Synthesis of Human and Animal Skin Deformation

by

Guo Zheng

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

at

School of Computer Engineering
Nanyang Technological University

December, 2007

Abstract

To synthesize skin deformations for an articulated character, the skin of the character is decomposed into patches and the deformation of every patch is created individually. Using this strategy of decomposition, two skin deformation methods are proposed: Example Based Transferable Layered Enveloping (EBTLE) and Skinning With Deformable Chunks (SWDC).

EBTLE uses the skeleton to deform the skin through a base control mesh, which is derived from the patch structure of the skin. Based on radial basis function interpolation technique, EBTLE learns how to synthesize the deformed skin shape of a character from its example shapes paired with typical skeleton postures. Furthermore, using the base control mesh as a common platform, with the parameterizations of the skin patches, EBTLE can use the examples of one character to guide the skin deformations of other geometrically similar characters. The experiments have shown that EBTLE can generate aesthetically pleasing results at interactive speed.

SWDC utilizes deformable chunks to represent the internal entities of a character, which are automatically generated beneath the skin depending on user specified patches on the skin. Powered by finite element method, these chunks can deform in a controllable manner according to the change of the skeleton posture. The deformations of the chunks lead to natural deformation of the skin. Intuitive rigs are devised to

facilitate the adjustments of deformation results. The experiments have demonstrated that most of the visual characteristics of skin deformations generated by real muscles can be accomplished by SWDC.

Acknowledgement

This work would not have been done without Dr. Wong Kok Cheong. Thanks for his guidance and advice during my Ph.D program. He helped me to understand all aspects of my research and to follow the right direction.

I would like to thank the head of our division, Dr. Goh Wooi Boon, for the encouragement and support, which helped me finish this thesis.

I would like to thank all the people in the Center of Graphics and Imaging Technology for their kind assistance, which facilitated the development of the research projects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

3D computer animation has demonstrated its fantastic and amazing charm in movies and video games in the last decade. It facilitates artists in expressing their imagination vividly and realistically. Great movies such as Star Wars, Jurassic Park, etc. have shown the magic of 3D animation. More and more movies benefit from the techniques of computer graphics. Another key application of computer animation is video games. Nowadays, CPUs with vastly increased processing power and sophisticated video cards make it possible to produce interactive lifelike 3D games. Both movies and video games face the same essential issue of devising effective systems for the synthesis of realistic virtual 3D characters. An intuitive criterion to judge the computer generated characters in movies and games is their behavior, which reveals their personalities. First of all, synthetic characters should have believable appearances.

The shape of a character can be made in two ways. Modern 3D modeling software (e.g. Maya [2], 3DS max [10]) enables skilled artists to sculpt in the

digital world freely by hand. Laser range scan devices powered by advanced 3D reconstruction methods can capture detailed forms from real objects. However, character shapes obtained via both methods share the same shortcomings: static and not animatable.

The most straightforward approach to animating a virtual character is manually modeling its shape for every frame through either one of the techniques mentioned above, exhaustively and redundantly. In the case of game development, this approach will be impractical if the movements of a character are unpredictable. An alternative approach is to utilize a skeleton to manipulate the character shape based on some deformation mechanisms. A skeleton is a hierarchical structure of joints connected by bones for defining the posture of a character. Thus synthesis of a virtual character can be naturally decoupled into two problems: motion control and skin deformation. A lot of schemes have been devised to control the motion of a character such as keyframing, inverse kinematics, etc. The purpose of skin deformation techniques is to properly change the shape of the skin geometry according to the posture defined by the skeleton. Generally, certain parameters of the deformation mechanisms are accessible to users for adjustments so as to achieve expected effects. Therefore, once the motion of the skeleton is designed, continuous animation sequences of the character shape can be produced automatically.

1.1 Objectives

Devising effective deformation mechanisms so that the skeleton drives the shape of the character skin properly is still a main concern for character animation and is the purpose of this dissertation. Considering the requirements in movie and video game industries, the skin deformation method should have the following features:

- It should be able to automatically elaborate realistic deformations on the skin of a character in accordance with the motion of its skeleton. The deformed shape of the character should look believable and aesthetically satisfactory.

- It should be generic and suitable for a wide range of characters with different skeleton structures (e.g. humanoid characters, animals and fictitious creatures).

- It should provide artists with a convenient and intuitive mechanism to directly describe deformation effects. Whenever animators are not satisfied with the results, quick and intuitive adjustments can be easily accomplished.

- It should keep the laborious manual work of users to the minimum, but meanwhile it should not deprive them of the ability to control the deformation results.

- It should not involve heavy computation at runtime and should be adequate to generate deformations for high resolution characters at interactive speed.

- It should be able to handle both global and local deformations. Global control allows users to describe animation in a high level, while local control enables users to fine tune the surface without affecting areas far away.

1.2 Proposed Solution

It is a challenge to produce realistic deformations on the skin of a computer generated character. Our strategy is to decompose the skin into smaller patches and create the deformation for each individual patch. The movements of the real muscles only cause deformations in limited areas of the skin, so it is natural to synthesize skin deformations patch by patch. Decomposing the skin into patches allows artists to locally specify the deformation and to improve the quality of the deformation in a certain area without influencing other areas. To free the creativity of artists, we provide them with a manual decomposition technique to dissect the skin. More realistic results can be achieved if the skin is decomposed in such a way that the patches are consistent with the real musculature. Based on the decomposition of the skin, we have developed two methods to synthesize deformations for articu-

lated virtual characters, namely *Example Based Transferable Layered Enveloping* (EBTLE) and *Skinning With Deformable Chunks* (SWDC).

1.2.1 Example Based Transferable Layered Enveloping

It is not intuitive to directly interpret the deformation of the skin shape as a function of the variation of the skeleton posture, because it is rather difficult to establish a perfect mapping from the simple hierarchical skeleton to the skin shape. Layered construction of skin deformations has become a common practice for better results and easier control. Example based methods use a set of examples to guide their computational models so that the skin deforms in the same style of the examples. Transferring deformations from one character to another geometrically similar character can save a lot of labor and reuse previously created animation. Example Based Transferable Layered Enveloping (EBTLE) is a framework that successfully combines the three above mentioned methodologies.

EBTLE can deduce the proper shape of the character in a new posture by observing example shapes of a character in given representative postures. With a low-resolution base control mesh derived from the structure of the skin patches, it also incorporates the advantage of layered skin deformation methods. Radial Basis Function (RBF) networks are developed and integrated in EBTLE to approximate the skin surface and its deformations caused by skeleton movements. Since the skin of a character is decomposed into patches, it is easy to build parameterizations for these skin patches. Parameterizations together with the base control mesh provide a common platform and offer the possibility of transferring deformations between different characters. Given a static shape of another geometrically similar character, EBTLE can synthesize its shape in a new posture from the deformation knowledge encrypted in the RBF networks.

1.2.2 Skinning With Deformable Chunks

As in many areas of computer engineering, there is also an issue about "the depth of simulation" in the domain of character skin deformation. EBTLE concentrates on the deformation of the skin surface and relies on examples to adjust the deformation results, so it is a pure kinematic computational model and provides a shallow simulation. Deep simulation of the skin deformations involves not only the skin itself but also the anatomical entities beneath the skin like muscles and fat. Once the anatomical structure of a character is constructed, the skin can automatically deform dependent on its interaction with underlying muscles and fat.

By modeling individual muscles and using them to influence the skin, recreation of realistic skin deformations in the virtual world is very promising. However, the complexity of actual anatomical entities makes the exact simulation of biological processes currently impossible. Simplified muscle models like ellipsoids [103], deformed cylinders [128] are used instead. Even in this case, building the anatomical structure of a synthetical character demands specifications of type, shape and placement of every individual muscle, and therefore needs a lot of labor. Skinning With Deformable Chunks (SWDC) is a new skinning model devoted to alleviating this problem. Like EBTLE, this skinning model is also based on the decomposition of the skin mesh.

Since skin is the only visible entity, as long as the internal entities can cause appropriate deformations on the skin, they do not have to resemble real anatomical entities. SWDC uses *deformable chunks* as the internal entities of a character. According to the patches defined on the skin of a character, deformable chunks can be automatically generated beneath the skin. All chunks connect with each other and form a continuous layer, to which the skin is attached. The alteration of the skeleton posture causes the deformations of chunks, which in turn result in the change of the skin shape. To make the underlying chunks deform realistically, we utilize Finite Element Method (FEM) to calculate the deformations of the chunks

during the animation.

1.3 Thesis Outline

The remainder of this dissertation is organized as follows. In Chapter 2, we give a review of existing deformation methods developed in the last two decades including geometric methods, physically based methods and skin deformation methods. In Chapters 3 and 4, the underlying concepts and computational models of the two proposed skin deformation methods, EBTLE and SWDC, are described in detail. The experimental results of the two methods are also included in Chapters 3 and 4 respectively. Chapter 5 concludes with discussion and possible future work.

Chapter 2

Related Work

Simulation of deformations has been investigated for decades in computer graphics across many applications. For computer aided design, deformation tools are used to modify and sculpt desired object shapes. In virtual reality, besides human beings and animals, there exist a large number of deformable objects like basketball, clothes etc. For medical applications, digitized organs deform subject to virtual surgical devices. There are two classes of methods reported in the literature of 3D computer graphics for modeling deformations. Geometric methods are usually based on cheap computational models and they alter the shapes of deformable objects according to the modification of configurations of deformers (such as lattices, wires etc). Usually, artists interactively change the parameters of deformers until desired results are achieved. Physically based methods automatically calculate the deformation of an object governed by physical laws. However they generally incur heavy computation.

The deformations on the skin of an articulated character are a specific kind of

deformations which in reality are caused by underlying musculature. Synthesis of realistic skin deformations requires the combination of science and art. A variety of algorithms have been designed to endow synthetic characters with life. We will focus our discussion on algorithms that are used to create deformations on the body of a character. The similarity of these methods is that they deform the skin of a character based on the posture of a skeleton. When the skeleton changes its posture, these methods change the shape of the character properly. Some cartoon characters can elongate their arms dramatically or do some other exaggerated shape changing. The deformation algorithms required for those characters are not the emphasis of our study. Although some artists may use a skeleton to create facial animation, most of the methods for facial animation [93] are not skeleton based. Facial animation is usually driven by the emotion and voice of the character, and problems encountered by facial animation are different from issues concerned by body deformation. For example, in facial animation, the movements of lips have to synchronize with the voice. We are not going to dig into the area of facial animation in this review.

Since the methods for skin deformation usually utilize general deformation techniques, including geometric methods, physically based methods or their combination, it is necessary to take a look at these general deformation methods before we discuss the methods specially designed for skin deformation. After geometric and physically based methods are discussed in Section 2.1 and Section 2.2 respectively, skin deformation methods for articulated figures are reviewed in Section 2.3.

2.1 Geometric Methods

Geometric methods emerged in the earliest stage of the research for modeling deformations. Manipulating the shapes of objects by changing their vertices or control points becomes highly inconvenient as the number of vertices or control points grows. Geometric deformation methods can provide a higher level of control of the

shapes of objects. One of the first spatial deformation approaches was presented by Barr [14], which specifies the deformation by affine transformations from $\mathbb{R}^3$ to $\mathbb{R}^3$. Due to the limited types of transformations, only some particular types of deformations are introduced. Bechmann [15] gave a survey about various geometric deformation methods. Milliron and his colleagues [78] presented a general framework that gives a uniform description of different geometric deformation methods based on feature specifications.

2.1.1 Lattice Based Free Form Deformation

Instead of applying transformations to the entire space, Free Form Deformation (FFD) confines the transformations to the space defined by a lattice. With an object entirely or partially embedded in the lattice, users can interactively deform the shape of the object by manipulating the control points of the lattice. It is one of the most important deformation techniques. The primary work of free form deformation was done by Sederberg and Parry [104]. They introduced the parallelepiped lattice that supports the definition of trivariate Bernstein volume. Using the Bernstein polynomials, a mapping from the local space defined by the lattice to the world space is constructed.

$$\mathbf{p} = \sum_{i,j,k} B_i(u) B_j(v) B_k(w) \mathbf{x}_{ijk}, \tag{2.1}$$

where $B_i(u), B_j(v), B_k(w)$ are Bernstein polynomials, $\mathbf{x}_{ijk}$ are the control points of the lattice. Before the deformation, the coordinates (u, v, w) of vertex $\mathbf{p}$ of the object in the local space are estimated. After the control points of the lattice are changed by users, the new position of vertex $\mathbf{p}$ is calculated using the same local coordinates (u, v, w) through Equation 2.1 with updated $\mathbf{x}_{ijk}$.

Coquillart [28] extended free form deformation by allowing prismatic lattice or composite of prismatic lattices, which results in flexible design of the deformation space and eases the process of realizing certain types of deformations. A further extension is the lattice with arbitrary topology proposed by MacCracken

and Joy [74]. The generalized Catmull-Clark subdivision scheme is utilized to subdivide the deformation space into a converging sequence of small cells, in which the relative coordinates of an object vertex can be estimated numerically. This technique has no restriction on the design of the lattice. However, the construction of the lattice topology still could be tedious. Moccozet and Magnenat-Thalmann [79] proposed Dirichlet FFD that removes the topology of the lattice. Only a set of control points are utilized and the local coordinates of the object vertices with respect to the control points are computed as the Sibson coordinates based on Delaunay triangulation. This technique has been successfully applied in the task of hand simulation.

Although free form deformation is intuitive and convenient, it is difficult to displace certain vertices of an object to the desired positions, and hence, to attain the exact specific shape of an object is not a trivial task. Hsu et al. [54] presented a scheme that can directly manipulate the positions of object vertices. The positions of control points of the FFD lattice are automatically computed to ensure the positions of specific object vertices are precisely placed in the desired locations. Composed FFD introduced by Hagenlocker and Fujimura [46] employs a sequence of unbounded uniform periodic B-spline FFD lattices and determines the positions of the control points of the lattice based on the trajectories of certain feature points. To make the deformation of an object exhibit some physical characteristics, Hirota et al. [53] used volume preservation constraint to update the positions of lattice control points.

2.1.2 Constraint Points Based Deformation

Borrel and Bechmann [17] introduced a deformation model for n-dimensional objects. The approach first transforms a n-dimensional object into a higher dimensional space and then projects it back into n dimensional space with deformed shape. The transformation of points from $\mathbb{R}^n$ to $\mathbb{R}^m (m > n)$ is accomplished by a function $\mathbf{f}$. The inverse mapping of points from $\mathbb{R}^m$ to $\mathbb{R}^n$ is done by multiplying

with a $n \times m$ matrix $\mathbf{M}$. The displacement of point $\mathbf{p}$ after deformation is given by:

$$\mathbf{d}(\mathbf{p}) = \mathbf{M}\mathbf{f}(\mathbf{p})$$

The values of the elements of matrix $\mathbf{M}$ are chosen so that the displacements of selected constraint points satisfy the user specifications. With this deformation model, the control of the object shape only involves manipulation of several constraint points, thus the manual work conducted by users are kept to the minimum.

Based on this deformation model, Borrel and Rappoport [18] proposed Simple COnstrained DEFormation (SCODEF). It utilizes B-spline functions to define $\mathbf{f}$ and associates every constraint point with a radius of influence. An extension of SCODEF is proposed by Raffin and his coworkers [100]. A geometric hull is used to specify the influence of a constraint point instead of a simple radius, and the trajectories of the constraint points are utilized to control the deformations. Derived from Borrel and Bechmann's n-dimensional deformation model, Aubert and Bechmann [9] presented a deformation method that preserves the volume of the deformable object during the animation. The matrix $\mathbf{M}$ is estimated in such a way that the resultant deformation minimizes the volume variation of the object and at the same time the displacements of constraint points still fulfill the requirements of users.

2.1.3 Other Types of Deformers

Curve is an intuitive tool that can be used to control the shape of a deformable object. A 3D axis that can bend, twist and stretch is used by Lazarus et al. [70] to appropriately deform an object. As the 3D axis is tweaked into a new shape, the vertices of the object are transformed to the new positions by the local frames defined along the 3D axis. The deformation technique introduced by Chang and Rockwood [25] adopts a Bézier curve with handles at the ends. The space is warped along the curve in the similar way as the points along the Bézier curve are evaluated, and intuitively follows the curve to accomplish several types of deformations.

Singh and Fiume [108] proposed a curve based deformation method called wires. The deformation is calculated according to the difference between the current and original shapes of the wire. Multiple wires can be used to sculpt complex deformations.

A set of triangles with arbitrary topology, namely t-FFD, is utilized by Kobayshi and Ootsubo [68] to define the deformation of an object. The manipulation of the triangles results in the proper change of the positions of the object vertices. Milliron et al. [78] proposed a new method based on their general deformation framework, which uses a coarse mesh to deform the detailed surface of an object.

Decaudin [35] presented a geometric deformation technique by merging the deformable object with simple 3D shapes like sphere and ellipsoid. These simple 3D shapes can be easily used to create bumps or dents on the surface of the object. By sweeping a tool (a simple shape), Sweepers introduced by Angelidis et al. [4] can create intuitive foldover-free deformations along the path of the tool. Hua and Qin [55] used the scalar field defined by an implicit function to guide the deformation of an object.

2.2 Physically Based Methods

Creating realistic deformations using geometric methods is dependent on the skills of artists. For a physically based deformation model with the relevant boundary conditions defined, it can automatically produce deformations approximating what is seen in the real world. Generally, physically based simulation of deformable objects demands heavy computation. Over the last twenty years, the computational power of hardware has been dramatically improved and a great variety of advanced techniques have been proposed to reduce the cost of the simulation. Consequently, these improvements have increased the popularity of physically based methods in computer animation. A survey about physically based methods of earlier research has been given by Gibson and Mirtich [41]. Nealen et al. [86] presented the state

of art report including recently developed techniques.

2.2.1 Approximate Continuum Models

Some earlier work of physically based simulation is not exactly based on continuum mechanics. Certain physical quantities are approximated to ease the computation or to achieve particular desired effects. Physically based deformation modeling was pioneered by Terzopoulos et al. [118]. They applied the Lagrange's equation with finite difference method to describe the behavior of a deformable object. The potential energy of the object caused by the deformation is estimated using the difference between the fundamental forms of the deformed and original shapes. Later this framework was extended to describe inelastic deformations [116] and to handle deformations with rigid rotational components [119]. Based on physics and optimization theory, Platt and Barr [96] introduced reaction constraints and augmented Lagrangian constraints to facilitate the control of the deformation results. Constraints that prevent flexible objects penetrating other objects were described by Baraff and Witkin [11]. Metaxas and Terzopoulos [76] presented a dynamic deformation model with generalized coordinates and used Lagrange multipliers method to constrain the behavior of a deformable object.

More recently, with the object represented by a tetrahedral mesh, Teschner et al. [120] have utilized the variation of the distances between mass points, surface areas of triangles and volumes of tetrahedrons to compute the potential energy. Their deformation model has been applied for simulation of the behavior of elasticity, plasticity and melting. A simple and effective dynamical simulation technique for point based objects has been proposed by Müller et al. [84], which guarantees unconditional stability. Shape matching of the original and current states of the object is utilized to estimate the goal positions of the object vertices and to guide the dynamical simulation during the deformation.

2.2.2 Mass Spring Systems

Mass spring system is another type of physically based deformation methods. It approximates a deformable object by a structure of discrete mass nodes linked by springs. Terzopoulos and Waters [117] used a three-layered mass spring system for facial animation. Based on this model, Lee et al. [72] proposed a framework that can adapt a generic face model to range scan data.

For a mass spring system, when the springs with very high stiffness are involved, the numeric solver for the system of ordinary differential equations has to take very small time steps to ensure stability, which largely increases the computational cost. Implicit integration method was proposed by Baraff and Witkin [12] for the mass spring system in the context of cloth simulation. It takes the end of a time step into account and enables integration with large time steps. A disadvantage of implicit integration method is that a linear system needs to be solved at each time step. The method described by Desbrun [36] et al. uses an approximation to the implicit integration method, and avoids solving the linear system. It corrects the integration by preserving the momentum of the whole mass spring system.

In spite of the efficiency of mass spring system, it is a very rough approximation to the real continuum solid. To achieve desired deformations, users have to choose a proper structure for the mass spring system and carefully tweak a big number of parameters like the elasticity coefficients, damping coefficients etc. Methods derived from elasticity theory using finite element method or boundary element method can give a more accurate description of the behavior of a deformable object.

2.2.3 Quasi-static Deformation Models

Quasi-static deformation models focus on the static deformations of objects and ignore the dynamic effects like jiggling and shaking. They do not involve solving ordinary differential equations and are suitable for interactive applications. Gourret [42] applied finite element method to simulate the deformations of both the hand and the object in a grasping task. In the setting of virtual surgery, Bro-Nielsen

and Cotin [19] described a real-time deformation method based on a linear elastic material model using finite element method. The stiffness matrix is condensed into a smaller matrix so that the computation only involves the nodes on the boundary of the object. The approach of Cotin et al. [29] augments the stiffness matrix by adding constraints with Lagrange multipliers method. It can achieve interactive speed due to pre-computation of tensors, which describe the relationship between displacements of free nodes and displacements of constrained nodes.

James and Pai [61] solved the linear elasticity deformation problem using boundary element method. Unlike finite element method that divides the interior space of an object into small 3D elements, boundary element method only requires that the boundary of the object is divided into 2D elements. The resultant linear system deduced from the Navier's equation is solved by a fast matrix update technique so as to maintain interactive deformation speed. Later, they incorporated this approach into a multiresolution framework [64] in order to handle interactive simulation of the deformations of objects with a large number of vertices. Based on the precomputed Green's function, James and Pai [63] also used the linear elasticity deformation model to deal with the deformations of the multi-zone system with a kinematic chain.

2.2.4 Dynamic Simulation Based on Continuum Mechanics

The dynamic deformation of an object is usually described by a system of ordinary differential equations. A numeric solver is required to evaluate the states of the system step by step. Roth et al. [101] extended the linear elasticity theory by adding incompressibility and nonlinear behavior in the context of facial surgery. Zhuang and Canny [130] presented a nonlinear deformation model, which is capable of dealing with global deformations. They used a graded FEM mesh structure with different sizes of elements so as to reduce the computational cost. Picinbono and his colleagues [95] modeled anisotropic deformations with large displacements for the simulation of laparoscopic liver surgery.

Cotin et al. [30] proposed a hybrid deformation model composed of a quasi-static model and a dynamic model for liver surgery simulation. Both deformation models are based on the same linear elasticity theory and have consistent behavior. The results created by the hybrid model exhibit dynamic characteristics with low computational budget.

The deformations of point based objects are investigated by Müller et al. [83]. The mesh-free representation of an object facilitates the simulation of a large range of material behavior including elasticity, plasticity, melting etc. A Moving Least Squares (MLS) procedure is devised to estimate the derivatives at a point based on its nearby points. With the derivatives available, quantities from continuum mechanics such as strain tensor, stress tensor and potential energy can all be deduced.

A nonlinear deformation model usually requires the computation of the stiffness matrix at every time step. It is computationally expensive and slows down the simulation. On the other hand, the stiffness matrix of a linear deformation model is constant and can be computed off-line, but it is only valid for small displacements. Müller et al. [82] wrapped the constant stiffness matrix, which is estimated based on the linear elasticity theory, with a rotational field. The deformation results are very close to the results of a nonlinear deformation model even for large displacements, while the computation is almost as efficient as a linear model. Later, Müller and Gross [83] improved the wrapping technique by estimating the rotational field based on elements, instead of nodes. Huang et al. [56] employed a domain decomposition method so that the precomputed force-displacement matrix based on the linear elasticity can be used to handle large displacements. They also utilized clustered principal component analysis to compress the force-displacement matrix for interactive speed.

Adaptive Dynamic Simulation

High-resolution mesh is required by severely deformed areas while low-resolution mesh is sufficient for flat areas. By reasonably allocating the computation resource, adaptive dynamic simulation techniques can produce detailed deformation results at a fast speed. Debunne et al. [32] approximated Lamé equation with discrete differential operators that can be applied to irregular mesh structures. The space of the object is adaptively sampled according to the rate of displacement variation. Furthermore, the states of different particles can evolve with different time steps. They proposed another adaptive simulation framework [33] with discrete differential operators obtained using finite volume method. Later they further improved their framework by using nonlinear finite element method with Green's strain tensor to calculate the elastic forces and used a non-nested multiresolution hierarchy of tetrahedral meshes to describe a deformable object [34].

Wu et al. [129] used the progressive mesh to represent the deformable object and dynamically refines the mesh using vertex splitting to simulate detailed deformations. Capell et al. [22] presented a dynamic framework that embeds the deformable object in a lattice. A subdivision scheme is applied to the control lattice, and a set of hierarchical basis functions are resultantly constructed. At runtime, basis functions are added or removed to increase or decrease the details according to deformation requirements. The adaptive framework of Hauth et al. [48] employs a sophisticated material law dependent on the history of strain rate tensor. Instead of refining elements, CHARMS proposed by Grinspun et al. [43] is based on the refinement of the basis functions.

Modal Analysis

Modal analysis can approximate the dynamic simulation at a fast speed. It was introduced into computer graphics by Pentland and Williams [94]. The stiffness and mass matrices are diagonalized by solving the generalized eigenvalue problem so that the system of ordinary differential equations can be written as separate

differential equations and each individual equation only involves one variable called *mode*. The states of each mode can be efficiently estimated analytically. The positions and velocity of all nodes of the object can be deduced from the states of modes through eigenvectors. Furthermore, the modes whose contribution to the deformation of the object is negligible can be discarded to further improve the speed. By evaluating the states of modes based on rigid body motions, James and Pai [62] mapped dynamic effects onto conventional deformations with negligible computational cost. Hauser et al. [47] demonstrated that collision and interactive manipulation with constraints can be incorporated into the modal framework. To apply modal analysis to simulate deformations with large displacements, Choi and Ko [27] separated the rotational component from the infinitesimal deformation and integrated it to compensate for the linear deformation. Their system also supports positional and rotational constraints for interactive user manipulation. An extension of modal analysis is the reduced coordinate deformable model described by Barbič and James [13]. With Green's strain tensor and the subspace bases, global deformations can be handled in real time.

2.3 Character Skin Deformation

The synthesis of skin deformations for articulated figures has its unique criteria. In the real world, skin deformations are caused by the movements of underlying musculature. Therefore, no matter what theory a skinning model is based on, it should make the skin deform as if there is musculature beneath it. A lot of efforts have been made to create aesthetically pleasing skin deformations. We review the skin deformation algorithms of the last two decades in five categories: surface skinning models, layered construction of skin deformations, anatomically based methods, example based methods and deformation transfer.

2.3.1 Surface Skinning Models

A surface skinning model is constituted only by the skeleton and the skin surface, and there is nothing between the skeleton and the surface. It usually uses some simple function of joints to determine the positions of skin vertices. The simplest skin deformation technique is rigid skinning [2]. The skin surface just follows the skeleton rigidly and no bulges are created on the skin. The surface may intersect with itself and discontinuities occur very commonly during the animation. Joint-dependent Local Deformation (JLD) operator introduced by Magnenat-Thalmann and her colleagues [75] calculates skin vertex positions according to the flexion of a joint, and inflation of the skin is also simulated. A realtime deformation technique for VR applications is described by Thalmann et al. [121] that represents the skin of the body by contours around the bones. During the animation, the contours rotate appropriately according to the joint angles to create skin deformations.

Skeleton Subspace Deformation (SSD) [73, 125] perhaps is the most widely used skin deformation algorithm for its simplicity and efficiency. It is contained in most 3D animation packages and accelerated by current graphics hardware. The positions of skin vertices are determined by weighted blending joint transformation matrices. It is very prevalent in many interactive applications, but it is also notorious for the painstaking process of weight painting and its characteristic "collapsing joint" defect. Mohr et al. [81] proposed a technique that allows users to directly manipulate the positions of skin vertices while the correct weights needed by SSD to reach the exact positions are automatically computed. To remove the "collapsing joint" artifact of SSD, Kavan and Žára [67] introduced Spherical Blend Skinning (SBS). This method interpolates the rotations of joints represented by quaternion instead of linear blending of transformation matrices in SSD.

2.3.2 Layered Construction of Skin Deformations

Layered skinning models utilizes an assistant layer between the skeleton and the skin to help improve the fidelity of the deformations. The middle layer is utilized

to deform the skin, while itself is controlled by the skeleton. Chadwick and his colleagues [24] pioneered layered deformation techniques by binding Free Form Deformation (FFD) lattices [104] with the skeleton. Kinematic deformations are implemented by establishing a relationship between control points of the mid planes of the lattices and joint angles, while dynamic effects are created by building mass spring systems in the FFD lattices. Capell et al. [21] extended a multiresolution dynamical simulation framework [22] to synthesize skin deformations for articulated characters. The character is embedded in a control lattice and the bones of the character skeleton coincide with some edges of the lattice. During the simulation, the skeleton serves as the constraint of the dynamic system.

A low-resolution triangular mesh is used by Singh and Kokkevis [109] as the middle layer. The skin vertices are registered to the local frames defined on the triangles of the low-resolution mesh and are deformed intuitively by them at run-time. Sun et al. [113] also utilized a low-resolution mesh to create deformations for highly detailed figures obtained through range scanning. They introduced novel point-to-line mapping and point-to-surface mapping to deform the low-resolution mesh and skin surface respectively.

Our Example Based Transferable Layered Enveloping (EBTLE) also utilizes a low-resolution mesh to facilitate the generation of natural skin deformations. This low-resolution mesh is automatically generated according to the decomposition of the skin. A new and simple technique is proposed to use the low-resolution mesh to deform the skin by maintaining the consistency of different patches. Another feature of EBTLE is that a set of example shapes of the low-resolution mesh are automatically generated and used to adjust its behavior, while the previous layered skin deformation techniques only use one static shape.

Bloomenthal [16] pointed out that the most decent tool to deform the skin is not the skeleton but the medial of the skin geometry, which is formed by the centers of all maximally sized spheres contained within the body of the character.

Implicit primitives like ellipsoids are another type of the middle layer for mod-

eling skin deformations. Thalmann and his coworkers [121] attached implicit primitives to the skeleton of the character. During the animation, these implicit primitives follow the skeleton and change their shapes accordingly. Hyun et al. [59] used a set of ellipsoids of varying sizes placed along the bones to simulate the bending of arms and legs. A base surface is derived from these ellipsoids and the skin surface is represented by the displacement map from this base surface. Later they extended this skinning model to handle the deformation of the torso [58]. The volume defined by the base (sweep) surface is preserved during the deformation. Self-penetration of the skin and protrusion of the bones are implemented based on GPU hardware.

2.3.3 Anatomically Based Methods

Anatomically based methods are a special type of layered skin deformation methods. To synthesize the shape of the skin, they simulate the fundamental cause of deformations. It makes sense to just model anatomical entities that contribute to the deformations of skin. Generally, the anatomy of a character can be divided into four layers: bones, muscles, fat and skin. By simulating the behavior of every layer and the interaction between adjacent layers, impressive results can be achieved.

Bones

Most of the time, bones are represented as abstract links between joints. If needed, polygon meshes rigidly following the skeleton can describe the bones in detail. The movements of bones are the starting point for the actions of other anatomical layers.

Muscles

Muscles are the major reason behind skin deformations. Contrary to the reality that muscle contraction results in the movement of the skeleton, in computer animation, alteration of the skeleton configuration triggers deformations of muscles. Muscle is contractile material with a sophisticated structure. There are two

types of muscle contraction. *Isotonic contraction* means that muscles change their lengths to produce force, while in *isometric contraction*, muscles contract with their lengths unchanged. Muscles are attached to the skeleton through tendons. Tendon is non-elastic connective material, which transmits the force generated by the attached muscle to the bones. The tendon attached at a higher position of the skeletal hierarchy is called origin and the other one insertion.

Chen and Zeltzer [26] built a muscle model based on the biomechanical theory. Finite element method with 20-node brick elements is used to compute the deformation of the muscle. The forces applied on the muscle are calculated according to Zajac's muscle model, which estimates the force generated by the muscle based on the length of its belly, the lengths of tendons, neural excitation level and contraction velocity. Lately, Teran et al. [114, 115] have proposed a method for biomechanically based muscle simulation using finite volume method. A nonlinear constitutive model for hyperelastic material is employed to describe the behavior of a muscle, which can undergo large deformations. This deformation method is extended by Irving et al. [60] to handle degenerate and inverted elements. Besides muscle simulation, it has also been applied to model the behavior of plasticity and to simulate thin shell objects.

In computer graphics, it is not necessary for the deformation of a muscle to exactly comply with the biomechanical theory. Instead of deriving the deformation of the muscle from the force acting on it, most approaches deform the muscle according to the changes in the positions of its tendons. Nedel and Thalmann [87, 88] used mass spring system to model the surface of a muscle. Angular springs are introduced to prevent distortion of the muscle surface. The mass points at the tendons of the muscle are attached to bones to provide constraint for the mass spring system.

Ng-Thow-Hing [90, 91, 89] argued that B-spline solids are a suitable representation of muscles for several kinds of applications. In [90], B-spline solids fitting and sculpting techniques are developed to facilitate the building of the entire anatomi-

cal sturcture with B-spline muscles. To animate muscles, sample points of B-spline solids are used to set up mass-spring systems. Anchored fields are also designed to specify deformations. In [91], Ng-Thow-Hing and Fiume showed that B-spline solids not only can capture shapes of muscles but also can describe internal fiber structures of muscles.

Some researchers believe that geometric methods are the optimum way of simulating muscles. Since muscles lie beneath the skin, accurate muscle contraction simulated using physically based methods may be unnecessary. Wilhelms [127] modeled muscles with ellipsoids. An ellipsoid muscle is attached to two bones and the volume of the ellipsoid is kept constant during the animation. The ellipsoid muscle bulges automatically when the bones pull towards each other. Wilhelms and Gelder [128] extended this work by modeling muscles with more generalized geometric primitives: deformed cylinders. The default muscle can be easily modified to describe complicated muscle shapes. The width and thickness of the muscle slices are scaled accordingly to maintain the approximate muscle volume at runtime.

Scheepers and his colleagues [103] presented appropriate models for different types of muscles. A fusiform muscle is represented by an ellipsoid with two tendons attached to the muscle belly, and the volume of the muscle belly is preserved during the deformation. Isometric contraction is simulated by introducing a tension parameter that can adjust the ratio between the width and height of the muscle belly. For multi-belly muscles, two curves are utilized to organize a set of ellipsoidal bellies. Furthermore, a general muscle model is devised to describe more complex shapes. It is formed by sweeping a varying ellipse along a cubic Bezier curve and its deformation is also based on volume preservation.

Dong and his coworkers [38] modeled a muscle by fitting its surface to segmented anatomical data and represented it with a set of contours. The deformation of the muscle is dependent on the change of its action line and the volume of the muscle is conserved.

Aubel and Thalmann [7, 8, 6] introduced a two-layer muscle model. The action

line of a muscle is approximated by linear segments, from which a 1D mass-spring system is constructed. The mass nodes representing the origins and insertions are attached to skeleton bones. The vertices of the muscle surface are mapped to the local frames associated with the mass nodes. The animation of mass nodes results in the deformation of the muscle surface.

In order to quickly determine whether a skin vertex is penetrated with muscles, the framework proposed by Turner et al. [124, 123] uses implicit surfaces like superquadrics to model muscles. These implicit surfaces serve as links between two joints. The deformations of the muscles are accomplished by adjusting the parameters of these implicit surfaces. When a skin vertex is detected in the muscles, reaction constrain force is exerted on the vertex to expel it out.

Hirota et al. [52, 50, 51] simulated deformations caused by collision in the context of anatomically based modeling. The deformations of anatomical components are computed using nonlinear finite element method. A novel concept of material depth is proposed for estimation of contact forces. It is robust and capable of handling both self-penetration and penetration between different objects.

Our Skinning With Deformable Chunks (SWDC) makes use of deformable chunks as the substitute for groups of muscles. The generation of these chunks only requires users to decompose the skin of the character into patches, which are more efficient and intuitive than to specify every individual muscle. In the domain of facial animation, Kähler et al. [65] proposed an approach similar in flavor with ours, which automatically generates muscles according to basic grids sketched by users. Another recent development with the similar spirit is the method of Pratscher et al. [97], which can automatically fit a pair of ellipsoidal muscles to a certain region of a character body based on the segmentation of the skin surface.

Fat

Fat is passive, viscoelastic material lying between muscles and the skin. Few anatomically based methods describe it explicitly. Aubel and Thalmann [7, 8, 6]

treated the fat layer as a continuous solid and used finite element method to calculate its deformation. The innermost nodes of the fat layer are attached to underlying muscles, while the outermost nodes are used to connect the skin surface. Deformations of muscles results in the deformation of the fat layer, which in turn causes the deformation of the skin. In the framework of Turner et al. [124, 123], fat is simulated by a mechanism that keeps the skin a certain distance away from muscles.

Skin

Skin is an elastic membrane that covers all internal anatomical entities and displays the final deformation effects. Wilhelms and Gelder [128] employed a relaxation procedure to make the skin vertices smoothly distributed across muscles. A virtual anchor is defined for every skin vertex as its position with respect to the underlying anatomical entities. The anchoring methods for both ellipsoid muscles [127] and deformed-cylinder muscles [128] are devised. Then every skin vertex is linked to all its direct neighbors and its virtual anchor via a nonlinear spring. Aubel and Thalmann [7, 8, 6] attached the skin tightly to the fat layer. Since the fat layer is simulated using a physically based method, the skin resultantly exhibits natural deformations. Turner et al. [124, 123] utilized the dynamical deformation method proposed by Terzopoulos and his colleagues [118] to simulate the skin. The positions of skin vertices are estimated according to the forces applied on them. Besides the elastic forces derived from the skin itself, the fat layer exerts forces to push the skin vertices away from muscles, while the connective tissues pull them back towards muscles. Constrained forces are also applied to prevent the skin vertices from penetrating muscles. The elastic skin can slide over the underlying anatomical entities during the animation. Kähler et al. [65] make use of a double-layered mirroring mass spring system to control the deformations of the skin for facial animation.

2.3.4 Example Based Methods

Using a set of example character shapes in different postures, example based methods learn to properly configure their computational models so that the skin deforms in the same way as exhibited by the examples. Interpolation or fitting techniques are commonly employed in these models. Pose Space Deformation (PSD) proposed by Lewis and his coworkers [73] is the first example based skinning model for articulated characters in the literature. It successfully combines shape interpolation and skeleton subspace deformation so that animators can use example character shapes to specify skin deformations. With proper examples, this method can easily overcome the defect of SSD. Instead of interpolation for every vertex of the skin surface as in PSD [73], Shape By Example (SBE) proposed by Sloan et al. [110] handles the interpolation problem in the level of example shapes. Kry et al. [69] introduced EigenSkin using examples obtained from the dynamic simulation of a finite element model. It converts all vertex displacements in a certain area into eigendisplacements with much smaller dimension through principal component analysis so that the interpolation task is greatly alleviated.

Allen and his colleagues [3] used range scan devices to capture example shapes from real subjects. Displaced subdivision surface [71] is employed as a template to fit the example shapes. At runtime, difference displacement maps derived from different examples are blended to create detailed skin deformations.

Wang and Phillips [125] deduced a more expressive skinning model called Multi-Weight Enveloping (MWE) from traditional SSD by assigning every entry of blending matrices a different weight value. The weights are automatically calculated based on the examples by solving a linear system. Another fitting solution is Mohr and Gleicher's Extended Linear Blend Skinning (ELBS) [80]. It is also derived from SSD, extra joints are added instead of multiple weights to eliminate the deficiency of SSD and forge muscle bulging effects. An iterative optimization procedure is utilized to find the right weight values and the most suitable dressing shape.

Grzeszczuk et al. [44] described an efficient computation model, namely NeuroAnimator, for motion control. Using the example motion generated by physically based models, neural networks can be trained to emulate the dynamics of a complex system.

Our EBTLE is a new example based skin deformation technique. It has the following characteristics:

- Like all example based skin deformation techniques, with EBTLE, control of deformations to generate a specific appearance is made easy. Artists can sculpt the desired shape by whatever means and include it to the set of example shapes.

- Similar to EigenSkin [69], EBTLE utilizes patches as the units of deformations instead of vertices [73, 125, 80], or entire skin [110]. Animators can improve the behavior of a particular patch without affecting other patches. Moreover, users can freely determine which skeletal parameters affect a patch.

- Unlike previous interpolation models [73, 110, 69] that limit the interpolation to skeleton pose space, EBTLE applies the interpolation in the hyper space of skeletal parameters augmented with skin vertex parameters, which makes it possible to transfer skin deformations between different characters.

- To synthesize deformations on the skin of a character, all previous example based techniques have to be trained using a set of examples of the character. Given a new character, these methods require artists to provide a completely new set of examples of this new character, even though it is quite similar to the formerly trained one. In contrast, EBTLE provides a uniform representation. Once our skinning model is properly configured using the examples of one character, it can then be applied for the skin deformations of other characters with similar appearances.

- Most of the previous example based techniques require quite a few examples

to achieve satisfactory results. For EBTLE, the decomposition of the skin largely reduces the complexity of the problem, and therefore the number of examples required to create pleasing deformations is quite small. Although EigenSkin [69] also handles deformation on a patch basis, it employs a fitting computational model, which demands a lot of examples.

2.3.5 Deformation Transfer

Reusing existing animation data to create skin deformations for new characters has recently attracted attention in the community of computer graphics. It will save a lot of efforts if the skin deformations sculpted for one character can be transferred to another character while the features of the individual character shape can still be kept. One of the first deformation transfer methods was proposed by Noh and Neumann [92] in the domain of facial animation. After dense correspondence is established, the motion vectors of the vertices of the target face can be derived from those of the source face using barycentric coordinates. The magnitudes and orientations of the motion vectors are adjusted accordingly before they are added to the neural target face to generate deformations. The facial expression transfer method proposed by Na and Jung [85] represents both the source and target faces by normal meshes [45]. For every layer of the hierarchical normal meshes, normal offsets are obtained from the deformed shape of the source face and added to the target face after appropriate scaling. Wang et al. [126] treated the facial animation as composition of three components: face geometry, facial expression content and expression style. The animation of the target face is accomplished by transferring the expression content from the source face and combining the target face geometry and expression style.

For articulated characters, Sumner and Popović [112] recently presented an elegant paradigm to derive the deformation of a target mesh from the deformation of a source mesh. The transformations applied on the triangles of the target character are required to be as similar as possible with the transformations of the corre-

sponding triangles of the source character and meanwhile the consistency of vertex positions for the deformed skin shape of the target character has to be maintained.

The deformation transfer algorithms mentioned above do not transfer the control mechanisms that create deformations. The deformation of the target is totally dependent on the deformation of the source. To deform the shape of the target for a certain expression or posture, the deformed shape of the source for that expression or posture has to be generated in the first place. The facial expression clone approach of Pyun et al. [99] requires coupling examples of both source and target faces. After the parameterization of expressions is derived from the examples of the source face, at runtime, any expression of the target face can be obtained by blending its own examples. To improve the versatility of their anatomically based deformation technique [128], Simmons et al. [107] showed a way to synthesize skin deformations for a new horse by morphing the canonical horse into the new horse. Since all anatomical entities of the canonical horse are adjusted to conform with the specifications of the new horse, it is immediately animatable once the morphing process is finished. Kähler et al. [66] used a reference head with all anatomical entities to fit different range scanned head models. Based on selected land marks, the skin, muscles and skull of the reference head are all adapted for the target head so that expressions can be synthesized on the target head using the same muscle contraction parameters. Seo et al. [106, 105] described an approach to fitting a template human body model bound with the skeleton via skeleton subspace deformation to scanned data. Furthermore, this framework enables users to create new human body models by simply providing a few anthropometric parameters. However since the template model is set up using the simple SSD, the skin deformations of the newly fitted human bodies inevitably suffer from the shortcomings of SSD. Capell et al. [20] introduced force to control skin deformations relying on physically based dynamical simulation. Since force is decoupled from the skin geometry of the character, it can be easily transferred to generate the skin deformations of another character.

Our EBTLE also allows the control mechanism for skin deformations to be transferred from one character to other characters. All users need to do is to provide a set of examples of one character, then the skin deformations of other characters can be derived from this very set of examples. With EBTLE, since the knowledge of deformations are stored in the RBF networks, to generate the deformations for a target character does not need the deformations of the source character to be generated in advance.

Using range scan data of different human bodies in different postures, Anguelov et al. [5] presented a very promising framework, called SCAPE, which can capture the skin deformations caused by the variation of both posture and individual character shape. After the learning is complete, it can create high quality skin deformations for a new character based on a single static scan of the character and the motion data of the marks on the character. Compared with our EBTLE, SCAPE has a more flexible representation of skin deformations. However, this also means that SCAPE demands more examples to learn from. As mentioned before, the number of examples required by our EBTLE is small.

2.4 Summary

In this chapter, we have given a survey of deformation methods. Both geometric and physically based methods have their own advantages and disadvantages, and neither of them are replaceable. Geometric methods have unsurpassed speed and are adequate for interactive applications. But they rely on the skills of artists to create natural deformations. Physically based methods can automatically achieve realistic results. However, the heavy computational budget of the dynamical simulation is always an issue. And since the simulation is based on physical laws, users can not directly sculpt desired deformations as offered by geometric methods.

Simulation of deformations on the skin of an articulated figure requires more advanced algorithms. Both geometric and physically based methods can be utilized

to create skin deformations. But the requirements of high quality animation and real-time speed usually place the design of a skin deformation method in a dilemma. Surface skinning models commonly use efficient geometric methods. They can achieve real-time speed and are suitable in realtime environments like video games. But the quality of the deformation results may not be able to fulfill the requirements of some applications like movies. The fidelity of results can be improved by layered construction of skin deformations. Nevertheless, the quality of the deformation still depends on artists to design and place the middle layer. Anatomically based methods can generate impressive skin deformations automatically, as soon as the whole anatomical structure of the character is built. However, the construction of the whole anatomical structure is a daunting job and animators can only indirectly modify the consequences by changing the configuration of the anatomical structure. The most obvious advantage of example based methods over other categories is that they enable users to directly sculpt the skin deformations as they wish. But the number of examples required to obtained pleasing results grows dramatically with the dimension of the interpolation space, and it may be overwhelming for artists. Deformation transfer algorithms allow artists to reuse the skin deformations of one character to deform the skins of other characters and therefore the burden of artists can be tremendously alleviated.

As we can see, different kinds of skin deformation methods have their own strength and shortcoming. The ideal skin deformation model should combine their advantages and remove their disadvantages. The fundamental task for a good skin deformation method is to make the skin of a character deform realistically when the skeleton changes its posture. The skin needs to remain continuous and to have proper muscular features like bulge. It is intended that a skinning model should use automatic processes to help artists set up a character. However it also should leave artists some space so that artists can adjust the deformation results as they wish. It is very important for a skin deformation method that artists can modify deformation results freely and intuitively. So the balance needs to be kept between

automatic process and manual intervention. To make a skinning model suitable for realtime applications, the computational cost at runtime should be low and stable. Finally, the ability of reusing deformations would definitely increase the popularity of a skinning model.

Chapter 3

Example Based Transferable Layered Enveloping

As discussed in the literature review, layered skinning models are more flexible than surface skinning models, however their ability of representation is still limited, and most of them cannot describe muscular features such as the bulge formed on the upper arm. Example based methods utilize powerful computational models to describe skin deformations. But, usually quite a few examples are required to properly configure these models. This is because that the representation for the deformation of the entire body is very complex, and it needs a large number of examples to learn from. If the representation is only for the deformation on a small area of the skin instead of the entire body, then it will be much simpler and easier for the computational model to obtain a good configuration. The number of examples required for this local area will be small. It would be perfect to combine layered skinning model with example based method, because layered skinning model can handle general deformation of the entire body, while example based

method can deal with fine deformations on certain areas of the skin. The computational models used by previous example based methods are usually based on skeletal parameters. To further increase the flexibility of a computational model, these skeletal parameters can be augmented with skin vertex parameters so that the computational model is built in a higher dimensional space. Once the skins of two characters are parameterized in the same way, the computational model configured for one character can be used to create deformation for the other character. With all the above ideas blended together, we proposed a new framework, namely Example Based Transferable Layered Enveloping (EBTLE).

EBTLE is an efficient skinning model that inherits the advantages of layered skinning models, example based methods and deformation transfer algorithms. Considering the locality of the deformations exhibited on the skin, EBTLE offers a way to synthesize skin deformations patch by patch. To set up a character with EBTLE, users first need to decompose its skin into patches. A low-resolution mesh, namely *Base Control Mesh* (BCM), is generated according to the structure of the skin patches and utilized as the middle layer between the skeleton and the skin mesh. At runtime, the skin shape is manipulated by the BCM, and in turn the shape of the BCM is controlled by the skeleton. The control mechanisms for the deformations of both the skin and the BCM are based on a set of examples of the character so that the final synthesized skin deformations have identical characteristics to the example shapes. Another unique feature of EBTLE is that the very same set of examples can be used for synthesizing skin deformations of other geometrically similar characters. EBTLE formulates the principle of skin deformations in a higher level and decouples it from the variation of individual characters so that it can be applied to other characters without any difficulties.

The character that provides examples is called *primary character*, and the other characters who do not have their own sets of examples and whose skin deformations depend on the examples of the primary character, are called *target characters*. The decomposition of the skins of the primary and target characters must result

in identical patch structure. The corresponding patches on the primary and target characters are parameterized in the same domain, based on which a universal deformation representation is developed using *Radial Base Function* (RBF) networks. The base control mesh is responsible for the global deformations of a character. For an individual patch, a RBF network can be constructed to add local fine deformation on it.

To explain EBTLE better, some concepts must be clarified first. Examples of the primary character are a variety of static shapes paired with skeleton postures. The connectivities of example shapes are required to be the same, i.e. every vertex on one example shape has a corresponding vertex on every other example shapes. The example shapes and their paired skeleton postures can be sculpted manually by artists using 3D modeling software. Range scanning devices powered by advanced reconstruction techniques can capture static shapes from a real world subject. But the obtained shapes usually need to be remeshed so that they have the identical connectivity, and the corresponding skeleton postures have to be estimated through certain landmarks on the subject. One of the example postures, usually a neutral posture with all the rotational Degrees Of Freedom (DOFs) as zeroes, is selected as the criterion and called the *dress posture*. The example shape corresponding with the dress posture is called the *dress shape*. We use n_e to denote the total number of available examples of the primary character. The examples are ordered and indexed by $1, 2, \ldots, n_e$, and the dress shape in the dress posture is assumed to be the first example. For a target character, only its static shape in the dress posture is demanded by our system. The skin mesh of the target character does not need to have the same number of vertices or the same connectivity as the primary character, and the skeleton of the target character is not required to be identical to the skeleton of the primary character. However the skeletons of the primary and target characters are supposed to share the same hierarchical structure, and they have the same number of joints and thus the same number of rotational DOFs. Two characters are deemed to be in the same posture if and only if all the corresponding

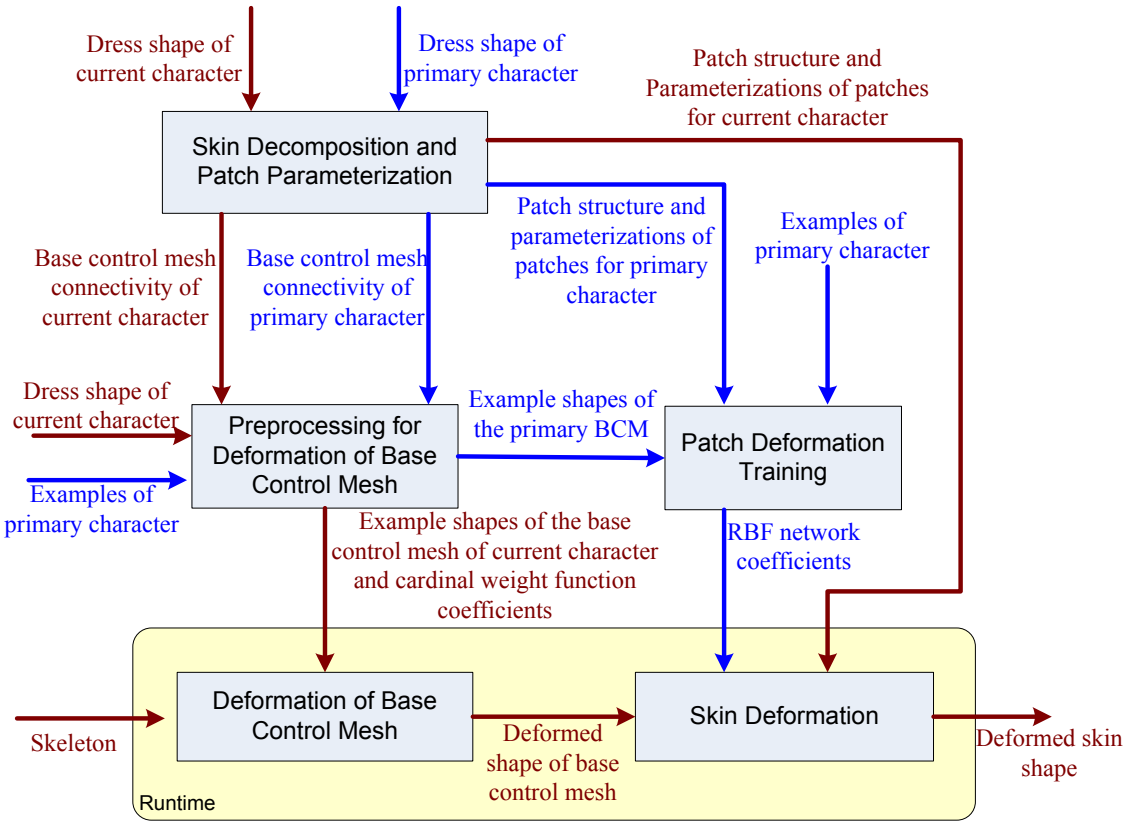

Figure 3.1: An overview of our example based transferable layered skinning model. Blue: The workflow of configuring the system using examples of the primary character. Red: The workflow of using EBTLE to create deformations for the primary character or target characters. The configuration of the system (blue) needs to be finished before the runtime synthesis (red) can be used. For runtime synthesis of character shapes, only the modules in the yellow area are involved. The other modules are used in the setup process.

rotational DOFs of their skeletons are identical.

The overview of EBTLE is given in Figure 3.1 and its pictorial version is shown in Figure 3.2. Section 3.1 describes the skin decomposition technique used in our system. Section 3.2 explains how to build a parameterization for every patch. In Section 3.3, radial basis function interpolation is explained. The method to manipulate the base control mesh using the skeleton is presented in Section 3.4, and the approach to deforming the skin using the base control mesh, which solely depends on the dress shape of the character, is introduced in Section 3.5. We will explain how to use examples of the primary character to adjust the deformation results generated by the pure layered skinning model for both the primary and

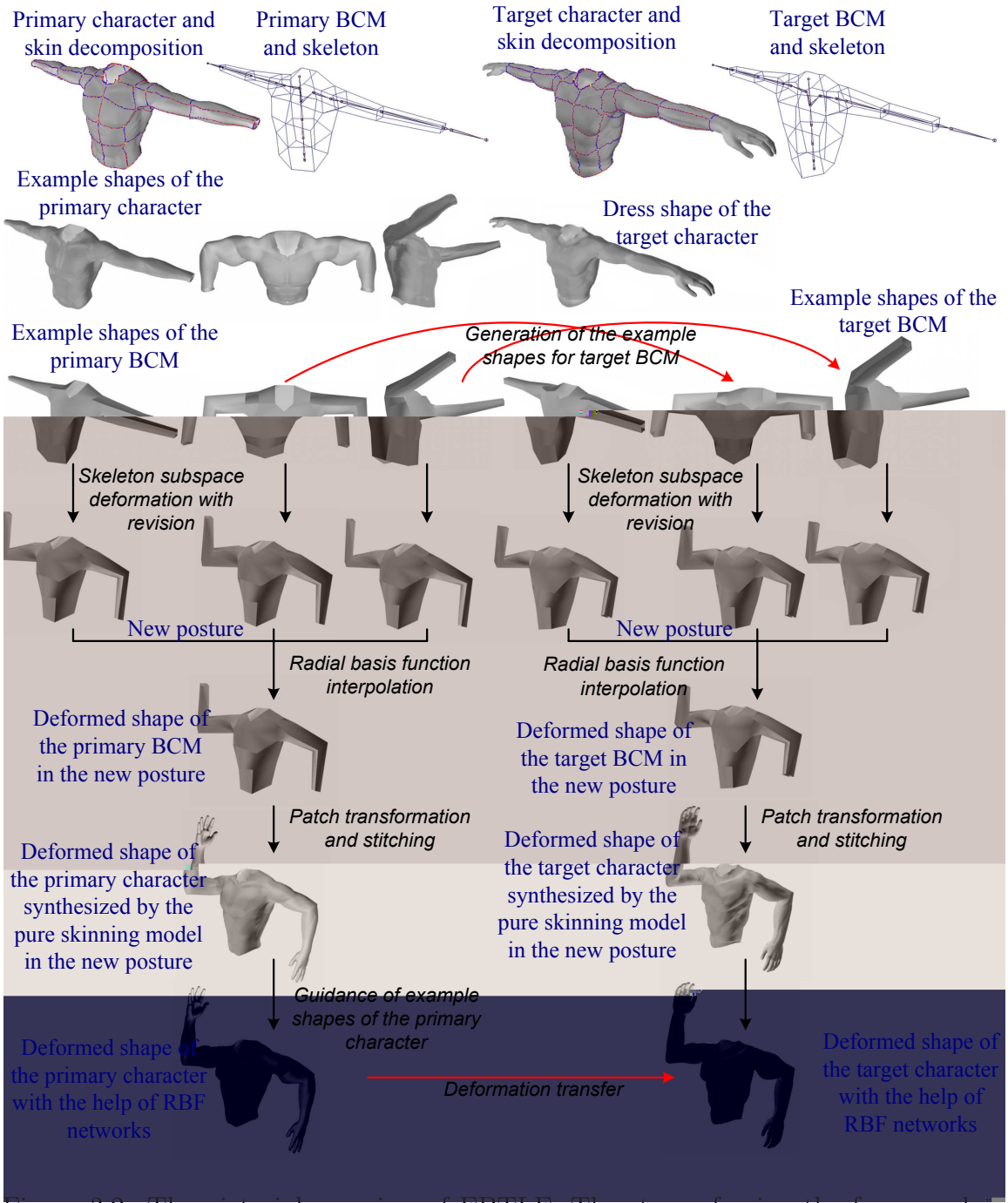

Figure 3.2: The pictorial overview of EBTLE. The steps of using the framework are illustrated from the top to the bottom.

target characters in Section 3.6. Experimental results of EBTLE are presented in Section 3.7.

3.1 Decomposition of Skin Mesh

Decomposing a single surface mesh into several smaller patches has proven to be useful in many research areas of computer graphics [98]. It is observed that most muscular characteristics of deformation like bulge are localized and only involves a limited area of the skin. Thus "divide and conquer" strategy can be applied for skin deformations of articulated figures. The skin of a character is decomposed into patches, and then the deformation of each patch is handled individually. Automatic decomposition techniques can reduce the burden of artists, but they give little freedom for artists to express their creativity and usually can not capture the semantic meanings of patches. Skin deformations are the results of movements of underlying musculature, so it helps to segment the skin mesh based on simple anatomy knowledge. A manual decomposition technique is used in our framework. To identify a patch on the skin mesh, its boundaries have to be specified first.

As illustrated in Figure 3.3, our system enables users to interactively specify a boundary on the skin mesh by manipulating a space NURBS curve. The boundary is the projection of the curve on the skin, which is a polyline consisting of vertices/edge-points of the skin geometry. It may cut some triangles of the mesh. When the curve is close to the skin of the character, its projection on the skin is pretty intuitive and easy to control.

The best way of constructing this kind of NURBS curve is to specify its control points along the desired path on the skin. In this way, the constructed NURBS curve is quite near to the skin, and its projection on the skin should follow the curve correspondingly. To adjust the boundary, users simple change the positions of the control points of the curve. As the curve changes its shape, its projection on the skin, the boundary, will also change accordingly. Another advantage of this

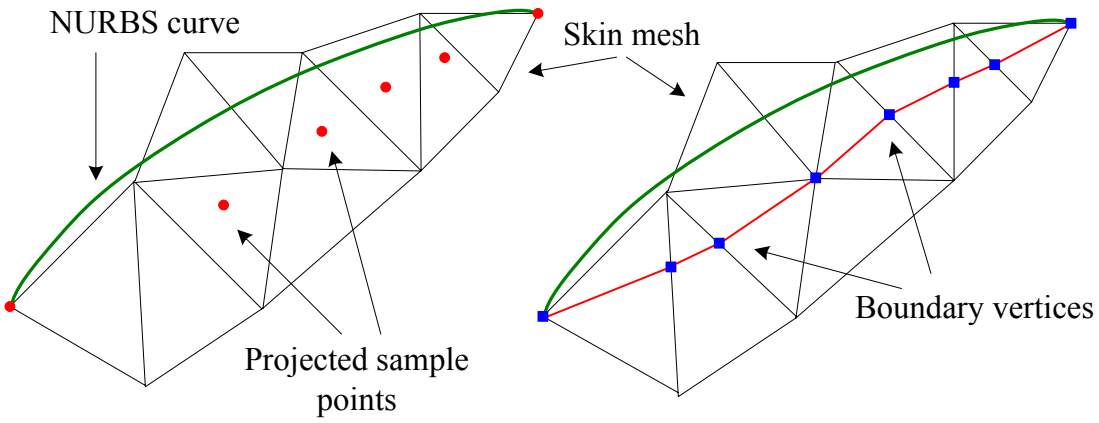

Figure 3.3: Specifying a boundary by projecting a NURBS curve on the skin mesh. Left: A NURBS curve near the skin mesh and the projected sample points on the mesh. Right: Based on the sample points, a boundary is generated using the vertices and edge points of the skin mesh.

projection technique is that users do not need to concern about the structure of the boundary, because the algorithm will automatically choose vertices or edge points on the skin mesh, and create a proper polyline as the boundary.

The manipulation of the NURBS curve can be done using any standard technique, and the only requirement is to keep the curve close to the skin so that the projection is intuitive. The most straightforward way of manipulating a NURBS curve is to change the positions of its control points, and the shape of the curve will follow its control points accordingly. Another way is to directly specify the positions of certain points of the curve and the positions of the control points can be automatically calculated so that the NURBS curve fit the specified curve points. In EBTLE, the purpose is to allow users to intuitively specify the boundary, and it is enough to manipulate the curve simply by adjusting its controls points. Usually a relatively smooth boundary is desired, so most of the time we can simply use the NURBS curve that only has four or five control points, therefore it is quite easy for users to manipulate.

To project a curve onto the skin mesh, the curve needs to be evenly sampled into discrete points, and these sample points are projected on the mesh by locating their nearest points on the mesh (see Figure 3.3). The number of sample points

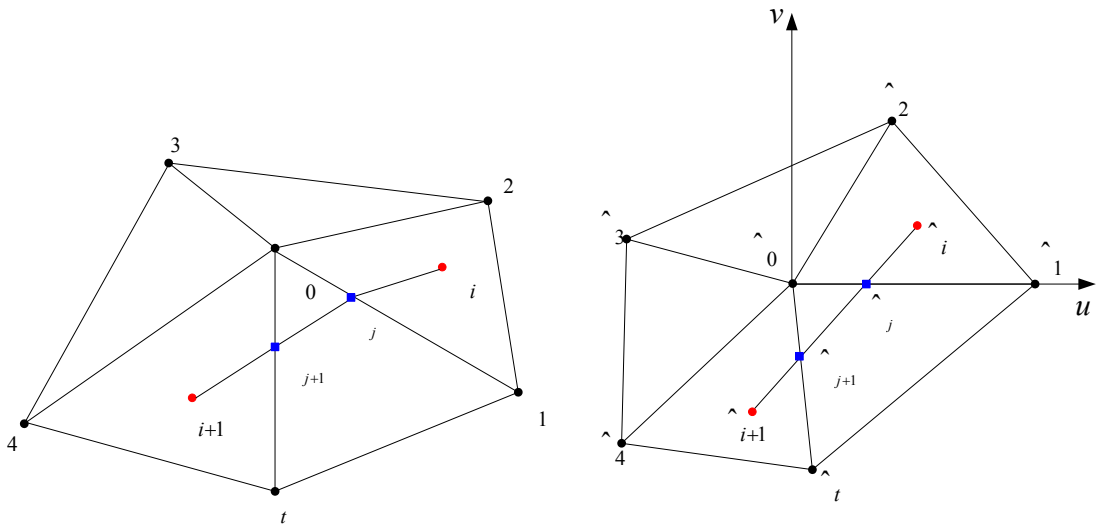

Figure 3.4: Finding the boundary vertices on the skin mesh. Left: One ring neighborhood of a vertex in the skin mesh and two successive projected sample points. Right: The parameterization of the one ring neighborhood of the vertex.

should be large enough to guarantee that, for any two successive projected sample points, there exists a vertex of the skin mesh whose one ring neighborhood contains them.

Let $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ be the two successive projected sample points that are not in the same triangle, and $\mathbf{p}_0$ be the vertex whose one ring neighborhood contains $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ (see Figure 3.4). To find out the boundary vertices between $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, we build a parameterization in the vicinity of $\mathbf{p}_0$. As illustrated in Figure 3.4, $\mathbf{p}_0$ and its direct neighbors $\mathbf{p}_1$, $\mathbf{p}_2$, ..., $\mathbf{p}_t$ are mapped to a plane using the following equation:

$$\hat{\mathbf{p}}_i = \begin{cases} (0,0) & (i=0) \\ \|\overrightarrow{\mathbf{p}_0\mathbf{p}_i}\|\,(\cos\chi_i, \sin\chi_i) & (1 \le i \le t) \end{cases}$$

where $\hat{\mathbf{p}}_0$, $\hat{\mathbf{p}}_1$, ..., $\hat{\mathbf{p}}_t$ are their corresponding points on the plane, $\chi_i = 2\pi \left(\sum_{j=1}^{i} \theta_j \Big/ \sum_{j=1}^{t} \theta_j \right)$ and θ_j is the angle between $\overrightarrow{\mathbf{p}_0\mathbf{p}_j}$ and $\overrightarrow{\mathbf{p}_0\mathbf{p}_{j+1}}$. Then $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ are mapped to $\hat{\mathbf{x}}_i$, $\hat{\mathbf{x}}_{i+1}$ on the plane via their barycentric coordinates. Suppose $\mathbf{x}_i$ lies in triangle $\{\mathbf{p}_0, \mathbf{p}_{i_0}, \mathbf{p}_{i_1}\}$, and its barycentric coordinates is (α, β, γ). The 2D parameter of $\mathbf{x}_i$ is:

$$\hat{\mathbf{x}}_i = \alpha\hat{\mathbf{p}}_0 + \beta\hat{\mathbf{p}}_{i_0} + \gamma\hat{\mathbf{p}}_{i_1}.$$

In the parameter domain, intersection points between $\{\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{i+1}\}$ and $\{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_j\}(1 \le$

$j \leq t$) can be easily detected (see Figure 3.4). Every intersection point corresponds to a boundary vertex on the skin mesh. Let $\hat{\mathbf{b}}_j$ be the intersection point between $\{\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{i+1}\}$ and $\{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_j\}$, and $r = \frac{|\hat{\mathbf{p}}_0\hat{\mathbf{b}}_j|}{|\hat{\mathbf{p}}_0\hat{\mathbf{p}}_j|}$. The corresponding boundary vertex of $\hat{\mathbf{b}}_j$ on the edge $\{\mathbf{p}_0, \mathbf{p}_j\}$ is $\mathbf{b}_j = (1-r)\mathbf{p}_0 + r\mathbf{p}_j$. If r is very close to 0 or 1, $\mathbf{b}_j$ will be snapped to $\mathbf{p}_0$ or $\mathbf{p}_j$. An example can be seen on the right of Figure 3.3, where the fourth boundary vertex (from the left) is snapped to the mesh vertex. We use this snapping rule to prevent very small triangles being introduced by the boundaries. After boundary vertices between every pair of $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ are found, the boundary is obtained by connecting all the boundary vertices sequentially. Note that if $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ are in the same triangle, no boundary vertices will be found. For the sake of convenience, a boundary can also be specified by the shortest path that complies with the connectivity graph of the skin mesh between two user-specified vertices.

With the help of boundaries, the skin mesh of a character can be dissected by users as they wish. The decompositions of two upper human body models are displayed in Figure 3.5. The point where more than one boundaries meet is called *feature vertex*. A patch is defined by several cyclically connected boundaries. The skin vertices that are enclosed by these boundaries belong to the patch. A patch can have several feature vertices. The term "feature vertex of a patch" is referred to the feature vertex that belongs to a patch. The connectivity and shape of a patch are extracted from the skin geometry of the character. Along the boundaries of the patch, some faces may need to be triangulated because a boundary specified by the curve projection technique can cut the triangles of the original skin mesh and introduce some non-triangular polygons. However, all patches are stored with internal data structures and the connectivity of the skin mesh remains undisturbed. The positions of all the vertices of a patch are recorded using relative coordinates with respect to the vertices of the original skin mesh. Hence, for the primary character, once the decomposition is finished on its dress shape, we actually can get the shapes of the same patch on all the example shapes.

To ensure that skin deformations can be transferred from the primary character

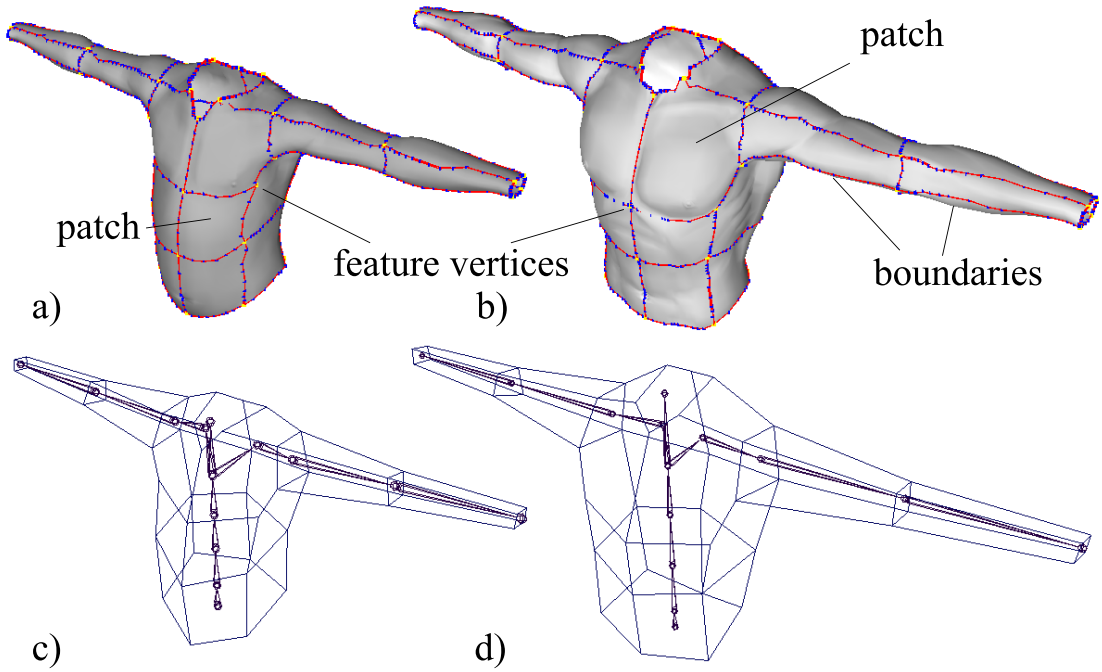

Figure 3.5: a) Skin decomposition of a target character. b) Skin decomposition of the primary character. c) The base control mesh and the skeleton of the target character. d) The base control mesh and the skeleton of the primary character.

to a target character, their dress shapes have to be partitioned in such a way that the graphs representing the decompositions of their skin meshes have the same topology (see Figure 3.5). Two perspective viewers are used to display the primary and target characters respectively using the same camera. Users need to specify boundaries on the skin of the target character with the patch structure of the primary character as the criterion. The corresponding patches of primary and target characters are assigned the same index to set up correspondence.

3.2 Patch Parameterization

Decomposing the skin meshes of the primary and target characters in the same way merely establishes a loose relationship between them. Building parameterizations for the corresponding patches of the primary and target characters in the same domain makes it possible to represent their skin deformations universally. Harmonic mapping [39, 1] is employed to compute the parameterizations and it can effectively project a 3D disk-like triangular mesh to a 2D planar graph.

For the sake of clarity, we re-index the vertices in a patch. The indices start with 0, and end with $(n_p - 1)$, where n_p is the total vertex number of this patch. Furthermore, we assume the indices of vertices contained by boundaries of this patch are in $[n_p - n_b, n_p - 1]$, where n_b is the total number of vertices in all boundaries. Without further specification, throughout this chapter, the index of a vertex refers to this in-patch index and vertex $\mathbf{p}_i$ means the i-th vertex in the patch.

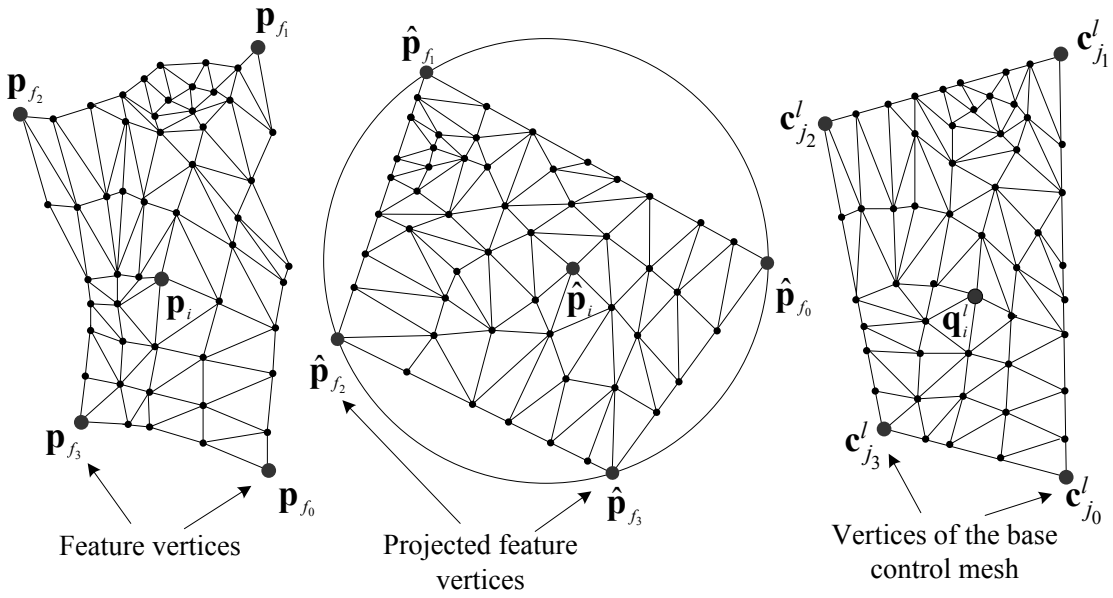

Figure 3.6: Left: A patch on the dress shape of a character. Middle: The parameterization of the patch. Right: Register points on the corresponding face of the base control mesh.

As shown in Figure 3.6, the algorithm first puts the feature vertices of a patch on a unit circle so that the ratios of the arc lengths between two projected feature vertices are the same with the ratios of the boundary lengths between the two feature vertices on the patch shape. Connecting the projected feature vertices consecutively yields a planar convex polygon. Then the vertices along a boundary are placed proportionally on the corresponding edge of the polygon based on the ratios of edge lengths. The internal vertices of the patch are embedded in the polygon by minimizing distortion energy ε_h:

$$\varepsilon_h = \frac{1}{2} \sum_{\{i,j\} \in \mathcal{G}} \lambda_{ij} \left((u_i - u_j)^2 + (v_i - v_j)^2 \right), \tag{3.1}$$

where $\mathcal{G}$ is the set of edges of the patch, (u_i, v_i) and (u_j, v_j) are the parameters of vertex $\mathbf{p}_i$ and $\mathbf{p}_j$, and λ_{ij} is the scalar coefficient, which depends on the properties of the patch in the dress shape of the character. Let $\{\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_{k_0}\}$ and $\{\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_{k_1}\}$ be the two triangles incident to edge $\{\mathbf{p}_i, \mathbf{p}_j\}$, and a_0, a_1 be their areas respectively. The scalar λ_{ij} is determined as follows:

$$\lambda_{ij} = \begin{cases} \frac{l_{ik_0}^2 + l_{jk_0}^2 - l_{ij}^2}{a_0} + \frac{l_{ik_1}^2 + l_{jk_1}^2 - l_{ij}^2}{a_1} & (\{i,j\} \in \mathcal{G}) \\ 0 & (\{i,j\} \notin \mathcal{G}) \end{cases}, \tag{3.2}$$

where l_{ij} is the length of edge $\{\mathbf{p}_i, \mathbf{p}_j\}$ evaluated on the dress shape of the character. The parameters of the internal vertices are chosen to make ε_h as small as possible. Setting the gradient of ε_h to zero gives:

$$\mathbf{F}(\ \mathbf{u}_r \quad \mathbf{v}_r\ ) = \mathbf{H}(\ \mathbf{u}_b \quad \mathbf{v}_b\ ), \tag{3.3}$$

where $\mathbf{F}$ is a $(n_p - n_b) \times (n_p - n_b)$ matrix with elements $f_{ij} = \begin{cases} -\lambda_{ij} & i \neq j \\ 1 & i = j \end{cases}$, matrix $\mathbf{H}$ has $(n_p - n_b)$ rows and n_b columns and its element h_{ij} equals $\lambda_{i(n_p - n_b + j)}$, $\mathbf{u}_r = \left(u_0, u_1, \ldots, u_{(n_p - n_b - 1)}\right)^T$, $\mathbf{v}_r = \left(v_0, v_1, \ldots, v_{(n_p - n_b - 1)}\right)^T$, $\mathbf{u}_b = \left(u_{(n_p - n_b)}, u_{(n_p - n_b + 1)}, \ldots, u_{(n_p - 1)}\right)^T$, $\mathbf{v}_b = \left(v_{(n_p - n_b)}, v_{(n_p - n_b + 1)}, \ldots, v_{(n_p - 1)}\right)^T$. Due to the non-singularity [1] of matrix $\mathbf{F}$, given $\mathbf{u}_b$ and $\mathbf{v}_b$, Equation 3.3 has a unique solution:

$$(\ \mathbf{u}_r \quad \mathbf{v}_r\ ) = \mathbf{F}^{-1}\mathbf{H}(\ \mathbf{u}_b \quad \mathbf{v}_b\ ). \tag{3.4}$$

Thus every vertex $\mathbf{p}_i$ in the patch has a 2D parameter $\hat{\mathbf{p}}_i = (u_i, v_i)$.

Besides harmonic mapping, other parameterization techniques like barycentric mapping and shape preserving parameterization [40, 1] have been devised in the literature of computer graphics. Barycentric mapping and shape preserving parameterization differ from harmonic mapping in how to choose the value of the coefficient λ_{ij} (Equation 3.2). Alexa [1] has given the comparison of these parameterization techniques. Our framework simply requires that the parameterization scheme can make the vertices of the patch reasonably embedded in the plane polygon. Harmonic mapping is employed in our framework because it can minimize

the distortion in the edge lengths and attempts to preserve aspect ratios of the triangles.

The computation of the parameterization of a patch depends on some geometric properties of the patch shape. For a patch of either the primary or target character, its parameterization is based on its shape extracted from the dress shape of the character. Corresponding feature vertices of the primary and target characters are used to indicate the same characteristics on their skin shapes, therefore they should have identical parameters. When harmonic mapping is used to calculate the parameterization of a patch for a target character, the parameters of its feature vertices are not determined according to the ratios of boundary lengths like the primary character. Instead, they are assigned the same values as the parameters of the feature vertices of the corresponding primary character patch.

The parameterization algorithm flattens the patch into a polygon as if the patch is made of rubber sheet. The primary and target patches do not have the same internal mesh connectivity. The parameterizations of the primary and target patches depend on their own mesh connectivities and shapes. However the primary and target patches have the same number of feature vertices and boundaries. The parameters for their feature vertices are the same. Thus, the parameterizations of the primary and target patches are embedded into the same polygon, although their parameterizations are different.

3.3 Radial Basis Function Interpolation

Radial basis function interpolation technique is employed in EBTLE for two different purposes: the deformation of the base control mesh and the deformation of the skin shape. It is necessary to discuss it separately in this section. The multivariate interpolation problem occurs frequently in many branches of science and engineering. Several approaches, such as polynomial interpolation, tensor product method, radial basis function interpolation etc, have been developed to meet

the requirements of different applications. Radial basis functions are adequate for interpolating between a set of scattered high dimensional points [49]. It is quite popular in areas of computer graphics [73, 110, 122, 23].

The interpolation problem can be stated as: given a set of points in n_x dimension $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^{n_x}, 1 \leq i \leq m\}$, and a corresponding set of points in n_y dimension $\{\mathbf{y}_i | \mathbf{y}_i \in \mathbb{R}^{n_y}, 0 \leq i \leq m\}$, find a function $\boldsymbol{\psi} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ that satisfies the interpolation condition:

$$\boldsymbol{\psi}(\mathbf{x}_i) = \mathbf{y}_i \quad (1 \leq i \leq m). \tag{3.5}$$

The radial basis function interpolation technique takes function $\boldsymbol{\psi}(\mathbf{x})$ as the form:

$$\boldsymbol{\psi}(\mathbf{x}) = \sum_{i=1}^{m} \mathbf{w}_i \varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right), \tag{3.6}$$

where $\mathbf{w}_i$ is the n_y dimensional vector coefficient, and $\varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right)$ is the so called radial basis function. Usually, $\mathbf{x}_i$ is also referred to as the center of the radial basis function. Some commonly used radial basis functions are presented below:

1. The Gaussian function:
$$\varphi(r) = \exp\left(-\frac{r^2}{\sigma^2}\right) \tag{3.7}$$

2. The multi-quadratic function:
$$\varphi(r) = \sqrt{r^2 + \sigma^2} \tag{3.8}$$

3. The inverse multi-quadratic function:
$$\varphi(r) = 1 \Big/ \sqrt{r^2 + \sigma^2} \tag{3.9}$$

4. The thin plate spline function:
$$\varphi(r) = r^2 \ln\left(\|r\|\right) \tag{3.10}$$

where σ is a scalar.

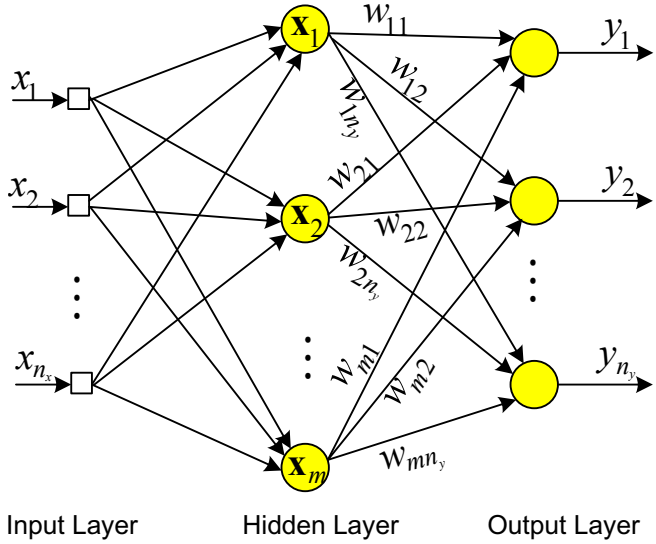

Figure 3.7: The architecture of a RBF network.

The representation of $\boldsymbol{\psi}(\mathbf{x})$ can also be described by a layered neural network as shown in Figure 3.7. The neurons in the hidden layer employ radial basis functions with different centers to do computation, while output neurons use weighted summations of their inputs as their outputs.

The weights associated with the links in the RBF network determine the behavior of the network. With appropriately configured weights, $\boldsymbol{\psi}(\mathbf{x})$ should be able to interpolate $\{\mathbf{y}_i(0 \leq i \leq m)\}$ exactly. To determine the proper weights, Equation 3.6 is substituted into Equation 3.5 and the following linear system is derived:

$$\begin{pmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1m} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{m1} & \varphi_{m2} & \cdots & \varphi_{mm} \end{pmatrix} \begin{pmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \\ \mathbf{w}_m^T \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \\ \mathbf{y}_m^T \end{pmatrix} \tag{3.11}$$

where $\varphi_{ij} = \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. Micchelli [77] has proved that for a large class of radial basis functions (including those defined by Equations 3.7 to 3.10), if $\{\mathbf{x}_i(1 \leq i \leq m)\}$ is a set of distinct points, then the matrix on the left side of Equation 3.11 is nonsingular. Thus the linear system has a unique solution. A lot of schemes, such as LU decomposition, singular value decomposition etc, have been devised to solve linear systems. They can effectively handle matrix equations of relatively large sizes. Function $\boldsymbol{\psi}(\mathbf{x})$ with coefficients as the solution of Equation 3.11 not only can

recreate the right values $\mathbf{y}_i (1 \leq i \leq m)$ at the training points $\mathbf{x}_i (1 \leq i \leq m)$ but also can continuously output reasonable value for any point in the space $\mathbb{R}^{n_x}$.

3.3.1 Variational Interpolation

Variational interpolation is an extension of RBF interpolation and is especially suitable for *smooth* function approximation [122, 23]. It is clearer to explain the variational interpolation technique in a two dimensional scenario, but the extension to higher dimension is straightforward. The interpolation problem now becomes:

Given a set of two dimensional points $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^2, 1 \leq i \leq m\}$, and a set of scalars $\{y_i | 1 \leq i \leq m\}$, find a *smooth* function $\psi(\mathbf{x})$ so that

$$\psi(\mathbf{x}_i) = y_i, \ (1 \leq i \leq m). \tag{3.12}$$

Function $\psi(\mathbf{x}) = \psi(x_1, x_2)$ defines a parametric surface in a three dimensional space. The smoothness of a surface in a 3D space [122] can be described by:

$$\varepsilon_s = \int_\Omega \left(\frac{\partial^2 \psi}{\partial x_1^2}\right)^2 + 2\left(\frac{\partial^2 \psi}{\partial x_1 \partial x_2}\right)^2 + \left(\frac{\partial^2 \psi}{\partial x_2^2}\right)^2 dx_1 dx_2 \tag{3.13}$$

ε_s is the integral of the squared summation of the partial derivatives of ψ over the total interested area Ω and describes the thin plate energy of the surface. A smooth surface will result in a small value of ε_s, while abrupt changes on the surface like creases will cause a large value of ε_s. Now the variational interpolation problem can be interpreted as: finding function $\psi(\mathbf{x})$ that satisfies the interpolation condition and minimizes ε_s. The solution of this problem can be expressed in the following form:

$$\psi(\mathbf{x}) = \sum_{i=1}^{m} w_i \varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right) + h_0 + h_1 x_1 + h_2 x_2$$

subject to the constraints:

$$\sum_{i=1}^{m} w_i = 0, \quad \sum_{i=1}^{m} w_i x_1^i = 0, \quad \sum_{i=1}^{m} w_i x_2^i = 0, \tag{3.14}$$

where $\mathbf{x}_i = (x_1^i, x_2^i)$, $\varphi\left(\|\mathbf{x} - \mathbf{x}_i\|\right)$ is the thin plate radial basis function (Equation 3.10), w_i is its scalar coefficient, and h_0, h_1, h_2 are the coefficients of a degree

one polynomial that accounts for the linear and constant portions of $\psi(\mathbf{x})$. The interpolation and constraint conditions (Equations 3.12 and 3.14) together lead to a linear system:

$$\begin{pmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1m} & 1 & x_1^1 & x_2^1 \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2m} & 1 & x_1^2 & x_2^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \varphi_{m1} & \varphi_{m2} & \cdots & \varphi_{mm} & 1 & x_1^m & x_2^m \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 \\ x_1^1 & x_1^2 & \cdots & x_1^m & 0 & 0 & 0 \\ x_2^1 & x_2^2 & \cdots & x_2^m & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \\ h_0 \\ h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.15}$$

with $\varphi_{ij} = \varphi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. The matrix in the the left side of the equation is symmetrical and positive semi-definite, so Equation 3.15 has a unique solution. With $w_i (1 \leq i \leq m)$ and h_0, h_1, h_2 found in this way, it is guaranteed that the surface described by function $\psi(\mathbf{x})$ has no excessive undulations and is as smooth as possible.

3.4 Deformation of Base Control Mesh

As shown in Figure 3.5, Base Control Mesh (BCM) is a very coarse polygonal mesh, but roughly describes the geometrical appearance of a character. It is hoped that the BCM can change its shape properly according to the skeleton movements at runtime. Previous layered skinning models [109, 113], which have a low-resolution mesh as the middle layer, only utilize the static shape of the low resolution mesh in the dress posture. Our framework makes use of multiple static shapes of the BCM in all example postures. The example shapes of the BCMs for both the primary and target characters are automatically generated in our framework.

The graph that represents the connectivity of the BCM is identical with the graph that describes the patch structure of the character skin. Every face and vertex of the BCM correspond to a patch and a feature vertex on the skin, respectively. The connectivities of the BCMs for the primary and target characters are equivalent because the skins of both characters are dissected in the same manner. Using a static shape of a character, we can obtain a static shape of the BCM by

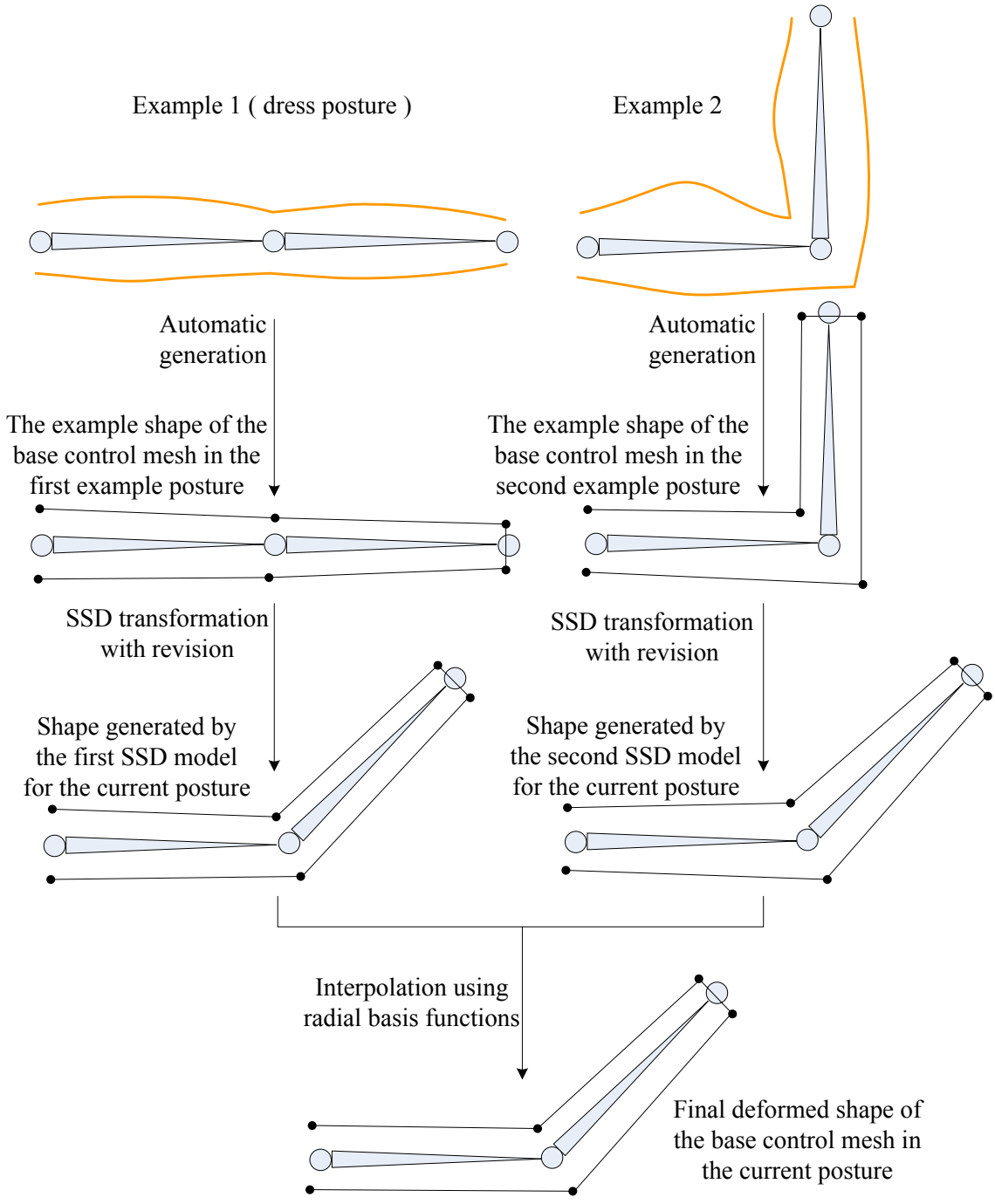

Figure 3.8: From the top to the bottom: Examples of the primary character. Example shapes of the BCM in different example postures. Shapes generated by different SSD models for the same new posture. The final deformed shape of the BCM in the new posture.

giving the BCM vertices the same positions of the corresponding feature vertices on the character shape. For the primary character, the example shapes of the BCM are easily generated using the example shapes of the character (see Figures 3.2 and 3.8). However, a target character only has one static shape in the dress posture, so we can only get one static shape for its BCM. The method that creates example shapes for the BCM of a target character in all example postures will be described later in Section 3.4.2.

With example shapes of the BCM in example postures, at runtime, our skinning model can synthesize the appropriate shape for the BCM from any skeleton posture. When the skeleton is in an example posture, the deformed shape of the BCM is supposed to be identical with the example shape of the BCM in that example posture. At runtime, the deformation of BCM in the posture that is not presented in the set of example postures should be consistent with its example shapes and should not have any obvious defects. As shown in Figures 3.2 and 3.8, the approach utilized to deform the shape of BCM is based on the simple Skeleton Subspace Deformation (SSD). Individual SSD models are created using example shapes of the BCM with example skeleton postures. On the fly, given a new skeleton posture, each individual SSD model can produce a deformed BCM shape based on the new posture. Since the deformed BCM shapes generated by different SSD models are different, they need to be combined to create the final shape of the BCM in the new posture. EBTLE utilizes radial basis function interpolation technique to blend different deformed BCM shapes.

3.4.1 Skeleton Subspace Deformation

Every example shape of BCM is bound with the skeleton in the corresponding example posture using SSD. SSD is an efficient technique to decide the deformed location of a skin vertex as a weighted sum of positions transformed by all influencing joints. There is a local frame associated with every joint in SSD. Suppose $\mathbf{T}_j^i$ is the transformation matrix of joint j when the skeleton is in the i-th example

posture, and $\mathbf{c}^i$ is the position of vertex $\mathbf{c}$ on the i-th example shape of the BCM. As the skeleton is animated to posture l, the new position of the vertex is:

$$\bar{\mathbf{c}}^{l,i} = \sum_j w_j \mathbf{T}_j^l \left(\mathbf{T}_j^i\right)^{-1} \mathbf{c}^i, \tag{3.16}$$

where j is the index of the joint which can affect vertex $\mathbf{c}$, $\mathbf{T}_j^l$ is the transformation matrix of joint j for posture l, and w_j is the weight of joint j to vertex $\mathbf{c}$.

Using n_e example shapes of the BCM, we can set up n_e SSD models with their binding postures as the corresponding example postures. The behavior of a SSD model is affected by the values of the weights. To make all SSD models deform in a similar manner, the weights of a vertex for an influencing joint in all SSD models are assigned the same value, which is estimated using the distance between the vertex of the BCM and the joint in the first example (dress posture). The first example shape of the BCM is generated from the first shape (dress shape) of a character and is also referred to as the dress shape of the BCM. Suppose vertex $\mathbf{c}$ is affected by the n nearest joints, where n can be selected by users, and the distance between vertex $\mathbf{c}$ and joint j is d_j. The weight w_j in Equation 3.16 can be formulated as:

$$w_j = \frac{1}{d_j} \Bigg/ \sum_{k=1}^{n} \frac{1}{d_k}$$

Although the weight values calculated above can make the shape of BCM deform appropriately, skeleton subspace deformation has its inherent shortcomings. When the rotation of a joint becomes large, the vertices around the joint move significantly towards the joint and unexpected shrinking happens near the joint as shown in Figure 3.9. It has been pointed out in [73] as "collapsing joint" and in [125] as "candy wrapper". To overcome this obvious defect, special attention has to be paid to where different body regions join with each other, such as elbow, shoulder etc.

The behavior of a SSD model is adjusted by maintaining the distances between a troublesome joint and those vertices around it during the animation, as illustrated in Figure 3.9. For vertex $\mathbf{c}$ of the base control mesh, if it is near a troublesome joint

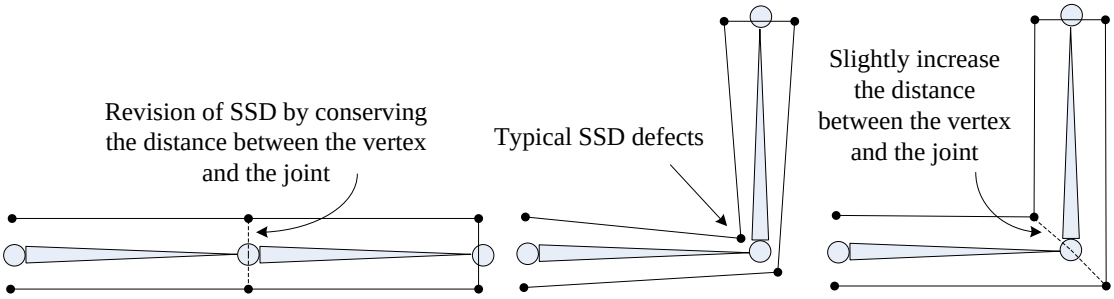

Figure 3.9: Left: The dress shape of the BCM in the dress posture. Middle: Deformation of the BCM through skeleton subspace deformation. Right: Revision of skeleton subspace deformation.

whose index is j, then its position evaluated by the i-th SSD model is corrected by following equation:

$$\tilde{\mathbf{c}}^{l,i} = \eta \left\| \mathbf{c}^i - \mathbf{o}_j^i \right\| \frac{\left(\bar{\mathbf{c}}^{l,i} - \mathbf{o}_j^l \right)}{\left\| \bar{\mathbf{c}}^{l,i} - \mathbf{o}_j^l \right\|} + \mathbf{o}_j^l, \tag{3.17}$$

where $\bar{\mathbf{c}}^{l,i}$ is the position of vertex $\mathbf{c}$ obtained via the i-th SSD model for the posture l (Equation 3.16), $\mathbf{c}^i$ is the position of vertex $\mathbf{c}$ in the i-th example shape of the BCM, $\mathbf{o}_j^i$ and $\mathbf{o}_j^l$ are the locations of joint j in the i-th example posture and current posture l respectively, and η is the scaling factor with the default value equivalent to 1.

A typical situation of setting η with a value other than one is during the flexion of an elbow joint. When the flexion angle is large, it is not sufficient to just conserve the distances between the elbow joint and the vertices around it. These distances need to be scaled slightly to make the area around the elbow look more reasonable (see Figure 3.9). The flexion angle is usually used to drive η so that the value of η changes as the arm bends. For example, if the SSD model is set up when the arm is straight, it will be better to make the value of η vary from 1 to around 1.4 when the flexion angle changes from 0 to 90 degrees at runtime. Since different situations may require this parameter to have different values, it is exposed to users so that users can tweak its value to obtain desired results. This parameter is simply a scale value, and its main purpose is to prevent shrinking or inflation. It is quite easy for users to figure out the right value. This amendment for SSD hardly requires

extra computation and is especially suitable for our situation. As the BCM is a coarse polygon, the correction of the positions of few vertices can largely improve the deformation quality of the BCM.

3.4.2 Example Shapes of the Base Control Mesh of a Target Character

At runtime, the behavior of the BCMs of the primary and target characters should be as similar as possible. The deformation of the primary BCM is dependent on its example shapes. (To make it more concise, we will use term "primary/target BCM" to denote the base control mesh of the primary/target character.) However, in our system, a target character is only required to supply one static shape in the dress posture, hence just one example shape for the target BCM can be derived. We need to generate examples shapes for the target BCM in all example postures (see Figure 3.2).

An example shape of the primary BCM can be treated as the transformation of its dress shape from the dress posture to the example posture. Assuming this transformation method is known, applying it to the dress shape of the target BCM would yield its example shape in the corresponding example posture. We take this transformation as the combination of skeleton subspace deformation and appropriate tweaking of the BCM vertices.

Suppose vertex $\mathbf{c}_{i,p}$ and $\mathbf{c}_{i,t}$ are corresponding vertices in the primary and target BCMs. The aim is to find the position of $\mathbf{c}_{i,t}$ in the j-th example posture. Given the positions of $\mathbf{c}_{i,p}$ and $\mathbf{c}_{i,t}$ in the dress shapes of the primary and target BCMs, we can use skeleton subspace deformation (Equation 3.16), with the same set of weights, to calculate their positions in the j-th example posture. Let $\bar{\mathbf{c}}_{i,p}^{j,1}$ and $\bar{\mathbf{c}}_{i,t}^{j,1}$ be the positions of $\mathbf{c}_{i,p}$ and $\mathbf{c}_{i,t}$ transformed from the dress posture to the j-th example posture through SSD using their positions on the dress shapes respectively. Usually, $\bar{\mathbf{c}}_{i,p}^{j,1}$ does not coincide with $\mathbf{c}_{i,p}^{j}$, the position of vertex $\mathbf{c}_{i,p}$ in the j-th example shape of the primary BCM. We expect that $\mathbf{c}_{i,t}^{j}$, the position of $\mathbf{c}_{i,t}$ in the unknown

example shape of the target BCM, will also not be in the same position with $\bar{\mathbf{c}}_{i,t}^{j,1}$. The displacement between $\mathbf{c}_{i,t}^{j}$ and $\bar{\mathbf{c}}_{i,t}^{j,1}$ is estimated according to the displacement between $\mathbf{c}_{i,p}^{j}$ and $\bar{\mathbf{c}}_{i,p}^{j,1}$. We need a transformation matrix $\mathbf{N}$, which can transform $\mathbf{c}_{i,p}^{j} - \bar{\mathbf{c}}_{i,p}^{j,1}$ into $\mathbf{c}_{i,t}^{j} - \bar{\mathbf{c}}_{i,t}^{j,1}$.

Since the primary and target BCMs have identical connectivity, we can assume that the direct neighbors of $\mathbf{c}_{i,p}$ and $\mathbf{c}_{i,t}$ are $\{\mathbf{c}_{k_1,p}, \mathbf{c}_{k_2,p}, \ldots, \mathbf{c}_{k_m,p}\}$ and $\{\mathbf{c}_{k_1,t}, \mathbf{c}_{k_2,t}, \ldots, \mathbf{c}_{k_m,t}\}$, respectively. The transformation matrix $\mathbf{N}$ needs to satisfy the following condition:

$$\mathbf{N}\left(\bar{\mathbf{c}}_{k_a,p}^{j,1} - \bar{\mathbf{c}}_{i,p}^{j,1}\right) = \left(\bar{\mathbf{c}}_{k_a,t}^{j,1} - \bar{\mathbf{c}}_{i,t}^{j,1}\right) \quad (1 \le a \le m), \tag{3.18}$$

where $\bar{\mathbf{c}}_{k_a,p}^{j,1}$ and $\bar{\mathbf{c}}_{k_a,t}^{j,1}$ are the positions of $\mathbf{c}_{k_a,p}$ and $\mathbf{c}_{k_a,t}$ calculated by SSD for the j-th example posture based on the their positions on the dress shapes of the primary and target BCMs respectively. By assembling the following two matrices:

$$\begin{aligned}
\mathbf{C}_1 &= \left(\bar{\mathbf{c}}_{k_1,p}^{j,1} - \bar{\mathbf{c}}_{i,p}^{j,1}, \bar{\mathbf{c}}_{k_2,p}^{j,1} - \bar{\mathbf{c}}_{i,p}^{j,1}, \ldots, \bar{\mathbf{c}}_{k_m,p}^{j,1} - \bar{\mathbf{c}}_{i,p}^{j,1}\right), \\
\mathbf{C}_2 &= \left(\bar{\mathbf{c}}_{k_1,t}^{j,1} - \bar{\mathbf{c}}_{i,t}^{j,1}, \bar{\mathbf{c}}_{k_2,t}^{j,1} - \bar{\mathbf{c}}_{i,t}^{j,1}, \ldots, \bar{\mathbf{c}}_{k_m,t}^{j,1} - \bar{\mathbf{c}}_{i,t}^{j,1}\right),
\end{aligned}$$

Equation 3.18 can be rewritten in a matrix form:

$$\mathbf{N}\mathbf{C}_1 = \mathbf{C}_2$$

The solution of the above equation is:

$$\mathbf{N} = \mathbf{C}_2\mathbf{C}_1^T\left(\left(\mathbf{C}_1\mathbf{C}_1^T\right)^{-1}\right)^T$$

Using matrix $\mathbf{N}$, the position of vertex $\mathbf{c}_{i,t}$ in the j-th example posture is:

$$\mathbf{c}_{i,t}^{j} = \bar{\mathbf{c}}_{i,t}^{j,1} + \mathbf{N}\left(\mathbf{c}_{i,p}^{j} - \bar{\mathbf{c}}_{i,p}^{j,1}\right)$$

After the positions of all the vertices of the target BCM in the j-th example posture are determined, the j-th example shape of the target BCM is formed.

Here we want to automatically generate example shapes for the target BCM. The difficulty is that there is no way to determine the exact appearances of the example shapes of the target BCM, because the example shapes of the target

character are not available. The technique is based on the assumption that the relationship between the example shape of the target BCM and its dress shape is the same as the one between the example shape of the primary BCM and its dress shape. This relationship is described as the combination of SSD algorithm and some matrix transformation. Although the rational behind this method is quite sound and the resultant example shapes for the target BCM are acceptable, the reality is that the assumption is not always true, because the preference of artists for the primary and target characters may be different.

The ideal situation is that the artists also provide example shapes for the target character, and then the example shapes of the target BCM will be derived from these example shapes. But, the example shapes of the target character are not required by our framework. As we can see, the example shapes of the target BCM are supposed to depend on the work of artists. So it makes sense for artists to be able to adjust the automatically generated example shapes of the target BCM. Since the target BCM has only a few vertices, it is much easier to adjust than sculpting example shapes for the target character. Artists can have the desired example shape of the target character in mind, and then according to the relation between the target character and its BCM, corresponding tweak can be easily made on the automatically generated example shape of the target BCM. The automatic technique can generates a prototype shape for the target BCM in the example posture, and then some adjustments can be done by simply changing the positions of a few vertices as desired.

In our experiments, usually less than ten vertices need to be adjusted for one example shape, and about half of the generated example shapes are adjusted. It can be finished in a matter of few minutes. We think it is worth manual adjustments to finely tune the deformation results, especially when the labor of adjustments is quite small.

3.4.3 Blending Skeleton Subspace Deformation Models

At runtime, when the skeleton is animated to a new posture, each SSD model built from the example shape of the BCM can generate a shape in that posture using the revised SSD transformation described in Section 3.4.1. These shapes created by different SSD models in the same posture are not exactly the same. We obtain the final deformed shape of the BCM by blending all these shapes (see Figure 3.2).

To determine the deformed shape of the BCM, the new coordinates of its vertices have to be decided. For vertex $\mathbf{c}$ of the BCM, we first need to determine the set of skeleton rotational DOFs, $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{n_c}\}$, which have an impact on how the position of $\mathbf{c}$ is changed. This set $\boldsymbol{\theta}$ can be automatically deduced from the set of influencing joints of vertex $\mathbf{c}$, which are used for skeleton subspace deformation, and DOFs of each joint, or can be directly specified by users. Then, a cardinal weight function is constructed for every example shape of the BCM. $\omega_i(\boldsymbol{\theta})$ is used to denote the cardinal weight function associated with the i-th example shape of the BCM. The value of $\omega_i(\boldsymbol{\theta})$ varies according to the posture of the skeleton, but it should meet the following requirements:

$$\omega_i(\boldsymbol{\theta}) = \begin{cases} 1 & \boldsymbol{\theta} = \boldsymbol{\theta}^i \\ 0 & \boldsymbol{\theta} = \boldsymbol{\theta}^j \quad (i \neq j) \end{cases} \tag{3.19}$$

where $\boldsymbol{\theta}^i = (\theta_1^i, \theta_2^i, \ldots, \theta_{n_c}^i)$ are the values of the joint angles $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_{n_c}\}$ in the i-th example posture.

Radial basis function technique is ideal for solving this kind of interpolation problem, and function $\omega_i(\boldsymbol{\theta})$ is written as:

$$\omega_i(\boldsymbol{\theta}) = \sum_{j=1}^{n_e} a_{ij} \varphi(\|\boldsymbol{\theta} - \boldsymbol{\theta}^j\|), \tag{3.20}$$

where a_{ij} is the scalar coefficients of radial basis function, $\varphi(\|\boldsymbol{\theta} - \boldsymbol{\theta}^j\|)$ is the radial basis function itself, and n_e is the number of examples. Gaussian function (Equation 3.7) is employed as the radial basis function, and the drop off ratio for every example is controlled by σ (Equation 3.7) and can be adjusted by animators

to control the interpolation quality. Since each individual SSD model can independently produce deformed shape of the BCM, the blending process is just to ensure that the blended shape fits the example shapes of BCM in corresponding example postures. The parameter σ is not very sensitive and easy to tune. If the value of σ is too small, the interpolation will be determined entirely by a single example when the skeleton posture is near the corresponding example posture. So a larger value of σ is recommend for smoother interpolation. In our experiments, when the parameter is set between 1.5 and 20, the deformation of the BCM shape behaves similarly well.

Similar as described in Section 3.3, the combination of Equation 3.20 and Equation 3.19 gives a linear system:

$$\mathbf{R}\mathbf{a}_i = \mathbf{e}_i,$$

where $\mathbf{R}$ is a $n_e \times n_e$ matrix with element $r_{ij} = \varphi(\|\boldsymbol{\theta}^i - \boldsymbol{\theta}^j\|)$, $\mathbf{e}_i$ is a unit vector with its i-th element equivalent to 1 and other elements equivalent to 0, and $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{in_e})^T$. The solution of this linear system is the coefficients of $\omega_i(\boldsymbol{\theta})$.

With a cardinal weight function associated with every example shape of the BCM, the position of vertex $\mathbf{c}$ in the new posture l is formulated:

$$\mathbf{c}^l = \sum_{i=1}^{n_e} \bar{\omega}_i(\boldsymbol{\theta}^l)\tilde{\mathbf{c}}^{l,i}, \tag{3.21}$$

where $\tilde{\mathbf{c}}^{l,i}$ is the position of vertex $\mathbf{c}$ calculated using the i-th revised SSD model for current posture l (Equation 3.17), and

$$\bar{\omega}_i(\boldsymbol{\theta}^l) = \frac{\omega_i(\boldsymbol{\theta}^l)}{\sum_{j=1}^{n_e} \omega_j(\boldsymbol{\theta}^l)}.$$

The normalization of the cardinal weights is to ensure that the blended shape of the BCM will never degenerate.

As the performance of each individual revised SSD model is quite acceptable, the interpolation between all of them is a very robust scheme and the synthesized shape of the BCM does not exhibit any prominent defects. According to Equations 3.16,

3.17 and 3.21, it is easy to conclude that if the skeleton is in the i-th example posture, the position of vertex $\mathbf{c}$ in the deformed shape of the BCM is identical with its position in its i-th example shape.

Different users have their own criteria for measuring how satisfactory the deformation of the BCM is. Due to the personal preference of users, EBTLE can not computationally determine the level of satisfactory. Users have to play a part in deciding whether the deformation of the BCM is appealing or not. If the user is not satisfied with the results, another example with desired deformation could be added and used to adjust the results. The shape of the BCM is the middle layer between the skeleton and the skin, and users should have some control over its deformation.

3.5 Skin Deformation Based on the Dress Shape of a Character

To make the skin of a character deform naturally as the skeleton changes its posture, a control mechanism that can manipulate the skin shape through the base control mesh (BCM) will have to be developed. A correspondence between the skin mesh and the BCM will be a helpful hint on devising such a mechanism. As mentioned in Section 3.1, the skin of a character is decomposed into patches and each patch corresponds with a face of the BCM. The mission will be easier to accomplish if a bijective mapping is set up between the vertices of a patch and the points on the corresponding face of the BCM. In this section, we describe the skin deformation method that is only based on the dress shape of a character, hence there is no problem to apply it to either the primary or target character. In the next section, we will extend this skinning model so that the deformation performances of both the primary and target characters will follow the lead of the examples of the primary character.

3.5.1 Register Points

For a vertex $\mathbf{p}_i$ in a skin patch, we need to locate a corresponding point $\mathbf{q}_i$, called *register point*, on the corresponding face of the base control mesh as illustrated in Figure 3.6. The position of the register point $\mathbf{q}_i$ will be described by its barycentric coordinates with respect to the vertices of the BCM face. In Section 3.2, the parameterization of a patch is embedded in a 2D planar polygon with the parameters of its feature vertices as the vertices of the polygon. Let $\hat{\mathbf{p}}_i$ be the 2D parameter of vertex $\mathbf{p}_i$, $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$ be the parameters of the feature vertices of the patch, and $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be the barycentric coordinates of $\hat{\mathbf{p}}_i$ in polygon $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$. Then the position of register point $\mathbf{q}_i$ can be described by:

$$\mathbf{q}_i^l = \sum_{k=1}^{n} \alpha_k \mathbf{c}_{j_k}^l, \tag{3.22}$$

where $\{\mathbf{c}_{j_1}, \mathbf{c}_{j_2}, \ldots, \mathbf{c}_{j_n}\}$ are the vertices of the BCM corresponding with the feature vertices of the patch $\{\mathbf{p}_{f_1}, \mathbf{p}_{f_2}, \ldots, \mathbf{p}_{f_n}\}$, and $\{\mathbf{c}_{j_1}^l, \mathbf{c}_{j_2}^l, \ldots, \mathbf{c}_{j_n}^l\}$ are their positions in posture l calculated using Equation 3.21.

The barycentric coordinates of $\hat{\mathbf{p}}_i$ with respect to the polygon $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$ in the parameter domain can be evaluated as follows:

- If $\mathbf{p}_i$ is a feature vertex, i.e. $\mathbf{p}_i = \mathbf{p}_{f_j}$, its barycentric coordinates $(\alpha_1, \alpha_2, \ldots, \alpha_n) = \mathbf{e}_j^T$, where $\mathbf{e}_j^T$ is the unit vector with its j-th element equivalent to one and other elements equivalent to zero.

- If $\mathbf{p}_i$ is a vertex along the boundary between $\mathbf{p}_{f_j}$ and $\mathbf{p}_{f_{j+1}}$, then $\alpha_j = (1-\gamma)$, $\alpha_{j+1} = \gamma$, and $\alpha_k = 0 (k \neq j, k \neq j+1)$, where $\gamma = \frac{\|\hat{\mathbf{p}}_{f_j}\hat{\mathbf{p}}_i\|}{\|\hat{\mathbf{p}}_{f_j}\hat{\mathbf{p}}_{f_{j+1}}\|}$

- Otherwise, $\hat{\mathbf{p}}_i$ lies inside polygon $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$. Solving the barycentric coordinates of $\hat{\mathbf{p}}_i$ in a polygon with more than three edges is a underdetermined problem. Firstly, we obtain n sets of barycentric coordinates based on triangular barycentric coordinates. For every $\hat{\mathbf{p}}_{f_j}$, we can find a triangle $\{\hat{\mathbf{p}}_{f_j}, \hat{\mathbf{p}}_{f_k}, \hat{\mathbf{p}}_{f_{k+1}}\}$ that contains $\hat{\mathbf{p}}_i$. Suppose the barycentric coordinates of $\hat{\mathbf{p}}_i$ in $\{\hat{\mathbf{p}}_{f_j}, \hat{\mathbf{p}}_{f_k}, \hat{\mathbf{p}}_{f_{k+1}}\}$ is $(\gamma_1, \gamma_2, \gamma_3)$, and $(\beta_1^j, \beta_2^j, \ldots, \beta_n^j)$ is the j-th

set of barycentric coordinates of $\hat{\mathbf{p}}_i$ in $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$. $(\beta_1^j, \beta_2^j, \ldots, \beta_n^j)$ can be constructed from $(\gamma_1, \gamma_2, \gamma_3)$ by making $\beta_j^j = \gamma_1$, $\beta_k^j = \gamma_2$, $\beta_{k+1}^j = \gamma_3$, and others equal zero. Then, using these n sets of barycentric coordinates $(\beta_1^j, \beta_2^j, \ldots, \beta_n^j)(1 \leq j \leq n)$, the final barycentric coordinates of $\hat{\mathbf{p}}_i$ in $\{\hat{\mathbf{p}}_{f_1}, \hat{\mathbf{p}}_{f_2}, \ldots, \hat{\mathbf{p}}_{f_n}\}$ are defined:

$$(\alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{1}{n} \sum_{j=1}^{n} \left(\beta_1^j, \beta_2^j, \ldots, \beta_n^j\right)$$

With the barycentric coordinates $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ of $\hat{\mathbf{p}}_i$, the world coordinates of the register point of vertex $\mathbf{p}_i$ can be easily located on the shape of the BCM for any skeleton posture using Equation 3.22. The mapping from the skin mesh to the base control mesh is thus established.

3.5.2 Transformation by the Base Control Mesh

During the animation, the shape of the BCM is naturally deformed by the skeleton, and the position of the register point of vertex $\mathbf{p}_i$ on the deformed shape of the BCM is evaluated through Equation 3.22. However, to determine the world coordinates of vertex $\mathbf{p}_i$ in the current posture, our skinning model should be able to estimate the displacement between $\mathbf{p}_i$ and $\mathbf{q}_i$ for any skeleton posture. With the dress shapes of the character and BCM available, the displacement between $\mathbf{p}_i$ and $\mathbf{q}_i$ in the dress posture can be easily measured. But in order to use this measured displacement in other skeleton postures, it should be measured in a local frame.

There is a local frame built on each face of the BCM with the center of the face as its origin. The z-axis of the frame is the normal of the face, the x-axis is orthogonal with z-axis and points from the origin of the frame to the middle point of the first edge of the face, and y-axis is the cross product of z-axis and x-axis. The location and orientation of the local frame will follow the change of the BCM face at runtime. In such a local frame, the displacement between $\mathbf{p}_i$ and $\mathbf{q}_i$ in the dress posture is defined as:

$$\mathbf{d}_i^1 = (\mathbf{M}^1)^{-1} \left(\mathbf{p}_i^1 - \mathbf{q}_i^1\right),$$

where $\mathbf{M}^1$ is the transformation matrix of the local frame associated with the BCM face in the dress posture, $\mathbf{p}_i^1$ is the position of vertex $\mathbf{p}_i$ in the dress shape of the character, and $\mathbf{q}_i^1$ is the position of the register point of vertex $\mathbf{p}_i$ on the dress shape of the BCM (the first example shape of the BCM). Then, at runtime, the deformed position of vertex $\mathbf{p}_i$ in posture l is:

$$\mathbf{p}_i^l = \mathbf{q}_i^l + \mathbf{M}^l \mathbf{d}_i^1, \tag{3.23}$$

where $\mathbf{M}^l$ is the transformation matrix of the local frame in posture l and $\mathbf{q}_i^l$ is the position of the register point in posture l calculated using Equation 3.22.

3.5.3 Patch Stitching

The skin vertices belonging to different patches are transformed by different local frames associated with different faces of the BCM, therefore it is very likely that discontinuities occur along the boundaries. We use a simple but effective postprocessing step to handle this situation. Firstly, the new position of a boundary vertex is computed as the average of its positions from the two neighboring patches (for a feature vertex, the new position is the average of its positions from all incident patches). Vector $\mathbf{s}_i$, termed *stitching displacement*, is the difference between the new position and old position of a boundary vertex $\mathbf{p}_i (n_p - n_b \leq i < n_p)$. Then, the stitching displacement of a non-boundary vertex in the patch is estimated as the linear combination of the stitching displacements of all its direct neighbors. Using the same coefficients from harmonic mapping (Equation 3.2), stitching displacement of vertex $\mathbf{p}_i (0 \leq i < n_p - n_b)$ is written out as:

$$\mathbf{s}_i = \sum_{\{i,j\} \in \mathcal{G}} \lambda_{ij} \mathbf{s}_j. \tag{3.24}$$

Putting Equation (3.24) for all non-boundary vertices altogether yields:

$$\mathbf{F}\mathbf{S}_r = \mathbf{H}\mathbf{S}_b, \tag{3.25}$$

where matrix $\mathbf{F}$ and $\mathbf{H}$ are identical with the matrices in Equation (3.3), and $\mathbf{S}_r = \left(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{(n_p - n_b - 1)}\right)^T$, $\mathbf{S}_b = \left(\mathbf{s}_{(n_p - n_b)}, \mathbf{s}_{(n_p - n_b + 1)}, \ldots, \mathbf{s}_{(n_p - 1)}\right)^T$. The solution

of Equation 3.25, $\mathbf{S}_r = \mathbf{F}^{-1}\mathbf{H}\mathbf{S}_b$, is the stitching displacements of non-boundary vertices. For implementation, $\mathbf{F}^{-1}\mathbf{H}$ is precalculated and stored for multiplication with $\mathbf{S}_b$ on the fly. Now the position of vertex $\mathbf{p}_i$ for posture l after stitching is updated using:

$$\mathbf{p}_i^l = \mathbf{p}_i^l + \mathbf{s}_i^l.$$

This simple stitching algorithm can only guarantee the first order continuity. It adjusts the interior vertices of a patch according to the positions of its boundary vertices, and the boundaries of neighboring patches are merged by averaging vertex positions. If the transformations applied to two patches are very different, it is possible that a crease will be formed along a boundary.

From our observation, the transformations for the patches on the torso area are usually not so different, and the stitching procedure is sufficient to handle the gaps between these patches. The troublesome area is the regions around the joints of the limbs of a character, e.g. the elbow area of a human arm, because usually these joints have large degree of freedom, and the transformations applied for the patches below and above the joint could differ greatly. However, in these cases, creases sometimes are acceptable. For example, when the elbow joint is bent too much, the upper and lower arms will push together with each other, and a crease is expected to appear on the skin around the elbow.

When an undesired crease is detected on the skin, there are still ways to remove it. The reason that a boundary is developed into a crease is that the difference between the transformations for neighboring patches is too large. The artists may increase the number of patches and the number of joints in the problematic region, so that different joints drive different patches while the rotation is evenly distributed to multiple joints. Thus, the transformation applied to different patches will not be so different and the stitching procedure will handle them properly.

This stitching algorithm is far from perfect. However it is computationally efficient. It is utilized as the compromise between the speed and continuity. Never-

theless, if high order continuity is the goal of our application, based on the results from this stitching algorithm, a relaxation procedure like the one used by Wilhelms and Gelder [128] can be applied on both interior and boundary vertices to smooth the creases. This kind of relaxation procedure may not be suitable for interactive application, because when the total number of the vertices is large, the relaxation procedure will take some time to reach suitable positions.

3.6 Guidance of Examples

Although the method outlined in the last section can generate seamless skin deformations at runtime, no muscular characteristics like bulge can be simulated by the current model, and there is no way for artists to control the fine details of the deformations. Setting examples of desired deformations is an intuitive and expressive approach to overcome these deficiencies. For the primary character, the skinning model should be able to recreate the example shapes in the example postures. For a target character, the shapes synthesized by the skinning model in the example postures should have the same style of skin deformations as the example shapes of the primary character. Moreover, the skin deformations of both the primary and target characters in skeleton postures which are not seen in the set of example postures are also supposed to be aesthetically satisfactory and consistent with the example shapes. The most important property of EBTLE is that the examples of the primary character can be used to guide the skin deformations for not only the primary character itself but also any target character with similar appearances. The control mechanism for the skin deformations that enables the primary character to change its shape in the same manner as shown by the examples can be transferred to the target characters.

EBTLE does not require users to provide a set of examples for every character, but it demands a set of examples for the primary character. Usually the example shapes are modeled by artists using 3D modeling software packages (e.g. Maya [2]).

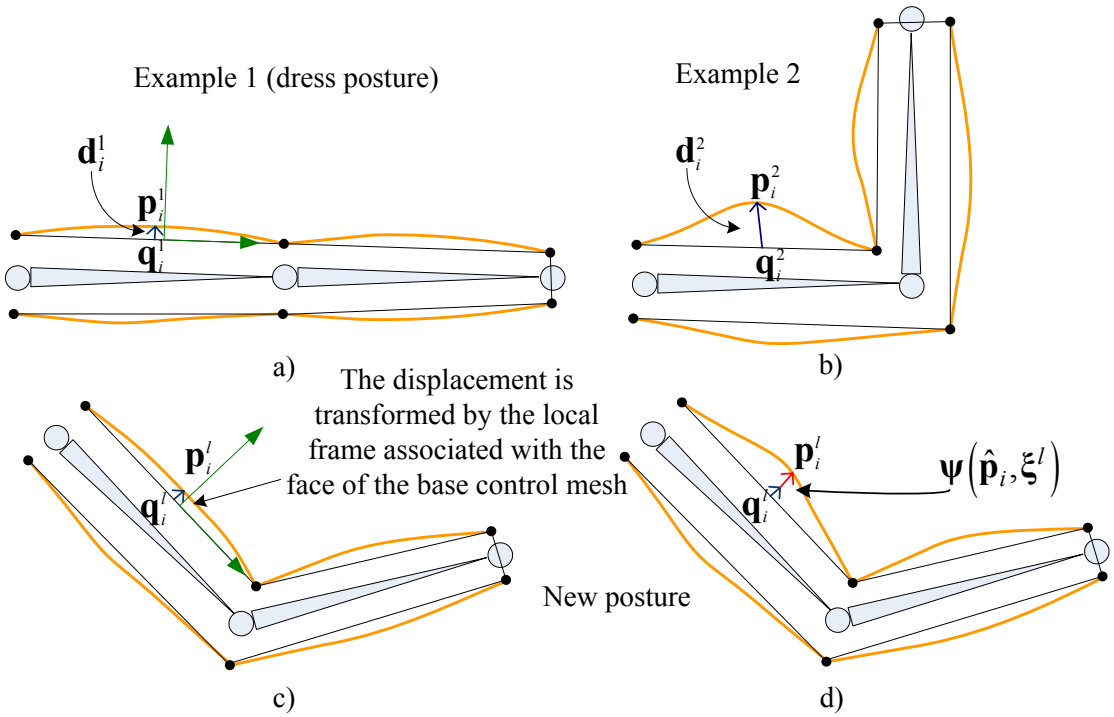

Figure 3.10: a) The dress shape of the character, the dress shape of the BCM and the dress skeleton posture. b) The second example character shape, the second example shape of the BCM and the second example posture. c) The deformed shape of the character in the new posture generated using the pure layered skinning model. d) The deformation results created by the extended skinning model which takes the second example into account.

Users may also create example shapes through other methods like range scan. As long as the example shapes have the same mesh connectivity as the dress shape of the primary character, our system will accept the examples.

3.6.1 Universal Representation of Skin Deformations

To incorporate examples of the primary character into our framework and make the skinning model flexible enough to represent all kinds of deformations, Equation 3.23 is extended as:

$$\mathbf{p}_i^l = \mathbf{q}_i^l + \mathbf{M}^l \left(\mathbf{d}_i^1 + \mathbf{B}\boldsymbol{\psi}(\hat{\mathbf{p}}_i, \boldsymbol{\xi}^l)\right) \tag{3.26}$$

$$= \mathbf{q}_i^l + \mathbf{M}^l \left(\mathbf{d}_i^1 + \mathbf{B}\boldsymbol{\psi}(u_i, v_i, \xi_1^l, \xi_2^l, \ldots, \xi_{n_a}^l)\right), \tag{3.27}$$

where $\boldsymbol{\psi}(\hat{\mathbf{p}}_i, \boldsymbol{\xi}^l)$ is the *residual* of vertex $\mathbf{p}_i$ for skeleton posture l (see Figure 3.10), $\hat{\mathbf{p}}_i = (u_i, v_i)$ is the 2D parameter of vertex $\mathbf{p}_i$, $\boldsymbol{\xi}^l = (\xi_1^l, \xi_2^l, \ldots, \xi_{n_a}^l)$ are values of the influencing joint angles in posture l, and $\mathbf{B}$ is a scaling matrix. The influencing

joint angles for the patch $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_{n_a})$ are the rotational DOFs of the skeleton whose values influence the deformations of the patch, and they are specified by animators. As different characters have different sizes, matrix $\mathbf{B}$ can make the residuals estimated by function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ more compatible with the current character. Not every patch needs the extended formulation for its deformation. If the artists are satisfied with the deformation results on a patch generated by the pure layered skinning model described in Section 3.5, then there is no need to utilized Equation 3.26 and no examples are demanded by this patch so that the computational cost can be minimized. For an individual patch, users can decide which examples are used to guide its deformation. Different patches may have different sets of examples.

Function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ can estimate the residual of a vertex based on its 2D parameter and the skeleton posture. For a patch that demands more flexible deformations than what can be provided by the pure layered skinning model, a RBF network representing function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ will be constructed, and the residuals of all vertices in the patch are estimated through it. Function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ is not fabricated for the patch of a particular character but for the corresponding patches of both primary and target characters. The primary character provides examples to configure this function. After its coefficients are determined, it can be used to calculate the residuals of the vertices in the corresponding patch of any character.

As mentioned in Section 3.1, for a patch of the primary character, its shape in every example character shape can be extracted straightforwardly. Using the example patch shape and the corresponding example shape of the base control mesh (BCM), as shown in Figure 3.10, we can measure the displacement between vertex $\mathbf{p}_i$ and its register point $\mathbf{q}_i$ for every example posture:

$$\mathbf{d}_i^j = (\mathbf{M}^j)^{-1} \left(\mathbf{p}_i^j - \mathbf{q}_i^j\right) \quad (1 \leq j \leq n_e),$$

where $\mathbf{p}_i^j$ is the position of $\mathbf{p}_i$ in the j-th example shape, $\mathbf{q}_i^j$ is the position of the register point of $\mathbf{p}_i$ in the j-th example shape of the BCM (Equation 3.22), and

$\mathbf{M}^j$ is the transformation matrix of the local frame built on the face of the j-th example shape of the BCM.

To make sure that the the shape synthesized by our skinning model for an example posture is identical with the corresponding example shape, function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ must satisfy the following condition:

$$\boldsymbol{\psi}(\hat{\mathbf{p}}_i, \boldsymbol{\xi}^j) = \mathbf{d}_i^j - \mathbf{d}_i^1 \quad (1 \le i \le n_p, 1 \le j \le n_e), \tag{3.28}$$

where n_p is the vertex number in the patch and n_e is the number of examples that influence this patch. Besides the above condition, function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ should be smooth enough to generalize well over both the skin vertex parameter variables $\hat{\mathbf{p}} = (u, v)$ and joint angle variables $\boldsymbol{\xi} = \{\xi_1, \xi_2, \ldots, \xi_{n_a}\}$.

3.6.2 Construction of Residual Function

Variational interpolation technique is utilized to construct function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$. Similar as outlined in Section 3.3.1, function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ is written as the weighted summation of $n_p \times n_e$ radial basis functions plus a linear term.

$$\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi}) = \sum_{j=1}^{n_e} \sum_{i=1}^{n_p} \mathbf{w}_i^j \varphi(\|(\hat{\mathbf{p}}, \boldsymbol{\xi}) - (\hat{\mathbf{p}}_i, \boldsymbol{\xi}^j)\|) + \mathbf{h}_0 + \mathbf{h}_1 u + \mathbf{h}_2 v + \sum_{i=1}^{n_a} \mathbf{h}_{i+2} \xi_i, \tag{3.29}$$

In the above equation, $\varphi\left(\|(\hat{\mathbf{p}}, \boldsymbol{\xi}) - (\hat{\mathbf{p}}_i, \boldsymbol{\xi}^j)\|\right)$ is the thin plate radial basis function (Equation 3.10), $\mathbf{w}_i^j$ is the 3D vector coefficient, and $\mathbf{h}_i (0 \le i \le n_a + 2)$ are the 3D vector coefficients of the linear term. To make $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ a smooth interpolant, the constraint condition is added:

$$\sum_{j=1}^{n_e} \sum_{i=1}^{n_p} w_{ik}^j \begin{pmatrix} 1 \\ \hat{\mathbf{p}}_i^T \\ (\boldsymbol{\xi}^j)^T \end{pmatrix} = \mathbf{0} \qquad (1 \le k \le 3) \ ,$$

where w_{ik}^j is the k-th element of vector $\mathbf{w}_i^j$. With these constraints, substituting Equation 3.29 into Equation 3.28 yields:

$$\mathbf{GW} = \boldsymbol{\Delta}\mathbf{D} \tag{3.30}$$

$$\mathbf{G} = \begin{pmatrix}
\varphi_{11}^{11} & \cdots & \varphi_{1n_p}^{11} & \varphi_{2i}^{11} & \varphi_{3i}^{11} & \varphi_{n_e1}^{11} & \cdots & \varphi_{n_en_p}^{11} & 1 & \hat{\mathbf{p}}_1 & \boldsymbol{\xi}^1 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\varphi_{11}^{1n_p} & \cdots & \varphi_{1n_p}^{1n_p} & \varphi_{2i}^{1n_p} & \varphi_{3i}^{1n_p} & \varphi_{n_e1}^{1n_p} & \cdots & \varphi_{n_en_p}^{1n_p} & 1 & \hat{\mathbf{p}}_{n_p} & \boldsymbol{\xi}^1 \\
\varphi_{11}^{2i} & \cdots & \varphi_{1n_p}^{2i} & \varphi_{2i}^{2i} & \varphi_{3i}^{2i} & \varphi_{n_e1}^{2i} & \cdots & \varphi_{n_en_p}^{2i} & 1 & \hat{\mathbf{p}}_i & \boldsymbol{\xi}^2 \\
\varphi_{11}^{3i} & \cdots & \varphi_{1n_p}^{3i} & \varphi_{2i}^{3i} & \varphi_{3i}^{3i} & \varphi_{n_e1}^{3i} & \cdots & \varphi_{n_en_p}^{3i} & 1 & \hat{\mathbf{p}}_i & \boldsymbol{\xi}^3 \\
\varphi_{11}^{n_e1} & \cdots & \varphi_{1n_p}^{n_e1} & \varphi_{2i}^{n_e1} & \varphi_{3i}^{n_e1} & \varphi_{n_e1}^{n_e1} & \cdots & \varphi_{n_en_p}^{n_e1} & 1 & \hat{\mathbf{p}}_1 & \boldsymbol{\xi}^{n_e} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\varphi_{11}^{n_en_p} & \cdots & \varphi_{1n_p}^{n_en_p} & \varphi_{2i}^{n_en_p} & \varphi_{3i}^{n_en_p} & \varphi_{n_e1}^{n_en_p} & \cdots & \varphi_{n_en_p}^{n_en_p} & 1 & \hat{\mathbf{p}}_{n_p} & \boldsymbol{\xi}^{n_e} \\
1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 & 0 & \mathbf{0} & \mathbf{0} \\
\hat{\mathbf{p}}_1^T & \cdots & \hat{\mathbf{p}}_{n_p}^T & \hat{\mathbf{p}}_i^T & \hat{\mathbf{p}}_i^T & \hat{\mathbf{p}}_1^T & \cdots & \hat{\mathbf{p}}_{n_p}^T & 0 & \mathbf{0} & \mathbf{0} \\
(\boldsymbol{\xi}^1)^T & \cdots & (\boldsymbol{\xi}^1)^T & (\boldsymbol{\xi}^2)^T & (\boldsymbol{\xi}^3)^T & (\boldsymbol{\xi}^{n_e})^T & \cdots & (\boldsymbol{\xi}^{n_e})^T & 0 & \mathbf{0} & \mathbf{0}
\end{pmatrix}$$

$$\begin{aligned}
\mathbf{W} &= \left(\mathbf{w}_1^1, \ldots, \mathbf{w}_{n_p}^1, \mathbf{w}_1^2, \ldots, \mathbf{w}_{n_p}^2, \ldots, \mathbf{w}_1^{n_e}, \ldots, \mathbf{w}_{n_p}^{n_e}, \mathbf{h}_0, \ldots, \mathbf{h}_{n_a+2}\right)^T \\
\boldsymbol{\Delta}\mathbf{D} &= \left(\Delta\mathbf{d}_1^1, \ldots, \Delta\mathbf{d}_{n_p}^1, \Delta\mathbf{d}_1^2, \ldots, \Delta\mathbf{d}_{n_p}^2, \ldots, \Delta\mathbf{d}_1^{n_e}, \ldots, \Delta\mathbf{d}_{n_p}^{n_e}, \mathbf{0}, \ldots, \mathbf{0}\right)^T
\end{aligned}$$

where $\varphi_{ab}^{cd} = \varphi\left(\|(\hat{\mathbf{p}}_d, \boldsymbol{\xi}^c) - (\hat{\mathbf{p}}_b, \boldsymbol{\xi}^a)\|\right)$, and $\Delta\mathbf{d}_i^j = \mathbf{d}_i^j - \mathbf{d}_i^1$. The dimension of matrix $\mathbf{G}$ is mainly determined by the number of examples and number of vertices in the patch. LU decomposition is applied to solve this linear system. The amount of work required by LU decomposition grows with the dimension of matrix $\mathbf{G}$. After the coefficients of function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ are determined, at runtime, all computation demanded is simply evaluation of the function. Function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ is able to generalize over variables $\hat{\mathbf{p}}$ and $\boldsymbol{\xi}$ well, hence it can give an appropriate estimation of the residual for any vertex parameter in any skeleton posture. It describes a smooth hyper-surface in the space with skeletal parameters and skin vertex parameters as its axes.

3.6.3 Residual Scaling

Function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ is built based on the example patch shapes of the primary character, therefore it can be directly used to compute the residuals of vertices of the primary character patch for any new skeleton posture. The scaling matrix $\mathbf{B}$ in Equation 3.26 for the primary character is the identity matrix. However it is not the case for a target character. Although function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ can generalize over vertex parameter variables $\hat{\mathbf{p}}$ and has no problem to calculate the residual of a vertex

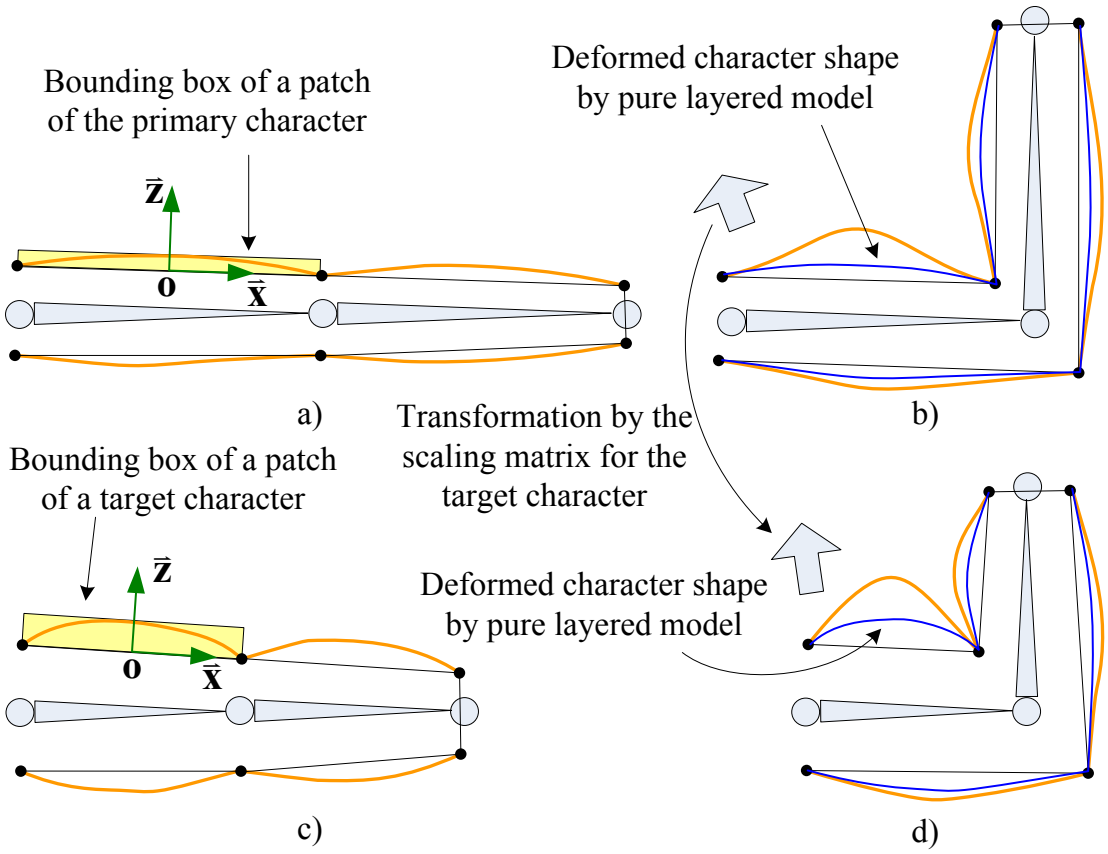

Figure 3.11: a) The dress shape of the primary character. b) The skin deformation of the primary character in a new posture. c) The dress shape of the target character. d) The skin deformation is transferred from the primary character to the target character in the same new posture.

of the target character patch based on its 2D parameters, the resultant residual is the interpolation of example residuals of the vertices of the primary character patch. The amplitude of estimated residual may not fit the target character. To apply function $\boldsymbol{\psi}(\hat{\mathbf{p}}, \boldsymbol{\xi})$ to the target character patch, the vector valued result of the function should be scaled properly as illustrated in Figure 3.11. The scaling matrix for the target character is decided by comparing the bounding boxes of the corresponding patches of the primary and target characters. The bounding box of a skin patch is estimated depending on the dress shape of the character. Suppose $\{\mathbf{o}, \vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}\}$ is the local frame associated with the BCM face, which corresponds with the skin patch, and $\mathbf{o}, \vec{\mathbf{x}}, \vec{\mathbf{y}}, \vec{\mathbf{z}}$ are the origin, x-axis, y-axis, and z-axis of the local frame respectively. The length, width and height of the bounding box are

measured as follows:

$$
\begin{aligned}
b_x &= p_x^{max} - p_x^{min}, \quad (p_x^{max}, p_x^{min}) = \Big(max, min\{p_x | p_x = \vec{\mathbf{x}}^T(\mathbf{p}_i^1 - \mathbf{o}), 0 \le i < n_p\}\Big) \\
b_y &= p_y^{max} - p_y^{min}, \quad (p_y^{max}, p_y^{min}) = \Big(max, min\{p_y | p_y = \vec{\mathbf{y}}^T(\mathbf{p}_i^1 - \mathbf{o}), 0 \le i < n_p\}\Big) \\
b_z &= p_z^{max} - p_z^{min}, \quad (p_z^{max}, p_z^{min}) = \Big(max, min\{p_z | p_z = \vec{\mathbf{z}}^T(\mathbf{p}_i^1 - \mathbf{o}), 0 \le i < n_p\}\Big)
\end{aligned}
$$

After the bounding boxes for the corresponding patches of the primary and target characters are estimated, the scaling matrix for the target character is:

$$
\mathbf{B} = \begin{pmatrix} b_x^t/b_x^p & 0 & 0 \\ 0 & b_y^t/b_y^p & 0 \\ 0 & 0 & b_z^t/b_z^p \end{pmatrix}
$$

where (b_x^p, b_y^p, b_z^p) and (b_x^t, b_y^t, b_z^t) are the lengths, widths and heights of the bounding boxes of the patches on the primary and target characters respectively.

3.7 Results

In this section, we will demonstrate the effectiveness of EBTLE by creating skin deformations for the upper bodies of a muscular man, a soldier, a superhero, a kid and a gibbon. The specifications of these five characters are listed in Table 3.1. We used these characters as experimental objects because a human upper body can exhibit common skin deformations, and its skeleton is capable of typical joint movements. These five characters have different body proportions and different mesh connectivities. The experiments are performed on a 2.8 GHz Pentium-4 workstation. The static shapes of the muscular man and the kid in the dress posture are exported from Poser 5 [31] and the other three characters are obtained from the Viewpoint premier model library [37]. As illustrated in Figure 3.12, the skins of the five characters are decomposed in the same way and there are 44 patches defined for each character. The segmentation of these characters were accomplished by specifying boundaries on its skin as described in Section 3.1, and then the corresponding feature vertices on different characters are identified. The base control meshes of these characters are automatically created as shown in

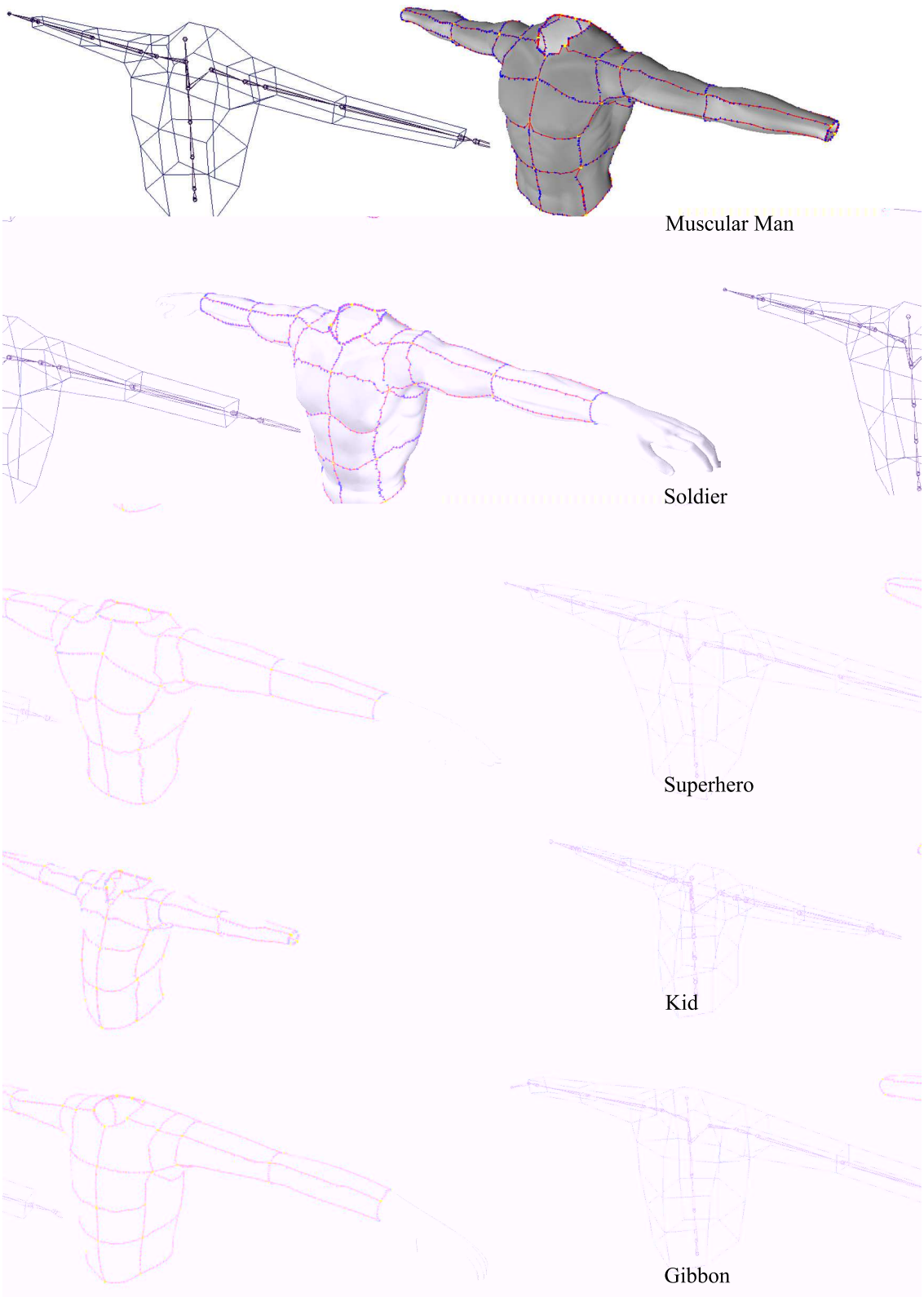

Figure 3.12: Left: The skeletons and base control meshes of the five characters. Right: The static shapes of the five characters in the dress posture and the decompositions of their skins. Note that there are no hands attached for the muscular man and the kin because this experiment doesn't include hand deformation, and for the other characters, their hands just rigidly follow the skeleton.

Character name	Muscular man	Soldier	Superhero	Kid	Gibbon
Triangle number	15660	23316	40740	11800	10514
Vertex number	7879	11713	20442	5945	5322

Table 3.1: Specifications of the experimental characters.

Figure 3.12 and they share the same connectivity. Every base control mesh has 55 vertices and 44 faces. Skeletons with identical hierarchical structure are fitted into these characters and each skeleton has 20 joints (see Figure 3.12). To generate smooth and round deformations around the shoulders, two closely placed joints (see Figure 3.12) are used for the movement of the shoulder so that each joint shares half of the rotation of the shoulder. And accordingly a thin strip area composed of 4 patches are defined around the bone that connects these two joints (see Figure 3.12). During the animation, this strip of patches can buffer the drastic differences between the patches on the arm and the patches on the torso.

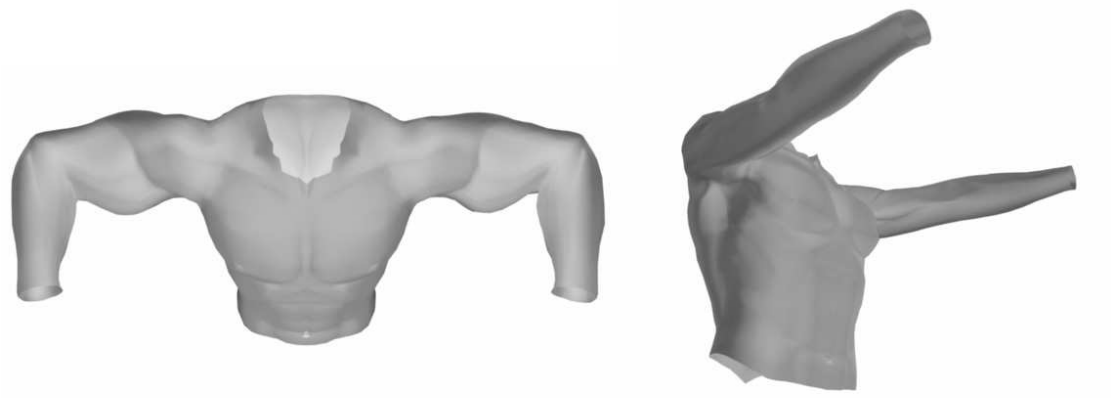

Figure 3.13: The second (left) and third (right) examples of the muscular man.

The muscular man serves as the primary character. Besides its first example (the dress shape in the dress posture), the second and the third examples of the muscular man are given in Figure 3.13. We modeled these two example shapes in Maya 4.5 [2] based on the dress shape of the muscular man. Various tools of Maya (like rigid binding, FFD lattice) were used to modify the dress shape into the example shapes. The benefit of decomposing the skin of a character into patches is that the number of examples required by one patch to achieve satisfactory results is quite small. Moreover, since our pure layered skinning model is already enough

	Influencing examples	Influencing joint angles	Number of RBFs	Coefficients solving (s)	Evaluation time (s)
Patch 1 on left upper arm	1,2	left elbow flexion angle	210	0.032	1.46E-04
Patch 2 on left upper arm	1,2	left elbow flexion angle	278	0.079	1.99E-04
Patch 3 on left upper arm	1,2	left elbow flexion angle	214	0.047	1.41E-04
Patch 4 on left upper arm	1,2	left elbow flexion angle	304	0.094	2.29E-04
Patch on left front chest	1,3	first left shoulder flexion angle	322	0.093	2.28E-04

Table 3.2: Statistics for the patches whose deformations are adjusted by examples.

to capture nice global deformations for a character, only the patches in highly deformable areas need examples to guide their behavior. More specifically, the second example (in the left of Figure 3.13) is only used by the patches on the upper arms to simulate the bulging of biceps, and the third example (in the right of Figure 3.13) is only used by the patches on the chest to create the rising effects of the chest as the arms fold in. The deformations of the patches on other regions do not depend on the second or the third example and there is no RBF network associated with them.

For every patch on the upper arms and the chest, there is a RBF network to make its deformations follow the examples. We have listed the influencing joint angles, the influencing examples, the configuration of the RBF networks and the time required to evaluate them for these patches in Table 3.2. Note that the time needed for solving the weights of RBF networks is done off-line and doesn't occupy runtime budget. Synthesizing skin deformations patch by patch simplifies the problem and make it possible that all the patches only need one influencing joint angle, although the properties of radial basis functions guarantee that multiple influencing joint angles can also be handled well.

The example shapes of the BCM of the muscular man are directly generated from its examples. The first rows of Figures 3.14 and 3.16 show its example shapes

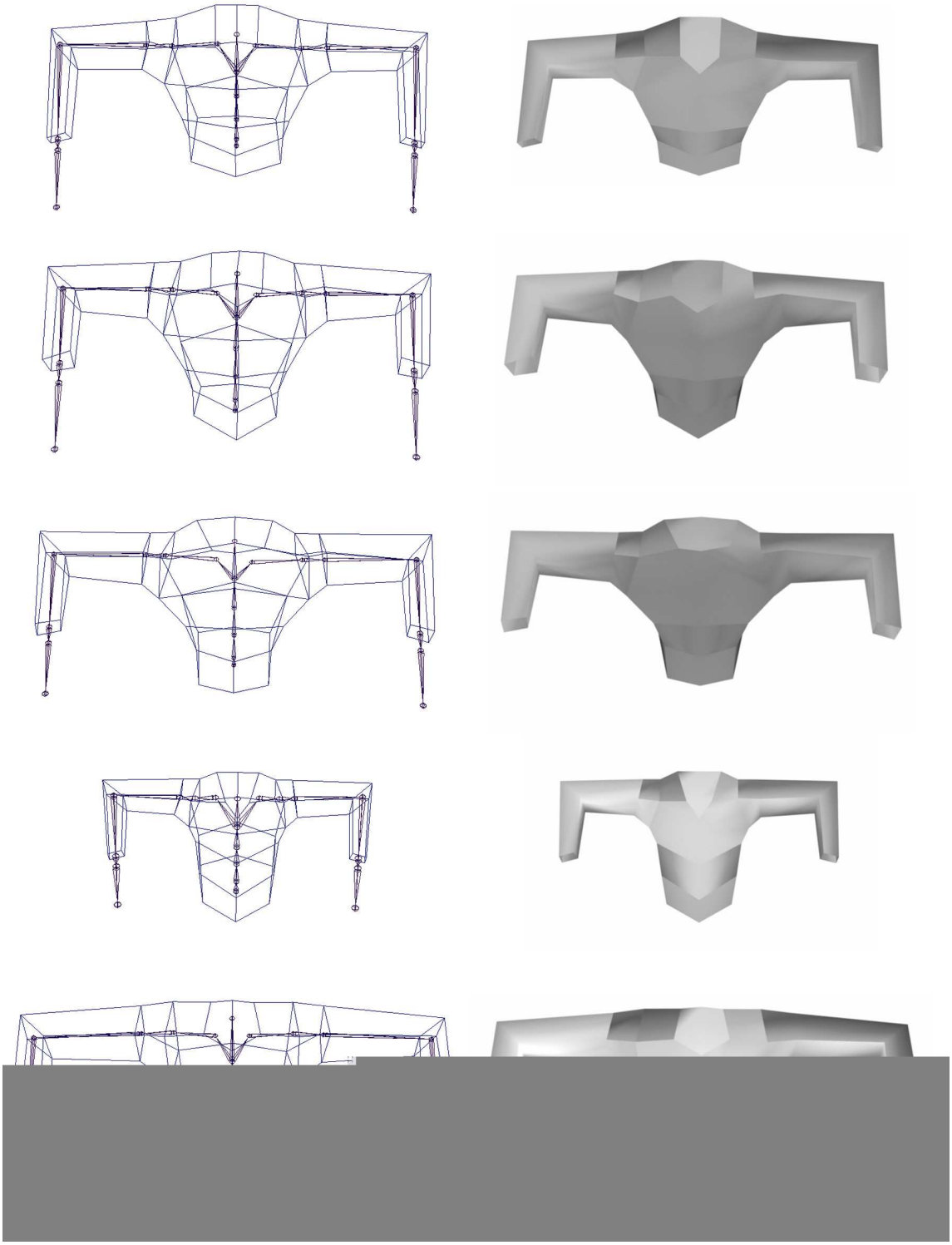

Figure 3.14: The shapes of base control meshes of the five characters in the second example posture, wireframe (left) and shaded (right).

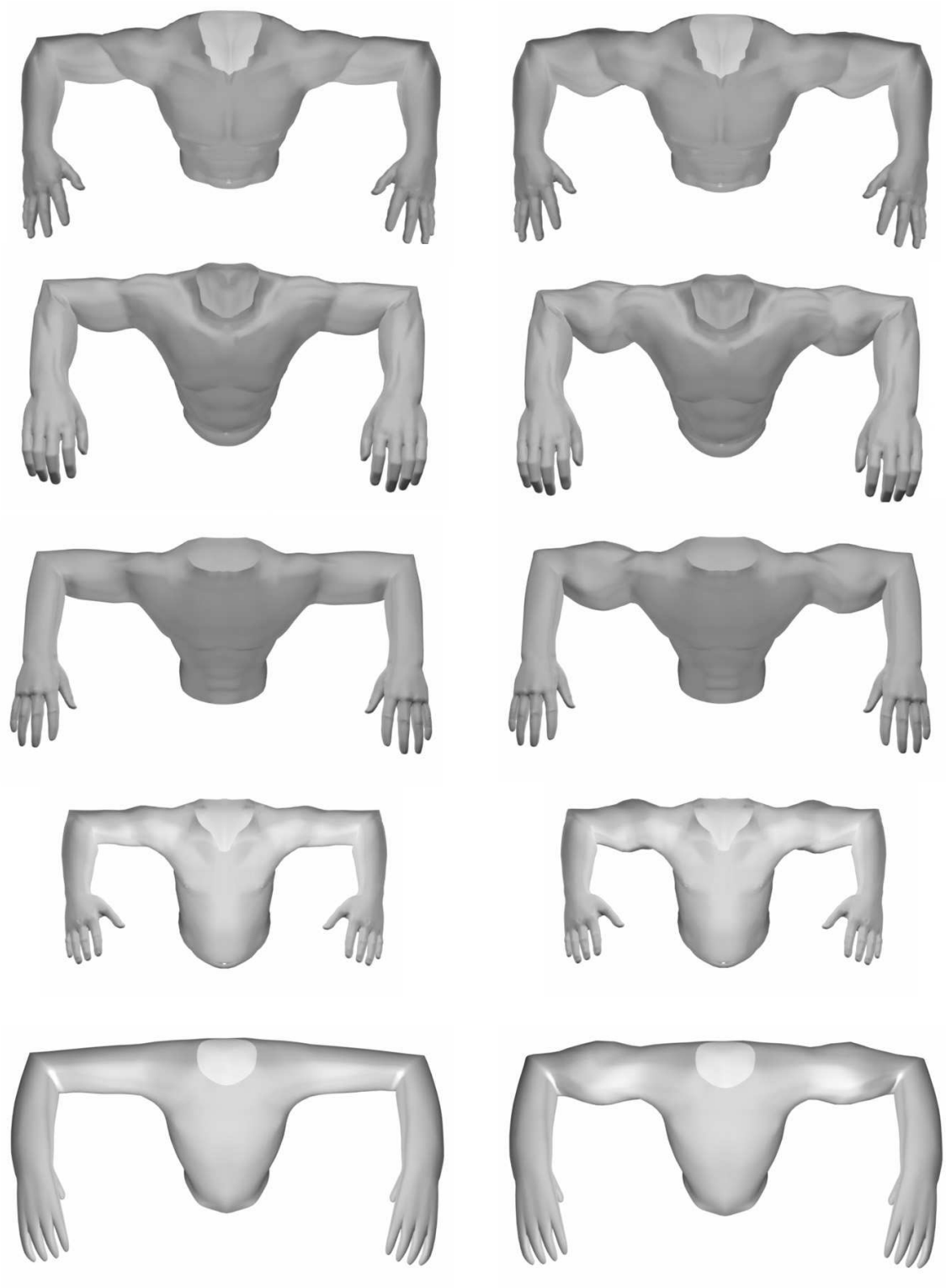

Figure 3.15: Deformed shapes of the five characters synthesized by the pure layered skinning model (left column) and with the help of the RBF networks (right column) in the second example posture. The bulge on the upper arm is due to the RBF networks.

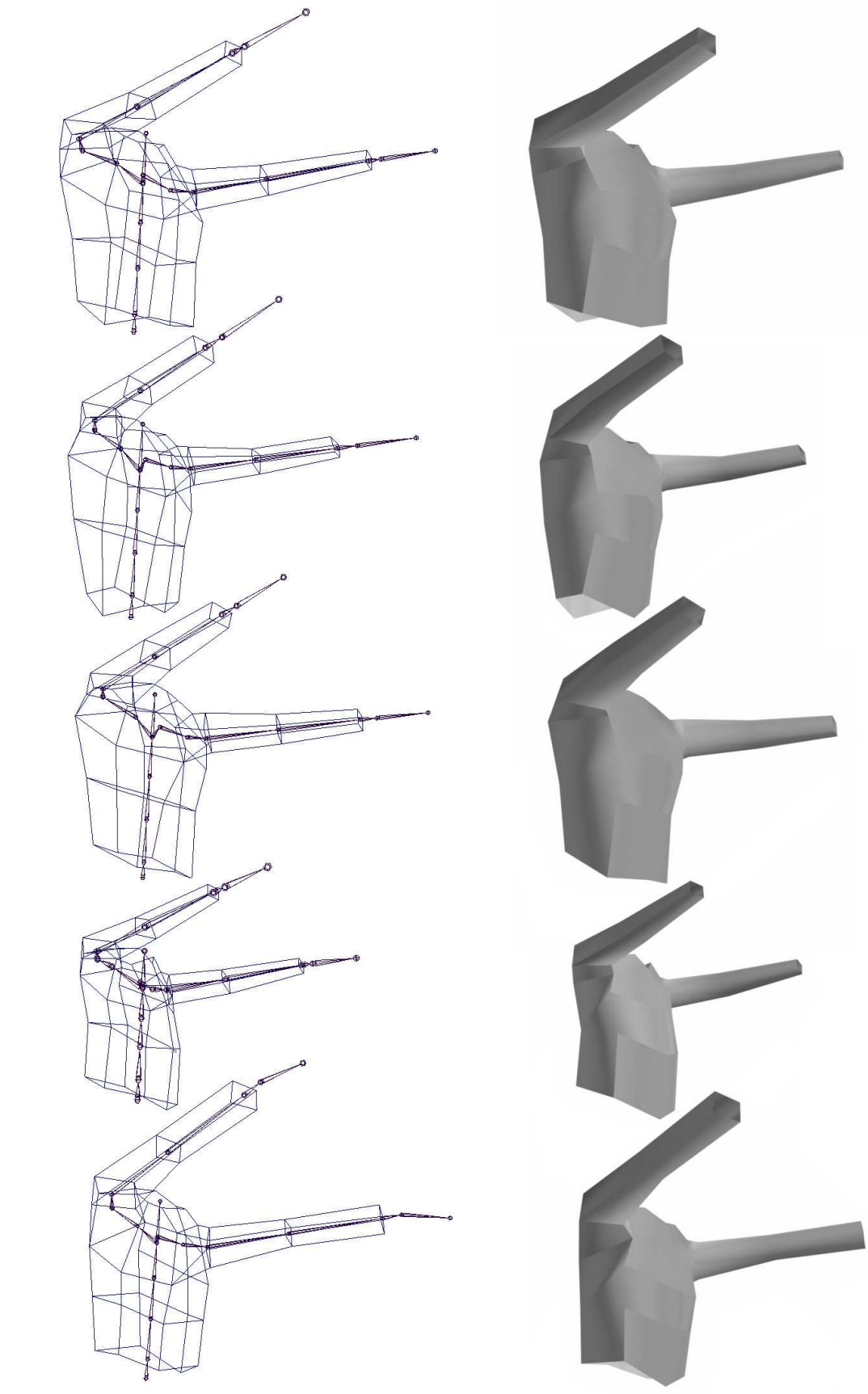

Figure 3.16: The shapes of base control meshes of the five characters in the third example posture, wireframe (left) and shaded (right).

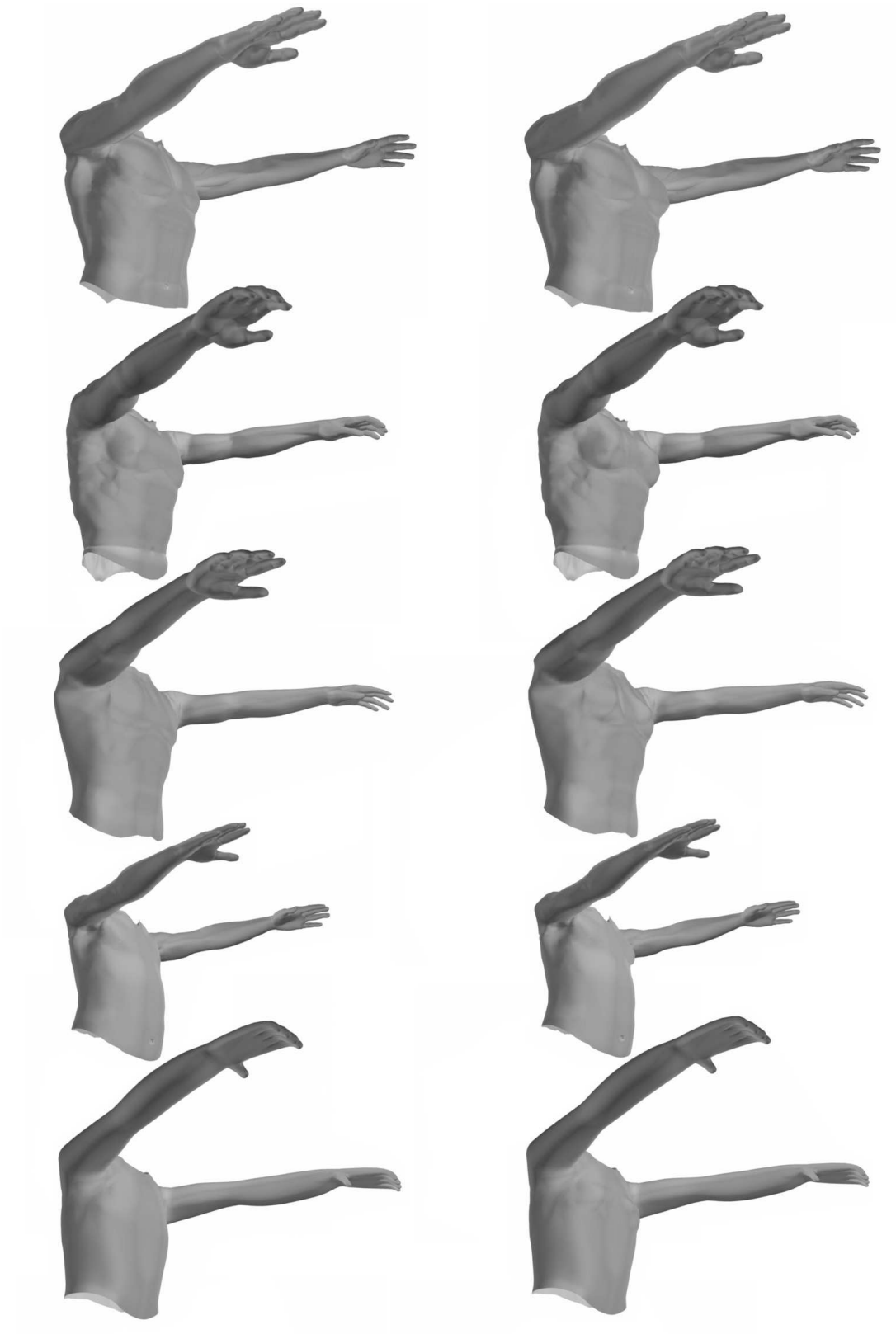

Figure 3.17: Deformed shapes of the five characters synthesized by the pure layered skinning model (left column) and with the help of the RBF networks (right column) in the third example posture. The chests rise due to the RBF networks.

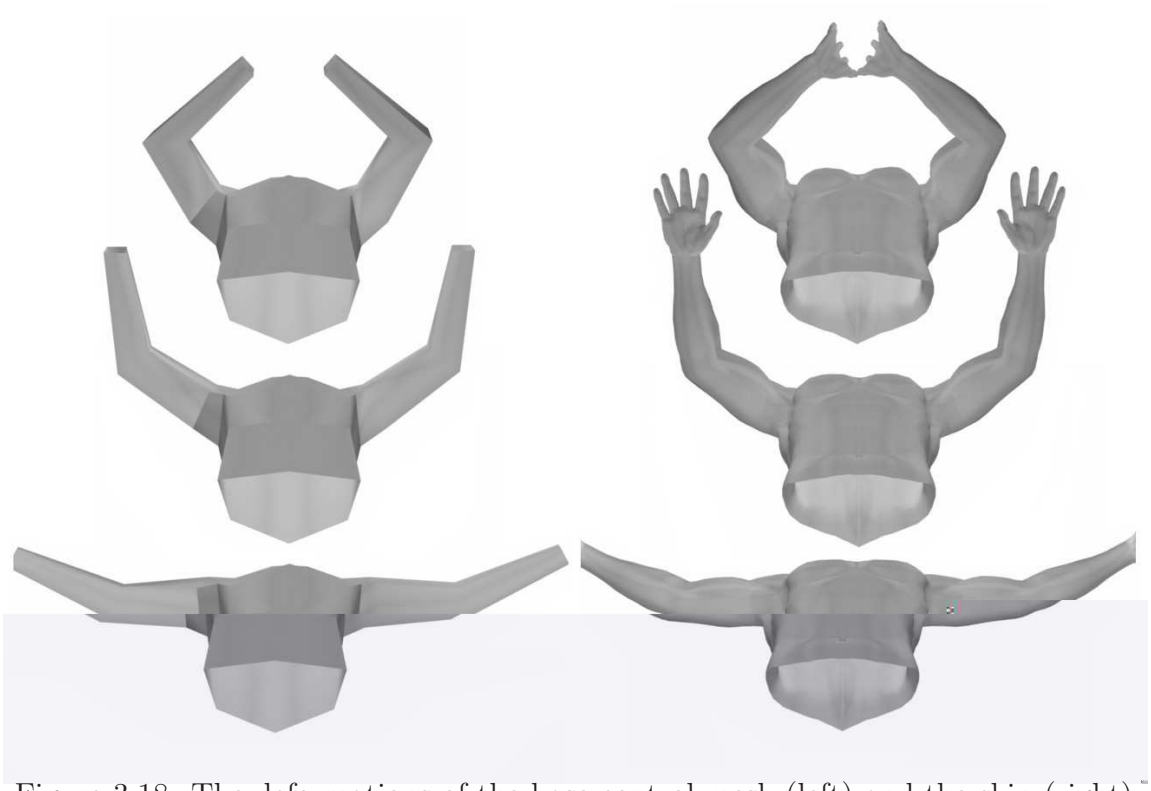

Figure 3.18: The deformations of the base control mesh (left) and the skin (right) of the muscular man in the postures that are obtained by interpolating example postures.

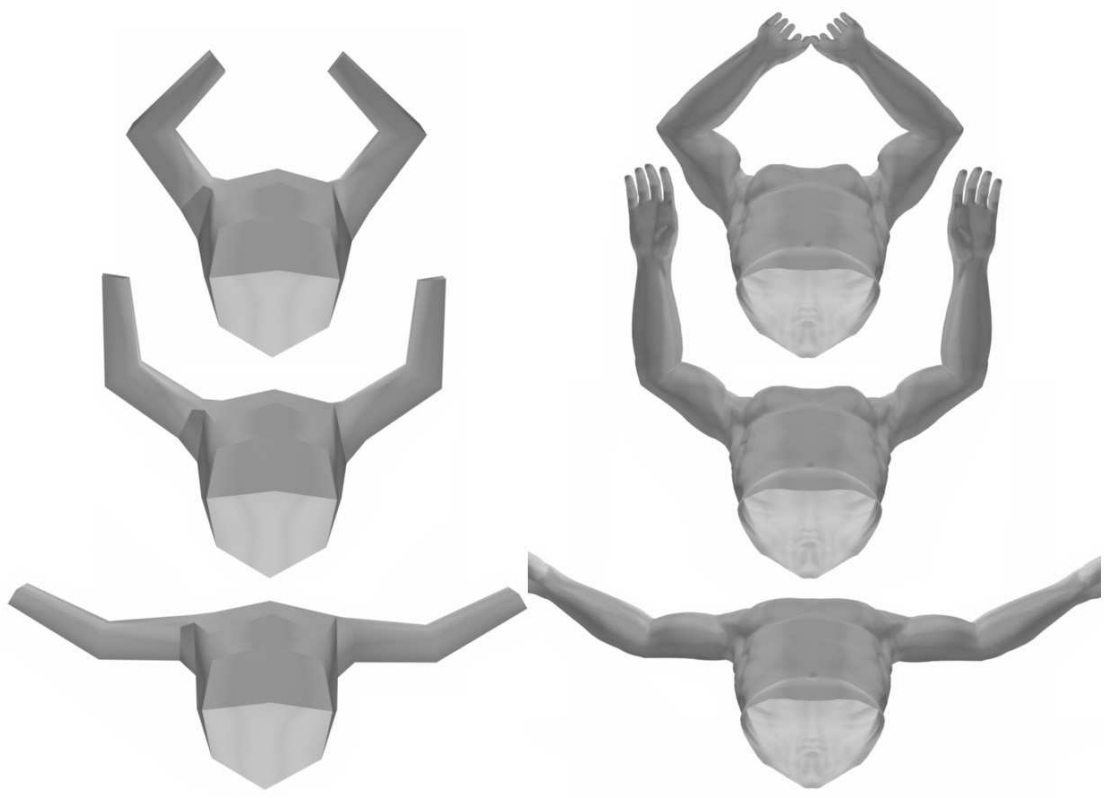

Figure 3.19: The deformations of the base control mesh (left) and the skin (right) of the soldier in the postures that are obtained by interpolating example postures.

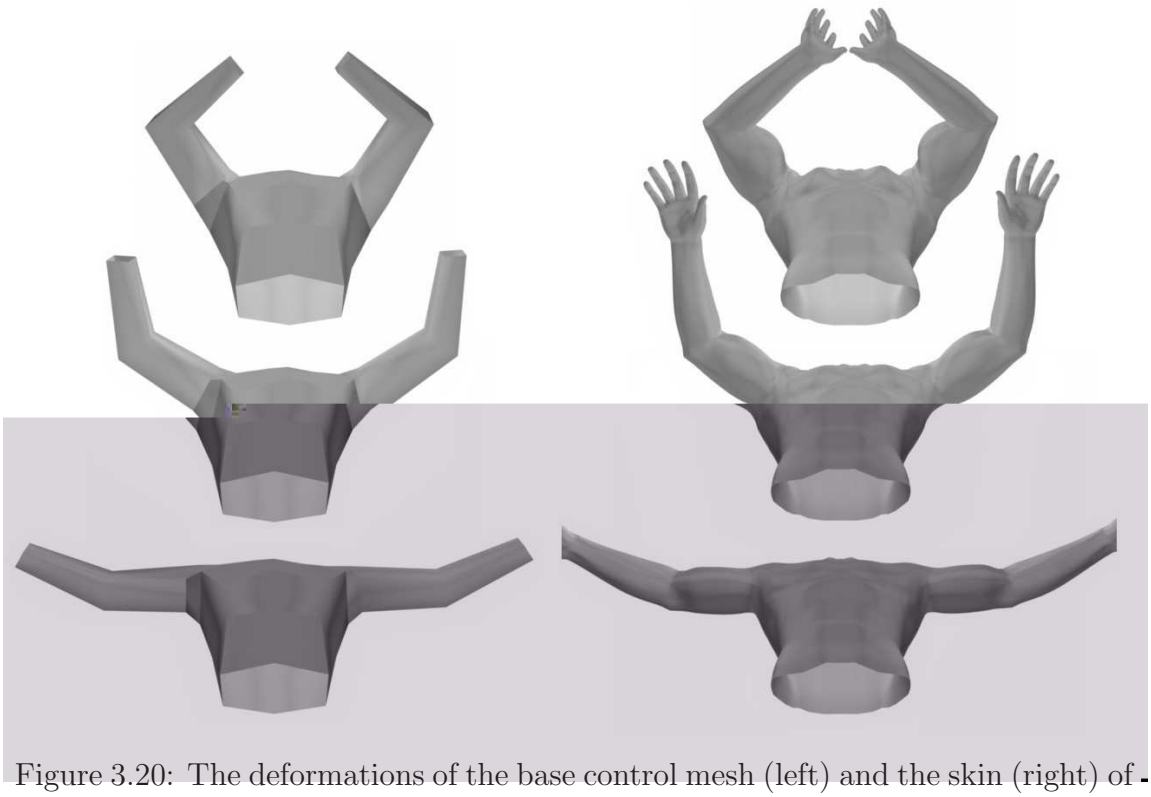

Figure 3.20: The deformations of the base control mesh (left) and the skin (right) of the superhero in the postures that are obtained by interpolating example postures.

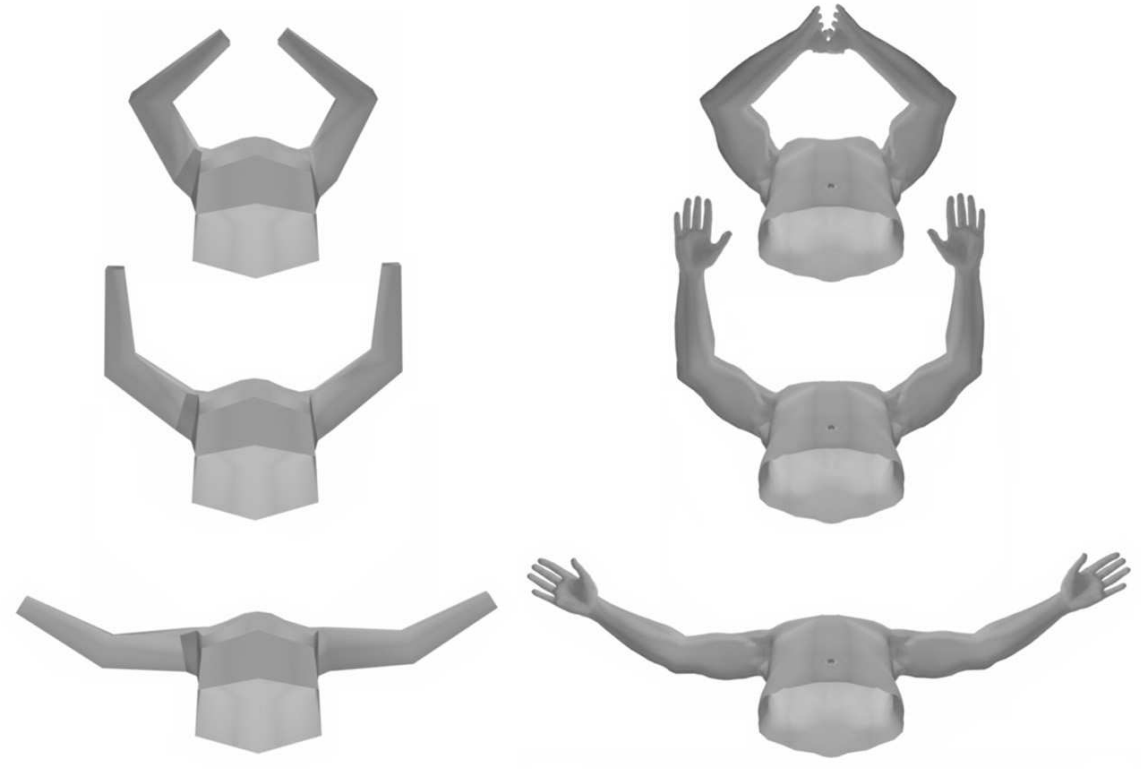

Figure 3.21: The deformations of the base control mesh (left) and the skin (right) of the kid in the postures that are obtained by interpolating example postures.

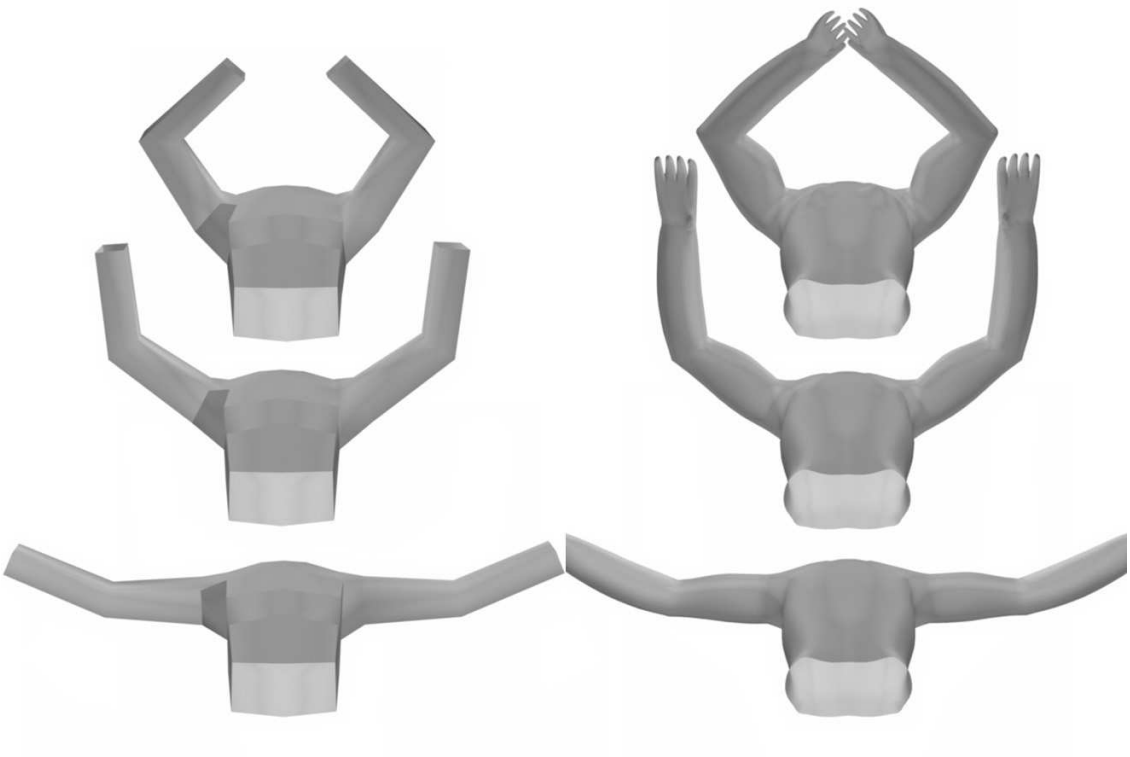

Figure 3.22: The deformations of the base control mesh (left) and the skin (right) of the monkey in the postures that are obtained by interpolating example postures.

in the second and third example postures. The second to fifth rows of Figures 3.14 and 3.16 exhibit the example shapes of the BCMs for the other four characters. They are derived using the technique described in Section 3.4.2. Nevertheless, we slightly adjusted the automatically generated shapes by hand as described in Section 3.4.2 according to our aesthetical preference. The left columns of Figures 3.15 and 3.17 display the deformed shapes of the five characters synthesized by the pure layered skinning model in the second and third example postures respectively. The skins of the characters remain seamless due to our patch stitching scheme. However, no bulging effects can be created by the pure layered skinning model.

As illustrated in the top right corners of Figures 3.15 and 3.17, with the construction of the RBF networks for the patches on the upper arms and the chest, our framework faithfully created the bulging of biceps and the rising of the chest for the muscular man (compared with the examples shapes in Figure 3.13). The same sets of RBF networks are applied for the skin deformations of the target characters and the results are given in the right columns of Figures 3.15 and 3.17. As we can see,

the bulging of the biceps and the rising of the chest are properly transferred from the muscular man to the other four characters. The scaling factors are estimated by means described in Section 3.6.3. We manually adjusted the scaling factors for the superhero and the monkey to generate more prominent bulging effects on the chest.

Our framework not only performs well in the example postures but also can produce good interpolation between the examples. Figures 3.18 to 3.22 demonstrate the skin deformations of the five characters in the postures generated by blending the example postures. The areas around the elbows show reasonable deformations due to the SSD correction trick (Section 3.4.1) and the contribution of the second example shapes of the BCMs. The upper arms of the characters swell as the elbows flex and the chests are inflated as the arms fold in. As shown in Figures 3.23 to 3.26, by articulating the skeletons of the five characters, our system can obtain the deformed shapes of these characters in postures that are never seen in the set of example postures. (For more results of animation videos, see the attached CD.)

The cost of setting up a character with EBTLE is to sculpt the example shapes for the primary character and decompose the skin. As we have demonstrated above, the number of examples needed has been dramatically reduced when the "patch" based strategy is applied. For the decomposition of the skin, since it still requires manual intervention, the time needed to finish it varies from one hour to half a day, which depends on the quality of the skin mesh of the character. When the skin mesh is well modeled, the boundaries can be defined mainly by the shortest path technique, which is pretty convenient. Although the curve projection technique is intuitive to use, it requires users to manipulate a space curve, and so it needs more labor than specifying two vertices on the skin as the shortest path technique requires. In our experiments, among the 101 boundaries of each character, only 0-8 boundaries need the curve projection technique.

The runtime budget of EBTLE for synthesizing deformed shapes is relatively low and it could be used for interactive applications. We have listed the time needed

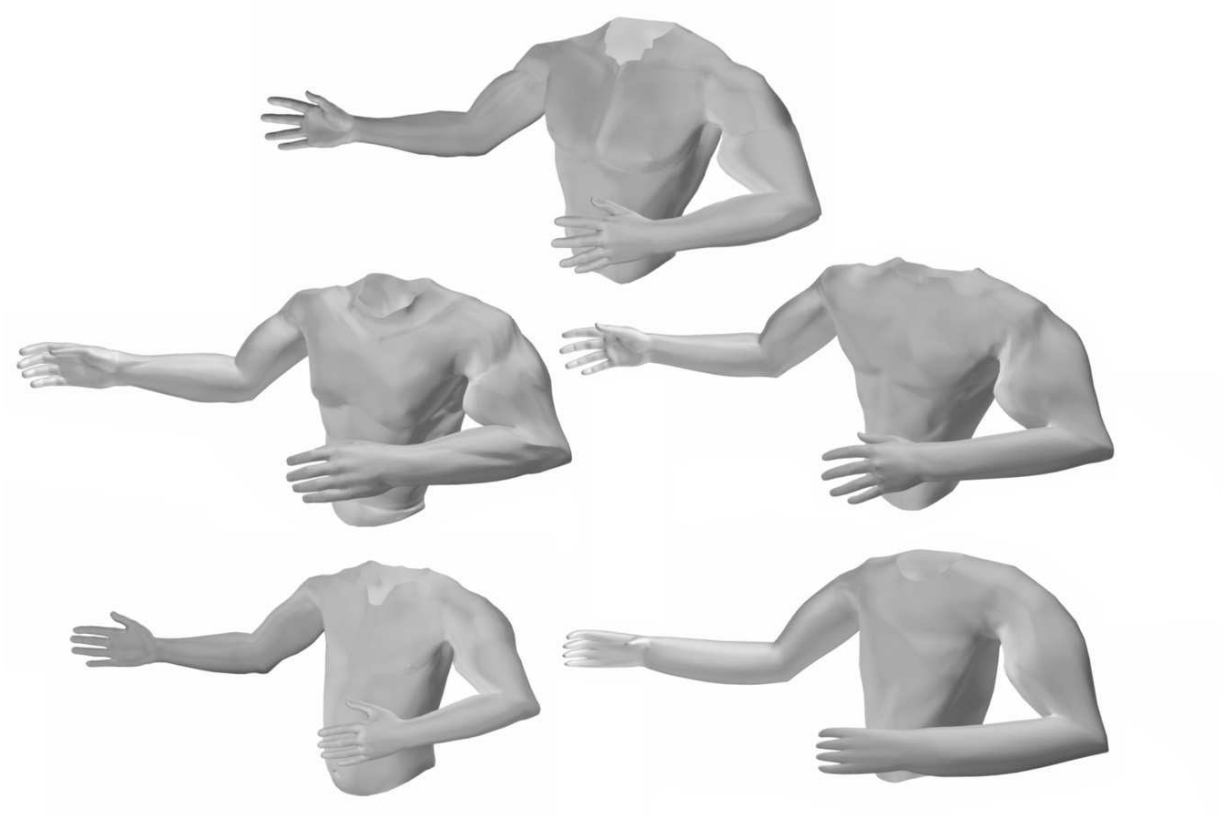

Figure 3.23: The skin deformations of the five characters in a new posture.

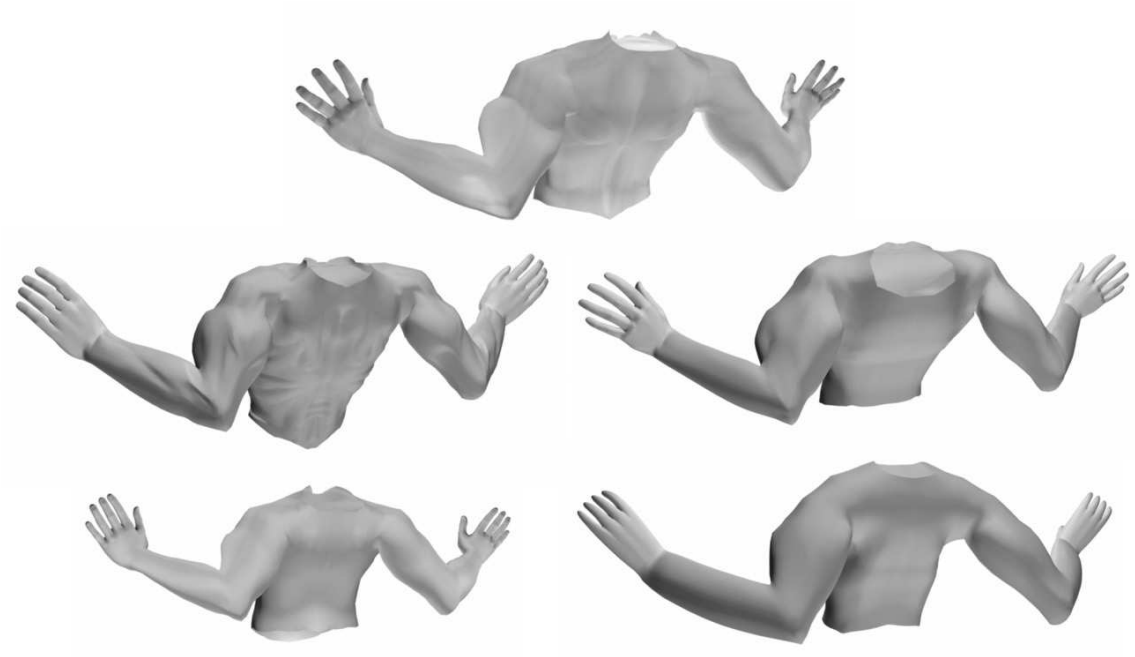

Figure 3.24: The skin deformations of the five characters in a new posture.

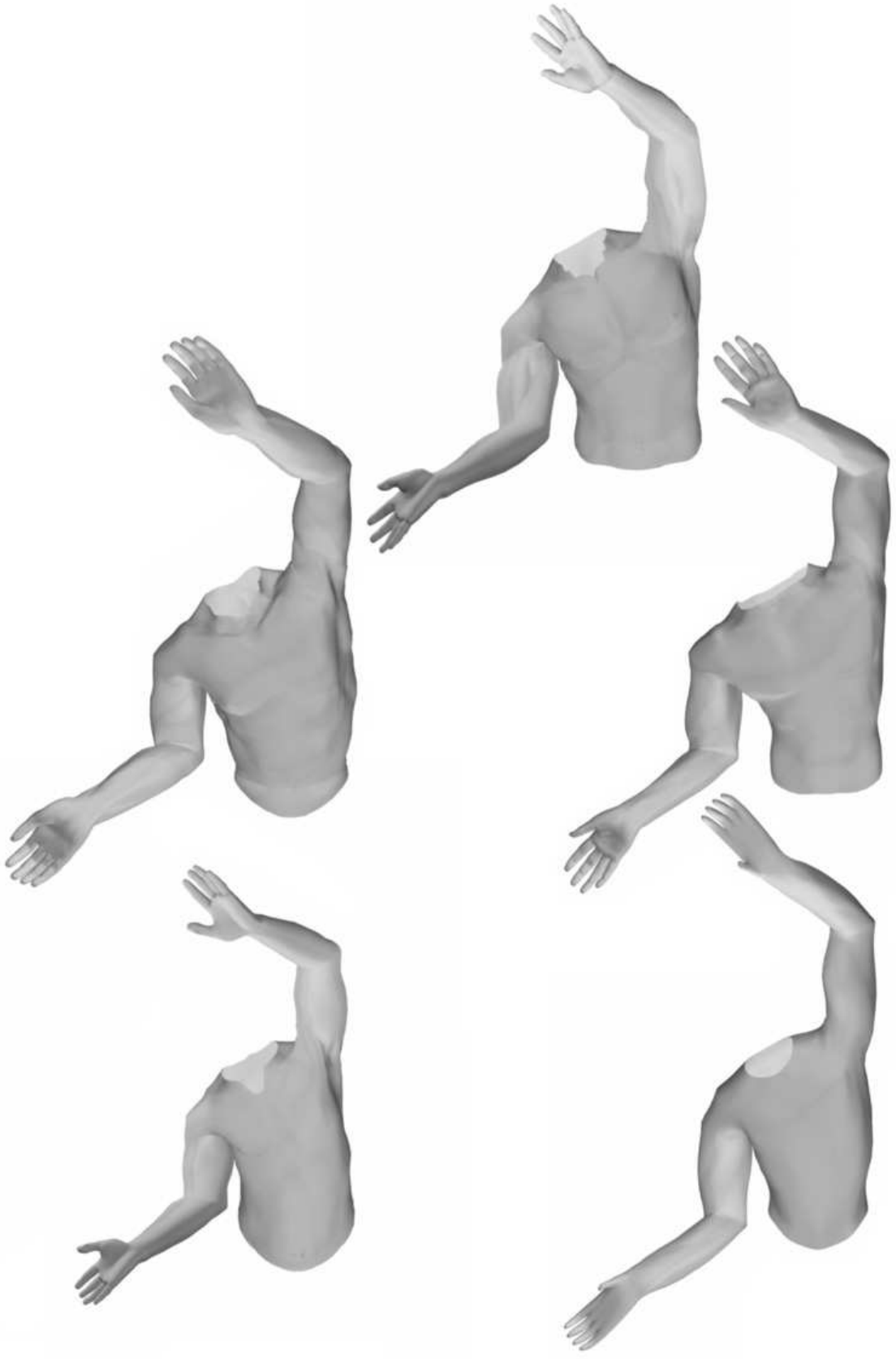

Figure 3.25: The skin deformations of the five characters in a new posture.

Figure 3.26: The skin deformations of the five characters in a new posture.

by different stages of synthesis for all five characters in Table 3.3. It includes the time for the deformation of the base control mesh, the transformations of the patches by the base control mesh and patch stitching, and the evaluation of RBF networks. The time for the RBF networks in Table 3.3 is the time recorded when all the ten RBF networks (four for the left upper arm, four for the right upper arm, two for the chest) are activated. However the RBF network for a patch will not be evaluated if its influencing joint angle does not change. Thus, in practice, during animation, not all the RBF networks are evaluated at the same time. Therefore the time cost of RBF networks could be less.

Character name	Deformation of BCM (s)	Transformation & patch stitching (s)	BRF networks (s)	Total time (s)
Muscular man	0.016	0.094	0.278	0.388
Soldier	0.016	0.172	0.353	0.541
Superhero	0.016	0.218	0.546	0.78
Kid	0.016	0.051	0.281	0.348
Gibbon	0.016	0.062	0.216	0.294

Table 3.3: The time cost of different stages of runtime synthesis.

The method does not always work well. As described in Section 3.1∼3.6, EBTLE needs a set up process before it can synthesize character shape at runtime. This set up process requires certain manual work from users. The quality of the final skin deformation depends on the effort made in the set up process. Without proper understanding the purpose of steps in the set up process, it is not unusual to get some bad deformation results.

The performance of the system in new skeleton posture partially relies on the examples supplied to the system. It can be deemed as a common criterion that the examples influencing a certain patch should be evenly scattered in the pose space, instead of clustering together and leaving a large region of the pose space empty. That is, if a bulge is supposed to be formed on a patch when some joints rotate in a certain range, the rotation angles of the joints in the example postures should cover that range evenly. Otherwise, if the examples are concentrated in a small region, then at runtime, when those joints rotate beyond this small region, the deformation may be unexpected. For example, if we want to create a bulge on a patch when a joint rotates from 0 to 100 degrees, and two examples are provided, one example with the rotation angle set to 0 degree and no bulge on the skin, and the other example with the rotation angle set to 40 degrees and half size of the desired bulge on the skin, then at runtime, when the joint rotates beyond 40 degrees, it is possible that instead of becoming bigger as expected, the bulge will remain the same size or become smaller. It is important that the examples should include extreme situations. The behavior of system for extrapolation may not be desired by users.

Another criterion for good quality of examples is to avoid using confusing examples. Examples in similar skeleton postures but with different deformations on the skin will confuse the RBF interpolation algorithm, and at runtime, some oscillation may be observed. Let's continue the previous case about the bulge on the patch. If we add the third example with rotation angle set to 42 degrees and a quarter size of the desired bulge, this example will be in conflict with the second example (rotation angle set to 40 degrees and half size of the desired bulge). At runtime, when the rotation angle is near 40 degrees, the bulge on the skin will have a sudden change of size.

Decomposition of the skin also plays a significant role in generating natural skin deformation. However, currently there is no way to computationally measure the quality of the decomposition. It does require some experience and "trial and error" to obtain a suitable decomposition of the skin. Simple anatomy knowledge can be used as the guide for the specification of patches. The patch structure that conforms to real musculature tends to give better results. The requirements of deformation algorithm also need to be taken into consideration during the decomposition procedure. The decomposition of the skin determines the shape of the BCM. So the decomposition process should also try to make the resultant BCM have an optimum mesh connectivity for animation. Decomposition with too few patches will make the BCM have too few vertices, and it will not be flexible enough to create good deformation. For complicated area of the skin, it makes sense to use more patches so that the BCM has a more flexible structure to handle the deformations. For example, as shown in Figure 3.12, around the shoulder area, a strip of four patches are used to deal with the deformations. Without these four patches, it is easy to see collapsed shoulder and discontinuities. As a general rule, the decomposition should not use too few patches to cover a complicated area and should use more feature vertices around troublesome joints (like shoulder, elbow joint). During the decomposition process, users use boundaries to specify patches. Boundaries are responsible for connecting the patches with each other. Based on

our experience, using relatively smooth boundaries during the decomposition has a better chance of avoiding creases or protruded vertices.

3.8 Summary

We have proposed a novel example based transferable layered skinning model in this chapter. A manual skin decomposition technique is developed to allow users to dissect the skin mesh into patches freely. According to the structure of the patches, the base control mesh is automatically generated to serve as a middle control layer between the skeleton and the skin mesh. A layered skinning model is naturally built so that the skin shape is controlled by the skeleton through the base control mesh. Radial basis function interpolation technique is seamlessly combined with the layered skinning model to adjust the deformations of both the base control mesh and the skin. The RBF interpolation for the deformation of a patch shape is constructed in the hyper space of skeletal parameters augmented with skin vertex parameters. Thus, it gives a uniform representation of skin deformations and has no problem to be applied to a target character as long as the target character is decomposed in the same way as the primary character. Using a set of examples of an individual character to control the skin deformations of a group of geometrically similar characters is a powerful technique that considerably reduces the manual labor since artists only need to provide one set of examples for the primary character instead of one set of examples for every character. At runtime, the skin deformations of both the primary and target characters are under the guidance of the examples of the primary character.

The ultimate goal of many topics in computer animation is to make the human eyes believe the computer generated image is real and has no difference from the image captured using real cameras. For skin deformations, our criterion is also similar, and that is to make the generated deformations look natural and aesthetically satisfactory. However, in terms of aesthetic judgment, it may not have a unified

point of view. Different artists probably have different standards. The artistic sense is also influenced by the time and culture. In the case of realistic skin deformations, it is natural that certain standards can not be measured quantitatively. Our framework tries to automatically maintain these standards by avoiding obvious or common shortcomings. But to adjust these standards according to the personal preference of individual artists, some manual adjustments will be necessary. Like most of the geometrical deformation methods, EBTLE requires users to adjust some parameters to achieve desired results. The manual tweaking in various stages of our framework gives users different level of controlling deformation results. These adjustments can be done easily and intuitively. Without manual tweaking, the automatic processes may create unwanted deformation results. However, through proper adjustment of parameters, the system can produce good results.

Although EBTLE is capable of creating good deformation results, there are still a few limitations. The proper decomposition of the skin is the first stage of generating natural skin deformation. Currently, this decomposition process still requires manual intervention. The quality of decomposition depends on the experience of artists, and "trial and error" has to be used to obtain a suitable decomposition. In order to achieve desired deformations, users may need to spend some time on adjusting the boundaries and structure of the patches. Another limitation resulting from the decomposition is that the deformation transfer can only be applied to similar characters. EBTLE demands the decomposition of the target character has to be the same with the decomposition of the primary character. For two very different characters, it will be difficult to keep their patch structures the same, so deformation transfer between them will be problematic. The stitching algorithm only maintains the first order of continuity, and creases may occur when the transformations applied to the neighboring patches are too large. When an unexpected crease is detected, users need to change the structure of skin patches, and in this way, it makes the duty of the decomposition process even heavier. As an example based skin deformation method, EBTLE requires users to provide example shapes

for the primary character. This requirement is regarded as both advantage and disadvantage. The advantage is that the users can directly specify the desired deformation using examples. The disadvantage is that artists need to make more effort to model the shapes of the character in other postures. Since ambiguous examples can cause unexpected results for EBTLE, users should be careful about the examples provided to the system. The computational cost of the RBF interpolation technique is proportional to the number of examples. Most of the time, EBTLE does not require a large number of examples. However, if in some particular case, the user decides to use a lot of examples, the speed of the system may not be able to fulfill the requirement of interactive applications.

Chapter 4

Skinning With Deformable Chunks

Anatomically based methods can produce impressive skin deformations by simulating underlying anatomical entities. Their main drawback is that the construction of the anatomical structure of a character demands users to model every individual muscle. In order to make this process automatic, the algorithm needs to analyze the shape and species of the character, and to determine the position, orientation, and shape of every muscle based on anatomy knowledge. To the best of our knowledge, such a system has not been developed yet. Simmons et al. [107] proposed a method that can fit the canonical anatomical structure of the horse into individual horses. However, this kind of methods can only be applied for the same species. It will be impossible to fit the canonical horse to a human being. In computer graphics, it is not necessary for the simulation to exactly follow the rules applied in the real world. As long as the skin of the character deforms naturally, it doesn't matter what kind of anatomical entities are used. Instead of standard muscles from real

world anatomy, we decide to use an alternative entity that makes the automatic generation process easier to implement.

Based on the decomposition of the skin of a character, Skinning With Deformable Chunks (SWDC) can automatically generate internal entities beneath the skin. A new type of entity, *deformable chunk*, is introduced to create deformations on the skin. A quasi-static deformation model with finite element method is employed to calculate the deformations of the chunks according to the posture of the skeleton. The deformation of the skin is then intuitively controlled by the deformations of the chunks.

The remainder of this chapter is organized as follows. In Section 4.1, the theory of elasticity is briefly discussed. Finite element method is explained in Section 4.2. Sections 4.3 to 4.8 describe SWDC in detail. Some experimental results are given in Section 4.9.

4.1 Elasticity Theory

To generate realistic deformations, it is necessary to study the theory of elasticity. In the real world, all materials deform, more or less, when there are external forces applied on them. Elasticity theory [102] provides the mathematical description of the deformation distribution in an elastic solid under the influence of external forces. It has been applied extensively in many engineering and scientific fields.

4.1.1 Strain Tensor

Continuum mechanics treats a solid object Ω as the composition of infinite points (see Figure 4.1). We start the development of elasticity theory by first investigating the kinematics of deformations. As shown in Figure 4.1, $\mathbf{x}$ is used to denote the position of a point in the original object and also used as the label of the point. Under the influence of external forces, every point in the continuum material may deviate from its former position. The notation $\mathbf{w}$ denotes the deformed position of point $\mathbf{x}$. $\mathbf{w}$ is a function of $\mathbf{x}$, and the gradient of $\mathbf{w}$ with respect to $\mathbf{x}$ can be

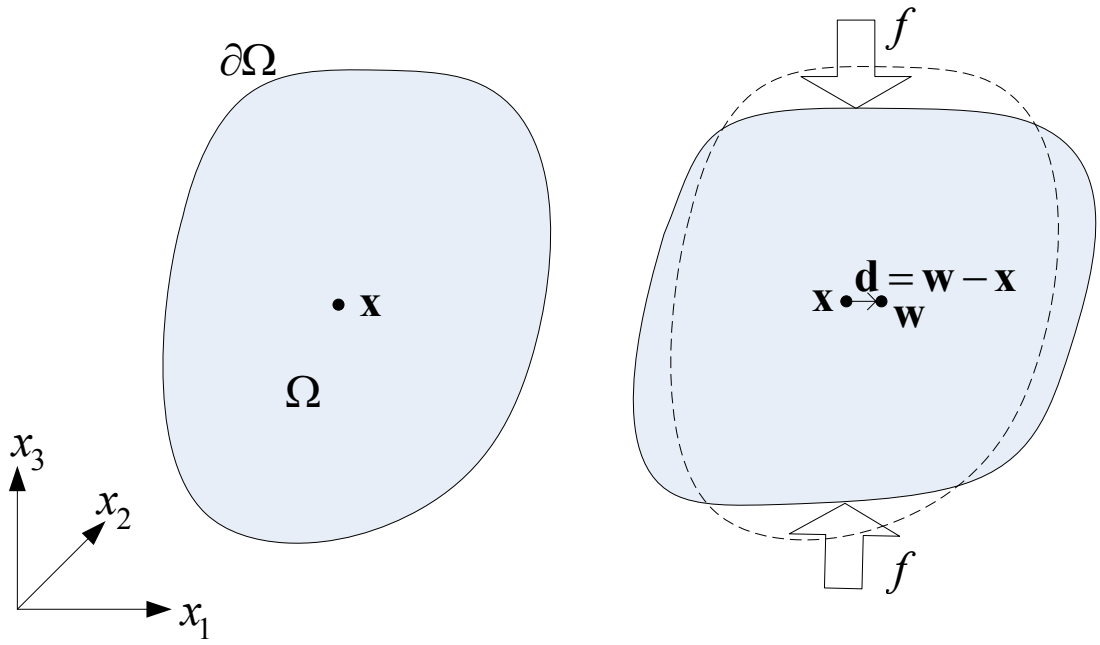

Figure 4.1: A solid object and its deformation.

written as:

$$\mathbf{F} = \frac{d\mathbf{w}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial w_1}{\partial x_1} & \frac{\partial w_1}{\partial x_2} & \frac{\partial w_1}{\partial x_3} \\ \frac{\partial w_2}{\partial x_1} & \frac{\partial w_2}{\partial x_2} & \frac{\partial w_2}{\partial x_3} \\ \frac{\partial w_3}{\partial x_1} & \frac{\partial w_3}{\partial x_2} & \frac{\partial w_3}{\partial x_3} \end{pmatrix}. \tag{4.1}$$

It is straightforward to measure the difference between the deformed and original shapes of the object using the displacement $\mathbf{d}$ between $\mathbf{w}$ and $\mathbf{x}$.

$$\mathbf{d} = \mathbf{w} - \mathbf{x} \tag{4.2}$$

However, the displacement $\mathbf{d}$ is not a good criterion because it only records the the spatial change in the Cartesian coordinate system. Even rigid translation or rotation can cause the alteration of $\mathbf{d}$ without deforming the object. The quantity describing deformation must be intrinsic to the object and be invariant to the choice of the coordinate system. For two different elementary vectors $d\mathbf{x}_1$ and $d\mathbf{x}_2$ in the original object shape, suppose their corresponding elementary vectors in the deformed object shape are $d\mathbf{w}_1$ and $d\mathbf{w}_2$. If the deformed shape of the object is the same as the original shape, then the angle between $d\mathbf{w}_1$ and $d\mathbf{w}_2$ will be the identical to the angle between $d\mathbf{x}_1$ and $d\mathbf{x}_2$, and the lengths of $d\mathbf{w}_1$ and $d\mathbf{w}_2$ will equal the lengths of $d\mathbf{x}_1$ and $d\mathbf{x}_2$. The inner product of two elementary vectors can

be employed to measure the difference between the deformed and original shapes. The difference between $d\mathbf{w}_1^T d\mathbf{w}_2$ and $d\mathbf{x}_1^T d\mathbf{x}_2$ is:

$$\begin{aligned}\frac{1}{2}\left(d\mathbf{w}_1^T d\mathbf{w}_2 - d\mathbf{x}_1^T d\mathbf{x}_2\right) &= \frac{1}{2}\left(d\mathbf{x}_1^T \mathbf{F}^T \mathbf{F} d\mathbf{x}_2 - d\mathbf{x}_1^T d\mathbf{x}_2\right) \\ &= d\mathbf{x}_1^T \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) d\mathbf{x}_2 \\ &= d\mathbf{x}_1^T [\boldsymbol{\varepsilon}] d\mathbf{x}_2\end{aligned}$$

where

$$[\boldsymbol{\varepsilon}] = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right), \tag{4.3}$$

and $\mathbf{F}$ is the gradient of $\mathbf{w}$ with respect to $\mathbf{x}$.

The 3×3 symmetric matrix, $[\boldsymbol{\varepsilon}]$, is termed *strain tensor*. It contains deformation information independent of the rigid body translation and rotation, so it is suitable for describing the deformation distribution of a solid object. The formulation of $[\boldsymbol{\varepsilon}]$ in Equation 4.3 is called Green strain tensor. Writing Equation 4.3 in element form gives:

$$\varepsilon_{ij} = \frac{1}{2}\left(\left(\frac{\partial \mathbf{w}}{\partial x_i}\right)^T \frac{\partial \mathbf{w}}{\partial x_j} - \delta_{ij}\right), \tag{4.4}$$

where δ_{ij} is Kronicker delta. (It is equal to one if $i = j$, otherwise it is zero.) In Equation 4.4, ε_{ij} is formulated using the deformed position $\mathbf{w}$ of point $\mathbf{x}$. It is intended to relate the strain tensor to the displacement $\mathbf{d}$. According to Equation 4.2,

$$\mathbf{w} = \mathbf{x} + \mathbf{d}.$$

Substitute the above equation into Equation 4.4, and ε_{ij} becomes:

$$\begin{aligned}\varepsilon_{ij} &= \frac{1}{2}\left(\left(\frac{\partial(\mathbf{x}+\mathbf{d})}{\partial x_i}\right)^T \frac{\partial(\mathbf{x}+\mathbf{d})}{\partial x_j} - \delta_{ij}\right) \\ &= \frac{1}{2}\left(\left(\mathbf{e}_i + \frac{\partial \mathbf{d}}{\partial x_i}\right)^T \left(\mathbf{e}_j + \frac{\partial \mathbf{d}}{\partial x_j}\right) - \delta_{ij}\right) \\ &= \frac{1}{2}\left(\frac{\partial d_i}{\partial x_j} + \frac{\partial d_j}{\partial x_i} + \left(\frac{\partial \mathbf{d}}{\partial x_i}\right)^T \frac{\partial \mathbf{d}}{\partial x_j}\right) \\ &= \frac{1}{2}\left(\frac{\partial d_i}{\partial x_j} + \frac{\partial d_j}{\partial x_i} + \sum_{k=1}^{3} \frac{\partial d_k}{\partial x_i}\frac{\partial d_k}{\partial x_j}\right)\end{aligned}$$

As it can be seen, ε_{ij} has a pretty complicated nonlinear relationship with the displacement $\mathbf{d}$. It will complicate the computation of the stiffness matrix when finite element method is applied. For small deformations, the displacement gradients $\partial d_i/\partial x_j$ are very small and the quadratic term in the above equation can be neglected, which gives Cauchy strain tensor:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial d_i}{\partial x_j} + \frac{\partial d_j}{\partial x_i}\right). \tag{4.5}$$

Cauchy strain tensor is a good approximation of Green strain tensor when the deformations are small, and it results in a constant stiffness matrix that can be computed before runtime deformation. As rotation usually causes large displacements, Cauchy strain tensor is not valid for rigid body rotation.

4.1.2 Stress Tensor

Strain tensor gives a purely kinematic description of the deformation of a solid object. When the shape of an object is deformed by external forces, adjacent parts inside or on the surface of the object push or pull each other causing internal force distributed throughout the body to counteract the external forces. This internal force is called traction force and it is measured per unit area. For a point $\mathbf{x}$ in a continuum solid, suppose there is a plane with normal $\mathbf{n}$ through point $\mathbf{x}$ which splits the object into halves as illustrated in Figure 4.2. Let ΔA be a small area in the vicinity of point $\mathbf{x}$ on the plane and $\Delta\mathbf{f}$ be the resultant surface force acting on ΔA. Then the traction at point $\mathbf{x}$ specified by unit vector $\mathbf{n}$ is defined by:

$$\mathbf{t} = \lim_{\Delta A \to 0} \frac{\Delta \mathbf{f}}{\Delta A}.$$

However traction $\mathbf{t}$ depends on the unit normal vector $\mathbf{n}$. To fully describe the status of the internal force at point $\mathbf{x}$, the traction for all directions is required. To solve this problem, another quantity, *stress tensor*, is introduced.

As illustrated in Figure 4.2, let's consider an infinitesimal cube with its surface normals lying along with the axes of the Cartesian coordinate system and its center

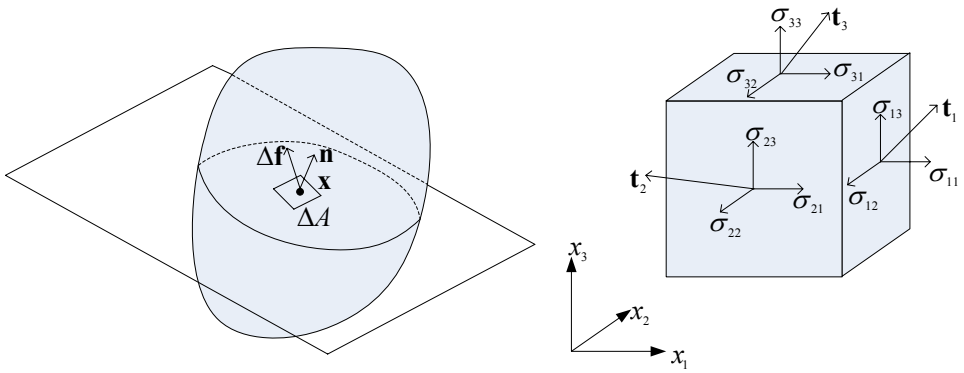

Figure 4.2: Left: The resultant force on the small oriented area. Right: The components of stress tensor.

at point $\mathbf{x}$. Suppose that $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ are the traction acting on the faces of the cube. Each $\mathbf{t}_i$ can be decomposed into three vectors along the directions of axes of the Cartesian coordinate system (Figure 4.2). The nine resultant components form the stress tensor:

$$[\boldsymbol{\sigma}] = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix},$$

where components $\sigma_{11}, \sigma_{22}, \sigma_{33}$ are called normal stresses and $\sigma_{12}, \sigma_{23}, \sigma_{31}, \sigma_{21}, \sigma_{32}, \sigma_{13}$ are called shearing stresses. For most materials, $[\boldsymbol{\sigma}]$ is a 3×3 symmetric matrix. With stress tensor $[\boldsymbol{\sigma}]$, the traction acting on a face element perpendicular to normal $\mathbf{n}$ can be easily obtained:

$$\mathbf{t} = [\boldsymbol{\sigma}]\mathbf{n}.$$

4.1.3 Stress-Strain Relationship

After the representations of strain and stress tensors are determined, we need to establish the relationship between them, which characterizes the physical properties of materials and are usually described by constitutive models. Because of the diversity of materials, different constitutive models are developed based on experimental evidence. These constitutive models generally describe the stress tensor as the function of the strain tensor, strain tensor rate, and strain tensor history. Sophisticated constitutive models may be able to express the stress-strain relationship in such a way that it agrees with the the experimental data perfectly, but they

also lead to complicated computation. We have chosen a simple constitutive model that specifies a linear stress-strain relation. As a matter of fact, many materials exhibit linear elastic behavior under small deformations. This linear elastic constitutive model is defined by generalized Hook's law. Together with Cauchy strain tensor, generalized Hook's law can make the potential energy of the solid object be represented by a quadratic formulation whose minimum can be conveniently obtained. It is probably the most widely used material law in computer animation.

$$\sigma_{ij} = \sum_{k=1}^{3}\sum_{l=1}^{3} c_{ijkl}\varepsilon_{kl}, \tag{4.6}$$

where $c_{ijkl}(1 \leq i, j, k, l \leq 3)$ are 81 coefficients that relate the elements of $[\boldsymbol{\varepsilon}]$ to elements of $[\boldsymbol{\sigma}]$. For isotropic materials, only two of the 81 coefficients are independent, the Lamé's constant λ and shear modulus μ, and Equation 4.6 becomes:

$$\sigma_{ij} = \delta_{ij}\lambda\sum_{k=1}^{3}\varepsilon_{kk} + 2\mu\varepsilon_{ij} \tag{4.7}$$

Lamé's constant λ and shear modulus μ are related to Young's modulus E and Poisson's ratio ν via the following Equations:

$$\begin{aligned} \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned}$$

Poisson's ratio ν determines the incompressibility of the object and it is in the range of $(0, 0.5]$ with $\nu = 0.5$ corresponding to incompressible material. Young's modulus E is non-negative for all materials. Large value of E means that the object requires large forces to accomplish certain amount of deformation.

Both $[\boldsymbol{\varepsilon}]$ and $[\boldsymbol{\sigma}]$ are symmetric matrices, and $\varepsilon_{ij} = \varepsilon_{ji}$, $\sigma_{ij} = \sigma_{ji}$, so the representation of either strain or stress tensor needs only six variables. For the convenience of formulation in finite element method, strain and stress tensors are

formulated as six dimensional vectors:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{pmatrix} = \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} + \varepsilon_{21} \\ \varepsilon_{23} + \varepsilon_{32} \\ \varepsilon_{31} + \varepsilon_{13} \end{pmatrix} = \begin{pmatrix} \frac{\partial d_1}{\partial x_1} \\ \frac{\partial d_2}{\partial x_2} \\ \frac{\partial d_3}{\partial x_3} \\ \frac{\partial d_1}{\partial x_2} + \frac{\partial d_2}{\partial x_1} \\ \frac{\partial d_2}{\partial x_3} + \frac{\partial d_3}{\partial x_2} \\ \frac{\partial d_3}{\partial x_1} + \frac{\partial d_1}{\partial x_3} \end{pmatrix} \tag{4.8}$$

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{zx} \end{pmatrix} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ (\sigma_{12} + \sigma_{21})/2 \\ (\sigma_{23} + \sigma_{32})/2 \\ (\sigma_{31} + \sigma_{13})/2 \end{pmatrix} = \begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{pmatrix} \tag{4.9}$$

Then Equation 4.7 is written as:

$$\boldsymbol{\sigma} = \boldsymbol{\Lambda}\boldsymbol{\varepsilon}$$

with

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}$$

4.1.4 Minimum of Total Potential Energy

The deformation of a solid object is determined by the external forces acting on it and its own material properties. During the deformation of an elastic object, elastic energy is stored inside the body of the object, which is dependent on the stress and strain tensors throughout the object. When the object reaches equilibrium, the distribution of deformation inside the object tends to minimize the total potential energy. The total potential energy of the object is defined by:

$$\Pi = U - W, \tag{4.10}$$

where U is the elastic energy arising from the deformation of the object and W is the summation of work done by all external forces. According to continuum

mechanics, the elastic energy U is calculated based on the strain and stress tensors using the following equation:

$$U = \frac{1}{2}\int_{\Omega}\sum_{i=1}^{3}\sum_{j=1}^{3}\varepsilon_{ij}\sigma_{ij}dV,$$

where Ω is the continuum domain of the solid object. With the vector representation of strain and stress tensors (Equation 4.8, 4.9), the above equation can be formulated as:

$$U = \frac{1}{2}\int_{\Omega}\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}dV = \frac{1}{2}\int_{\Omega}\boldsymbol{\varepsilon}^T\boldsymbol{\Lambda}\boldsymbol{\varepsilon}dV. \tag{4.11}$$

On the other hand, the total work W done by external forces is evaluated by:

$$W = \int_{\Omega}\mathbf{d}^T\mathbf{f}_b dV + \int_{\partial\Omega}\mathbf{d}^T\mathbf{f}_s dS + \sum_k \mathbf{d}_k^T\mathbf{f}_k, \tag{4.12}$$

Where $\partial\Omega$ denotes the boundary of the solid object, $\mathbf{f}_b$ is the force density distributed over the body of the object, $\mathbf{f}_s$ is the force acting on the surface of the object, $\mathbf{f}_k$ is the k-th concentrated force applied on certain point of the object and $\mathbf{d}_k$ is the displacement of that point. Now the deformation problem of an elastic solid subject to external forces is stated as:

Given the geometry of a solid object, all external forces acting on it, and the stress-strain relationship of the material, the deformation or the displacement of every point of the object is determined by minimizing the total potential energy (Equation 4.10).

Due to the arbitrariness of the shape of an object, it is impossible to solve the above problem analytically, finite element method tackles the problem by discretizing the continuous domain into small elements so as to obtain an approximate solution.

4.2 Finite Element Method

Finite element method [57, 111] is a useful tool for predicting the responses of large physical systems subject to external influences. These systems are usually governed by sophisticated laws over non-uniform continuous spatial domain. Since

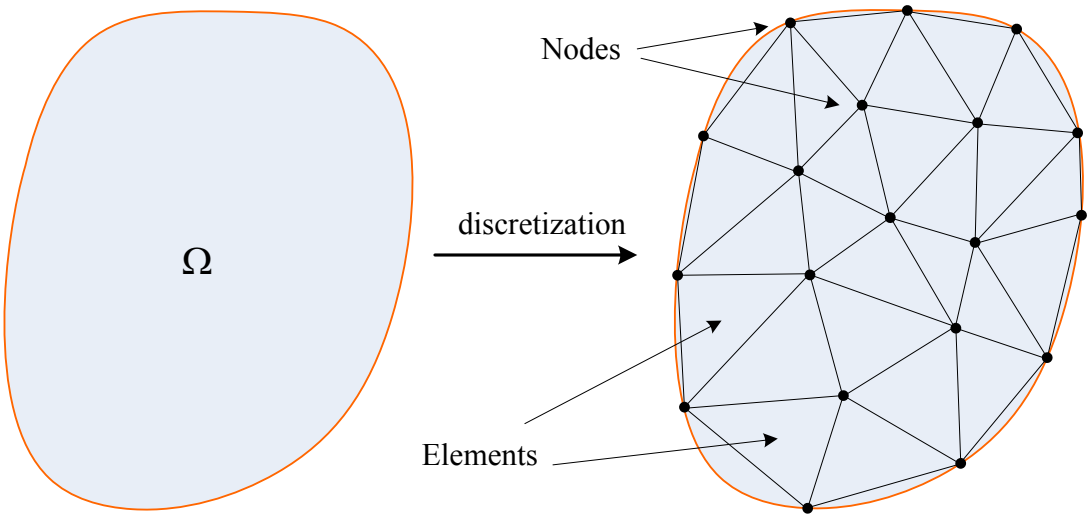

Figure 4.3: Dividing the continuum solid into elements.

the domain of the problem contains an infinite number of points, the system has infinite degrees of freedom and hence it is difficult to handle. Finite element method discretizes the domain into small elements composed of a finite number of special points, termed nodes, and represents the value of a certain quantity at any point in the continuum domain as the interpolation of the values of the quantity at these nodes. It is widely used in engineering areas like solid mechanics, heat transfer and acoustics etc. In this section, we focus on using finite element method to solve the elastic deformation problem addressed in the last section.

4.2.1 Elements and Shape Functions

The first step of applying finite element method is to divide the continuum solid object into elements as illustrated in Figure 4.3. An element is a small simple geometrical structure constituted by several nodes. The elements connect with each other through nodes and together they approximate the shape of the solid object. The accuracy of approximation is influenced by the number of elements and the type of the element. For discretization of a three dimensional object, there are several types of elements for choice like tetrahedral element, wedge element and hexahedral element as shown in Figure 4.4. Tetrahedral element is very popular in applications of computer animation since it can be used to approximate arbitrary

geometric shape. We have employed the wedge element in SWDC. A wedge element has six nodes and five faces (two triangles and three quadrilaterals). As the wedge element is utilized in SWDC, the discussion that follows will be based on wedge element. However, it is straightforward to change to another type of element.

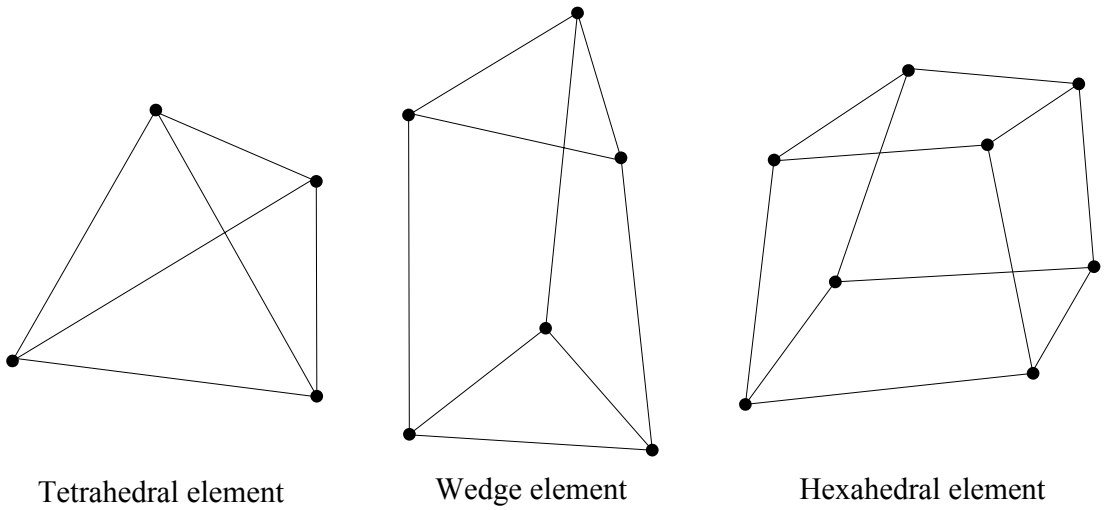

Figure 4.4: Different types of 3D elements.

Given the values of a certain quantity at the nodes of an element, we intend to estimate the value of the quantity at a point inside the element through interpolation. A shape function $N_i(\mathbf{x})$ should be constructed for every node $\mathbf{x}_i (1 \leq i \leq 6)$ of the wedge element. It describes the influence of the i-th node on point $\mathbf{x}$ and satisfies the following condition:

$$N_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{x}_i \\ 0 & \mathbf{x} = \mathbf{x}_j (j \neq i) \end{cases}$$

Generally, polynomial interpolation is used to build shape functions. Higher order polynomials give more degrees of freedom and can result in better approximation to the continuum solid. In most situations, first or second order polynomials are already enough for pretty accurate approximation.

As the solid object is divided into elements, for any point $\mathbf{x}$ in the object, we can find an element that contains this point. Assuming the shape functions for the

six nodes of the element are defined, then displacement $\mathbf{d}$ of point $\mathbf{x}$ is:

$$\mathbf{d}(\mathbf{x}) = \sum_{i=1}^{6} N_i(\mathbf{x})\mathbf{d}_i, \tag{4.13}$$

where $\mathbf{d}_i (1 \leq i \leq 6)$ are the displacements of the six nodes of the element which contains point $\mathbf{x}$.

4.2.2 Governing Equation for an Individual Element

Based on the elasticity theory developed in Section 4.1, a matrix equation can be derived for every element of the continuum solid. The element matrix equation can be used to determine the displacements of the six nodes of the element subject to the forces acting on it. It is easier to deduce this governing matrix equation for a simple individual element than for the whole object with a complex shape. Later, it will be shown that the equations for all elements can be assembled into one large equation which controls the deformation of the entire object.

Let's write Equation 4.13 in matrix form:

$$\mathbf{d} = \begin{pmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \dots & N_6 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \dots & 0 & N_6 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \dots & 0 & 0 & N_6 \end{pmatrix} \mathbf{D}^E = \mathbf{N}\mathbf{D}^E \tag{4.14}$$

where vector $\mathbf{D}^E$, with 18 dimensions, is the concatenation of $\mathbf{d}_i (1 \leq i \leq 6)$, i.e. $\mathbf{D}^E = (\mathbf{d}_1^T, \mathbf{d}_2^T, \dots, \mathbf{d}_6^T)^T$. The superscript E is used to denote the quantities for one individual element. Following Equation 4.8, the strain tensor can now be written as:

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \frac{\partial N_1}{\partial x_1} & 0 & 0 & \frac{\partial N_2}{\partial x_1} & 0 & 0 & \dots & \frac{\partial N_6}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial x_2} & 0 & 0 & \frac{\partial N_2}{\partial x_2} & 0 & \dots & 0 & \frac{\partial N_6}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial x_3} & 0 & 0 & \frac{\partial N_2}{\partial x_3} & \dots & 0 & 0 & \frac{\partial N_6}{\partial x_3} \\ \frac{\partial N_1}{\partial x_1} & \frac{\partial N_1}{\partial x_2} & 0 & \frac{\partial N_2}{\partial x_1} & \frac{\partial N_2}{\partial x_2} & 0 & \dots & \frac{\partial N_6}{\partial x_1} & \frac{\partial N_6}{\partial x_2} & 0 \\ 0 & \frac{\partial N_1}{\partial x_2} & \frac{\partial N_1}{\partial x_3} & 0 & \frac{\partial N_2}{\partial x_2} & \frac{\partial N_2}{\partial x_3} & \dots & 0 & \frac{\partial N_6}{\partial x_2} & \frac{\partial N_6}{\partial x_3} \\ \frac{\partial N_1}{\partial x_1} & 0 & \frac{\partial N_1}{\partial x_3} & \frac{\partial N_2}{\partial x_1} & 0 & \frac{\partial N_2}{\partial x_3} & \dots & \frac{\partial N_6}{\partial x_1} & 0 & \frac{\partial N_6}{\partial x_3} \end{pmatrix} \mathbf{D}^E \tag{4.15}$$

$$= \mathbf{B}^E\mathbf{D}^E \tag{4.16}$$

where $\partial N_i / \partial x_j$ is the partial derivatives of the shape functions with respect to $\mathbf{x}$. Substituting the above equation into Equation 4.11 gives the elastic energy of the

element:

$$\begin{aligned} U^E &= \frac{1}{2}\int_{\Omega^E} (\mathbf{D}^E)^T (\mathbf{B}^E)^T \mathbf{\Lambda} \mathbf{B}^E \mathbf{D}^E dV \\ &= \frac{1}{2}(\mathbf{D}^E)^T \left(\int_{\Omega^E} (\mathbf{B}^E)^T \mathbf{\Lambda} \mathbf{B}^E dV \right) \mathbf{D}^E \\ &= \frac{1}{2}(\mathbf{D}^E)^T \mathbf{K}^E \mathbf{D}^E \end{aligned}$$

where Ω^E means the space occupied by the element in Cartesian coordinate system, and $\mathbf{K}^E$ is the element stiffness matrix.

$$\mathbf{K}^E = \int_{\Omega^E} (\mathbf{B}^E)^T \mathbf{\Lambda} \mathbf{B}^E dV \tag{4.17}$$

Similarly, with the formulation of $\mathbf{d}$ in Equation 4.14, we can describe the total work done by external forces (Equation 4.12) using $\mathbf{D}^E$.

$$\begin{aligned} W^E &= \int_{\Omega^E} (\mathbf{D}^E)^T \mathbf{N}^T \mathbf{f}_b dV + \int_{\partial\Omega^E} (\mathbf{D}^E)^T \mathbf{N}^T \mathbf{f}_s dS + \sum_k (\mathbf{D}^E)^T \mathbf{N}^T(\mathbf{x}(k)) \mathbf{f}_k \\ &= (\mathbf{D}^E)^T \left(\int_{\Omega^E} \mathbf{N}^T \mathbf{f}_b dV \right) + (\mathbf{D}^E)^T \left(\int_{\partial\Omega^E} \mathbf{N}^T \mathbf{f}_s dS \right) + (\mathbf{D}^E)^T \left(\sum_k \mathbf{N}^T(\mathbf{x}(k)) \mathbf{f}_k \right) \\ &= (\mathbf{D}^E)^T \mathbf{F}^E \end{aligned}$$

where $\mathbf{N}(\mathbf{x}(k))$ is the value of matrix $\mathbf{N}$ (Equation 4.14) at the k-th concentrated load point, and $\mathbf{F}^E$ is an 18 dimensional vector, called the generalized force of the element. Then the total potential energy of the element is:

$$\Pi^E = \frac{1}{2}(\mathbf{D}^E)^T \mathbf{K}^E \mathbf{D}^E - (\mathbf{D}^E)^T \mathbf{F}^E.$$

The total potential energy of the entire object will be at the minimum point, if the potential energy of every element is minimized. Setting the gradient of Π^E with respect to $\mathbf{D}^E$ to zero yields the governing matrix equation for the element:

$$\mathbf{K}^E \mathbf{D}^E = \mathbf{F}^E. \tag{4.18}$$

Given the external forces, the displacements of the six nodes of the element can be computed as the solution of the above equation.

4.2.3 Implementation Details

There are several potential problems for computing matrix $\mathbf{K}^E$ in Equation 4.18. Firstly, the shape functions for a non-uniform wedge element in the Cartesian coordinate system can not be defined in a straightforward manner. Secondly, for different elements, the sets of shape functions are different. Finally, integration over a non-uniform wedge element could be troublesome. To remove these hindrances, the isoparametric element is introduced. Instead of defining the shape functions for every individual element in the Cartesian coordinate system, we only need to construct shape functions for one isoparametric element in an intrinsic coordinate system. The isoparametric wedge element is illustrated in Figure 4.5. The coordinates of its six nodes $\mathbf{a}_i (1 \leq i \leq 6)$ in the intrinsic coordinate system are: $(1, 0, -1)$, $(-1/2, -\sqrt{3}/2, -1)$, $(-1/2, \sqrt{3}/2, -1)$, $(1, 0, 1)$, $(-1/2, -\sqrt{3}/2, 1)$ and $(-1/2, \sqrt{3}/2, 1)$. The shape functions [131] for this isoparametric element are easily defined:

$$\begin{aligned}
N_1(\mathbf{a}) &= \frac{1}{6}(1 + 2a_1)(1 - a_3) \\
N_2(\mathbf{a}) &= \frac{1}{6}(1 - a_1 - \sqrt{3}a_2)(1 - a_3) \\
N_3(\mathbf{a}) &= \frac{1}{6}(1 - a_1 + \sqrt{3}a_2)(1 - a_3) \\
N_4(\mathbf{a}) &= \frac{1}{6}(1 + 2a_1)(1 + a_3) \\
N_5(\mathbf{a}) &= \frac{1}{6}(1 - a_1 - \sqrt{3}a_2)(1 + a_3) \\
N_6(\mathbf{a}) &= \frac{1}{6}(1 - a_1 + \sqrt{3}a_2)(1 + a_3)
\end{aligned}$$

Given the positions $\mathbf{x}_i (1 \leq i \leq 6)$ of the six nodes of a wedge element in the Cartesian coordinate system, a mapping from the isoparametric element to the real element can be established:

$$\mathbf{x}(\mathbf{a}) = \sum_{i=1}^{6} N_i(\mathbf{a})\mathbf{x}_i$$

Using the above mapping, the integration in Equation 4.17 can be formulated using

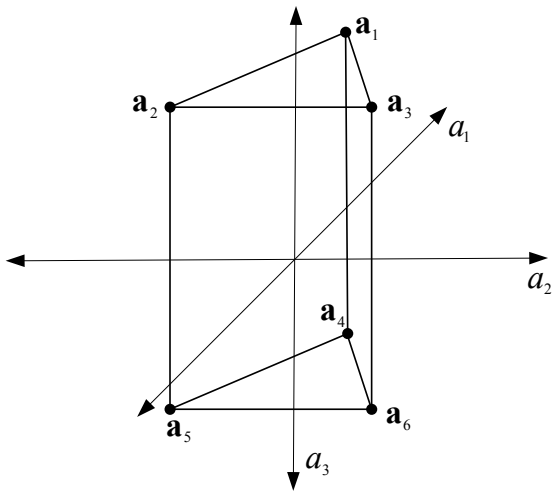

Figure 4.5: The isoparametric element in the intrinsic coordinate system.

the variables $\mathbf{a}$ in the intrinsic coordinate system, and the integration domain can be transformed to the space $\bar{\Omega}^E$, which is occupied by the isoparametric element in the intrinsic coordinate system.

$$\mathbf{K}^E = \int_{\bar{\Omega}^E} (\mathbf{B}^E)^T \mathbf{\Lambda} \mathbf{B}^E \, \|\mathbf{J}\| \, da_1 da_2 da_3, \tag{4.19}$$

where $\mathbf{J}$ is the Jacobin matrix with the partial derivatives of $\partial x_i / \partial a_j (1 \leq i \leq 3, 1 \leq j \leq 3)$ as its elements.

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}} = \begin{pmatrix} \frac{\partial x_1}{\partial a_1} & \frac{\partial x_1}{\partial a_2} & \frac{\partial x_1}{\partial a_3} \\ \frac{\partial x_2}{\partial a_1} & \frac{\partial x_2}{\partial a_2} & \frac{\partial x_2}{\partial a_3} \\ \frac{\partial x_3}{\partial a_1} & \frac{\partial x_3}{\partial a_2} & \frac{\partial x_3}{\partial a_3} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_6 \end{pmatrix} \begin{pmatrix} \frac{\partial N_1}{\partial a_1} & \frac{\partial N_1}{\partial a_2} & \frac{\partial N_1}{\partial a_3} \\ \frac{\partial N_2}{\partial a_1} & \frac{\partial N_2}{\partial a_2} & \frac{\partial N_2}{\partial a_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_6}{\partial a_1} & \frac{\partial N_6}{\partial a_2} & \frac{\partial N_6}{\partial a_3} \end{pmatrix} \tag{4.20}$$

where $\partial N_i / \partial a_j (1 \leq i \leq 6, 1 \leq j \leq 3)$ are given in Table 4.1.

/	∂a_1	∂a_2	∂a_3
∂N_1	$\frac{1}{3}(1-a_3)$	0	$-\frac{1}{6}(1+2a_1)$
∂N_2	$-\frac{1}{6}(1-a_3)$	$-\frac{\sqrt{3}}{6}(1-a_3)$	$-\frac{1}{6}(1-a_1-\sqrt{3})$
∂N_3	$-\frac{1}{6}(1-a_3)$	$\frac{\sqrt{3}}{6}(1-a_3)$	$-\frac{1}{6}(1-a_1+\sqrt{3})$
∂N_4	$\frac{1}{3}(1+a_3)$	0	$\frac{1}{6}(1+2a_1)$
∂N_5	$-\frac{1}{6}(1+a_3)$	$-\frac{\sqrt{3}}{6}(1+a_3)$	$\frac{1}{6}(1-a_1-\sqrt{3})$
∂N_6	$-\frac{1}{6}(1+a_3)$	$\frac{\sqrt{3}}{6}(1+a_3)$	$\frac{1}{6}(1-a_1+\sqrt{3})$

Table 4.1: Derivatives of shape functions in the intrinsic coordinate system

The integration in Equation 4.19 is computed using Gaussian quadrature. Gaussian quadrature is a numeric integration technique and it evaluates the integration

as the weighted summation of the values of the integrand at specific evaluation points. The locations of the evaluation points and values of associated weights are determined to attain maximum accuracy. A m-point Gaussian quadrature rule can integrate a polynomial of $(2m-1)$ order exactly. A 6-point quadrature rule is used for integration in Equation 4.19:

$$\mathbf{K}^E = \sum_{k=1}^{6} w_k (\mathbf{B}^E(\mathbf{a}^k))^T \mathbf{\Lambda} \mathbf{B}^E(\mathbf{a}^k) \|\mathbf{J}(\mathbf{a}^k)\|, \tag{4.21}$$

where $\mathbf{a}^k$ and w_k $(1 \leq k \leq 6)$ are the quadrature points and the associated weights for integration over the isoparametric wedge element. Their values are listed in Table 4.2.

$\mathbf{a}^k$	$1, 0, -1$	$-\frac{1}{2}, -\frac{\sqrt{3}}{2}, -1$	$-\frac{1}{2}, \frac{\sqrt{3}}{2}, -1$	$1, 0, 1$	$-\frac{1}{2}, -\frac{\sqrt{3}}{2}, 1$	$-\frac{1}{2}, \frac{\sqrt{3}}{2}, 1$
w_k	$\frac{\sqrt{3}}{4}$	$\frac{\sqrt{3}}{4}$	$\frac{\sqrt{3}}{4}$	$\frac{\sqrt{3}}{4}$	$\frac{\sqrt{3}}{4}$	$\frac{\sqrt{3}}{4}$

Table 4.2: Quadrature points and the associated weights

The determinant of the Jacobin matrix $\|\mathbf{J}(\mathbf{a}^k)\|$ at those quadrature points can be obtained using Equation 4.20. In order to calculate $\mathbf{B}^E(\mathbf{a}^k)$ (Equation 4.16) at quadrature points, we need to calculate the gradients of $N_i (1 \leq i \leq 6)$ with respect to $\mathbf{x}$. According to differential chain rule,

$$\begin{pmatrix} \frac{\partial N_1}{\partial x_1} & \frac{\partial N_1}{\partial x_2} & \frac{\partial N_1}{\partial x_3} \\ \frac{\partial N_2}{\partial x_1} & \frac{\partial N_2}{\partial x_2} & \frac{\partial N_2}{\partial x_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_6}{\partial x_1} & \frac{\partial N_6}{\partial x_2} & \frac{\partial N_6}{\partial x_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial N_1}{\partial a_1} & \frac{\partial N_1}{\partial a_2} & \frac{\partial N_1}{\partial a_3} \\ \frac{\partial N_2}{\partial a_1} & \frac{\partial N_2}{\partial a_2} & \frac{\partial N_2}{\partial a_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_6}{\partial a_1} & \frac{\partial N_6}{\partial a_2} & \frac{\partial N_6}{\partial a_3} \end{pmatrix} \begin{pmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_1}{\partial x_2} & \frac{\partial a_1}{\partial x_3} \\ \frac{\partial a_2}{\partial x_1} & \frac{\partial a_2}{\partial x_2} & \frac{\partial a_2}{\partial x_3} \\ \frac{\partial a_3}{\partial x_1} & \frac{\partial a_3}{\partial x_2} & \frac{\partial a_3}{\partial x_3} \end{pmatrix} \tag{4.22}$$

with

$$\begin{pmatrix} \frac{\partial a_1}{\partial x_1} & \frac{\partial a_1}{\partial x_2} & \frac{\partial a_1}{\partial x_3} \\ \frac{\partial a_2}{\partial x_1} & \frac{\partial a_2}{\partial x_2} & \frac{\partial a_2}{\partial x_3} \\ \frac{\partial a_3}{\partial x_1} & \frac{\partial a_3}{\partial x_2} & \frac{\partial a_3}{\partial x_3} \end{pmatrix} = \frac{d\mathbf{a}}{d\mathbf{x}} = \mathbf{J}^{-1}$$

The formulations of both $\mathbf{J}$ (Equation 4.20) and $\partial N_i / \partial a_j$ (Table 4.1) are already known, so the values of all $\partial N_i / \partial x_j$ at any quadrature point $\mathbf{a}^k$ can be attained through Equation 4.22, therefore the value of matrix $\mathbf{B}^E(\mathbf{a}^k)$ is obtained. With all the components in Equation 4.21 available, the element stiffness matrix $\mathbf{K}^E$ can be easily calculated.

4.2.4 Assembling of Stiffness Matrix

After the governing matrix equation (Equation 4.18) for every element in the object is acquired using its individual element stiffness matrix $\mathbf{K}^E$, to discover the equation that describes the deformation of the entire object, all element matrix equations are assembled into a large linear system. For a node in an element, it has two indices: one is used in the description above, the in-element index, and the other one is its unique global index among all nodes in the entire object. Suppose the total number of nodes is n. The assembling procedure is given below:

```
double K[3n][3n];   /*  initialized using 0  */
double F[3n];       /*  initialized using 0  */
for( t = 0; t < the number of total element; t++ )
{
  compute element stiffness matrix KE[18][18] for element t;
  for( i = 0; i < 6; i++ )
    for( a = 0; a < 3; a++ )
      for ( j = 0; j < 6; j++ )
        for ( b = 0; b < 3; b++ )
          K[g(i,t)*3 + a][g(j,t)*3 + b] += KE[i*3 + a][j*3 + b];
  if( necessary )
  {
    compute generalized force FE[18] for element t;
    for( i = 0; i < 6; i++ )
      for( a = 0; a < 3; a++ )
        F[g(i,t)*3 + a] += FE[i*3 + a];
  }
}
```

In the above program, both the in-element and global indices of the nodes start from 0, and function $g(i,t)$ is responsible for finding the global index for the i-th

node of the t-th element. As the result of this procedure, the equilibrium equation of the entire object is achieved:

$$\mathbf{KD} = \mathbf{F}, \tag{4.23}$$

where $\mathbf{K}$, the stiffness matrix, is the assembling of all element stiffness matrices, $\mathbf{F}$ is the assembling of the generalized force for all elements, and vector $\mathbf{D}$ is the concatenation of the displacements of all nodes in the solid object according to their global indices. If all external forces are known, the displacements of all the nodes in the object can be calculated by solving Equation 4.23. Because of the adoption of Cauchy strain tensor and linear strain-stress relationship, the whole process of computing stiffness matrix $\mathbf{K}$ only depends on the original shape of the solid object, hence it can be accomplished off-line and does not take the runtime budget. Green strain tensor and nonlinear constitutive models make the calculation of stiffness matrix rely on the deformed shape of the object, thus it has to be estimated step by step along with the deformation of the object at runtime. Although the stiffness matrix derived from linear elastic model can greatly reduce the computational cost, it is only suitable for small deformations and it will generate unrealistic results if the condition is violated.

4.3 Overview of SWDC

As shown in Figure 4.7, SWDC can be divided into three layers:

- The skeleton and the *base shell* reside inside the character to impose constraints on deformable chunks. The skeleton defines the motion of the character, and the base shell follows the skeleton.
- Deformable chunks fill the space between the skin and the base shell/skeleton bones, and they are used to simulate the combined effects of muscles and fat.
- Skin, represented by a triangular mesh, is the outermost layer and is influenced by underlying deformable chunks.

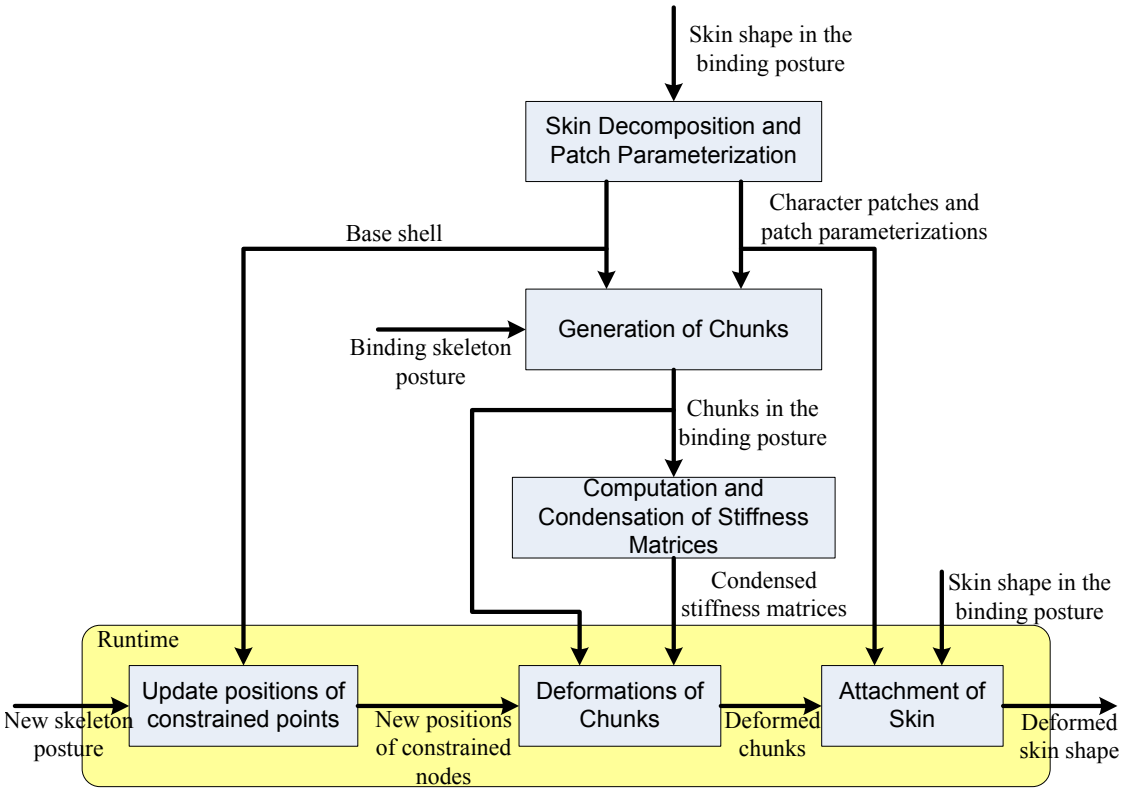

Figure 4.6: The work flow of SWDC. The runtime synthesis only invokes the modules in the yellow areas.

The skin mesh of a character and its skeleton will have to be provided by users as the input of our system. The initial posture of the skeleton is referred to as the *binding posture*. The base shell and deformable chunks are created automatically according to the specifications of the patches on the skin.

Figure 4.6 gives the workflow of SWDC. As in EBTLE (Chapter 3), to set up a character with SWDC, its skin mesh has to be decomposed into patches first. This step is finished using the same technique outlined in Section 3.1, and Figure 4.7-a gives an example of the decomposition of a human upper body model. The polylines on the skin mesh that are used to separate patches from each other are called *boundaries*, and the points connecting different boundaries are called *feature vertices*. To facilitate the generation of chunks, it is necessary to build a parameterization for each skin patch. Harmonic mapping (Section 3.2) is used to compute the parameterizations for all patches of the skin. Figure 4.11 shows an example of the parameterization of a patch. The 2D parameter of vertex $\mathbf{p}$

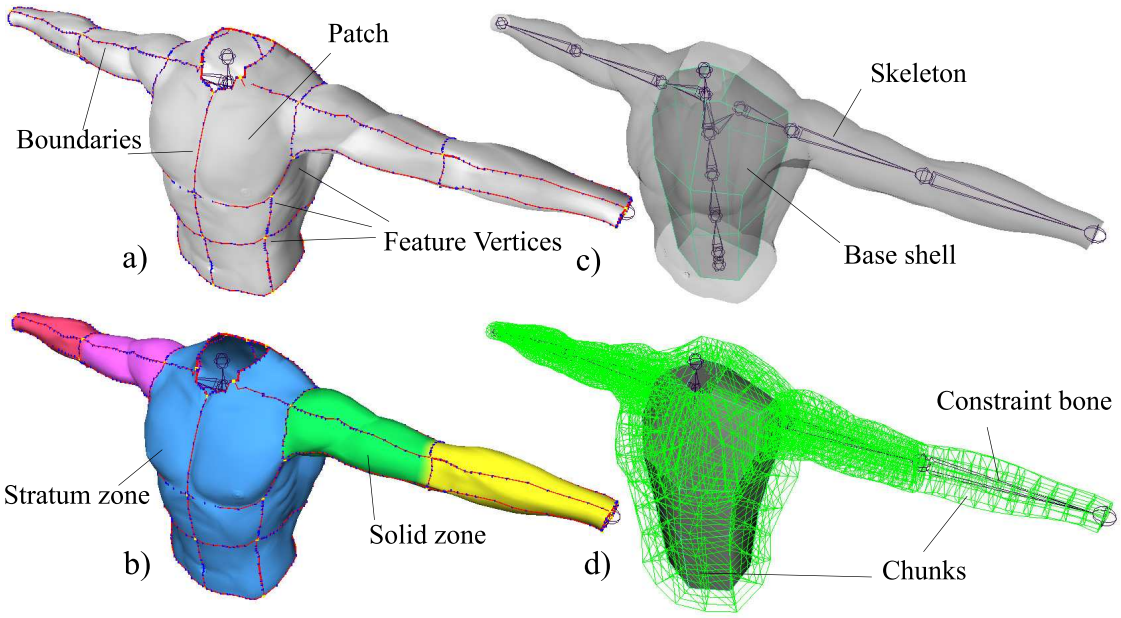

Figure 4.7: a) Decomposition of the skin. b) Grouping patches into zones. c) The skeleton and base shell. d) Deformable chunks.

is denoted by $\hat{\mathbf{p}}$. Then deformable chunks are automatically generated inside the body of the character right beneath the skin. Based on the shapes of these chunks in the binding posture, the stiffness matrices for all chunks are computed and condensed. At runtime, when the skeleton is animated to a new posture, the positions of constrained nodes of all chunks are updated according to the skeleton posture. Constrained nodes are a specific set of nodes in a chunk. The change of the positions of the constrained nodes results in the modification of the positions of other nodes of the chunk. Finally the deformations of the chunks are propagated to the skin shape of the character.

A chunk is modeled as a collection of nodes connected by links in a multi-sliced structure as shown in Figures 4.10 and 4.12, and it has one top surface, one bottom surface and several side surfaces. A slice is referred to one layer of triangular mesh inside the chunk parallel with the top and bottom surfaces (including the top and bottom surfaces). All slices in a chunk share the same connectivity and have the same number of triangles. A side surface is constituted by quadrilaterals. We use m to denote the number of internal slices and r to denote the number of side

surfaces. Throughout this chapter, nodes in a chunk can be referred to by one of the two symbols: $\mathbf{x}_i^j$ denotes the position of the i-th node in the j-th slice of the chunk, and $\mathbf{x}_k$ denotes the position of the k-th node in the chunk.

From the perspective of finite element method, chunks are composed of wedge elements joined through discrete nodes. The total number of nodes in all the chunks of a character has a direct impact on the runtime performance of the deformation. It is observed that the torso of a character usually occupies a large volume of space and requires a considerable number of nodes to fill it up. It is possible to reduce computational cost by only taking a layer of space beneath the skin. A base shell is introduced in our framework to specify the inner bound of nodes. From the anatomy point of view, creatures like human beings always have a rib cage in their torsos. The internal organs inside the rib cage deform in a different way from the outside muscles because their deformations are restricted by the rib cage. So there is no need to simulate the deformation of internal volume using chunks.

As demonstrated in Figure 4.7-b, users are required to group patches into *zones*. By doing so, the interior space of a character is divided into several regions and every region is wrapped by a zone. There are two types of zones: *stratum zone* and *solid zone*. A stratum zone is purposely used for covering the torso region of a character, inside which a base shell is generated (Figure 4.7-c). Only the space between the skin and the base shell will be stuffed with chunks. The base shell is a polygonal mesh whose topology is isomorphic with the patch structure of the zone. Every face of the base shell corresponds with a patch in the zone. The region wrapped by a solid zone will be entirely filled by one chunk. Figure 4.7-b shows a typical example of a solid zone, which consists of all patches on the upper arm. Chunks used for stratum zones are called ordinary chunks, while chunks for solid zones are called separable chunks.

In the stratum zone, the positions of the vertices of the base shell are obtained by moving corresponding feature vertices inwards from the skin along the negative directions of their normals. The distance of the offsets of feature vertices are

decided by users, and it can be used to adjust deformation results. This distance determines the thickness of the chunks filled in the torso region. Chunks form an effective buffer layer between the base shell and the skin. When deformation is passed from the base shell to the skin through the chunks, thick chunks results in smooth and continuous deformation on the skin, while thin chunks cause the skin to closely follow the deformation of the base shell. With thick chunks, the skin deformation caused by the interaction between chunks is more obvious. With thin chunks, the user specified bugling effects are more prominent on the skin. In our experiments, we find that the distance that makes the base shell occupy about half of the internal volume of the torso works well. The shape of the base shell can be further adjusted by animators as they wish.

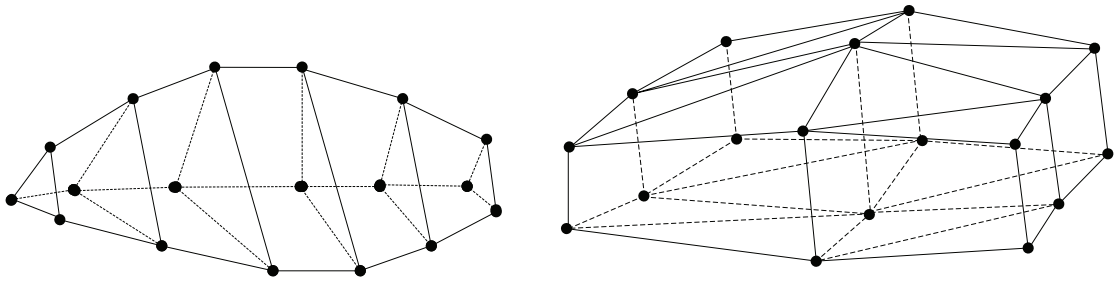

Figure 4.8: Left: Wedge elements are connected sequentially through triangle faces to approximate a fusiform muscle. Right: Wedge elements are connected parallelly through quadrilateral faces to approximate a flat shaped muscle.

We have employed the wedge element in SWDC because it can be used to easily approximate two typical kinds of muscles: fusiform and flat shaped muscles. The typical place to observe fusiform muscles is the upper arm of human body, where a few fusiform muscles like the biceps brachii are aligned along the bone. Flat shaped muscles can usually be found in the torso region. An example of flat shaped muscles from anatomy is the pectoralis major. As shown on the left of Figure 4.8, a few wedge elements can be put together sequentially with their triangle faces connecting with each other. This kind of arrangement is a natural way of approximating a fusiform muscle. If a sequence of wedge elements can be treated as one fusiform muscle, then the separable chunk (see Figure 4.12) can be

regards as the composition of multiple parallel fusiform muscles, which bears some similarity to the situation in reality. On the other hand, when these wedge elements are assembled parallelly and connected by their quadrilateral faces, they can be treated as a flat shaped muscle (see the right of Figure 4.8). As demonstrated in Figure 4.10, the ordinary chunk can be regarded as a stack of flat shaped muscles, and this kind of structure can also find its counterpart in real anatomy.

4.4 Generation of Ordinary Chunks

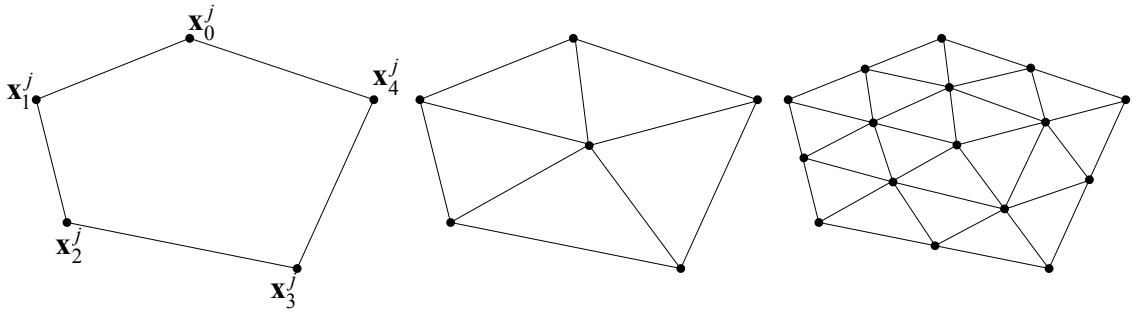

Figure 4.9: Left: A polygon slice. Middle: Triangulation of the polygon by introducing the center node. Right: Uniform quaternary subdivision of the triangles.

In a stratum zone, between every patch and its corresponding base shell face, an ordinary chunk is generated to fill the gap as shown in Figures 4.10 and 4.14. The prototype of a chunk is a polyhedron constituted by $\mathbf{x}_0^0$, $\mathbf{x}_1^0$, ..., $\mathbf{x}_{r-1}^0$ and $\mathbf{x}_0^{m+1}$, $\mathbf{x}_1^{m+1}$, ..., $\mathbf{x}_{r-1}^{m+1}$, where $\mathbf{x}_0^0$, $\mathbf{x}_1^0$, ..., $\mathbf{x}_{r-1}^0$ are equal to the coordinates of the feature vertices of the patch, and $\mathbf{x}_0^{m+1}$, $\mathbf{x}_1^{m+1}$, ..., $\mathbf{x}_{r-1}^{m+1}$ are identical to the positions of the vertices of the base shell face. Then, m slices are inserted into the polyhedron, and the j-th slice is polygon $\{\mathbf{x}_0^j, \mathbf{x}_1^j, \ldots, \mathbf{x}_{r-1}^j\}$, where

$$\mathbf{x}_i^j = (1 - \frac{j}{m+1})\mathbf{x}_i^0 + \frac{j}{m+1}\mathbf{x}_i^{m+1} \quad (0 \leq i < r).$$

Next, all the polygon slices are divided into triangles in the same way. Triangulation of a polygon slice can be done by simply connecting certain pairs of the nodes of the polygon or by introducing a center node, whose position is the average of the coordinates of all nodes, and connecting every node with it (Figure 4.9). After triangulation, uniform quaternary subdivision is applied to all triangles in

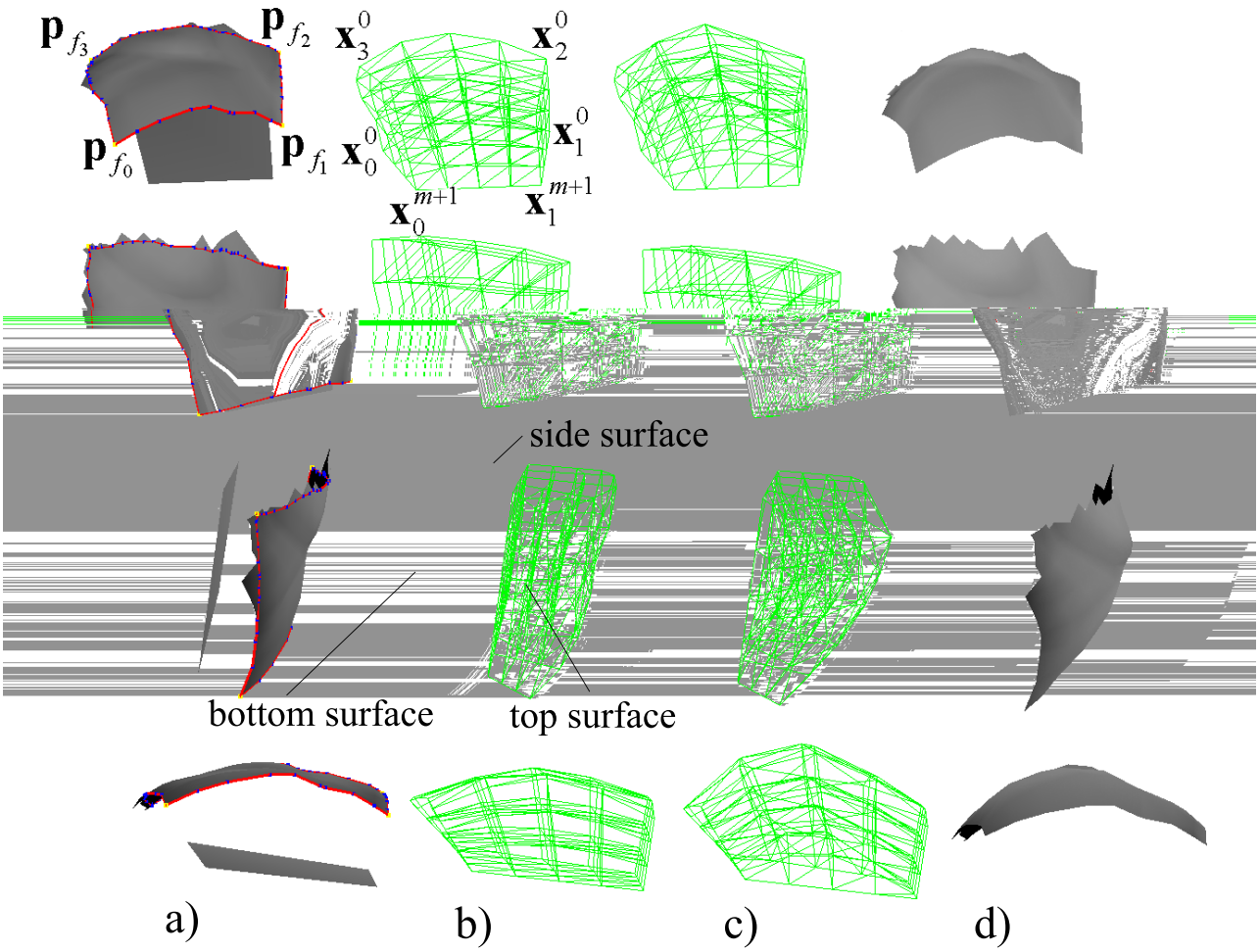

Figure 4.10: a) A patch in the stratum zone and its corresponding shell base face. b) The ordinary chunk generated to fill the gap. c) The deformation of the ordinary chunk. d) The deformation of the patch shape.

all slices s times, where s is determined by users and may be different for each individual chunk. As shown in Figure 4.9, uniform quaternary subdivision divides a triangle into four triangles by inserting new nodes at the middle points of the three edges of the triangle and connecting them with each other. Finally, linking the corresponding nodes in adjacent slices yields the default form of the chunk.

Usually, the default form of the chunk does not completely fill the space between the patch and the base shell face. The nodes in the top surface have to be displaced onto the patch shape. To accomplish this, the top surface of the chunk is parameterized so that every node $\mathbf{x}_i^0$ has a 2D parameter $\hat{\mathbf{x}}_i^0$. The corner nodes of the top surface $\mathbf{x}_0^0$, $\mathbf{x}_1^0$, ..., $\mathbf{x}_{r-1}^0$ are first assigned parameters $\hat{\mathbf{p}}_{f_0}$, $\hat{\mathbf{p}}_{f_1}$, ..., $\hat{\mathbf{p}}_{f_{r-1}}$,

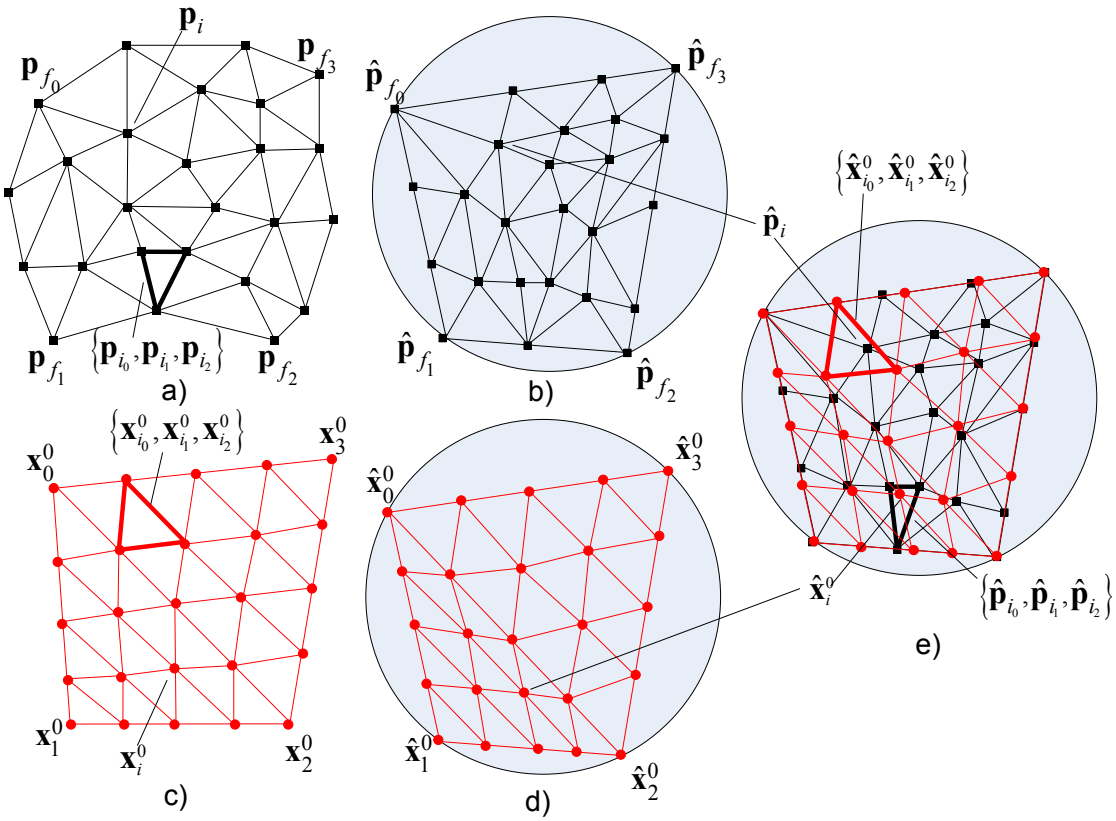

Figure 4.11: a) A patch in the stratum zone. b) The parameterization of the patch. c) The top surface of the default form of the corresponding chunk. d) The parameterization of the top surface. e) Superposed parameterizations.

i.e. $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{p}}_{f_i} (0 \leq i < r)$, where $\hat{\mathbf{p}}_{f_i}$ is the parameter of the i-th feature vertex $\mathbf{p}_{f_i}$. Then, the parameters of other nodes in the top surface can be deduced from $\hat{\mathbf{x}}_0^0$, $\hat{\mathbf{x}}_1^0$, $\ldots$, $\hat{\mathbf{x}}_{r-1}^0$ using the triangulation and subdivision rules applied for the refinement of a polygon slice. If the triangulation of polygon slices is accomplished by introducing a center node, the parameter of the center node in the top surface is the average of $\hat{\mathbf{x}}_i^0 (0 \leq i < r)$. From the parameters of the nodes in the 0-th subdivision level, we can estimate the parameters of nodes in any subdivision level. Suppose the parameters of the nodes in the k-th level are known, when the quaternary subdivision is applied, there is a node of the $(k+1)$-th level added at the middle point of an edge, and its parameter is assigned as the average of the parameters of the two nodes incidental to the edge. Figure 4.11 shows the parameterization of the top surface of a chunk. Now, the parameterization of the patch shape and the parameterization of the top surface can be superposed. Given node $\mathbf{x}_i^0$, the triangle $\{\hat{\mathbf{p}}_{i_0}, \hat{\mathbf{p}}_{i_1}, \hat{\mathbf{p}}_{i_2}\}$

that contains $\hat{\mathbf{x}}_i^0$ is found in the parameter domain (see Figure 4.11-e). Suppose $\{\alpha^{\mathbf{p}}, \beta^{\mathbf{p}}, \gamma^{\mathbf{p}}\}$ are the barycentric coordinates of $\hat{\mathbf{x}}_i^0$ w.r.t. $\{\hat{\mathbf{p}}_{i_0}, \hat{\mathbf{p}}_{i_1}, \hat{\mathbf{p}}_{i_2}\}$. The new location of node $\mathbf{x}_i^0$ on the patch shape is:

$$\mathbf{x}_i^0 = \alpha^{\mathbf{p}}\mathbf{p}_{i_0} + \beta^{\mathbf{p}}\mathbf{p}_{i_1} + \gamma^{\mathbf{p}}\mathbf{p}_{i_2}.$$

After the positions of the nodes in the top surface are changed, the positions of the nodes inside the chunk should also be adjusted so that they can be evenly distributed. Relaxation is an iterative procedure that displaces a node according to the positions of its direct neighbors. The updating rule of the position of node $\mathbf{x}_i$ is described by following equation:

$$\mathbf{x}_i = \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} w_{ij} \left(\mathbf{x}_j - \mathbf{x}_i\right), \tag{4.24}$$

where $\mathcal{N}(i)$ is the set of indices of direct neighbors of node $\mathbf{x}_i$, and w_{ij} is the coefficient associated with link $\{\mathbf{x}_i, \mathbf{x}_j\}$ and is estimated as follows:

- If node $\mathbf{x}_j$ is in the same slice with $\mathbf{x}_i$, $w_{ij} = 0.5\,(1/v)$, where v is the valence of $\mathbf{x}_i$ in the slice.
- If node $\mathbf{x}_j$ is not in the same slice with $\mathbf{x}_i$, $w_{ij} = 0.25$.

The relaxation is applied to all interior nodes of the chunk, whilst the nodes in all surfaces are kept stationary. In our experiments, only several iterations are required for the nodes to reach reasonable positions.

4.5 Generation of Separable Chunks

A patch in a solid zone is restricted to have exactly four boundaries, and all patches in the zone together form a cylinder-like surface. The region enclosed by this cylinder-like surface only requires one separable chunk as stuffing. Figure 4.12 exhibits a solid zone composed of four patches and the separable chunk wrapped by it. A solid zone has two closed joint boundaries constituted by boundaries of its patches. Suppose $\mathbf{p}_{f_0^0}$, $\mathbf{p}_{f_1^0}$, ..., $\mathbf{p}_{f_{r-1}^0}$ are the feature vertices on the first

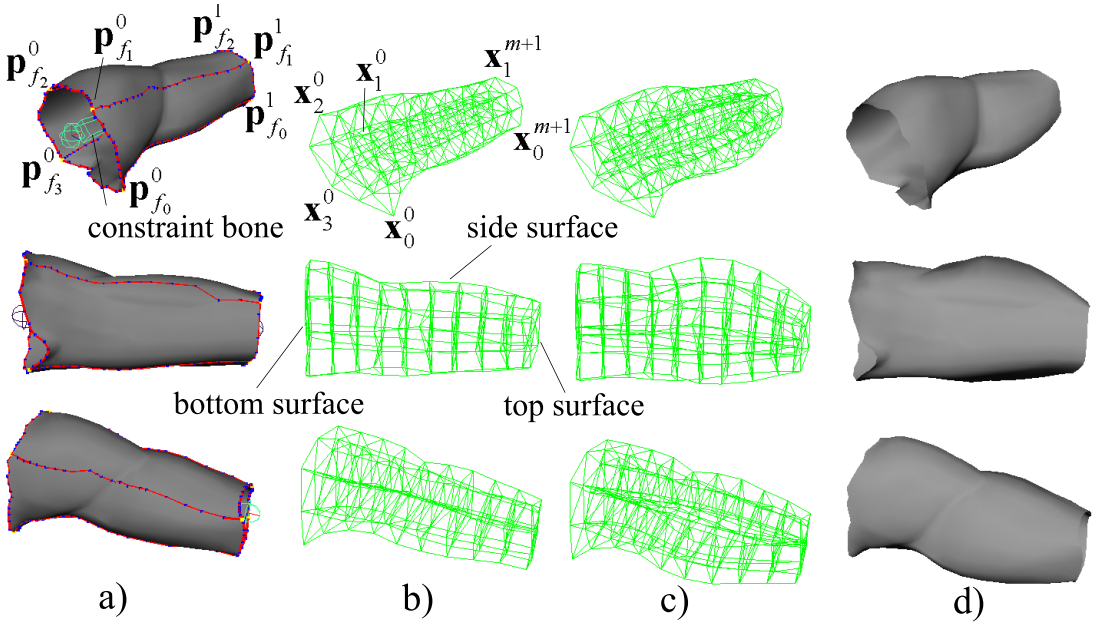

Figure 4.12: a) A solid zone and its constraint bone. b) The separable chunk that fills the region. c) The deformation of the separable chunk. d) The deformations of patch shapes in the zone.

joint boundary and $\mathbf{p}_{f_0^1}$, $\mathbf{p}_{f_1^1}$, ..., $\mathbf{p}_{f_{r-1}^1}$ are the feature vertices on the second joint boundary. There is a skeleton bone surrounded by the solid zone as shown in Figures 4.12 and 4.15. This bone will be used to specify the positions of constrained nodes of the separable chunk, and it will be referred to as the constraint bone of the chunk. The positions of the two joints incidental to the bone are $\mathbf{o}_0$ and $\mathbf{o}_1$ (see Figure 4.15).

The prototype polyhedron $\{\mathbf{x}_0^0, \mathbf{x}_1^0, \ldots, \mathbf{x}_{r-1}^0, \mathbf{x}_0^{m+1}, \mathbf{x}_1^{m+1}, \ldots, \mathbf{x}_{r-1}^{m+1}\}$ of the chunk is generated by making $\mathbf{x}_i^0 = \mathbf{p}_{f_i^0}$, $\mathbf{x}_i^{m+1} = \mathbf{p}_{f_i^1} (0 \leq i < r)$. Internal polygon slices are inserted as described in Section 4.4. The triangulation of polygon slices of a separable chunk is different from an ordinary chunk. In fact, the j-th polygon slice $\{\mathbf{x}_0^j, \mathbf{x}_1^j, \ldots, \mathbf{x}_{r-1}^j\}$ is not triangulated, but split. r coincidental nodes $\mathbf{x}_{c_0}^j, \mathbf{x}_{c_1}^j, \ldots, \mathbf{x}_{c_{r-1}}^j$ (Figure 4.15) on the constraint bone $\{\mathbf{o}_0, \mathbf{o}_1\}$ are introduced, where

$$\mathbf{x}_{c_i}^j = (1 - \frac{j}{m+1})\mathbf{o}_0 + \frac{j}{m+1}\mathbf{o}_1 \quad (0 \leq i < r).$$

Polygon $\{\mathbf{x}_0^j, \mathbf{x}_1^j, \ldots, \mathbf{x}_{r-1}^j\}$ is split into r triangles $\{\mathbf{x}_0^j, \mathbf{x}_1^j, \mathbf{x}_{c_0}^j\}$, $\{\mathbf{x}_1^j, \mathbf{x}_2^j, \mathbf{x}_{c_1}^j\}$, ..., $\{\mathbf{x}_{r-1}^j, \mathbf{x}_0^j, \mathbf{x}_{c_{r-1}}^j\}$. Then all triangles in all slices are subdivided the same number

of times, and the default form of the chunk is obtained by connecting nodes in consecutive slices.

Unlike an ordinary chunk that adjusts the positions of the nodes in its top surface, the nodes in every side surface of a separable chunk are moved to the corresponding patch in the solid zone (see Figures 4.12 and 4.15). The parameterization of the k-th side surface should be constructed and superposed with the parameterization of the k-th patch. It requires that the parameters of the four corner nodes of the k-th side surface $\hat{\mathbf{x}}_k^0, \hat{\mathbf{x}}_{k+1}^0, \hat{\mathbf{x}}_{k+1}^{m+1}, \hat{\mathbf{x}}_k^{m+1}$ equal the parameters of the feature vertices of the k-th patch $\hat{\mathbf{p}}_{f_k^0}, \hat{\mathbf{p}}_{f_{k+1}^0}, \hat{\mathbf{p}}_{f_{k+1}^1}, \hat{\mathbf{p}}_{f_k^1}$. The parameters of other nodes in the k-th side surface are determined using simple bilinear interpolation of $\hat{\mathbf{x}}_k^0, \hat{\mathbf{x}}_{k+1}^0, \hat{\mathbf{x}}_{k+1}^{m+1}, \hat{\mathbf{x}}_k^{m+1}$. Suppose $\mathbf{x}_{k_i}^j$ is the i-th node on slice edge $\{\hat{\mathbf{x}}_k^j, \hat{\mathbf{x}}_{k+1}^j\}$. The parameter of node $\mathbf{x}_{k_i}^j$ is given by:

$$\begin{aligned}\hat{\mathbf{x}}_{k_i}^j &= \left((1-\frac{j}{m+1})\hat{\mathbf{x}}_k^0 + \frac{j}{m+1}\hat{\mathbf{x}}_k^{m+1}\right)(1-\frac{i}{2^s}) \\ &+ \left((1-\frac{j}{m+1})\hat{\mathbf{x}}_{k+1}^0 + \frac{j}{m+1}\hat{\mathbf{x}}_{k+1}^{m+1}\right)\frac{i}{2^s},\end{aligned}$$

where s is the subdivision level of the chunk. Using its 2D parameter, a node in the k-th side surface is projected to the k-th patch in the same manner as explained in Section 4.4.

Except for the nodes on the level-0 triangle edges $\{\mathbf{x}_i^j, \mathbf{x}_{c_i}^j\}$, $\{\mathbf{x}_{i+1}^j, \mathbf{x}_{c_i}^j\}$, $\{\mathbf{x}_i^j, \mathbf{x}_{i+1}^j\}$ $(0 \le j \le m+1, 0 \le i < r)$, all the nodes of the separable chunk are relaxed using Equation 4.24 to achieve smooth node configuration.

4.6 Constrained Quasi-static Deformation Model

Finite element method is used to calculate the deformation of every chunk when the skeleton changes its posture. A constrained quasi-static deformation model is used to compute the positions of free nodes of a chunk depending on the positions of constrained nodes. Chunks are treated as linear isotropic material and Hooke's law is employed to describe the relationship between stress and strain tensors (Section 4.1). Cauchy strain tensor enables off-line computation of the stiffness matrix,

but limits the application of the deformation model to small displacements. To incorporate this computation model into our framework, a local frame is associated with every chunk, and computation is conducted in the local frame.

The local frame associated with an ordinary chunk is built on the corresponding base shell face with the normal of the face as its z-axis (see Figure 4.13). The base shell is bound with the skeleton of the character using Skeleton Subspace Deformation (SSD) (Section 3.4.1) so that its shape and the local frames on it change properly according to the skeleton posture. The local frame of a separable chunk is the coordinate system defined on the constraint bone of the chunk and its z-axis follows the direction of the bone.

As outlined in Section 4.2, minimizing the total potential energy of the chunk yields a linear system:

$$\mathbf{KD} = \mathbf{F}, \tag{4.25}$$

where matrix $\mathbf{K}$ is the stiffness matrix, vector $\mathbf{D}$ is the concatenation of displacements of all nodes, and vector $\mathbf{F}$ is the result of all external forces applied on the chunk. Constrained deformation model is based on the assumption that the displacements of constrained nodes are known, while there are no external forces applied on the free nodes. Given the new positions of constrained nodes, the new position of every free node can be calculated using Equation 4.25. However, in our framework, only the nodes in the top surface of an ordinary chunk and the nodes in the side surfaces of a separable chunk are utilized to determine the deformation of the skin (described in Section 4.8). Therefore, the computation of the positions of interior free nodes of a chunk is unnecessary and should not be conducted so that the runtime synthesis of the skin shape would be faster. Stiffness matrix condensation technique proposed by Bro-Nielsen and Cotin [19] enables the computation to focus on the boundary of a deformable object and perfectly meets our requirements.

All the nodes in a chunk are divided into three mutually exclusive sets. Let $\mathcal{C} =$

$\{i_0, i_1, \ldots, i_{n_c}\}$, $\mathcal{S} = \{j_0, j_1, \ldots, j_{n_s}\}$, $\mathcal{F} = \{k_0, k_1, \ldots, k_{n_f}\}$ be the sets of indices of constrained nodes, free surface nodes influencing the skin, and free interior nodes, respectively. Suppose $\mathbf{d}_i$ is the displacement of node $\mathbf{x}_i$ in the local frame and $\mathbf{D}_c = (\mathbf{d}_{i_0}^T, \mathbf{d}_{i_1}^T, \ldots, \mathbf{d}_{i_{n_c}}^T)^T$, $\mathbf{D}_s = (\mathbf{d}_{j_0}^T, \mathbf{d}_{j_1}^T, \ldots, \mathbf{d}_{j_{n_s}}^T)^T$, $\mathbf{D}_f = (\mathbf{d}_{k_0}^T, \mathbf{d}_{k_1}^T, \ldots, \mathbf{d}_{k_{n_f}}^T)^T$. By reordering the rows and columns of the stiffness matrix, Equation 4.25 can be reformulated as:

$$\begin{pmatrix} \mathbf{K}_{00} & \mathbf{K}_{01} & \mathbf{K}_{02} \\ \mathbf{K}_{10} & \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{20} & \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{D}_c \\ \mathbf{D}_s \\ \mathbf{D}_f \end{pmatrix} = \begin{pmatrix} \mathbf{F}_c \\ \mathbf{F}_s \\ \mathbf{F}_f \end{pmatrix}. \tag{4.26}$$

With $\mathbf{F}_s = \mathbf{0}$ and $\mathbf{F}_f = \mathbf{0}$, we can derive the following equations:

$$\mathbf{K}_{10}\mathbf{D}_c + \mathbf{K}_{11}\mathbf{D}_s + \mathbf{K}_{12}\mathbf{D}_f = \mathbf{0} \tag{4.27}$$

$$\mathbf{K}_{20}\mathbf{D}_c + \mathbf{K}_{21}\mathbf{D}_s + \mathbf{K}_{22}\mathbf{D}_f = \mathbf{0} \tag{4.28}$$

Substituting Equation 4.28 into Equation 4.27 gives:

$$\mathbf{D}_s = \mathbf{A}\mathbf{D}_c, \tag{4.29}$$

where

$$\mathbf{A} = \left(\mathbf{K}_{11} - \mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{K}_{21}\right)^{-1}\left(\mathbf{K}_{12}\mathbf{K}_{22}^{-1}\mathbf{K}_{20} - \mathbf{K}_{10}\right).$$

Matrix $\mathbf{A}$ is computed off-line. At runtime, the displacements of free surface nodes of the chunk can be interactively obtained by simply multiplying matrix $\mathbf{A}$ with vector $\mathbf{D}_c$ (Equation 4.29).

In reality, muscle deformation is a nonlinear process due to its complex structure. It requires a nonlinear elastic model to simulate the deformation. However, in SWDC, the deformation of a chunk is an approximation to the deformations of a group of muscles. The calculation of the deformation for the chunk is done in a local frame, which always follows the skeleton. The deformation of the chunk in this local frame can be considered as small deformation, and a linear elastic model can be applied. Another reason for using the linear model is to meet the requirement of realtime speed. With a nonlinear elastic model, the resultant stiffness matrix will not be a constant matrix, and it needs to be updated along with the deformation

at runtime. When the chunk has a large number of nodes, the calculation of this matrix is quite expensive. For the linear elastic model, this stiffness matrix is a constant matrix, and it can be calculated during the setup process. Furthermore, with the matrix condensation technique, the dimension of this matrix can be reduced to improve the speed. Since usually there are quite a few chunks beneath the skin and high-resolution chunks are needed in highly deformable areas, it will be a problem to achieve interactive speed using a nonlinear elastic model.

4.7 Chunk Deformation

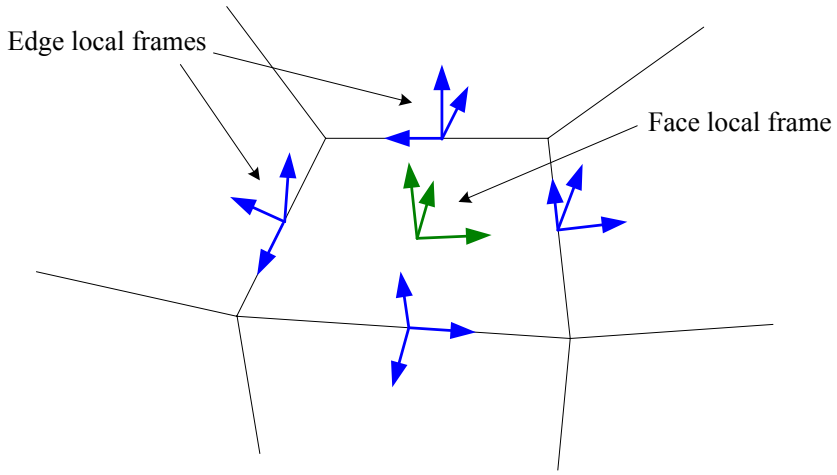

Figure 4.13: The local frame defined on the face of the base shell and the local frames associated with the edges of the face.

To calculate the deformation of a chunk, the positions of its constrained nodes have to be determined first. The constrained nodes of an ordinary chunk include the nodes in its bottom surface and all side surfaces (Figure 4.14). Every side surface of an ordinary chunk corresponds with an edge of the base shell. There is a local frame defined on each edge of the base shell (Figure 4.13). The z-axis of this local frame is the direction of the summed vector of the normals of the faces incidental to the edge, the y-axis has the same orientation with the edge, and the x-axis is the cross product of the z-axis and the y-axis. At runtime, the constrained nodes in a side surface are transformed to new positions by the local coordinate system associated

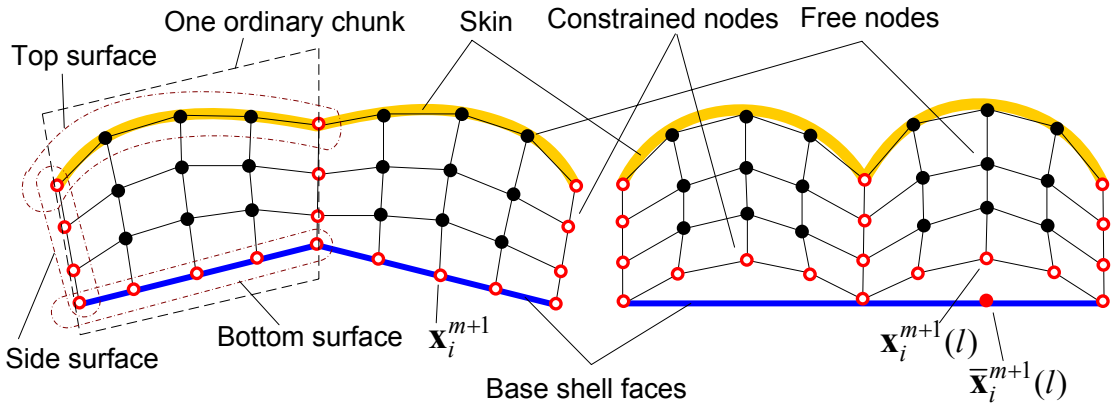

Figure 4.14: Left: Ordinary chunks in the binding posture. Right: Deformed ordinary chunks in the new posture.

with the corresponding edge. The constrained nodes in the bottom surface of an ordinary chunk move together with the corresponding base shell face. In order to generate bulging effect on the top surface of the chunk, the nodes in the bottom surface are lifted from the base shell face through a Gaussian function. Figure 4.14 illustrates the deformations of ordinary chunks. As the skeleton is animated to new posture l ($l = 0$ if the skeleton is in the binding posture), the new location of node $\mathbf{x}_i^{m+1}$ is:

$$\mathbf{x}_i^{m+1}(l) = \bar{\mathbf{x}}_i^{m+1}(l) + \mathbf{n}he^{-\frac{\left\|\hat{\mathbf{x}}_i^{m+1}-\hat{\mathbf{c}}\right\|^2}{R^2}}, \tag{4.30}$$

where $\bar{\mathbf{x}}_i^{m+1}(l)$ is the position of node $\mathbf{x}_i^{m+1}$ on the deformed base shell shape, $\mathbf{n}$ is the normal of the base shell face, $\hat{\mathbf{x}}_i^{m+1}$ is the 2D parameter of node $\mathbf{x}_i^{m+1}$ with $\hat{\mathbf{x}}_i^{m+1} = \hat{\mathbf{x}}_i^0$, and $\hat{\mathbf{c}}$, h and R are bulging parameters that can be adjusted by users. $\bar{\mathbf{x}}_i^{m+1}(l)$ is evaluated using its barycentric coordinates with respect to the vertices of the base shell face in the similar fashion as described in Section 3.5.1 except that parameter $\hat{\mathbf{x}}_i^{m+1}$ is used instead.

In Equation 4.30, $\hat{\mathbf{c}}$, h and R control the center, height and drop off ratio of the bulge created on the top surface. $\hat{\mathbf{c}}$ denotes the center of the displacement field, and usually $(0, 0)$ means that the maximum displacement is at the center of the patch. Other value of between $[-1, 1]$ will cause the maximum displacement to drift from the center of the patch. R controls the rate of the height decrease for

the displacement field when the parameter of the node is away from the specified center $\hat{\mathbf{c}}$. Large R indicates that the height drops slowly, and a relatively flat bulge will be formed on the skin. Small R means the height drops quickly, and a sharp bulge will be generated. h specifies the height of maximum displacement. When h is equivalent to zero, there will be no offsets for the nodes of the bottom surface and they just follow the base shell face closely. Thus, no bulge will be formed on the skin. Since the value of h generally varies with the posture of the skeleton, users may want to describe it as some simple function of certain skeletal parameters. We also provide an alternative scheme that connects the value of h with the area of the base shell face:

$$h = h_0(a_0/a - 1),$$

where h_0 is a user defined parameter, a_0 is the area of the base shell face in the binding posture, and a is its area in current posture.

The constrained nodes of a separable chunk contain not only the nodes in its top and bottom surfaces but also the nodes on its r axes $\{\mathbf{x}_{c_i}^0, \mathbf{x}_{c_i}^{m+1}\}(0 \leq i < r)$ (Figure 4.15). During the animation, when the skeleton reaches a new posture, the new positions of nodes in the top or bottom surface are estimated as follows:

- If the surface is adjacent to another separable chunk, it rotates around the joint properly to bisect the joint angle between the constraint bones of the two chunks. The surface could also be scaled slightly in a certain direction to compensate for the collapsing effects that may happen near the joint.

- If the surface is adjacent to an ordinary chunk, the positions of the nodes in it are determined by the deformation of the ordinary chunk.

- Otherwise, the nodes in the surface just follow the skeleton rigidly.

Bulging effect can be forged on each side surface of a separable chunk by curving the corresponding axis as illustrated in Figure 4.15. Let $\mathbf{o}_0(l)$ and $\mathbf{o}_1(l)$ be the joint positions of the constraint bone of the chunk in current posture l. The new

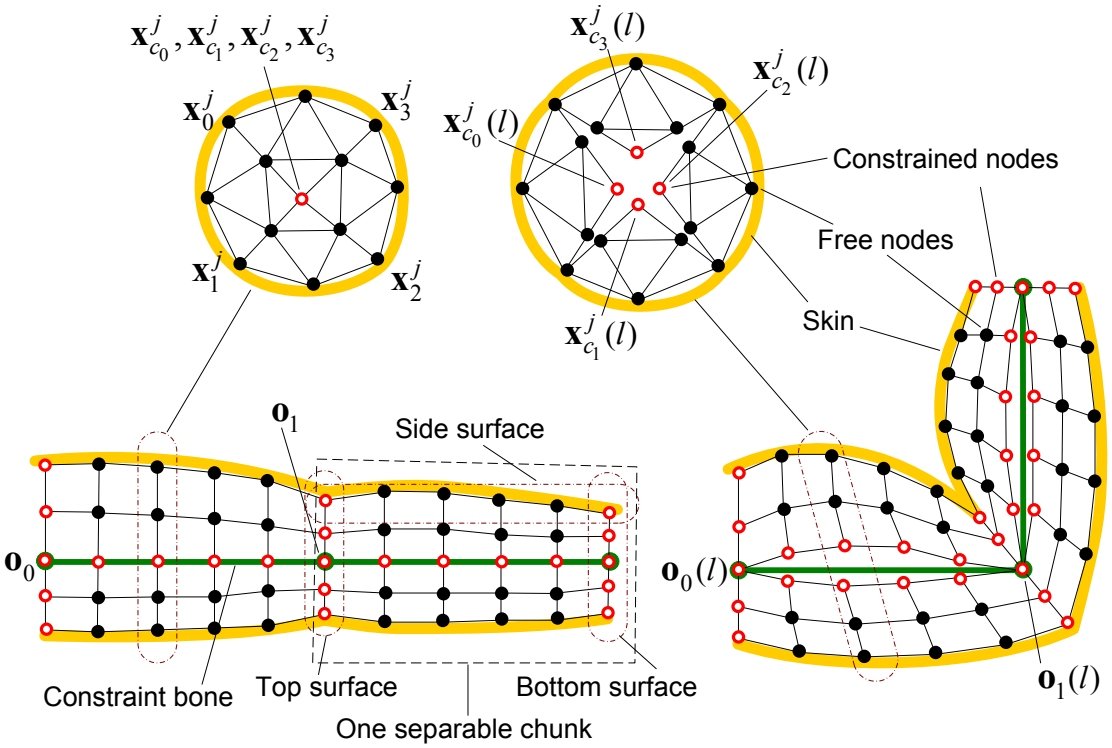

Figure 4.15: Left: Separable chunks in the binding posture. Right: Deformed separable chunks in the new posture.

location of node $\mathbf{x}^j_{c_k}$ on the k-th axis is described by:

$$\mathbf{x}^j_{c_k}(l) = (1-u)\mathbf{o}_0(l) + u\mathbf{o}_1(l) + \mathbf{n}he^{-\frac{(u-c)^2}{R^2}}, \tag{4.31}$$

where $u = \frac{j}{m+1}$, $\mathbf{n}$ is the vector orthogonal to vector $\overrightarrow{\mathbf{o}_0(l)\mathbf{o}_1(l)}$ and pointing towards the k-th side surface, and c, h, R influence the center, height and drop off ratio of the bulge on the k-th side surface. c, h and R have similar meaning with the parameters in Equation 4.30. c determines the center of the displacement field. When its value is set to 0.5, the center is at the center of the bone. R determines the drop off ratio of the displacement field along the bone. h is the height of the maximum displacement. The value of h is usually controlled by the value of a joint angle. For example, the elbow flexion angle drives the bulging parameters of both the upper and lower arm chunks.

Using the position of constrained node $\mathbf{x}_i (i \in \mathcal{C})$ in posture l, its displacement in the local frame is computed as:

$$\mathbf{d}_i(l) = (\mathbf{M}^l)^{-1}\mathbf{x}_i(l) - (\mathbf{M}^0)^{-1}\mathbf{x}_i,$$

where $\mathbf{M}^0$ and $\mathbf{M}^l$ are the transformation matrices of the associated local frame in the binding and current postures respectively. Collecting $\mathbf{d}_i(l)$ for all constrained nodes and substituting them into Equation 4.29 result in the local displacements of free surface nodes. The world coordinates of free surface node $\mathbf{x}_i (i \in \mathcal{S})$ in the new posture l are now determined by:

$$\mathbf{x}_i(l) = \mathbf{M}^l \left((\mathbf{M}^0)^{-1} \mathbf{x}_i + \mathbf{d}_i(l) \right).$$

4.8 Attaching Skin

The deformations of chunks lead to the deformation of the skin. For a skin vertex $\mathbf{p}_i$ in a patch belonging to a stratum zone, according to the superimposed parameterizations of the patch and the top surface of the underlying chunk, we can find the triangle $\{\hat{\mathbf{x}}^0_{i_0}, \hat{\mathbf{x}}^0_{i_1}, \hat{\mathbf{x}}^0_{i_2}\}$ that includes $\hat{\mathbf{p}}_i$ (see Figure 4.11-e). Let the barycentric coordinates of $\hat{\mathbf{p}}_i$ in triangle $\{\hat{\mathbf{x}}^0_{i_0}, \hat{\mathbf{x}}^0_{i_1}, \hat{\mathbf{x}}^0_{i_2}\}$ be $\{\alpha^{\mathbf{x}}, \beta^{\mathbf{x}}, \gamma^{\mathbf{x}}\}$. The *register point* of vertex $\mathbf{p}_i$ on the deformed chunk in posture l is defined as:

$$\mathbf{q}_i(l) = \alpha^{\mathbf{x}} \mathbf{x}^0_{i_0}(l) + \beta^{\mathbf{x}} \mathbf{x}^0_{i_1}(l) + \gamma^{\mathbf{x}} \mathbf{x}^0_{i_2}(l)$$

where $\{\mathbf{x}^0_{i_0}, \mathbf{x}^0_{i_1}, \mathbf{x}^0_{i_2}\}$ is the triangle in the top surface of the underlying ordinary chunk and its vertex parameters are $\{\hat{\mathbf{x}}^0_{i_0}, \hat{\mathbf{x}}^0_{i_1}, \hat{\mathbf{x}}^0_{i_2}\}$.

Similarly, for a skin vertex in a patch of a solid zone, we can also find its register point on one of the side surfaces of the underlying separable chunk, except that bilinear coordinates are used instead of barycentric coordinates because the parameterization of the side surface is composed of planar quadrilaterals.

$$\begin{aligned} \mathbf{q}_i(l) &= \left((1-\alpha)\mathbf{x}^j_{i_0}(l) + \alpha \mathbf{x}^j_{i_1}(l) \right)(1-\beta) \\ &+ \left((1-\alpha)\mathbf{x}^{j+1}_{i_0}(l) + \alpha \mathbf{x}^{j+1}_{i_1}(l) \right)\beta, \end{aligned}$$

where $\{\mathbf{x}^j_{i_0}, \mathbf{x}^j_{i_1}, \mathbf{x}^{j+1}_{i_1}, \mathbf{x}^{j+1}_{i_0}\}$ is the small quadrilateral in the k-th side surface of the underlying chunk and it corresponds with the planar quadrilateral $\{\hat{\mathbf{x}}^j_{i_0}, \hat{\mathbf{x}}^j_{i_1}, \hat{\mathbf{x}}^{j+1}_{i_1}, \hat{\mathbf{x}}^{j+1}_{i_0}\}$ which involves $\hat{\mathbf{p}}_i$ in the parameterization domain, and (α, β) is the bilinear coordinates of $\hat{\mathbf{p}}_i$ with respect to $\{\hat{\mathbf{x}}^j_{i_0}, \hat{\mathbf{x}}^j_{i_1}, \hat{\mathbf{x}}^{j+1}_{i_1}, \hat{\mathbf{x}}^{j+1}_{i_0}\}$.

The *residual displacement* between the position of vertex $\mathbf{p}_i$ and its register point is estimated and recorded in the binding posture using the following equation:

$$\mathbf{r}_i = (\mathbf{M}^0)^{-1}(\mathbf{p}_i - \mathbf{q}_i(0))$$

The residual displacement of a vertex is very small, considering the intimacy between the skin and underlying chunks. Finally, the deformed position of vertex $\mathbf{p}_i$ in posture l is:

$$\mathbf{p}_i(l) = \mathbf{q}_i(l) + \mathbf{M}^l\mathbf{r}_i$$

4.9 Results

We have implemented our skinning model as a plugin of Maya [2] on a workstation with a 2.8-GHz processor. The deformations caused by a single chunk are shown in Figure 4.10 and 4.12 with different perspectives. To demonstrate the capability of multiple chunks, an upper human body (Figure 4.7) has been set up. This character contains 4697 vertices (excluding the hands). As illustrated in Figure 4.7, its skin is decomposed into 34 patches, and one stratum zone and four solid zones are formed using these patches. After the generation of the base shell, 20 ordinary chunks are created for the torso and 4 separable chunks for the arms. There are 5784 nodes contained in all the chunks. Figure 4.16 displays the deformation of the chest of the human when his arms fold forward. As shown in Figure 4.17, the pleasing appearances of bending and twisting of the arm have been successfully generated using separable chunks. During the flexion of the elbow, the upper arm bulges as if it is caused by contraction of real biceps. Figure 4.18 shows some snapshots of the entire upper human body in various skeleton postures. The deformations of the chunks in one of the postures are exhibited in Figure 4.19.

To verify the versatility of our method, we have experimented on another character, a polar bear, which consists of 12913 vertices. Figure 4.20 illustrates the shape of the bear and the decomposition of its skin. From the 82 patches, a stratum zone (torso) and nine solid zones (two for every leg and one for the neck)

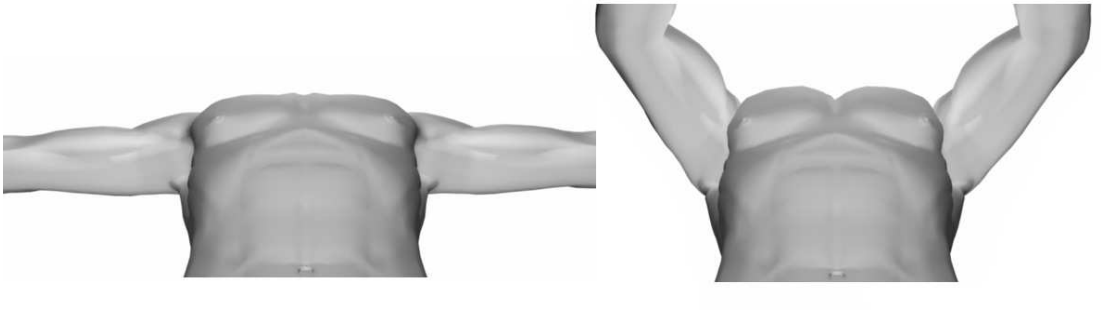

Figure 4.16: The deformation of the chest.

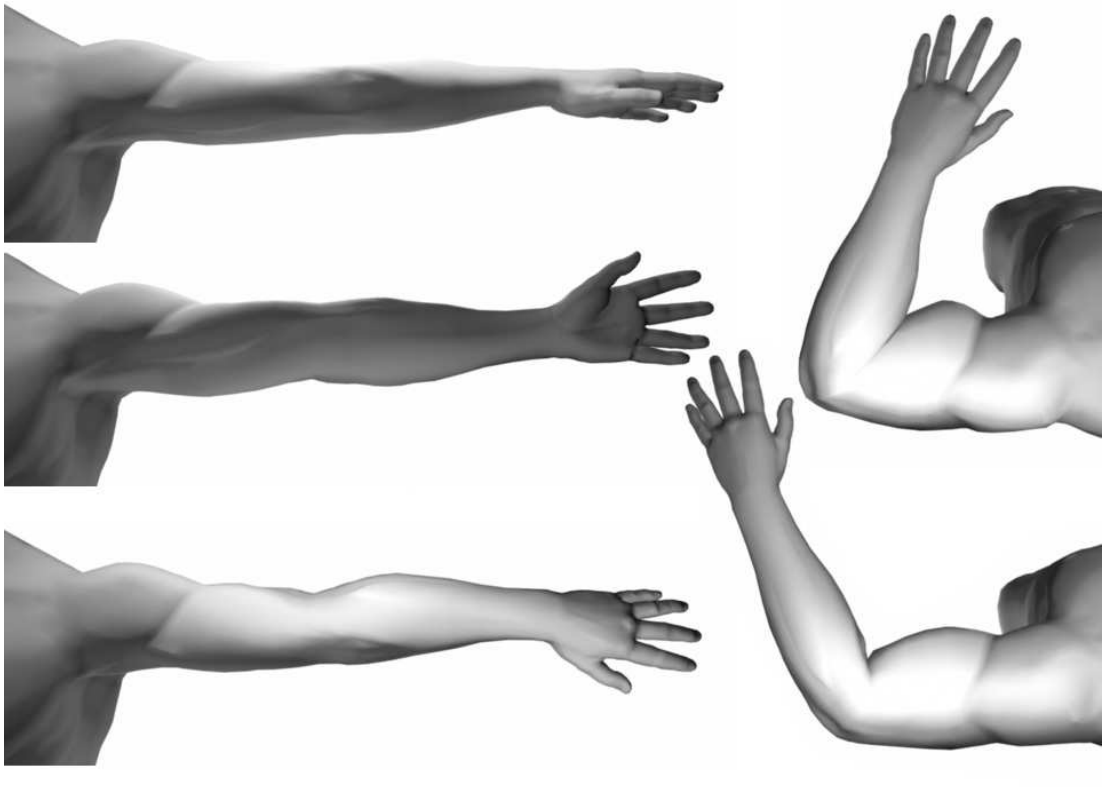

Figure 4.17: Twisting and bending of the arm.

are formed. The internal space of the bear is filled with 35 ordinary chunks and 9 separable chunks (Figure 4.20). The total number of nodes of all the chunks is 11100. As shown in Figure 4.21, the skin deformations of the bear are smooth and aesthetically satisfying in different postures. Note that the bulging effects are set to be minor on the skin considering that the experimental character is a bear. Figure 4.19 displays the deformed chunks residing in the bear when it is in a standing posture. (For more results of animation videos, see the attached CD.)

In both experiments, the number of slices and the number of subdivision level

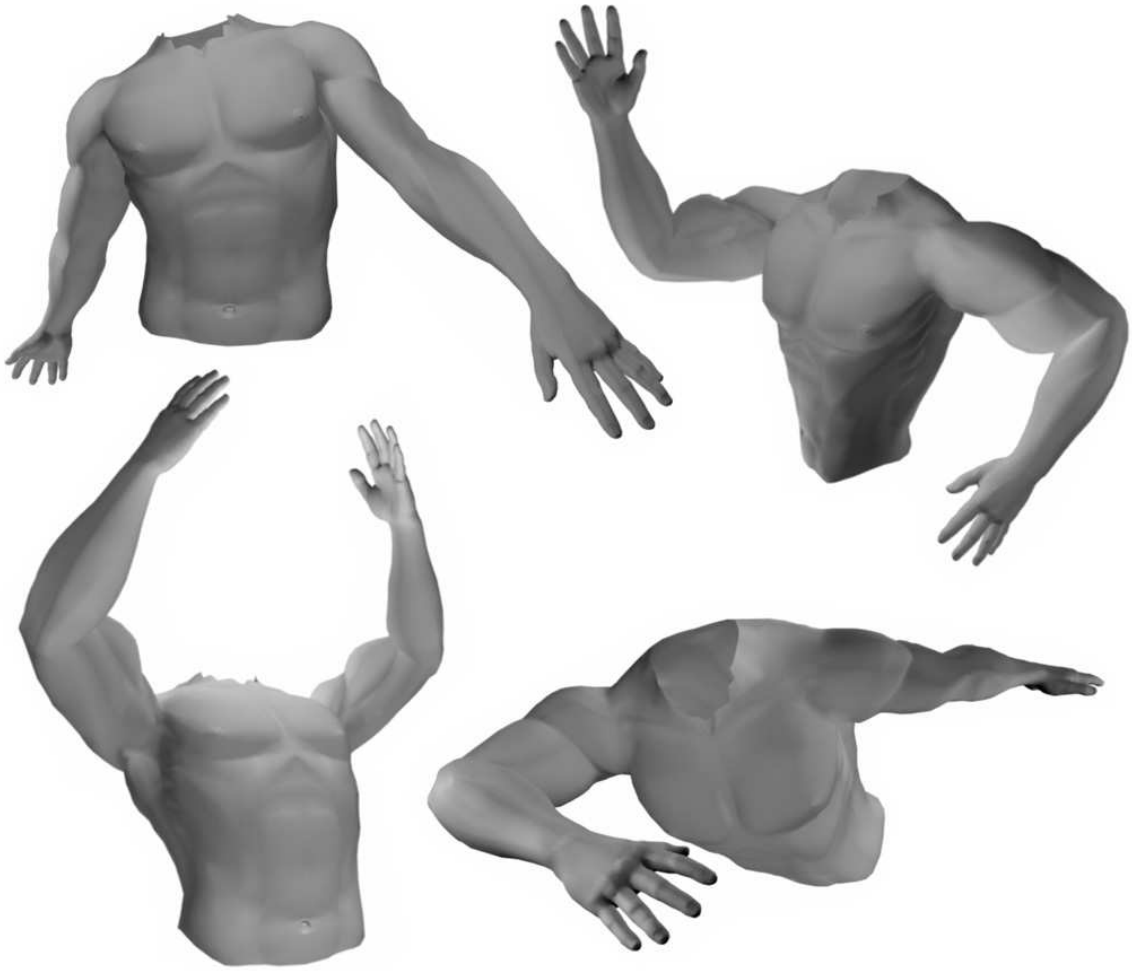

Figure 4.18: Snapshots of the upper human body in various skeleton postures.

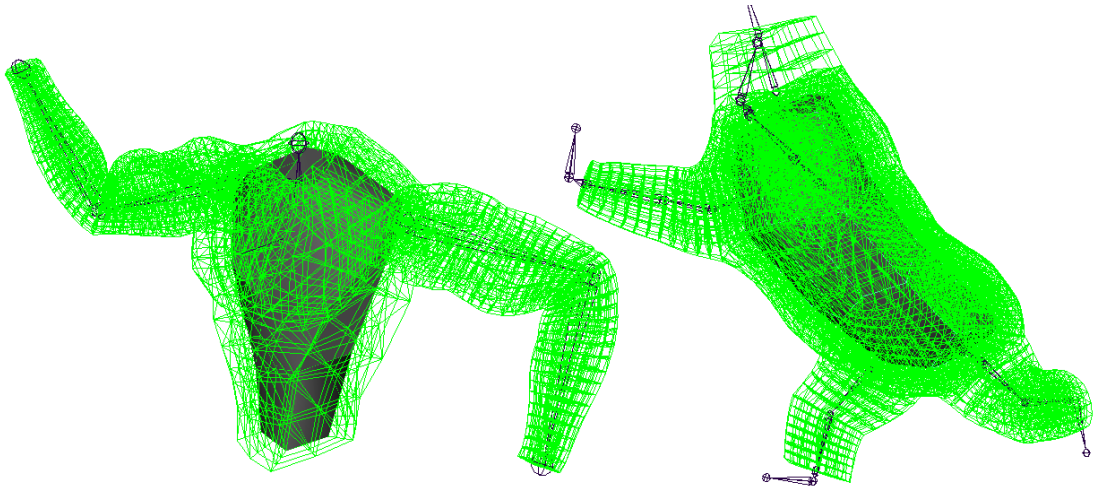

Figure 4.19: As the skeletons change their postures, chunks inside the human (left) and the bear (right) deform via FEM.

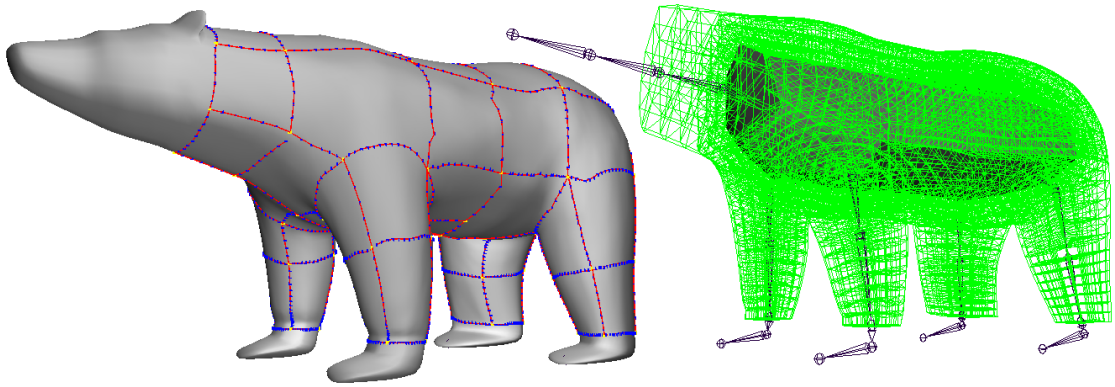

Figure 4.20: Left: A polar bear and the decomposition of its skin. Right: The internal deformable chunks.

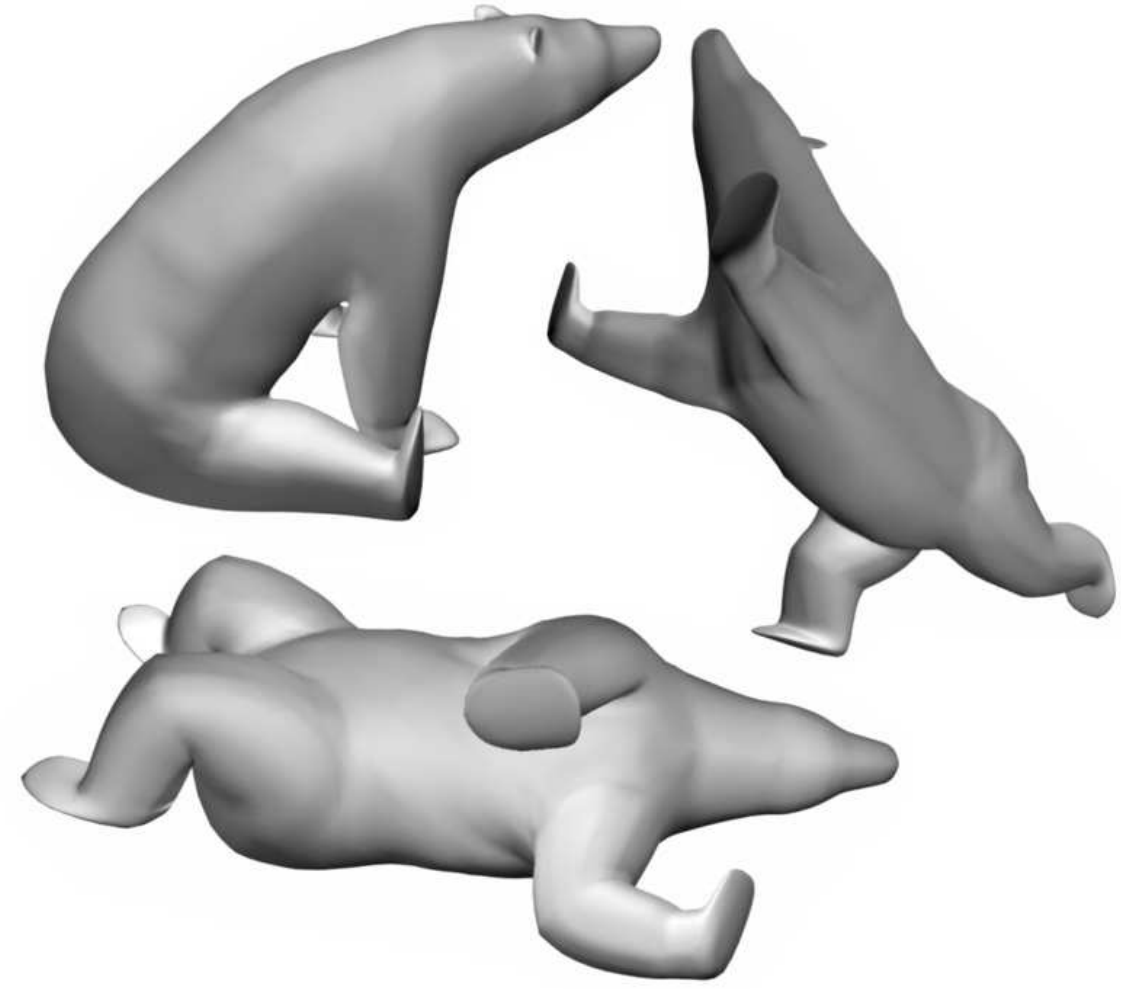

Figure 4.21: Skin deformations of the bear in different postures.

of a chunk are selected according to deformation requirements of the corresponding skin area. Table 4.3 gives the statistics of chunks in the typical body regions. We choose Poisson's ratio ν between $0.4 \sim 0.49$ and E between $1e+6 \sim 9.9e+7Pa$ (see Section 4.1.3), and the material described by such kind of parameters is easy to deform and hard to compress.

Body region	Human left upper arm	Human left chest	Bear upper leg	Bear left back
Chunk type	separable	ordinary	separable	ordinary
Number of slices	18	4	6	4
Subdivision levels	2	2	2	3
Triangles per slice	64	96	64	128
Number of side surfaces	4	6	4	4
Number of nodes	1008	244	336	324

Table 4.3: Statistics of chunks for typical body regions.

In our framework, only the manual decomposition of the skin demands some labor from users. The chunk generation process takes about one to two minutes (including the computation of parameterizations). During the animation, it takes about 0.09 seconds to calculate the skin deformation of the upper human body and about 0.17 seconds for the polar bear. The off-line computation of stiffness matrices and condensation of these matrices requires 56.7 seconds for the upper human body and 16.7 seconds for the polar bear.

In SWDC, the quality of skin deformation depends on the resolutions of chunks and their placement beneath of the skin. Highly deformable area should use high-resolution chunks to create smooth deformation. If a low-resolution chunk is used to create severe deformation, discontinuity will be observed on the skin. However, high-resolution chunks require more computational power, so in order to have an optimum chunk structure, users should use high-resolution and low-resolution chunks in suitable areas accordingly.

4.10 Summary

In this chapter, we have introduced deformable chunks, powered by finite element method, to create skin deformations for virtual characters. Compared with conventional anatomically based models, it is more convenient for users to set up a character using SWDC, because the generation of chunks only requires decomposition of the skin geometry. Two types of chunks are proposed to satisfy the different skin deformation requirements of different body regions. The deformations of the chunks with bulging effects are adequate for most of the visual characteristics of skin deformations generated by real musculature. The constrained quasi-static deformation model governed by generalized Hook's law is perfectly incorporated into our framework. The runtime synthesis of our skinning model can be conducted at an interactive speed due to the pre-computation of the stiffness matrices and their condensation. Intuitive rigs are also devised to facilitate the adjustments of deformation results.

SWDC offers another method to generate natural skin deformations. However it also has its own shortcomings. The structure of patches determines the placement of the chunks inside the character. Although it is much easier to specify patches on the skin than to build the entire internal anatomical structure using individual muscles, the decomposition of the skin is still a manual process, and it takes artists skills and time to obtain a suitable decomposition. The attaching process in SWDC uses a simple transformation technique based on register point. It is possible that unexpected creases can occur around the connecting area of two chunks. This kind problem is obvious when SWDC is used to synthesize certain types of characters like snake, which require continuous smooth deformation. A deformable chunk with linear elastic model is capable of many kinds of deformations, but it is not as flexible as a group of muscles. There may be some complex situations in which deformable chunks can not fully describe the desired deformations. Another limitation of SWDC is that users need to decide the resolutions of the chunks used inside the

character, and no automatic process is available. Users need to make sure highly deformable areas are represented by high-resolution chunks so that the deformation doesn't have discontinuities, and flat areas are represented by low-resolution chunks so that the computational cost can be reduced.

Chapter 5

Conclusion

We have applied the tactic of decomposition for character skin deformations. To synthesize skin deformations for an articulated character, the skin of the character is decomposed into patches and the deformations are created for individual patches. Under the direction of this tactic, two skinning models, Example Based Transferable Layered Enveloping (EBTLE) and Skinning With Deformable Chunks (SWDC), are developed. The contributions of our work are summarized as follows:

- Based on layered construction of skin deformations, EBTLE uses examples of the primary character to correct the behavior of both the skin and the middle layer, the base control mesh. Relying on the radial basis function interpolation in the hyper space of the skeletal parameters augmented with skin vertex parameters, EBTLE can use the examples of the primary character to guide the skin deformations of not only itself but also a group of target characters with similar appearances. The decomposition of the skin simplifies the problem and the number of examples required to achieve satisfactory results is

quite small.

- SWDC provides a way to automatically generate internal entities inside the body of a character instead of building the anatomical structure of the character muscle by muscle. The only manual work, decomposition of the skin, is intuitive, and does not need profound knowledge of anatomy. Common skin deformations caused by a group of muscles can be accomplished by one chunk.

- EBTLE and SWDC give users different options of describing skin deformations. With EBTLE, the specification of the deformations is intuitive and direct. The users only need to sculpt the desired shapes and put them in the set of examples. SWDC does not require the users to provide extra examples. Once a static shape of a character and the decomposition of its skin are available, SWDC can make the character immediately animatable. However, users can only indirectly specify deformation results by adjusting the bulging parameters of chunks.

- Since both methods create skin deformations based on patches, it is straightforward for artists to locally refine the deformation of a specific patch. For EBTLE, the examples used for different patches can be different. To increase the subtlety of the deformation of a skin patch, users may have to use more examples to guide the behavior of that patch. For SWDC, the deformable chunk under the patch can be individually refined to create detailed deformation.

- Both methods are efficient and can produce realistic deformations at interactive speed.

- For both EBTLE and SWDC, the deformation is unique regarding the skeleton posture. If the skeleton is in the same posture, the final synthesized

shape will be the same, because the deformation is calculated based on rotation angles of the joints, and same posture means same rotation angles. So it doesn't matter whether the character goes through different animation sequences, as long as the final posture is the same, the skin deformation will be same, under the condition that all the other parameter setting is identical.

Both EBTLE and SWDC also have their own limitations. Following this dissertation work, there are several areas of future work that we believe would be worthy of pursuing.

- EBTLE uses a coarse polygon mesh, the base control mesh, as the common platform, while other alternatives like subdivision surface may be more flexible and be able to approximate different target characters better.
- In EBTLE, the radial basis function interpolation is a main cost of runtime synthesis, and the time and memory required by it grow proportionally with the number of vertices in the patch and the number of examples. It is preferred that the computational cost at runtime is independent of these two factors.
- With EBTLE, to create desired deformations, users need to make sure that the examples provided to the system are scattered in the pose space, and there are no confusing examples. The computational model for learning from examples should be more robust so as to handle all kinds of examples.
- The deformation transfer of EBTLE is limited to similar characters due to the requirement of same patch structure. It will be interesting to expand the technique for very different characters.
- In SWDC, the bulging effects caused by a chunk are less delicate than the deformation effects generated by a group of muscles, and may not be capable of describing the desired deformations in certain cases. One way of remedying

this problem is to combine the advantage of example based methods. With example shapes of the character, multiple instances of the same chunk with different shapes can be automatically generated in different postures. When the skeleton is changed to a new posture, every instance of the chunk deforms independently according to the skeleton posture, and the final deformed shape of the chunk will be the interpolation of all the deformed instances.

- In SWDC, the resolution (the number of slices and the number of subdivision level) of a chunk influences the quality and speed of its deformation and is selected by users. Higher resolution generates better results but demands more computation. An automatically adaptive chunk structure can help to improve the quality while maintaining interactive speed.

- SWDC utilizes quasi-static linear deformation model to calculate the deformation of a chunk for its efficiency, while nonlinear material models should also be investigated.

- The starting point for both EBTLE and SWDC, the decomposition of the skin, is currently a manual process. Users should have a basic understanding of the decomposition's influence on these two frameworks, and then they can try different patch structures to obtain a suitable decomposition for the experimental character. The decomposition of the skin should be more user-friendly. The set up process of our framework would be largely simplified if some techniques can be developed to computationally measure the quality of decomposition and automatically produce the suitable decomposition for the experimental character.

- Both EBTLE and SWDC use some simple schemes for the final transformations of the skin vertices (patch stitching for EBTLE and attaching skin for SWDC). They need to be improved so that they can automatically remove unwanted creases. A more advanced technique, like a physically based

dynamical skin model, could further increase the fidelity of the deformations.

- In either EBTLE or SWDC, there is no mechanism to prevent skin from self-penetration. So with both frameworks, it is the artists' job to make sure the skin doesn't intersect with itself. Collision detection techniques should be utilized to prevent penetration in these areas.

- Neither the radial basis function interpolation in EBTLE nor the quasi-static deformation model in SWDC handles dynamical effects of the skin such as jiggling and shaking. For some characters, these dynamical effects are obvious on certain areas of their skin. To simulate these effects, physically based methods should be used for the dynamical simulation of the internal entities beneath the skin.

- In our research, the skin mesh of the character should not have too many fine features like wrinkles or goose bumps. Since these types of skin features usually just follow the skin during the deformation, they can be handled by rendering techniques like normal mapping. In that case, these fine skin features are represented using normal map textures, and when the character is rendered, the lighting model will recreate these features using the textures. If these fine features are represented by polygonal mesh, the skin mesh will need a large number of vertices, and it will increase the burden of both deformation and rendering. In both EBTLE and SWDC, the mesh connectivity and shape of a patch determine its parameterization. Fine skin features of the patch will increase the complexity of its parameterization, and unnecessary information will be embedded in the parameterization. In EBTLE, since the computational model of the deformation is based on this parameterization, radial basis function interpolation technique will face a more difficult situation when it uses this complicated parameterization to configure the computational model. In SWDC, the parameterization is used to facilitate the generation of the chunks. When the parameterization contains unneces-

sary information of the fine features, it can make the generated chunks not suitable for good deformations. Thus, with EBTLE and SWDC, it is recommended to leave fine skin features to the rendering techniques. As long as the normals of the vertices in the deformed skin are calculated properly, these fine features will be nicely mapped to the skin surface during rendering.

Bibliography

[1] M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, June 2002.

[2] Alias|wavefront. http://www.alias.com/maya.

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

[4] A. Angelidis, G. Wyvill, and M.-P. Cani. Sweepers: Swept user-defined tools for modeling by deformation. In *Proceedings of Shape Modeling International*, 2004.

[5] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408–416, 2005.

[6] A. Aubel. *Anatomically Based Human Body Deformations*. PhD thesis, Swiss Federal Institute of Technology, 2002.

[7] A. Aubel and D. Thalmann. Realistic deformation of human body shapes. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pages 125–135, 2000.

[8] A. Aubel and D. Thalmann. Interactive modeling of the human musculature. In *Proceedings of Computer Animation*, 2001.

[9] F. Aubert and D. Bechmann. Volume-preserving space deformation. *Computer & Graphics*, 21(5):625–639, 1997.

[10] Autodesk. www.autodesk.com/3dsmax.

[11] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Proceedings of ACM SIGGRAPH*, pages 303–308, 1992.

[12] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH*, pages 43–54, 1998.

[13] J. Barbič and D. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Transactions on Graphics*, 24(3):982–990, 2005.

[14] A. H. Barr. Global and local deformations of solid primitives. In *Proceedings of ACM SIGGRAPH*, volume 18, pages 21–30, 1984.

[15] D. Bechmann. Space deformation models survey. *Computer & Graphics*, 18(4):571–586, 1994.

[16] J. Bloomenthal. Medial-based vertex deformation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–151, 2002.

[17] P. Borrel and D. Bechmann. Deformation of n-dimensional objects. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, pages 351 – 369, 1991.

[18] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, 1994.

[19] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, 1996.

[20] S. Capell, M. Burkhart, B. Curless, T. Duchamp, and Z. Popović. Physically based rigging for deformable characters. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 301 – 310, 2005.

[21] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *Proceedings of ACM SIGGRAPH*, pages 586–593, 2002.

[22] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. A multiresolution framework for dynamic deformations. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 41–47, 2002.

[23] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of ACM SIGGRAPH*, pages 67–76, 2001.

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

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

[26] D. T. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Proceedings of ACM SIGGRAPH*, pages 89–98, 1992.

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

[28] S. Coquillart. Extended free form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of ACM SIGGRAPH*, volume 24 of *4*, pages 187–196, 1990.

[29] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999.

[30] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[31] Curious Labs Inc., Santa Cruz, CA. *Poser 5 Reference Manual*, 2002.

[32] G. Debunne, M. Desbrun, A. Barr, and M.-P. Cani. Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation*, 1999.

[33] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Adaptive simulation of soft bodies in real-time. In *Proceedings of Computer Animation*, pages 133–144, 2000.

[34] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proceedings of ACM SIGGRAPH*, pages 31–36, 2001.

[35] P. Decaudin. Geometric deformation by merging a 3d-object with a simple shape. In *Proceedings of the conference on Graphics interface*, pages 55 – 60, 1996.

[36] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Proceedings of Graphics interface*, pages 1–8, 1999.

[37] Digimation. http://www.digimation.com/models/.

[38] F. Dong, G. J. Clapworthy, M. A. Krokos, and J. Yao. An anatomy-based approach to human muscle modeling and deformation. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):154–170, 2002.

[39] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH*, pages 173–182, 1995.

[40] M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[41] S. F. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, 1997.

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

[43] E. Grinspun, P. Krysl, and P. Schröder. Charms: A simple framework for adaptive simulation. In *Proceedings of ACM SIGGRAPH*, pages 281–290, 2002.

[44] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: fast neural network emulation and control of physics-based models. In *Proceedings of ACM SIGGRAPH*, pages 9–20, 1998.

[45] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of ACM SIGGRAPH*, pages 95–102, 2000.

[46] M. Hagenlocker and K. Fujimura. Cffd: a tool for designing flexible shapes. *The Visual Computer*, 14(5-6):271–287, 1998.

[47] K. K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Proceedings of Graphics Interface*, 2003.

[48] M. Hauth, J. Groß, and W. Straßer. Interactive physically based solid dynamics. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 17–27, 2003.

[49] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998.

[50] G. Hirota. *An Improved Finite Element Contact Model for Anatomical Simulations*. PhD thesis, The University of North Carolina, 2002.

[51] G. Hirota, S. Fisher, and A. State. An improved finite-element contact model for anatomical simulations. *The Visual Computer*, 19(5):291–309, 2003.

[52] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Proceedings of Computer Animation*, 2001.

[53] G. Hirota, R. Maheshwari, and M. C. Lin. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 234 – 245, 1999.

[54] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Proceedings of ACM SIGGRAPH*, volume 26 of *2*, pages 177–184, 1992.

[55] J. Hua and H. Qin. Scalar-field-guided adaptive shape deformation and animation. *The Visual Computer*, 20(1):47–66, 2004.

[56] J. Huang, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Clustering method for fast deformation with constraints. In *Proceedings of ACM symposium on Solid and physical modeling*, pages 221–226, 2005.

[57] T. J. R. Hughes. *The Finite Element Method : Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[58] 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), 2005.

[59] 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 Conference on Computer Graphics and Applications*, pages 204–213, 2003.

[60] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 131 – 140, 2004.

[61] D. L. James and D. K. Pai. Artdefo: Accurate real time deformable objects. In *Proceedings of ACM SIGGRAPH*, pages 65–72, 1999.

[62] D. L. James and D. K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *Proceedings of ACM SIGGRAPH*, pages 582–585, 2002.

[63] D. L. James and D. K. Pai. Real time simulation of multizone elastokinematic models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 927–932, 2002.

[64] D. L. James and D. K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics*, 22(1):47–82, 2003.

[65] K. Kähler, J. Haber, and H.-P. Seidel. Geometry-based muscle modeling for facial animation. In *Proceedings of Graphics Interface*, pages 37 – 46, 2001.

[66] K. Kähler, J. Haber, H. Yamauchi, and H.-P. Seidel. Head shop: generating animated head models with anatomical structure. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 55–63, 2002.

[67] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of symposium on Interactive 3D graphics and games*, pages 9–16, 2005.

[68] K. G. Kobayashi and K. Ootsubo. t-ffd: free-form deformation by using triangular mesh. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, pages 226 – 234, 2003.

[69] P. G. Kry, D. L. James, and D. K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 153–159, 2002.

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

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

[72] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proceedings of ACM SIGGRAPH*, pages 55–62, 1995.

[73] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH*, pages 165–172, 2000.

[74] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitary topology. In *Proceedings of ACM SIGGRAPH*, pages 181–190, 1996.

[75] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface*, pages 26–31, 1988.

[76] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Proceedings of ACM SIGGRAPH*, pages 309–312, 1992.

[77] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986.

[78] T. Milliron, R. J. Jensen, and R. Barzel. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, January 2002.

[79] L. Moccozet and N. M. Thalmann. Dirichlet free-form deformations and their application to hand simulation. In *Proceedings of the Computer Animation*, pages 93–102, 1997.

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

[81] A. Mohr, L. Tokheim, and M. Gleicher. Direct manipulation of interactive character skins. In *Proceedings of symposium on Interactive 3D graphics*, pages 27–30, 2003.

[82] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–54, 2002.

[83] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics interface*, pages 239–246, 2004.

[84] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 24(3):471–478, 2005.

[85] K. Na and M. Jung. Hierarchical retargetting of fine facial motions. *Computer Graphics Forum*, 23(3):687–695, 2004.

[86] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. In *proceedings of EUROGRAPHICS*, 2005.

[87] L. P. Nedel and D. Thalmann. Real time muscle deformations using mass-spring systems. In *Proceedings of Computer Graphics International*, pages 156–165, 1998.

[88] L. P. Nedel and D. Thalmann. Anatomic modeling of deformable human bodies. *The Visual Computer*, 16(6):306–321, 2000.

[89] V. Ng-Thow-Hing. *Anatomically-Based Models for Physical and Geometric Reconstruction of Humans and Other Animals*. PhD thesis, University of Toronto, 2001.

[90] V. Ng-Thow-Hing and E. Fiume. Interactive display and animation of b-spline solids as muscle shape primitives. In *Eurographics Computer Animation and Simulation Workshop*, pages 81–97, 1997.

[91] V. Ng-Thow-Hing and E. Fiume. Application specific muscle representations. In *Proceedings of Graphics Interface*, pages 107–116, 2002.

[92] J. Noh and U. Neumann. Expression cloning. In *Proceedings of ACM SIGGRAPH*, pages 277–288, 2001.

[93] F. I. Parke and K. Waters. *Computer Facial Animation*. AK Peters, Ltd., 1996.

[94] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. In *Proceedings of ACM SIGGRAPH*, pages 215–222, 1989.

[95] G. Picinbono, H. Delingette, and N. Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical Models*, 65(5):305–321, 2003.

[96] J. C. Platt and A. H. Barr. Constraint methods for flexible models. In *Proceedings of ACM SIGGRAPH*, pages 279–288, 1988.

[97] M. Pratscher, P. Coleman, J. Laszlo, and K. Singh. Outside-in anatomy based character rigging. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2005.

[98] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proceedings of ACM SIGGRAPH*, pages 179–184, 2001.

[99] H. Pyun, Y. Kim, W. Chae, H. W. Kang, and S. Y. Shin. An example base approach for facial expression cloning. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 167–176, 2003.

[100] R. Raffin, M. Neveu, and F. Jaar. Curvilinear displacement of free-form-based deformation. *The Visual Computer*, 16(1):38–46, 2000.

[101] S. H. M. Roth, M. H. Gross, S. Turello, and F. R. Carls. A bernstein-bézier based approach to soft tissue simulation. *Computer Graphics Forum*, 17(3):285294, 1998.

[102] M. H. Sadd. *Elasticity : Theory, Applications, and Numerics*. Academic Press, 2004.

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

[104] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of ACM SIGGRAPH*, pages 151–159, 1986.

[105] H. Seo, F. Cordier, and N. Magnenat-Thalmann. Synthesizing animatable body models with parameterized shape modifications. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 120–125, 2003.

[106] H. Seo and N. Magnenat-Thalmann. An automatic modeling of human bodies from sizing parameters. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 19–26, 2003.

[107] M. Simmons, J. Wilhelms, and A. V. Gelder. Model-based reconstruction for creature animation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 139 – 146, 2002.

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

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

[110] P.-P. J. Sloan, C. F. Rose, and M. F. Cohen. Shape by example. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 135–143, 2001.

[111] I. M. Smith and V. Griffiths. *Programming the Finite Element Method*. John Wiley & Sons, 2004.

[112] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399 – 405, 2004.

[113] W. Sun, A. Hilton, R. Smith, and J. Illingworth. Layered animation of captured data. *The Visual Computer*, 17(8):457–474, November 2001.

[114] J. Teran, S. Blemker, V. Ng-Thow-Hing, and R. P. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 68 – 74, 2003.

[115] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):317–328, 2005.

[116] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proceedings of ACM SIGGRAPH*, volume 22, pages 269–278, 1988.

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

[118] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of ACM SIGGRAPH*, volume 21, pages 205–214, 1987.

[119] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Transactions on Visualization and Computer Graphics*, 8(6):41–51, 1988.

[120] M. Teschner, B. Heidelberger, M. Müller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In *Proceedings of Computer Graphics International*, pages 312–319, 2004.

[121] D. Thalmann, J. Shen, and E. Chauvineau. Fast realistic human body deformations for animation and vr applications. In *Proceedings of Computer Graphics International*, pages 166–174, 1996.

[122] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of ACM SIGGRAPH*, pages 335–342, 1999.

[123] R. Turner and E. Gobbetti. Interactive construction and animation of layered elastically deformable characters. *Computer Graphics Forum*, 17(2):135–152, 1998.

[124] R. Turner and D. Thalmann. The elastic surface layer model for animated character construction. In *Proceedings of Computer Graphics International*, pages 399–412, 1993.

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

[126] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. N. Metaxas, A. M. Elgammal, and P. Huang. High resolution acquisition, learning and transfer of dynamic 3d facial expressions. *Computer Graphics Forum*, 23(3):677–686, 2004.

[127] J. Wilhelms. Modeling animals with bones, muscles, and skin. Technical Report UCSC-CRL-95-01, University of California, Santa Cruz, CA95064, 1994.

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

[129] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum*, 20(3):349–358, 2001.

[130] Y. Zhuang and J. Canny. Real-time simulation of physically realistic global deformations. In *Proceedings of SIGGRAPH'99 Sketches and Application*, August 1999.

[131] O. C. Zienkiewicz. *The Finite Element Method*, pages 174–176. McGraw-Hill, 1977.