Interactive Character Deformation Using Simplified Elastic Models

Zhiping Luo

Cover design by Zhiping Luo
ISBN xxx-xx-xxx-xxxx-x

Interactive Character Deformation Using Simplified Elastic Models

---

Interactive Character Deformation Using Simplified Elastic Models

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op maandag 11 januari 2016 des ochtends te 12.45 uur.

door

Zhiping Luo

geboren op 13 maart 1986
te Jiangxi, China

Promotor:	Prof. dr. R.C. Veltkamp
Co-promotor:	Dr. ir. A. Egges

The work described in this thesis has been supported by the Dutch national program COMMIT.

Contents

Part I

Introduction

Introduction

1

During the last two decades, we have observed that many 3D animation movies fetched large audience [BoxOfficeMojo] partly due to the interesting deformations they provided. Perhaps the most successful series are, to name just a few, **Madagascar** (2005, 2008, 2012) [DreamWorks b], **Kung Fu Panda** (2008, 2011) [DreamWorks a], **Shrek** (2001, 2004, 2007, 2010) [DreamWorks c], and **Ice Age** (2002, 2006, 2009, 2012) [20th-CenturyFox]. The deformations in movies are generated in an off-line fashion. Thus, artists can employ some elegant but computationally expensive techniques to produce visually interesting deformations.

Deformations are also found in computer games since a long time. Since speed is a critical concern, the game characters are typically low resolution polygonal models and deformed using a linearized skeletal skinning scheme. Thus, there are deformation artifacts likely to be perceived. A search for skinning variants that can reduce the number of artifacts without compromising speed is still in progress.

In virtual human modeling, the major concern for deformation is the realism. The skin deformation should be not only realistic, but also reflect the motions of underlying muscles and soft tissues. To develop such a deformation method, interdisciplinary knowledge has to be elegantly combined, including that of human anatomy, geometric modeling, Newton's second law of motion and so on.

Nowadays, visually pleasing or physically accurate deformations are increasingly demanded in the aforementioned scenarios. This is mainly attributed to the commodity powerful computers and the better realism or visuals provided by deformations. The former fact provides a solid testbed such that virtual models can fast mimic the material dynamics of real-world objects as well as high-quality rendering, visualization of the synthesized deformations. The latter fact is the direct driver, pushing people both in research and industry community to strengthen the visual effects via providing high-quality deformations as best as they can.

The Computer Graphics community has been researching efficient deformation methods, skinning schemes and authoring pipelines, so that deformable bodies and characters can be efficiently incorporated into computer games, animation and other graphics applications. A

perfect deformation technique should (i) preserve shape details and volume, (ii) support skin and interior dynamics, and (iii) provide artist-friendly control that is consistent with an existing rigging pipeline for instance.

This thesis presents results of several novel deformation approaches we have developed to show obtained improvements on character deformation. First, we review the most related methods in the first four sections of this chapter, each of which covers a branch of research on deformation.

1.1 DIRECT SKINNING

Skinning, often referred to as skeletal skinning, is to bind the character skin, often represented by a polygonal mesh, to a hierarchical set of interconnected rigid bones. The binding is established once the influence weights have been specified. The weights should support that a bone will affect a set of vertices in an anatomical and intuitive manner. Artists transform relevant bones to produce desired deformations, and, in practice, they usually create user-friendly rig controls to reduce the man effort. The modern approach of specifying bone transformations is to use a motion capture pipeline, which is the process of recording actions of human actors and using that information to animate virtual characters. Through skinning, character skin deformations are represented as data in the form of bone-vertex weights and bone transformations rather than directly the vertex displacements, thereby largely compressing the animation data to be stored.

Linear blend skinning (LBS) is the simplest and fastest skinning scheme, which combines the transformation matrices with per-bone weights for each vertex. LBS still dominates the practical usage, especially in computer games due to its efficiency, though it exhibits shape collapses near joints (the well-known *candy-wrapper* or *joint-collapse* in cases of large rotations at joints such as shoulders. This is because a weighted combination of the transformation matrices results in inaccurate averages of rotations in 3D. Such LBS artifacts should be suppressed in order to provide a high degree of realism in skin deformations.

One possibility is to expand the expressive power of LBS via providing additional weights per bone [Wang & Phillips 2002, Merry et al. 2006] or calculating new transformations from original transformations typically using nonlinear procedures [Mohr & Gleicher 2003, Kavan et al. 2009, Le & Deng 2013]. The new date is computed in a least-

squares sense, hence a set of example poses has to be provided. The exemplars, however, are often not readily available, more or less hindering the practical usages.

A more practical solution is to use a rotation-based blending that results in intrinsic averages of rotations in 3D, thereby suppressing the candy-wrapper artifacts. The accurate averaging, however, comes at the price of reduced speed due to the nonlinear parameterizations of rotations as unit quaternions [Kavan & Žára 2005], or dual quaternions[Kavan et al. 2008] for instance. Dual quaternion also effectively handles the translational component of the transformation in addition to the rotational component, hence dual quaternion skinning (DQS) is an ideal alternative to LBS. The DQ blending can be linearized as multiple linear blending operators [Kavan et al. 2009] to reduce the runtime cost. To accommodate interesting, expressive character poses, which involve extremely large rotations such as large bends and twists, the *differential blending* [Öztireli et al. 2013] instead of conventional rotational blending provides a solution. In this scheme, differential transformations are defined to represent the whole, large transformation so that an intrinsic averaging is applied to small rotations. This method also overcomes discontinuities arising in complex regions, where vertices are influenced by multiple bones.

Drawbacks of LBS can be alternatively reduced using morphing. *Pose Space Deformation* (PSD) [Lewis et al. 2000] is a model of that class of techniques. Vertex displacement offsets in the object local coordinate frames are computed as pose corrections, between the LBS surface and the desired surface that is resculpted by a modeler or animator. The desired deformation is associated with a set of parameters including joint configurations that define the corresponding pose. One or more deformations in the pose space are then interpolated by a scattered data approach based on radial basis functions. The de facto interpolated data is the displacement offsets. Storing the displacements per pose in a large pose space is a memory inefficient approach, thereby making the hardware rendering inefficient. *EigenSkin* [Kry et al. 2002] enables a hardware implementation based on Principal Component Analysis (PCA) [Jolliffe 2002]. By exploiting the significant redundancy that exists in pose displacements, and decomposing the model into joint-based domains that are learned from deformation influences obtained using physical simulations, a reduced set of error-optimal eigenbases for describing the displacements is constructed. The coefficients of the eigenbasis instead of the displacement offsets are interpolated at runtime, resulting in much lower memory consumption.

Skeletal skinning has inherent limitations due to the use of hierarchical, interconnected rigid bones referred to as *handles*. The influence weights per bone (bone weights) do not vary much over a skin of interest but do at joints. Thus, conventional blending using bone weights effectively bends the limbs. This form of rigid bending is highly desired for articulated deformable characters. However, many short bones are required for stretching, twisting, and supple deformations which need higher degrees of freedom of the manipulations, thereby increasing the skinning complexity. So, rigid bones as handles would limit the space of deformations possible to LBS or DQS.

Generalized LBS schemes replace the rigid bones by other types of handles in order to expand the space of deformations. Curve skeleton [Yang et al. 2006, Forstmann et al. 2007] supports a wide range of (expressive) poses, but coming at the price of requiring special rig controls that are inconsistent with a traditional rigging pipeline. For example, the *line-of-action* [Guay et al. 2013, Öztireli et al. 2013] concept from 2D sketching animation is implemented to support the rigging of curve skeleton. The line-of-action curves capture the characters overall motion and balance, and naturally match the curve skeleton to provide a guide to the artist for rigging.

The cage which is a coarser control mesh enclosing the subject surface is also a widely used handle type. Cage-based deformation, where the displacement of a mesh vertex is computed as the barycentric coordinates interpolation of its nearby cage vertices, provides direct and precise manipulations, hence it is well suited for bulging and thinning. Tedious tasks in practice are to establish a proper cage and then manipulate it. The coordinates, used as weights for summing the deformed cage vertices in the deformation process, are critical to accurate deformations. Several coordinate functions are proposed, leading to different generalized barycentric coordinates presenting their own advantages and disadvantages. The Mean Value Coordinates (MVC) [Ju et al. 2005] which are derived via mean value interpolation, though, are popular to use, they may contain negative values such that inevitable artifacts may appear. Harmonic Coordinates (HC) [Joshi et al. 2007] are non-negative, but do not have closed-form expressions as the coordinate function is harmonic. Thus, to compute them is difficult. Green Coordinates (GC) [Lipman et al. 2008] have closed-form expressions and shape-preserving property. They are defined on vertices as well as the faces of the cage, thereby requiring larger number of terms (both normals and displacements) of the summing than MVC and HC.

A recent trend is to use point handles. The starting motivation is

that points are perhaps the easiest handles to position. In contrast, to embed a skeleton into a character volume is more difficult, as transforming a joint would shift its incident bones outside the volume due to the hierarchy of skeleton. LBS with point handles is well suited for handling twisting, stretching, and supple deformations, which are difficult to achieve by skeleton-based LBS. This is because the per point handle influence weights have the characteristics: they vary much over a limb. Computed for point handles by the *Bounded Biharmonic Weights* method [Jacobson et al. 2011] which computes the influence weights via minimizing a Laplacian energy subject to bound constraints, the weights satisfy the properties critical to accurate deformations: locality, sparsity, and smoothness.

Several works have demonstrated the effectiveness of using point handles in character deformation. These skinning schemes use a hybrid rig controls composed of point handles, rigid bones and or cages [Jacobson et al. 2011, Jacobson & Sorkine 2011, Kavan & Sorkine 2012]. However, for the simplicity of skinning, it is worth researching a fully point handles based skinning scheme.

In the case where a vertex is influenced by many (normally, > 4) handles, the skinning model is associated with dense-weights per handle. A weight reduction scheme (e.g. [Landreneau & Schaefer 2010] or [Le & Deng 2013]) can be employed to reduce the number of non-zero values of the weights (normally represented as a vector) per handle, so that not only making the hardware rendering efficient, but also largely accelerating the computations involving blending on CPU. The speed ups come at the price of giving rise to only an approximation of the original deformation.

Modern research on skinning seeks an automatic rigging pipeline: automatic skeletonization from a static shape [Baran & Popović 2007, Shapira et al. 2008, Jacobson et al. 2014], automatic weights [Baran & Popović 2007, Jacobson et al. 2011, Dionne & de Lasa 2013], and automatic specifications of the degrees of freedom [Gleicher 1999, Shi et al. 2007, Jacobson et al. 2012]. Encouraging results specific to humanoid shapes have been obtained, but to achieve a fully automatic pipeline still remains an open question as there are indeed intractable problems [Jacobson et al. 2014]. For example, the skeleton is embedded sub-optimally into the volume due to the large degrees of freedom of locating the joints.

The automatic skinning method based on *skinning decomposition* [James & Twigg 2005, Kavan et al. 2010] is more attractive than the automatic methods mentioned above, which is an algorithm that learns a LBS

model from a mesh sequences. By imposing orthogonal constraints on bone rotations [Le & Deng 2012], the resulting LBS model is compatible with rigid bones suitable for motion editing. By removing redundant bones, and smoothing the weights using a rigidity regularization term [Le & Deng 2014], the skeleton and joint positions were robustly extracted, thereby resulting in an artist-friendly rig control.

Direct skinning does not support nonlinear elastic deformations. As a result, clearly visible distortions may appear at vertices that are located close to joints in the skeleton. This is because these vertices are bound to multiple bones and then the closed-form blending for each of that is performed. The nonlinear elastic deformations could be mimicked by optimizing the influence weights [Kavan & Sorkine 2012], hence delivering results visually similar to a nonlinear elastic deformation method. This extension of skinning lets professional rigging artists enjoy the convenience of LBS, while the joint-bulging artifacts are suppressed. However, the post-optimization of weights is achieved by energy minimization, thus the method is no longer a direct skinning method.

1.2 ELASTIC ENERGY

Methods that cast deformations as an elastic energy minimization problem are employed to produce shape details preserving deformations. Without loss of generality, the energy is referred to as the rigidity energy constructed over the shape. The energy minimization is a form of quadratic variational minimization problem [Botsch & Sorkine 2008]. The global unique minimum is achieved by solving a linear system of equations, hence such methods are called linear.

Though surface deformation problem is inherently non-linear, linear variational deformation methods are attractive due to their robustness and computational efficiency. Linear methods, however, provide an approximation of nonlinear deformation and thus cannot handle large rotations. Thus, local rotations of a surface element, which is often a vertex-based definition (the element consists of the faces incident to a vertex), have to be incorporated into the objective function. Commonly, the local rotation of a vertex is deduced by Singular Value Decomposition (SVD) [De Lathauwer et al. 1994] of the matrix built based on one-ring neighbors of that vertex. This is a nonlinear procedure, so the rigidity energy turns out to be nonlinear accordingly. The Laplacian-based energy formulation used for as-rigid-as-possible surface modeling [Sorkine & Alexa 2007] is a model of nonlinear methods. This

formula is conceptually easy to understand, easy to implement, and leads to no translation-insensitivity.

For intuitive manipulations, the objective function is subject to positional constraints. Users can fix certain mesh vertices, and prescribe the positions of some others at runtime. However, large rotational deformations would lead to undesired volume changes.

To preserve the volume throughout the deformation, global or local volume constraints need to be defined [Huang et al. 2006, Ben-Chen et al. 2009]. Such constraints are often complex to design, and are not as-intuitive-as positional constraints to edit on-the-fly. Thus, many surface-based variational energies with volume constraints are not able to preserve the shape locality and the volume simultaneously. In general, the surface-details preservation comes at the price of a loss of volume. For example, the method proposed by Lipman et al. [Lipman et al. 2007], first guaranteed shape details preserving deformations, and then scaled the frames which represent the mesh to compensate for volume changes using a local volume analysis, which links the volume of an object to its surface curvature.

Surface-based energy minimization is bound to the surface representation, hence it may encounter geometric issues such as degenerated triangles. Also, surface-based energy might not be as natural as a volumetric discretization with a physically plausible elastic energy. By directly formulating the elastic energy over the volume domain, the complexity of elasticity modeling is reduced.

There are remarkable advantages when using a volumetric representation. On the one hand, the elastic energy is free from volume constraints. On the other hand, the volumetric energy easily draws from the surface-based objective function. Only the definition of discrete element and associated weights need to be adapted.

However, vulnerabilities also exist. A volumetric discretization usually results in denser data to be processed than the original surface. Volumetric deformations also have to be projected back to the original surface using a mapping function. The function should support high-order smoothness, but often coming at the expense of increased computational cost. For instance, the volumetric energy using voxelization proposed by Botsch et al. [Botsch et al. 2007a] is associated with triharmonic radial basis functions [Sieger et al. 2012] used for scattered data interpolation. These basis functions are highly, computationally expensive, though they provide high-quality and C^2-continuous deformation fields.

For incompressible objects, the volume is ideally held constant throughout the deformation. The above methods based on variational minimization generally cannot realize that. If accurate volume preservation is demanded, to date the best way is to apply geometric volume corrections. A displacement vector field, which could be computed automatically [von Funck et al. 2008], or controlled manually via one-dimensional profile curves [Rohmer et al. 2009], is applied as constraints on vertices to conserve the volume after deformation. The accuracy comes at the price of more user interactions with respect to variational deformation. Users have to specify the weights used to control the degrees of volume preservation when dragging the profile curves [Rohmer et al. 2009].

1.3 ELASTICITY SIMULATION

Both elastic energy and elasticity simulation produce elastic deformations, but the latter approach involves constructing dynamic models of animated objects and computing their motions via physical simulation. One goal of simulation is to provide physical accuracy and correctness, thereby leading to physically plausible realism of the interior dynamics and the quasistatic skin.

According to the elastic theory, the object's material has the property that the object is deformed under external forces and can regain its original shape after the forces are removed. The external force applied on a specified area is known as *stress*, and the amount of deformation is called the *strain*. This theory is employed to construct the ordinary differential equations governing the material dynamics, namely Newton's second law of motion. The dynamics of a deformable model are simulated by updating the time dependent coordinates of points in the model. This is obtained by numerically solving (i.e. integrating) the resulting system of algebraic equations from the differential equations. Stability and accuracy are two main criteria we need to consider when choosing a suitable time integration scheme. The scheme is in fact an approach used in numerical analysis for obtaining numerical solutions of time-dependent ordinary and partial differential equations. In general, explicit schemes that advance the state of the simulation system from the current state are fast to implement and compute while they are only conditionally stable. Implicit schemes, which update the system by solving an equation involving both the current state and the state at a latter time, are stable for arbitrarily large time steps, but this gain comes at the price of additional computational overhead.

To realize a computer implementation, the differential equations governing the material dynamics need to be discretized as they describe the mechanical behavior of materials as a continuous mass (continuum mechanics [Fung 1977]). There are typically two methods discretizing the continuum model, namely the mass-spring system and the Finite Element Method (FEM) [Cook 1994].

A mass-spring system constructs the equations of motion on discrete models. These models consist of point masses connected together by a network of massless springs, which are regularly spaced in a lattice [Terzopoulos & Waters 1990], or a grid [Liu et al. 2013] for instance. In common, the springs are modeled as being linearly elastic according to the classic Hooke's law, which is a principle of physics that states that the force needed to extend or compress a spring by some distance is proportional to that distance. It is also possible to reduce oscillations by viscoelastic springs that damp out relative motions. Time-dependent dynamics of Hookean springs can be integrated quickly [Liu et al. 2013] though with a loss of precision within an acceptable range, thereby supporting real-time simulations.

Using mass-spring systems to conserve the volume is difficult, since normally only the spring lengths are preserved. FEM is more suitable for this task. The object is viewed as a continuously connect volume whose domain is discretized using a mesh composed of 3D polyhedra called elements. Then, FEM is employed to turn the Partial Differential Equations (PDE) of continuum mechanics into ordinary differential equations over a finite number of elements.

In FEM, the material properties of the deformable object need to be defined. Linear materials that have a linear stress-strain relationship are commonly used in graphics due to their computational efficiency. They have also been exploited for large rotational deformations using the so-called *Stiffness Warping* [Müller et al. 2002, Müller & Gross 2004] method. The basic idea is to warp the constant stiffness matrix of the resulting system of equations used in linear approaches along a rotation field. Significant, additional overload mainly lies in polar decompositions, used to compute the rotations.

The model would distort badly for large deformations when using materials with geometric linearity. The St. Venant-Kirchhoff (StVK) material, which has a linear stress-strain relationship with the nonlinear *Green-Lagrange strain* [Barbič & James 2005], has been defined to suppress such distortions. For its applications in real-time applications, the method called *dimensional model reduction* [Barbič & James 2005] is performed on deformable models with StVK material, result-

ing in simple internal force polynomials associated with precomputed coefficients. Large rotational dynamics are then integrated at extremely fast rates using an integration scheme called *subspace implicit Newmark integrators* [Barbič & James 2005].

Domain decomposition is an alternative solution for simulating large deformation dynamics with a linear model. For example, the simulation model of an articulated deformable character is decomposed into bone-associated domains, each which uses its local, rotated coordinate frame to compute the linear elastic forces [Capell et al. 2002]. However, it leads to discontinuity at the boundary of contact domains. A better solution is to estimate a subspace deformation (also known as dimensional model reduction) model [Kim & James 2011] for each domain, and then deal with the seam artifacts via coupling domains by springs that result in penalty-based coupling forces [Kim & James 2011].

Element inversion [Irving et al. 2006] can occur in large deformation, even when simulating incompressible material, since one typically cannot conserve volume for each individual element. Since constitutive models for real materials are meaningful only for uninverted material [Irving et al. 2004], the inversion issue should be addressed. The mechanism proposed by Irving et al. [Irving et al. 2004] can be employed to handle inverted elements. It first computes a diagonalization of the deformation mapping to determine the direction along which a given element is inverted, and then extends the FE model past the origin into the inverted regime to obtain valid forces for inverted elements. These forces act to uninvert the element so that the deformable object is guided towards a desired final shape.

Finite element simulation works well for small time steps but is unreliable for large time steps at or near the frame rate, largely hindering potential accelerations. There is growing interest in performing simulations at larger time steps [Su et al. 2013, Gast & Schroeder 2014].

Both Mass-spring and FEM are widely used in virtual human modeling and simulation: torso model for laugh animations [DiLorenzo et al. 2008] and breath simulation [Zordan et al. 2004] where spring are muscle elements as force actuators, FEM based comprehensive upper body model [Lee et al. 2009], and human neck modeling [Luo et al. 2013] based on springs and FEM. In general, the modeling is anatomically inspired and often requires interdisciplinary knowledge. Typically, a reduced set of anatomical structures including bones as rigid bodies, and tissues as deformable bodies are incorporated into the modeling.

1.4 SPACE DEFORMATION

Space deformation was first introduced by Sederberg and Parry [Sederberg & Parry 1986], which is a universal deformation technique working with a wide range of object representations as it does not compute deformations directly of the surface. The basic space deformation method called *Free Form Deformation* (FFD) defines a lattice with a rather small number of control points that encloses the target object. By manipulating the control points, the embedded geometry is deformed smoothly. However, the regularity of the lattice makes FFD less flexible to manipulate the concave regions of the embedded geometry.

Cage-based deformation uses a more general control polyhedron to enclose the subject model in a tighter fashion. The polyhedron has a better match of degrees of freedom to the embedded geometry. Deformations from the cage are propagated back to the geometry by interpolation based on barycentric coordinates that are generalized to polygons and polyhedra. The interpolation is often an affine sum of the deformed cage vertices. The coordinates are ideally non-negative and do not possess a local extremum in order to allow intuitive control in the deformation process. For fast computations, the coordinates should also have closed-form formulas. For high-quality deformations, the coordinates should lead to space deformations with a shape-preserving property [Lipman et al. 2008], or provide more local and finer control on deformation [Zhang et al. 2014]. Local Barycentric Coordinates [Zhang et al. 2014], which select for each mesh vertex a small set of cage vertices rather than all as MVC, HC and GC mentioned in the previous section, provide local control, and result in a compact representation of weights that reduces the memory footprint. Thus, they allow for fast deformations without performing weight reduction (e.g. [Landreneau & Schaefer 2010]) on coordinates.

Scattered data interpolation based on radial basis functions (RBF) is an alternative approach to compute space deformations. RBF supports smoothness but has no shape-preserving property. A trade-off between speed and smoothness is often required. For example, the compactly supported Gaussian functions [Luo et al. 2015] result in a sparse linear system to be solved. The tri-harmonic radial basis functions [Botsch et al. 2007a] support higher-order smoothness but give rise to a dense linear system which is expensive and difficult to solve.

Space deformations are considered as a remarkable accelerating strategy in computationally expensive deformation methods, in particular the nonlinear, volumetric elastic energy (e.g. [Botsch et al. 2007a]), and

the Finite Element models (e.g. [Barbič & James 2005]). By resorting to space deformation, the relatively coarser representation of the subject model carries out the energy minimization, or simulation, while the original surface deformations of the subject model are maintained by efficient space deformations.

Shape matching that establishes the correct correspondences between two shape representations [Faugeras & Hebert 1983] also benefits from the use of space deformation. The per-cell complex, used to compute the least-squares transformations between its current configuration and its goal configuration, is not bound to the original model representation. In contrast, the complex can be a structure that makes the computations of transformations robust. The *Fast Lattice Shape Matching* (FastLSM) [Rivers & James 2007a] method, used to to simulate volumetric, large-deformation dynamics, heavily leverages the embedding lattice which neighborhood structure is intuitive and easy to access, thereby reducing the complexity of defining and summing the shape matching regions.

1.5 CONTRIBUTION

Dedicating four years of research to the topics related to character deformation has been a joy. The experience has bought me many insights and a much richer understanding of the science behind these topics. Partly because of the exploratory nature of my PhD trajectory, the research topics and contributions in this thesis are diverse.

An early PhD research was a detailed overview of existing work regarding shape deformation and character deformation. In that review, the various methods described in research publications on (character) deformation were classified into different aspects. This provides a blueprint for PhD research to me as a novice researcher in the beginning.

In addition to the common smooth skinning, there are a number of methods that have been employed in character deformation. In recent years, a growing trend is to improve the deformation realism in terms of shape details and volume preservation, or provide the extra elasticity dynamics. Thus, research has increasingly stepped into the fields of energy minimization and physics-based simulation. My PhD work resulted in scientific contributions to these fields, which are briefly summarized below.

We first explored physics-based simulations of virtual humans in order to provide lifelike character deformations. In deformable character

animation, the skin deformation of the neck is important to reproduce believable facial animation, and the neck also plays an important role in supporting the head in balance while generating the controlled head movements that are essential to so many aspects of human behavior. Although the head-neck part plays an important role in animation, it is largely overlooked in computer graphics. Additionally, in practice we found its anatomy highly complex. Thus, developing a virtual model of head-neck for simulation is extremely challenging. By consulting a detailed 3D human musculoskeletal model and the neck biomechanics reported in literature, we propose a human-neck model ready for simulation. The modeling involved a lot of components, including multi-body system, mass-spring network, and meshing that converts polygonal meshes into the representations required by Finite Element modeling. Algorithms have also been researched to deal with the geometric issues such as holes and degeneration of the polygonal meshes. As a physics-based simulation model, there are inevitably quite a lot of necessary parameters need to be specified. Our solution is to adaptively adjust the parameters reported in existing biomechanical neck models. To perform simulations on a modern laptop, simplifications are carefully made without notable downside to the realism. The results demonstrated the effectiveness of the model. However, physics-based simulation is computationally expensive and has high implementation complexity. Thus, we researched methods that are also able to produce elastic deformations as simulation while they are much easy to realize. Our methods are based on variational energy minimization, which is typically the way for elastic deformation. To preserve both shape details and the object's volume throughout the deformation, we present a volumetric energy based on lattice which facilitates the implementation due to its regularity. Space deformations were developed such that only the hull voxels but not all need to be processed, as the mesh vertices are contained only by these voxels. Our volumetric method has versatile performance in various application scenarios such as games and virtual human simulation, as we provided various space deformations that trade off speed against the realism.

To this end, our research on elastic deformation only contributed to the improvements of deformation quality. For character deformation, there are other aspects we must consider, notably localized deformation changes of limbs and the usability of the rig control. To employ our deformation methods in character deformation, a novel method and scheme are proposed, which will be presented in Chapter 5 and 6 of this thesis. The method was to compute space deformation, which

is a typical accelerating strategy used in computationally expensive deformation methods, especially the Finite Element models and the nonlinear variational energies. By partitioning the character into multiple bone-associated domains, and attaching each to a linear system that smoothly interpolates surfaces, our space deformation method guarantees the smoothness and avoids interplay between limbs. Thus, using our method, elastic deformations will be produced and the characteristics of character deformation will be also supported. For animation, we need an artist-friendly rig control that not only largely reduces the rigging effort, but also retains the space of deformations supported by the aforementioned deformation methods. Otherwise, to manipulate the character skin and pose the character for a sequence of deformed skin will be very time consuming. We developed a novel skinning scheme that elegantly combines point handles, blend skinning, and nonlinear variational energy minimization. Point handles are much easier to design than skeleton and cages, while skinning with them the space of deformations is expanded due to the introduced elastic deformations. The new scheme has been examined by a variety of elaborate experiments and employed in character animation.

All the methods presented in this thesis are easy to implement and have high performance. As such, they are attractive to real-time graphics applications such as computer games. Thus, the methods may have immediate effect in boosting the deformation quality of such applications that currently endure poor user experience on deformation, partly because the linear blend skinning as the dominant deformation technique in games exhibits shape-collapse artifacts at the joints.

1.6 THESIS OUTLINE

This thesis is divided into four parts. Part I contains this introduction and the preliminaries. We present the deformation methods in the main two parts, Parts II and III, of this thesis, each of which covers a different aspect of research on deformation with a particular focus on character animation. Part IV contains some concluding remarks and an overview of future work.

Part II – Elastic Deformation

This part of this thesis describes several methods that reproduce skin deformation dynamics in virtual human modeling, via simulating elas-

tic materials, or preserve shape details and volumetric properties throughout the deformation using geometrically-motivated elastic models. The content is based on the results from our research on elastic deformation.

Chapter 3 describes a biomechanical human neck model for simulating the neck dynamics, which are important to reproduce believable facial animation, and support the head in balance while generating the controlled head movements essential to so many aspects of human behavior. In our model, relevant anatomical structures available in a 3D model of human musculoskeletal system are modeled as deformable bodies or linked rigid bodies. These virtual bodies are then coupled using soft constraints via elastic springs. Based on our model, both underlying muscular dynamics and skin deformations of the neck can be simulated.

Chapter 4 describes two methods based on voxels as the volumetric representation of an object. With voxels, the methods are easy to implement, and have stable performance. A variety of results show the promising prospects of their applications in surface deformation.

Part III – Character Deformation

This part of this thesis contains the results from our research on adapting the methods of elastic deformation to character deformation and rigging.

Chapter 5 describes how to compute space deformations based on domain-decomposition when employing variational energies, or Finite Element models to increase the realism of character deformation. Our method supports smoothness and local controllability of deformations, and can achieve interactive interpolating rates.

Chapter 6 describes a novel rig control based on point handles for character skinning. We design a deformation metaphor which is well suited to point handles, resulting in a novel skinning scheme fully based on point handles. Our scheme is as efficient as skeletal skinning for bending limbs, while it is more flexible by allowing versatile character posing on-the-fly.

Preliminaries

2

In this chapter, the fundamentals necessary to the work of this thesis are presented. Each section covers the brief mathematics essential to a research line of blend skinning, smooth data interpolation, elastic energy, and physics-based elasticity simulation, respectively.

2.1 SKELETAL SKINNING

A skeletal model consists of B bones, and a skin mesh S containing N vertices $\{v_1, ..., v_N\}$ in rest pose. Each vertex v_i is bound to the bones by influence weights, expressed as a vector $\mathbf{w}_i = (w_{i,1}, ..., w_{i,B})$. Given an arbitrary blending technique Φ, when specifying the rigid transformations of bones $\{T_1^f, ..., T_B^f\}$ at frame f, the deformed vertex positions are computed as follows:

$$v_i^f = \Phi(\mathbf{w}_i; T_1^f, ..., T_B^f) v_i \tag{2.1}$$

The simplest and fastest blending is a weighted combination of the transformation matrices (a.k.a. linear blending) in the form of

$$\Phi_{LBS}(\mathbf{w}_i; T_1^f, ..., T_B^f) = \sum_{j=1}^{B} w_{i,j} T_j^f. \tag{2.2}$$

This linear technique results in non-intrinsic averages of rotations in 3D, thereby leading to joint-collapse artifacts in case of large rotational deformations. One common solution is to re-parameterize the transformation by dual quaternion [Kavan et al. 2008] that represents both 3D rotation and translation. The resulting dual quaternions are then blended with influence weights, leading to the final dual quaternion used to transform the vertex.

The following describes how to blend the transformation matrices by dual quaternions. Let T be represented by a 4×4 homogeneous matrix, denoted by $\mathbf{M}$, that performs the rotation, followed by a translation given by $(t_x, t_y, t_z)^\intercal$. Without loss of generality, Euler angles, denoted by $R(\alpha, \beta, \gamma)$, are used to describe the orientation. Thus, $\mathbf{M}$ is written

as follows:

$$\mathbf{M} = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & t_x \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & t_y \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Let $q(\omega, x, y, z) = \omega + xi + yj + zk$ denote the quaternion. The norm of q is $\|q\| = \sqrt{\omega^2 + x^2 + y^2 + z^2}$, hence q is normalized as $q/\|q\|$. The unit quaternion describes the rotation given by ω about the axis given by $(x, y, z)^\intercal$. The routine that converts $\mathbf{M}$ to q is given as follows:

Algorithm 2.1.1: toQuaternion($\mathbf{M}$)

comment: Compute coefficients $\omega,\ x,\ y,\ z$

$Tr \leftarrow \mathrm{tr}(\mathbf{M})$
if $Tr > 0$

$$\textbf{then} \begin{cases} S \leftarrow 2 \cdot \sqrt{Tr} \\ x \leftarrow (\mathbf{M}_{2,1} - \mathbf{M}_{1,2})/S \\ y \leftarrow (\mathbf{M}_{0,2} - \mathbf{M}_{2,0})/S \\ z \leftarrow (\mathbf{M}_{1,0} - \mathbf{M}_{0,1})/S \\ \omega \leftarrow 0.25 \cdot S \end{cases}$$

$$\textbf{else} \begin{cases} \textbf{if } (\mathbf{M}_{0,0} > \mathbf{M}_{1,1}) \wedge (\mathbf{M}_{0,0} > \mathbf{M}_{2,2}) \\ \quad \textbf{then} \begin{cases} S \leftarrow 2 \cdot \sqrt{1.0 + \mathbf{M}_{0,0} - \mathbf{M}_{1,1} - \mathbf{M}_{2,2}} \\ x \leftarrow -0.25 \cdot S \\ y \leftarrow -(\mathbf{M}_{0,1} + \mathbf{M}_{1,0})/S \\ z \leftarrow -(\mathbf{M}_{2,0} + \mathbf{M}_{0,2})/S \\ \omega \leftarrow (\mathbf{M}_{1,2} - \mathbf{M}_{2,1})/S \end{cases} \\ \quad \textbf{else if } \mathbf{M}_{1,1} > \mathbf{M}_{2,2} \\ \quad \textbf{then} \begin{cases} S \leftarrow 2 \cdot \sqrt{1.0 + \mathbf{M}_{1,1} - \mathbf{M}_{0,0} - \mathbf{M}_{2,2}} \\ x \leftarrow -(\mathbf{M}_{0,1} + \mathbf{M}_{1,0})/S \\ y \leftarrow -0.25 \cdot S \\ z \leftarrow -(\mathbf{M}_{2,1} + \mathbf{M}_{1,2})/S \\ \omega \leftarrow (\mathbf{M}_{2,0} - \mathbf{M}_{0,2})/S \end{cases} \\ \quad \textbf{else} \\ \quad \textbf{then} \begin{cases} S \leftarrow 2 \cdot \sqrt{1.0 + \mathbf{M}_{2,2} - \mathbf{M}_{1,1} - \mathbf{M}_{0,0}} \\ x \leftarrow -(\mathbf{M}_{2,0} + \mathbf{M}_{0,2})/S \\ y \leftarrow -(\mathbf{M}_{1,2} + \mathbf{M}_{2,1})/S \\ z \leftarrow -0.25 \cdot S \\ \omega \leftarrow (\mathbf{M}_{0,1} - \mathbf{M}_{1,0})/S \end{cases} \end{cases}$$

The quaternion q is converted back to a 4×4 matrix as follows:

$$\mathbf{M}^{'} = \begin{pmatrix} 1.0 - 2y^2 - 2z^2 & 2yx + 2zw & 2zx - zyw & 0 \\ 2yx - 2zw & 1 - 2x^2 - 2z^2 & 2zy + 2xw & 0 \\ 2zx + 2yw & 2zy - 2xw & 1.0 - 2x^2 - 2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

As the conversion implies, the quaternion cannot represent the translational component of the transformation matrix $\mathbf{M}$ since the last column of $\mathbf{M}^{'}$ is not the original translation vector $(t_x, t_y, t_z, 1)^{\intercal}$.

Let $\hat{q}(\hat{\omega}, \hat{x}, \hat{y}, \hat{z}) = \hat{w} + \hat{x}i + \hat{y}j + \hat{z}k$ denote the dual quaternion. A dual quaternion is also considered as a sum of two ordinary quaternions, $\hat{q} = q_0 + \varepsilon q_\varepsilon$, where q_0 is the non-dual part, q_ε the dual part, and ε a dual unit satisfying $\varepsilon^2 = 0$. To convert M to $\hat{q}$, first, using Algorithm 2.1.1 to construct $q_0 = a + bi + cj + dk$ that describes the rotation, and second, using the following formula to construct q_ε that represents the translation:

$$\begin{aligned} q_\varepsilon &= \frac{\varepsilon}{2.0}(t_x i + t_y j + t_z j) q_0 \\ &= -0.5(t_x b + t_y c + t_z d) + 0.5(t_x a + t_y d - t_z c) i \\ &\quad +0.5(-t_x d + t_y a + t_z b) j + 0.5(t_x c - t_y b + t_z a) k \end{aligned} \tag{2.5}$$

Multiple dual quaternions are blended with influence weights as follows:

$$\begin{aligned} \hat{q}_{blend} &= \Phi_{DQ}(\mathbf{w}_i; \hat{q}_1^f, ..., \hat{q}_B^f) \\ &= \frac{w_{i,1}\hat{q}_1^f + ... + w_{i,B}\hat{q}_B^f}{\|w_{i,1}\hat{q}_1^f + ... + w_{i,B}\hat{q}_B^f\|} \end{aligned} \tag{2.6}$$

This in fact blends the non-dual part and dual parts, respectively. Blended q_0 and q_ε are then normalized by $\|q_0\|$, and denoted by $\overline{q_0} = \overline{a_0} + \overline{b_0}i + \overline{c_0}j + \overline{d_0}k$ and $\overline{q_\varepsilon} = \overline{a_\varepsilon} + \overline{b_\varepsilon}i + \overline{c_\varepsilon}j + \overline{d_\varepsilon}k$ for the sake of convenience in writing.

Let $\overline{\mathbf{v}_\varepsilon} = (\overline{b_\varepsilon}, \overline{c_\varepsilon}, \overline{d_\varepsilon})$ and $\overline{\mathbf{v}_0} = (\overline{b_0}, \overline{c_0}, \overline{d_0})$. The formula that recovers the translation from a dual quaternion is

$$(\overline{t_x}, \overline{t_y}, \overline{t_z}) = \overline{a_0}\overline{\mathbf{v}_\varepsilon} - \overline{a_\varepsilon}\overline{\mathbf{v}_0} + \overline{\mathbf{v}_0} \times \overline{\mathbf{v}_\varepsilon}. \tag{2.7}$$

The dual quaternion is converted to a 4×4 transformation matrix used to transform the vertex in the form of

$$\left|\begin{array}{ccc:c} & \overline{\mathbf{M}^{'}_{3\times 3}} & & \begin{matrix}\overline{t_x}\\ \overline{t_y}\\ \overline{t_z}\end{matrix} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right|,$$

where the rotation matrix $\overline{\mathbf{M}'_{3\times3}}$ is recovered from quaternion $\overline{q_0}$ using Eq. 2.4. Thus, the dual quaternion describes both rotation and translation in a compact representation.

2.2 RBF INTERPOLATION

Scattered data interpolation based on radial basis function (RBF) is a well behaved and efficient interpolation scheme in high dimensional spaces:

i. It guarantees smooth interpolation from a sparse set of point samples.

ii. It allows direct and compact representation of the function parameters (only in a matrix).

iii. It interpolates any desired values in the whole space spanned by the basis vectors.

Thus, RBF interpolation is suitable for computing space deformations in case the target surface is interpolated by the displacements of the enclosing mesh's control points.

Given the value of a scalar valued function $F : \mathbb{R}^3 \longrightarrow \mathbb{R}$ in N distinct discrete points $\{p_i | p_i \in \mathbb{R}^3, i \in \{0, ..., N-1\}\}$, the RBF system learns a smooth interpolation function of F in $\mathbb{R}^3$. Each point $p(x, y, z) \in \mathbb{R}^3$ is evaluated in the form of a weighted sum of N radial basis functions. Without loss of generality, the evaluation is a function in the form of

$$F(p) = \sum_{i=0}^{N} a_i \ker g(\|p - p_i\|) + C \begin{pmatrix} 1 \\ p^\intercal \end{pmatrix}, \tag{2.8}$$

where a_i are weights, $\ker g$ a kernel function taking the norm $\|p - p_i\|$ as input, and $C = (c_0, c_1, c_2, c_3)$ the coefficient vector determining the polynomial term. There are various kernel functions (e.g. Gaussian) and norms (e.g. Euclidean norm) can be used.

Denote a $(N+4) \times (N+4)$ matrix $\mathbf{G}$, a vector $\mathbf{A}$, and a vector $\mathbf{F}$ as follows:

$$
\begin{aligned}
\mathbf{F} &= (F_0, ..., F_{N-1}, 0, 0, 0, 0) \\
\mathbf{A} &= (a_0, ..., a_{N-1}, c_0, c_1, c_2, c_3) \\
\mathbf{G}_{(N+4)\times(N+4)} &= \begin{pmatrix}
\ker g_{0,0} & ... & \ker g_{0,N-1} & (1, p_0) \\
. & ... & . & . \\
. & ... & . & . \\
. & ... & . & . \\
\ker g_{N-1,0} & ... & \ker g_{N-1,N-1} & (1, p_{N-1}) \\
\begin{pmatrix} 1 \\ p_0^\intercal \end{pmatrix} & ... & \begin{pmatrix} 1 \\ p_{N-1}^\intercal \end{pmatrix} & (0,0,0,0)^\intercal \cdot (0,0,0,0)
\end{pmatrix}
\end{aligned}
$$

Equation 2.8 is then rewritten in a compact form of

$$\mathbf{GA} = \mathbf{F}. \tag{2.9}$$

In the concept of subspace projection, the original space is in fact projected onto the space spanned by N basis vectors. The resulting system of equations is solved in a least-squares sense where the error $\mathbf{F} - \mathbf{GA}$ is orthogonal to each basis vector:

$$\mathbf{G}^\intercal(\mathbf{GA} - \mathbf{F}) = 0. \tag{2.10}$$

The coefficients are obtained by computing

$$\mathbf{A} = (\mathbf{G}^\intercal\mathbf{G})^{-1}\mathbf{G}^\intercal\mathbf{F}. \tag{2.11}$$

This is often solved by LU decomposition [Press et al. 1992b], and a unique solution is available.

2.3 VARIATIONAL ENERGY MINIMIZATION

Elastic energy measures how much the object has been deformed from its initial configuration. By minimizing the energy, the local appearance of the surface or object's volume under deformation is preserved. Additionally, the elastic energy minimization is generally designed to generate elastic deformations.

Let $S \subset \mathbb{R}^3$ denote a two-manifold surface, parameterized by a function $\rho : \Omega \subset \mathbb{R}^2 \longrightarrow S \subset \mathbb{R}^3$. By adding a displacement vector $\mathbf{d}(u, v)$ to each point $\mathbf{p}(u, v)$ in the parameterized space, the surface is to be deformed to S'. Commonly, Ω is equal to the initial surface, so that $\mathbf{d}$ is defined on the manifold surface itself.

The classic elastic energy (so-called shell energy) measures the difference between S and $S^{'}$ in terms of stretching and bending, which is

$$E(S, S^{'}) = \int_{\Omega} k_s \|\mathbf{I}^{'} - \mathbf{I}\|_F^2 + k_b \|\mathbf{II}^{'} - \mathbf{II}\|_F^2 du dv, \tag{2.12}$$

where $\|\cdot\|_F$ is the Frobenius norm, k_s and k_b are used to control the resistance to stretching and bending. The first and second fundamental forms, $\mathbf{I}(u,v), \mathbf{II}(u,v) \in \mathbb{R}^{2\times 2}$, and $\mathbf{I}^{'}$ and $\mathbf{II}^{'}$, measure geometrically intrinsic properties of S and $S^{'}$, respectively, such as lengths, areas, and curvatures. However, the minimization of this shell energy is too computationally expensive for interactive applications.

To reduce the computations, the energy E is simplified by replacing the fundamental forms with first and second order partial derivatives of $\mathbf{d}$. The minimization of E is performed by applying variational calculus [Botsch & Sorkine 2008], finally yielding the following formula:

$$-k_s \Delta_S \mathbf{d} + k_b \Delta_S^2 \mathbf{d} = \mathbf{0}. \tag{2.13}$$

Δ and Δ^2 are the Laplacian (second order) and the bi-Laplacian (fourth-order) operator [Hazewinkel 2001] w.r.t. S, respectively. Let us use the notion $\mathbf{d}_x = \frac{\partial}{\partial_x}\mathbf{d}$ and $\mathbf{d}_{xy} = \frac{\partial^2}{\partial_x \partial_y}\mathbf{d}$, then we have

$$\Delta_S \mathbf{d} = \mathbf{d}_{uu} + \mathbf{d}_{vv}, \tag{2.14}$$

$$\Delta_S^2 \mathbf{d} = \Delta(\Delta_S \mathbf{d}) = \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv}. \tag{2.15}$$

In deformation, without loss of generality, the surface S is represented by a polygonal mesh, whose topology is determined by N vertices $\{v_1, ...v_N\}$ and m triangles $\{t_1, ..., t_m\}$, $t_i \in \{1, ..., N\}^3$.

For a numerical solution of Eq. 2.13, a discretization of the differential equations has to be defined. This can be obtained by defining a discretization of the Laplacian operator at the vertex based on finite differences which are the discrete analog of the derivatives, leading to the form

$$\Delta_S(v_i) = \omega_i \sum_{v_j \in \mathcal{N}_1(v_i)} \omega_{ij}(v_i - v_j), \tag{2.16}$$

where $v_j \in \mathcal{N}_1(v_i)$ are one-ring neighbors of v_i (cf. Fig.2.1), and

$$\omega_i = \frac{1}{A_i},\ \omega_{ij} = \omega_{ji} = \frac{1}{2}(cot\alpha_{ij} + cot\beta_{ij}). \tag{2.17}$$

So the Laplacian operator for the whole mesh can be written in a

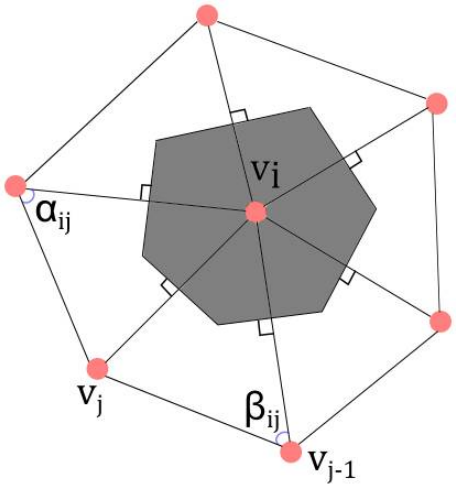

Figure 2.1 One-ring neighbors (red dots) of v_i and the dark grey area is the Voronoi area A_i.

matrix notion $\mathbf{L}_s$, which is

$$(\mathbf{L}_s)_{ij} = \begin{cases} \sum_{v_j \in \mathcal{N}_1(v_i)} \omega_{ij}, & i = j, \\ -\omega_{ij}, & v_j \in \mathcal{N}_1(v_i), \\ 0, & \text{otherwise.} \end{cases} \tag{2.18}$$

There are many variants drawing from the shell energy explained above. One is based on the observation that the energy is minimized if local transformations, denoted by $\mathbf{R}_i$, that occur between S and $S^{'}$ are as-rigid-as-possible. The local transformations are defined on surface elements, each of which is composed of a vertex and its one-ring neighbors. In a simple form, for each v_i, Equation 2.13 is rewritten as

$$\sum_{v_j \in \mathcal{N}_1(v_i)} \omega_{ij}(v_i^{'} - v_j^{'}) = \sum_{v_j \in \mathcal{N}_1(v_i)} \frac{\omega_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)(v_i - v_j). \tag{2.19}$$

The linear combination on the left-hand side is no other than the Laplacian operator applied to all vertices $\mathbf{V}^{'}$

$$\mathbf{L}_s \mathbf{V}^{'} = \mathbf{b}, \tag{2.20}$$

where $\mathbf{b}$ is a $N-$vector containing values of right-hand side expression from Eq. 2.19. $\mathbf{R}_i$ of vertex v_i is derived from the singular value decomposition (SVD) of the covariance matrix $\mathbf{C}_i = \sum_{v_j \in \mathcal{N}_1(v_i)} (\omega_{i,j} e_{ij} e_{ij}^{'\mathsf{T}}) =$

$\mathbf{U}_i\sum_{\mathbf{i}}\mathbf{V}_i^\intercal$, and $\mathbf{R}_i = \mathbf{V}_i\mathbf{U}_i^\intercal$, where $e_{ij} = v_i - v_j$ and e'_{ij} is the edge after reconstruction [Sorkine & Alexa 2007].

In deformation, some vertices are constrained. This typically means to fix certain surface part(s), and prescribe the displacements for other part(s), called *handle region(s)*. Such constraints are incorporated into Eq. 2.20 by erasing respective rows and columns from $\mathbf{L}_S$ then substituting 1.0 or 0 for corresponding variables, and updating $\mathbf{b}$ with known values, which is demonstrated as follows:

$$
\begin{array}{rcl}
\mathbf{L}_{N\times N} & = & \begin{array}{c} \\ 1 \\ 2 \\ \cdot \\ \cdot \\ \cdot \\ N-1 \\ N \end{array}
\begin{array}{c}
\begin{array}{ccccc} 1 & 2 & \cdots & N-1 & N \end{array} \\
\begin{pmatrix}
\sum\omega_{\cdot,\cdot} & -\omega_{1,2} & \cdots & 0 & 0 \\
0 & 1.0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & \omega_{\cdot,N-1} & 0 \\
0 & \omega_{\cdot,2} & \cdots & 0 & 0 \\
\omega_{\cdot,1} & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 1.0 & 0 \\
0 & 0 & \cdots & -\omega_{N,N-1} & \sum\omega_{\cdot,\cdot}
\end{pmatrix}
\end{array}, \\
\mathbf{b} & = & (b_1, v'_2, ..., b_{\lfloor N/2 \rfloor}, ..., v'_{N-1}, b_N)^\intercal.
\end{array}
$$

In this example, the vertex v_2 and v_{N-1} are constrained (marked in green).

Since $\mathbf{L}_S$ is sparse and symmetric positive definite, it can be factored efficiently by the direct sparse Cholesky factorization [Press et al. 1992a]. $\mathbf{L}_S$ only depends on the initial surface, hence the factorization is performed only once. For each new right-hand side, only three times back-substitution has to be performed for each coordinate using the obtained factorization. Finally, the solution results in the optimal positions of unconstrained vertices.

2.4 ELASTIC MATERIALS

Physics-based simulation implies that the motion of a deformable model is governed by Newton' second law

$$\mathbf{M}\ddot{\mathbf{x}}(t) = \mathbf{f}, \tag{2.21}$$

where $\mathbf{x}(t)$ represents the entire vector field of positions at time t, $\ddot{\mathbf{x}}$ the second derivatives, $\mathbf{M}$ a mass matrix, and $\mathbf{f}$ the force vector field. There are also internal forces, computed from discretized continuum elasticity, in additional to external forces due to the deformations.

Continuum elasticity. A deformable object is defined by its rest shape, and coordinates $\mathbf{m}$ of a point in that rest shape are called *material*

coordinates of that point. The object is deformed when forces act on it, such that a point originally at location $\mathbf{m}$ moves to a new location $\mathbf{x}(\mathbf{m})$, which are the *world coordinates* of that point. This leads to the displacement vector field $\mathbf{u}(\mathbf{m}) = \mathbf{x}(\mathbf{m}) - \mathbf{m}$ that actually specifies the deformation.

The elastic strain measures the spatial variations of the displacement field, which can be nonlinear Green-Lagrange strain

$$\varepsilon_G = \frac{1}{2}(\nabla_{\mathbf{u}} + [\nabla_{\mathbf{u}}]^\intercal + [\nabla_{\mathbf{u}}]^\intercal \nabla_{\mathbf{u}}), \tag{2.22}$$

or linear Cauchy strain

$$\varepsilon_C = \frac{1}{2}(\nabla_{\mathbf{u}} + [\nabla_{\mathbf{u}}]^\intercal), \tag{2.23}$$

where $\nabla\cdot$ denotes the divergence. The material law governs the relationship between the internal stress tensor σ and the strain. In computer graphics, the stress is often linearly related to the strain, e.g.,

$$\sigma = \lambda(\mathrm{tr}(\varepsilon))\mathbf{I}_3 + 2\mu\varepsilon, \tag{2.24}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix, λ and μ are material-dependent quantities called Lam*é* constants. This formula actually defines the St.Venant-Kirchhoff material, a class of isotropic material, where the stress-strain relationship only depends on Young's modulus and Poisson's ratio, which are two common coefficients defining the material's mechanical property. This is because the calculations of λ and μ only depend on those two coefficients [Lam].

There are typically two methods used to discretize the continuum elasticity, namely the Finite Element Method (FEM) and the mass-spring network. Both of them have been widely used in computer graphics for simulating deformable objects.

Finite Element Method. First, the volume domain of an object is discretized into a finite number of disjoint elements, and then, the continuous function of the position vector field $\mathbf{x}(\mathbf{m}, t)$ at time t is approximated using the nodal positions of the elements. Within an element e, we have

$$\mathbf{x}_e(\mathbf{m}, t) = \sum_i \mathbf{x}_i(t)\mathbf{b}_i(\mathbf{m}), \tag{2.25}$$

where $\mathbf{x}_i(t)$ is the unknown position of the *ith* element node, and $\mathbf{b}_i(\mathbf{m})$ are fixed nodal basis functions (also called element shape functions [Maas et al. 2012]) that define the relationship between the interior

point's displacement and the nodal positions. Thus, to solve a continuous deformation turns into a solution of nodal positions.

In computer graphics, a simple FEM is used, where both masses and internal and external forces are lumped to the nodes. So, given nodal displacements and nodal basis functions, the strain tensor vector field can be computed (cf. Eq. 2.22 or 2.23), in turn the stress (cf. Eq.2.24). The internal forces acting on nodes are then computed as the derivatives of $\varepsilon(\mathbf{m}) \cdot \sigma(\mathbf{m})$ with respect to the nodal positions. This computation actually results in the stiffness matrix $\mathbf{K} \in \mathbb{R}^{3n \times 3n}$ that relates the forces to nodal displacements

$$\mathbf{f} = \mathbf{K}(\mathbf{u}), \tag{2.26}$$

where $\mathbf{f} \in \mathbb{R}^{3n}$, $\mathbf{u} \in \mathbb{R}^{3n}$, and n is the number of nodes. This relationship is often nonlinear due to the nonlinearity of the more popular Green-Lagrange strain tensor.

After FEM discretization, the dynamics of elastic object can be described as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u}) = \mathbf{f}_{ext}, \tag{2.27}$$

where $\dot{\mathbf{u}}$ are first derivatives of $\mathbf{u}$, and $\mathbf{D} \in \mathbb{R}^{n \times n}$ usually a local Rayleigh damping model [Liu & Gorman 1995] where the damping is proportional to a linear combination of mass and stiffness.

Mass-spring. A mass-spring network consists of a number of point masses connected by elastic springs. The motion of each particle (point mass) is governed by Newton's second law. The internal forces acting on a particle are computed from the changes in lengths of all incident springs.

Mostly, the spring is modeled as linear spring according to Hooke's law [Rychlewski 1984], which has linear elasticity

$$\mathbf{f}_{int} = k(\|\mathbf{x}_{ij}\| - l_{ij}) \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}. \tag{2.28}$$

Here, l_{ij} and $\|\mathbf{x}_{ij}\|$ are rest length and current length of the spring connecting the *ith* and *jth* particles, respectively, and k the spring stiffness.

The Newton's second law of motion for the entire mass-spring network is

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}). \tag{2.29}$$

At a given time t, the state of the mass-spring network is defined by positions $\mathbf{x}$ and velocities $\mathbf{v}$ of the particles.

The motion of deformable object resulting from either FEM or mass-spring is actually a second order system of ordinary differential equations (ODE). The simulation of the material dynamics is to compute time-dependent $\mathbf{x}(\mathbf{m}, t)$ by numerically solving the ODE (i.e. time integration).

Time integration. Given the time step Δt, the integration scheme calculates the time-dependent values of the system at a later time $t + \Delta t$. By iterating this procedure, the motion of the model is simulated.

The simplest way is to use the explicit Euler integration, where the second order system is reduced to two first order equations as follows:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t), \tag{2.30}$$
$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \mathbf{f}(\mathbf{x}(t), \mathbf{v}(t), t). \tag{2.31}$$

Explicit integrator, though, is fast and easy to implement, it is stable only if Δt is smaller than a stability threshold. This is because the method extrapolates a constant right-hand side blindly into the future [Nealen et al. 2006].

An implicit integration scheme is stable at arbitrary Δt (large or small), as it considers the right-hand side as part of the solution process. For instance, a simple form is

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t + \Delta t), \tag{2.32}$$
$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \mathbf{f}(\mathbf{x}(t + \Delta t), \mathbf{v}(t + \Delta t), t). \tag{2.33}$$

However, this gain of stability comes at the price of computational overload at runtime.

Part II

Elastic Deformation

Physics-based Human Neck Modeling and Simulation

3

This chapter describes a method for elastic deformation based on physically and anatomically motivated simulation. The method is particularly employed to animate the skin of the human neck. In addition to the skin dynamics, our method is also able to produce the interior biomechanics due to the underlying musculoskeletal system we modeled. Thus, our method can be used to improve the realism of deformation.

Characters should be capable of expressing their physical states and emotional expressions convincingly in a 3D virtual environment. To achieve that, many efforts are afforded to increase the realism of important aspects, e.g. facial animation [Sifakis et al. 2005, Fratarcangeli 2005, Zhang et al. 2004], tear simulation [van Tol & Egges 2009], skinning of a skeletally-based deformable body [Kavan et al. 2007], and breathing simulation [Zordan et al. 2004].

As an aspect of non-verbal communication in character animation, neck animation should be considered important. First, natural interaction may be compromised by perceiving unrealistic deformations of the neck in facial animation. Second, the neck plays an important role in supporting the head in balance while generating the controlled head movements that are essential to so many aspects of human behavior.

However, to our knowledge the neck animation has been overlooked both in computer graphics and animation except [Lee & Terzopoulos 2006] and [Vasavada et al. 1998]. These two works have investigated human neck simulation where kinematically and biomechanically generated motions are produced, but without explicitly focusing on realism. The proposed models are generally not suitable for skin deformation and therefore more detailed models accounting for the deformation of underlying soft tissues were suggested [Lee et al. 2009] for the upper human body and [Assassi et al. 2012] for the lower body. Nonetheless, they are not suitable for interactive character animation due to the high computational cost.

Current approaches rely on approximations by using techniques such as skeleton based skinning [Kavan et al. 2007] for real-time neck animation. In general, such approximations guarantee a degree of realism

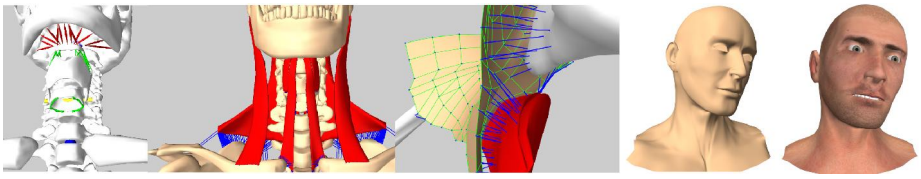

Figure 3.1 We simulate the biomechanics of the human neck and develop a physical skinning approach, and make use of the simulation in character animation. From left to right: skeleton, skeleton-driven muscle deformation, skinning, simulation, animation.

only from the graphical visualization and therefore lack physical accuracy, or like [Lewis et al. 2000] often require considerable manual work. In the case the propagation of muscle motion to the skin layer is visible, modeling muscle deformation based on dense motion capture data has been investigated [Park & Hodgins 2008, Neumann et al. 2013]. Nevertheless, accounting for the physical or biomechanical facts, learning a model of muscle deformation only from captured skin motion cannot guarantee the accuracy of simulating the muscle dynamics. Attempts to alleviate this shortcoming have been given by [Sifakis et al. 2005] who learned the facial muscle functions from skin motion based on a pre-defined physical face model, and [Tan et al. 2012] decided muscle activations in a fibre-driven soft body simulator from prescribed locomotion.

In this chapter, we present a physical model for neck animation. We directly investigate the biomechanics of the neck to solve the complexity of the cervical anatomy and then simulate the skin deformation based on simulated dynamics underneath (see Figure 3.1). The neck anatomy is from the Ultimate Human Model (UHM) data set [UHM] which includes a complete and accurate human musculoskeletal system. A novel approach is proposed to construct the musculoskeletal model that consists of deformable bodies and linked rigid bodies. In more detail, we integrate the deformable bodies with the skeleton using a soft constraint concept and therefore the skeleton drives the muscle deformation. In order to use the simulated dynamics of the underlying system for skin deformation, we bind each skin vertex to one muscle or bone by an elastic spring and this processing is automated. As a result, the skin deforms when the skeleton moves. We are able to simulate the neck at interactive rates because our modeling is based on linear elasticity (continuum) theory which is fast and easy to implement.

We broadly classify the approaches proposed for character animation into the following categories. Our approach falls into the category of physically based methods.

Geometric skinning. For its simplicity and efficiency, skinning of skeletally deformable body is extensively used in real-time character animation, especially the standard solution, linear blend skinning. Popularly, we bind each skin vertex to one or more joints. However, simple skinning will exhibit artifacts including skin collapsing effects. A few alternatives have been approached, e.g. [Kavan et al. 2007], that can remove some artifacts yet they still fall short of delivering natural skin deformation or producing realistic musculature or dynamic effects.

Example based techniques. Such techniques beforehand provide a number of input examples, and then synthesize the surface deformation using either direct interpolation between examples [Lewis et al. 2000], or more accurate yet more complex example interpolation [Rhee et al. 2006], or fitting the linear parameters to match the provided examples [Mohr & Gleicher 2003]. Generally, a level of realism only limited by the number of provided examples is offered. However, the generation of examples can be costly, requiring a lot of memory to store them and animator's labour [Lewis et al. 2000].

Capturing real subjects. These methods either exploit 3D scanning device to directly capture the skin deformation [Allen et al. 2003], or complete the shape based on motion capture data [Park & Hodgins 2008, Anguelov et al. 2005] of real people. Furthermore, muscle deformations also can be captured [Neumann et al. 2013]. While these approaches are highly accurate, they require expensive hardware and are subject-specific.

Physically based methods. From the anatomical or physical view, it is logical to animate characters by simulating the underlying musculoskeletal structures. Generally, the skeleton is modeled as an articulated multibody dynamic system [Lee & Terzopoulos 2006, Stavness et al. 2011], muscles, fat and skin tissue can be modeled by either finite element method [Capell et al. 2002, Müller et al. 2002, Georgii et al. 2010] or mass-spring system [Zordan et al. 2004, Zhang et al. 2004, Fratarcangeli 2005].

Assassi et al. [Assassi et al. 2012] developed a lower human body model consisting of finite element models, while Lee et al. [Lee et al. 2009] developed a similar model for the upper body. For visualization

of the skin deformation, the former directly renders the model surface, whereas the latter embeds a high-resolution skin surface as the visualization geometry by means of barycentric interpolation of the surface nodes from the nodes of the tetrahedral simulation mesh.

3.2 MODELING

This section presents the neck mechanical system of which each major component is described hereafter, namely the skeleton, musculoskeletal structure, skin model and the numeric simulation.

3.2.1 *Skeletal Model*

In character animation, generally head movements including flexion, extension, bending and rotation are generated directly by the configurations of joints in the neck spine instead of muscle actuation, and consequently result in deformation of the surrounding muscles. According to the reference anatomical data from the UHM data set, C1-C7 cervical vertebrae, hyoid, thyroid and cricoid are in the neck, and a collection of neck muscles span the bones including sternum, skull, clavicle scapula, costal cartilage and T1-T12 thoracic vertebrae. In our model, we will only model the muscles (see Section 3.2.2) which are the most relevant to skin deformation. Thus, only the bones that are spanned by one or more of these muscles are incorporated into our skeletal model.

We model the skeleton as an articulated, multi-body dynamic system where the bones are rigid bodies. As shown in Figure 3.2, the skeletal model contains skull, hyoid, thyroid, cricoid, C1-C7, the base that is the combination of sternum, clavicle scapula, costal cartilage and T1-T3, and 3-DOF ball joints inserted between adjacent vertebrae, C1 and skull, base and C7 by carefully locating the pivot points as [Lee & Terzopoulos 2006]. Additionally, we adopt the structure from [Stavness et al. 2011] which consists of a revolution joint constraining thyroid and cricoid using the same way as locating ball joints, crossed elastic springs (Eqn. (3.8)) connecting cricoid and base, hyoid and thyroid, point-to-point muscle actuators (Section 3.2.1) coupling skull and hyoid.

Since Lee et al. [Lee et al. 2009] modeled the upper human body also from the UHM data set, hence we are motivated to use the same mass value assigned to each bone, and the details of the parameters are provided in [Lee 2008]. However, unlike their approximation of the inertial properties of the skeleton from the dense volumetric mesh,

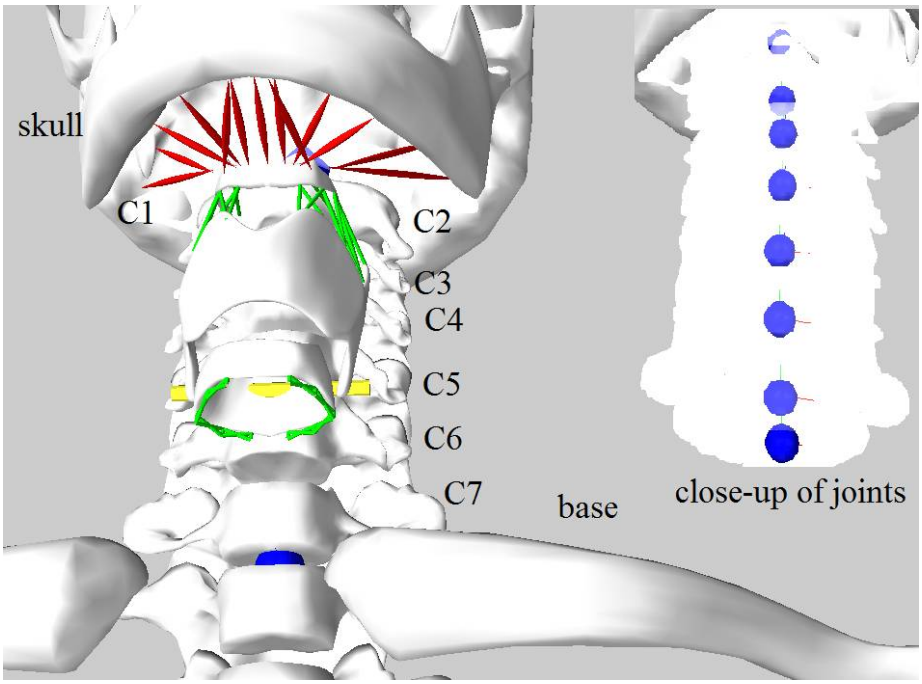

Figure 3.2 The neck skeleton where red lines represent muscle actuators, green lines springs, yellow cylinder the revolution joint and blue dots the pivots of the eight cervical joints.

we approximate the inertial parameters of each bone directly from its density $d = m/v$ where m is the mass and v the volume.

Muscle Actuator

We model the muscle actuator as a linearized Hill-type muscle model [Lee & Terzopoulos 2006]. The total muscle force is the sum of the forces from a contractile element (CE) and a parallel element (PE), whereas the length of the tendon is assumed be constant. The CE force is expressed as

$$f_c = \alpha f_o \mathbf{F}_l(l), \tag{3.1}$$

where $0 \leq \alpha \leq 1$ is the activation level of the muscle, and f_o the maximum force of active muscle. The force-length curve $\mathbf{F}_l(l)$ is in the form

$$\mathbf{F}_l(l) = \max(0, 0.5(1 + \cos(2\pi(\frac{l - l_o}{l_o})))), \tag{3.2}$$

where l is the length and l_o the optimal muscle length at which the maximum isometric force of active muscle is developed. The maximum

and minimum at which muscle can produce force is set $0.5l_o$ and $1.5l_o$, respectively.

The PE force is expressed as

$$f_p = \gamma f_o \min(\frac{l - l_o}{l_{max} - l_o}, 1.0) + d_m \dot{e}, l \geq 0, \tag{3.3}$$

where l_{max} is the maximum stretched length of the muscle, d_m damping coefficient, $\dot{e}$ strain rate, γ weighting factor of the passive tension in $f_m = f_c + f_p$.

3.2.2 *Musculature and Skin Structure*

Muscle modeling

The muscles close to the skin layer are chosen as the relevant underlying soft tissue which are trapezius, sternocleidomastoid, sternohyoid and thyrohyoid. We model them as deformable bodies with volume preservation that introduces visual richness of a more detailed 3D muscular model, resulting in realistic skin deformation. We use finite element analysis due to its convenience in volume conservation. Therefore a simulation mesh that is a discrete representation of the muscle volume is generated beforehand based on the geometric data of the polygon mesh in the UHM data set.

There are several meshing algorithms [Labelle & Shewchuk 2007, Molino et al. 2003, Si 2006] aimed at handling complex geometry that can be applied. Lee et al. [Lee et al. 2009] used the method proposed by Monilo et al. [Molino et al. 2003] for the meshing of polygon meshes from the UHM data set by first generating a Body-Centered-Cubic tetrahedral lattice completely covering the volume of the human body, and then use an algorithm similar to [Sifakis et al. 2007] to cut this lattice along the skin surface to obtain the volumetric mesh of the human body which resolves the muscle models. However, automatic meshing using the geometric data can produce a large number of elements, hence making the simulation impractical. Despite that we can generate coarser simulation meshes by increasing the element size as shown by Lee et al. [Lee et al. 2009], a refinement will be required to the region especially with high-curvature feature due to higher aspect ratios the elements exhibited and so the large number of simulation elements is still there.

To reduce the number of simulation elements, we do not generate the volumetric meshes using the complete geometric data of the UHM data set. We start by a cleaning step in which we subjectively select the

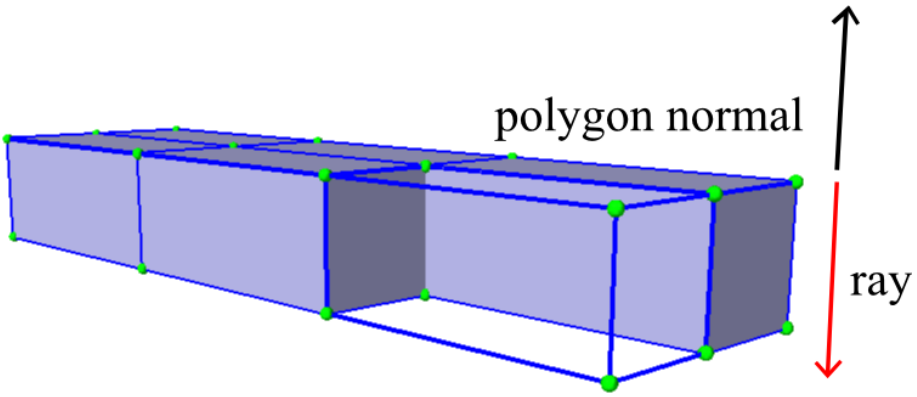

Figure 3.3 Illustration of the meshing method. The polygon faces with nearly parallel normals are extracted, a ray along the inverse normal direction of each vertex is cast, and then the inner node of a hexahedron is defined by picking a point on the 3D line representing the ray according to the parameters (segments and offset).

geometric data of each muscle polygon mesh as the input to the next step, meshing. Next we will explain how the pipeline is realized.

Based on an observation of the geometric complexity of the four selected muscles in the UHM data set, we individually take care of each polygon mesh. For sternocleidomastoid, sternohyoid and thyrohyoid particular faces are extracted for which adjacent vertex normals nearly parallel to each other as regular polygons are distributed on the surface. We cast rays along the inverse normals and take the points as new nodes on the line according to a pre-defined offset (2.5 mm) and number of segments (1) (Figure 3.3), hence generate 91 hexahedrons for sternocleidomastoid, 90 for sternohyoid and 152 for thyrohyoid. However, this method is not applicable to trapezius due to irregular faces and higher complexity in its geometry. To reduce the trapezius model size, we only extract the neck part because, from an anatomical view, by its FE model the skin patch at the back of the neck is mainly influenced. A tetrahedral mesh generator [Si 2006] is used, and using a maximum element volume of 1000 mm^3 and minimum radius-edge ratio of 2.0, the final simulation mesh for the trapezius results in 3,497 tetrahedrons. An overview of the modeled muscles is shown in Figure 3.4.

We use the Finite Element Method [Cook 1994] to solve the governing partial differential equations for continuum behavior. Each simula-

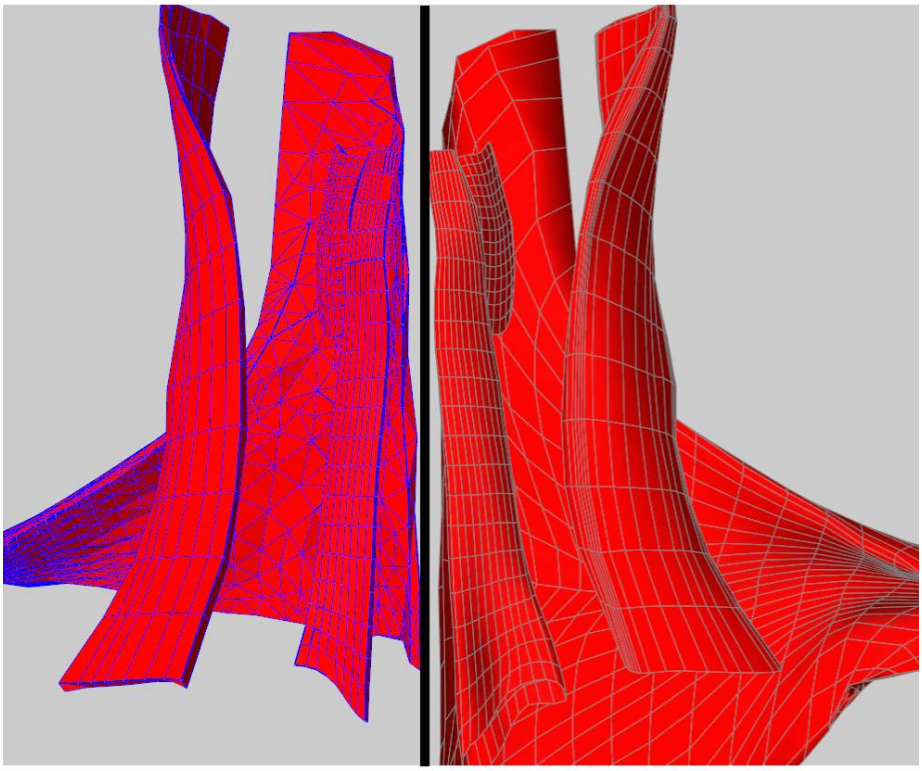

Figure 3.4 Left part is the representations of muscles resulting from meshing, and right part the corresponding polygonal meshes.

tion element is modeled as a co-rotated linear elastic material [Müller & Gross 2004] instead of a hyperelastic Mooney-Rivlin material as [Lee et al. 2009]. This is because linear elasticity constitutive models are computationally cheap, and a method called *Stiffness Warping* proposed by Müller and Gross [Müller & Gross 2004] can be integrated to handle the displeasing distortion in large rotational deformation. It is worth noting that the numerical stabilities of the corotational FEM can be further improved by the approaches proposed by Georgii and Westermann [Georgii & Westermann 2008].

In our FE models, the solid elements are isoparametric elements associated with a defined local coordinate system, the isometric coordinates. Within an element, the position vector in the global Cartesian coordinate system, written as a function of the isoparametric coordinates, is the linear interpolation of the spatial coordinates of the element nodes by the element shape functions. The same parametric interpolation is also used for the displacement field. In our model,

we use the shape functions for hexahedral and tetrahedral elements as defined in FEBio [Maas et al. 2012].

Within each element e, the deformation map Ψ maps every point $\mathbf{X}$ in the undeformed body to point $\mathbf{x} = \mathbf{X} + \mathbf{u}(\mathbf{X})$, where $\mathbf{u}(\cdot)$ is the displacement vector field, in the deformed body. The total strain is obtained by integrating the strain tensor $\Psi_X = \partial\Psi/\partial\mathbf{X}$ at each point over the entire element in the form

$$\varepsilon = \frac{1}{2}(\mathbf{E_C} - \mathbf{I}), \tag{3.4}$$

where $\mathbf{E_C} = \Psi_X^{\mathrm{T}}\Psi_X$ is the deviatoric Cauchy strain tensor. Given the Young's modulus and Poisson's ratio that are used to form a 6×6 matrix $\mathbf{J}$, the Cauchy stress tensor is calculated by the equation

$$\sigma = \mathbf{J} \cdot \varepsilon. \tag{3.5}$$

The elastic forces f_e exerted on the element nodes are derived from the total strain energy and turn out to be linearly dependent on the nodal displacement $\hat{x} = x - x_0$ by using the Cauchy strain:

$$f_e = -\frac{\partial\varepsilon}{\partial x} = \mathbf{K}_e\hat{x} \tag{3.6}$$

where $\mathbf{K}_e$ is the 12×12 stiffness matrix of the element and the calculation is described in [Müller et al. 2002].

The artifacts arising from large rotational deformation that happens in neck movements where moment is generated, are removed using the warped stiffness concept on element scale [Müller & Gross 2004]. The rotated stiffness matrix of an element is calculated by $\mathbf{K}_e^{'} = \mathbf{R}_e\mathbf{K}_e\mathbf{R}_e^{-1}$ where the rotational component $\mathbf{R}_e$ of the element is calculated using the method described in [Müller et al. 2002]. Therefore f_e is modified as

$$f_e^{'} = \mathbf{K}_e^{'}\hat{x} = \mathbf{R}_e\mathbf{K}_e\mathbf{R}_e^{-1}x - \mathbf{R}_e\mathbf{K}_e x_0, \tag{3.7}$$

and the whole stiffness matrix $\mathbf{K}^{'}$ of the entire mesh is the sum of all element's rotated stiffness matrix.

Integration with Skeleton

In our model, the muscles deform when the skeleton moves. In this section, we describe a method to integrate the skeleton with FE models.

A widely used solution consists in constraining any node of the simulation mesh that lies inside or near a bone to a fixed position within the local coordinate frame of that bone. For example the FE simulation

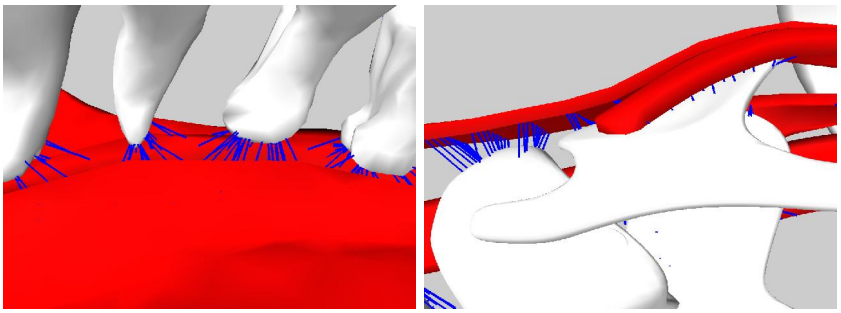

Figure 3.5 Illustration of the soft constraint on FE models, the blue lines represent constraint springs.

software FEBio [Maas et al. 2012] provides a tool where we can subjectively select such nodes by area, surface, or volume selection function. Lee et al. [Lee et al. 2009] proposed a method to automatically find the nodes to be constrained, and its feasibility relies on the fact that they created one simulation mesh resolving all tissues except the bones of the human body and the mesh overlaps with the skeleton tightly. Unlike their modeling, we model part of the superficial muscles individually and only some nodes (mainly in the attachment areas where the muscles connect to the bones) of their simulation meshes lie inside or near bones, hence it is difficult to automatically find these nodes accurately. Therefore, we have chosen to opt for the selection functions provided by FEBio.

Some parts of the simulation mesh are very close to the skin surface in our model, especially the part of the mesh belonging to the sternocleidomastoid. It will lead to odd looking patches of the skin if the attached nodes move rigidly with the bones since we will propagate the simulated nodal motion to the skin layer and consequently deform the skin (see Section 3.2.2). We address the limitations by implementing the soft constraint concept proposed by Lee et al. [Lee et al. 2009].

We have developed an interface to FEBio so the simulation mesh can be imported into its environment. Nodes are selected based on the observation on the anatomy available in the UHM data set. In particular, the nodes which are closest to the bones underneath are selected. Despite that it requires manual labour, in practice, this process is fast and it only needs to be done once for each model. Next we project the node to the surface of the bone by a fast closest point projection method (Section 3.2.2) and the projection is attached to the bone. Finally, we connect the node and its projection using an elastic spring which ap-

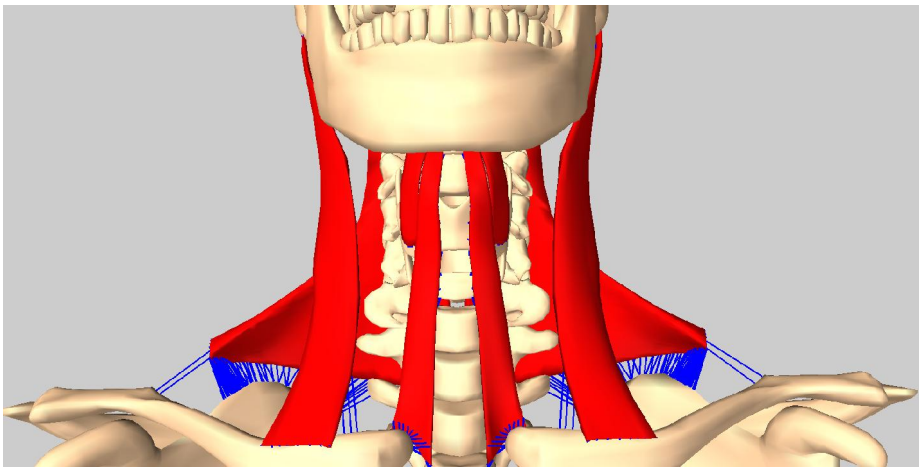

Figure 3.6 An overview of the soft constraints on all FE models. Soft constraints (blue lines) are used to couple the bones and the muscles (red).

plies traction forces on the node as the bone moves. Figure 3.5 depicts how the FE models are coupled with the bones underneath, and Figure 3.6 shows an overview of the soft-hard coupling of the neck model.

Skin Model

Simulating the dynamic skin deformation based on the simulation of the underlying musculoskeletal model has been investigated. Assassi et al. [Assassi et al. 2012] and Lee et al. [Lee et al. 2009] generated the simulation mesh resolving the skin tissue in their lower and upper body, respectively. Nonetheless, the former directly renders the model surface so the visualization largely depends on the element size while the latter barycentrically embeds a high-resolution skin surface as the visualization geometry into the mesh. While it experimentally shows a high degree of realism in skin deformation, we cannot afford such simulation considering our performance objective.

In the torso model by [Zordan et al. 2004], skin simulation is decoupled from the simulation of the underlying model by first recording the trajectories of pre-selected points attached to the model that are the control vertices of a NURBS (Non-uniform rational B-spline) surface. The surface shape is then updated to show skin deformation. The shape of the visualization geometry is implicitly defined by how the control vertices are selected.

We use the mass-spring approach which is popular in real-time simulators like facial animation [Fratarcangeli 2005, Zhang et al. 2004] be-

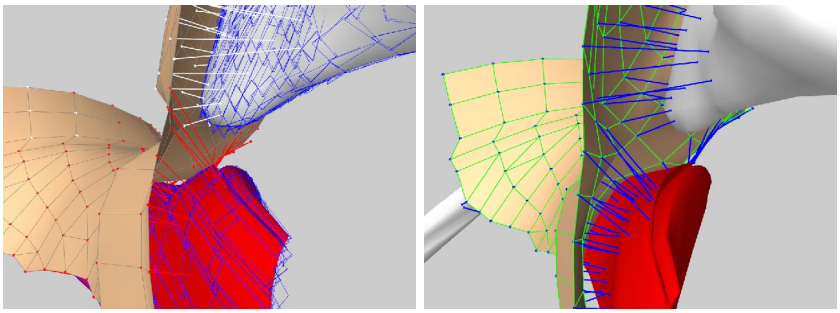

Figure 3.7 Illustration of the skin modeling. Left: White and red lines depict the projection maps and blue rectangles are the volume bounding boxes. Right: Blue lines represent connecting springs and green lines the epidermal springs.

cause it is simple to implement and meets our performance requirements. The human skin is experimentally shown as a multilayered elastic material with non-linear stress-strain relationship based on the study on the rabbit abdominal skin [Fratarcangeli 2005], and the epidermal layer is stiffer than inner layers so its spring stiffness are set to make it moderately resistant to deformation [Zhang et al. 2004].

We use linear spring embedding a damper. The spring force magnitude is a function expressed as

$$f_s = k_s(l - l_0) + d_s\dot{e}, \tag{3.8}$$

where k_s, ds are elastic and damping coefficients, l, l_0 are its length and slack length, $\dot{e} = \dot{l}/l_o$ is the strain rate.

The geometry data of the skin mesh serves as the basis to construct a mass-spring network representing the epidermal layer. We do not model the inner layers yet we use springs as the medium through which the underlying simulated motion is propagated to the upper mass-spring network, and consequently result in the dynamic skin deformation.

From an anatomical view, a patch of the skin surface deforms mainly from the anatomically closest tissue underneath. On account of this fact, the propagation medium should be located between them. To do that, the nodes in the epidermal layer are connected with the points that are attached to the anatomically closest bone or muscle. These points are automatically found based on closest point projection method.

First, to reduce the computation time of finding closest points, we build the oriented bounding box (OBB) trees of the polygon mesh that

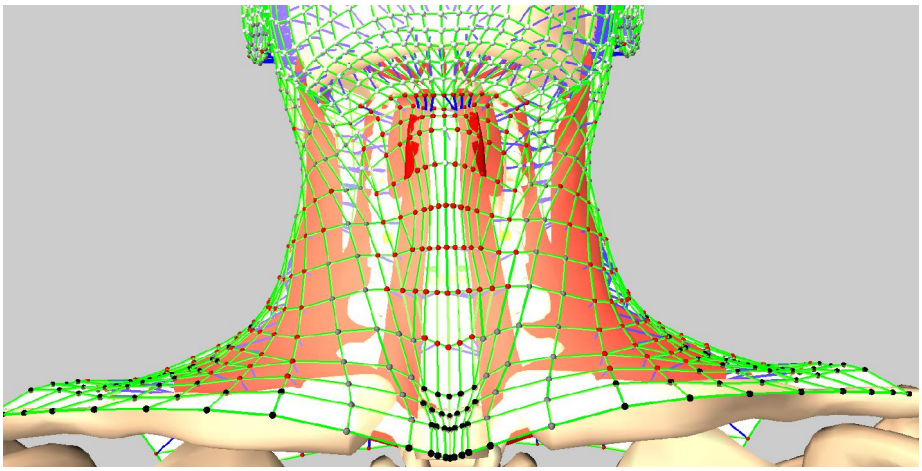

Figure 3.8 An overview of the skin model: green lines denote the elastic springs, red balls denote the point masses connecting to muscles, white balls the ones connecting to bones, gray balls the free point masses, and black balls the fixed ones.

combines all surface meshes of the underlying bodies using an implementation of the algorithm proposed by Gottschalk et al. [Gottschalk et al. 1996]. Each skin vertex is projected onto the closest OBB in world coordinates. Secondly, given a skin vertex we decide on which body surface its projection exactly is. We test the projection against each triangulated body surface. If its barycentric coordinate $(1 - \lambda_1 - \lambda_2, \lambda_1, \lambda_2)$ with respect to a triangle satisfies $0 \leq \lambda_1 \leq 1, 0 \leq \lambda_2 \leq 1$ and $0 \leq \lambda_1 + \lambda_2 \leq 1$ then the projection is in this triangle belonging to the body surface, as follows. If the body is a FE model, the projection is barycentrically embedded by the nodes of the element containing it, otherwise is attached to the coordinate frame of the rigid body. Finally springs connecting the skin vertices and their projection are added into the spring network (see Figure 3.7). Figure 3.8 shows an overview of the skin model as a mass-spring system.

3.2.3 *Numerical Simulation*

Let $\mathbf{q}$ be the positions, and $\mathbf{v}$ the velocities of all the dynamical components of the mechanical system, with $\dot{\mathbf{q}}$ related to $\mathbf{v}$ by $\dot{\mathbf{q}} = \mathbf{Q}\mathbf{v}$. Let $\mathbf{f}(\mathbf{q}, \mathbf{v}, t)$ be the force produced by all the force effector components, let $\mathbf{M}$ be the (block-diagonal) composite mass matrix. We can ensure that $\mathbf{M}$ is constant by representing rigid-body velocity and acceleration

in body coordinates. The following governing equation describes the Lagrangian dynamics of the system according to Newton's second law

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{f}(\mathbf{q}, \mathbf{v}, t), \tag{3.9}$$

and it is constrained due to the presence of bilateral, $\mathbf{G}(\cdot)$, such as joints and point-surface constraints, and unilateral constraints, $\mathbf{N}(\cdot)$, such as joint limits, in the form

$$\mathbf{G}(\mathbf{q}) \geq \mathbf{0}, \quad \mathbf{N}(\mathbf{q})\mathbf{v} \geq \mathbf{0}. \tag{3.10}$$

For FE simulation, a lumped mass model [Stavness et al. 2011] is used to ensure that $\mathbf{M}$ is block-diagonal in order to improve the solution efficiency.

Due to the presence of the FE models, the system is stiff and therefore we need an implicit time integration method for efficient performance. We opt for the simulation framework of Stavness et al. [Stavness et al. 2011], where the Newmark integrator with $\lambda = \frac{1}{2}$ and $\beta = \frac{1}{4}$ is used and hence the step-based updating rule is formulated as a mixed linear complementarity problem which we solve by using the Pardiso solver[Schenk & Gärtner 2004].

3.3 EXPERIMENTS

In the experiments, we use the parameters of the muscle actuators (maximum muscle forces) and crossed springs (stiffness) from [Stavness et al. 2011]. The FE models have a Young's modulus of 50 MPa and a Poisson's ratio of 0.4. We set stiffness to $1.2{\times}10^3 N \cdot m^{-1}$ to epidermal springs, connecting springs to $0.5 \times 10^3 N \cdot m^{-1}$, and constraint springs to $2.0 \times 10^3 N \cdot m^{-1}$. All simulations are run on a laptop with an Intel Core i7 2.4Ghz processor and 6 GB memory.

The head movements mainly include flexion, extension, rotation and lateral flexion:

- **Flexion**: moving the head forward at the joint just below the skull.
- **Extension**: moving the head backward at the joint just below the skull.
- **Rotation**: turning the head to the side (right or left) at a joint below the skull.

- **Lateral flexion**: moving the head toward the shoulder (left or right) at a joint below the skull.

In facial animation, a notable interaction between the head and the neck is the neck skin deforms when the jaw opens. Therefore, to show the level of realism of the neck animation our model offers, we demonstrate the result of the four head movements and the jaw opening.

We extract the frames under extreme postures for each testing movements from the simulation video of the musculoskeletal system and present as shown in Figure 3.9.

A comparison with the popular technique, namely linear blend skinning (LBS), is conducted. We compare the animation at a pose, and refer to a static photo of a human neck at the same pose, to show that the simulation deliver more accurate results than LBS. The comparison is shown in Figure 3.10. Muscles in our model are modeled symmetrically from the left and right, therefore we only show the simulation frames of one-side rotation and lateral flexion.

Our model based on biomechanical modeling also can reproduce a wide range of lifelike animation. It is interesting to animate the model by exciting the muscle actuators and rotate any cervical joints, while realistic animation is generated. We show some frames from the simulation video in Figure 3.11.

Recall that, for performance reasons, we do not model all superficial muscles available in the UHM data set. Thus, the simulated musculoskeletal dynamics are not complete enough to conform with the real neck animation. However, the model still can guarantee the high-fidelity emulation (as shown in Figure 3.10). We bind the vertices to underlying bodies according to the anatomy we incorporated, hence the less anatomy we use, the less accurate the vertex binding is.

Modeling more anatomical structures, or a nonlinear multilayered skin modeling like [Zhang et al. 2004, Fratarcangeli 2005] will definitely increase the memory consumption in order to store the states during the simulation. Additional constraints (attaching proper finite nodes to bones) would be required and therefore would increase the size of influenced matrices in the numerical simulation, hence resulting in higher time complexity. Based on these facts, we trade speed for accuracy, yet as shown in the figures and accompanying video, our model still offers realistic animation.

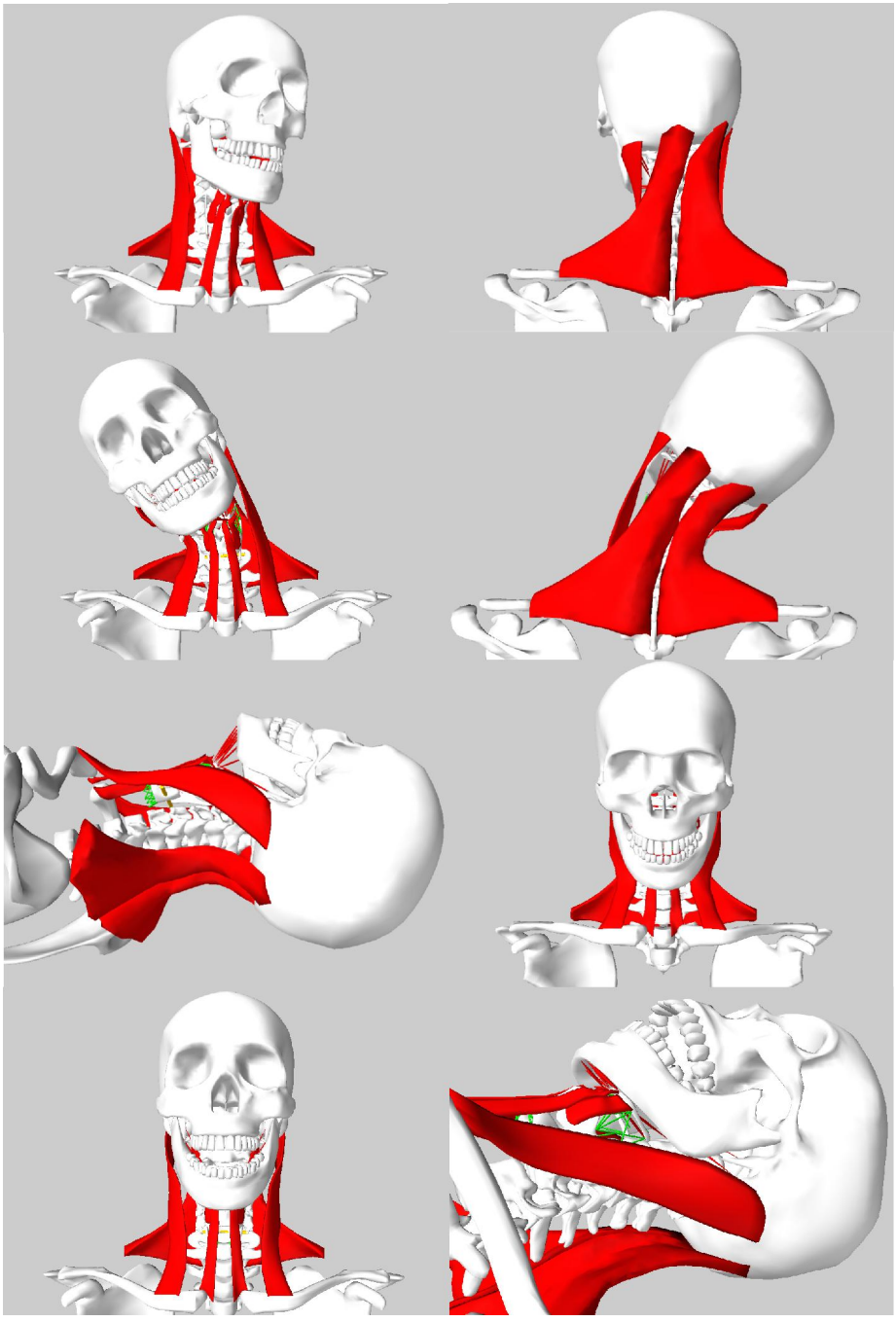

Figure 3.9 Frames from the simulation of the underlying components (muscles, skeleton, actuators and springs).

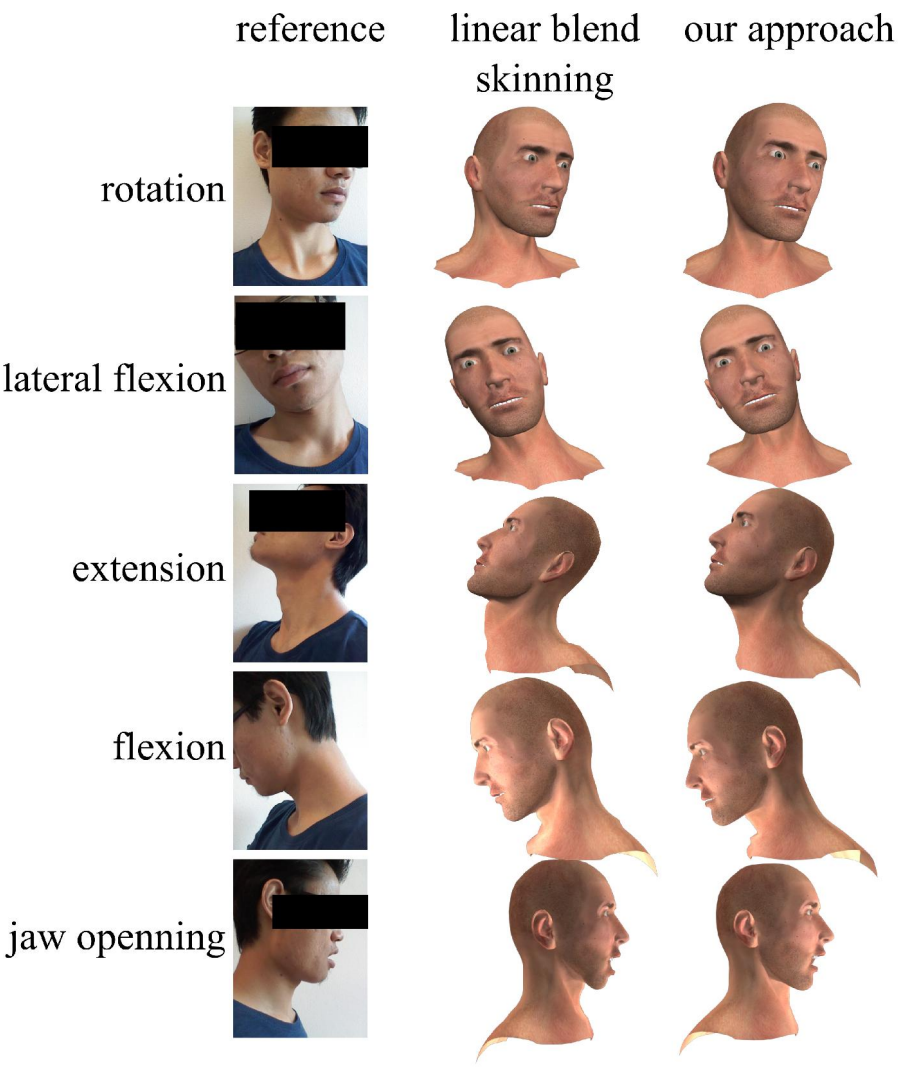

Figure 3.10 Referring to the corresponding static photos, linear blend skinning exhibits artifacts and cannot generate muscular and dynamic effects reflected in the skin, while our model simulates lifelike head poses and neck skin deformation where the visible muscle motion is reflected.

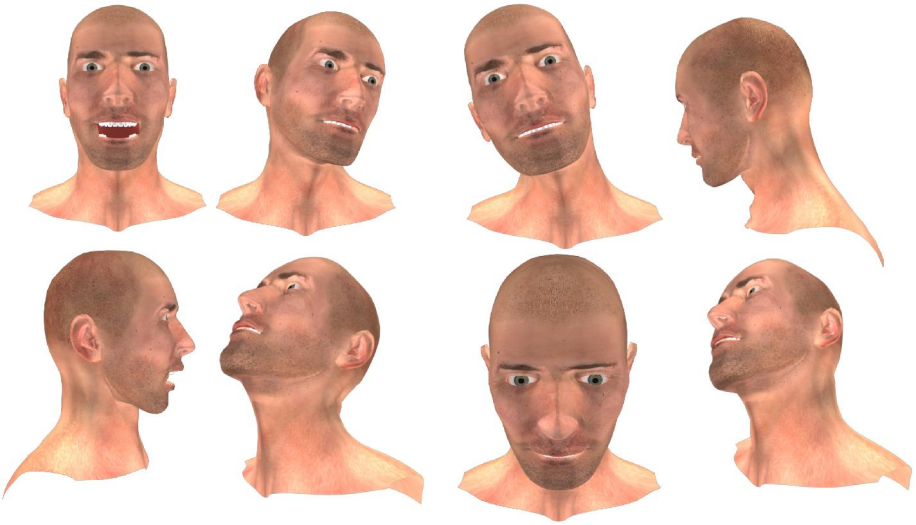

Figure 3.11 Some frames of animation we generated using the simulation results of our physical neck model which is from biomechanical modeling.

3.4 CONCLUSIONS

In this chapter, we have presented a physically-based model of the human neck. The simulations were performed at interactive rates (average 20 FPS) and results showed the realism the model can deliver. Considering the computational cost, we only modeled the most relevant anatomical structures available in a 3D digital model and form a Lagrangian dynamic system which is solved by a semi-implicity time integrator.

Our model experimentally demonstrated that physically-based modeling can produce skin deformation reflecting the muscular and dynamic effects, and showed that the linear elasticity constitutive models are both stable and computationally cheap. Thus, the model is suitable for soft tissue simulation in neck animation. Our method of skin modeling highlights in automatically binding of skin vertices to underlying muscles and bones via linear elastic springs in an anatomical manner. Thus, it is quite easy to implement and fast to simulate.

Elastic deformations are highly desired for skin deformation. Although our model supports such deformations by simulating the material elasticity in an anatomically and physically motivated manner, the model is still computationally expensive for real-time applications.

In next chapter, we will describe a more efficient method producing elastic deformations that support volume and shape-details preservation. The method is based on minimizing a rigidity energy constructed over a voxel grid, which represents the volume domain enclosed by the target surface.

Volumetric Space Deformations with Auxiliary Voxel Grids

4

Physically-based simulation is computationally expensive for real-time applications. This chapter describes a lightweight method that produces nonlinear elastic deformations by preserving the shape details based on volumetric rigidity energy minimization. Though the method does not support interior dynamics as simulations, it still retains the volume-preserving property. Additionally, another closed-form solution for volume preservation is also presented, but it cannot preserve the shape details.

Volume preservation has long been a point of focus in deformation in order to enhance the visual plausibility. A widely applied technique is the dual quaternion skinning (DQS) [Kavan et al. 2008], which is a closed-form solution approximately preserving the volume. Deformation cast as a surface-based energy minimization problem [Huang et al. 2006, Ben-Chen et al. 2009] better preserves the shape details, but for volume preservation, complicated volume constraints need to be defined. Moreover, the resulting deformations might not be as natural as a full volumetric discretization with physically plausible deformation energies [Botsch et al. 2007b]. The implementations of such volumetric energies, however, are often difficult to achieve due to the complex optimization procedures of their objective functions. Physics-based simulations of incompressible solids [Irving et al. 2007, Diziol et al. 2011] improves physical correctness of volume preservation, while they are difficult to control due to the presence of many parameters [Bridson & Batty 2010].

In this chapter, we present two volumetric methods that approximately preserve the volume of an object throughout the deformation. Our methods are easy to implement and run fast on a modern laptop. One method is to apply DQS to voxels that represent the volume domain enclosed by the target surface, and consequently, deform the surface by the efficient trilinear embedding. The embedding is also extended as blending of transformations if the surrounding voxel nodes of a vertex store rotations. For large polygonal meshes, this strategy generally accelerates the computations by at least an order of magnitude. The other method is to cast the voxel deformation as

a rigidity energy minimization problem based on as-rigid-as-possible (ARAP) surface modeling, preserving volumetric properties and producing reasonably natural poses. To maintain the shape smoothness, embedding based on radial basis functions (RBFs) is implemented. To keep user interactions to a minimum in this method, the voxel deformations are controlled by moving point handles in the spirit of linear blend skinning (LBS) instead of directly manipulating voxels. Moreover, a deformed configuration resulting from LBS assists to speed up the iterative minimization and robustly obtain large deformations. In the context of embedding, our two methods are suitable for a wide range of mesh representations.

Both visual and quantitative evaluations on a variety of 3D models are provided in Sect. 5.3; the results show the effectiveness of our methods. The volumetric ARAP yields a relative volume similar to volumetric ProMO energy [Botsch et al. 2007b], which is also a voxel-based rigidity energy, while our method performs at relatively higher speed. We also consult the volume preservation provided by simulations of Finite Element models in particular adapted to animation [Barbič & James 2005], and geometrically motivated deformable models [Rivers & James 2007b]; the comparisons enhances the promise for volumetric deformations held by our methods.

The remainder of this chapter is structured as follows: in Sect. 4.1, we introduce related work on volume and shape details preserving deformation. In Sect. 4.2, we present how we define our volumetric methods based on voxels. In Sect. 4.3, we describe how to compute space deformations that are used to project back the voxel deformations to the original surface, and in Sect. 5.3 we discuss our experimental results.

4.1 RELATED WORK

During the past two decades, a long series of work on volume-preserving shape and character deformation has been proposed. We boldly classify them into five categories.

Geometric volume correction. Aubert and Bechmann [Aubert & Bechmann 1997] introduced displacement constraints on vertices to hold the volume constant. von Funck et al. [von Funck et al. 2008] automatically computed a displacement vector field, along which the vertices were moved to recover the lost volume of smooth skinning. Rohmer et al. [Rohmer et al. 2009] proposed one-dimensional profile curves to assist the volume control of skinning, leading to exact volume

preservation. However, users have to manually specify the influence weights used to control the degrees of volume correction. Pose space deformation [Lewis et al. 2000] requires heavily user-involved displacement corrections, interpolated to adjust the vertex displacements, in order to achieve a desired volume-preserving effect. The above methods are unable to preserve the shape details.

Rotation interpolation approaches. Linear blend skinning suffers from *shape collapse*, as it is essentially a weighted sum of the transformation matrices, where the rotations are, but should not be, linearly parameterized in 3D. Rotation based blending techniques were proposed to produce interpolation paths that follow along geodesic curves in the rigid transformation manifold, leading to more accurate averages of rotations. As a result, the volume is approximately preserved in skinning. Spherical Linear Interpolation [Shoemake 1985] is a manifold-intrinsic averaging of two 3D rotations. However, in the case of more than two rotations, an iterative procedure has to be performed [Buss & Fillmore 2001]. Blending based on quaternion [Kavan & Žára 2005], which is a compact representation of a rotation, approximates the correct manifold-intrinsic averages of multiple rotations, but without effectively handling the translations. Dual Quaternion Skinning [Kavan et al. 2008] blends unit quaternions that represent rigid transformations, successfully eliminating collapse artifacts.

Surface-based energy minimization. Huang et al. [Huang et al. 2006] proposed a gradient domain technique to preserve the volume. The method performs in a subspace rather than the original full space (directly the subject surface), thereby leading to significant acceleration. However, it requires a coarse control mesh that is tedious to establish, and explicit volume constraints. Lipman et al. [Lipman et al. 2007] represented a polygonal mesh by moving frames, preserving both shape details and volumetric properties. The preservation of shape details, however, might come at the expense of volume loss, they therefore scale frames according to local curvature information after obtaining satisfied surface deformations. Ben-Chen et al. [Ben-Chen et al. 2009] preserved shape and volumetric details simultaneously via variational harmonic maps, whose rigidity energy was constructed over the cage in order to speed up the minimization process of the energy function. To avoid tedious manipulations of the cage, positional constraints were modeled. However, to achieve volume preservation, the additional, complex orientation constraints are required. Surface-based energies might not be as natural as a full volumetric discretization with physically plausible deformation energies if without well defined volume

constraints.

Volumetric approaches. Zhou et al. [Zhou et al. 2005] minimized apparent volume changes by the volumetric graph Laplacian, while preserving shape details during deformation. The method, however, works well only for small deformations. Botsch et al. [Botsch et al. 2007a] adapted the shell PriMo energy [Botsch et al. 2006] to voxels, leading to the volumetric PriMO energy. The volumetric version preserves both volumetric properties and shape details based on a powerful embedding into volumetric space. Although the embedding provides high-order shape smoothness, it is too computationally expensive to afford by us.

Deformable solids. Physical accuracy and correctness is a primary goal in simulation. Geometrically motivated simulation of incompressible solids, based on Fast Lattice Shape Matching (FastLSM) [Rivers & James 2007b], provides a robust approximation of volumetric, large-deformation dynamics. Finite Element model with St. Venant-Kirchhoff material [Barbič & James 2005] is much more computationally expensive than FastLSM, but it supports more accurate simulation of the physical dynamics. To achieve a desired simulation, a set of parameters has to be specified by values either from literature, or empirically tuned, making the simulation difficult to control [Bridson & Batty 2010].

4.2 VOLUMETRIC DEFORMATION

The volume domain of a mesh denoted by S is discretized by a hardware accelerated voxelization method [Lefebvre et al. 2013], resulting in a voxel grid denoted by V. In our case, voxels are isotropic, and those across the mesh surface are retained for embedding introduced in Sect. 4.3. As Fig. 4.1 illustrates, voxels are structured in a grid so that they are easily accessible.

4.2.1 *Volumetric DQS*

Given a skinned model consisting of H bones and P mesh vertices, for each vertex $p_j, j \in \{1, ..., N\}$, associated with influence weights $\mathbf{W}_j = (w_1, ..., w_H)$, DQS parameterizes the transformations of handles $h_i, i \in \{1, ..., H\}$ using dual quaternions denoted by $\dot{q}_i$, and transforms a vertex from rest pose to the deformed position by the final dual quaternion as $\dot{q} = (\sum_{i=1}^{H} w_i \dot{q}_i) / \parallel \sum_{i=1}^{H} w_i \dot{q}_i \parallel$.

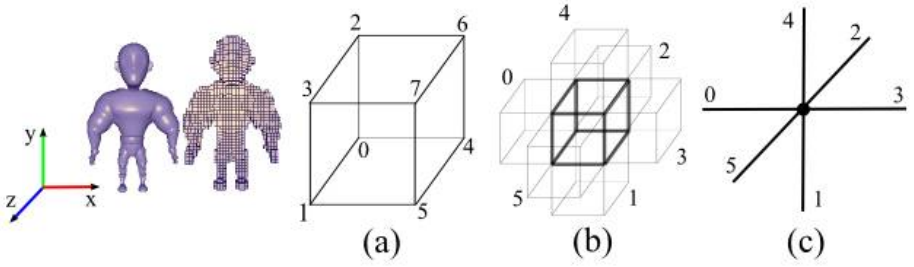

Figure 4.1 The structures of the voxel grid. Each voxel has eight nodes (a) and six immediate neighbors (b), and each node has a 6-connectivity (c). The numbers show the order.

For each vertex, the runtime cost mainly lies in the interpolations of handle rotations, which are nonlinear calculations. To decrease the cost, a straightforward approach is to reduce the number of vertices involving interpolations. In most cases, a voxel grid that is a valid object

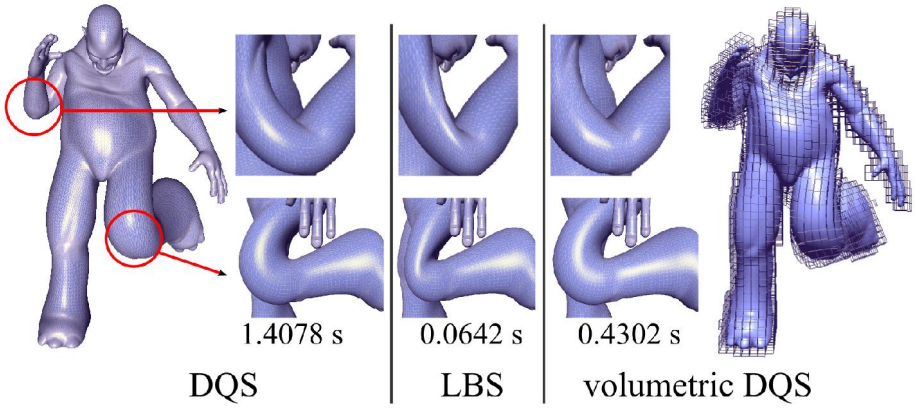

Figure 4.2 An animation still of a skinned character consisting of 84,888 vertices. Volumetric DQS (5,542 voxels) results in volumetric deformation fairly similar to DQS while it is at least an order of magnitude faster. Numbers below are average time in seconds per frame.

representation will be coarser than the embedded polygonal mesh, in particular for high resolution meshes. In the context of space deformation, furthermore only the voxels on the grid hull will be deformed by DQS. As a result, applying DQS to a voxel grid together with trilinear embedding (see Sect. 4.3) largely decreases the cost, see Fig. 4.2.

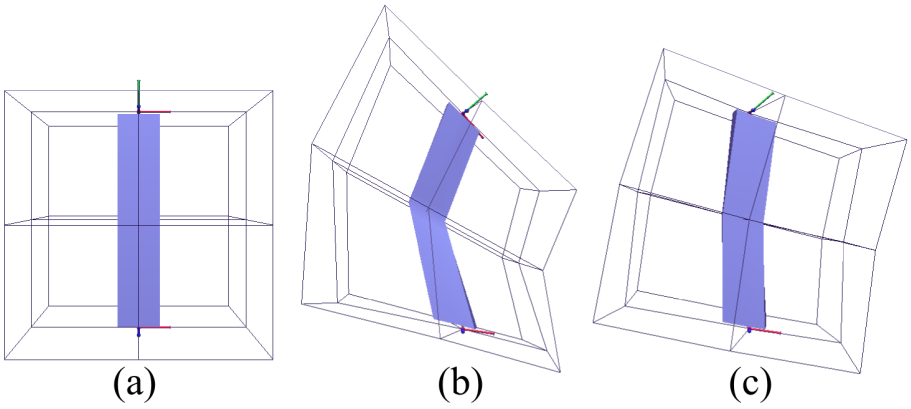

Figure 4.3 A voxel grid with resolution of $2 \times 2 \times 2$ of the box is in rest pose (a). In contrast to volumetric DQS (b), the edge lengths are approximately preserved by volumetric ARAP (c), during the voxel deformation.

The surface-based ARAP energy formulation of [Sorkine & Alexa 2007] is adopted to define our volumetric rigidity energy, as it is conceptually easy to understand and easy to implement. This ARAP formula minimizes the changes of edge lengths, and it addresses the nonlinearity of transformations of each discrete cell.

In our definition, each volumetric cell C_i consists of a voxel and its at most six immediate neighbors. As Fig. 4.3 depicts, the volumetric ARAP approximately preserves the edge lengths between voxel nodes by iteratively minimizing the cell energy as

$$E(C_i^{'}) = \sum_{j \in N(i)} \omega_{ij} \parallel (n_i^{'} - n_j^{'}) - R_i(n_i - n_j) \parallel^2 . \tag{4.1}$$

Edges between a voxel node n_i and its immediate neighbors $n_j, j \in N(i)$ are weighed by ω_{ij} correspondingly. $n_i^{'}$ is the deformed position at an iteration. The presence of rotation R_i of n_i enforces that cell C_i deforms as-rigidly-as-possible. In detail, R_i is derived from the singular value decomposition (SVD) of the covariance matrix $\mathbf{S}_i = \sum_{j \in N(i)} (\omega_{ij} e_{ij} e_{ij}^{'T}) = U_i \sum_i V_i^T$, and $R_i = V_i U_i^T$ [Sorkine & Alexa 2007], where $e_{ij} = n_i - n_j$ and $e_{ij}^{'}$ is the edge after deformation.

The total energy is the sum of all cell energies, and it can be compactly written as a matrix form of equations [Sorkine & Alexa 2007] subject to positional constraints. Weights ω_{ij} are assembled into a

matrix called Laplace-Beltrami operator denoted by $\mathbf{L}$. In surface-based ARAP, cotangent weights [Desbrun et al. 1999] instead of uniform weights are computed to avoid deformation artifacts [Sorkine & Alexa 2007], but at the risk of the emergence of degeneration, as they are only defined for 2D manifolds. In our volumetric ARAP, because the graph Laplacian was easily established beforehand based on the voxel grid, the Laplace-Beltrami operator is easily defined based on uniform weights as

$$(\mathbf{L_V})_{ij} = \begin{cases} \sum_{j \in N(i)} 1.0, & i = j, \\ -1.0, & j \in N(i), \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

To keep user interactions to a minimum, points as *handles* are placed within the volume. The voxel grid is bound to the point handles by influence weights computed using the method of [Jacobson et al. 2011], simply because it yields smooth weights. Instead of directly manipulating the voxels, voxel deformations are controlled by moving points in the sprit of LBS.

Since LBS is efficient for large deformations, the minimization of Eq. (4.1) is accelerated and stabilized for large rotations. First, rotations from rest pose to current configuration resulting from LBS are provided as an initial guess. Second, new positions are computed by minimizing Eq. (4.1). The two procedures are alternately performed to achieve the global unique minimum.

Our volumetric rigidity energy is similar to the volumetric PriMO energy [Botsch et al. 2007a], which also enforces rigidity of the voxels by a nonlinear deformation energy. However, the volumetric PriMO energy is based on directly computing the difference between rigid transformations of two neighboring cells. The transformations are linearized by linear and angular velocities which yield affine approximations, such that the energy minimization will result in a linear system. By solving the system, the optimal affine approximations are found, but which need to be projected to their closed rigid transformations using a nonlinear procedure, which is the eigenanalysis of a 4×4 covariance matrix [Botsch et al. 2007b]. The resulting rigid transformations are used to update the voxels' positions and orientations. In contrast, the volumetric ARAP directly finds the optimal configurations of the voxels also by solving a linear system. The local rotations (not affine approximations) need to be computed using a nonlinear procedure (SVD of a 3×3 covariance matrix) somewhat similar to the one used by volumetric PriMO energy. Therefore, the volumetric ARAP is

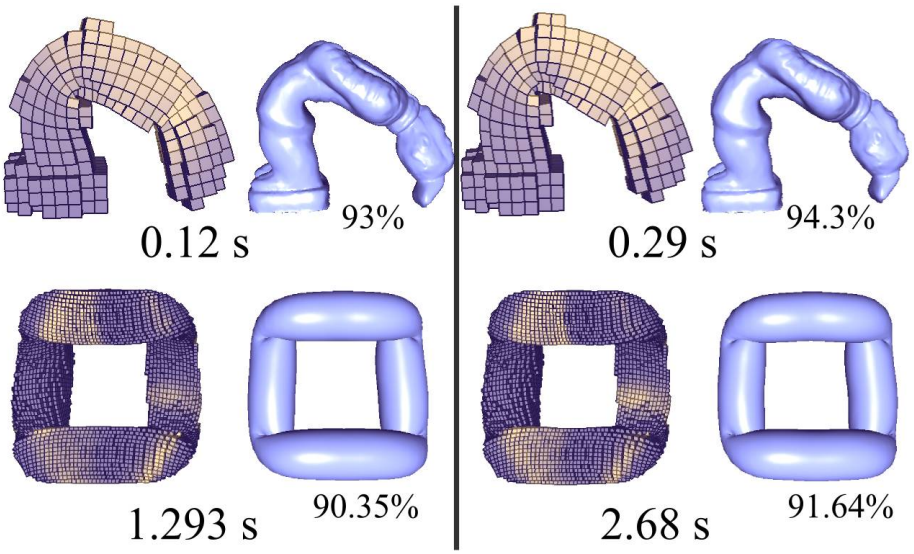

Figure 4.4 The model consists of 40,197 vertices and 1,108 voxels, and the numbers in percentage are relative volume. Volumetric ARAP (left) is faster than volumetric PriMO energy (right), see the average time in seconds per frame, and yields comparable volume preservation.

easier to implement. In practice, we found our volumetric ARAP is also faster than the volumetric PriMO energy, and it results in comparable relative volume, see Fig.4.4.

4.3 SPACE DEFORMATION

The voxel deformations are projected back onto the original surface by space deformation. We construct space deformation $\mathbf{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ based on spatial interpolation. Trilinear interpolation is the fastest way to embed a mesh within voxels, which is the form of weighted sum of the surrounding nodes' displacements. If the surrounding eight nodes of a mesh vertex store rotations that are from the rest pose to the deformed positions, the interpolation is easily extended as blending of transformations, see Fig. 4.5 for their differences. However, these two methods belong to Shepard's scheme [Barnhill et al. 1983] that is not smooth enough to interpolate surfaces.

To address the limitations, smooth interpolation of displacements based on radial basis functions (RBFs) is developed. In particular, we

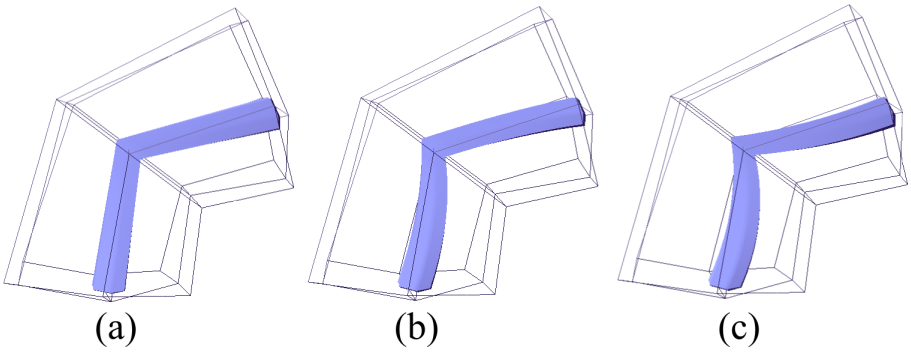

Figure 4.5 Blending transformation matrices (b) or their dual quaternion representations (c) has less translation-insensitivity observed, with respect to trilinear embedding (a).

Figure 4.6 Surfaces are rendered under the same point light. Trilinear interpolation and blending of transformations give rise to a pattern of strips on the surface. RBFs interpolation is smooth enough to interpolate the surface.

use RBF with compact support to yield a sparse matrix

$$\ker g(x,c) = e^{\left(\frac{-\|x-c\|^2}{\sigma^2}\right)} + a^T \cdot x + b, \tag{4.3}$$

where $c \in \mathbb{R}^3$ is the RBF center which is the center of a hull voxel in our case, $x \in \mathbb{R}^3$ is the evaluation point, a^T and b are the coefficients determining the polynomial term. The presence of the linear term helps to reproduce the global behavior of the function. σ is the average distance between RBF centers following the guideline of ALGLIB [ALG], which is a popular numerical analysis library, guaranteeing that there will be several centers at distance σ around each evaluation point. The resulting system of equations can be solved by LU decomposition. This RBF interpolation gives rise to smoothness that satisfies our expectation, see Fig. 4.6.

Model	# Voxels	# Vertices	DQS		Volumetric DQS		Volumetric ARAP	
			Volume (%)	Time (s)	Volume (%)	Time (s)	Volume (%)	Time (s)
Long cube	4,800	1,002	99.419	0.0068	99.4188	0.038	95.3269	0.423796
Fat man	5,542	84,888	97.315	1.232	97.308	0.364397	98.3852	2.74804
Warrior	1,108	40,197	86.327	0.2692	86.3242	0.01398	93.0	0.0631
Head	18,296	48,624	99.434	6.69669	99.422	0.453636	94.1977	23.9236
Arm	13,263	49,874	98.5681	8.21843	98.5649	0.227855	99.3745	7.32747
Name card	10,849	130,340	110.461	0.89	110.425	0.418142	99.8083	6.39093
Mushroom	16,408	131,304	99.981	0.938	99.9798	0.2152	98.5793	26.13
Tree	3,261	11,645	107.258	0.0828	107.26	0.0457	105.032	1.99311

Table 4.1 Model statistics: Timings are the overall time of reaching the target pose, and percentages are relative volume also indicating whether an increasing or decreasing volume. Volumetric ARAP (2 iterations) has relative volume similar to volumetric DQS, while it produces more natural poses. Note that the number of voxels is much smaller than the number of vertices in cases of large meshes, where volumetric DQS is much faster than DQS.

Model	Trilinear	Linear blending	DQ blending	RBFs
Long cube	0.0008	0.0016	0.0203	0.1154
Fat man	0.0581489	0.0882	4.1103	9.20568
Warrior	0.0305	0.0526	0.56019	1.30276
Head	0.0291048	0.0547445	1.44623	9.52889
Arm	0.030507	0.056674	1.76649	8.46751
Name card	0.0686187	0.132592	9.52885	23.0921
Mushroom	0.0739739	0.135898	71.059	82.674
Tree	0.004	0.0094	1.32838	1.14412

Table 4.2 Space deformation timings: Trilinear and linear blending are sufficient for real-time requirements. Time is in seconds.

4.4 RESULTS AND DISCUSSION

Our methods were implemented in C++ on a laptop with an Intel Core i7 2.4Ghz processor and 6 GB memory. The LU decomposition, sparse Cholesky solver and two-sided Jacobi SVD are provided by the Eigen library [Guennebaud 2011].

Model statistics and quantitative comparisons are reported in Table 4.1. Volumetric DQS yields volume preservation fairly similar to DQS, while it is often much faster. Volumetric ARAP is more computationally expensive than volumetric DQS. Statistics on space deformation are presented in Table 4.2. Trilinear embedding and linear blending are efficient enough for real-time applications, while RBFs

interpolation is of course more computationally expensive. As linear blending, DQ blending supports no smoothness but it is much slower, thus, linear blending should be the space deformation method, used to work with rotations of voxel nodes.

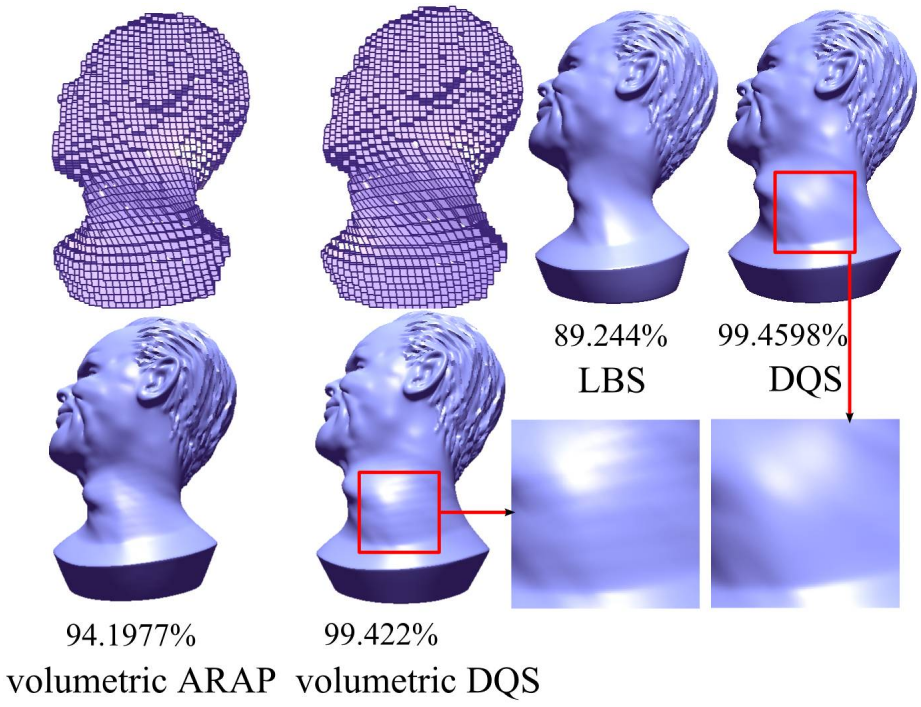

Figure 4.7 Numbers below are relative volume. With trilinear embedding, the volumetric methods approximately preserve the volume w.r.t. LBS, and lead to a pattern of stripes on the surface.

Both volumetric DQS and volumetric ARAP approximately preserve the volume as opposed to LBS, see Fig. 4.7, and even in complex motions such as bending and twisting, see Figs. 4.8, 4.9, respectively. Fig. 4.7 also shows that, with trilinear embedding, volumetric DQS has much higher performance than DQS in terms of computational time (see timings in Table 4.1), though it sacrifices a certain degree of smoothness. Volumetric ARAP produces more natural poses than Volumetric DQS. This is because it essentially preserves the locality, see Fig. 4.10. As Fig. 4.11 shows, volumetric ARAP also fits user constraints better than volumetric DQS. This explains the cases where volumetric DQS preserved the volume better than volumetric ARAP, as indeed it did not fit point transformation well.

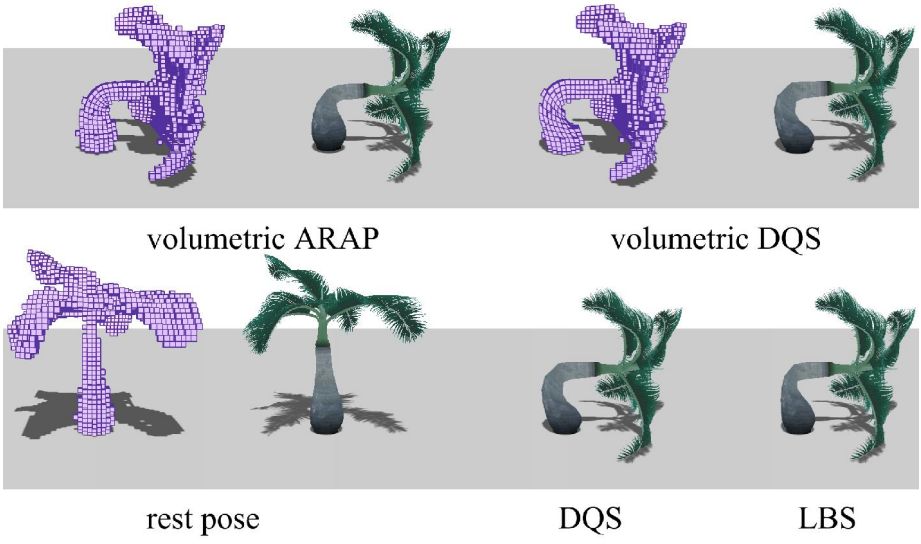

Figure 4.8 Volumetric ARAP and volumetric DQS are able to preserve the volume to a degree in large bending with respect to LBS. Volumetric ARAP, however, produces more natural poses than DQS and volumetric DQS. Note that a surface-based deformation energy might be unsuitable for this tree model, due to the presence of degenerated triangles in the leaf parts.

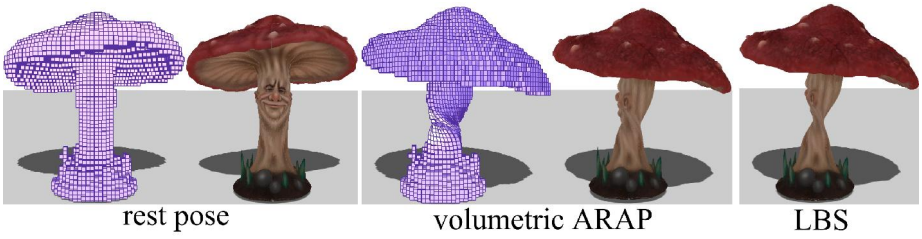

Figure 4.9 The mushroom model twists by 135 degrees. Volumetric ARAP preserves 98.5793% of the volume, while LBS gives rise to the well-known artifact of shape collapse.

Our methods are employed in character rigging, and the examples are shown in Figure 4.12 and 4.13. The volumetric ARAP method requires only translations, as it computes local rotations behind. This fact largely reduces the man effort in rigging, as translations are intuitive while rotations are non-trivial and much less intuitive to specify.

Figure 4.10 Volumetric ARAP prevents shape distortion. The letters in the name card model deform as-rigidly-as-possible while they distort by using volumetric DQS, see the close-ups.

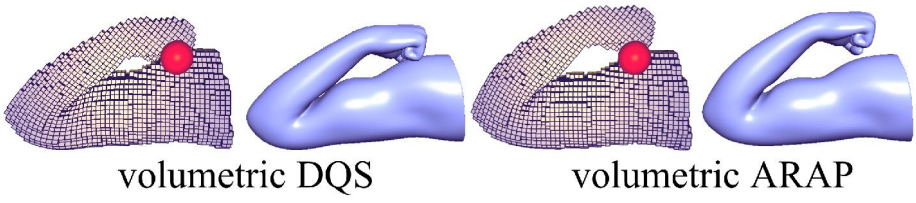

Figure 4.11 The arm bends to fit the point constraint (red ball) by volumetric ARAP, whereas it bends excessively by volumetric DQS.

4.4.1 *Simulation*

The simulation of deformable model aims to provide physical correctness of volume preservation. Our methods are compared with simulation models in terms of computational time and relative volume. In particular, the geometrically motivated deformable body, called *Fast Lattice Shape Matching* (FastLSM) [Rivers & James 2007b], and the Finite Element Model with St.Venant-Kirchhoff material (FEM-StVK) [Barbič & James 2005], are consulted.

FastLSM. At each time step, FastLSM finds the least-squares transformations of shape matching regions, which are comprised of a set of simulation particles. The region size is controlled by a high-level parameter called *region half-width* denoted by κ. The larger the value is, the stiffer the model will be. The distances between current positions,

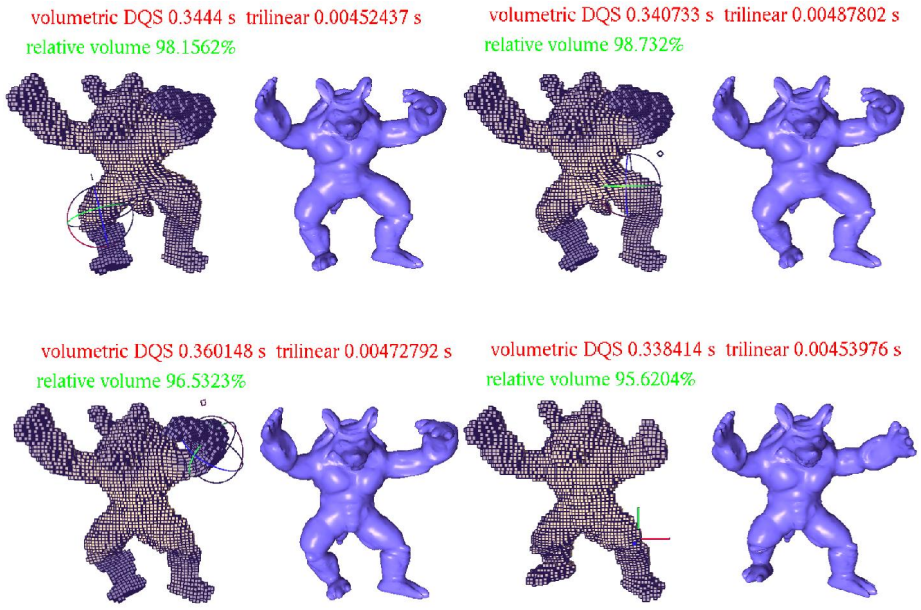

Figure 4.12 Character rigging with volumetric DQS. Both translations (frame) and rotations (trackball) have to be specified.

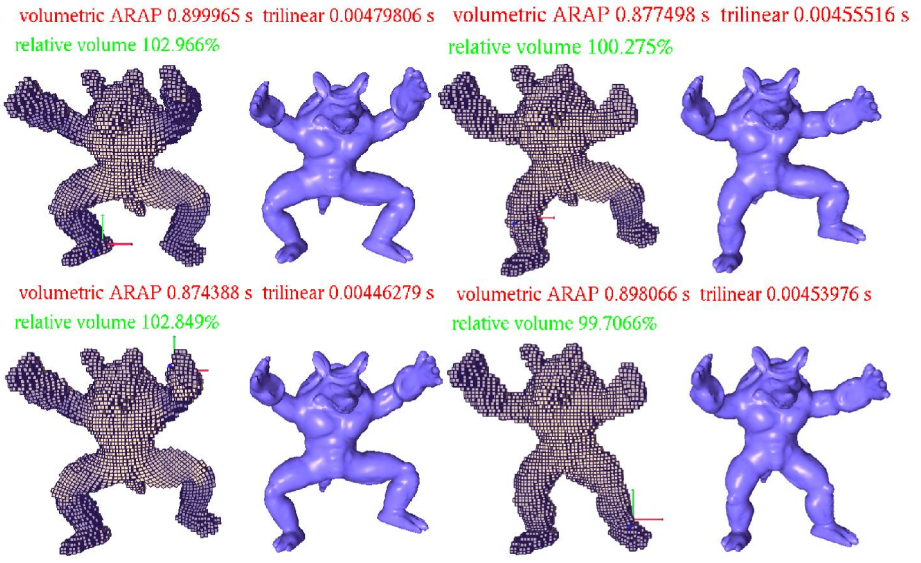

Figure 4.13 Character rigging with volumetric ARAP. Only translations (frame) needs to be specified.

and goal positions computed by transforming the rest configurations

using optimal transformations, are clamped to replace the force of the Euler integration scheme [Müller et al. 2005], thereby achieving stable numerical integration.

FEM-StVK. StVK is mathematically defined to adapt FEM to animation, as it has a linear stress-strain relationship. Fast implicit Newmark subspace integrators [Barbič & James 2005], with a local Rayleigh damping model [Barbič & James 2005], are used to timestep the the ordinary differential equations. This FE model can rapidly simulate the nonlinear dynamics.

Model		Fast LSM		FEM	
	κ	Volume (%)	Time (s)	Volume (%)	Time (s)
Long cube	2	96.8872	10.832	98.1246	35.6615
Fat man	3	98.1003	12.4213	99.1088	47.24
Warrior	2	96.33	3.4531	97.43	11.3481
Head	3	97.5433	51.78		
Arm	3	99.102	37.81		
Name card	3	99.103	25.672		
Mushroom	4	99.012	42.3		
Tree	2	103.549	8.9511	102.03	34.371

Table 4.3 Model statistics of simulation: Percentages are relative volume also indicating whether an increasing or decreasing volume. Time in seconds is the overall time of reaching the target pose. Diagonal means where the time is larger than one minute.

We use the parameters reported in the published literature: (i) FastLSM uses a region damping constant of 0.25, and timestep of 0.033 [Rivers & James 2007b]; (ii) FEM-StVK uses mass density of 1000 kg/m^3, Young's modulus of 10^8 N/m^2, Poisson's ratio of 0.45, for all elements, $\hat{\beta} = 0.25$, $\hat{\gamma} = 0.5$ as parameters of integrators, Rayleigh damping constants of $\alpha = 0$, $\beta = 0.01$, and timestep of 0.033 [Barbič & James 2005]. As we can see, there are a number of parameters need to be defined before a simulation, making the user control difficult.

Model statistics are presented in Table 4.3. As we can see, simulations require more time to reach the target poses produced by volumetric ARAP and volumetric DQS. The volume preservation, supported

by our volumetric methods, are comparable to simulations (recall Table. 4.1). In the case of a large-size model, FEM-StVK is unsuitable for interactive applications, as it requires more than one minute.

4.4.2 *Discussion*

In conclusion, our methods target different applications. For real-time applications such as computer games where volume preservation is desired, volumetric DQS with trilinear embedding, or its extension as linear blending of transformations, gives the solution. In feature films production or virtual human simulation, where smoothness, natural posture and volume preservation are desired for higher degrees of realism, volumetric ARAP combined with RBFs can be a solution.

4.5 CONCLUSIONS

In this chapter, we have presented two space deformation methods based on voxelization. The methods, named volumetric DQS and volumetric ARAP, both preserve the object's volume to a degree while the latter method produces more natural poses. In general, speedups are obtained for volumetric DQS with respect to DQS. A variety of results of large polygonal meshes have been demonstrated, showing the effectiveness of the methods. Discussions on promising applications have been detailed.

The volumetric ARAP method holds promise for skeletal deformation. Currently, it cannot always achieve realistic character postures and skin deformations, as additional constraints should be introduced, notably constraints that keep bone lengths constant. Moreover, physically the character should stand throughout the animation, so the center of mass of the body has to be optimized at run time. Since the volumetric ARAP energy aims to conserve edge lengths, hence it may result in a locking problem that can be solved by the method of [Irving et al. 2007].

Our method of elastic deformation improves the surface deformation in terms of volume and shape-details preservation. However, the method does not support the local controllability of deformations which is necessary to character deformation. In next chapter, we will present a novel space deformation method based on domain decomposition, which will address the problem. Thus, combined with this space deformation method, the physically-based simulation models or

the approaches based on volumetric rigidity energy minimization can be employed to animate the character skin.

Part III

Character Deformation

Multi-Domain Smooth Embedding for Character Deformation

5

In chapter 3 and 4, we have described methods that produce elastic deformations in order to improve the deformation quality. However, as explained previously, our methods still have certain drawbacks when employed in character deformation. First, the methods do not guarantee that each branch of a character model deforms without interplay between other closed branches. Second, the methods require much higher computational cost with respect to direct skinning. Thus, we need to solve the computational cost when employing these methods in order to improve the realism of character deformation. In this chapter, we present a space deformation method based on domain decomposition and scattered data interpolation, which addresses the aforementioned issues.

To provide more realistic skin deformation, many advanced deformation techniques have been proposed. For example, methods of casting deformation as a nonlinear energy minimization problem [Huang et al. 2006, Botsch et al. 2007a, Ben-Chen et al. 2009], support preservation of shape details and volumetric properties. Simulation models of deformable solids [Barbič & James 2005, Rivers & James 2007a] provide interior dynamics in addition to quasistatic skin.

However, such techniques are either too computationally expensive, or have high degrees of modeling complexity. A common solution to reduce the computational cost is to use space deformation. In order to reduce computational complexity, a much coarser mesh is established to loosely enclose the original mesh, carrying out the expensive nonlinear optimization, and interesting deformations are propagated back to the target surface by mean value interpolation [Huang et al. 2006] or harmonic coordinates [Ben-Chen et al. 2009] for instance. In space deformation based on interpolation, the deformed position of a vertex is an interpolation value of surrounding data points.

The widely adopted Shepard's interpolation scheme [Barnhill et al. 1983] as space deformation, such as trilinear interpolation, is not smooth enough to interpolate the target surface. In many applications including feature film production and surface visualization of virtual human simulation, smoothness is highly desired. In these scenarios, adopt-

ing Shepard's scheme would violate the intention of targeting high-quality deformations by applying aforementioned, advanced deformation techniques.

Scattered data interpolation based on radial basis functions (RBFs) is extensively used to smoothly interpolate surfaces. Generally, the basis functions use Euclidean distances without addressing the locality property. So when used in character deformation, embedding as a single RBF interpolation system would lead to interplay between segments that are close to each other. The use of geodesic distances [Levi & Levin 2014] reduces the number of such artifacts. However, this distance metric heavily depends on the data type and the object shape, leading to a loss of generality.

Another typical purpose of using space deformation is to decouple the surface, used for visualization of the deformations, from the representation carrying out simulation or volumetric deformation. In such a context, the volumetric representation required for modeling a deformable solid is not bound to the surface [Barbič & James 2005, Botsch et al. 2007a], or the fast shape matching model [Rivers & James 2007a] is fairly easy to achieve with the help of a lattice. Thus, relevant modeling is simplified.

Our work. We propose space deformation as multi-domain embedding for character deformation. The embedding is realized by scattered data interpolation based on RBFs with compact support, supporting shape smoothness and leading to substantial speedups. The domains, each of which corresponding to a bone-associated segment and attached to a small, linear system of RBFs simply with Euclidean distances, are obtained based on skinning weights that actually encode a skin segmentation. So in this context, each domain-specified RBF independently interpolates the surface of interest, ensuring that the deformation of a segment is locally controlled. To cope with seam artifacts, sharp features at and around boundaries of contact regions are smoothed out by a simple, geometric smoothing method.

Our multi-domain RBF interpolation is easy to implement. It performs at interactive rates, and maintains the smoothness of the embedded surface during deformation. The method is successfully employed in various scenarios including physics-based simulation of deformable solids, geometrically-motivated soft body animation and a nonlinear, volumetric deformation energy.

Embedding via interpolation. Space deformation dates back to the Free Form Deformation [Sederberg & Parry 1986] (FFD) that uses a regular lattice as control structure. The regularity complicates the control of concave regions of interest, and FFD would generate spatially-variant deformations. Cage-based deformation [Ju et al. 2005] requires a well-established control mesh that loosely encloses the target surface, ensuring that the computed barycentric weights are positive. However, the popular mean value coordinates [Ju et al. 2005] may still contain negative values, leading to inevitable deformation artifacts.

In the presence of a regular voxel grid, the target surface embedded inside the grid is deformed often by trilinear interpolation. For each vertex, its deformed position is a weighted interpolation of the displacements of the eight nodes of the containing voxel. This linear space deformation is very efficient and GPU-friendly as the data size is fixed and small. If the surrounding nodes store full transformations composed of translation and rotation from rest position to current position, an extension such as linear blending of transformation matrices or dual quaternions [Kavan et al. 2008] can be easily achieved, resulting in a more robust interpolation approach [Müller & Chentanez 2011]. Similar to skinning, in this case, the target surface is bound to the control mesh by trilinear weights. The above methods essentially belong to Shepard's interpolation scheme [Barnhill et al. 1983], which would lead to a loss of shape smoothness during deformation, visually demonstrated by Lewis et al. [Lewis et al. 2000].

Moving Least Squares (MLS) is a popular scattered data interpolation method, and is often used in surface reconstruction [Pauly et al. 2003] from an unstructured point cloud. MLS builds an approximation, fitting the data in a weighted least-squares fashion, by a local polynomial function. The weight functions of MLS, however, should not fall quickly to zero, otherwise the method will result in many small linear systems each which only involves a few data points, leading to undesired interpolation results. If the weight functions are an inverse power of the distance, MLS then reduces to Shepard's scheme.

Radial basis functions (RBFs) are the most versatile and commonly used scattered data interpolation techniques. RBFs are successfully applied in graphics and animation: shape interpolation [Lewis et al. 2000], surface reconstruction from point cloud [Carr et al. 2001], shape editing [Botsch & Kobbelt 2005], facial animation [Noh et al. 2000], and marker-based skin deformation capturing [Park & Hodgins 2006]. RBF

interpolation results in smooth deformation fields and can be evaluated at arbitrary sample locations of the embedded mesh. Mostly, the basis functions are based on Euclidean distances. In this case, movements of a branch of a model therefore may affect other branches which are close to that branch in Euclidean space, which occurs quite often in character deformation. Levi and Levin [Levi & Levin 2014] used geodesic distances to overcome this limitation. Nevertheless, this distance metrics heavily depend on the mesh topology or representation, thereby leading to a loss of generality.

Domain decomposition. An articulated deformable character consists of limb segments encoded by skinning weights, and the deformation of each segment is locally controlled. Based on this fact, several methods were proposed to decompose the character into multiple domains each which corresponds to a segment.

Kim and James [Kim & James 2011] proposed to partition the character into multiple bone-associated domains, and then coupled them using a multibody formulation in order to remove the rigidity artifacts in physics-based simulations. Park and Hodgins [Park & Hodgins 2006] leveraged domain-decomposition to capture skin deformation. First, they computed near-rigid segments of a marker-based, reconstructed surface. Second, they approximated the local deformations of each segment with a continuous deformation field based on radial basis interpolation. Vaillant et al. [Vaillant et al. 2013] obtained segments based on skinning weights. Each segment was represented as an implicit surface that is an iso-surface of a smooth scalar field, also using radial basis functions. However, in addition to vertex displacements, vertex normals were introduced into the functions in order to obtain robustness. Vertices belong to a segment are then projected back onto the corresponding iso-surface according to iso-values. Their method preserves shape details, and can be used in real-time animation but by resorting to hardware acceleration. Later, the method was improved by using the skin mesh to directly track iso-surfaces of the field over time [Vaillant et al. 2014], avoiding tedious specification of skinning weights. Nevertheless, the field functions still have to be composed into a global field function, which is difficult to realize in practice.

Our method interpolates the target surface rather than reconstructing followed by vertex projection, and also needs no blending of field functions [Vaillant et al. 2013] to handle seam artifacts. Instead, a simple, geometric smoothing as post-processing is introduced to cope with the sharp features at and around boundaries of contact regions.

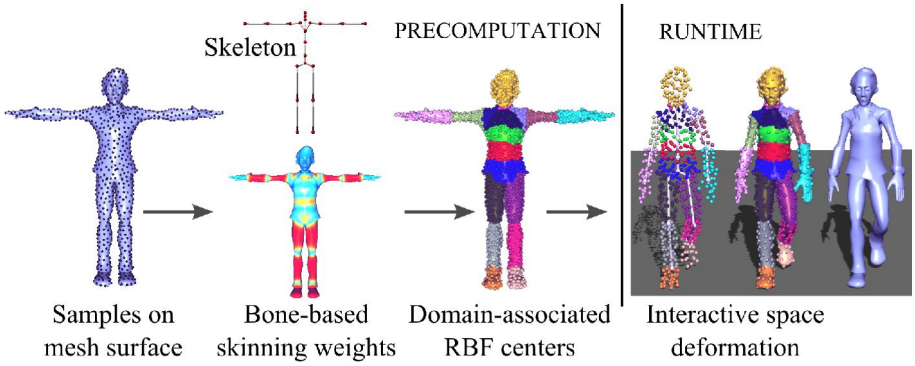

Figure 5.1 Pipeline overview: We first sample a set of points (black dots), ready to be RBF centers, on surface, and then partition the character into multiple domains based on skinning weights, indicated by colors. Each segment, indicated by a colour, is accordingly associated with a small group of samples to construct a RBF system. Our multi-domain, smooth space deformation is applied in interactive animation applications.

5.2 MULTI-DOMAIN SMOOTH SPACE DEFORMATION

Figure 5.1 shows an overview of our method. Given a skinned character, it is decomposed into multiple bone-associated domains, and each domain is attached to a smooth interpolation system based on radial basis functions.

5.2.1 *Smooth interpolation*

Given the value of a scalar-valued function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ in N distinct discrete points $c_i \in \mathbb{R}^3$, called RBF centers, the RBFs learn a smooth interpolation function of F in $\mathbb{R}^3$. Each point $x \in \mathbb{R}^3$ is evaluated as a weighted sum of evaluations of N radial basis functions.

Globally supported basis function provides high-quality smooth results and evaluate at arbitrary sample locations. However, it results in a dense linear system which might be highly ill-conditioned in the case of a large number of samples, making it difficult to numerically deal with that system of equations.

To obtain substantial speedups, RBF with compact support, yielding a sparse matrix, is used to interpolate axis-aligned displacements. Specifically, we use the same Gaussian kernel described in chapter 4

$$\ker g(x) = e^{\left(\frac{-\|x-c\|^2}{\sigma^2}\right)} + a^T \cdot x + b. \tag{5.1}$$

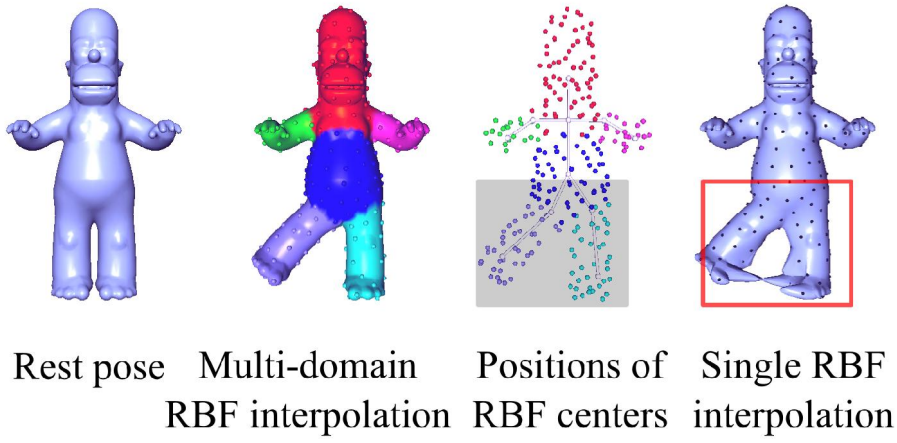

Figure 5.2 Our method avoids undesired mutual effects between segments in character deformation. A global linear interpolation, however, results in serious artifacts, where the localized changes in the right leg affect the left leg, even when these two point clouds are apart.

5.2.2 *Multi-domain embedding*

The use of a single, large linear system of RBFs has limitations. On the one hand, available distance metrics, such as Euclidean and geodesic distances, cannot guarantee that a movement of a branch of the model does not affect other branches which are close to that branch. In a humanoid character for instance, the left leg must animate without affecting the right leg. On the other hand, the space of deformations for humans is likely too complex to be approximated by a standard linear interpolation technique [Park & Hodgins 2006], and so by a single radial basis interpolation.

To address these limitations, we decompose the character into multiple domains, and attach each to a RBF interpolation. The domains are obtained based on skinning weights that often encode a limb segmentation of the character. A domain is a group of vertices each which the largest influence weight is provided by the same bone. If there are multiple bones having the largest weight for a vertex, the vertex will be associated to only one of them. Since encouragingly, skinning weighs are often painted or adjusted by skillful artists to guarantee the correctness, hence actually a domain is a bone-centered segment which skin

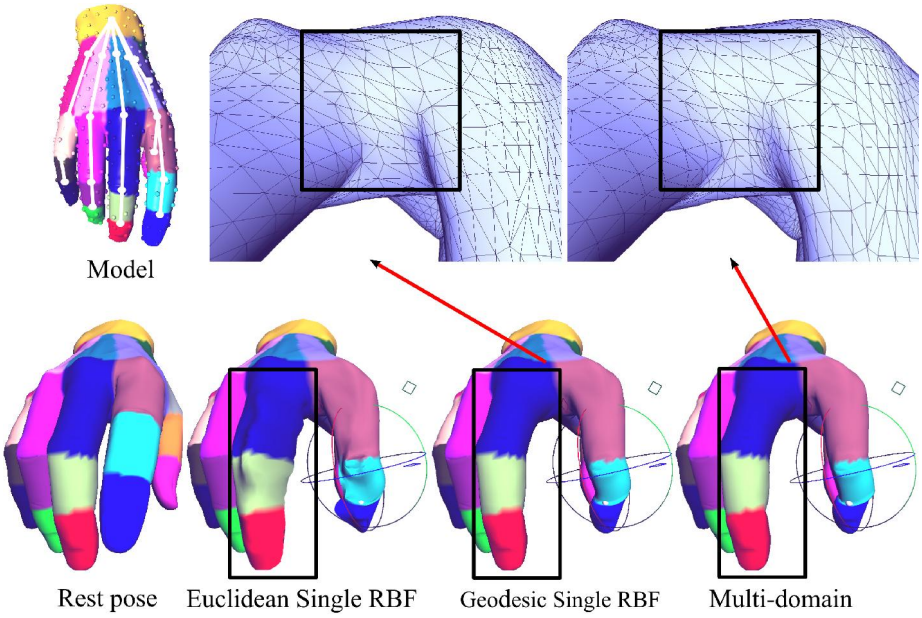

Figure 5.3 Skeletal hand model. Both single RBF with geodesic distances and our method avoid interplay between fingers, whereas single RBF with Euclidean distances does not. Our method, however, supports higher-order smoothness than geodesic-based single RBF. The trackball indicates rotating motion.

deformation will be locally controlled.

To construct the domain-associated interpolation systems, we need the definition of RBF centers. In our method, RBF centers are a set of points sampled on the mesh surface in particular using Poisson disk sampling [White et al. 2007] that yields a uniform sampling. Based on the resulting segmentation mentioned above, the samples are put into corresponding domains. Then, samples within a domain are used to build the RBF system of that domain. So in our method, the number of RBF centers is proportional to the number of samples but directly proportional to the much larger number of vertices, largely reducing the runtime cost. In space deformation, e.g., of Finite Element simulation with hexahedra, only the RBF centers are embedded by trilinear weights, and the target surface is left to a set of domain-associated interpolation systems.

The benefit of using domain decomposition is two-fold: (i) within each domain-associated RBFs, the Euclidean norm, easily computed as point-to-point Euclidean distance, can be used without interplay be-

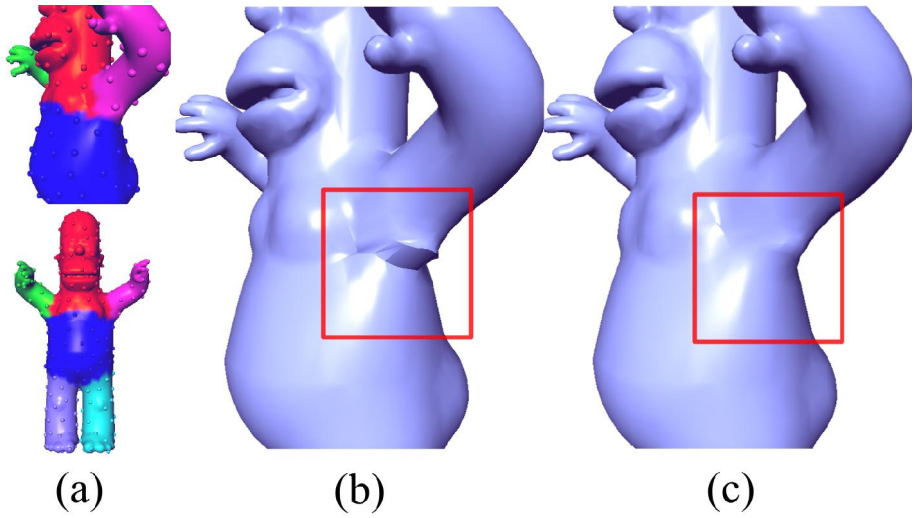

Figure 5.4 Animation of the arms (a). (b) Without a smoothing as post-processing, sharp features appear at and around the boundaries of contact regions, indicated by different colors in (a). (c) We employ Laplacian smoothing (2 iterations) to remove the artifacts.

tween branches, see Figure 5.2; (ii) in the case of the large number of samples, the likelihood of emerging ill-conditioned matrix is reduced.

As Figure 5.3 shows, for complex models such as the human hand, our method also works well if a proper weight map per bone is provided. In this example, RBFs with geodesic distances [Xin & Wang 2009] yields similar results to our method, however, it may have drawbacks of, e.g., low order smoothness at and around contact regions. This is because geodesic-based single RBF cannot strictly distinguish the vertices that are close to each other but lie inside different domains. Note that geodesic distances also heavily depend on the object shape, for example the mesh connectivity, leading to a loss of generality.

However, since each domain-associated system independently interpolates the surface of segment, therefore, without union operators, sharp features would appear at and around the boundaries of contact regions. To cope with them, our solution is to apply Laplacian smoothing.

Laplacian smoothing

First, triangles occupying multiple segments are easily detected as the contact regions. Then, for vertices v_i of these triangles, we iteratively

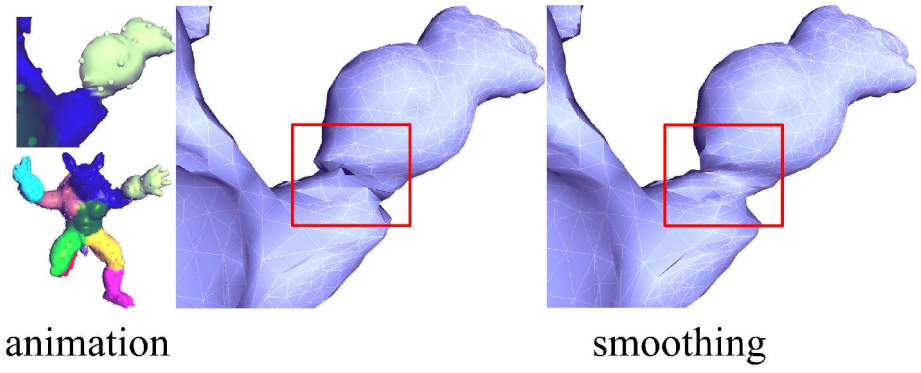

Figure 5.5 Elbow twisting. The Laplacian smoothing method is applied (5 iterations), resulting in better deformation at the elbow.

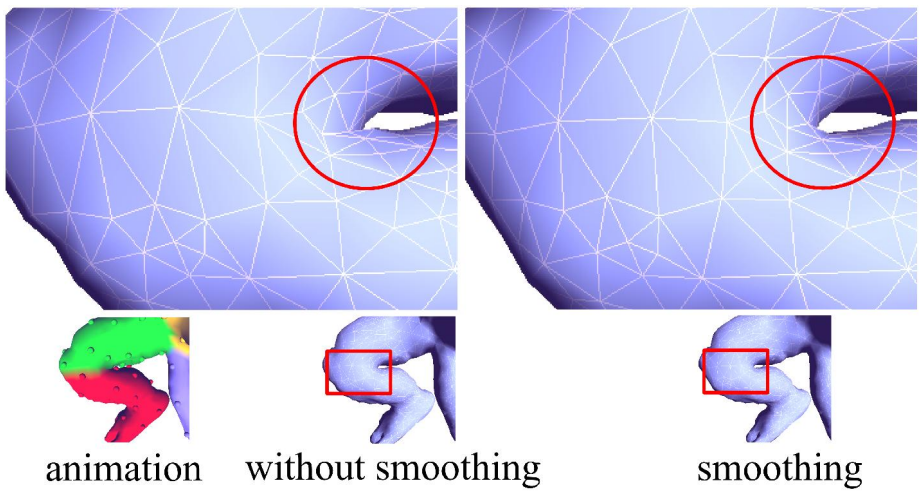

Figure 5.6 Leg bending. Laplacian smoothing (3 iterations) enables higher-order smoothness.

employ Laplacian smoothing, specifically in the form of

$$v_i^{t+1} = v_i^t + \frac{1}{\sum_{j \in \mathcal{N}(i)} \omega_{i,j}} \sum_{j \in \mathcal{N}(i)} \omega_{i,j}(v_j^t - v_i^t), \tag{5.2}$$

where $\mathcal{N}(i)$ are one-ring neighbors of v_i, and $\omega_{i,j}$ are weights computed by the cotangent weighting scheme [Meyer et al. 2003]. In our implementation, the negative $\omega_{i,j}$ are replaced with zero. This smoothing method is simple and satisfies our expectation even in complex motions including twisting and bending, see Figure 5.4 - 5.6.

Computation analysis

In solving the linear system resulting from a radial basis interpolation function, we have time complexity typically cubic in number of RBF centers. The computation mainly includes a one-time, pre-computed LU decomposition of matrix and the runtime evaluations on vertices. In multi-domain smooth interpolation, we LU-decompose multiple, small matrices instead of a single, large one. In practice, our results show that this way is efficient, especially in the case of a dense sampling on surface, and allows for interactive interpolating rates. Euclidean distances, faster-to-compute than geodesic distances, also can be pre-computed and cached, hence only the runtime calculations of coefficients are involved, further speeding up the method.

5.3 RESULTS

We have implemented our method in C++ on a laptop with an Intel Core i7 2.4Ghz processor and 6 GB memory. The LU decomposition is provided by the Eigen library [Guennebaud 2011].

Our method of smooth embedding, based on radial basis functions with compact support, is smooth enough to interpolate surfaces for high-quality visualization and rendering. To illustrate this advantage, three Shepard's interpolation methods are compared with our method: trilinear embedding, blending of transformation matrices, and blending of dual quaternions. As Figure 5.7 shows, these three methods are not as smooth as RBF interpolation, as they result in unpleasing surface visualization of a high-resolution polygonal mesh.

An experiment is conducted to investigate how the number of samples affects the performance. We increase the number of RBF centers, report the deformation results and the resulting timings, shown in Figure 5.8 and Table 5.1, respectively. Notice that the total time of our

Figure 5.7 Space deformation in Finite Element simulation of voxels as the volumetric representation of a high-resolution mesh. Trilinear interpolation, linear blending of transformations or dual quaternions (DQ), are not smooth enough, resulting in a pattern of stripes on the surface. Our space deformation method, based on radial basis functions with compact support, maintains the smoothness, see the interpolated surfaces visualized under the same rendering setting, and the close-ups.

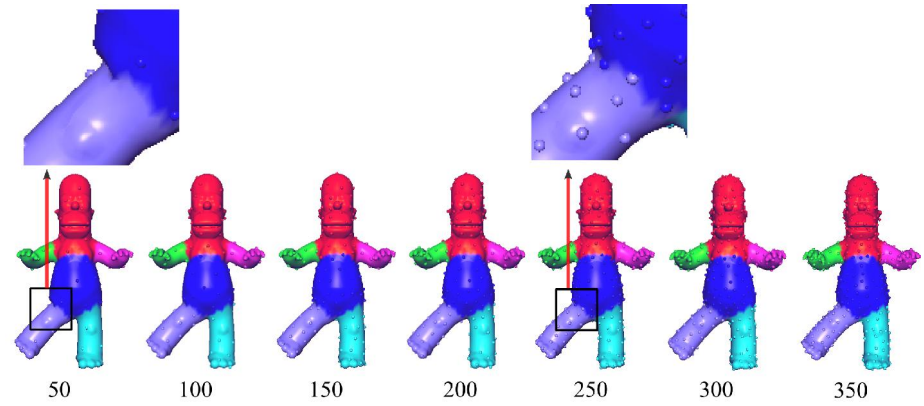

Figure 5.8 Increasing the number of samples (dots on surface), shown as the numbers below, gives rise to similar results, but may improve the smoothness, see skin regions in black boxes.

	Method	50	100	150	200	250	300	350
Initialize (ms)	Single	1.02	3.92	13.63	27.98	52.34	91.48	144.99
	Multi-domain	0.18	0.43	1.08	1.90	3.26	5.53	8.51
Runtime (ms)	Single	19.54	30.51	44.94	57.81	72.56	86.56	103.09
	Multi-domain	28.17	42.52	58.77	74.11	89.81	105.73	119.82
Total (ms)	Single	20.56	34.43	58.57	85.79	124.90	178.04	248.08
	Multi-domain	28.35	42.95	59.85	**76.01**	**93.07**	**111.26**	**128.33**

Table 5.1 Space deformation timings: Multi-domain interpolation shows advantage increased in number of samples. Note that the additional runtime cost in our method, with respect to single RBF interpolation, is introduced by the smoothing step rather than solving multiple linear systems themselves. The time is in milliseconds, the mesh has 5,103 vertices and 10,202 triangles.

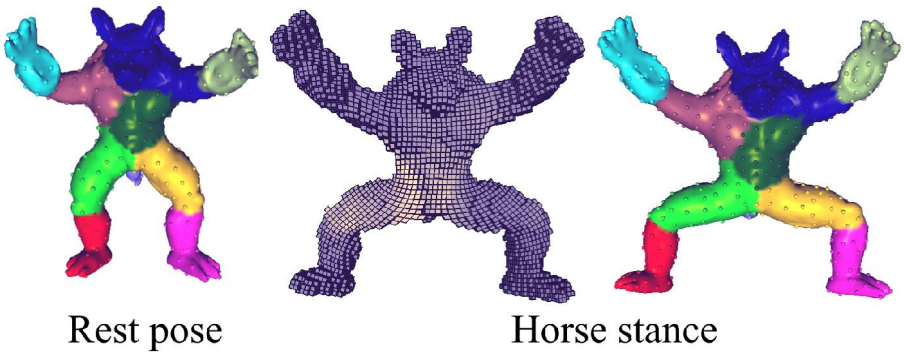

Figure 5.9 Our method is applied in volumetric PriMO energy with voxels.

method includes the time for domain decomposition and Laplacian smoothing (2 iterations), which are two pretty lightweight procedures involving simple computations as explained above. Our method can achieve interactive even real-time interpolating rates, and results in good deformations, with a reasonably small number of samples. Compared to single RBF, multi-domain RBF are more efficient for a denser sampling on surface.

The versatility of the method is tested both in simulation of deformable solid and volumetric energy, where surface deformation is decoupled from interior simulation and volumetric deformation, respectively, by embedding, in order to obtain substantial speedups and simplification of relevant volumetric modeling while maintaining high-quality visualization of the deformed surface. Recall that the points (RBF centers) sampled on the surface are embedded inside the vol-

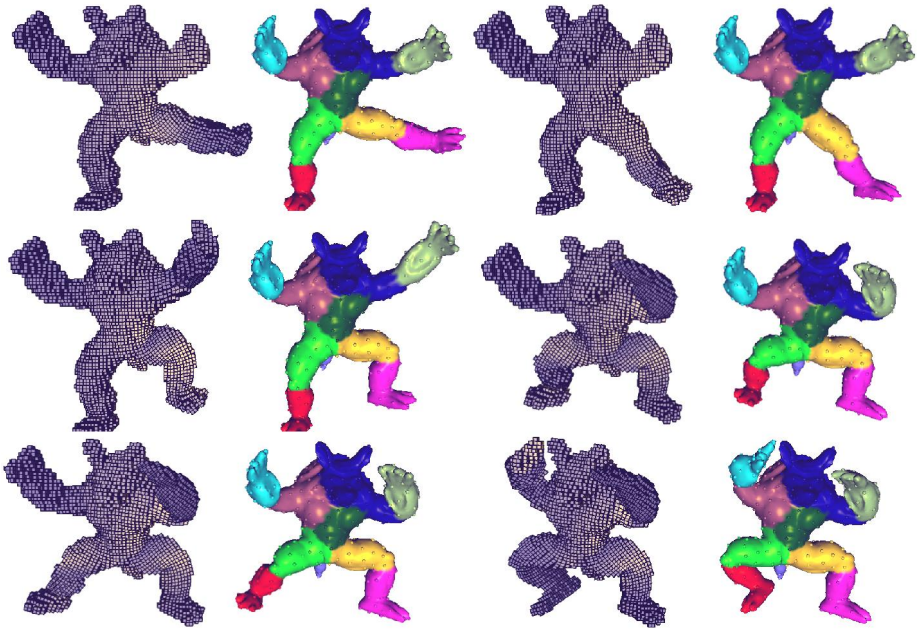

Figure 5.10 Examples of applying our method in volumetric PriMO energy.

umetric mesh by trilinear embedding, and the surface is deformed by computing space deformations using our method. The domains are achieved based on skinning weights. However, instead of manual painting, we compute the weights using an automatic-weighting method [Dionne & de Lasa 2013] simply because it works with production meshes. Thus, the decomposition may yield domains that do not perfectly correspond to the body parts in an anatomical meaning, see Figure 5.9. Nevertheless, due to this setting, the results can demonstrate the effectiveness of our method when it is fully automated. As Figure 5.9 and 5.10 shows, our method is successfully employed in volumetric deformation based on volumetric PriMO energy [Botsch et al. 2007b]. Figure 5.11 shows that our method also succeeds in soft body animation in particular based on Fast Lattice Shape Matching [Rivers & James 2007a].

In more detail, the volumetric PriMO energy is a computationally expensive, nonlinear deformation technique. In its test of the Armadillo model with 5,406 vertices, 10,808 triangles and 300 samples on surface (Figure 5.9), the space deformation using multi-domain smooth embedding only takes average 0.07 s per frame, whereas the volumetric deformation costs average 1.64 s per frame, showing that the runtime

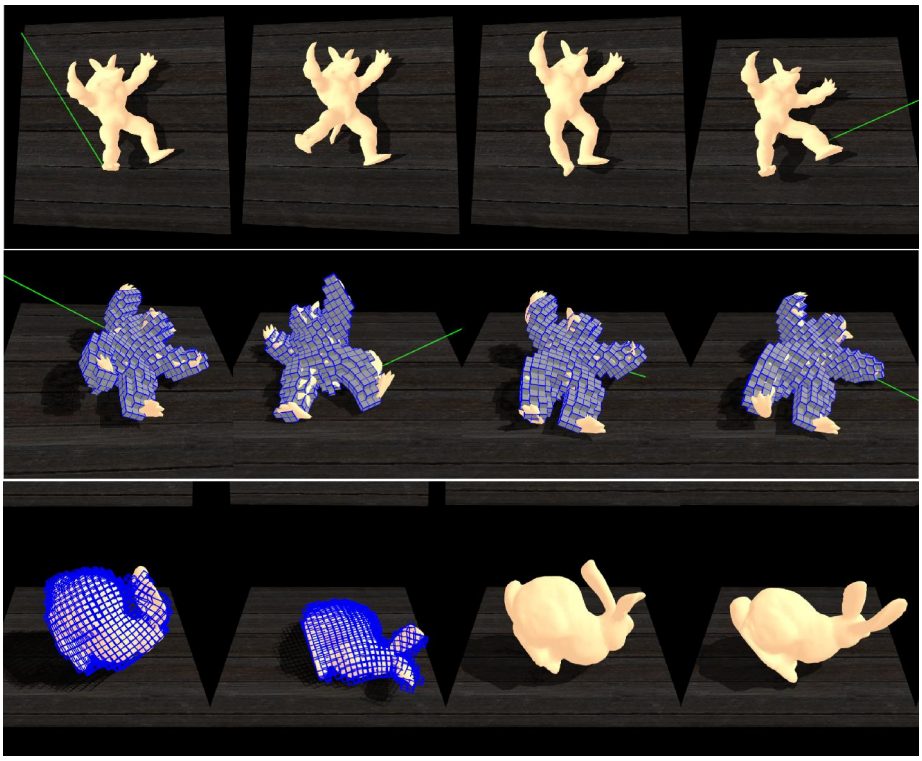

Figure 5.11 Soft body animation: Our space deformation method is applied in Fast Lattice Shape Matching simulation. The green line indicates the force exerted on the lattice.

cost of space deformation is minimum without introducing significant overload. Since our method also supports smoothness and local controllability, hence it can be elegantly combined with the aforementioned approaches that add realism by supporting interior dynamics and or volume preservation to deform the character surface.

Our method is also tested in a context of meshless simulation [Müller et al. 2005, Müller & Chentanez 2011] where the simulation is performed on a set of particles. Particularly, we opt for the one based on shape matching that is proposed by Müller et al. [Müller et al. 2005]. Given the skinning weights, domain-associated RBF interpolation systems are constructed from the sampled points (RBF centers) on surface. To deform the target surface during the simulation, we need compute the displacements of the RBF centers based on the positions of the simulation particles. However, there is no explicit spatial coherence between the mesh vertices and the particles for embedding. In our

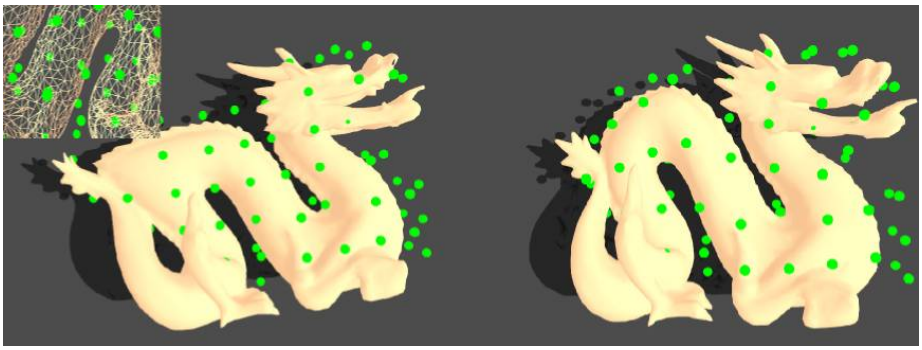

Figure 5.12 Our space deformation method is applied in a meshless simulation environment by exploiting spatial coherence, in the absence of explicit, spatial relationship between the simulation particles (green dots) and the target surface. The close-up shows the wire mesh.

implementation, we search $k = 8$ spatially nearest simulation particles for each RBF center, in particular using the Approximate Nearest Neighbor (ANN) searching algorithm [Arya & Mount 1998]. Then, the displacement of a RBF center is the weighted sum of the eight particles' displacements. Here the weight of a vertex regarding a spatially coherent particle is simply inverse proportional to the distance between the vertex and the particle. Figure 5.12 shows soft body animation of a dragon model that is produced by the meshless simulation [Müller et al. 2005] combined with our embedding method.

5.4 CONCLUSIONS

In this chapter, we have presented a multi-domain space deformation for character deformation. Our method is based on radial basis functions and domain decomposition, providing high-order smoothness of the deformed surfaces, and local control over deformations of a branch. Domains of a character were obtained according to skinning weights, each domain was attached to a linear RBF system and the seam artifacts were resolved by Laplacian smoothing. Our method was applied as space deformation in several scenarios including physics-based character simulation based on the Finite Element Method, geometrically-motivated soft body animation, and volumetric deformation based on minimization of a nonlinear, elastic energy. The results demonstrated the effectiveness.

For a skinned character with many bones, domains with very small

number of samples are likely to be obtained, making the resulting interpolation matrices inefficient as too few coefficients will be determined for evaluations. To overcome this, a possibility is to group such domains each which contacts with which others and then update the RBF systems. Another limitation is that our method itself is not able to handle skin contact effects, muscular bulges and elastic deformations which are often left to physics-based simulations [Barbič & James 2005, Kim & James 2011] in space deformation, hence the method is not well suited to direct skinning.

In deformable character animation, generating a sequence of skin deformations by user manipulations is called character rigging. For this procedure, generally an artist-friendly rig control is required in order to reduce the man effort as the artists do not need to directly manipulate the skin mesh. For example, the skeleton is often used in animation authoring, but it has certain drawbacks which will be explained in the next chapter. We will present a novel rig control based on 3D points as handles in the next chapter, which overcomes the limitations introduced by the skeleton.

Flexible Point Handles Metaphor for Character Deformation

6

In this chapter we describe a novel rig control for character skinning based on exploiting the 3D points as skinning handles. The space of deformations provided by the skeleton-based blend skinning is expanded by a deformation metaphor we proposed based on point handles. Our point handles support rigid bending, in addition to stretching, twisting, and supple deformations which are difficult to achieve using a skeleton consisting of rigid bones. The point handles are easily placed inside the character body in an automatic manner. Only the pseudo-edges connecting the point handles need to be manually provided by users in order to automatically compute the rotations of point handles from user prescribed translations. However, the connections are freely defined and do not necessarily result in a skeleton hierarchy. Thus, our skinning scheme still requires much less man-effort than the skeletal skinning.

To date, skeletal skinning has dominated the practical usages in deformable character animation due to its simplicity and efficiency. In this skinning scheme, the skeleton is a hierarchy of rigid bones referred to as *handles* inside the character, and the skin is moved as an explicit function of handle transformations.

A manual pipeline of skeleton-based character rigging mainly consists of three sequential phrases: first, designing and fitting a skeleton into the character body, and second, painting the influence weights per bone, and finally specifying the transformations of a set of bones. Before painting, usually a method that automatically computes the weights is applied in order to reduce the time, as in this context artists only need to adjust some of the weights. There are several cases where the number of corrections is reduced. First, the computed weights are good enough. Second, artists can easily determine the influencing bone(s) of a body segment so that they will intuitively repaint the regions assigned with improper weights. At last but not least, the rotation center of each bone is correct enough, as artists will transform the bones for animation previewing, which is necessary to detect problematic regions and would be repeated. A common solution is to fit a skeleton to the character body and anatomically collocate it with the

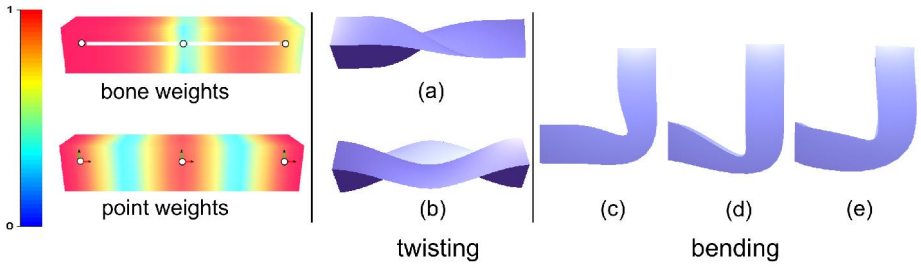

Figure 6.1 The weights are visualized using a jet colormap. The white chain (left) denotes the skeleton and the coordinate frames denote point handles. **Twisting**: by twisting the middle joint (point handle), the deformation generated by LBS with bone weights (a), and LBS with point weights (b). **Bending**: a rigid bending generated by LBS with bone weights (c), and a bending (not rigid) by LBS with point wights (d). Our scheme aims to approximately recover the local rigidities of a region between two point handles (e).

skin. However, this solution is cumbersome, as transforming a joint often shifts its incident bone(s) outside the body due to the skeleton hierarchy. The hierarchical structure also complicates skeleton-based automatic skinning. On the one hand, automatic skeleton extraction [Shapira et al. 2008] does not guarantee that the resulting line segments lie inside the object's volume. On the other hand, embedding an existing skeleton inside a shape is a difficult problem. Although well embedded skeletons have been produced, they were achieved by solving a continuous optimization of joint locations specific to humanoid shapes [Baran & Popović 2007], or a nonlinear optimization where users provide joint location hints [Jacobson et al. 2014].

In contrast to the skeleton, point handles are easy to place inside an object's volume, and each point handle is independently positioned without influencing the positioning of other point handles. Moreover, point handles with associated weight functions (e.g. [Jacobson et al. 2011]) are highly suitable for handling stretching, twisting [Jacobson & Sorkine 2011], and supple skin regions [Jacobson et al. 2011], which are difficult to achieve by linear blend skinning (LBS) with only a scalar weight function per bone. However, LBS with point handles are unsuitable to bend limbs rigidly if the point handles are placed at joint locations so that they are rigged as intuitively as joints. In this chapter, the influence weights computed for point handles are referred to as *point weights*, and the weights computed for bones are referred to as *bone weights*. As Figure 6.1 illustrates, because point weights by con-

struction vary over the shape more than bone weights which only vary around joints, LBS with them respectively will result in rather different deformations. LBS with point weights produces interesting twisting spread over the entire shape, while LBS with bone weights twists the shape only near the middle joint. LBS with bone weights produces a rigid bending, while LBS with point weights fails. Alternatively, for a rigid bending point handles can be placed at bone centers if one already knew the characteristics of point weights. However, the intuitiveness of specifying transformations is lost.

Because of the aforementioned flexibility of point handles, we propose a fully point handles based skinning scheme that also produces rigid bending desired for articulated deformable characters. This novel scheme will richly expand the space of deformations possible with LBS with only per point handle scalar weight functions.

To support rigid bending, we propose a method that automatically recovers the local rigidities of limbs via minimizing a surface-based, as-rigid-as-possible deformation energy. The minimization problem is subjected to deformed positions, computed by LBS with point weights, of a set of point handles' nearby vertices. As point handles are placed exactly at joints which are located where they would be in a real world skeleton, a region between two point handles deforms as-rigidly-as-possible as a rigid limb by solving the constrained minimization problem. Point handles are not in a hierarchical structure as rigid bones, hence inverse kinematics or motion capture techniques cannot be applied to facilitate the input of transformations. Instead, their transformations have to be manually specified through a two-dimensional interface common to animation authoring. To reduce the degrees of freedom, a functionality inferring rotations from translations is developed. So only translations, easily and intuitively prescribed by moving the mouse, are required to users. Compared to automatic skeletal skinning, automatic skinning with point handles is easily achieved based on a mesh-segmentation method.

Our scheme has shown advantages by a variety of experimental results:

i. It provides a cheap and robust way of producing rigid bending, and maintaining shape details, in terms of user interactions and computational time.

ii. It allows versatile posing by adjusting the positional constraints of the as-rigid-as-possible deformation energy on-the-fly.

iii. It retains the intuitiveness of rigging a skeletal character by placing point handles at joint locations, and the simplicity of LBS.

6.1 PREVIOUS WORK

Linear blend skinning (LBS) with a hierarchy of rigid bones gives rise to *joint-collapse* artifacts near joints such as shoulders, wrists, or even elbows if which exhibit large rotations. To cope with such artifacts, one possiblity is to expand the expressive power of LBS by either supplying additional weights per bone [Wang & Phillips 2002, Merry et al. 2006], or adding virtual bones expressed as their transformations and weights [Wang et al. 2007, Kavan et al. 2009]. The new data has to be automatically computed using example poses [Jacobson & Sorkine 2011], which often do not exist. Another solution is to linearly blend unit dual quaternions instead of transformation matrices, forming the method called dual quaternion skinning (DQS) [Kavan et al. 2007] which is more computationally expensive than LBS.

It is also possible to reduce the number of LBS artifacts by replacing rigid bones with other types of handles. Works such as [Yang et al. 2006, Forstmann et al. 2007] use curve skeletons to fix joint-collapse. Curved skeletons need new rigging controls that are inconsistent with the existing rigging pipeline [Kavan et al. 2008]. A lattice of control points, used in free-form deformation [Sederberg & Parry 1986], is easy to design, but it is unsuitable for the control of concave objects due to its regular structure. A cage [Ju et al. 2005], as a general control polyhedron loosely enclosing the target surface, has a better match of degrees of freedom to the enclosed geometry, thereby allowing precise area control such as bulging and thinning in regions of interest. However, a control mesh is tedious to establish and manipulate.

Points are perhaps the easiest handles to establish. Existing automatic bone weights methods such as [Baran & Popović 2007] may be trivially adapted to compute the influence weights for point handles. Jacobson et al. [Jacobson et al. 2011] proposed a method well suited for computing point weights. Although the method is slow to compute and requires a volume discretization, its resulting weights produce the highest quality deformations [Jacobson & Sorkine 2011]. Once good point weights are available, the transformation of a vertex at runtime is computed by linearly blending point transformations. LBS with point handles is suitable for handling stretching, twisting [Jacobson & Sorkine 2011], and supple deformation which requires a skeleton with many bones in skeletal skinning. However, it is much less effective

at bending limbs rigidly than skeleton-based LBS. Furthermore, LBS, as a closed-form blending technique, has no shape-preserving property. Variational methods [Sorkine & Alexa 2007, Botsch & Sorkine 2008] compute high-quality shape preserving deformations for arbitrary handles at points, but at a greater runtime cost.

Since each of the aforementioned handle types shows its own advantages, a hybrid of two or more handle types has been demonstrated to expand the space of deformations possible with LBS: a hybrid of point and bones [Jacobson & Sorkine 2011], and a hybrid of points, cages and bones [Jacobson et al. 2011]. Although the flexibility of point handles has been demonstrated, until now a fully point handles based skinning method remains to be developed.

6.2 SKINNING WITH POINT WEIGHTS

To achieve rigid bending with point handles is difficult. On the one hand, this is because it is not so clear where the deformation point handles should be placed for rigid bending. On the other hand, rigging the point handles for rigid bending is not as intuitive as rigging a skeleton, in which case the point handles are intuitively and anatomically placed between conjoint near-rigid segments. In contrast, tediously and carefully rotating the point handles is required, as point weights vary over the region between two point handles.

Assuming that good point weights are available, it is no doubt that associated weights of the vertices in some distances to each point handle fall into a rather small range close to 1. These vertices form a small region in the vicinity of each point handle. Here, good weights should at least satisfy the properties of locality and smoothness. So in the context of blend skinning, on the one hand the deformation of the proximal region per point handle is nearly rigid. In other words, the weight map per point handle captures the local rigidities of such a region. On the other hand, the region's deformation accurately and intuitively fits the transformation of the point handle since the associated weights are large enough.

Our solution is to impose a rigidness regularization to the vertices between two point handles, by solving an energy minimization problem. The objective function is subjected to the positions of proximal vertices per point handle, which are computed in the spirit of LBS. Thus, our framework retains the simplicity of modeling the user constraints as blend skinning, while the users specify the handle transformations rather than directly prescribe the vertex positions.

6.2.1 *Modeling framework*

Given a skinning model consisting of H point handles, a skin mesh S containing N vertices $\{v_1, ..., v_N\}$ and M triangles $\{t_1, ..., t_M\}$ at rest pose, each vertex v_i is bound to the handles by influence weights expressed as a vector $\mathbf{w}_i = (w_{i,1}, ..., w_{i,H})$. Given the transformations of point handles $(C_1^f, ..., C_H^f)$ at frame f, optimal deformation is computed by solving the minimization problem

$$\begin{aligned} \min \quad & E_s(S^{'}) = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_1(v_i)} \omega_{i,j} \parallel (v_i^{'} - v_j^{'}) - R_i(v_i - v_j) \parallel^2 \\ \text{s.t.} \quad & v_k^{'} = v_k \sum_{j=1}^{H} w_{k,j} C_j^f, \quad k \in \mathcal{H}, \end{aligned} \tag{6.1}$$

where $\mathcal{H}$ is the set of indices of the proximal vertices, $\omega_{i,j}$ are per-edge weights between v_i and each one-ring neighbor $v_j \in \mathcal{N}_1(v_i)$. R_i of vertex v_i is derived using the same procedure described in chapter 2.

The solution consists of two steps. First, compute an initial guess as $v_i^{'} = v_i \sum_{j=1}^{H} w_{i,j} C_j^f$, and approximate the local rotations R_i based on $v_i^{'}$. Second, new positions $v_i^{'}$ are obtained by solving Eq. (6.1). We follow the same procedure as [Sorkine & Alexa 2007] to compute the partial derivatives w.r.t. $v_i^{'}$. By setting the partial derivatives to zero w.r.t. each $v_i^{'}$, a sparse linear system of equations is achieved, which can be solved efficiently by the sparse Cholesky factorization. The minimization is iteratively performed by alternately re-computing local rotations and solving the equation until it reaches the number of iterations specified by users.

Closely similar to the surface modeling framework [Sorkine & Alexa 2007], our framework also minimizes the changes of edge lengths to obtain as-rigid-as possible (ARAP) deformations. However, we compute positional constraints in the sprit of linear blend skinning. Thus, users only need to move the point handles without manually selecting a set of vertices on the surface. Also, an initial guess is computed by LBS rather than using the Laplacian surface editing (LSE) [Sorkine et al. 2004] method. Since LBS has a closed-form formula, it is much faster than LSE, and can produce a reasonable degree of local rigidity in bending with point weights. Thus, the number of iterations required to reach a desired rigid bending will be reduced w.r.t. the LSE-based solution.

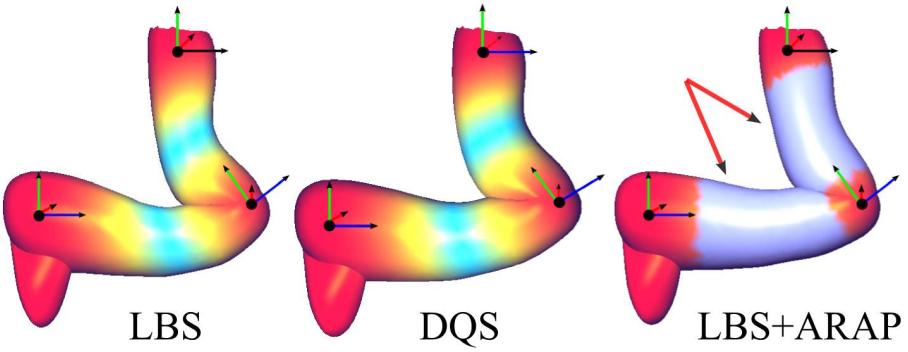

Figure 6.2 Point weights are displayed as colors in the two images on the left. Both LBS and DQS with point handles are unsuitable to bend a shape rigidly. Our scheme (LBS+ARAP) automatically refines the region between two point handles to be as-rigid-as-possible via minimizing a rigidity energy constructed over the shape. The minimization problem is subjected to the positions of a set of point handles' proximal vertices (reddish parts of the rightmost image).

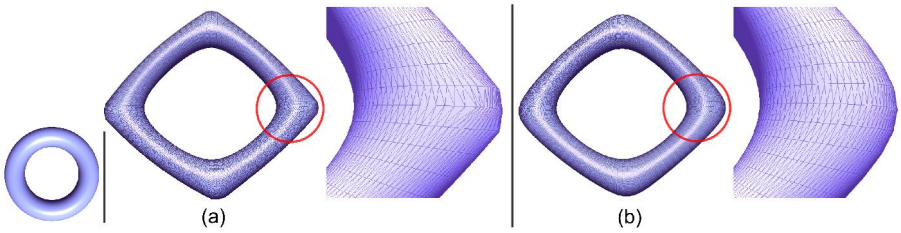

Figure 6.3 The torus is deformed. (a) Discontinuities appear around the point handles. (b) With a Lplacian smoothness regularization, such artifacts are suppressed.

Our method is depicted in Figure 6.2. Since the objective function is subjected to hard constraints consisting of computed vertex positions using LBS, discontinuities can appear near handle positions. Such artifacts can be easily suppressed, see Figure 6.3, by moving the vertices in the direction of the Laplacian

$$v'_{i\_opt} = v'_i + \frac{1}{\sum_{j\in\mathcal{N}(i)} \omega_{i,j}} \sum_{j\in\mathcal{N}(i)} \omega_{i,j}(v'_j - v'_i). \tag{6.2}$$

In fact, only the positions of proximal vertices need to be re-optimized. For the regions between two point handles, optimal results have al-

ready been reached due to the rigidness regularization resulting from a solution of Eq. 6.1.

6.2.2 *Degrees of freedom*

The transformations of point handles are manually specified through a 2D user interface common to animation authoring. Through such an interface, rotations that are necessary to LBS (see Figure 6.4), however, are non-trivial and less intuitive to specify than translations that are directly proportional to mouse movements. We present methods that automatically infer rotations from user-specified translations.

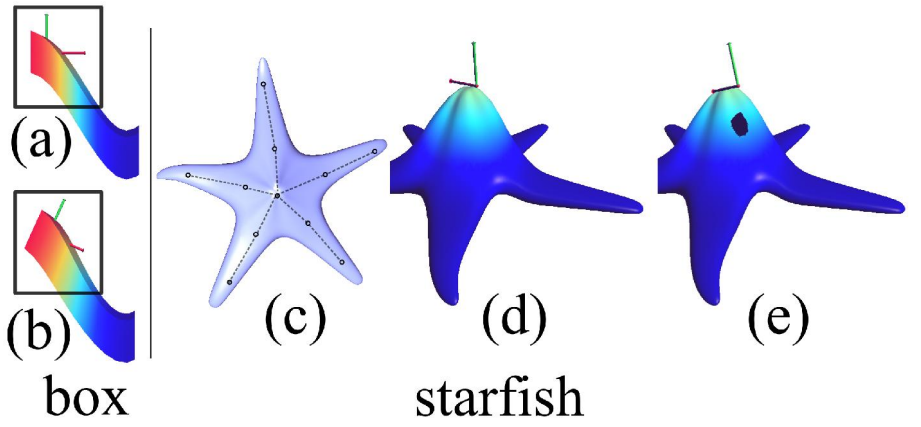

Figure 6.4 **Box**: With the presence of only translation of the transforming point handle (coordinate frame), LBS gives rise to shape distortion (a). Additionally providing a rotation removes the distortion artifact (b). **Starfish**: given the pseudo-edges (dash lines) (c), accompanying rotations are inferred from user-prescribed translations. For the middle point handle, the fast recursion sum method taking about 6 ms yields self-intersection, rendered into a black hole (e). The method based on computing eigenvector of a matrix addresses this limitation (d), though it takes about 16.6 ms.

Given P point handles denoted by $p_i \in \mathbb{R}^3, i \in \{1, ..., P\}$, each point handle has E user-defined incident pseudo-edges, defined as three-dimensional vectors $p_{i(k)} = p_i - p_j, k \in \{1, ..., E\}$ at rest pose, and $p'_{i(k)} = p'_i - p'_j$ after translations. The rotation between two vectors $p_{i(k)}$ and $p'_{i(k)}$ is represented using quaternion denoted by $q_{i(k)}$. The final rotation denoted by q_i of point handle p_i is an average of all $q_{i(k)}$, $k \in \{1, ..., E\}$.

Notice that, unlike the skeleton, the pseudo-edges do not necessarily lie inside the volume and conform to a skeleton structure. In fact, there is a large degree of freedom on connecting the point handles as long as no one is isolated, such that, for each point handle, the definition of the rotation axis is facilitated.

The half-quaternion method is used to calculate the rotation about an axis between two vectors, simply because it saves some square-root calculations and ensures a "shortest arc" solution. For an average of only two quaternions, the well-known spherical linear interpolation (Slerp) can be applied. In the presence of more quaternions, a simple and fast way is to average by a recursion sum algorithm

$$avg(q_{i(E)}) = \begin{cases} \frac{1}{E}(q_{i(E)} + avg(q_{i(E-1)})) & E \geq 2 \\ q_{i(1)} & E = 1. \end{cases} \tag{6.3}$$

This method yields a valid average only if the quaternions are relatively close to each other.

A more accurate method comes at the price of reduced speed. However, because the number of incident edges per point handle is small, the overload is minimal. This method first converts the quaternions $q_{i(k)}(w, x, y, z)$ as four-dimensional column vectors $\vec{q_{i(k)}}(x, y, z, w)$ and then build a matrix as

$$\mathbf{M} = \frac{1}{E} \sum_{k=1}^{E} \vec{q_{i(k)}} \vec{q_{i(k)}}^{T}. \tag{6.4}$$

The final quaternion is the (convert back) eigenvector corresponding to the largest eigenvalue of the matrix. Figure 6.4 depicts a visual comparison between two methods.

6.2.3 *Automatic skinning*

Point handles are easily placed within the character volume with the help of ray-triangle intersection for instance, and their positions can be anatomically guided to users. For example, the positions of point handles tend to be joint locations in a humanoid character. Targeting a fully automatic skinning scheme, a method is developed to automatically determine the placements of deformation point handles.

In articulated deformable characters, the thickness of a body segment is often different to its conjoint segments, and the skin of such a segment deforms as-rigidly-as-possible. Recall that in our method the region between two point handles is as-rigid-as-possible, thus, for

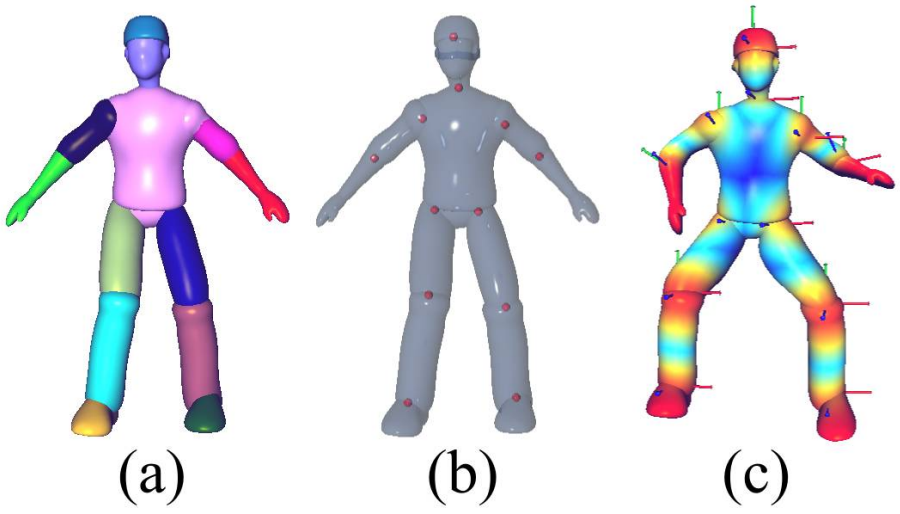

Figure 6.5 Segments with different thicknesses, indicated by different colors, are obtained based on SDF values (a). Point handles are automatically placed between conjoint segments (b), and their influence weights, displayed as colors in (c), are computed. The character is posed by translating point handles (coordinate frames). Note that rotations are inferred.

a polygonal mesh, it is natural to place the point handles between conjoint segments with different thickness. The shape-diameter-function (SDF) values [Shapira et al. 2008] measure the diameter of an object's volume in the neighborhood of each vertex on the surface, providing a direct way to detect near-rigid segments. Our method is based on SDF values.

First, a hard-clustering is performed on the raw SDF values to obtain pairwise facet and cluster-id. Second, candidate vertices occupying multiple clusters are picked out if their hosting triangles have a different cluster-id. For each candidate: a ray along the inverse normal is cast from the vertex position, and then a point as an intermediate result is picked on the line of the ray according to half of the associated SDF value. This ensures that all picked points lie inside the volume. Finally, the position of a point handle is the center of a spatially grouped intermediate points, guaranteeing that all point handles are inside the volume.

Once the skin is bound to deformation point handles by influence weights computed using an automatic weights method such as [Jacobson et al. 2011], the character is ready for animation. Figure 6.5 depicts

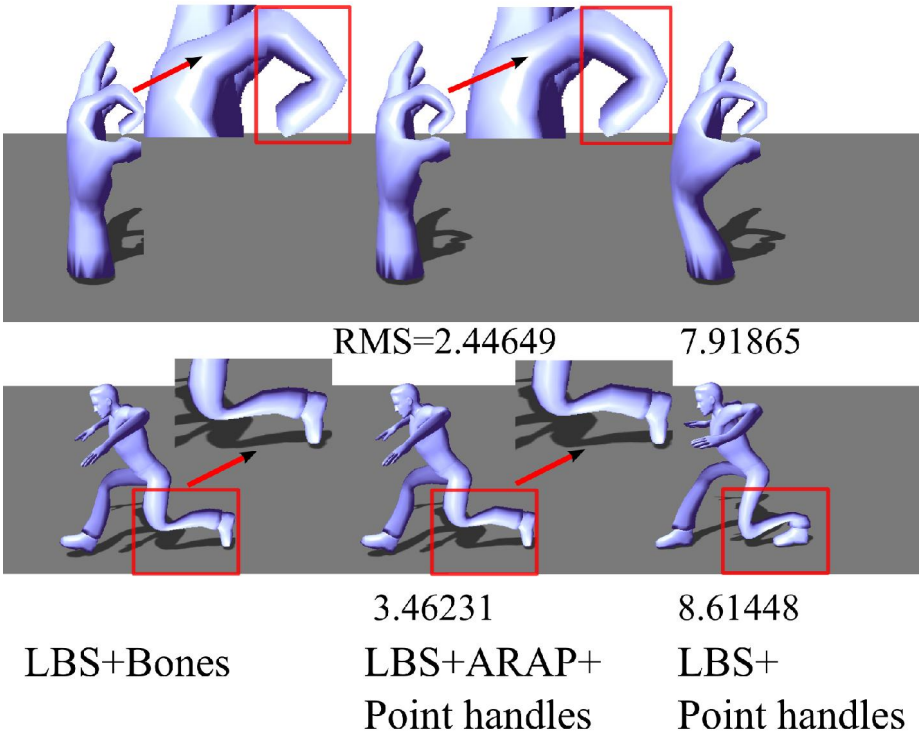

Figure 6.6 $l = 0.98$ for the handle model, $l = 0.95$ for the human model, and the number of iterations is 3, giving rise to the results. The numbers below are RMS errors. LBS with point handles fails to bend limbs rigidly. Our scheme gives rise to comparable, visual and quantitative results w.r.t. skeleton-based LBS.

our method applied to a humanoid character.

6.3 RESULTS AND DISCUSSION

Our method was implemented in C++ on a laptop with an Intel Core i7 2.4Ghz processor and 6 GB memory. The sparse Cholesky solver and two-sided Jacobi SVD decomposition, provided by the Eigen library [Guennebaud 2011], were used to solve the minimization formula and to compute rotations R_i, respectively. The cotangent-weight formula [Meyer et al. 2003] instead of uniform weights is used to compute per-edge weights $\omega_{i,j}$, avoiding deformation artifacts [Sorkine & Alexa 2007].

Bone weights are much more intuitive to paint by artists than point

weights. However, seeking an automatic scheme is a primary goal in skinning. Thus, to obtain fair comparisons, the influence weights of bones and point handles are computed. Bounded biharmonic weights (BBW) [Jacobson et al. 2011] are computed as influence weights for point handles and bones, respectively, because they are smooth, local and shape-aware, and have experimentally produced high-quality deformations [Jacobson et al. 2011, Jacobson & Sorkine 2011]. Since BBW requires a priori volume discretization, we obtain it by voxelization instead of tetrahedral meshing, avoiding dealing with self-intersection and non-manifoldness. The computation then could be largely increased since a voxel grid is usually denser than a tetrahedral representation, but it is a tolerable, one-time precomputation taking dozens of seconds for a polygonal mesh with reasonable geometric complexity.

Two animated models mainly containing deformations of rigid bending are used to demonstrate the effectiveness of our method. They include a hand model with 33 frames and 1,926 vertices, a human model with 54 frames and 3,557 vertices. Deformations resulting from skeleton-based LBS (LBS+bones) are considered as the ground-truth, as bones are naturally effective for rigid bending. Joint locations are directly used as point handle positions. For rendering of the deformed mesh, vertex normals are recomputed on CPU.

For each point handle, our method needs to determine $v_k, k \in \mathcal{H}$ (recall Section 6.2.1) as positional constraints. A vertex is selected if its largest weight is larger than a threshold l. Note that the associated weights of a vertex consist of a set of per-point weights, as mostly a vertex is influenced by multiple point handles. Due to the locality offered by BBW, such vertices are well located in the vicinity of point handles.

An error metric w.r.t. the ground truth is formulated to quantitatively evaluate the methods. The lower the computed value is, the more rigidly the limbs are bent. We first resize the models so that their rest poses are tightly enclosed by a unit cube. Then, the metric is calculated as

$$E_{RMS} = \frac{\sum_{f=1}^{F} \sum_{i=1}^{N} \parallel bv_i^f - pv_i^f \parallel^2}{\sqrt{3NF}}, \tag{6.5}$$

where F denotes the number of frames, N is the number of vertices, bv^f and pv^f are deformed vertices at frame f resulting from a method with bone weights and point weights, respectively. This form of metric has been reported to be less sensitive to global motions [Kavan et al. 2010].

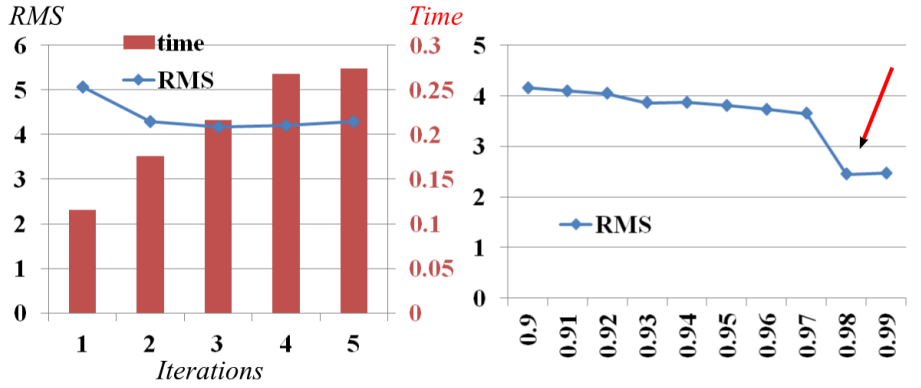

Figure 6.7 **Left** ($l = 0.9$): Computational time (in seconds) increases along with the number of ARAP iterations. The shape is bent more rigidly by increasing iterations, hence mainly the error is reduced along with iterations. By specifying 3 iterations, the lowest error is obtained. **Right** ($iterations = 3$): The pattern of the discrete curve shows that mainly the error is reduced by increasing l. In the handle model, mostly, the largest weights of the vertices in the vicinity of a point handle lie in the range from 0.97 to 0.98. Specifying l as a value in this range yields positional constraints, in turn deformations, different to the results generated by other values also in that range. As a result, a sudden drop happens.

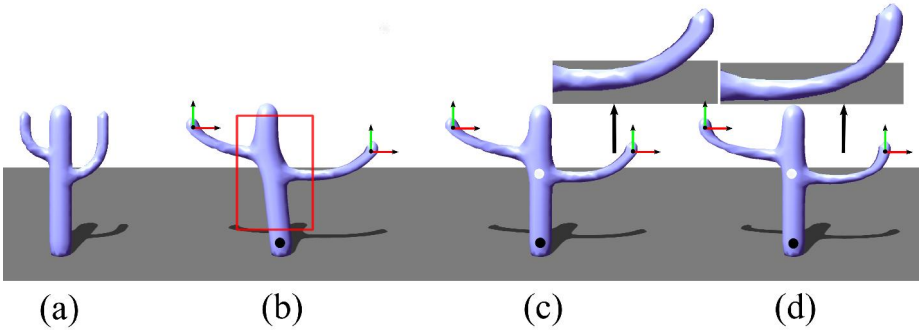

Figure 6.8 The cactus at rest pose (a) is skinned with four point handles. A pose is achieved by moving two point handles in branch (coordinate frames) and fixing the one at the base (b). In this case, only these three point handles are used to define positional constraints of ARAP. If one desires locally controlled deformation of a branch, our scheme allows it by additionally enabling the middle point handle (white) to provide its proximal vertices as new constraints on-the-fly (c). LBS also produces such a deformation style (d), but the branches are over-stretched, compared to our scheme, see close ups.

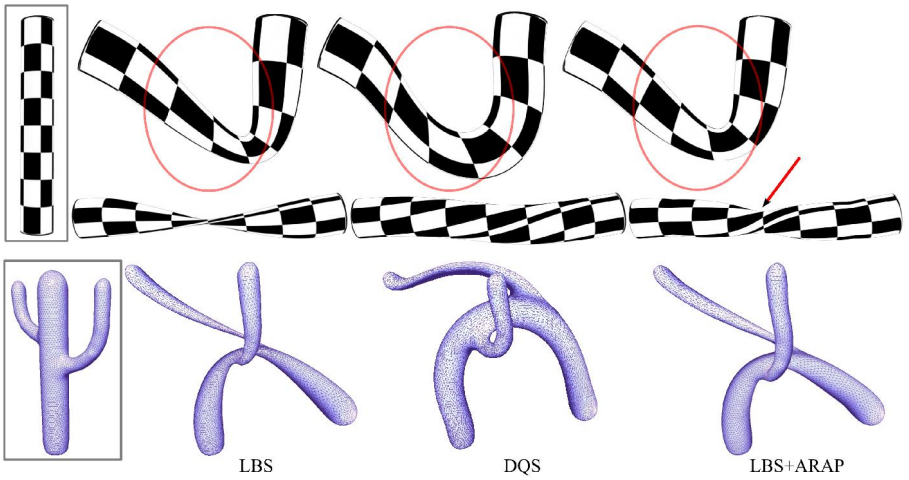

Figure 6.9 LBS and DQS do not maintain surface details during deformations, while our method (LBS+ARAP) does. DQS aims at volume preservation, but without considering the realism of posture. For example, the interesting twisting of the cylinder is lost when using DQS. Models in rectangle are in their rest poses.

As Figure 6.6 shows, LBS with point handles fails in producing rigid bending. Our method, called *LBS+ARAP+Point handles*, produces better visual and quantitative results.

Relationships between computational time and the number of iterations used to solving the minimization problem, called *ARAP iterations* for short, between E_{RMS} and the number of ARAP iterations, between E_{RMS} and threshold l, are tested on the hand model. Figure 6.7 reports the results. Though the computational time increases along with the number of iterations, our energy minimization runs three iterations to yield the lowest error. In cases of larger iterations, the shape is bent more rigidly by our scheme than by skeleton-based LBS. So, the error then is slightly increased. This happened as the bone lengths of the hand model are small. Since BBW yields local and shape-aware weights [Jacobson et al. 2011, Jacobson & Sorkine 2011], the threshold l is decided intuitively, reducing user errors. As a result, no sharp line is observed in the right subplot.

Discussion. The use of a skeleton is known to facilitate the input of bone motions by either motion capture techniques or inverse kinematics. Currently, point handles' counterparts are joints of a corresponding skeletal model in order to maintain an anatomical embedding. Due to

this setting, bone transformations were directly reused as point handle transformations in our experiments. As a consequence, the aforementioned techniques (e.g. inverse kinematics) facilitating motion input have been investigated in skinned models based on the point handle metaphor, and encouraging results have been obtained.

Compelling deformations can be obtained by ARAP (as-rigid-as-possible deformation energy) with few user edits. This has been demonstrated in ARAP-based surface modeling [Sorkine & Alexa 2007] for instance. This advantage can be augmented to reduce the degrees of freedom for certain character postures in our skinning scheme based on point handles. Given a skinned model with many point handles, the user specifies only a subset of the degrees of freedom, and the rest deformations are automatically determined using the nonlinear ARAP. This is similar to the method [Jacobson et al. 2012] that uses nonlinear, rigidity energies to infer the rest skinning transformations. It is also allowed to adjust the positional constraints of ARAP on-the-fly. As Figure 6.8 shows, an extra point handle is enabled to provide its proximal vertices as positional constraints. As a result, resulting deformations are locally controlled if desired. In this case, the newly enabled point handle actually serves to occlude the rotation propagations from the moving point handle to the regions far away. As Figure 6.8 and 6.9 show, closed-form blending techniques, such as LBS and DQS, do not support shape-details preservation, but our scheme dose.

6.4 CONCLUSIONS

In this chapter, we have presented a novel skinning scheme based on point handles, which are much easier to design and embed into the character body than a skeleton. Our method addressed the limitation of bending, when using a conventional blend skinning with point weights, by optimizing the vertex positions using a nonlinear, rigidity energy. We have demonstrated the effectiveness of our method by experimental results of two animation models mainly containing rigid bending. Automatic skinning has been explored in order to reduce the manual work. Also, our method allows for versatile posing and it is robust to large deformations.

Currently, point handles are considered as counterparts of joints. In cases of skinned models with long bones, the mesh part wrapping around a long bone might not be rigid enough when bending by our scheme. This is because we only defines constraints on edge lengths in order to obtain substantial speedups. To overcome this, a possibility is

to automatically place the point handles based on *Schelling points* [Chen et al. 2012], which is a method used to predict that where 3D points as handles are likely to be on a mesh. To further obtain speedups, existing weight reduction techniques [Landreneau & Schaefer 2010, Le & Deng 2013] can be incorporated into our skinning scheme to reduce the computational cost of a dense-weight skinning model.

Part IV

Concluding Remarks

Conclusion

7

In this thesis we have presented the results of various studies on elastic deformation with a focus on deformable character animation. Our research has resulted in a number of scientific contributions, which we will summarize here.

In Chapter 3 we studied physics-based virtual human modeling, in particular the simulation of skin dynamics. We have presented a method for modeling and simulating the human neck. Relevant anatomical structures available in a 3D model of human musculoskeletal system were modeled as deformable or linked rigid bodies. We coupled the soft-hard bodies using soft constraints via elastic springs, and bound the skin to underlying bodies in an anatomical manner. Finally, we formulated a Lagrangian dynamic system numerically solved by time integration. Our work involved linear elastic constraints for soft-hard bodies coupling, manual mesh pruning, manual muscles-bones coupling, automatic skin-muscle connection based on the closest point projection, semi-implicit integrator, making our approach very pragmatic. The simulations were performed at interactive rates on a modern computer. Experimental results were provided to show the high level of realism our model offered. In a conclusion, we successfully found the tradeoff between speed and accuracy to obtain a realistic yet interactive simulation.

In Chapter 4, we have presented two methods that can approximately preserve the volume of an object and or the shape locality. The use of a voxel grid as the volumetric representation led to a simplification of the implementation and computational acceleration of the methods. Three space deformation methods, trading off speed against realism, were developed to project back the volumetric deformations to the target surface. The results showed the effectiveness in character deformation, implying that our methods can be used in computer games and animation. In particular, the method based on energy minimization efficiently suppressed well-known artifacts such as candy wrapping and joint collapsing of linear bend skinning, and joint bulging of dual quaternion skinning.

In Chapter 5, we have presented a method that computes space deformations for deformable character animation based on domain-

decomposition and scattered data interpolation. Given a character, we partitioned it into multiple domains according to skinning weights, and attached each to a linear system without seam artifacts. Examples were presented for articulated deformable characters with localized changes in deformation of each near-rigid body part. The results showed that our method supported smoothness and local controllability of deformations, and achieved interactive interpolating rates. Our method can be used in combination with deformation energies, known to provide preservation of shape and volumetric features, or simulation of deformable bodies, known to enable interior dynamics. As a result, our method is suitable for character deformation where elastic deformations are desired.

In Chapter 6 we studied the flexibility of character rigging. To reduce the man effort required by skeletal skinning and allow for larger space of deformations, we developed a skinning scheme based on point handles, which are automatically placed inside the character body. The scheme bends limbs nearly as skeletal skinning does, and also effectively accomplishes other tasks such as shape details preservation and supple deformations, which are difficult to achieve by rigid bones. The science behind our scheme lies in the minimization of a surface-based, nonlinear rigidity energy subject to point handles. The effectiveness of our scheme was demonstrated by a variety of experimental results, showing that it could be an alternative to skeletal skinning.

7.1 OUTLOOK

The work in this thesis presents a step forward in the field of character deformation. Even so, there are numerous directions for interesting future work. In this section we will discuss some possible guidelines.

In Chapter 3, we only presented the empirical results of the human neck simulation. This weakness could be overcome by performing detailed studies: how the manual pruning of muscles, the bone-muscle coupling, and the number of simulation elements impact on the resulting visual quality. The resulting analysis will report the robustness of our neck model. Another possible contribution is to assess the visual quality of the results by conducting a comparison with the state of the art animation techniques like *capturing real subjects* that exploit 3D scanning device to directly capture the skin deformation.

For the work described in Chapter 4, certain balance constraints should be incorporated into the volumetric, elastic energy which currently only preserves the edge lengths of the lattice. By implementing

that, our method would provide the physical aspects of the deformation realism for character animations such as walking, running, and dancing. It would be also interesting to research a way that supports skeletal rigging of our volumetric, elastic model, notably by adding constraints on bone lengths.

In Chapter 5, the space deformation technique is based on radial basis functions and domain decomposition. However, the computations introduced by RBF are still somewhat expensive regarding the real time request. Thus, the main future research should lie in how to obtain better speed ups. On the one hand, we could resort to parallel computing. On the other hand, we may explore other alternatives of the RBF kernel like the thin plate spline. The naive Laplacian smoothing used to deal with the seam artifacts has drawbacks: it might be not able to maintain the smoothness in large rotational deformations. Thus, a search for a better alternative method should be performed.

In Chapter 6, we minimized a surface-based elastic energy to improve the surface quality in deformation. However, the minimization was directly performed in the full space which is expensive, where vertex positions are degrees of freedom. It might be possible to reduce the computational costs by calculating minimum within the skinning subspace as the method described in [Jacobson et al. 2012], where only the skinning transformations are degrees of freedom. Also, it could be possible to support limb bending by providing additional skinning degrees of freedom rather than to tune the proximity of the point handles. The automatic placement of salient point handles is worth further research. It would be elegant if the point handle placements are anatomically-inspired rather than joint-centered. If so, the skinning weights of point handles will leave room for artistic control such as manually painting the weights.

Since elastic models have demonstrated their effectiveness in deformation as described in this thesis, they hold promise for character deformation. In the foreseeable future, there will be increasingly emerging techniques based on elasticity simulation or geometrically-motivated elastic models for interactive, or even real-time character deformation.

Bibliography

[ALG] Alglib: a cross-platform numerical analysis and data processing library. available online at `http://www.alglib.net/`. Accessed: 2015-03-03. (Cited on page 61.)

[Lam] Lamé constants". Weisstein, Eric. Eric Weisstein's World of Science, A Wolfram Web Resource. Retrieved 2015-02-22. (Cited on page 27.)

[20thCenturyFox] 20thCenturyFox. Ice age. available online at `http://www.iceagemovies.com/`. Accessed: 2015-09-10. (Cited on page 3.)

[Allen et al. 2003] Allen, B., Curless, B., & Popović, Z. (2003). The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, *22*(3), 587–594. (Cited on page 35.)

[Anguelov et al. 2005] Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., & Davis, J. (2005). Scape: Shape completion and animation of people. *ACM Trans. Graph.*, *24*(3), 408–416. (Cited on page 35.)

[Arya & Mount 1998] Arya, S. & Mount, D. (1998). Ann: library for approximate nearest neighbor searching. available online at `http://www.cs.umd.edu/~mount/ANN`. Accessed: 2014-11-10. (Cited on page 87.)

[Assassi et al. 2012] Assassi, L., Becker, M., & Magnenat-Thalmann, N. (2012). Dynamic skin deformation based on biomechanical modeling. In *Proceedings of the 25th Annual Conference on Computer Animation and Social Agents*. (Cited on pages 33, 35, and 43.)

[Aubert & Bechmann 1997] Aubert, F. & Bechmann, D. (1997). Volume-preserving space deformation. *Comput. Graph.*, *21*(5), 625–639. (Cited on page 54.)

[Baran & Popović 2007] Baran, I. & Popović, J. (2007). Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, *26*(3). (Cited on pages 7, 90, and 92.)

[Barbič & James 2005] Barbič, J. & James, D. L. (2005). Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM*

Trans. Graph., *24*(3), 982–990. (Cited on pages 11, 12, 14, 54, 56, 65, 67, 73, 74, and 88.)

[Barnhill et al. 1983] Barnhill, R., Dube, R., & Little, F. (1983). Properties of shepard's surfaces. *Journal of Mathematics*, *13*(2). (Cited on pages 60, 73, and 75.)

[Ben-Chen et al. 2009] Ben-Chen, M., Weber, O., & Gotsman, C. (2009). Variational harmonic maps for space deformation. *ACM Trans. Graph.*, *28*(3), 34:1–34:11. (Cited on pages 9, 53, 55, and 73.)

[Botsch & Kobbelt 2005] Botsch, M. & Kobbelt, L. (2005). Real-time shape editing using radial basis functions. *Computer Graphics Forum*, *24*(3), 611–621. (Cited on page 75.)

[Botsch et al. 2006] Botsch, M., Pauly, M., Gross, M., & Kobbelt, L. (2006). Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, (pp. 11–20). (Cited on page 56.)

[Botsch et al. 2007a] Botsch, M., Pauly, M., Wicke, M., & Gross, M. (2007a). Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, *26*(3), 339–347. (Cited on pages 9, 13, 56, 59, 73, and 74.)

[Botsch et al. 2007b] Botsch, M., Pauly, M., Wicke, M., & Gross, M. (2007b). Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, *26*(3), 339–347. (Cited on pages 53, 54, 59, and 85.)

[Botsch & Sorkine 2008] Botsch, M. & Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph.*, *14*(1), 213–230. (Cited on pages 8, 24, and 93.)

[BoxOfficeMojo] BoxOfficeMojo. Animation movies at the box office. available online at `http://www.boxofficemojo.com/genres/chart/?id=animation.htm`. Accessed: 2015-09-10. (Cited on page 3.)

[Bridson & Batty 2010] Bridson, R. & Batty, C. (2010). Computational physics in film. *Science*, *330*(6012), 1756–1757. (Cited on pages 53 and 56.)

[Buss & Fillmore 2001] Buss, S. R. & Fillmore, J. P. (2001). Spherical averages and applications to spherical splines and interpolation. *ACM Trans. Graph.*, *20*(2), 95–126. (Cited on page 55.)

[Capell et al. 2002] Capell, S., Green, S., Curless, B., Duchamp, T., & Popović, Z. (2002). Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.*, *21*(3), 586–593. (Cited on pages 12 and 35.)

[Carr et al. 2001] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., & Evans, T. R. (2001). Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (pp. 67–76). (Cited on page 75.)

[Chen et al. 2012] Chen, X., Saparov, A., Pang, B., & Funkhouser, T. (2012). Schelling points on 3d surface meshes. *ACM Trans. Graph.*, *31*(4), 29:1–29:12. (Cited on page 104.)

[Cook 1994] Cook, R. D. (1994). *Finite Element Modeling for Stress Analysis* (1st ed.). New York, NY, USA: John Wiley & Sons, Inc. (Cited on pages 11 and 39.)

[De Lathauwer et al. 1994] De Lathauwer, L., De Moor, B., Vandewalle, J., & by Higher-Order, B. S. S. (1994). Singular value decomposition. In *Proc. EUSIPCO-94, Edinburgh, Scotland, UK*, volume 1, (pp. 175–178). (Cited on page 8.)

[Desbrun et al. 1999] Desbrun, M., Meyer, M., Schröder, P., & Barr, A. H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, (pp. 317–324). (Cited on page 59.)

[DiLorenzo et al. 2008] DiLorenzo, P. C., Zordan, V. B., & Sanders, B. L. (2008). Laughing out loud: Control for modeling anatomically inspired laughter using audio. *ACM Trans. Graph.*, *27*(5), 125:1–125:8. (Cited on page 12.)

[Dionne & de Lasa 2013] Dionne, O. & de Lasa, M. (2013). Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, (pp. 173–180)., New York, NY, USA. ACM. (Cited on pages 7 and 85.)

[Diziol et al. 2011] Diziol, R., Bender, J., & Bayer, D. (2011). Robust real-time deformation of incompressible surface meshes. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, (pp. 237–246)., New York, NY, USA. ACM. (Cited on page 53.)

[DreamWorks a] DreamWorks. Kung fu panda. available online at http://www.dreamworks.com/kungfupanda/. Accessed: 2015-09-10. (Cited on page 3.)

[DreamWorks b] DreamWorks. Madagascar. available online at http://madagascar.dreamworks.com/. Accessed: 2015-09-10. (Cited on page 3.)

[DreamWorks c] DreamWorks. Shrek. available online at http://www.dreamworksanimation.com/shrek/. Accessed: 2015-09-10. (Cited on page 3.)

[Faugeras & Hebert 1983] Faugeras, O. D. & Hebert, M. (1983). A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, (pp. 996–1002)., San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. (Cited on page 14.)

[Forstmann et al. 2007] Forstmann, S., Ohya, J., Krohn-Grimberghe, A., & McDougall, R. (2007). Deformation styles for spline-based skeletal animation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, (pp. 141–150)., Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. (Cited on pages 6 and 92.)

[Fratarcangeli 2005] Fratarcangeli, M. (2005). Physically based synthesis of animatable face models. In *Proceedings of the Second International Workshop on Virtual Reality and Physical Simulation (VRIPHYS05)*, (pp. 32–39). (Cited on pages 33, 35, 43, 44, and 47.)

[Fung 1977] Fung, Y.-c. (1977). A first course in continuum mechanics. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1977. 351 p., 1*. (Cited on page 11.)

[Gast & Schroeder 2014] Gast, T. F. & Schroeder, C. (2014). Optimization integrator for large time steps. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, Copenhagen, Denmark. Eurographics Association. (Cited on page 12.)

[Georgii et al. 2010] Georgii, J., Lagler, D., Dick, C., & Westermann, R. (2010). Interactive deformations with multigrid skeletal constraints. In *Proceedings of the 7th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS) 2010*, (pp. 39–47). (Cited on page 35.)

[Georgii & Westermann 2008] Georgii, J. & Westermann, R. (2008). Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, (pp. 11–19). (Cited on page 40.)

[Gleicher 1999] Gleicher, M. (1999). Animation from observation: Motion capture and motion editing. *SIGGRAPH Comput. Graph.*, *33*(4), 51–54. (Cited on page 7.)

[Gottschalk et al. 1996] Gottschalk, S., Lin, M. C., & Manocha, D. (1996). Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (pp. 171–180). (Cited on page 45.)

[Guay et al. 2013] Guay, M., Cani, M.-P., & Ronfard, R. (2013). The line of action: An intuitive interface for expressive character posing. *ACM Trans. Graph.*, *32*(6), 205:1–205:8. (Cited on page 6.)

[Guennebaud 2011] Guennebaud, G. (2011). Eigen: a c++ linear algebra library,version 3.0. available online at `http://eigen.tuxfamily.org/`. Accessed: 2014-11-22. (Cited on pages 62, 82, and 99.)

[Hazewinkel 2001] Hazewinkel, M. (2001). *Laplace operator, Encyclopedia of Mathematics*. Springer. (Cited on page 24.)

[Huang et al. 2006] Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.-Y., Teng, S.-H., Bao, H., Guo, B., & Shum, H.-Y. (2006). Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, *25*(3), 1126–1134. (Cited on pages 9, 53, 55, and 73.)

[Irving et al. 2007] Irving, G., Schroeder, C., & Fedkiw, R. (2007). Volume conserving finite element simulations of deformable models. *ACM Trans. Graph.*, *26*(3). (Cited on pages 53 and 68.)

[Irving et al. 2004] Irving, G., Teran, J., & Fedkiw, R. (2004). Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, (pp. 131–140). (Cited on page 12.)

[Irving et al. 2006] Irving, G., Teran, J., & Fedkiw, R. (2006). Tetrahedral and hexahedral invertible finite elements. *Graph. Models*, *68*(2), 66–89. (Cited on page 12.)

[Jacobson et al. 2012] Jacobson, A., Baran, I., Kavan, L., Popović, J., & Sorkine, O. (2012). Fast automatic skinning transformations. *ACM Trans. Graph.*, *31*(4), 77:1–77:10. (Cited on pages 7, 103, and 109.)

[Jacobson et al. 2011] Jacobson, A., Baran, I., Popović, J., & Sorkine, O. (2011). Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, *30*(4), 78:1–78:8. (Cited on pages 7, 59, 90, 92, 93, 98, 100, and 102.)

[Jacobson et al. 2014] Jacobson, A., Deng, Z., Kavan, L., & Lewis, J. P. (2014). Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*. (Cited on page 7.)

[Jacobson et al. 2014] Jacobson, A., Panozzo, D., Glauser, O., Pradalier, C., Hilliges, O., & Sorkine-Hornung, O. (2014). Tangible and modular input device for character articulation. *ACM Trans. Graph.*, *33*(4), 82:1–82:12. (Cited on pages 7 and 90.)

[Jacobson & Sorkine 2011] Jacobson, A. & Sorkine, O. (2011). Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph.*, *30*(6), 165:1–165:8. (Cited on pages 7, 90, 92, 93, 100, and 102.)

[James & Twigg 2005] James, D. L. & Twigg, C. D. (2005). Skinning mesh animations. *ACM Trans. Graph.*, *24*(3), 399–407. (Cited on page 7.)

[Jolliffe 2002] Jolliffe, I. (2002). *Principal component analysis* (second edition ed.). New York, NY, USA: Springer-Verlag. (Cited on page 5.)

[Joshi et al. 2007] Joshi, P., Meyer, M., DeRose, T., Green, B., & Sanocki, T. (2007). Harmonic coordinates for character articulation. *ACM Trans. Graph.*, *26*(3). (Cited on page 6.)

[Ju et al. 2005] Ju, T., Schaefer, S., & Warren, J. (2005). Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, *24*(3), 561–566. (Cited on pages 6, 75, and 92.)

[Kavan et al. 2009] Kavan, L., Collins, S., & O'Sullivan, C. (2009). Automatic linearization of nonlinear skinning. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, (pp. 49–56). (Cited on pages 4, 5, and 92.)

[Kavan et al. 2007] Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2007). Skinning with dual quaternions. In *Proceedings of the 2007*

Symposium on Interactive 3D Graphics and Games, I3D '07, (pp. 39–46). (Cited on pages 33, 35, and 92.)

[Kavan et al. 2008] Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.*, *27*(4), 105:1–105:23. (Cited on pages 5, 19, 53, 55, 75, and 92.)

[Kavan et al. 2010] Kavan, L., Sloan, P.-P., & O'Sullivan, C. (2010). Fast and efficient skinning of animated meshes. *Comput. Graph. Forum*, *29*(2), 327–336. (Cited on pages 7 and 100.)

[Kavan & Sorkine 2012] Kavan, L. & Sorkine, O. (2012). Elasticity-inspired deformers for character articulation. *ACM Trans. Graph.*, *31*(6), 196:1–196:8. (Cited on pages 7 and 8.)

[Kavan & Žára 2005] Kavan, L. & Žára, J. (2005). Spherical blend skinning: A real-time deformation of articulated models. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, (pp. 9–16). (Cited on pages 5 and 55.)

[Kim & James 2011] Kim, T. & James, D. L. (2011). Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, (pp. 63–72). (Cited on pages 12, 76, and 88.)

[Kry et al. 2002] Kry, P. G., James, D. L., & Pai, D. K. (2002). Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, (pp. 153–159)., New York, NY, USA. ACM. (Cited on page 5.)

[Labelle & Shewchuk 2007] Labelle, F. & Shewchuk, J. R. (2007). Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, *26*(3), 57.1–57.10. (Cited on page 38.)

[Landreneau & Schaefer 2010] Landreneau, E. & Schaefer, S. (2010). Poisson-based weight reduction of animated meshes. *Computer Graphics Forum*, *29*(6), 1945–1954. (Cited on pages 7, 13, and 104.)

[Le & Deng 2012] Le, B. H. & Deng, Z. (2012). Smooth skinning decomposition with rigid bones. *ACM Trans. Graph.*, *31*(6), 199:1–199:10. (Cited on page 8.)

[Le & Deng 2013] Le, B. H. & Deng, Z. (2013). Two-layer sparse compression of dense-weight blend skinning. *ACM Trans. Graph.*, *32*(4), 124:1–124:10. (Cited on pages 4, 7, and 104.)

[Le & Deng 2014] Le, B. H. & Deng, Z. (2014). Robust and accurate skeletal rigging from mesh sequences. *ACM Trans. Graph.*, *33*(4), 84:1–84:10. (Cited on page 8.)

[Lee 2008] Lee, S. H. (2008). *Biomechanical Modeling and Control of the Human Body for Computer Animation*. PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA. (Cited on page 36.)

[Lee et al. 2009] Lee, S.-H., Sifakis, E., & Terzopoulos, D. (2009). Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph.*, *28*(4), 99:1–99:17. (Cited on pages 12, 33, 35, 36, 38, 40, 42, and 43.)

[Lee & Terzopoulos 2006] Lee, S.-H. & Terzopoulos, D. (2006). Heads up!: Biomechanical modeling and neuromuscular control of the neck. *ACM Trans. Graph.*, *25*(3), 1188–1198. (Cited on pages 33, 35, 36, and 37.)

[Lefebvre et al. 2013] Lefebvre, S., Hornus, S., & Lasram, A. (2013). Habuffer: Coherent hashing for single-pass a-buffer. Research Report RR-8282, INRIA. (Cited on page 56.)

[Levi & Levin 2014] Levi, Z. & Levin, D. (2014). Shape deformation via interior rbf. *Visualization and Computer Graphics, IEEE Transactions on*, *20*(7), 1062–1075. (Cited on pages 74 and 76.)

[Lewis et al. 2000] Lewis, J. P., Cordner, M., & Fong, N. (2000). Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, (pp. 165–172). (Cited on pages 5, 34, 35, 55, and 75.)

[Lipman et al. 2007] Lipman, Y., Cohen-Or, D., Gal, R., & Levin, D. (2007). Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph.*, *26*(1). (Cited on pages 9 and 55.)

[Lipman et al. 2008] Lipman, Y., Levin, D., & Cohen-Or, D. (2008). Green coordinates. *ACM Trans. Graph.*, *27*(3), 78:1–78:10. (Cited on pages 6 and 13.)

[Liu & Gorman 1995] Liu, M. & Gorman, D. (1995). Formulation of rayleigh damping and its extensions. *Computers & structures*, *57*(2), 277–285. (Cited on page 28.)

[Liu et al. 2013] Liu, T., Bargteil, A. W., O'Brien, J. F., & Kavan, L. (2013). Fast simulation of mass-spring systems. *ACM Trans. Graph.*, *32*(6), 214:1–214:7. (Cited on page 11.)

[Luo et al. 2013] Luo, Z., Pronost, N., & Egges, A. (2013). Physics-based human neck simulation. In *Proceedings of Virtual Reality Interaction and Physical Simulation*, VRIPHYS'13, (pp. 51–60). (Cited on page 12.)

[Luo et al. 2015] Luo, Z., Veltkamp, R., & Egges, A. (2015). Space deformation for character deformation using multi-domain smooth embedding. In *Proceedings of Computer Animation and Social Agents*, CASA'15, (pp. 51–54). (Cited on page 13.)

[Maas et al. 2012] Maas, S. A., Ellis, B. J., Ateshian, G. A., & Weiss, J. A. (2012). FEBio: finite elements for biomechanics. *Journal of biomechanical engineering*, *134*(1). (Cited on pages 27, 41, and 42.)

[Merry et al. 2006] Merry, B., Marais, P., & Gain, J. (2006). Animation space: A truly linear framework for character animation. *ACM Trans. Graph.*, *25*(4), 1400–1423. (Cited on pages 4 and 92.)

[Meyer et al. 2003] Meyer, M., Desbrun, M., Schröder, P., & Barr, A. H. (2003). Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III* (pp. 35–57). Springer. (Cited on pages 82 and 99.)

[Mohr & Gleicher 2003] Mohr, A. & Gleicher, M. (2003). Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, *22*(3), 562–568. (Cited on pages 4 and 35.)

[Molino et al. 2003] Molino, N., Bridson, R., Teran, J., & Fedkiw, R. (2003). A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *In 12th Int. Meshing Roundtable*, (pp. 103–114). (Cited on page 38.)

[Müller & Chentanez 2011] Müller, M. & Chentanez, N. (2011). Solid simulation with oriented particles. *ACM Trans. Graph.*, *30*(4), 92:1–92:10. (Cited on pages 75 and 86.)

[Müller et al. 2002] Müller, M., Dorsey, J., McMillan, L., Jagnow, R., & Cutler, B. (2002). Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, (pp. 49–54). (Cited on pages 11, 35, and 41.)

[Müller & Gross 2004] Müller, M. & Gross, M. (2004). Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, (pp. 239–246). (Cited on pages 11, 40, and 41.)

[Müller et al. 2005] Müller, M., Heidelberger, B., Teschner, M., & Gross, M. (2005). Meshless deformations based on shape matching. *ACM Trans. Graph.*, *24*(3), 471–478. (Cited on pages 67, 86, and 87.)

[Nealen et al. 2006] Nealen, A., Mller, M., Keiser, R., Boxerman, E., & Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, *25*(4), 809–836. (Cited on page 29.)

[Neumann et al. 2013] Neumann, T., Varanasi, K., Hasler, N., Wacker, M., Magnor, M., & Theobalt, C. (2013). Capture and statistical modeling of arm-muscle deformations. *Computer Graphics Forum*, *32*(2), 285–294. (Cited on pages 34 and 35.)

[Noh et al. 2000] Noh, J.-y., Fidaleo, D., & Neumann, U. (2000). Animated deformations with radial basis functions. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '00, (pp. 166–174). (Cited on page 75.)

[Öztireli et al. 2013] Öztireli, A. C., Baran, I., Popa, T., Dalstein, B., Sumner, R. W., & Gross, M. (2013). Differential blending for expressive sketch-based posing. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, (pp. 155–164). (Cited on pages 5 and 6.)

[Park & Hodgins 2006] Park, S. I. & Hodgins, J. K. (2006). Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, *25*(3), 881–889. (Cited on pages 75, 76, and 78.)

[Park & Hodgins 2008] Park, S. I. & Hodgins, J. K. (2008). Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph.*, *27*(3), 96:1–96:6. (Cited on pages 34 and 35.)

[Pauly et al. 2003] Pauly, M., Keiser, R., Kobbelt, L. P., & Gross, M. (2003). Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, *22*(3), 641–650. (Cited on page 75.)

[Press et al. 1992a] Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1992a). *Cholesky Decomposition, Numerical Recipes in FORTRAN: The Art of Scientific Computing* (second edition ed.). Cambridge, England: Cambridge University Press. (Cited on page 26.)

[Press et al. 1992b] Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1992b). *LU decomposition and its applications, Numerical Recipes in FORTRAN: The Art of Scientific Computing* (second edition ed.). Cambridge, England: Cambridge University Press. (Cited on page 23.)

[Rhee et al. 2006] Rhee, T., Lewis, J. P., & Neumann, U. (2006). Real-time weighted pose-space deformation on the gpu. *Computer Graphics Forum*, *25*(3), 439–448. (Cited on page 35.)

[Rivers & James 2007a] Rivers, A. R. & James, D. L. (2007a). Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.*, *26*(3). (Cited on pages 14, 73, 74, and 85.)

[Rivers & James 2007b] Rivers, A. R. & James, D. L. (2007b). Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.*, *26*(3). (Cited on pages 54, 56, 65, and 67.)

[Rohmer et al. 2009] Rohmer, D., Hahmann, S., & Cani, M.-P. (2009). Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, (pp. 83–92). (Cited on pages 10 and 54.)

[Rychlewski 1984] Rychlewski, J. (1984). On hooke's law. *Journal of Applied Mathematics and Mechanics*, *48*(3), 303–314. (Cited on page 28.)

[Schenk & Gärtner 2004] Schenk, O. & Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, *20*(3), 475–487. (Cited on page 46.)

[Sederberg & Parry 1986] Sederberg, T. W. & Parry, S. R. (1986). Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, (pp. 151–160). (Cited on pages 13, 75, and 92.)

[Shapira et al. 2008] Shapira, L., Shamir, A., & Cohen-Or, D. (2008). Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, *24*(4), 249–259. (Cited on pages 7, 90, and 98.)

[Shi et al. 2007] Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H., & Guo, B. (2007). Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, *26*(3). (Cited on page 7.)

[Shoemake 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, (pp. 245–254). (Cited on page 55.)

[Si 2006] Si, H. (2006). Tetgen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany. (Cited on pages 38 and 39.)

[Sieger et al. 2012] Sieger, D., Menzel, S., & Botsch, M. (2012). High quality mesh morphing using triharmonic radial basis functions. In *Proceedings of the 21st International Meshing Roundtable*, IMR '12, (pp. 1–15). (Cited on page 9.)

[Sifakis et al. 2007] Sifakis, E., Der, K. G., & Fedkiw, R. (2007). Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, (pp. 73–80). (Cited on page 38.)

[Sifakis et al. 2005] Sifakis, E., Neverov, I., & Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.*, *24*(3), 417–425. (Cited on pages 33 and 34.)

[Sorkine & Alexa 2007] Sorkine, O. & Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, (pp. 109–116). (Cited on pages 8, 26, 58, 59, 93, 94, 99, and 103.)

[Sorkine et al. 2004] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., & Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, (pp. 175–184). (Cited on page 94.)

[Stavness et al. 2011] Stavness, I., Lloyd, J. E., Payan, Y., & Fels, S. (2011). Coupled hardsoft tissue simulation with contact and constraints applied to jawtonguehyoid dynamics. *International Journal*

for Numerical Methods in Biomedical Engineering, *27*(3), 367–390. (Cited on pages 35, 36, and 46.)

[Su et al. 2013] Su, J., Sheth, R., & Fedkiw, R. (2013). Energy conservation for the simulation of deformable bodies. *IEEE Transactions on Visualization and Computer Graphics*, *19*(2), 189–200. (Cited on page 12.)

[Tan et al. 2012] Tan, J., Turk, G., & Liu, C. K. (2012). Soft body locomotion. *ACM Trans. Graph.*, *31*(4), 26:1–26:11. (Cited on page 34.)

[Terzopoulos & Waters 1990] Terzopoulos, D. & Waters, K. (1990). Physically-based facial modelling, analysis, and animation. *The journal of visualization and computer animation*, *1*(2), 73–80. (Cited on page 11.)

[UHM] UHM. Ultimate human model data set. `http://www.cgcharacter.com/ultimatehuman.html`. Accessed: 2013-02-12. (Cited on page 34.)

[Vaillant et al. 2013] Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.-P., Rohmer, D., Wyvill, B., Gourmel, O., & Paulin, M. (2013). Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph.*, *32*(4), 125:1–125:12. (Cited on page 76.)

[Vaillant et al. 2014] Vaillant, R., Guennebaud, G., Barthe, L., Wyvill, B., & Cani, M.-P. (2014). Robust iso-surface tracking for interactive character skinning. *ACM Trans. Graph.*, *33*(6), 189:1–189:11. (Cited on page 76.)

[van Tol & Egges 2009] van Tol, W. & Egges, A. (2009). Real-time crying simulation. In *Intelligent Virtual Agents*, (pp. 215–228). Springer. (Cited on page 33.)

[Vasavada et al. 1998] Vasavada, A. N., Li, S., & Delp, S. L. (1998). Influence of muscle morphometry and moment arms on the moment-generating capacity of human neck muscles. *Spine*, *23*(4), 412–422. (Cited on page 33.)

[von Funck et al. 2008] von Funck, W., Theisel, H., & Seidel, H.-P. (2008). Volume-preserving mesh skinning. In *Vision, Modeling and Visualization*, (pp. 409–414). (Cited on pages 10 and 54.)

[Wang et al. 2007] Wang, R. Y., Pulli, K., & Popović, J. (2007). Real-time enveloping with rotational regression. *ACM Trans. Graph.*, *26*(3). (Cited on page 92.)

[Wang & Phillips 2002] Wang, X. C. & Phillips, C. (2002). Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, (pp. 129–138). (Cited on pages 4 and 92.)

[White et al. 2007] White, K. B., Cline, D., & Egbert, P. K. (2007). Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, RT '07, (pp. 129–132). (Cited on page 79.)

[Xin & Wang 2009] Xin, S.-Q. & Wang, G.-J. (2009). Improving chen and han's algorithm on the discrete geodesic problem. *ACM Trans. Graph.*, *28*(4), 104:1–104:8. (Cited on page 80.)

[Yang et al. 2006] Yang, X., Somasekharan, A., & Zhang, J. J. (2006). Curve skeleton skinning for human and creature characters: Research articles. *Comput. Animat. Virtual Worlds*, *17*(3-4), 281–292. (Cited on pages 6 and 92.)

[Zhang et al. 2014] Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K., & Liu, L. (2014). Local barycentric coordinates. *ACM Trans. Graph.*, *33*(6), 188:1–188:12. (Cited on page 13.)

[Zhang et al. 2004] Zhang, Y., Prakash, E. C., & Sung, E. (2004). A new physical model with multilayer architecture for facial expression animation using dynamic adaptive mesh. *IEEE Transactions on Visualization and Computer Graphics*, *10*(3), 339–352. (Cited on pages 33, 35, 43, 44, and 47.)

[Zhou et al. 2005] Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., & Shum, H.-Y. (2005). Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, *24*(3), 496–503. (Cited on page 56.)

[Zordan et al. 2004] Zordan, V. B., Celly, B., Chiu, B., & DiLorenzo, P. C. (2004). Breathe easy: model and control of simulated respiration for animation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, (pp. 29–37). Eurographics Association. (Cited on pages 12, 33, 35, and 43.)