. 이러한 키 단면들의 위치와 방향 그리고 가장자리 형상들은 매스 스프링 시스템의 단순한 물리적 시뮬레이션에 의해 결정된다. 사용자는 외부의 힘을 적용함으로써 물체의 형상을 동적으로 변형하고, 외부의 힘을 회전성분과 단면성분으로 분해하여 다양한 변형효과를 얻게 된다.

스윕기반의 접근법은 점군의 형태로 주어진 기하학적 물체에도 적용된다. 이러한 목적으로, 점군의 데이터를 다양체로 부터 변위된 곡면을 사용하여 근사한다. 점군의 데이터로 부터 제어메쉬가 생성되고, 제어메쉬의 각 정점위에서 국소적인 패치가 생성된다. 점군의 각 점들은 가까운 국소적 패치에 사영되고, 점들의 변위가 최적화 과정을 통하여 조정되어, 최종적 변위곡면은 점군의 데이터를 매우 정확하게 근사하게 된다. 스윕기반의 접근법은 변위곡면의 제어메쉬에 적용된다. 제어메쉬가 변형함에 따라, 변위곡면은 재구성되고, 부드러운 형상변형이 이루어진다. 다양한 실험결과를 통해서, 스윕기반의 접근법의 효율성을 입증하고 또한 이러한 접근법이 3차원 물체를 제어하기 위한 탁월한 방법임을 보인다.

주요어: 스윕곡면, 인체형상 모델링, 자유형상변형, 탄력적 형상변형, 다양체, 변위곡면.

학번: 2001-21510

감사의 글

먼저 지난 수년간, 많이 부족한 저를 지도해 주시고, 물심양면으로 지원과 격려를 아끼지 않고 돌봐주신 김명수 교수님에게 진심으로 감사드립니다. 교수님의 이러한 배려와 격려에 힘입어 마음편히 연구에만 전념할 수 있어, 제게는 너무나 큰 행운이었습니다. 항상 한결같은 모습과 학문에 대한 뜨거운 열정으로 학생들에게 귀감이 되어 주신 교수님의 모습은 평생동안 잊혀지지 않을 것 같습니다.

논문 심사과정에서 많은 충고와 조언을 해주신 신영길 교수님, 고형석 교수님, 이제희 교수님 그리고 이인권 교수님께도 진심으로 감사드립니다. 심사위원 교수님들의 아낌없는 충고와 조언은 제가 학위 논문을 완성하는데 커다란 도움이 되었으며, 앞으로의 연구방향을 위한 좋은 가르침이 되었습니다.

또한, 지난 수년간 함께 생활하면서 제게 많은 도움을 주고, 좋은 추억을 만들어 준 연구실 가족들과 운동연구실의 강훈이에게도 고마운 마음을 전하고 싶습니다.

무엇보다도, 부족한 아들을 지금까지 믿어주시고 지켜봐주신 부모님께 진심으로 감사드립니다. 어린시절 부터 잦은 사고와 말썽을 묵묵히 지켜봐주신 아버지 그리고 세상에서 가장 큰 사랑으로 저를 돌봐주신 어머니, 이 두 분의 믿음과 정성으로 제가 학위과정을 무사히 마칠 수 있었다고 생각합니다. 그리고 아들처럼 따뜻한 사랑과 온정을 배풀어 주시고, 제가 학업에만 전념할 수 있도록 많은 도움을 주신 장인, 장모님께도 진심으로 감사드립니다. 누나들과 매형들 그리고 항상 궂은 일을 도맡아 처리해준 처남에게도 감사의 말을 전합니다.

끝으로, 그동안 저를 믿고 따라준 소중한 아내 민수와 큰 아이 지현이 그리고 곧 태어날 둘째 아이에게도 감사의 말과 더불어 가족의 행복을 위해 항상 최선을 다하는 가장이 되겠다는 약속의 말의 전하고 싶습니다.

이제부터 시작이라고 생각합니다. 지난 수 년간의 학위과정은 저의 부족한 면을 발견하는 깨닭음의 시간이었다고 생각합니다. 항상 자만하지 않고 배움에 대한 겸허한 자세로 이러한 부족한 면을 채워가고자 합니다.