NANYANG TECHNOLOGICAL UNIVERSITY

Simulating Deformable Models with Anisotropic Materials

CAI JIANPING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2016

Simulating Deformable Models with Anisotropic Materials

CAI JIANPING

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

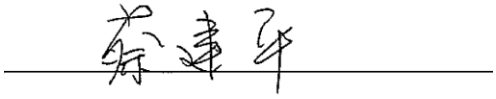
2016

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

23 May 2016

Date



Cai Jianping

Abstract

This research work is on dynamics simulation of deformable objects. We focus on the simulation of anisotropic materials, which is less exploited in existing research. To do this, it is essential to improve the physical realism of simulation, since many real-world objects have complex mechanical rather than isotropic properties. Both physically-based and geometrically-based approaches are studied, and contributions are made in modeling and control of anisotropic dynamics deformations.

First, we studied transversely isotropic materials for the simulation of deformable objects with fibrous structures. In existing work, direction-dependent behaviors of transversely isotropic materials can only be achieved with an additional energy function which incorporates the material preferred direction. Such an additional energy term increases the computational complexity. We propose a *fiber-field incorporated corotational finite element model* (CLFEM) that works directly with a constitutive model of transversely isotropic material. A smooth *fiber-field* is used to establish the local frames for each element. The orientation information of each element is incorporated into the CLFEM model by adding local transformations onto each element of the stiffness matrix. With pre-computation, it adds no additional computational cost on the existing model during dynamics simulation.

We further introduce deformation simulation for orthotropic materials. Technical innovations are made in several aspects: An orthotropic deformation controlling *frame-field* is conceptualized and a frame construction tool is developed for users to define the desired material properties. A quaternion Laplacian smoothing algorithm is designed for propagating the user-defined sparsely distributed frames into the entire object. The orthotropic frame-field is coupled with the CLFEM model to complete an orthotropic deformable model.

Finally, we present an integrated real-time system for animation of skeletal characters with anisotropic tissues. Existing geometrically-based skinning techniques suffer from obvious volume distortion artifact, and they cannot produce secondary dynamic motions, such as *jiggling* effects. *Physically-based skinning* with FEM models has high computational cost that restricts its practical applications. To solve these problems, we developed a novel strain-based PBD framework for skeletal animation. It bridges the gap between geometric models and physically-based models, and achieves both efficient and physically-plausible performance. Natural secondary motion of soft tissues is produced. Anisotropic deformations are made possible with separately defined stretch and shear properties of the material, using the user-designed *frame-field*. Owing to the efficiency and stability of our proposed layered constraint solving scheme, we can achieve real-time performance, and the system is robust with large deformations and degenerate cases.

Limitations of our system and directions of future work, such as self-collision constraint and two-way coupling of rigid and soft bodies for locomotion control, are also discussed.

Acknowledgements

I would like to show my sincere gratitude to my supervisors Professor *Lin Feng* and Professor *Seah Hock Soon*, without whom this work would be impossible; and many thanks go to Professor *Qian Kemao*, Professor *Lee Yong Tsui* and Professor *Wu Zhongke*. I benefit a lot from their inspirational suggestions and helpful discussions during my PhD study.

I am very thankful to my labmates, *Movania Muhammad Mobeen*, *Wei Ming Chiew*, *Xu Xiang*, *Liu Hui*, *Kuleesha Yadav*, and *Hao Shuji* for their friendship, help and encouragement. Special thanks go to Dr. *Mobeen*, who shared his research work selflessly and offered a lot of helpful advice on my research topic.

I am very grateful to all my friends back in China and here in Singapore. With them I do not feel alone, and have more courage to face the difficulties along the way of research and life. I would like to thank the music that relaxes and inspires me every day, and I am grateful to these gifted musicians who have created them.

Finally, I would like to express my gratitude and love to my family members. Though being far away from my parents, I can always feel their love and support, and wish I could spend more time with them in the future. I am very fortunate to have my kind, ingenuous, optimistic, and lovely wife with me all these years, who shares joys and sorrows with me and is supportive all the time.

Author's Publications

[1] **Jianping Cai**, Feng Lin*, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Modeling and Dynamics Simulation for Deformable Objects of Orthotropic Materials.* The Visual Computer, 1-12, 2016

[2] **Jianping Cai**, Feng Lin*, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Keynotes Lecture*: *Simulation and Visualization of Deformation with Anisotropic Materials*. The 19th International Conference on Information Visualization (IV'15), Barcelona, Spain, 21-24 July, 2015

[3] **Jianping Cai**, Feng Lin, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Dynamics Simulation of User-specified Orthotropic Materials*. The 32nd Computer Graphics International (CGI'15), Strasbourg, France, 24-26 June, 2015

[4] **Jianping Cai**, Feng Lin, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Incorporating Fiber Controls into FEM Model for Transversely Isotropic Materials*. The 22nd Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2014) (PG'14), Seoul, Korea, 8-10 October, 2014

[5] **Jianping Cai**, Feng Lin, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Fiber Orientation Guided Deformable Models for Representation and FEM Implementation of Anisotropic Elastic Objects*. International Workshop on Advanced Image Technology (IWAIT'13), Nagoya, Japan, 7-9 January, 2013

Revised version submitted

[6] **Jianping Cai**, Feng Lin, Yong Tsui Lee, Kemao Qian, Hock Soon Seah. *Skeleton Driven Animation of Deformable Objects with Anisotropic Materials*. Graphical Models, Elsevier (**submitted**)

Contents

List of Figures

List of Tables

Notational Conventions

Most symbols in this work are denoted according to the notational conventions as follows,

Notations:

Scalar: a (*Italic* lowercase)

Vector: $\boldsymbol{a}$ (*Italic* and bold lowercase)

Matrix and Tensor: A (Italic and uppercase)

Space of n dimensional real numbers: $\mathbb{R}^n$

Space of $m \times n$ real matrices: $\mathbb{R}^{m \times n}$

Identity matrix: I

Operations:

Dot product: "·"such as $\boldsymbol{a} \cdot \boldsymbol{b}$;

Cross product of vector: "×", such as $\boldsymbol{a} \times \boldsymbol{b}$;

Tensor product of two vectors: "⊗", such as $\boldsymbol{a} \otimes \boldsymbol{b}$

Double product or double contraction of two matrices: ":", such as $A : B$

Chapter 1. Introduction

Deformable object modeling has been an active research topic in computer graphics [Gibson '97, Nealen '06, Müller '08, Sifakis '12]. Since it was introduced to the graphics community in the late 1980s [Terzopoulos '87], many works with various modelling approaches have been published.

In computer graphics, researches focus on visually realistic physical simulations of the motion and properties of deformable objects. Different from rigid body simulation, the shape of deformable objects can change due to external impacts with an infinite number of degrees-of-freedom. The scope of deformable objects is large, including objects of different dimensions, such as 1D ropes and hairs, 2D shells and cloths, and 3D solids such as animal organs.

The ability to model and manipulate deformable objects is essential to many applications, such as geometry modeling and processing in computer aided design, animations, computer games, virtual reality, visual effects industries. Although the computational power of modern computers is increasing, there are still a lot of challenges to achieve deformable models with more physical realism, more stability and controllability. We are quite interested in this topic, and make an effect to push the technologies to further development.

1.1 Technical Challenges

Different from the research of deformation simulation in mechanical science and engineering, where computational *accuracy* is vital for rigorous scientific analysis and validation, computer graphics applications often have their challenges as follows:

- ***Stability*** – Large deformations caused by strong external impacts are common in graphics applications, where degenerate and inverted deformations would occur, causing numerical instability or even failure of simulation. Relatively large time step size, which is desirable for simulation efficiency, can also lead to convergence problem. Therefore, robust algorithms with guaranteed convergence are required for a simulation to be used in a practical application.
- ***Efficiency*** – In contrast to dynamics simulation of rigid bodies that only involves a few degrees-of-freedom (DOF), a deformable object usually contains a large number of DOF, which leads to the need to solve large systems of equations. Both computation and memory intensive solvers are commonly required. However, there is usually a strictly limited budget of computing resources for the simulation phase in an application, especially in real-time or interactive applications such as games or virtual surgery applications. Therefore, an efficient solver with fast convergence rate plays a critical role. Computation reduction and approximation strategies are also desirable for performance improvement.
- ***Controllability*** – Deformation control is an important concern for many graphics applications. For example, in animation design, instead of passively accepting the results automatically generated by a simulation, artists would often require intuitive and stable controls over material properties and animation behaviors in order to realize artistic or creative motions; in other words, to have *directable* deformations. Deformation behaviors can be controlled by stably adjusting material parameters, or by adding artificial forces of least interference with physical authenticity, etc. This kind of controllability should be intuitive and easy for user interactions.
- ***Accuracy*** also plays an *important* role here. However, with the limitation of computing resources, there are always trade-offs, and a proper solution is to seek

good compromise among these criteria. For example, in computer games, stable and fast responses are of greater importance than physical accuracy, thus certain accuracy has to be sacrificed for efficiency and stability; in animation design, controllability is desirable for artistic designs with certain loss of accuracy; in graphics industries such as film production, efficient algorithms would save expenses in terms of time of money. In these cases, only visual plausibility in terms of accuracy is required if physical accuracy is expensive to achieve.

1.2 Objectives and Work

We mainly work on physically-based deformable models. The mathematical models are continuum-based rather than geometrically motivated. In contrast to the existing research that mainly works on isotropic materials, we focus on complex anisotropic materials that commonly exist in the real world.

Anisotropic materials are less studied in the field of computer graphics, and most of the publications focus on modeling of isotropic materials. Due to the computational complexity of anisotropic deformations, anisotropic materials are rarely used in practical applications; moreover, there is lack of ways to control anisotropic mechanical properties.

We investigate formulation for deformable models of anisotropic materials, and design interactive tools for intuitive control of anisotropic deformation behaviors. In contrast with the previous work, our proposed methods add no computational cost during simulation, with the help of precomputation.

Furthermore, based on the observation that many real-world soft objects, such as the skins and soft tissues of animal and human characters, are linked to rigid skeletons,

we also consider augmenting skeleton control with deformable objects, and finally develop a skeletal animation system that encompasses all these techniques.

With the computational efficiency and stability, our methods have a great potential to be used in many real-time applications, such as virtual surgery and animations, with improved physical realism.

1.3 Organization of the Thesis

In this first chapter, we have given a brief introduction of the research problems, and have described the research background, challenges and our objectives. The rest of the thesis is organized as follows:

Chapter 2 gives an in-depth literature review of research on simulation of deformable models. We discuss and analyze various approaches including geometrical models, physically-based models, hybrid models, and deformation control methods. Our focus is on physically-based approaches, which build a foundation for our work.

Chapter 3 provides a brief summary and analysis of fundamental theories on continuum-based deformable models, including elasticity theory, finite element discretization, dynamics equations of motion and numerical integration schemes.

Chapter 4 presents our work on dynamics simulation of transversely isotropic deformable objects. We propose a fiber-field incorporated FEM model for deformation control, which is our first effort on simulation of anisotropic materials.

Chapter 5 further investigates orthotropic materials. We develop a frame-field augmented FEM model for dynamics simulation of orthotropic deformable objects. A Laplacian smoothing method is developed to generate a frame-field for an orthotropic model. An interactive approach for deformation control is presented.

Chapter 6 extends our research to skeletal character animation. We combine continuum-based models with position-based dynamics, and incorporate skeleton control into deformable character animation. Finally, we design a skeletal character animation system that can generate physically-plausible dynamic motions of soft tissues with stable and efficient performance.

Chapter 7 concludes our work and discusses potential research possibilities in the future.

Chapter 2. Literature Review

Deformable models have been researched for nearly three decades in computer graphics since the late 1980s. In this chapter, we review some of the influential works that have fueled the development of deformable objects simulation in graphics community. Various deformable models have been proposed, focusing on *efficiency*, *stability*, *controllability*, and *accuracy* of simulation. We discuss these approaches from the following aspects:

- Firstly, we discuss some ***geometrically-based deformable models***, which formulate a mathematical model of deformable objects from a geometric perspective rather than rigorous physics. Some deformation approaches in *geometry processing* are introduced. A group of *position-based methods* for dynamics simulation are discussed. Although being geometrically motivated, these models are related to physical models to some extent.
- Secondly, a simple physical model of ***mass spring system*** and its extension to particle systems are reviewed;
- Thirdly, we give a thorough review of ***physically-based deformable models*** based on continuum mechanics, from which we initiated our research. Continuum-based models in combination with finite element discretization can produce physically-realistic results; however, in graphics applications, they encounter challenges related to high computational complexity, numerical instability and controllability. Various approaches have been proposed to solve these problems, and we analyze them from different perspectives.

- Fourthly, we discuss some ***hybrid models*** developed in recent years, which attempt to reconcile geometrical methods with physical models and bridge the gap between these two groups, leading to new research potentials.
- Finally, we discuss some ***deformation control*** methods, which provide users with control over material properties or complex deformation behaviors.

2.1 Geometrically-Based Methods

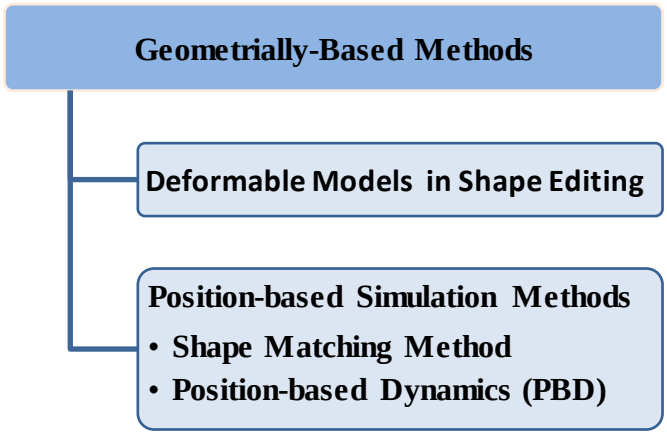
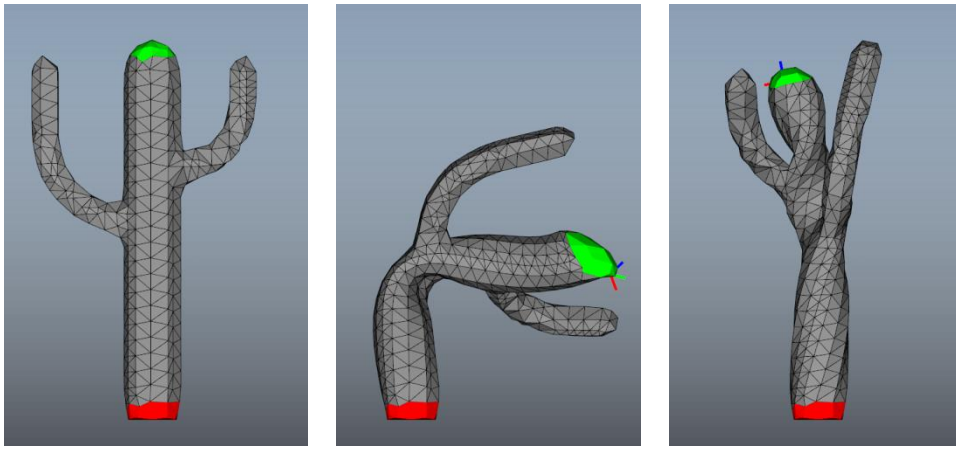


Fig. 2.1 Geometrically-based deformable models

2.1.1 Deformable Models in Shape Editing

Elastic models are commonly used in *geometry processing*, such as *morphing*, *shape space interpolation*, and *shape editing*. Note that here 'elastic models' refer to *elasticity-inspired* models, which utilize geometric quantities to define elasticity but not according to rigorous physics (i.e., continuum mechanics). Here we only refer to some deformation methods in *shape editing* (i.e., *handle-based shape deformations*) that are related to our research. Other methods such as *free form deformation* are not discussed here, and readers can refer to [Botsch '10, Crane '13, Panozzo '15] for more details.

In some shape modeling process, users set some vertices of a mesh fixed in position (as *anchor points*) and select a region-of-interest (ROI) for manipulation (as *handles*). The orientation and position of the ROI are interactively manipulated, and then transformations (such as translations and rotations) of the ROI are propagated smoothly to the rest of the shape by deforming the unconstrained part. A basic requirement is that local details of the shape should be preserved during deformations. As shown in Fig. 2.2, the green region represents the handle for manipulation, and the red region is fixed in position as anchor points. Fig. 2.2 (a) shows the original shape, and Fig. 2.2 (b) and Fig. 2.2 (c) show the bending and twisting deformations respectively.



(a) (b) (c)

Fig. 2.2 As-rigid-as-possible surface editing.
(a) The initial shape; (b) Bending deformation; (c) Twisting deformation.

A brief review of a prominent work by *Sorkine et al.* [Sorkine '07], called *as-rigid-as-possible* (ARAP) surface deformation, is described here: given a triangle mesh $\boldsymbol{S}$ consisting of n vertices, it is decomposed into overlapping cells (e.g., the one-ring-neighboring triangles $N(i)$ of each vertex i). The key idea is that the transformation of each cell should be kept as-rigid-as-possible. For a cell C_i

corresponding to a vertex i and its deformed counterpart C_i' in the deformed mesh $\boldsymbol{S}'$, a deformation energy (called *local rigidity* energy) is defined by summing up the *deviations from rigidity* of all the cells:

$$E(\boldsymbol{S}') = \sum_{i=1}^{n} \omega_i \sum_{j \in N(i)} \omega_{ij} \left\| \left(\boldsymbol{p}_i' - \boldsymbol{p}_j'\right) - R_i(\boldsymbol{p}_i - \boldsymbol{p}_j) \right\|^2 , \tag{2.1}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_i'$ are the vertex positions, and ω_i and ω_{ij} are per-cell and per-edge weights respectively. The matrix $R_i = R_i(\boldsymbol{S}') \in \mathbb{R}^{3\times3}$ is the rotation of a cell from $\boldsymbol{S}$ to $\boldsymbol{S}'$, which depends on the deformed shape $\boldsymbol{S}'$. In order to preserve local details, this nonlinear *rigidity energy* needs to be minimized. *Sorkine et al.* proposed an *alternating optimization strategy*, which is an iterative process consisting of a local solver and a global solver: in the local step, the rotation R_i of each cell is computed by a *singular value decomposition* of a covariance matrix; in the global step, a global deformation state is obtained by solving a convex quadratic minimization problem.

This alternating approach decomposes a nonlinear minimization problem into two simple sub-problems, which guarantees convergence. Another advantage is its computational efficiency, due to the facts that: 1) the local solving step can be done in parallel (i.e., parallel computing for all the cells), and 2) the system matrix of the global solver is constant and can be pre-factored.

A similar idea of energy formulation can be found in an early work in Laplacian surface editing [Sorkine '04]. Both of the methods formulate a *deformation energy* based on changes of local details, which is essentially in terms of *differential representations* (here, *Laplacian coordinates* are used to encode local details). This idea is further investigated by many works. For example, a skeleton augmented ARAP method was presented in [Zhang '11] for better volume preservation of a surface mesh;

Zhou et al. [Zhou '05] extended it to volumetric mesh editing using *volumetric graph Laplacian*.

The idea of using an *alternating local/global scheme* to solve a complex nonlinear optimization problem is very attractive, due to its efficiency and stability. It was further exploited in physically-based models such as the fast mass-spring system in [Liu '13] and the projective dynamics approach in [Bouaziz '14].

The idea of finding a *linear transformation* between an original shape and a deformed counterpart inspired the development of *shape matching method* [Müller '05] (refer to **Section 2.1.2.1**). The corotational FEM model [Müller '02] [Müller '04] (refer to **Section 2.3.1.1**) also shares a similar spirit in terms of finding local rotational transformations.

This kind of geometrically-based deformation models (*Laplacian deformation methods*) was further analyzed by *Chao et al.* [Chao '10], which proved a connection between these *physics-like* geometrical models and physical models in solid mechanics.

Our work is related to the Laplacian deformation method [Sorkine '04], not from the perspective of the formulation of elastic deformation energy, but from the way of propagating transformations smoothly.

2.1.2 Position-Based Simulation Methods

Formulations of classical continuum-based dynamics simulation are *force-based* (refer to **Section 2.2 and 2.3**), where internal forces caused by deformation are computed and positions are evolved through numerical integration of accelerations and velocities. In contrast, position-based simulation methods are purely geometrically motivated. By replacing deformation energies with geometric constraints, they modify positions directly and omit the velocity and acceleration layer.

We firstly introduce *shape matching methods*, and then move to *position-based dynamics* (PBD) that becomes popular recently in real-time applications.

2.1.2.1 Shape Matching Method

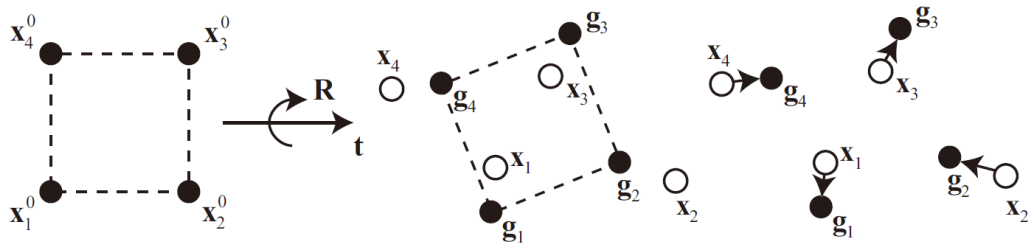


Fig. 2.3 Shape matching method [Müller '05]. (Left) is the initial configuration $\boldsymbol{x}_i^0$; (Middle) is a deformed configuration $\boldsymbol{x}_i$, and $\boldsymbol{g}_i$ is the computed *goal* positions; (Right) during deformation, $\boldsymbol{x}_i$ is pulled towards its corresponding $\boldsymbol{g}_i$.

The shape matching method proposed by *Müller et al.* [Müller '05] is a geometrically-motivated approach [Jakobsen '01]. Its key idea is intuitive and simple as shown in Fig. 2.3: given two sets of particles with $\boldsymbol{x}_i^0$ as an initial configuration and $\boldsymbol{x}_i$ as a deformed configuration with mass m_i, a *goal position* $\boldsymbol{g}_i$ is firstly obtained by a rigid transformation (a rotation R and a translation $\boldsymbol{t}$) that transforms $\boldsymbol{x}_i^0$ to $\boldsymbol{x}_i$. The particles of the deformed set $\boldsymbol{x}_i$ are pulled towards the goal positions in order to restore its original shape at each time step. Therefore, this method does not require particle connection information, i.e., it is a meshless method.

An extended *Lattice Shape Matching* (LSM) [Rivers '07] method uses a regular lattices discretization of a deformable object in order to support large deformations. *Adaptive lattices* are considered in [Steinemann '08], which can support dynamic adaptive selection of level of details and handle topological change. The shape matching method is also employed in a real-time *example-based* dynamics simulation [Koyama '12].

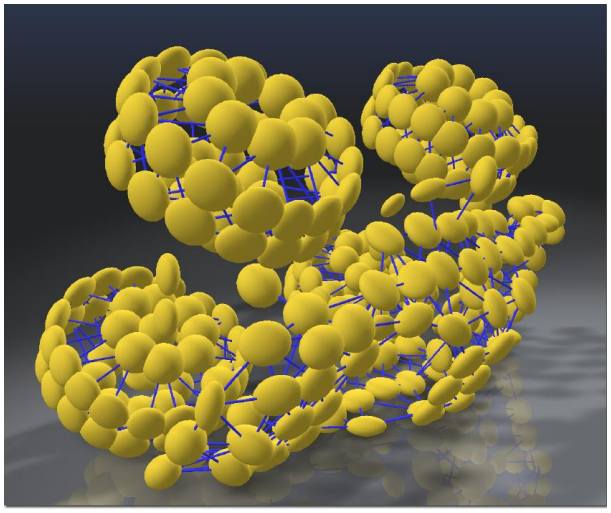


Fig. 2.4 Oriented particles representation [Müller '11]

Müller et al. [Müller '11] later extended the shape matching method to *oriented particles* (as shown in Fig. 2.4). By adding *orientation* information, it solves the ill-conditioned matching problem caused by co-linear or co-planar particles.

The shape matching can be seen as a type of constraint projection, which can be directly integrated into the *position-based dynamics* framework discussed below.

2.1.2.2 Position-Based Dynamics

In recent years, position-based dynamics (PBD) becomes popular for simulation of deformable objects (refer to the survey paper [Bender '15]), and has been implemented in many high-end products, such as *PhysX*, *Havok Cloth*, *Maya nCloth*, and *Bullet Physics Engine* [Coumans '10] .

A classical PBD framework [Müller '07] performs a simulation loop in three steps:

1) The next-time-step positions are firstly predicted by a simple symplectic integration, which takes into account only external forces. Thus there is no need of computation of internal forces and stiffness matrix as in continuum-based models;

2) Then, the *predicted positions* are corrected with respect to some geometric constraints, such as area constraints, bending constraints [Micky Kelager '10] and volume constraints [Diziol '11], which are solved by a Gauss-Seidel-type method. This procedure is called *position projection* (or, *constraint projection*).
3) Finally, the velocities are updated, which might be damped.

2.1.2.3 Exploiting for the PDB Framework

Though a simple explicit time integration scheme is used, the PBD method can avoid *overshooting problem*, because the predicted positions are always projected towards well-defined shapes. They can easily handle collision constraints. Due to their simplicity, efficiency and stability, they are particularly useful in interactive environments such as computer games, where only *visual plausibility* is required.

One problem with these methods is that material behaviors not only depend on constraint parameters, but also depend on the time-step size and the number of iterations in the constraint solver. The Gauss-Seidel-type solver converges faster than the Jacobi-type solver (e.g., [Liu '13]), but it can hardly be used for parallel computing. We refer the reader to the papers [Müller '08, Bender '14b, Bender '15] for more details about the PBD method.

In our work, we have exploited the PBD framework to develop a skeletal animation system, which achieves real-time and stable performance. Continuum-based constraint is used in our system, instead of geometric constraints used here.

2.2 Mass Spring System and Particles System

Among physically-based deformable models, a *mass-spring model* is a rather simple model that was commonly used in the past [Tu '99, Mollemans '03, Selle '08, Halic '09,

San-Vicente '12, Liu '13]. As a non-rigorous physical model, it represents a deformable object by a spring-network of mass points. Each massless spring (an edge of a discretized mesh) connects two points with masses m_i, m_j, positions $\boldsymbol{x}_i, \boldsymbol{x}_j$ and velocities $\boldsymbol{v}_i, \boldsymbol{v}_j$ respectively, as shown in Fig. 2.5.

Fig. 2.5 A mass spring connecting two points

The internal force along a spring is computed by the *Hooke's Law*, such that

$$\boldsymbol{f} = \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|}\left(k_s\left(|\boldsymbol{x}_j - \boldsymbol{x}_i| - L_i\right) + k_d|\boldsymbol{v}_j - \boldsymbol{v}_i|\right),$$

where k_s is the stretch stiffness, k_d the damping coefficient, and L_i is its rest length.

The mass-spring concept can be further generalized to a *particles system* that has the same mesh representation. However, the internal forces are computed differently, which is derived as the gradient of a deformation energy defined in terms of some constraints. Given a constraint function $C(\boldsymbol{p}_1, .., \boldsymbol{p}_n) = 0$, the deformation energy is defined as $U = \frac{1}{2}k \cdot C^2$, with k as the stiffness of the constraint, which is equal to zero at the initial pose. The internal forces are then computed as $\boldsymbol{f} = \frac{\partial U}{\partial \boldsymbol{p}}$. Therefore, more general types of springs can be defined, such as distant-spring, angular-spring, area-spring, and volume-spring [Baraff '98] [Teschner '04].

A state-of-the-art work on the mass-spring system was recently proposed by *Liu et al.* [Liu '13]. It reformulates the implicit Euler integration of equations of motion as an energy minimization problem, and achieves much faster and stable performance than the former mass-spring methods.

Modeling of a mass-spring system is straightforward. Due to its low computational complexity, real-time performance can be obtained. However, it has the following drawbacks that limit its applications:

- It is not based on strict physics laws, thus it can hardly obtain physically accurate results;
- It is difficult for users to tune the spring constants for a specific material, which means it is hard to control the mechanical behaviors;
- Deformations depend on mesh discretization, which means that the mechanical behaviors depend on the topology of the spring-mesh.

2.3 Physically-Based Deformable Models

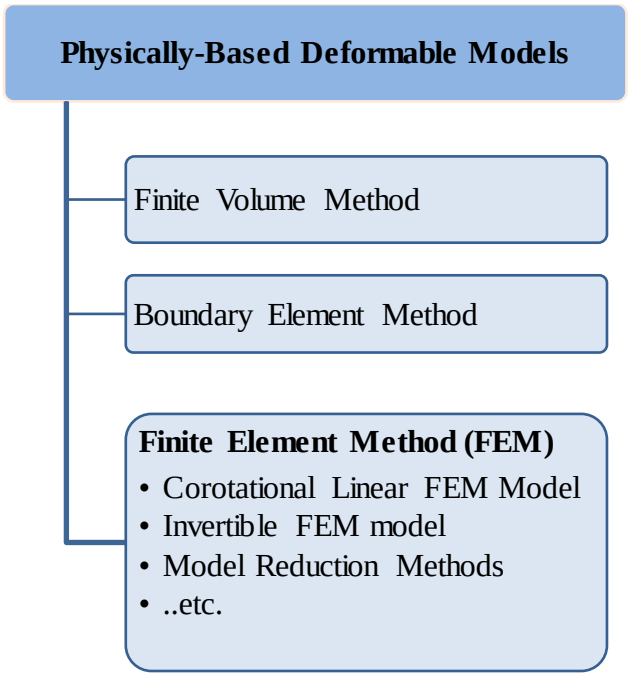


Fig. 2.6 Physically-baseddeformable models

Physically-based deformable models actually originate from continuum mechanics [Ward '92] [Kelly]. The continuum-based problem is commonly discretized and solved numerically by the *finite element method* (FEM) [Cook '07] [Bonet '08]. Since the

pioneering work by *Terzopoulos et al.* [Terzopoulos '87] on elastic models, physically based deformable models have been an active research area in computer graphics.

For most graphics applications such as animations, computer games, visual effects and virtual reality, physics simulation only plays a small part in the whole system. In contrast to physical models in *mechanical engineering* that require high accuracy, researchers in computer graphics [Gibson '97, Nealen '06, Müller '08, Sifakis '12] have made more efforts to achieve the goals of better stability, faster performance, and more controls over mechanical behaviors. There are always trade-offs between these goals and physical accuracy. Different models with specific concerns have been proposed, which are discussed below.

2.3.1 Stability-Concerned Models

Stability of simulation is of great concern for many applications. For example, if a simulation failed due to numerical insatiability in a game, it is not possible to re-run it (i.e., there is no second chance to correct it). Therefore, stability is more important than accuracy in this kind of scenarios.

Numerical instability is commonly caused by large deformations, large stiffness of system, large time steps, etc. In this section, we discuss some approaches dealing with these issues.

2.3.1.1 Large Deformations with Linear Models

Large deformations are quite common in graphics applications. A linear FEM model is computationally much more efficient than nonlinear FEM models; however, it is not rotation invariant (refer to **Section 3.1.2**), meaning that obvious volume distortions

occur when a linear model is under large deformations. As shown in Fig. 2.7 (b), the volume of the simulated object gets inflated due to a large external force.

(a) (b) (c)

Fig. 2.7 Comparison of Linear FEM and CLFEM.
(a) is the original shape; (b) shows the inflated-volume artifact with linear FEM; (c) shows the corrected deformation with CLFEM FEM. Red arrow represents the external load.

To solve this problem, *Müller et al.* [Müller '02] [Müller '04] proposed a *corotational FEM model* (CLFEM, also called *warped stiffness model*). The CLFEM model removes the factor of rigid rotational transformation in computation of internal forces, which makes it rotation invariant. As shown in Fig. 2.7 (c), the CLFEM model allows large deformations with well-preserved volume. Essentially being a linear elastic model, the CLFEM model can achieve deformation results comparable to a nonlinear FEM model, but at much less computational cost.

Here, rotation matrices between the initial and deformed shapes are obtained by a *polar decomposition*. A less costly approximation was implemented by *Allard et al.* [Allard '11]. *Georgii et al.* [Georgii '08] proposed an energy minimization method to get a more stable corotational formulation, and used a multi-grid FEM solver to further improve simulation performance.

2.3.1.2 Degenerate and Inverted Elements

Degenerate and inverted elements are inevitable when an object performs large deformations (e.g., under heavy compression or by overstretching). However, constitutive models are meaningful only for regular deformations. Moreover, the wrongly deformed elements would cause numerical instability, even leading to simulation failure. An example is shown in Fig. 2.8 (Top): a model is over-stretched, causing unstable deformation behaviors.

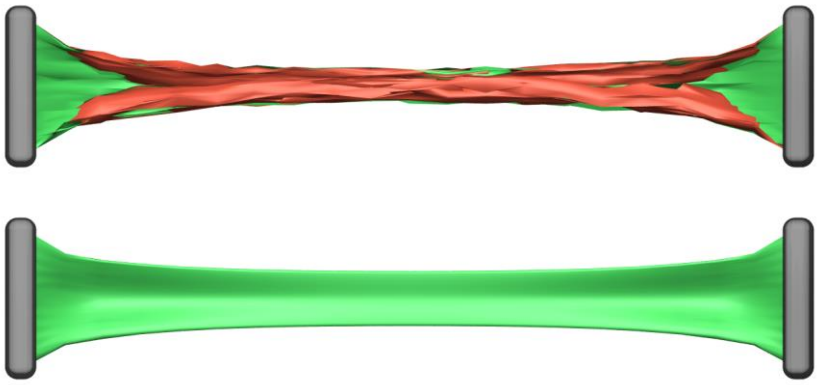


Fig. 2.8 (Top) unstable deformations caused by inverted elements; (Bottom) stable performance by a constitutive model with an energetically-based extension [Stomakhin '12]

Stress-Based Extension

An invertible FEM model (IFEM) was proposed by *Irving et al.* [Irving '04] [Irving '06]. The deformation gradient $F \in \mathbb{R}^{3\times3}$ (refer to **Section 3.1.1**) is used to detect whether an element is inverted, by checking the sign of its determinant. An inverted element is corrected by modifying the components of *SVD* decomposition of F. Meanwhile, a *modified neo-Hookean* material model was proposed to avoid extremely large stiffness under heavy compression. However, an explicit integration scheme was used here that required stringent restriction of time-step size. To alleviate this problem, an implicit time integration was formulated by *Teran et al.* [Teran '05]. By

manipulating element stiffness matrices, a positive-definiteness global Hessian matrix (i.e., the tangent stiffness matrix) was formulated to ensure numerical stability. *Sin et al.* [Sin '11] proposed an alternative method to compute the tangent stiffness matrix, which works on *constitutive models* defined by *principal stretches*, instead of the *invariants* based models as in [Teran '05].

Energetically-Based Extension

Modified constitutive models lose certain accuracy under extreme deformations. However, this trade-off for stability and visual plausibility is acceptable. In the aforementioned papers, modifications with constitutive models are *stress based*. An *energetially-based extension* was recently proposed by *Stomakhin et al.* [Stomakhin '12]. A stable simulation under large stretching deformations is shown in Fig. 2.8 (Bottom). The *primary contour* of a strain energy density function is used to analyze the robustness and stability of a constitutive model. A *smooth energy extension of* a constitutive model is formulated to get a favorable primary contour, and it is more robust than the stress-based extensions.

2.3.1.3 Large Time Steps

Implicit Time Integration

Explicit integration schemes are quite efficient for solving the system of equations of motion; however, they are only *conditionally stable*, meaning that the time-step sizes must be sufficiently small to ensure numerical stability, which limits their usage in practical applications. In contrast, implicit integrators are unconditionally stable but computationally much more expensive. Due to their stability with large time steps and large stiffnesses, they are popular in practical applications. A prominent work using an implicit integration scheme was proposed by *Baraff et al.* [Baraff '98] for cloth

simulation. Implementations of an implicit Euler integrator and an implicit *Newmark* integrator [Wriggers '06] can be found in the Vega FEM library [Jernej '12].

2.3.2 Efficiency-Concerned Models

Rigorous *nonlinear* constitutive models can produce physically accurate results, but expensive computations become a bottleneck for performance. Especially, it is not practical to use these nonlinear models in real-time or interactive applications. There is always trade-off between accuracy and speed in graphics applications. In this section, we discuss various approaches that were proposed to speed up the performance of dynamics simulation.

2.3.2.1 Optimization Implicit Euler

For nonlinear elastic models, large systems of nonlinear equations are produced with implicit integration, which are often solved by *Newton's method* or its variations. However, convergence behavior of Newton's method is unreliable at large time steps, particularly for stiff systems. Recasting the implicit Euler integration as a minimization problem (refer to **Section 3.3.3**) allows Newton's method to be stabilized by more robust optimization strategies [Nocedal '06]. Though essentially the same equations are solved, an optimization solver can be more robust and efficient. This approach recently gains popularity in graphics community [Kharevych '06, Martin '11, Liu '13, Bouaziz '14, Gast '14, Deng '15, Fr âncu '15].

With a specially designed energy function, an alternating optimization scheme was used in [Liu '13] and [Bouaziz '14]. It guarantees convergence and makes safeguards as in Newton's method unnecessary. Furthermore, in their formulations the global system matrix is constant thus can be pre-factored, resulting in a much more efficient solver than using implicit integrators.

2.3.2.2 Simplified Constitutive Models

For applications not requiring physically accurate results, simplified constitutive models (i.e., strain energy density functions) can be devised, which reduce cost related to computations of internal forces, tangent stiffness matrix, etc.

These simplified energy functions are geometrically motivated or simplified continuum-based constitutive models, which often have intuitive meanings for tuning material properties. For instance, in the seminal work of *Terzopoulos et al.* [Terzopoulos '87], the elastic models actually do not follow constitutive models as in continuum mechanics, but are formulated based on the *fundamental forms* of a parametric geometric shape (curve, surface, or solid). *Teschner et al.* [Teschner '04] formulated a simple constraint based energy: a constraint of the form $C(\boldsymbol{x}_0, .., \boldsymbol{x}_{n-1})$ is defined for the preservation of length, surface area or volume, based on which a potential energy can be formulated as $E = \frac{1}{2}kC^2$, where k is the stiffness of the constraint. Then the corresponding internal forces can be computed as $\boldsymbol{f}_i = -\frac{\partial E}{\partial \boldsymbol{x}_i}$, and the stiffness matrix K as $K_{ij} = \frac{\partial^2 E}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}$. *Huang et al.* [Huang '06] presented a potential energy in terms of the L^2 norm of the change of the *differential coordinates*, which was inspired by mesh editing methods [Alexa '03, Sorkine '04, Sorkine '07] discussed in **Section 2.1.1**; the corresponding stiffness matrix is approximately constant, which enables fast and stable implicit time integration. In [Martin '11] and [Coros '12], a simplified *St. Venant-Kirchhoff* material model was used to reduce computation.

2.3.2.3 Model Reduction Methods

Model reduction methods (also known as *subspace methods*) reduce computational complexity by decreasing the system DOF of dynamics simulation. The idea is that a

full DOF deformation space is approximated by a subspace spanned by a small number of *modes* (i.e., the *subspace basis*).

Suppose that a number of r vectors ($\in \mathbb{R}^{3n}$) are already obtained as the subspace basis, and are assembled by columns into a *modal matrix* $U \in \mathbb{R}^{3n\times r}$, where n is the number of vertices of a volumetric mesh. The full DOF displacement vector $\boldsymbol{u} \in \mathbb{R}^{3n}$ is projected into the subspace spanned by the column vectors of U, and we have $\boldsymbol{u} = U\boldsymbol{q}$, where $\boldsymbol{q} \in \mathbb{R}^{r}$ is called *modal coordinates*. Therefore, the original full DOF equations of motion $M\ddot{\boldsymbol{u}} + D\dot{\boldsymbol{u}} + \boldsymbol{f}_{int}(\boldsymbol{u}) = \boldsymbol{f}_{ext}$ (refer to **Section 3.3.1**) become a reduced system with DOF of r, as

$$U^T M U\ddot{\boldsymbol{q}} + U^T D U\dot{\boldsymbol{q}} + U^T \boldsymbol{f}_{int}(U\boldsymbol{q}) = U^T \boldsymbol{f}_{ext},$$

where $M \in \mathbb{R}^{3n\times 3n}$ is the mass matrix, $D \in \mathbb{R}^{3n\times 3n}$ the damping matrix, and $\boldsymbol{f}_{int} \in \mathbb{R}^{3n}$ and $\boldsymbol{f}_{ext} \in \mathbb{R}^{3n}$ are internal and external forces respectively. Therefore, instead of solving a system with full DOF unknowns, only r unknowns need to solved, where $r \ll 3n$.

Various subspace basis generation methods are discussed below.

Linear Modal Analysis

A common way of generating deformation modes is to use *linear modal analysis* (LMA) [Shabana '96] [Hunter '01]. *Pentland et al.* [Pentland '89] pioneered the use of LMA in graphics: given the mass matrix M and the system stiffness matrix K, a generalized eigen-value problem $K\boldsymbol{x} = \lambda M\boldsymbol{x}$ is solved. The eigenvectors are called *vibration modes*, with *frequencies* equal to the roots of the corresponding eigenvalues. Only a small number of modes with *low frequencies* need to be pre-computed, which are used to approximate the full space of deformations. Fig. 2.9 shows ten linear modes with low frequencies of a dino model, with its feet fixed in positions.

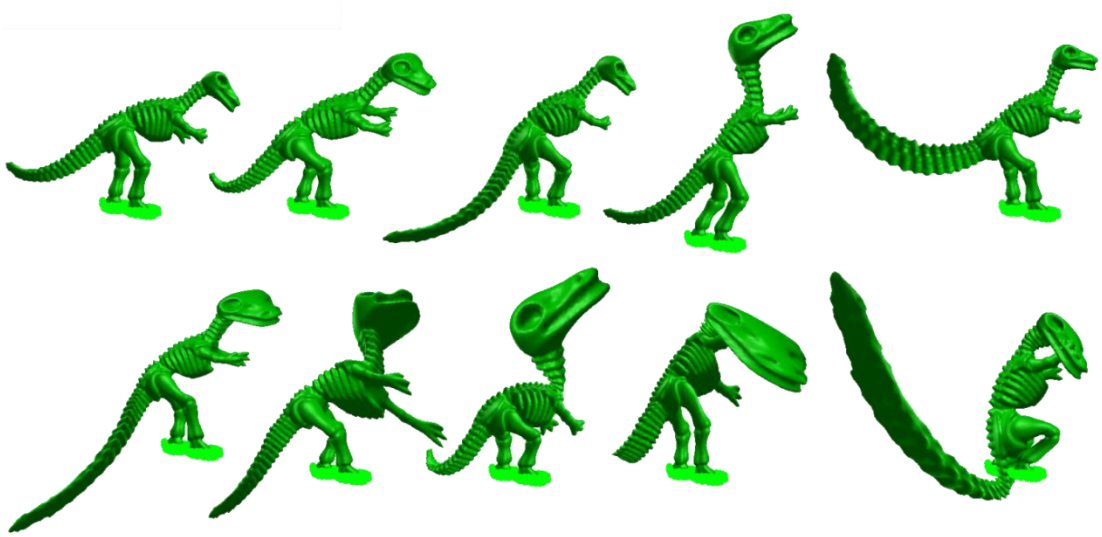


Fig. 2.9 Ten linear models of the a constrained dino model (with its feet fixed in positions)

An interactive reduced simulation with LMA was presented by *Hauser et al.* [Hauser '03]. With pre-computed linear modes and Rayleigh damping, they showed that the system of motion equations could be *decoupled* into a set of second-order differential equations, which could be solved independently and efficiently.

The advantages of using linear modes are that the linear modes can be precomputed efficiently (using math libraries such as *ARPACK* [Lehoucq '98]), and that simulation complexity only depends on the number of modes and is independent of the complexity of the simulated mesh. Real-time performance can be achieved, which is useful in interactive applications. However, since only linear modes are used, noticeable artifacts would occur for large deformations, such as volume distortion (similar to the artifacts caused by the linear FEM model) and locking artefacts [Barbič '07] (also known as *artificial stiffening*, meaning that deformations are constrained to a limited range, even if large external loads are applied).

Modal Derivatives

To support large deformations with subspace methods, *Barbič et al.* [Barbič '05] proposed a *modal derivatives* method for generating nonlinear subspace basis. A combination of linear modes and *tangent linear vibration modes* are shown in Fig. 2.10.

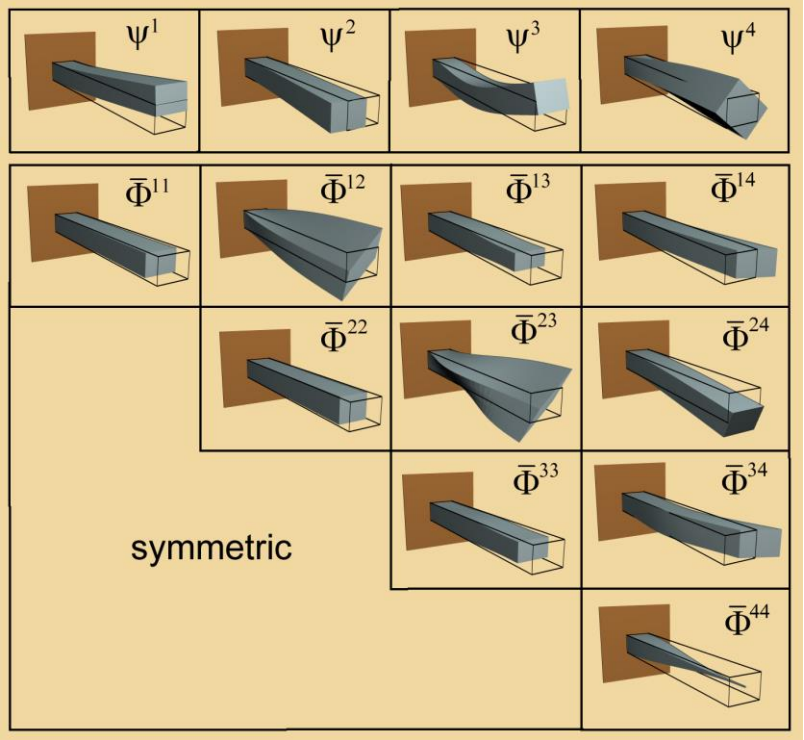


Fig. 2.10 Linear modes Ψ^i and mass-normalized modal derivates $\bar{\Phi}^{ij}$ [Barbič '05]

The formulation is based on a particular *St. Venant-Kirchhoff* (StVK) material. With r linear modes, $r * (r + 1)/2$ modal derivatives are computed, and the final low-dimensional deformation basis is obtained by applying *mass-PCA* on a total number of $(r + r * (r + 1)/2)$ modes. The generated modes of the dino model are shown in Fig. 2.11: Mass-PCA is applied on a combination of 10 linear modes and 55 modal derivatives, and the first 10 modes are shown here.

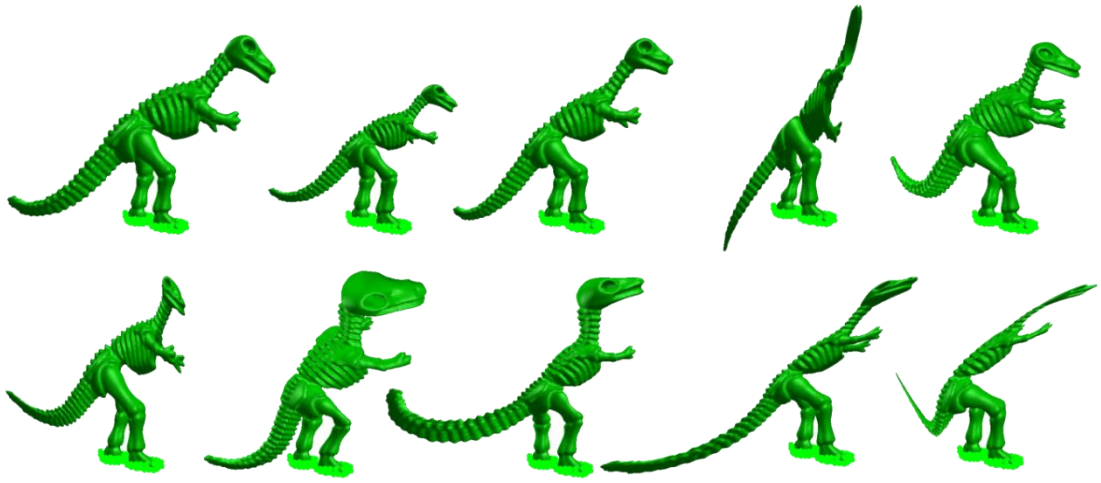


Fig. 2.11 By applying mass-PCA on 65 combined modes, the first ten modes are shown here.

For comparison, we show the deformations generated by a full DOF model, the modal derivatives method and the LMA method respectively in Fig. 2.12. It shows that the modal derivatives method (Fig. 2.12 (b)) can generate deformations comparable to a full DOF simulation (Fig. 2.12 (a)), but the LMA method exhibits a *stiffening* artifact (Fig. 2.12(c)).

(a) (b) (c)

Fig. 2.12 (a) full simulation; (b) modal derivatives method; (c) linear modes

Modal Warping

The modal derivatives method with the StVK material can generate physically-plausible results, but it cannot be easily extended to other constitutive models. *Tycowicz et al.* [von Tycowicz '13] proposed a simple extension by changing

components of the linear modes. *Choi et al.* [Choi '05] proposed a *modal warping* method for large deformations, eliminating distortions caused by linear modes through extrapolating element rotations (similar to the idea of stiffness warping). *Huang et al.* [Huang '11] proposed a warping method based on *Rotation-Strain coordinates* (RS), which corrected the distortions of linear modes by post processing. This method was also employed in a deformation editing system [Barbič '12]. *Li et al.* [Li '14a] extended this RS concept to a reduced RS space to further improve performance.

Neither of the modal warping method nor the RS method is physically correct, but both of them can produce visually-plausible results with only linear modes.

Considering Local Deformations

Subspace methods greatly reduce computational cost; however, the subspace basis only has global support such that local deformation details can hardly be captured. One practical solution is to decompose a simulated object into a number of small subdomains, which is known as a *multi-domain* approach. *Barbič et al.* [Barbič '11] presented a deformable model with articulated tree-structured subdomains, which are connected by small or nearly rigid interfaces. *Kim et al.* [Kim '11b] employed the subspace techniques in physically-based skinning, where a character is decomposed according to its bone structures. The inter-domain interfaces can be large, and the subdomains are coupled using penalty forces (soft constraints) to avoid crack artifact. *Yang et al.* [Yang '13] improved the coupling constraints to avoid over-constraint problem. A more physically accurate method [Harmon '13] is to compute local deformation basis from the *analytic* solutions of a *static loading problem*, but it costs much more than the former multi-domain methods.

Discussion

Compared with full DOF simulation methods, subspace techniques can greatly reduce computational complexity. By trading accuracy for speed, they can be applied to interactive applications that require only visual plausibility. Note that since the subspace basis is pre-computed, topology of the simulated mesh should not be changed (e.g., no fracturing) during simulation. The subspace concept is also employed for shape analysis and shape editing [Hildebrandt '10, Hildebrandt '11, Hildebrandt '12, von Tycowicz '14] , where eigen values and eigen vectors (also called *spectra*, or *eigen modes*) of a Hessian matrix of a discrete surface energy are analyzed.

2.3.2.4 Cubature Schemes

Reduced internal forces in the former subspace methods are evaluated, either by computing full DOF internal forces (at high computational cost) and projecting to a subspace by $U^T \boldsymbol{f}_{int}$, or by a particular formulation of the StVK material with complexity of $O(r^4)$ [Barbič '05]. *An et al.* [An '08] proposed a *cubature* scheme (as an extension of *Gaussian quadrature* scheme) for efficiently integrating reduced internal forces, which is at a cost of $O(r^2)$ for $O(r)$ cubature points. This scheme is successfully applied in real-time applications, such as physically-based characters skinning [Kim '11b] and deformation editing [Li '14a].

2.3.2.5 Generalized Continuum-based Models

Frame-based Elastic Models

Jumping out of rigorous physics-thinking, *Gilles et al.* [Gilles '11] [Faure '11] proposed a new deformable model that combines concepts of continuum mechanics and frame-based skinning techniques (such as *linear blending* and *dual-quaternion blending*) used in skeletal animation. As shown in Fig. 2.13, a sparse set of frames is

distributed inside a mesh object. The DOF required to be solved are the low dimensional coordinate frames, instead of the mesh nodes as in previous FEM discretization. Based on continuum mechanics, mechanical quantities (such as mass matrix, internal forces and stiffness matrix) are generalized to be formulated in terms of the DOF of the frames. The equations of motion are also reformulated, yielding a system of rather small dimensions. Dynamics motion of the frames is computed, and finally the coupled mesh is deformed by a skinning method.

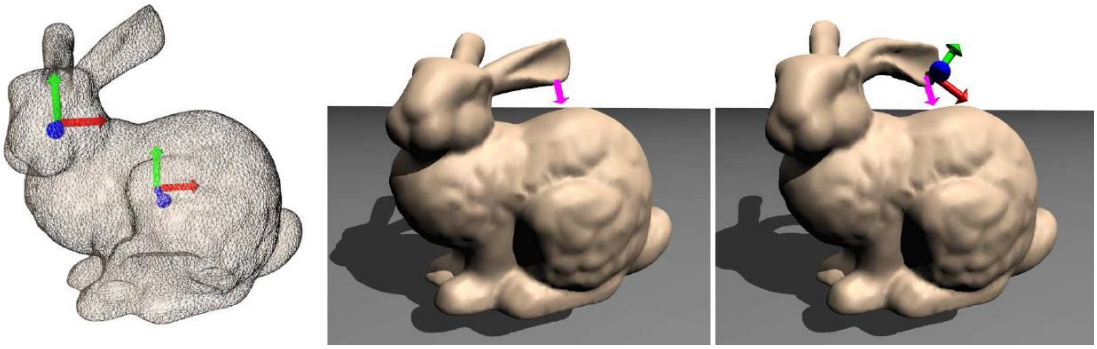


Fig. 2.13 Frame-based deformable model. Two frames are able to model a deformable body (Left). Besides material properties, flexibility of the simulated object also depends on the distribution of frames: by inserting a new frame at one ear (Right), the ear becomes more flexible than the previous one (Middle). [Gilles '11]

One advantage of this method is that local deformations can be supported by inserting new frames on-the-fly. More follow-up works can be found in [Gilles '13, Tournier '14a, Tournier '14b].

Rig-Space Physics

Similar in spirit with the frame-based generalization, *Hahn et al.* [Hahn '12] [Hahn '13] further generalized the formulation in a rig-space for a rigged character, which unifies physics simulation and keyframing. The output consists of animation curves. It is useful in animation design, which provides artists with intuitive control of mechanical

behaviors through high-level rig parameters, and meanwhile automatically produces secondary dynamic motions.

2.4 Hybrid Models: Bridge the Gap between Geometrical and Physical Models

Geometrically-based and physically-based deformable models are not totally disconnected from each other, but to some extent closely related. The elastic models proposed by *Terzopoulos et al.* [Terzopoulos '87] aforementioned are a typical example. *Chao et al.* [Chao '10] provided a proof of the relationship between a geometric model and a *Biot strain* based model. In this section, we discuss some hybrid models that benefit from both groups: they take advantage of the stability and efficiency of geometrical models, while still have continuum-based foundation.

2.4.1 Continuum-Based Constraints within a PBD Framework

In the original PBD framework (as discussed in **Section 2.1.2.2**) geometric constraints are used. Limitations with geometric constraints are that deformation behaviors depend on the tessellation (or, discretization) of a simulated object, and material properties are defined in terms of constraint stiffnesses.

Recently, *Müller et al.* [Müller '14] presented a strain-based constraint defined on the entries of the rotation-invariant *Green strain tensor* (refer to **Section 3.1.2**). *Anisotropic* material properties can be defined in terms of the stiffnesses of stretches and shears along different material directions. Similar in spirit, [Bender '14a] used a *continuum-based energy* as a constraint, such as the potential energy of the StVK material or the *Neo-Hookean* material.

This kind of continuum-based formulations does not suffer from the tessellation bias problem, and provides continuum mechanics support for the PBD framework.

2.4.2 Continuum-Based Constraints within an Optimization Framework

Bouaziz et al. [Bouaziz '14] proposed a *projective dynamics* approach that recasts implicit Euler integration to an optimization problem (as discussed in **Section 2.3.1.3**). It generalizes the *local/global optimization scheme* from a mass-spring system [Liu '13] to the FEM methods. Various constraints are supported, including both geometric constraints (e.g., [Bouaziz '12]) and continuum-based constraints. With a specially designed continuum-based energy function that includes additional auxiliary variables, the connection between PBD and FEM methods is further analyzed.

2.5 Control Methods of Deformable Models

We have discussed various methods for dynamics simulation of deformable models. However, this is only part of the story. In graphics applications, *deformation control* is also much desirable but poses challenges, such as stably intuitive adjustment of material properties, and interactive control over complex mechanical behaviors. In this section, we discuss several methods related to deformation control.

2.5.1 Example-Based Methods

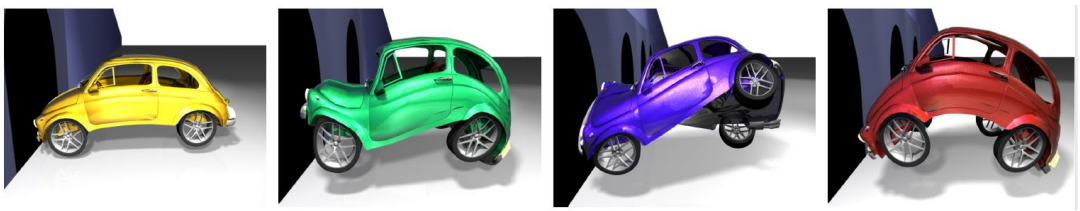


Fig. 2.14 Example-based deformations [Martin '11]. Deformations are artistically controlled by various example poses.

In computer animations, dynamics simulation relieves artists of laborious keyframing for modeling dynamic behaviors of deformable objects. Secondary dynamic motions like jiggling effects can be obtained automatically. However, deformations produced by simulation are not always what an artist wants, and it is often very difficult or impractical to tune the material properties of a complex model in order to obtain the desired deformations.

Usually, an artist has a vision of how a deformable object should be deformed in a certain scenario. Motivated by this requirement, *Martin et al.* [Martin '11] proposed an *example-based method*, which takes some pre-designed example poses as input and generates deformations complied with these examples, as shown in Fig. 2.14. The idea is that an example space spanned by the example poses is used to derive an *example-based potential*, from which attractive forces are derived to project a deformed configuration into the example space. *Schumacher et al.* [Schumacher '12] improved the efficiency of this method by using incompatible rest shapes. *Koyama et al.* [Koyama '12] applied this idea within a *shape matching* framework (refer to **Section 2.1.2.1**) to achieve real-time performance. *Zhang et al.* [Zhang '15] proposed a real-time integration method in a reduced subspace, where the example poses are included to form a subspace basis.

Example-based methods provide artistic control over complex deformation behaviors without manipulating complicated material properties, though the artificial forces induced by the example-based potential affect physical accuracy. A limitation is that the deformation behaviors might not be predictable if too many example poses are provided, leading to the question of how many examples should be designed to achieve both complex and *directable* deformations.

2.5.2 Space-Time Control

Space-time control method is used for editing elastic object animation. It is formulated as an optimization problem: existing animation sequences are edited through position constraints, then an optimization procedure seeks *optimal control forces* (i.e., least amount of non-physical/artificial forces) to match the input spatial constraints in time.

Barbič et al. [Barbič '08] used a time-varying *linear quadratic regulator* to drive an object to follow pre-defined trajectories. Later, a *reduced space-time optimization* method [Barbič '09] was proposed to enforce a deformable object to follow a sparse set of pre-designed keyframes. Furthermore, an interactive animation editor [Barbič '12] was presented: some keyframes of an existing animation are edited, then the whole animation is reproduced, respecting both physics and user editing. Recently, *Li et al.* [Li '14a] proposed a space-time editing scheme that optimizes not only control forces but also material properties to better match user defined constraints.

Instead of optimizing control forces or material properties, *Coros et al.* [Coros '12] proposed a *rest shape adaption method*. It dynamically changes the rest shape of a deformable object during simulation. In order to satisfy a user-defined *locomotion goal* (e.g., position and velocity of the center-of-mass of an object) at a certain time step, the rest shape is adapted such that a new deformed shape propels itself through interactions with the environment (e.g., collision and friction). It makes a deformable object acts like a living creature, in a self-propelled way that changes its shape through internal forces instead of external artificial forces. However, this method is limited with a planning horizon of one time step, and can be difficult to generate meaningful motions for a complex model in a long time range rather than a single time step.

2.6 Discussion

We have done a survey on simulation of deformable models. Various approaches, which have their own advantages and limitations, have been analyzed. The main space of this chapter is devoted to continuum-based approaches in combination with the finite element method. Other related topics such as fracture [Chen '14], and collision detection and response [Jiménez '01] [Teschner '05] are beyond this survey, thus they are not covered here.

There are some helpful open source libraries worth mentioning. We began with the *OpenCloth* library [Movania '11] that is simple and easy-to-follow. It includes cloth simulation, the corotational FEM model and various time integrators. The *Vega FEM* library [Jernej Barbič '12] has implemented several physically based deformable models, and provided useful data structures and linear algebra algorithms. The *SOFA* framework [Allard '07], which includes GPU implementations, primarily targets at real-time medical simulation for soft tissues,. There are also some open source physics engines that support deformable models, such as *Bullet physics engine* [Coumans '10], which is a professional 3D real-time multi-physics library.

Chapter 3. Dynamics Simulation in a Nutshell

Physically-based simulation of deformable models is based on continuum mechanics, and the mathematical models originated from engineering disciplines rather than computer science. Therefore, knowledge of fundamentals such as solid mechanics, elasticity theory, dynamics and numerical methods is a prerequisite for research on physics simulation.

In this chapter, we briefly introduce some of the essential concepts and formulations, which mainly include the following aspects:

- Important concepts in **solid mechanics**: strain, stress, elasticity, constitutive models, deformation energy, internal forces, etc.;
- The **finite element analysis/method**: as a numerical technique to solve continuum-based problems, by discretizing a continuous domain into small *finite elements*;
- **Time integration schemes**: numerical methods used to solve the dynamics equations of motion (i.e., system of differential equations with boundary value problems) .

3.1 Elasticity in Three Dimensions

In three-dimensional space, deformation of a solid body can be formulated as a function $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}^3$, mapping an *initial* (or *reference*) *configuration* into a *deformed configuration*. As shown in Fig. 3.1, a particle (*material point*) in the undeformed shape is denoted by the coordinate $\boldsymbol{X} = (X_1, X_2, X_3) \in \mathbb{R}^3$ with respect to the Cartesian basis $\boldsymbol{E}_i, i = 1,2,3$; the corresponding particle in the deformed shape is denoted as $\boldsymbol{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$ with respect to another Cartesian basis $\boldsymbol{e}_i$. Generally, these two coordinate systems are coincident.

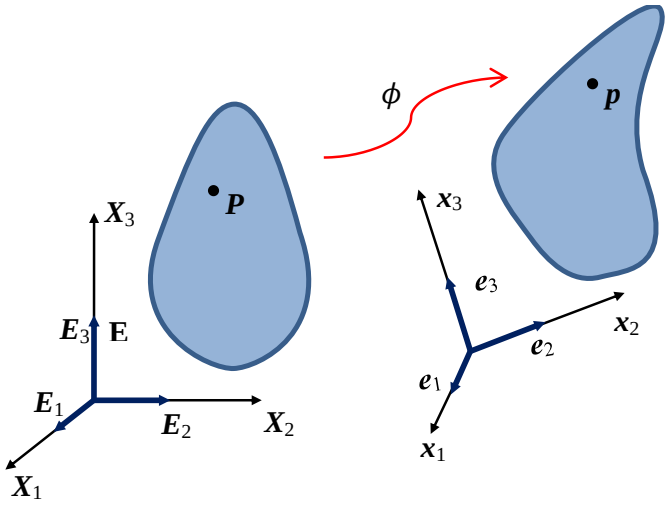


Fig. 3.1 Deformation mapping ϕ

Thus we have the deformation mapping function as

$$\boldsymbol{x} = \phi(\boldsymbol{X}) \tag{3.1}$$

In dynamics simulation, this mapping also includes another *time* variable t as

$$\boldsymbol{x}(t) = \phi(\boldsymbol{X}, t)$$

This deformation can also be depicted by a *displacement* vector $\boldsymbol{u}(\boldsymbol{X})$, such that

$$\boldsymbol{x} = \boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X}) \tag{3.2}$$

3.1.1 Deformation Gradient

A key quantity in deformation analysis is the *deformation gradient*, defined as

$$F = \frac{\partial \phi(\boldsymbol{X})}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + I, \tag{3.3}$$

where $I \in \mathbb{R}^{3\times3}$ is an identity matrix. From Eq. (3.3) we know that $F \in \mathbb{R}^{3\times3}$ is a second-order tensor,

$$F = \begin{pmatrix} \partial x_1/\partial X_1 & \partial x_1/\partial X_2 & \partial x_1/\partial X_3 \\ \partial x_2/\partial X_1 & \partial x_2/\partial X_2 & \partial x_2/\partial X_3 \\ \partial x_3/\partial X_1 & \partial x_3/\partial X_2 & \partial x_3/\partial X_3 \end{pmatrix}.$$

Meanings of F

The deformation gradient has certain physical meanings and contains the information connecting the initial and deformed configurations. For example, suppose that a material vector $d\boldsymbol{X}$ is deformed into a corresponding spatial vector $d\boldsymbol{x}$, we have

$$d\boldsymbol{x} = F d\boldsymbol{X}.$$

Considering an element of infinitesimal volume in the original configuration, which is formed by three orthogonal material vectors $d\boldsymbol{X}_i = dX_i \boldsymbol{E}_i, i = 1,2,3$, the volume in the initial configuration can be computed as

$$d\boldsymbol{V} = d\boldsymbol{X_1} \cdot d\boldsymbol{X_2} \times d\boldsymbol{X_3} = d\boldsymbol{X_1} d\boldsymbol{X_2} d\boldsymbol{X_3}.$$

And the corresponding deformed volume dv is computed as

$$dv = (F d\boldsymbol{X}_1) \cdot (F d\boldsymbol{X}_2) \times (F d\boldsymbol{X}_3) = \det(F)\, dV.$$

Therefore, the Jacobian $J = \det(F) = \frac{dv}{dV}$ represents the *fraction of volume change* during deformation.

Moreover, the deformation gradient can be decomposed by *polar decomposition* [Bonet '08] into a *rotation tensor* R and a *stretch tensor* U as $F = RU$. So we get

$$d\boldsymbol{x} = RU d\boldsymbol{X} = R(U d\boldsymbol{X}).$$

From this decomposition, it means that a material vector $d\boldsymbol{X}$ is firstly stretched in the material space, and then is rotated in the spatial space.

Discussion

From the value of J, we can judge how the volume changes during deformation. For example, $J = 1$ means no volume change; $J < 0$ means that there is inverted deformation. The invertible FEM model in [Irving '04] utilized the value to detect and deal with the issue of inverted elements.

In [Huang '11], the polar decomposition of F was used to construct a *rotation-strain space* that describes a deformed object with *rotation-strain coordinates*. As a *descriptor* of deformation, the *rotation-strain coordinates* were also exploited in shape interpolation and shape editing (e.g., [Barbič '12]).

The deformation gradient F can be used as *measure* of deformation; however, due to its variance under rotations, it is not a proper choice. A better choice for the measure of deformation is described below.

3.1.2 Deformation Measure by Strain Tensor

3.1.2.1 The Nonlinear Green Strain Tensor

Given two material vectors $d\boldsymbol{X}_1, d\boldsymbol{X}_2$ in the reference configuration and their counterparts $d\boldsymbol{x}_1, d\boldsymbol{x}_2$ in the deformed configuration, a deformation can be measured by the difference of two scalar products, as

$$\frac{1}{2}(d\boldsymbol{x}_1 \cdot d\boldsymbol{x}_2 - d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2)$$

$$= \frac{1}{2}[(F d\boldsymbol{X}_1) \cdot (F d\boldsymbol{X}_2) - d\boldsymbol{X}_1 \cdot d\boldsymbol{X}_2]$$

$$= d\boldsymbol{X}_1 \cdot \tfrac{1}{2}(F^T F - I) d\boldsymbol{X}_2.$$

From the above, the *right Cauchy-Green deformation tensor* is defined, as

$$C = F^T F \in \mathbb{R}^{3\times 3}\ , \tag{3.4}$$

and the *Green strain tensor* is defined as

$$E = \frac{1}{2}(F^T F - I)\ \in \mathbb{R}^{3\times 3}. \tag{3.5}$$

Thus we have

$$E = E(F) = \frac{1}{2}(C - I) = \frac{1}{2}[(RU)^T(RU) - I] = \frac{1}{2}(U^2 - I) \tag{3.6}$$

It shows that the Green strain tensor $E \in \mathbb{R}^{3\times 3}$ is symmetric, and *rotation invariant* by removing the effect of rotational transformation, with only deformation information contained in the quantity U^2. If a deformation only contains a pure rotation, i.e., $F = R$, then $E = \frac{1}{2}(R^T R - I) = \mathbf{0}$, meaning there is no deformation but a rigid transformation. With this rigid motion invariant property, F is often used as a *measure* of deformation in the literature. In the example-based method of [Martin '11], the Green strain tensor is used as a unique *descriptor* of element deformation.

From Eq. (3.6), we know that the Green strain tensor is *nonlinear* in terms of deformation (i.e., the displacement vector). Nonlinear elastic model based on this strain tensor is computationally expensive. In order to reduce computation and improve computational efficiency, a linearized strain tensor is often used in case of small deformations as described below.

3.1.2.2 The Linear Cauchy Strain Tensor

The nonlinear Green strain can be linearized by *Taylor expansion* around the undeformed configuration (i.e., $F = I$) [Fackler '05] to obtain a linearized form (see **Appendix A.1.1**), as

$$\varepsilon = \frac{1}{2}(F + F^T) - I, \tag{3.7}$$

which is called the *Cauchy strain tensor* or the *small strain tensor*.

The advantage of using this linear strain tensor is that it greatly reduces computational complexity. However, it is not rotation invariant, thus is only suitable for small deformations [Müller '04].

3.1.3 Elasticity and Measure of Deformation Energy

Here we introduce *elasticity theory* from an energy perspective. Potential energy is stored in a deformed body, which is called the *strain energy*. If the strain energy only depends on the final deformed configuration and is independent of deformation paths, this kind of ideal elastic materials is considered being *hyperelastic*. For hyperelastic materials, the *strain-stress* relationship derives from a strain energy density function. Only *hyperelasticity* is discussed in our work, where an object is deformed under loads and returns to its original configuration when unloaded.

3.1.3.1 Constitutive Models of Isotropic Materials

In order to obtain the strain energy of a deformed body, a *strain-energy density* function $\psi(\phi; \boldsymbol{X})$ needs to defined, measuring the strain energy *per unit undeformed volume* dV. By integrating the energy density function over the entire undeformed domain Ω, we obtain the strain energy, as

$$W = \int_{\Omega} \psi(\phi; \boldsymbol{X})\, dV.$$

The strain-energy density function $\psi(\phi; \boldsymbol{X})$ can be defined by different mathematical models, which are called the ***constitutive models***. A constitutive model connects the measure of deformation with the properties of a material.

A commonly used strain-energy density function for *isotropic* materials is defined as

$$\psi(\varepsilon) = \mu\varepsilon : \varepsilon + \frac{\lambda}{2}[tr(\varepsilon)]^2. \tag{3.8}$$

Here, $\varepsilon \in \mathbb{R}^{3\times3}$ denotes a strain tensor (either in Eq.(3.5) or in Eq. (3.7)). λ and μ are *Lamé coefficients*, which are related to *Young's modulus* k (as a measure of resistance of stretching) and *Poisson's ratio* υ (as a measure of resistance of compression), and

$$\mu = \frac{\kappa}{2(1+\nu)},\ \lambda = \frac{\kappa\nu}{(1+\nu)(1-2\nu)}.$$

If the nonlinear Green strain tensor E in Eq. (3.7) is used in place of ε, we get a simple *hyperelastic model* called *St. Venant-Kirchhoff* (StvK) model, which is often used in graphics community (e.g., [Barbič '05]). There are many other constitutive models of isotropic materials, such as the *Mooney–Rivlin materials* (see **Appendix B.1**) and the *Neo-Hookean materials* (see **Appendix B.2**), which are defines in terms of the invariants of the Cauchy-Green tensor in Eq. (3.4) .

3.1.3.2 Discussion

In graphics applications, extremely large deformations and degenerate meshes often occur. These cause numerical instability of computation with the rigorously defined constitutive models. One possible solution is to modify a constitutive model (i.e., the deformation energy function) so that these extreme cases can be properly handled without affecting the normal deformation behaviors, such as [Irving '04, Irving '06, Stomakhin '12] discussed in **Section 2.3.1.2**.

3.1.4 Measure of Forces by Stress Tensor

Another important quantity in solid mechanics is the *stress tensor*, which is a fundamental *force descriptor* that measures the internal forces incurred by deformation. The well-known *Cauchy stress tensor* $\boldsymbol{\sigma}$ is often used, measuring *force per unit area* in the current configuration. For a linear model, it can be computed as $\boldsymbol{\sigma} = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}}$.

In a contracted form, the symmetric strain tensor $\boldsymbol{\varepsilon}$ (either in Eq.(3.5), or in Eq. (3.7)) can be written as a 6×1 vector (see **Appendix A.1.2**) as

$$\boldsymbol{\varepsilon} = (\varepsilon_{11}\ \varepsilon_{22}\ \varepsilon_{33}\ \gamma_{23}\ \gamma_{31}\ \gamma_{12})^T = (\varepsilon_{11}\ \varepsilon_{22}\ \varepsilon_{33}\ 2\varepsilon_{23}\ 2\varepsilon_{31}\ 2\varepsilon_{12})^T, \tag{3.9}$$

and then the contracted Cauchy stress vector is derived as

$$\boldsymbol{\sigma} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{23}, \sigma_{31}, \sigma_{12})^T = G\boldsymbol{\varepsilon},$$

where G is called *material stiffness* matrix (see **Appendix A.1.3**), as

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

$$= \frac{\kappa v}{(1+v)(1-2v)} \begin{pmatrix} 1-v & v & v & 0 & 0 & 0 \\ v & 1-v & v & 0 & 0 & 0 \\ v & v & 1-v & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-v & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-v & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-v \end{pmatrix}.$$

Therefore, a *constitutive model* can also be defined in terms of stress-strain relationship.

Another two stress tensors commonly used are:

1) The *First Piola–Kirchhoff stress tensor*, which can be interpreted as force (in the deformed configuration) per unit area (in the initial configuration),

$$P = \frac{\partial \psi}{\partial F},$$

which is used for computing internal forces in papers such as [Teran '03, Irving '04, An '08].

2) The *Second Piola–Kirchhoff stress tensor*, which can be interpreted as force (in the initial configuration) per unit area (in the initial configuration) is defined as

$$S = \frac{\partial \psi}{\partial E} = 2\frac{\partial \psi}{\partial C}.$$

The relationship between the two stress tensors is that

$$P = FS.$$

3.2 Discretization with Finite Element Method

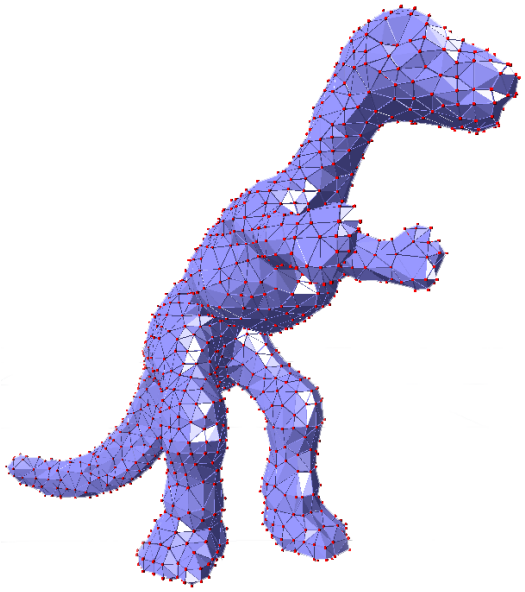


Fig. 3.2 A tetrahedral finite element mesh

So far, all the physical quantities are described in a continuous space. However, in most cases it is difficult or impossible to find analytical solutions for a simulation

problem with complex geometry. Therefore, it needs to be solved numerically. The most universal finite element formulation works on a large number of discretized elements. The discretization is established in the initial configuration using *isoparametric* elements ([Felippa '04], Chapter 16 & 18). The physical state (such as positions, displacements, and velocities) is assigned to the nodes of discretized mesh, and the continuous functions (such as deformation map, strain energy and internal forces) are reformulated in terms of the discrete variables.

In our work, we use a *tetrahedral mesh* as the representation of discretization, as shown in Fig. 3.2. Suppose that a tetrahedral mesh contains n nodes formed into n_e elements. For simplicity, we firstly discuss the related discretization within an element. The initial *nodal positions* of an element are defined as $(\boldsymbol{X}_0^T, \boldsymbol{X}_1^T, \boldsymbol{X}_2^T, \boldsymbol{X}_3^T)^T \in \mathbb{R}^{12\times 1}$ where $\boldsymbol{X}_i \in \mathbb{R}^{3\times 1}$, $i = 1,..,4$, and the deformed counterpart as $(\boldsymbol{x}_0^T, \boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \boldsymbol{x}_3^T)^T \in \mathbb{R}^{12\times 1}$. By interpolation of geometry using standard shape functions $N_i(\boldsymbol{\xi}) = N_i(\xi_1, \xi_2, \xi_3)$, a material point position in the initial configuration can be interpolated as $\boldsymbol{X} = \sum_{i=1}^{4} N_i(\boldsymbol{\xi})\boldsymbol{X}_i$ and the deformed one as $\boldsymbol{x} = \sum_{i=1}^{4} N_i(\boldsymbol{\xi})\boldsymbol{x}_i$. Thereafter, the following quantities can be computed:

1) The **deformation gradient** is obtained by $F = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \frac{\partial}{\partial \boldsymbol{X}}\left(\sum_{i=1}^{4} N_i(\boldsymbol{\xi})\boldsymbol{x}_i\right)$, which involves solving the following term,

$$\frac{\partial N_i}{\partial \boldsymbol{X}} = \left(\frac{\partial \boldsymbol{X}}{\partial \xi}\right)^{-T} \frac{\partial N_i}{\partial \xi}.$$

2) The **strain energy** is obtained by an integral of a given constitutive model $\psi(F)$,

$$W^e = \int_{\Omega^e} \psi(F)\, dV.$$

3) The **nodal internal force** is computed as

$$\boldsymbol{f}_i^e = -\frac{W^e}{\boldsymbol{x}_i} \in \mathbb{R}^{3\times 1}, i = 1,..,4\,.$$

4) The **element stiffness matrix** is computed as

$$K^e = \frac{\partial(\boldsymbol{f}_1^e, \boldsymbol{f}_2^e, \boldsymbol{f}_3^e, \boldsymbol{f}_4^e)}{\partial(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)} \in \mathbb{R}^{12\times 12}.$$

Assembly of all the elements can obtain the global internal force $\boldsymbol{f} \in \mathbb{R}^{3n\times 1}$ and global stiffness matrix $K \in \mathbb{R}^{3n\times 3n}$.

3.3 Formulation of Dynamics Simulation

3.3.1 The Euler-Lagrangian Equations of Motion

Dynamics of a deformable object is governed by the *Euler-Lagrange equations* of motion, which are derived by Lagrangian mechanics [Morin '08] [Shabana '09], as

$$\begin{aligned} M\ddot{\boldsymbol{u}} + D\dot{\boldsymbol{u}} + \boldsymbol{f}_{int}(\boldsymbol{u}) &= \boldsymbol{f}_{ext}, \\ or,\ M\ddot{\boldsymbol{x}} + D\dot{\boldsymbol{x}} + \boldsymbol{f}_{int}(\boldsymbol{x}) &= \boldsymbol{f}_{ext} \end{aligned} \tag{3.10}$$

where $\boldsymbol{u} = \boldsymbol{u}(t) \in \mathbb{R}^{3n\times 1}$ is the unknown displacement vector, $\boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^{3n\times 1}$ the current positions vector, $M \in \mathbb{R}^{3n\times 3n}$ the mass matrix, $D\dot{\boldsymbol{u}} \in \mathbb{R}^{3n}$ is the damping forces with $D \in \mathbb{R}^{3n\times 3n}$ as the damping coefficients, $\boldsymbol{f}_{int}(\boldsymbol{u}) \in \mathbb{R}^{3n\times 1}$ the internal forces, and $\boldsymbol{f}_{ext} \in \mathbb{R}^{3n\times 1}$ the external forces. Since $\boldsymbol{x} = \boldsymbol{X} + \boldsymbol{u}$, we have $\dot{\boldsymbol{x}} = \dot{\boldsymbol{u}}$, and $\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{u}}$, and the two equations above are equivalent.

By applying

$$\begin{cases} \dot{\boldsymbol{v}} = \ddot{\boldsymbol{u}} = M^{-1}\boldsymbol{f} \\ \dot{\boldsymbol{x}} = \boldsymbol{v} \end{cases},$$

where $\boldsymbol{f}$ denotes the resultant force, the second-order order equation in Eq. (3.10) can be transformed into a couple of first-order differential equations [Abell '14], which can be solved by a integration scheme discussed below.

3.3.2 Time Integration Schemes

3.3.2.1 Explicit Euler Time Integration

The simplest time integration method is the *explicit Euler integration* (also known as *forward Euler method*), with the time-stepping rules as

$$\begin{cases} \boldsymbol{v}_{t+1} = \boldsymbol{v}_t + h\dot{\boldsymbol{v}}_t \\ \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + h\boldsymbol{v}_t \end{cases}, \tag{3.11}$$

where h is the size of each time step, and $\dot{\boldsymbol{v}}_t = M^{-1}(\boldsymbol{f}_{ext} + \boldsymbol{f}_{int}(\boldsymbol{x}_t) - D\boldsymbol{v}_t)$.

This explicit integration scheme proceeds blindly into the future. With relatively large time steps, it would increase the system energy (i.e., *overshooting problem*) and finally cause an explosion/failure of simulation. Thus, the step size has to be sufficiently small to ensure numerical stability ([Fasshauer '07], Chapter 4).

3.3.2.2 Improved Explicit Integrators

To improve the stability of the explicit Euler method, a revised *symplectic Euler scheme* [Stern '06] can be applied, with the time-stepping rules as

$$\begin{cases} \boldsymbol{v}_{t+1} = \boldsymbol{v}_t + h\dot{\boldsymbol{v}}_t \\ \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + h\boldsymbol{v}_{t+1} \end{cases}. \tag{3.12}$$

There are also other more accurate and stable schemes with higher degree of approximation, such as second and fourth order *Runge-Kutta* integrators, and *Verlet* integration (refer to [Müller '08] for more details).

3.3.2.3 Implicit Integration Method

A popular implicit integration scheme is the *implicit Euler method* (also known as *backward Euler method*) with time-stepping rules as follows

$$\begin{cases} \boldsymbol{v}_{t+1} = \boldsymbol{v}_t + h\dot{\boldsymbol{v}}_{t+1} \\ \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + h\boldsymbol{v}_{t+1} \end{cases} \tag{3.13}$$

where $\dot{\boldsymbol{v}}_{t+1} = M^{-1}(\boldsymbol{f}_{ext} + \boldsymbol{f}_{int}(\boldsymbol{x}_{t+1}) - D\boldsymbol{v}_{t+1})$.

Other commonly used implicit schemes are the *Newmark* method and its variants [Wood '90], which have better energy conservation property. The implicit schemes are unconditionally stable. They lead to a large nonlinear system that is solved by *Newton's method* or some variation of Newton's method, which iteratively solves a large linear system of equations at each time step. Thus, it is much more computationally expensive than the explicit methods. Another issue is that it introduces numerical damping (i.e., energy dissipation) that dissipates energy (especially for the *backward Euler method*).

3.3.3 Variational/Optimization Implicit Euler

Another successful way of solving the dynamics problem is to recast the implicit integration of dynamics equations into a variational form, such that the time integration is as reformulated as an optimization problem (as discussed in **Section 2.3.2.1**).

For example, the system of differential equations in Eq. (3.13) are re-written as

$$M(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t - h\boldsymbol{v}_t) = h^2(\boldsymbol{f}_{ext} + \boldsymbol{f}_{int}(\boldsymbol{x}_{t+1})), \tag{3.14}$$

where the damping term is omitted for brevity here. This system can be re-formulated to an optimization problem

$$min_{\boldsymbol{x}_{t+1}} \left\{ \frac{1}{2h^2} \left\| M^{\frac{1}{2}}(\boldsymbol{x}_{t+1} - \boldsymbol{s}_n) \right\|_F^2 + W(\boldsymbol{x}_{t+1}) \right\}, \tag{3.15}$$

where $\boldsymbol{s}_n = \boldsymbol{x}_{t+1} + h\boldsymbol{v}_t + h^2 M^{-1}\boldsymbol{f}_{ext}$, and $\|\cdot\|_F$ denotes the Frobenius norm, W is the strain energy.

For an optimization problem, more robust optimization strategies [Nocedal '06] can be employed, which can be more efficient than the nonlinear root-finding problem.

3.4 Discussion and Conclusion

Physics simulation in computer graphics is a challenging task, which requires an integrated solution of linear algebra, differential equations, variational calculus (e.g., to understand the *Laplacian mechanics*), differential geometry, numerical methods, optimization, continuum mechanics, dynamics, etc.

For this PhD study, we address the important concepts in solid mechanics. First, we give a brief introduction of continuum mechanics in combination with the finite element methods, dynamics equations and numerical integrations, which form an essential basis for our further research. Constitutive models are formulated for isotropic materials. Based on these foundations, we can investigate more complex anisotropic material models and their formulations, as presented in our work of Chapter 4, Chapter 5, and Chapter 6.

Resorting to but not being strictly restricted by these physics principles, we also study geometrical techniques developed in computer graphics field, and combine them together to be better applied in graphics applications. Upon that, a physically based skeletal animation system is developed in Chapter 6.

Chapter 4. Incorporating Fiber Controls into FEM Model for Transversely Isotropic Materials

4.1 Introduction

In this chapter, we investigate the modeling of *transversely isotropic* materials. Transversely isotropic materials have symmetric mechanical properties about an axis that is normal to a plane of isotropy. They are commonly found in real-world objects with fibrous structures, such as plants and biological soft tissues. Simulating these deformable objects using transversely isotropic materials can generate more realistic behaviors than using isotropic ones.

In virtual surgery applications, transversely isotropic materials are often used for modeling soft tissues [Ogden '03]. *Picinbono et al.* [Picinbono '01] presented a linear elasticity model combined with a fiber-reinforced energy, and later they proposed a nonlinear elasticity model to better support large deformations [Picinbono '03]. *Irving et al.* [Irving '04] proposed an invertible FEM model to deal with degenerate and inverted elements during large deformations, which is applicable to transversely isotropic materials. *Teran et al.* [Teran '03] proposed to use *B-spline solids* for assigning fiber directions to each element of a simulated mesh, and a *finite volume method* was used to simulate muscles. In [Liu '12], a fiber-reinforced model was presented that some curves (as the fibers) are interactively embedded into a solid object, and the internal forces are then computed in terms of deformation energies of both the isotropic solid and the curves.

In the previous works, directional dependent behaviors of transversely isotropic material are achieved by introducing an additional energy incorporating material preferred directions. The total deformation energy is the sum of an *isotropic* strain energy and a fiber-reinforced strain energy. This kind of approaches can produce

physically-plausible results, but the additional energy term increases computational cost. Here we present a fiber-field incorporated corotational FEM model that works directly with the constitutive model of transversely isotropic materials. Furthermore, a smooth *fiber-field* is used to establish local frames for each element, which is used as references for defining material properties. This orientation information of each element is incorporated into the corotational FEM model, through adding a local transformation upon each elemental stiffness matrix.

Large deformations are supported, and physically-realistic deformations can be achieved. With pre-computation, it adds no computational cost on the existing corotational FEM model during simulation. Meanwhile, it provides a directable control of anisotropic behaviors, which is desirable for designing anisotropic deformable models.

4.2 Constitutive Model of Transversely Isotropic Materials

Let's start with the measure of deformation. For a solid object, the deformation gradient is defined as $F = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} \in \mathbb{R}^{3\times3}$, where $\boldsymbol{X} \in \mathbb{R}^3$ denotes the undeformed material point in the material space and $\boldsymbol{x} \in \mathbb{R}^3$ is the corresponding deformed point. The *small strain tensor* $\boldsymbol{\varepsilon} = (F^T + F)/2 - I$ is used in our work for its computational efficiency (refer to **Section 3.1.2**).

In continuum mechanics, mechanical behaviors of a material are defined by a *constitutive model* (refer to **Section 3.1.3**), which represents the relationship between stress and strain as $\boldsymbol{\sigma} = C\boldsymbol{\varepsilon}$, where $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are second-order stress and strain tensor respectively, C is a fourth-order tensor called *elastic stiffness* [Ting '96]. Due to the

symmetry property of the strain tensor, the relationship can be written in a contracted form as

$$\begin{Bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{31}\\ \sigma_{12}\end{Bmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ & C_{22} & C_{23} & 0 & 0 & 0 \\ & & C_{33} & 0 & 0 & 0 \\ & & & C_{44} & 0 & 0 \\ & sym. & & & C_{55} & 0 \\ & & & & & C_{66} \end{pmatrix} \begin{Bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ 2\varepsilon_{23}\\ 2\varepsilon_{31}\\ 2\varepsilon_{12}\end{Bmatrix}, \tag{4.1}$$

where it has

$$C_{11} = C_{22}, C_{13} = C_{23}, C_{55} = C_{66}, C_{44} = \frac{1}{2}(C_{11} - C_{12}),$$

with the third material axis as the axis of symmetry. Thus, there are totally *five* independent material parameters for a transversely isotropic material.

4.3 Fiber-Field incorporated FEM Model

In order to simulate a deformable model with transversely isotropic materials, we propose to incorporate a *fiber-field* that contains material orientation information into the corotational linear FEM (CLFEM) model [Müller '04]. Large deformations are allowed without visual artifact of distorted volumes.

Firstly a brief formulation of the CLFEM model is described. Then we explain how to incorporate the fiber orientation information with CLFEM. Finally we present the formulation for dynamics simulation using implicit time integration. Note that for clarity most of the formulations here are based on a single tetrahedral (TET) element, where the symbols of mechanical quantities are denoted with a superscript 'e'. A tetrahedral element is denoted by its nodal positions $\boldsymbol{X}^e = (\boldsymbol{X}_0^T, \boldsymbol{X}_1^T, \boldsymbol{X}_2^T, \boldsymbol{X}_3^T)^T \in \mathbb{R}^{12\times 1}$ where $\boldsymbol{X}_i \in \mathbb{R}^{3\times 1}$ in the initial configuration, and $\boldsymbol{x}^e = (\boldsymbol{x}_0^T, \boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \boldsymbol{x}_3^T)^T \in \mathbb{R}^{12\times 1}$ as a counterpart in the deformed configuration.

4.3.1 The CLFEM Model

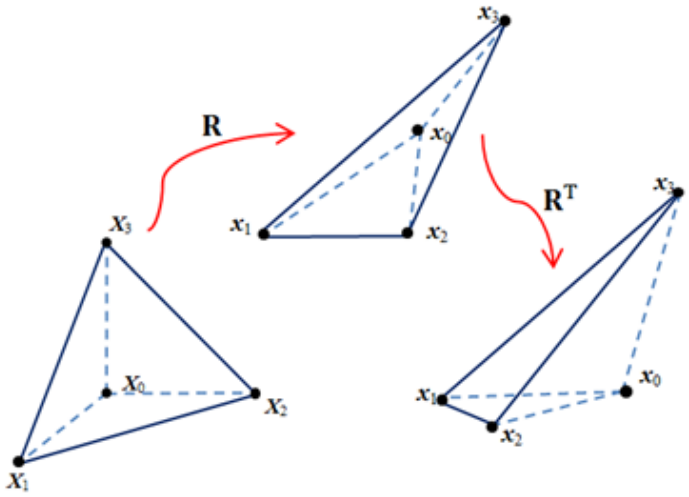


Fig. 4.1 The CLFEM model

The linear FEM model is only suitable for infinitesimal deformations (refer to **Section 3.1.2.2**), and obvious volume distortion artifact occurs when an object is under large deformations (as shown in Fig. 2.7 (b)). The CLFEM model improves the linear FEM model by removing the rotational effect of deformations. As shown in Fig. 4.1, $R \in \mathbb{R}^{3\times3}$ is a rotational transformation from the initial configuration to the deformed shape, then the elemental internal forces $\boldsymbol{f}_{int}^{e} \in \mathbb{R}^{12\times1}$ are computed as

$$\boldsymbol{f}_{int}^{e} = \begin{pmatrix} R & 0 & 0 & 0 \\ 0 & R & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{pmatrix} K^{e} \left(\begin{bmatrix} R^{T}\boldsymbol{x}_0 \\ R^{T}\boldsymbol{x}_1 \\ R^{T}\boldsymbol{x}_2 \\ R^{T}\boldsymbol{x}_3 \end{bmatrix} - \begin{bmatrix} \boldsymbol{X}_0 \\ \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \boldsymbol{X}_3 \end{bmatrix} \right) = R^{e}K^{e}\left(R^{eT}\boldsymbol{x}^{e} - \boldsymbol{X}^{e}\right), \qquad (4.2)$$

where $R^{e} \in \mathbb{R}^{12\times12}$, and $K^{e} \in \mathbb{R}^{12\times12}$ is a constant element stiffness matrix (refer to **Appendix A.1.3** for more computation details).

4.3.2 Fiber-Field Incorporated FEM Model

Now we present the mechanism for a fiber-field incorporated FEM model. Each TET element is assign with a direction vector, which forms a volumetric fiber-field for the

whole TET mesh. For deformable objects with internal fiber structures, it is these internal fiber structures that actually affect their transversely anisotropic behaviors. Based on this observation, we formulate an FEM model by utilizing orientation information of a fiber-field that is smoothly distributed inside an object.

Suppose a fiber-field, which consists of a preferred direction for each element, is given. An example is shown in Fig. 4.2 (c). Then a local orthonormal frame $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3\}$ where $\boldsymbol{q}_i \in \mathbb{R}^{3\times1}$, is established for each element: with one axis in accordance with the fiber orientation (here we use $\boldsymbol{q}_3$). Since material properties are symmetric about the $\boldsymbol{q}_3$ axis, the orientations of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are arbitrary and are only required to be on the plane perpendicular to $\boldsymbol{q}_3$. We define a local orientation matrix as $Q^e = (\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}) \in \mathbb{R}^{3\times3}$, and denote a quantity in this local frame by a hatted symbol ˆ, e.g., $\widehat{K^e}$ is denoted as a local element stiffness matrix. The matrix Q^e plays an essential role in our model, and it is extended to a 12×12 matrix like R^e and represented by a same notation for simplicity.

In our approach, instead of computing the element stiffness matrix in a global frame, which is the case for simulation of isotropic materials, we do computation in the local frames. The element internal forces and stiffness matrix are computed by the following procedure:

1) Firstly, the displacement vector $\widehat{\boldsymbol{u}^e} \in \mathbb{R}^{12\times1}$ in the local frame of an element is computed as

$$\widehat{\boldsymbol{u}^e} = Q^{eT}(R^{eT}\boldsymbol{x}^e - \boldsymbol{X}^e).$$

where $R^e, Q^e \in \mathbb{R}^{12\times12}$.

2) The local constant element stiffness matrix $\widehat{K^e}$ is computed as

$$\widehat{K^e} = \hat{B}^T C \hat{B}\, V^e,$$

where V^e is the volume of the element, and $\hat{B} \in \mathbb{R}^{6\times12}$ is the *strain-displacement matrix* (refer to **Appendix A.2.1.4**) that is also computed locally in terms of $\widehat{\boldsymbol{u}^e}$.

3) The local element internal forces are computed as

$$\widehat{\boldsymbol{f}^e}_{int} = \widehat{K^e}\widehat{\boldsymbol{u}^e}.$$

4) Finally, the element internal forces in the global frame are obtained, as

$$\boldsymbol{f}^e_{int} = R^e Q^e \widehat{\boldsymbol{f}^e}_{int} = R^e Q^e \widehat{K^e} Q^{eT}(R^{eT}\boldsymbol{x}^e - \boldsymbol{X}^e) = R^e K^e\left(R^{eT}\boldsymbol{x}^e - \boldsymbol{X}^e\right), \tag{4.3}$$

where $K^e = Q^e \widehat{K^e} Q^{eT}$ is the new element stiffness matrix of our model.

By *assembly of elements*, the matrices K^e of all the elements can be assembled to a global stiffness matrix $\bar{K} \in \mathbb{R}^{3n\times3n}$, where n is the total number of elements. Due to the fact that Q^e is pre-defined which is unchanged during simulation, $\bar{K}$ can also be pre-computed. Therefore, in our fiber-field incorporated model, no additional computational cost is introduced to the original CLFEM simulation.

4.4 Implicit Time Integration for Dynamics

For the simulation of dynamic deformation, the dynamics equations of motion are then given by a system of second-order ordinary differential equations (refer to **Section 3.3.1**):

$$M\ddot{\boldsymbol{u}} + D\dot{\boldsymbol{u}} + \boldsymbol{f}_{int} = \boldsymbol{f}_{ext}$$

For a tetrahedral mesh with n vertices, $\boldsymbol{u} \in \mathbb{R}^{3n\times1}$, $\dot{\boldsymbol{u}}$ and $\ddot{\boldsymbol{u}}$ are velocity and acceleration vectors respectively. $M \in \mathbb{R}^{3n\times3n}$ is the mass matrix, D the damping matrix, $\boldsymbol{f}_{int} \in \mathbb{R}^{3n\times1}$ the internal forces, and $\boldsymbol{f}_{ext} \in \mathbb{R}^{3n\times1}$ the applied external forces.

An *implicit backward Euler* integration scheme [Baraff '98] is used in our model, for the reason that it is stable and suitable for large time steps. The time-stepping rules are as follows:

$$\begin{aligned} \dot{\boldsymbol{u}}_{t+1} &= \dot{\boldsymbol{u}}_t + h\ddot{\boldsymbol{u}}_{t+1} \\ \boldsymbol{u}_{t+1} &= \boldsymbol{u}_t + h\dot{\boldsymbol{u}}_{t+1} \end{aligned} \tag{4.4}$$

where h is the size of each time step. At time $(t+1)$, we get

$$\begin{aligned} \dot{\boldsymbol{u}}_{t+1} &= \dot{\boldsymbol{u}}_t + hM^{-1}[\boldsymbol{f}_{ext} - D\dot{\boldsymbol{u}}_{t+1} - R\bar{K}(R^T\boldsymbol{x}_{t+1} - \boldsymbol{x}_0)] \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + h\dot{\boldsymbol{u}}_{t+1} \end{aligned} \tag{4.5}$$

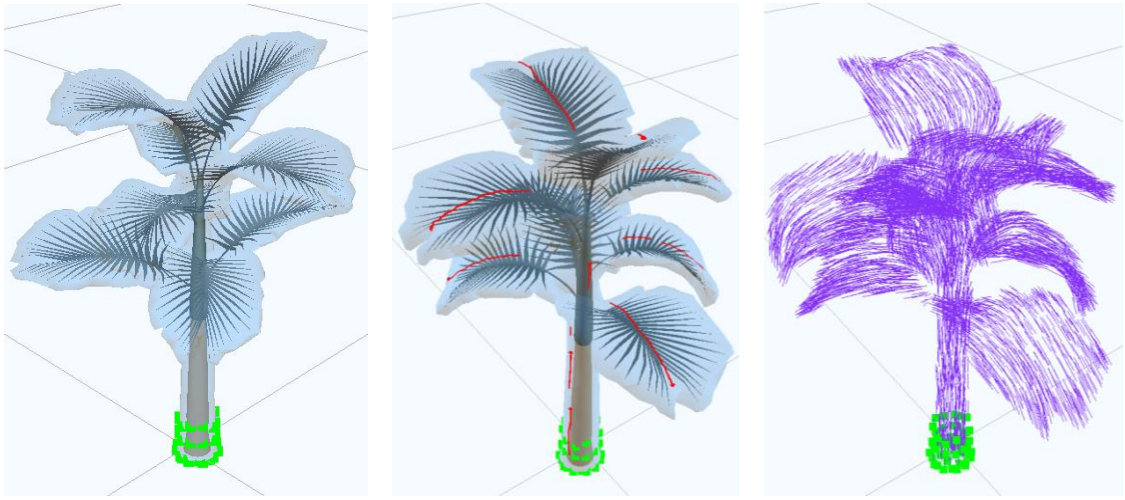
Finally we can get a linear system:

$$(M + hD + h^2 R\bar{K}R^T)\dot{\boldsymbol{u}}_{t+1} = M\dot{\boldsymbol{u}}_t + h(\boldsymbol{f}_{ext} - R\bar{K}R^T\boldsymbol{x}_t + R\bar{K}\boldsymbol{x}_0)\ . \tag{4.6}$$

By solving the linear system, the simulation state (velocity and position vectors) can be updated. Here, it shows the advantage of the CLFEM model that there is no need of re-computing the stiffness matrix at each step (unlike the nonlinear model, the tangent stiffness matrix has to be updated at each iteration in an iterative solver), but only to update the element rotation matrices.

4.5 Experiments and Assessments

To assess the effectiveness of our algorithm, in this section, we construct several dynamics simulations of deformable objects with transversely isotropic materials. Here, a tetrahedral mesh is used as a discretized representation of a solid body, with an embedded high resolution surface mesh for rendering purpose, as shown in Fig. 4.2 (a), the transparent mesh denotes a tetrahedral mesh, with a high resolution textured surface mesh embedded inside it. Note that the green dots represent nodes fixed in positions.



(a) (b) (c)

Fig. 4.2 (a) a palm tree model with a volumetric mesh and an embedded surface mesh; (b) User-defined strokes (*red*); (c) a fiber-field of all the tetrahedral elements

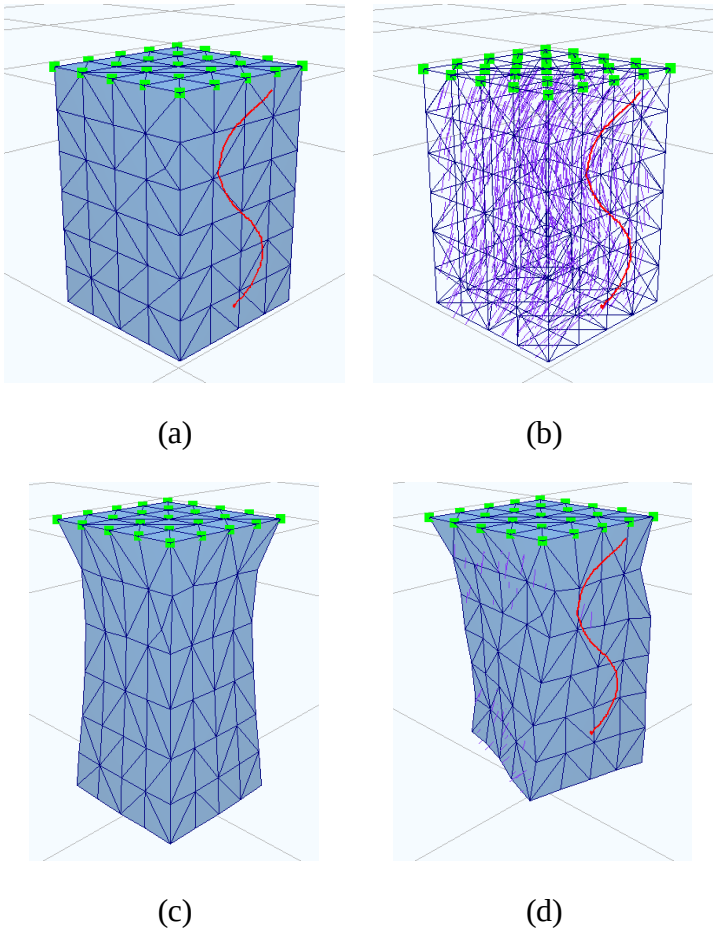
Without loss of generality, the fiber-field of a tetrahedral mesh is generated by a sketch-based interface (as in [TAKAYAMA '08, Ijiri '12]). The user is allowed to draw a few strokes on the surface or a sliced internal surface boundary of the tetrahedral mesh, to roughly define the fiber directions; as shown in Fig. 4.2 (b), the red lines are the strokes by the user. Then, a smooth interpolation is performed to automatically generate a smooth fiber orientation field on each nodes of the tetrahedral mesh. Eventually, a fiber orientation is generated for each tetrahedron element with *barycentric interpolation* of the orientations of its four nodes; as shown in Fig. 4.2 (c), the generated fiber-field vectors are rendered as purple line segments.

4.5.1 Impact of Fiber Field on the Elastic Stiffness

As in Eq. (4.1), five parameters, which represent the material's resistance to normal and shear deformations, can be used to define a transversely isotropic material. For comparison, we start with an isotropic material that has

$$C_{11} = C_{22} = C_{33} = \lambda + 2\mu\,, C_{12} = C_{13} = C_{23} = \lambda\,, C_{44} = C_{55} = C_{66} = \mu\,,$$

where λ and μ are *Lam écoefficients* (related to Young's modulus and Poisson's ratio). It can be changed according to Eq. (4.1) while keeping its positive-definiteness. Changing C_{33} to a larger value makes the material stiffer along the fiber orientation. Likewise, other parameters can also be changed to alter the material's resistance to normal and shear forces with respect to the local frame Q^e.



(a) (b)

(c) (d)

Fig. 4.3 Deformation of transversely isotropic material with a fiber-field
(a) drawing strokes on undeformed model; (b) element fiber-field;
(c) deformation under gravity(isotropic material);
(d) deformation under gravity (transversely isotropic material)

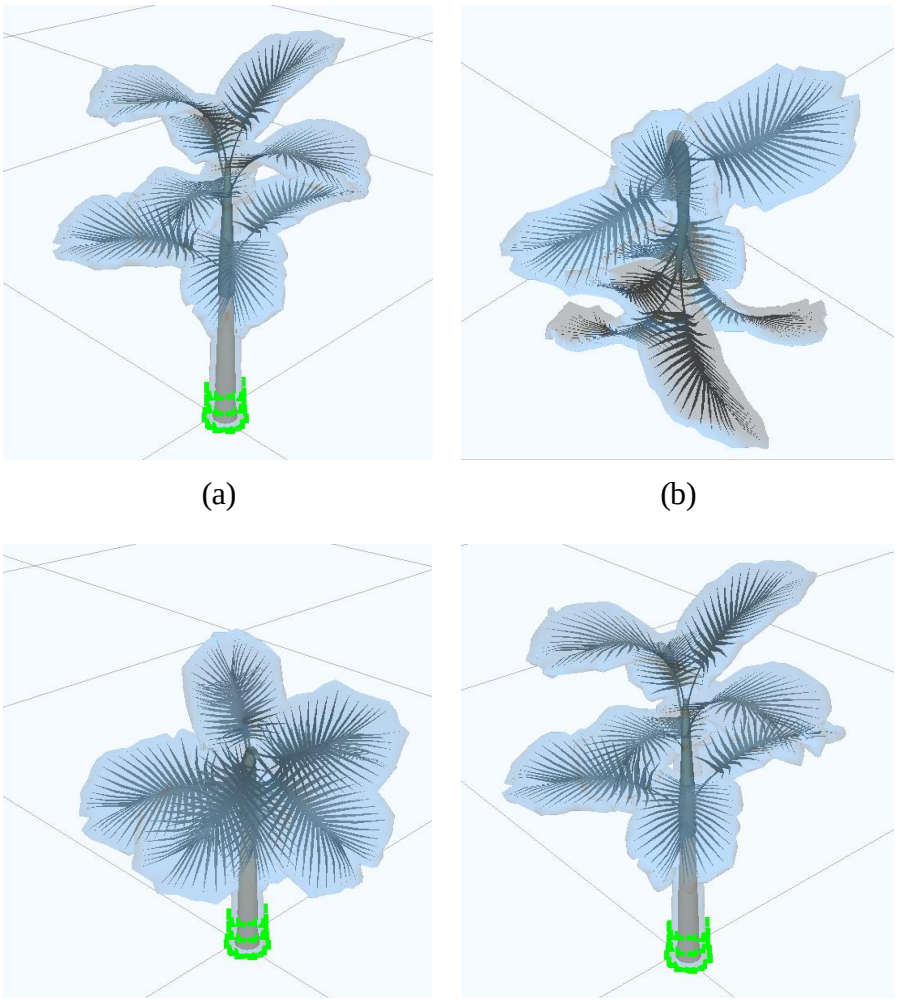
A comparison for a simple cubic model is shown in Fig. 4.3. Under gravity, the cubic model of isotropic material in Fig. 4.3 (c) deforms uniformly, while the transversely isotropic model in Fig. 4.3 (d) shows anisotropic behavior, which is in accordance with the fiber-field in Fig. 4.3 (b).

4.5.2 Fibers with Heterogeneous Materials

To demonstrate the capability of our fiber-field incorporated FEM model, a complex palm tree model made of heterogeneous materials is simulated, i.e., the trunk and leaves have different material properties. In Fig. 4.4, we show the deformations of the palm tree model under gravity. Fig. 4.4 (a) shows the original undeformed model. As a basis for comparisons, Fig. 4.4 (b) shows deformation of the tree made of an isotropic material. Being physically-plausible, it bends to the ground if the material is too soft (or, inflexible to move if the material is too stiff). Fig. 4.4 (c) shows the tree consisting of heterogeneous materials: it has stronger support from its trunk, preventing unnatural bending, but it is very difficult to obtain physically-plausible behavior for the leaves by tuning isotropic parameters. In nature, a leaf is much stiffer along its vein than in the other directions, and the tree also exhibits a stiffer material property along its trunk. Based on this observation, we should use a transversely isotropic material with a larger value of C_{33}. By using the fiber-field incorporated FEM model, Fig. 4.4 (d) shows a physically-plausible deformation of the palm tree: it becomes stiffer along the fibers while being flexible in the other directions. Fig. 4.5 shows the comparison of deformations under gravity and a dragging force. In contrast to the unnatural deformations of an isotropic model in Fig. 4.5 (a), the fiber-field incorporated model in Fig. 4.5 (b) exhibits strong stiffness along fiber directions that prevents over-bending of the trunk and over-stretching of the leaves.

Besides its convincing improvement in the visual results in physically-plausible deformation, the fiber-field incorporated FEM model only adds computational cost in the pre-computation step, thus achieving the same performance as the existing CLFEM model. In our experiments, both the palm models with and without fibers

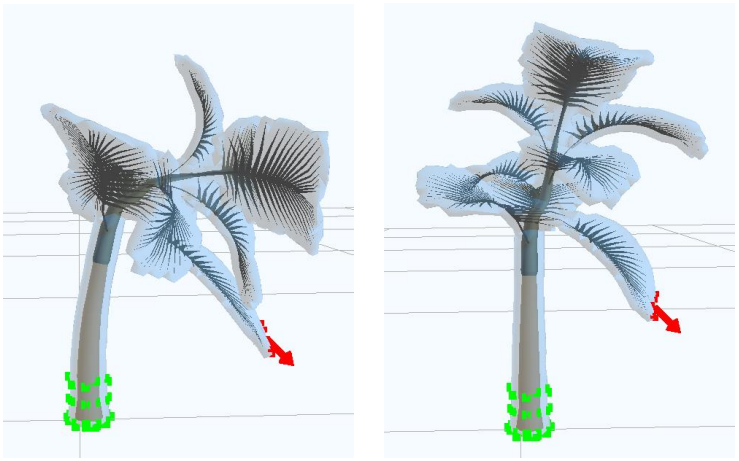
perform at 15 fps, with the tetrahedral mesh consisting of 5664 tetrahedra and 2064 nodes (including 16 fixed nodes).



(a) (b)

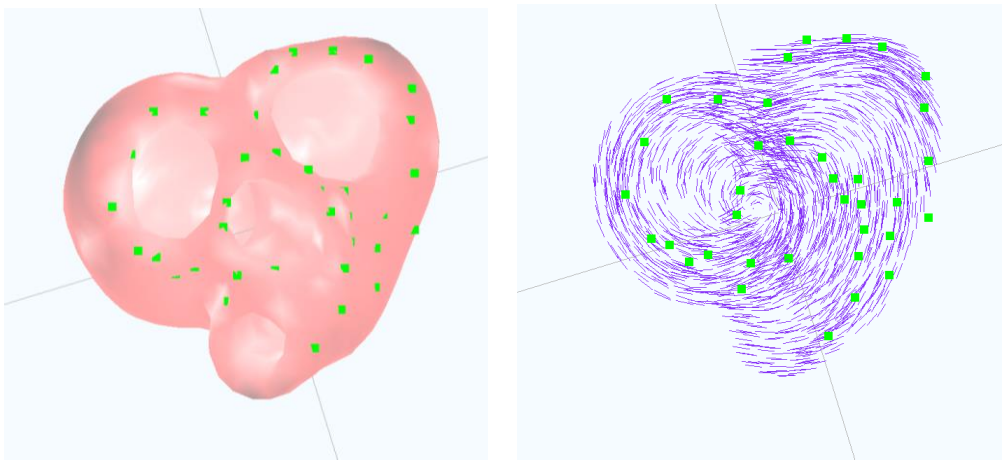
(c) (d)

Fig. 4.4 Deformations of different FEM models under gravity: (a) original shape; (b) homogeneous and isotropic material; (c) heterogeneous and isotropic materials; (d) fiber-field incorporated model, with heterogeneous and isotropic materials.



(a) (b)

Fig. 4.5 Deformations under a dragging force. (a) without fiber; (b) fiber-field incorporated model.



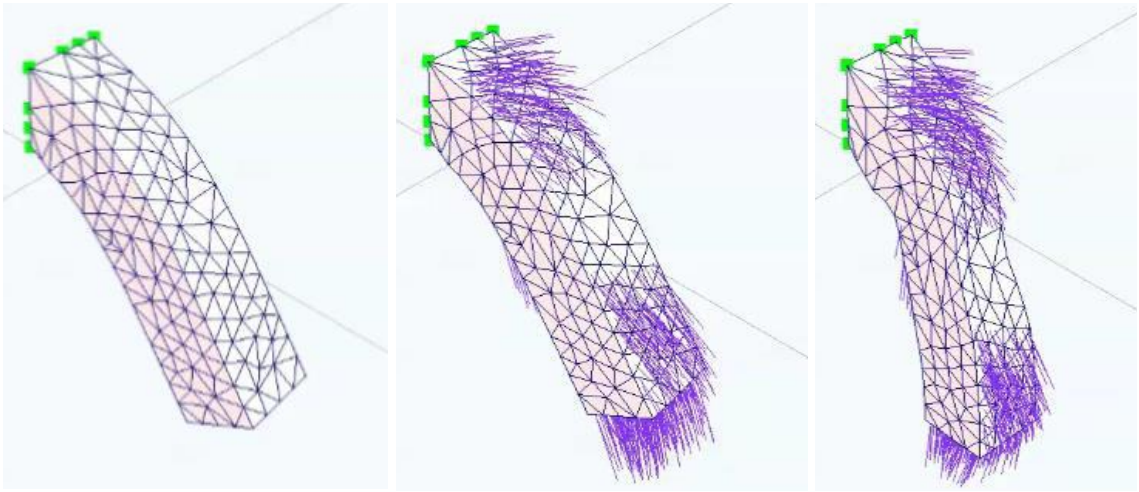
A heart model Fiber field

Fig. 4.6 A beating heart simulation using our fiber-field incorporated FEM model Conclusion (fiber-field generated by [TAKAYAMA '08])

Muscular tissue is a typical example of transversely isotropic materials, whose mechanical response is controlled by the internal muscle fibers. For example, the beating movement of the heart is controlled by the regular periodic contraction of the myocardium. Using the fiber-guided model, we can simulate the complex movements of the heart beating. To further explore the application of our fiber-field incorporated

FEM model for more complex topological structures, we conduct an investigation with a heart which possesses two chambers and contracting muscles. A screenshot of the beating heart is shown at left in Fig. 4.6 (a), and its corresponding fiber-field vectors are displayed in Fig. 4.6 (b).

4.5.3 Validation



(a) (b) (c)

Fig. 4.7 Validation of the correctness of the fiber-field incorporated models. (a) CLFEM for isotropic material; (b) Fiber incorporation for isotropic material; (c) Fiber incorporation for anisotropic material

We show that isotropic materials can be dealt with as a special case by our *fiber-field incorporated model*; the same deformations by the existing *isotropic FEM model* can be reproduced by our model. In Fig. 4.7(a), the deformed result of an isotropic rectangular bar under gravity is represented by the tetrahedral mesh computed by the existing CLFEM model. In Fig. 4.7(b), the fiber field is generated for the isotropic material by our new model, which preserves the same deformation of the tetrahedral mesh. In contrast, we can redefine the fiber in the same FEM model for an anisotropic material to view its effect. Fig. 4.7(c) shows the deformed results of a transversely isotropic material under gravity (with smaller stretching resistance along the fiber

directions), which clearly illustrate the different fiber field from that in Figure 1b and the different tetrahedral mesh from those in Fig. 4.7 (a) and (b).

Quantitative Comparison For isotropic materials, mechanical response is independent of directions in the material space. Thus theoretically, the local frame transformations in our model will not affect its behaviors and should produce exactly the same results, i.e. displacement field $\boldsymbol{u}$. To validate this, we give a quantitative comparison of the two models in Fig. 4.7 (a) and (b). The bar model in Fig. 4.7 (a) has 311 vertices (1039 elements), with 20 fixed vertices; that is, the degree of freedom is 873. We compare the displacement field and of the two simulations at equilibrium, using the square difference formula $diff \coloneqq \sum_i (\boldsymbol{u}_{1i} - \boldsymbol{u}_{2i})^2$. The computed difference is 3.95e-012, resulting in the same $\boldsymbol{u}_1$and $\boldsymbol{u}_2$. Thus, we have quantitatively validated the correctness of our fiber-field incorporated model for the special case of isotropic material.

4.6 Conclusion

In previous work, Physically-plausible deformable models have been developed mainly for isotropic materials. We have presented a novel fiber-field incorporated FEM model to deal with transversely isotropic anisotropic materials. The key idea is to use a fiber-field to establish local element coordinate frames, and to formulate a corotational linear FEM model based on local frames. For simulation in virtual surgery, our fiber-field incorporated models can yield realistic deformations. The main advantages of our model are:

i. Rather than computing additional strain energy of embedded fibers as in the conventional fiber-reinforced models, we only use the orientation information to form a fiber field.

ii. The fiber-field incorporated model is based on corotational linear FEM model, which is more computational efficient than nonlinear FEM models, and large deformations can be accommodated.

iii. Additional computation in the proposed model can be done by pre-computation, thus no additional cost is added during simulation.

A more flexible user interface for fiber generation and material parameter adjustment tools are yet to develop. In specific applications, this orientation field can be generated from Diffusion Tensor MRI (DTMRI) techniques when muscular tissues are acquired (e.g. [Rohmer '07] [Sosnovik '09]) for more accurate mechanical and physiological analysis. In the future, this idea can be further extended to simulate other kinds of anisotropy, such as orthotropic materials, given a local frame which can represent different material plane symmetries.

Acknowledgement The sketch-based interface is implemented with reference to *Kenshi's* paper [TAKAYAMA '08].

Chapter 5. Dynamics Simulation of Orthotropic Deformable Models

5.1 Introduction

In the previous chapter, we have studied transversely isotropic materials and developed a *fiber-field incorporated FEM model*. Here, we further investigate orthotropic materials.

As a special class of anisotropic materials, orthotropic materials exhibit different mechanical behaviors along three orthogonal directions. They have more degrees-of-freedom with material properties than transversely isotropic materials, and can generate more complex mechanical behaviors. Therefore, it requires careful adjustment of material parameters to ensure numerical stability.

Most of the existing physically based deformable models in the literature are mainly based on the assumption of isotropic materials, such as the *St.Venant-Kirchhoff* material and the *Neo-Hookean* material. In terms of anisotropy, transversely isotropic materials are often discussed in existing work, and a review can be found in the previous chapter. To date, modeling methods for simulation of deformable objects with orthotropic materials are little discussed; and there is a lack of intuition in designing orthotropic models in graphics applications.

We proposed a solution here for modeling and deformation control of orthotropic deformable objects.

- Firstly, constraints of material properties with respect to the strain energy density are analyzed, and a positive-definite elasticity tensor is derived for an orthotropic material.
- Secondly, an orthotropic deformation controlling frame-field is conceptualized and a frame construction tool is developed for users to define the desired material

properties. Rotation minimizing frames along with several user-defined NURBS curves are propagated into the entire body of the deformable object, forming a frame-field. A quaternion Laplacian smoothing algorithm is developed for generating such a smooth frame-field.

- Thirdly, the corotational linear FEM (CLFEM) model coupled with the orthonormal frame-field is formulated to realize a dynamics simulation system, which is computationally efficient and supports large deformations.

All the algorithms have been implemented in a modeling and simulation system, and a GUI is provided to design the orthotropic model. Experiments on real-time dynamics simulation and analytical comparisons are presented.

5.2 Related Work

In terms of anisotropy, transversely isotropic materials are often researched in computer graphics area, as discussed in the last chapter. However, orthotropic models are less discussed. *Li et al.* [Li '14b] proposed a stable way to tune the complicated material properties for anisotropic materials. There is lack of approaches for intuitive deformation controls.

We aim at a more effective deformation control method. In previous research, *example-based* methods and *space-time control* methods have been proposed to provide users with intuitive art-directed control of complex anisotropic behaviors; a review of these approaches can be found in **Chapter 2**. However, these deformation controls achieve anisotropy by adding artificial forces that violate physics rules.

Inspired by our first work that incorporates a *fiber-field* for transversely isotropic materials, we propose a *frame-field* incorporated deformation control here. We provide a novel interactive tool to generate a frame-field for orthotropic materials, which is

coupled with a corotational FEM model. Our method can control the orthotropic behaviors and make the deformations more predictable.

5.3 Computational Model of Orthotropic Materials

5.3.1 Elasticity Tensor of Orthotropic Materials

In contrast with a transversely isotropic material that has preferred material directions (as described previously in **Section 4.2**), an orthotropic material has three mutually orthogonal planes of rotational symmetry. It exhibits different stiffness along these three orthogonal directions, and has *nine* independent parameters. The symmetric elasticity tensor C is defined as

$$C = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}, \tag{5.1}$$

where

$$A = \gamma \begin{bmatrix} E_1(1-\nu_{23}\nu_{32}) & E_1(\nu_{21}+\nu_{23}\nu_{31}) & E_1(\nu_{31}+\nu_{21}\nu_{32}) \\ E_2(\nu_{12}+\nu_{13}\nu_{32}) & E_2(1-\nu_{13}\nu_{31}) & E_2(\nu_{32}+\nu_{12}\nu_{31}) \\ E_3(\nu_{13}+\nu_{12}\nu_{23}) & E_3(\nu_{23}+\nu_{13}\nu_{21}) & E_3(1-\nu_{12}\nu_{21}) \end{bmatrix},$$

$$B = \begin{bmatrix} \mu_{23} & 0 & 0 \\ 0 & \mu_{31} & 0 \\ 0 & 0 & \mu_{12} \end{bmatrix},$$

$\gamma = \frac{1}{1-\nu_{12}\nu_{21}-\nu_{23}\nu_{32}-\nu_{31}\nu_{13}-2\nu_{21}\nu_{32}\nu_{13}}$, and we have $\nu_{21}\nu_{32}\nu_{13} = \nu_{12}\nu_{23}\nu_{31}$.

From the above, we need to determine 9 independent parameters:

- 3 Young's moduli E_1, E_2, E_3, along three orthogonal principal axes respectively;
- 6 Poisson's ratios $\nu_{ij}, (i \neq j)$, satisfying that $\nu_{ij}/E_i = \nu_{ji}/E_j$, so only 3 independent degrees-of-freedom remain;
- 3 shear moduli $\mu_{12}, \mu_{23}, \mu_{31}$ on three principal planes.

The inverse elasticity tensor, $S = C^{-1}$, can be derived and be represented by a relatively simple form as follows

$$S = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\mu_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\mu_{31}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\mu_{12}} \end{bmatrix},$$

which is in a simpler form than that of C, and can be used for the positive-definiteness analysis in the following section.

5.3.2 Computation for Strain Energy Density

The strain energy density is defined by

$$\psi(\boldsymbol{\varepsilon}) = \frac{1}{2}\boldsymbol{\sigma}^T\boldsymbol{\varepsilon} = \frac{1}{2}\boldsymbol{\varepsilon}^T C\boldsymbol{\varepsilon} = \frac{1}{2}\boldsymbol{\sigma}^T S\boldsymbol{\sigma}, \tag{5.2}$$

Here ψ should be made a positive-definite function of $\boldsymbol{\varepsilon}$ (or, $\boldsymbol{\sigma}$); it is equivalent to require that C (or S, equivalently) be positive-definite. According to the *Sylvester's criterion* that a real symmetric matrix is positive-definite if all its principal minors are positive, parameters of an orthotropic material should satisfy the following conditions:

$$\begin{gathered} E_1, E_2, E_3 > 0, \\ \mu_{12}, \mu_{23}, \mu_{31} > 0, \\ \nu_{12}\nu_{21} < 1, \nu_{23}\nu_{32} < 1, \nu_{13}\nu_{31} < 1, \gamma > 0. \end{gathered} \tag{5.3}$$

Li et al. [Li '14b] proposed a strategy to reduce the number of material parameters to 4. Although the simplified set cannot express the entire family of orthotropic materials, it helps animators stably tune the material properties.

However, the previous work lacks a spatial definition of principal axes for an orthotropic model. *Li et al.* [Li '14b] considered directional derivatives of a 3D texture map, but this suggests the use of a complex tool for designing 3D texture maps. To solve this problem, we propose a user-guided tool for defining a spatially varying *frame-field* in the following section.

5.4 Model Control with Spatially Varying Frame-field

As discussed above, orthotropic materials exhibit different stiffnesses along different directions, and the material parameters are specified in a frame of principal axes. Therefore, for the simulated mesh (tetrahedral mesh in our study) of a deformable model, a local frame needs to be assigned for each element, which forms a spatially varying *frame-field*.

Our goal here is to design an interactive and intuitive tool to generate such a frame-field that helps to specify material properties and control the orthotropic deformation behaviors in a more predictable way. A relevant work by *Choi et al.* [Choi '13] proposed a numerical representation of fascicle trajectories of skeletal muscle, using the gradient of a smooth scalar field generated by heat diffusion. The scalar field is obtained by solving a Laplacian equation using the finite volume method. Our work in the previous chapter proposed an FEM model augmented with a smooth vector-field (i.e., fiber-field) to simulate transversely isotropic materials. Similar in spirit, however, here an *orthonormal* frame-field is formed for orthotropic models.

The main idea of our proposed approach is that:

1) Firstly, the user forms some smooth curves upon the TET mesh, by only putting control points on the mesh surface. The NURBS curves are generated according to the control points, which indicate a fibrous structure of the shape, and a series of orthonormal frames are generated simultaneously along the curves, forming a sparse distribution of frames.

2) Then, a frame-field is automatically generated by a smooth interpolation of the sparse frames, which consists of local frames for all the TET elements.

3) Finally, this frame-field is augmented with the corotational FEM model to formulate an orthotropic model.

Detailed algorithms with respect to the three steps are presented separately in this section.

5.4.1 Rotation Minimizing Frames as the Indication of Material Principal Axes

Let $\boldsymbol{x}\ (u) = (x(u), y(u), z(u))$ be a C^1 regular curve in 3D Euclidean space $\mathbb{E}^3$, its unit tangent vector can be computed as $(u) = \frac{\boldsymbol{x}'}{\|\boldsymbol{x}'\|}$, where $\boldsymbol{x}' = \frac{d\boldsymbol{x}}{du}$. Then a *moving frame* associated with the curve can be defined as an orthonormal system composed of a tangent vector and two orthogonal vectors on a normal plane at the current position.

The *Frenet* frame $(\boldsymbol{t}(u), \boldsymbol{n}(u), \boldsymbol{b}(u))$ is a familiar example from differential geometry, where $\boldsymbol{b} = (\boldsymbol{x}' \times \boldsymbol{x}'')/\|\boldsymbol{x}' \times \boldsymbol{x}''\|$ and $= \boldsymbol{b} \times \boldsymbol{t}$. However, the Frenet frame is not a suitable choice here, due to its *'unnecessary rotation'* in the curve normal plane and lack of controllability.

With convenient and intuitive control as our first concern, we employ the NURBS curves and associated *rotation minimizing frames*, to generate a sparse distribution of orthonormal frames, which will be further used for interpolation.

5.4.1.1 A Recap of Rotation Minimizing Frames

A *rotation minimizing frame* (RMF) [Bloomenthal '88] does not rotate about the instantaneous tangent of the curve, meaning that among all moving frames on a space curve the RMF yields least elastic energy associated with twisting.

A definition by differential equations is as follows: given a C^1 curve $\boldsymbol{x}(u) \in \mathbb{E}^3$ and its unit tangent vector $\boldsymbol{t}(u)$, a moving frame $(\boldsymbol{r}(u), \boldsymbol{s}(u), \boldsymbol{t}(u))$, where $\boldsymbol{r}(u) \times \boldsymbol{s}(u) = \boldsymbol{t}(u)$, is called a *rotation minimizing frame* if $\boldsymbol{r}(u)$ satisfies the following system of differential algebraic equations (DAE) [Wang '08]:

$$\begin{cases} \boldsymbol{r}'(u) - \phi(u)\, \boldsymbol{t}(u) = \boldsymbol{0} \\ \boldsymbol{r}(u) \cdot \boldsymbol{t}(u) = 0 \end{cases},$$

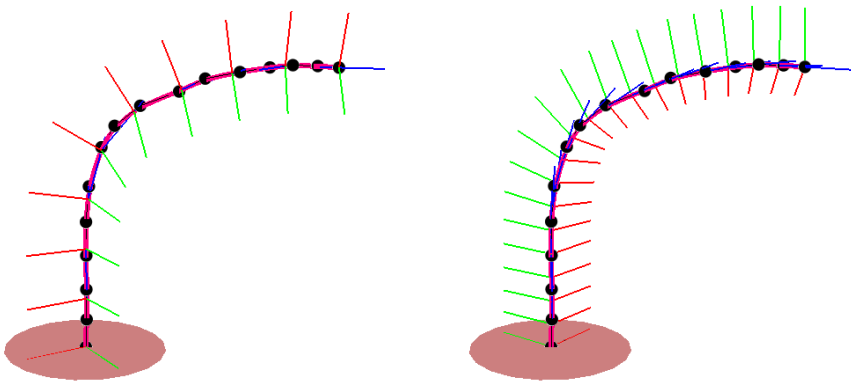
where $\phi(u)$ is some scalar function. Such a vector function $\boldsymbol{r}(u)$ perpendicular to the tangent vector exhibits minimal rotation, it is thus called a *rotation minimizing vector*. $\boldsymbol{r}(u)$ is not necessarily differentiable for a C^1 curve, thus an adapted *weak form* of the DAE can be defined as

$$\begin{cases} \boldsymbol{r}'(u) - \int_0^u \phi(v)\, \boldsymbol{t}(u) dv = \boldsymbol{0} \\ \boldsymbol{r}(u) \cdot \boldsymbol{t}(u) = 0 \end{cases}.$$

5.4.1.2 RMFs Associated with NURBS curves

RMFs are commonly used in sweep surface modeling and path planning of animations. However, here we choose the RMFs associated with NURBS curves as the constraints for generating material principal axes, which to some extent can be considered as an indication of the material distribution of an orthotropic model: tangent direction of the

curve is represented as one principal axis, and thus a corresponding RMF is denoted as a local material frame.



(a) (b)

Fig. 5.1 The orientation of the RMFs can be adjusted by changing the orientation of the first frame. The 2D-disk, which is orthogonal to the blue axis of the first frame and contains the other two axes (red and green), is an intuitive display of the orientation of the first frame.

An example is shown in Fig. 5.1. A third-degree NURBS curve is chosen here for its smoothness and controllability. Note that only the two end-nodes of the control points are interpolated and the in-between ones are extrapolated. This can be useful for tolerance of bumpy surfaces. The NURBS curves intuitively imply the main structures of the orthotropic object, which is similar to shape medial-lines or fiber direction of a fibrous object.

We choose the RMF for its nice properties that are desirable features in design, as follows:

1) The RMF is stable, unlike the Frenet frame that leads to undesirable twists in motion along a general curve; and its *minimal twist property* reflects the natural structure of an object, e.g., fibers of muscle or a tree.

2) The RMF is easy to control. The user can simply manipulate the orientation of the RMFs by changing the orientation of the first frame, causing the following frames to be changed accordingly (due to the minimal twist property), as shown in Fig. 5.1. The number of frames generated can be set on-the-fly as well.

Exact computation of the RMFs is either impractical or computationally expensive. The *double reflection* method based on the discrete approximation proposed by *Wang et al.* [Wang '08] is used here. It is stable, efficient and accurate (with fourth order global approximation error). All the design and manipulations can be done at an interactive rate.

5.4.2 Laplacian Smoothing of the RMFs

Our goal now is to generate a frame-field (e.g., as in Fig. 5.3) for a TET mesh with the obtained sparsely distributed RMFs (as in Fig. 5.2).

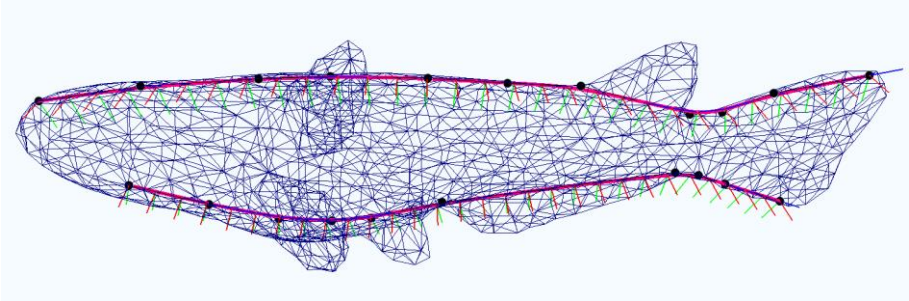


Fig. 5.2 The user plots some control points (the *black* dots) on the surface of the simulated mesh, then the NURBS curves (in *purple*) and associated RMFs are automatically generated (the *RGB* colored orthotropic frames).

Given a global coordinate system, orthogonal frame of the RMFs can be directly represented by a 3×3 matrix,

$$Q_i = (\boldsymbol{r}_{i1}, \boldsymbol{r}_{i2}, \boldsymbol{r}_{i3}), i = 0, .., m, \tag{5.4}$$

where m is the total number of the RMFs, $(\boldsymbol{r}_{i1}, \boldsymbol{r}_{i2}, \boldsymbol{r}_{i3})$ are the coordinate-vectors of the three axes respectively (as column vectors). Q_i^T can be considered as a rotation matrix that transforms a vector from the global frame to a local frame. Therefore, the task here turns into a problem of interpolation of rotations, which is commonly used in animation and skinning methods.

An early and simple solution is the *linear blend skinning* (LBS) [Magnenat-Thalmann '88]. It essentially does a linear element-wise interpolation of rotation matrices that is computationally efficient; however, it leads to non-rotations and thus suffers volume-collapse artifacts. The *spherical linear interpolation* (SLERP) [Shoemake '85] works well for blending two rotations, using quaternions to represent rotations, but it cannot be directly applied in a scenario involving more than two rotations. A fast approximate *quaternion linear blending* (QLERP) [Kavan '05] uses a simple linear quaternion averaging of multiple quaternions and re-normalization. It offers reasonable skinning results with low time and memory complexity. There are also advanced skinning methods, such as *dual-quaternion blending* [Kavan '06], which blend rigid transformations including not only rotations but also translations, and achieves better skinning results than the former methods. With only rotations involved in our task, an essentially linear interpolation scheme of quaternions is employed here for the generation of the desired frame-field. An advantage of this linear combination is that it is computationally efficient, which is important for interactive design.

Instead of using a blending scheme as in skinning, a Laplacian smoothing method ([Sorkine '04] [Takayama '07]) is exploited here. A big advantage of this approach is that a complicated weight function (such as *radial basis function*, *biharmonic weights* [Lipman '10].) or a labor-intensive weight-painting is not required to define vertex weights, which is a prerequisite for the previous method. Weights are defined based on

Euclidean distances between neighboring elements. With an intuitive geometric meaning, this method relieves a user of a laborious task. It is computationally efficient, producing results sufficient for the task here.

5.4.2.1 Formulation of Quaternion Laplacian

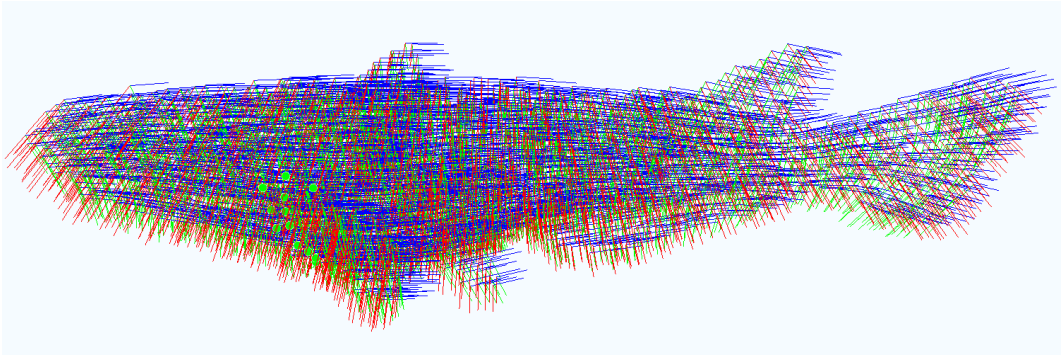


Fig. 5.3 The frame-field is generated by Laplacian smoothing of the RMFs associated with the NURBS curves.

Quaternions are used as representations for local frames. Each frame Q_i is firstly converted to a quaternion representation $\boldsymbol{q}_i = (\theta_i, x_i, y_i, z_i)$. The Laplacian smoothing is performed on the TET elements for the interpolation of the quaternions, with the RMFs as the constraints. We do the Laplacian smoothing on each θ-, x-, y-, z-component respectively, and finally obtain the frame-field by conversion of the resulting quaternions. Here we formulate the Laplacian method via the θ-component, and perform the same processing on the other three components (x, y, and z).

The ***Laplacian*** based on the quaternion component is defined as

$$\delta_i = \sum_{j \in N_{r_i}} \omega_j^i \, \theta_j - \theta_i, \qquad i = 1, .., n, \tag{5.5}$$

where n is the total number of elements, and N_{r_i} is the index set of the neighboring elements of the i-th element. Here, we choose the face-sharing neighbors, which form

a smaller set than the vertex-sharing neighbors and thus need less computation. ω_j^i represents the weight defined as

$$\omega_j^i = \frac{exp(-d_{ij}^2)}{\sum_{k \in N_{r_i}} exp(-d_{ik}^2)}, \tag{5.6}$$

where d_{ij} is the Euclidean distance between two neighboring elements (here we use the distance between their centroids).

The constraints of the Laplacian smoothing are set for the elements near the RMFs, as

$$\sum_{j \in \overline{Nr_i}} \bar{\omega}_j^i \theta_j = \bar{\theta}_i, \qquad i = 1, .., m, \tag{5.7}$$

where m is the total number of RMF frames, $\bar{\theta}_i$ is the known component of the $i\text{-}th$ quaternion (i.e., corresponding to the $i\text{-}th$ RMF frame), and $\overline{N_{r_i}}$ is the index set of the elements close to the position of the $i\text{-}th$ frame, and the weight $\bar{\omega}_j^i$ is defined similarly to Eq. (5.6), as

$$\sum_{j \in \overline{N_{r_i}}} \bar{\omega}_j^i \theta_j = \bar{\theta}_i, \qquad i = 1, .., m, \tag{5.8}$$

where $\bar{d}_{ij}$ is the Euclidean distance between the centroid of an element and the origin of a frame.

This Laplacian smoothing problem, with the definition of Laplacian in Eq. (5.5) and the constraint in Eq. (5.7), can be formulated in a matrix form respectively, as

$$\begin{aligned} &\boldsymbol{\delta} = L\boldsymbol{\theta}, \qquad \boldsymbol{\delta} = (\delta_1, .., \delta_n)^T, \boldsymbol{\theta} = (\theta_1, .., \theta_n)^T, \\ &\bar{C}\boldsymbol{\theta} = \overline{\boldsymbol{\theta}}, \qquad \overline{\boldsymbol{\theta}} = (\bar{\theta}_1, .., \bar{\theta}_m)^T, \end{aligned} \tag{5.9}$$

where $L_{ij} = \begin{cases} -1, & j = i \\ \omega_j^i, & j \in Nr_i \\ 0, & otherwise \end{cases}$ and $\bar{C}_{ij} = \begin{cases} \bar{\omega}_j^i, & j \in \bar{N}r_i \\ 0, & otherwise \end{cases}$.

Thus, $L \in \mathbb{R}^{n \times n}$ and $\bar{C} \in \mathbb{R}^{m \times n}$, where usually $m << n$, i.e., the constrained elements are a small set of the TET mesh.

5.4.2.2 Solving a Minimization Problem

The Laplacian smoothing process can be formulated as a minimization problem,

$$\min_{\boldsymbol{\theta}} \left\| \begin{pmatrix} L \\ \bar{C} \end{pmatrix} \boldsymbol{\theta} - \begin{pmatrix} \mathbf{0} \\ \bar{\boldsymbol{\theta}} \end{pmatrix} \right\|_2^2, \tag{5.10}$$

where $\|\bullet\|$ is the L-2 norm. This formulation implies that the quaternion-components of an element should be a linear combination of those of its neighbors, and some of them are constrained by the user-defined RMFs. With the above formulation, both of the conditions are satisfied in a least square sense. Solving the quadratic minimization problem in Eq. (5.10) is equivalent to solving a sparse linear system:

$$A^T A\boldsymbol{\theta} = A^T \boldsymbol{b}, \tag{5.11}$$

where

$$A = \begin{pmatrix} L \\ \bar{C} \end{pmatrix} and\ \boldsymbol{b} = \begin{pmatrix} \mathbf{0} \\ \bar{\boldsymbol{\theta}} \end{pmatrix}.$$

Furthermore, we have the $n \times n$ sparse symmetric system matrix and the right hand side in (5.11) are computed as

$$A^T A = L^T L + \bar{C}^T \bar{C},$$

$$A^T \boldsymbol{b} = \bar{C}^T \bar{\boldsymbol{\theta}}.$$

Here, $L^T L$ is a constant matrix that can be pre-computed.

The Laplacian smoothing is done on the four quaternion-components separately, meaning that four linear equations with the same system matrix are to be solved.

Therefore, a direct linear system solver, which performs LU factorization of the system matrix only once and then solves the four equations independently, would be a good choice here. The UMFPACK library [Davis '04] is used here. The direct solver performs very fast due to pre-factorization of the system matrix. It takes much more time to prepare the sparse matrices L and $\bar{C}$, involving neighbors search, weights computation and sparse matrices construction, which is the performance bottle-neck in our unoptimized implementation.

The above Laplacian smoothing performs an element-wise method, i.e., quaternions are computed for the elements directly. An alternative is to compute the quaternion at each vertex, and then obtain the element quaternions by barycentric interpolation. The vertex-wise approach usually reduces computation, since in most cases the number of vertices is smaller than that of the elements; for example, the TET mesh of the raptor model in Fig. 5.5 contains 2996 vertices and 8148 elements. For the raptor model, the vertex-wise method takes about $367ms$ to prepare the matrices and $32ms$ to solve all the linear equations, while the element-wise one takes about $1665ms$ and $52ms$ respectively. An effective way to improve performance is to pre-compute $L^T L$ and only to update the $\bar{C}$ related computation on-the-fly.

In conclusion, the RMF generation involving user interaction can be done in real-time, and for a TET mesh with tens of thousands of elements the frame-field generation takes a few seconds, which is acceptable as a data preparation phase in the simulation system; and the generated frame-field can be stored in an off-line file.

5.4.3 Simulation of Orthotropic Deformable Models

The corotational FEM (CLFEM) model (as described in **Section 4.3**) is also used here to support large deformations, and augment it with a frame-field to deliver orthotropic

behaviors. Similar to the idea of fiber-field incorporated model in the last chapter, we augment the CLFEM with the frame-field, in a way that the internal force is computed in the local frames instead of the single global frame, as

$$\boldsymbol{f}_{int}^{e} = R^{e}Q^{e}\,\widehat{\boldsymbol{f}^{e}}_{int} = R^{e}Q^{e}\widehat{K^{e}}Q^{eT}(R^{eT}\boldsymbol{x} - \boldsymbol{X}) = R^{e}K^{e}\left(R^{eT}\boldsymbol{x} - \boldsymbol{X}\right),$$

The element stiffness matrix becomes $\mathrm{K}^{e} = \mathrm{Q}^{e}\mathrm{K}^{e}\mathrm{Q}^{e\mathrm{T}}$. Through cumbersome algebra, K^{e} here is essentially equivalent to the stiffness matrix used in the paper [Li '14b] (or, refer to [Bower '09] for more details). However, here we present and compute it in a more easily comprehensive way. Due to the fact that Q^{e} is pre-computed, K^{e} can also be pre-computed. Therefore, in our frame-field augmented approach, no additional computational cost is introduced into the CLFEM simulation.

For the dynamics simulation, the equations of motion are solved by an implicit backward Euler integration with a conjugate gradient solver.

5.5 Experiments and Discussions

In this section, we validate the effectiveness of our proposed modeling approach using some typical models, and compare the deformations of isotropic and orthotropic materials. The same external loads including gravity and directional pulling force are applied to both the isotropic and the orthotropic models.

5.5.1 Orthotropic FEM Dynamics

In the following screenshots, the tetrahedral meshes are rendered in wire-frame mode to reveal the model structures, and a surface mesh of high resolution, which is deformed by the tetrahedral mesh by interpolation, is embedded for better visual effects in the final renderings. The green dots denote the constraint vertices that are

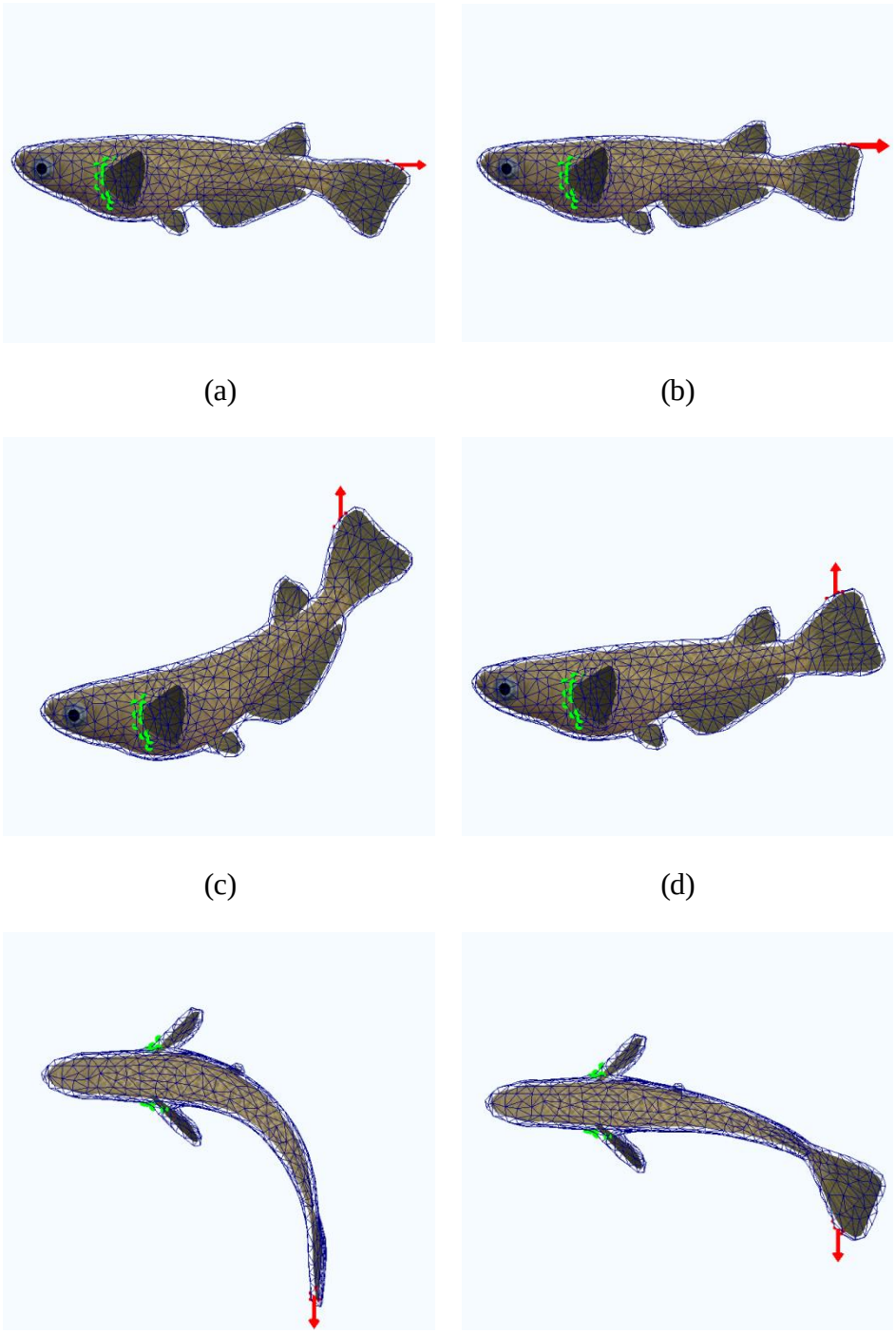
fixed in positions, and the red arrows represent the external directional force applied onto the selected vertices. In the descriptions, we denote ***r-, g-*** and ***b-***axis as the three principal axes in the frame-field.

For more robust simulation, the invertible elasticity approach [Irving '04] is applied to deal with degenerate and inverted elements during simulation.

In the first experiment, referring to Fig. 5.4, we use the fish model which represents a typical structure with a simple topology. We have used it to illustrate the generation of the frame-field in the previous sections. We present the comparison of the isotropic and the orthotropic deformations under three directional forces. The first column screenshots show the isotropic deformations and the second column shows the corresponding orthotropic deformations under the same directional force.

In this experiment, we define an orthotropic material with stiffness of 0.1x, 1.0x and 10.0x times of that of the isotropic material along the respective ***r-, g-*** and ***b-***axis directions in the frame-field, highlighted features are:

- In the pair of Fig. 5.4(a) and Fig. 5.4(b), the orthotropic model in the latter shows the desired resistance against a stretch along the longitudinal direction (10.0x in b-axis). It avoids the incorrect geometric deformation commonly seen in the isotropic model as the former.
- In the pair of Fig. 5.4(c) and Fig. 5.4(d), due to the increased stiffness from 1.0x to 10.0x in the b-axis of the latter, pulling the tail makes it bend upwards but the fish body is not wrongly elongated as in the former. This behavior matches with the dynamics of vertebrate animals.



(a) (b)

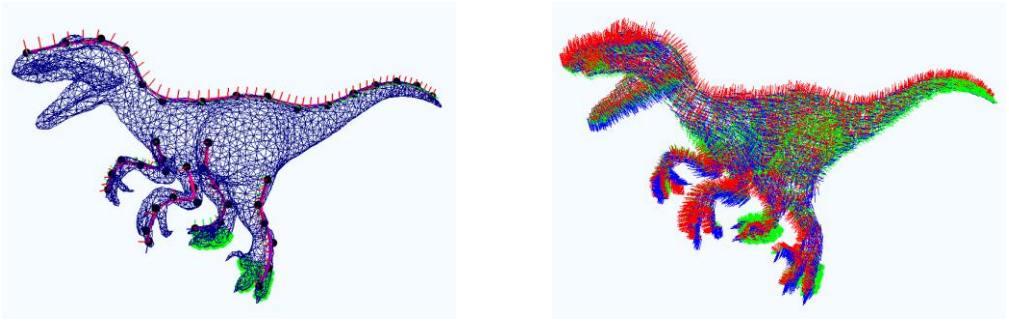
(c) (d)

(e) (f)

Fig. 5.4 Dynamics simulation of the fish model

- In the pair of Fig. 5.4(e) and Fig. 5.4(f), due to the decreased stiffness from 1.0x to 0.1x in the r-axis of the latter, pulling a corner of the tail makes a realistic twisting deformation which cannot be achieved in the former.

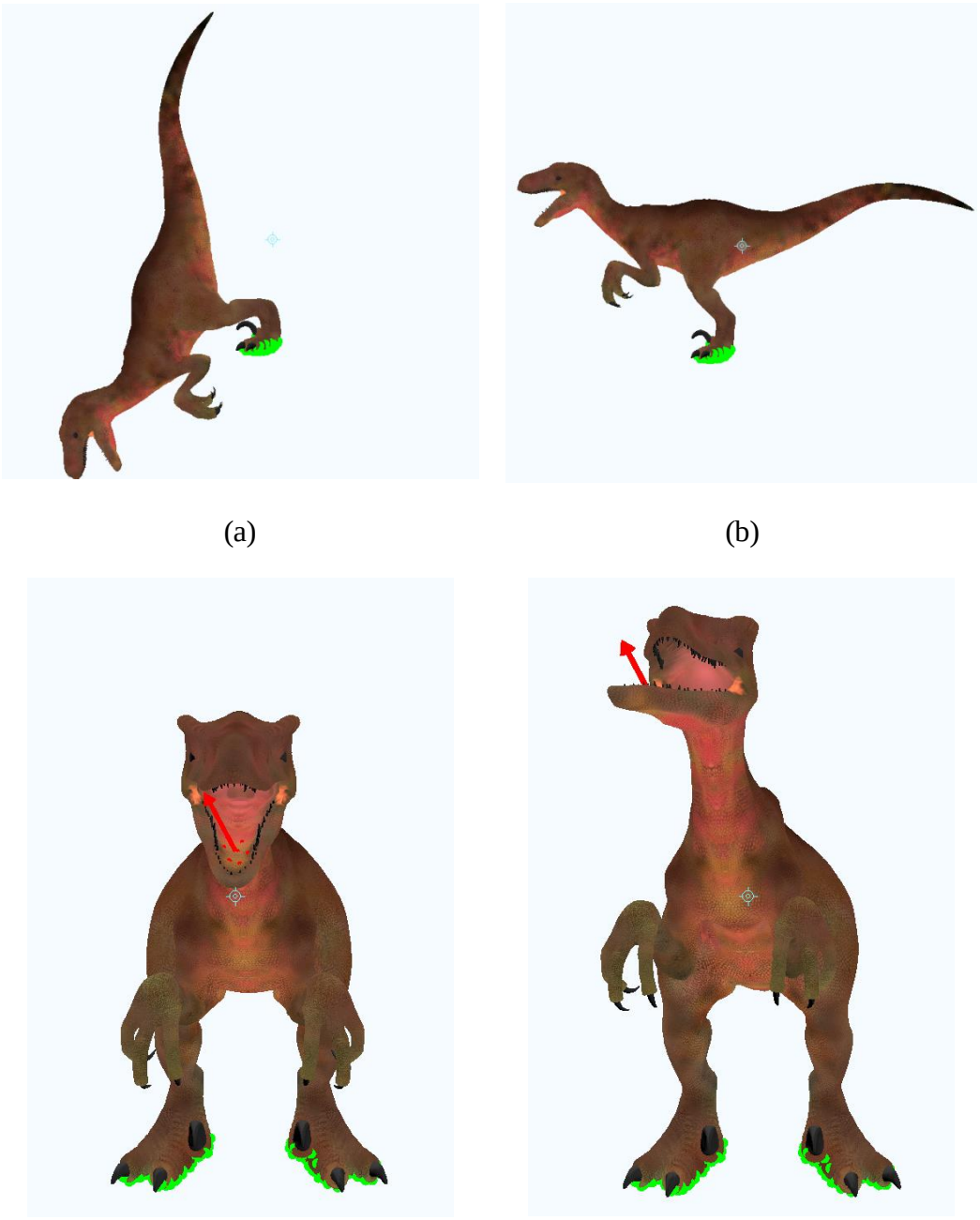
In the second experiment, referring to Fig. 5.5 and Fig. 5.6, we use a raptor model with more complex structures. We verify the controllability of multiple RMFs and the effectiveness of the Laplacian smoothing in defining an orthotropic material.



(a) (b)

Fig. 5.5 (a) Multiple RMFs for a raptor model, (b) The frame-field with the Laplacian smoothing

In this experiment, we also define an orthotropic material with stiffness of 0.1x, 1.0x and 10.0x times of that of the isotropic material along the respective ***r-***, ***g-*** and ***b-*** axis in the frame-field. If the isotropic model is set as a soft material, the raptor model cannot stand on its feet under gravity (as shown in Fig. 5.6(a)), acting like boneless soft body; or if set as a stiff material that 10.0x times of the one in Fig. 5.6(a), it lacks of flexibility in other directions (as shown in Fig. 5.6(c) under the directional force). In contrast, the orthotropic model shows the expected deformations under both gravity and the external force (as shown in Fig. Fig. 5.6(b) and (d)), which reflect to certain extent their internal structures.

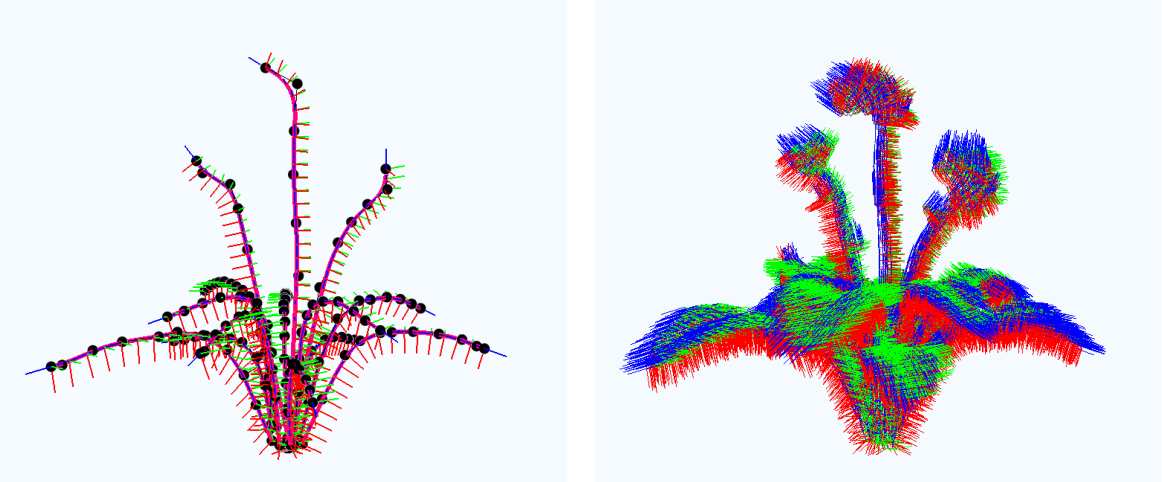


(a) (b)

(c) (d)

Fig. 5.6 Dynamics simulation of the raptor model

In the last experiment, referring to Fig. 5.7 and Fig. 5.8, we use a hosta plant model with structures of heterogeneous materials (soft leaves and stiff stems) to verify the effectiveness of the orthotropic model in combination with the stiffness matrices.

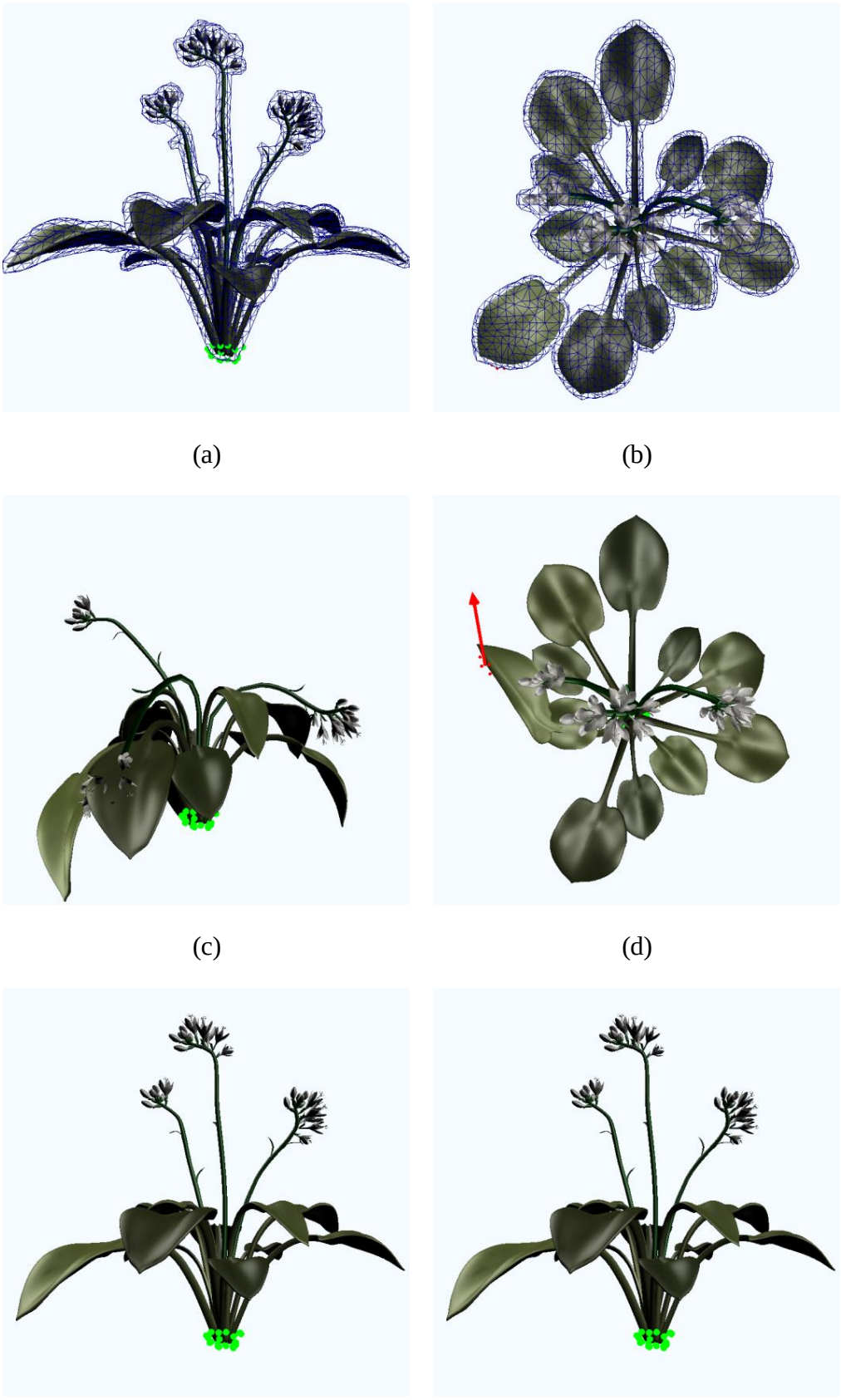


(a) (b)

Fig. 5.7 (a) Multiple RMFs for a hosta plant model, (b)The frame-field with Laplacian smoothing

For references, we present two views of the rest poses of the hosta plant model in the first row of Fig. 5.8. Highlighted features in this experiment are:

- In the second row of Fig. 5.8, the model of isotropic materials lacks varying stiffness in different parts, thus is pulled down uniformly by the gravity in Fig. 5.8 (c) and freely bent by the directional pulling force in Fig. 5.8(d).
- In the orthotropic model of the third row, we define different stiffness for the leaves and other parts, that is, 0.2x, 1.0x and 10.0x times of the isotropic material for the leaves, and 1.0x, 1.0x and 10.0x times for the others parts, along ***r-***, ***g-*** and ***b***-axis respectively. As expected, the heterogeneous orthotropic model well approximates the natural deformations of the plant model. Under the gravity and directional pulling force, the greater stiffness along its longitudinal direction tends to resist the deformation from the rest status. Meanwhile, the leaves with less stiffness exhibit the flexibility in deformation along the other directions, as shown in Fig. 5.8(e) and Fig. 5.8(f).



(a) (b)

(c) (d)

(e) (f)

Fig. 5.8 Dynamics simulation of the hosta plat model

To validate its consistent performance as in the isotropic model with CLFEM, we implement our algorithms for the orthotropic model on the same platform. The frames-per-second (FPS) performance of the dynamics simulation in the three experiments is given in Table 5.1.

Table 5.1 Simulation Performance

	vertices	tets	FPS (Iso)	FPS (Ortho)
fish	1114	4081	40	40
raptor	2296	8418	21	21
hosta	3367	9512	18	18

As can be seen, it is verified that there is no additional computational cost for the orthotropic deformations compared with the isotropic ones.

5.6 Conclusion

Modeling the anisotropic materials is highly desired but has been proven to be a challenging task. In this paper, we have studied the orthotropic materials, and proposed a deformable object modeling and dynamics simulation method for orthotropic materials of a wide range of objects such as animals and plants. Technical innovations are made in several aspects:

i. An orthotropic deformation controlling frame-field is conceptualized and the frame construction tool is developed for the user to define the desired material properties. Moreover, a quaternion Laplacian smoothing algorithm is designed for propagating the sparse and distributed frames to the entire object.

ii. To enable large anisotropic deformations, the orthotropic frame-field is coupled with the CLFEM model. In this way, we can complete a dynamics system for use in physically based deformation simulation.

To verify the effectiveness of the proposed algorithms, we have developed a comprehensive modeling and simulation system, and conducted experiments on real-time deformation simulation. Our analytical comparisons of isotropic and orthotropic materials in deformation behaviors have confirmed good real-time performance.

Though our model can couple with CLFEM for large deformations, and the degenerate and inverted cases can be handled properly in most cases, more restricted time step is required for scenarios where the anisotropy is increasing, that is, there are large differences among stiffnesses along the material axes. We propose study more on general nonlinear orthotropic models in the future.

Chapter 6. Skeleton Driven Animation of Deformable Objects

6.1 Introduction

In this work, we extend our research to more complicated deformable models with skeletons. Skeletal characters are widely used in animations, computer games, movies, etc. It is an active research field in computer graphics. There are still challenges in rigging and skinning techniques, with the goal to perform fast, stable and realistic character animation, as well as to achieve simple and intuitive control of material properties and dynamic behaviors.

Here, we develop a framework for animating skeletal characters, which combines traditional *skinning* and *dynamics simulation* of deformable objects:

- We present a fast skinning method with a physically based deformable model. The *position-based dynamics* framework is employed for its simplicity and stability, and a *strain-based constraint* is exploited instead of geometric constraints.
- We have devised a new layered constraint solving scheme, which significantly increases the convergence rate of the constraint solver.
- Furthermore, we use an interactively generated *frame-field* in combination with strain-based constraint. It helps a user to intuitively control the anisotropic material properties along different directions. This controllability is a desirable feature in animation design.

Real-time animation with Physically-plausible deformations and secondary motions are obtained. Anisotropic deformations are produced, where stretch and shear properties of an anisotropic material can be defined separately. With good computational efficiency, stability and controllability, our approach is a promising method for practical applications.

Following a review of the related works in Section **6.2**, we first introduce the new skeletal animation system in Section **6.3**. A simple rigging scheme is introduced to bind a volumetric mesh with a skeleton. In Section **6.4**, dynamic motion of the unconstrained nodes is simulated within a PBD framework, enhanced by strain-based constraints and a volume constraint. A layered constraint solving scheme is proposed for fast convergence. And thus, anisotropic material properties can be set with the help of a frame-field generated by the editing tool. In Section **6.5**, real-time animations are demonstrated, with comparisons of performance with previous methods and analyses of results. Section **6.6** concludes our techniques and discusses future work.

6.2 Related Work

In conventional skeletal animation, a character consists of a surface mesh (called *skin*) and a hierarchy of bones (called *skeleton*). A *rigged* mesh is controlled in accordance with the motion of the embedded skeleton. This is called *skinning*.

Linear blend skinning (LBS) [Magnenat-Thalmann '88] is a popular method for its simplicity and efficiency. The transformation matrix for each vertex of the rigged surface mesh is the weighted sum of those of its associated bones. However, in case of large rotating or bending motion, LBS suffers from the *candy-wrapper* and the *collapsing-joint* artifacts at some joint positions. *Dual-quaternion skinning* (DQS) [Kavan '07] [Kavan '08] resolves the problem, but it introduces the bulging-joint artifact when the bending angle is large. These skinning methods are purely geometrically-based and do not consider the underlying material properties of an animated character, which makes it difficult for them to generate both visually and physically-realistic results for versatile skeletal motions. Manual editing or post processing (e.g., [Kim '14]) is often needed for volume correction. A manual weight

painting process for weight refinement is required, which is quite tricky and time consuming. Moreover, they cannot produce secondary dynamic motions and small deformation details, such as jiggling and bulging effects, which are critical to the physical realism of an animation. Handcrafting all these detailed deformations is not practical.

Physically based skinning methods, where deformations of a skeletal character are generated by dynamics simulation, have been developed in recent years. Based on continuum mechanics, it generates physically accurate skinning results with a high degree of realism, but at high computational cost. *Capell et al.* [Capell '02] [Capell '07] presented a framework of skeleton-driven character animation. It models a character with an elastic *control lattice* attached to its skeleton and uses a linear elastic model in the simulation, which is solved by the finite element method (FEM). Constructing the control lattice is laborious and time consuming, and its resolution is not high enough to avoid undesirable artifacts. *Teran et al.* [Teran '05] proposed a quasistatic FEM method for flesh simulation that can robustly deal with inverted elements, but the scheme ignores the inertial effect and thus cannot capture dynamic details of secondary motions. *Gilles et al.* [Gilles '11] combined DQS with continuum mechanics, and formulated a generalized frame-based elastic model. The frame positions are updated in reaction to the internal forces instead of a scripted motion, thus the method cannot be applied to skeletal animation directly. A fast approach with nonlinear FEM proposed in [Kim '11a] considers two-way interactions among the skeleton, the body, and the environment, and real-time or near real-time performance can be obtained with explicit time integration. However, it is well-known that explicit integration is only conditionally stable and not suitable for practical applications. *Kim et al.* [Kim '11b] proposed a multi-domain subspace method to boost the simulation

speed, but it requires a time-consuming pre-processing including domain decomposition, subspace basis analysis and cubature optimization, and the range of deformations is constrained within a subspace.

Position-based dynamics proposed by *Müller et al.* [Müller '07] provides a simple framework for dynamics simulation of deformable objects (refer to the survey paper [Bender '14a]). The main advantages are its computational efficiency and stability. Explicit time integration can be used to produce stable results without suffering from overshooting problems. Thus it is suitable for real-time applications. *Deul et al.* [Deul '13] proposed a multi-layer surface model for character skinning within the PBD framework. It employs a method of shape matching with oriented particles [Müller '11] for elasticity simulation, and uses position-based constraints for contact handling, volume conservation and coupling of the skeleton with the deformable model. *Rumman et al.* [Rumman '14] [Rumman '15] presented a skeleton-driven deformable model with a volumetric mesh, which uses LBS and imposes three geometric constraints (stretch, volume, and bind constraints) within PBD. Geometric constraints which depend on the tessellation of the simulated mesh are used, thus making the deformation behaviors dependent on the mesh topology. Meanwhile, only isotropic deformations can be simulated by tuning the coefficients of the constraints.

In contrast, a strain-based constraint [Müller '14a], instead of geometric constraints, is exploited in our system. The constraint is defined in terms of a nonlinear *Green-Lagrange strain tensor* derived from solid mechanics [Bower '09]. Stiffness values of the constraints are independent of tessellation. Stretch and shear properties of a material are defined separately such that anisotropic deformations can be produced, with reference to the local frames generated by an interactive tool.

6.3 Formation of the Skeletal Animation System

6.3.1 Workflow of the System

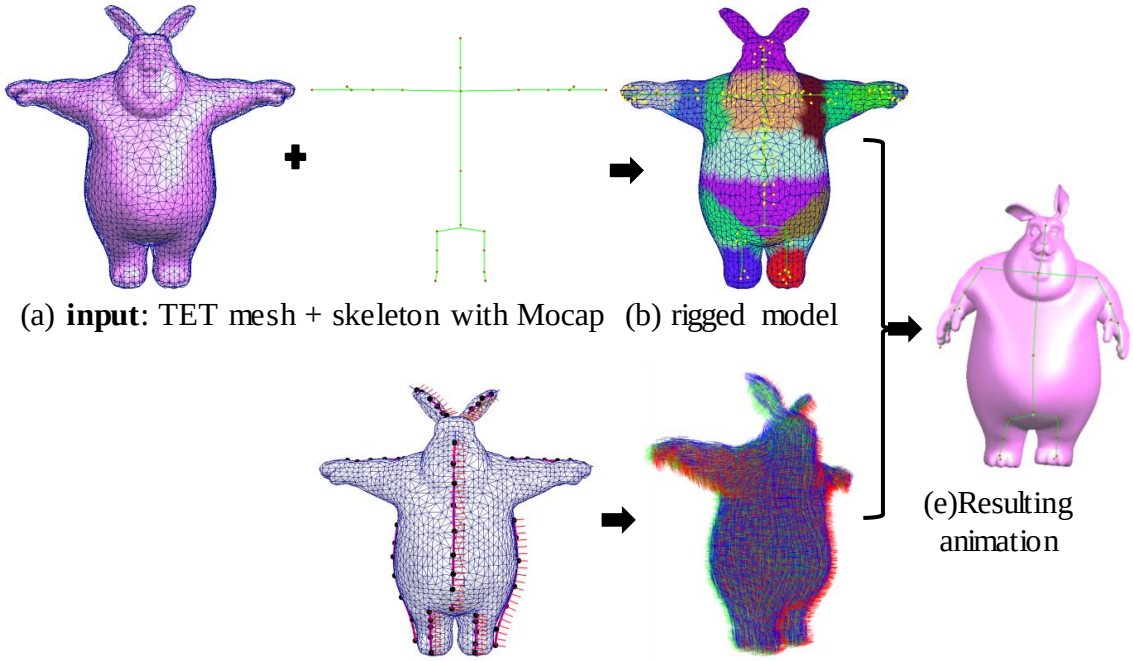


Fig. 6.1: Workflow of the skeletal animation framework

The workflow of our framework of skeletal animation is shown in Fig. 6.1. The input data includes a tetrahedral (TET) mesh and a skeleton with motion capture (Mocap) data, as shown in Fig. 6.1a. The wireframe mesh represents the simulated TET mesh, and the pink-colored mesh is a high-resolution surface mesh embedded in the TET mesh for high-quality rendering (as in [Müller '04]). By applying a simple rigging method, the TET mesh is attached to the skeleton, as shown in Fig. 6.1b. Optionally, an interactive tool is used to define NURBS curves upon the TET mesh (Fig. 6.1c), which is used to generate a volumetric frame-field (Fig. 6.1d) that helps to set anisotropic material properties. A frame of the resulting real-time skeletal animation is shown in Fig. 6.1e. In this section, we explain the proposed rigging and skinning

method, and how to generate a spatially varying frame-field for defining material properties.

6.3.2 A Simplified Rigging Scheme

A skeleton is represented as a hierarchical structure of bones and joints. In contrast with the LBS and DQS where many vertices are affected by multiple bones, a simple rigging scheme is proposed here: the tetrahedral elements intersected by the skeleton are constrained, and each constrained element is only attached to one bone, which makes the advanced weighting functions as in the LBS or DQS unnecessary. Therefore, it relieves an animator from a painful weight painting procedure.

Each frame of the skeletal motion data consists of the local translational and rotational transformations of each joint. The global transformation of each bone is computed as an accumulated transformation from the root joint to the current joint, which is represented by a 4×4 homogeneous matrix. In contrast to the LBS where each vertex of the rigged mesh is transformed by a weighted sum of the global transformations of its associated bones, a simple skinning scheme is sufficient here: the current position v' of a constrained node in an animation-frame is computed as

$$v' = M\,T^{-1}v_0 \tag{6.1}$$

where v_0 is the global position of the node in the initial configuration (i.e., the rest pose), M is the current global transformation matrix of its attached bone, and T is a transformation matrix defining the global position of the bone in the rest pose, which is only a translation matrix. This simple skinning scheme binds a TET mesh with an associated skeleton.

6.4 Dynamics Simulation within the PBD Framework

By considering the TET mesh as an elastic deformable object, the unconstrained nodes are then updated by a dynamics simulation driven by the skeletal motion. Instead of formulating the simulation as a rigorous solid mechanics problem and solving it by the commonly used finite element method [Sifakis '12] at a high computational cost, the PBD framework is employed here for its fast and stable performance.

Algorithm 1. Skeleton-driven PBD simulation

(1) Initialize positions and velocities: $\boldsymbol{x}_0$, and $\boldsymbol{v}_0$;
(2) **while** (simulating)
// update skeleton constraint at a Mocap frame
(3) update a motion frame (frameID++);
(4) update constrained nodes;
// symplectic Euler integration
(5) $\boldsymbol{v}_{t+1} \leftarrow \boldsymbol{v}_t + h * m_i^{-1} * \boldsymbol{f}_{ext}$;
(6) $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t + h * \boldsymbol{v}_{t+1}$;
//solve the constraints by projection
(7) **for** (i=0; i<#iterations; i++)
(8) $\boldsymbol{x'}_{t+1} \leftarrow$ ConstraintSolver ($\boldsymbol{x}_{t+1}$);
(9) **end**
// update velocities
(10) $\boldsymbol{v}_{t+1} \leftarrow (\boldsymbol{x'}_{t+1} - \boldsymbol{x}_t)/h$;
(11) DampedVelocity($\boldsymbol{v}_{t+1}$);
(12) **end**

Combined with the motion data, the simulation loop of the PBD is depicted in Algorithm 1. In the first step, a motion frame is updated and the positions of the constrained nodes are updated by the simple skinning in Eq. (6.1). Then, a *symplectic*

Euler integration computes the predicted positions (as in steps 5 and 6). Here h denotes the time-step size, $\boldsymbol{f}_{ext}$ the external force with only gravity considered here. m_i^{-1} is the inverse mass of each node, which is set to zero for a constrained node. The key component of the algorithm is the constraint solver (in steps 7 - 9). It corrects the blindly predicted positions in the time integration step by iteratively solving different constraints in a Gauss-Seidel-like way, which is called *positional projection*. Finally, the nodal velocities are updated and damped in steps 10 and 11.

6.4.1 Strain-based Constraint

Instead of geometric constraints used in the previous works for skeletal animation [Deul '13, Rumman '14, Abu Rumman '15], a strain-based constraint [Müller '14a] is exploited here, which adds more physical authenticity to the PBD method. Furthermore, anisotropic dynamic behaviors can be obtained.

In solid mechanics, deformation can be measured by a strain tensor defined as $C = \frac{1}{2}(F^{\mathrm{T}}F - I) \in \mathbb{R}^{3\times3}$, called the *Green-Lagrange strain tensor*, where $F \in \mathbb{R}^{3\times3}$ is the *deformation gradient*, $I \in \mathbb{R}^{3\times3}$ is the identity matrix. Given a TET element with its nodal positions $(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)$ in the initial configuration and $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ in the deformed configuration, the deformation gradient can be approximately computed as

$$F = D_s\, D_m^{-1}, \tag{6.2}$$

where $D_s = [\boldsymbol{x}_1 - \boldsymbol{x}_0, \boldsymbol{x}_2 - \boldsymbol{x}_0, \boldsymbol{x}_3 - \boldsymbol{x}_0]$, and $D_m^{-1} = [\boldsymbol{X}_1 - \boldsymbol{X}_0, \boldsymbol{X}_2 - \boldsymbol{X}_0, \boldsymbol{X}_3 - \boldsymbol{X}_0]^{-1}$. Here, D_m^{-1} can be pre-computed, and F is constant within an element. The nonlinear strain tensor is rotation-invariant, thus it supports large deformations.

A constraint function can be defined based on the strain tensor, as

$$C(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \frac{1}{2}(F^T F - I) = \mathbf{0}. \tag{6.3}$$

With this constraint satisfied, a deformed element can be restored to its initial pose. Since C is symmetric, six scalar functions of constraints are obtained here, including three stretch constraints with respect to the diagonal components and three shear constraints with respect to the off-diagonal components, written as $C_{ij}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = 0, i < j$. For faster convergence of the constraint solver, we used a linearized stretch constraint [Müller '14] as $C_{ii}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \sqrt{S_{ii}} - 1 = 0$, where S_{ii} are the diagonal elements of $F^{\mathrm{T}}F$.

In contrast with geometric constraints on edge lengths and element volumes, the strain-based constraint is independent of the mesh tessellation, and provides the user with more control over the material properties along different directions. Later we design an interactive tool to define a frame-field as a reference for defining the anisotropic material properties.

6.4.2 Volume Constraint

Theoretically, a deformed element can be restored to its original shape if the strain-based constraints are completely satisfied. However, since the strain-based constraint is solved in a Gauss-Seidel-like manner and only by a few iterations, an additional volume constraint for volume conservation is useful and necessary to deal with scenarios of large deformations. From the fact that $J = \det(F)$ represents the fraction of volume change of an element, where $J = 1$ means no volume change, a volume constraint can be derived as

$$J(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) - 1 = 0. \tag{6.4}$$

The advantage of this volume constraint is that inverted or degenerated elements can be restored without special treatments such as the methods in [Irving '06] and [Stomakhin '12].

6.4.3 The Layered Constraint Solver

A fundamental scheme for solving a constraint function can be formulated as follows [Müller '07]. Given a constraint function $c(\boldsymbol{x}) = 0$ where $\boldsymbol{x}$ is a vector of constrained variables (e.g., $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ for a TET element), the positional correction $\Delta\boldsymbol{x}$ can be obtained by firstly applying a first-order Taylor expansion as

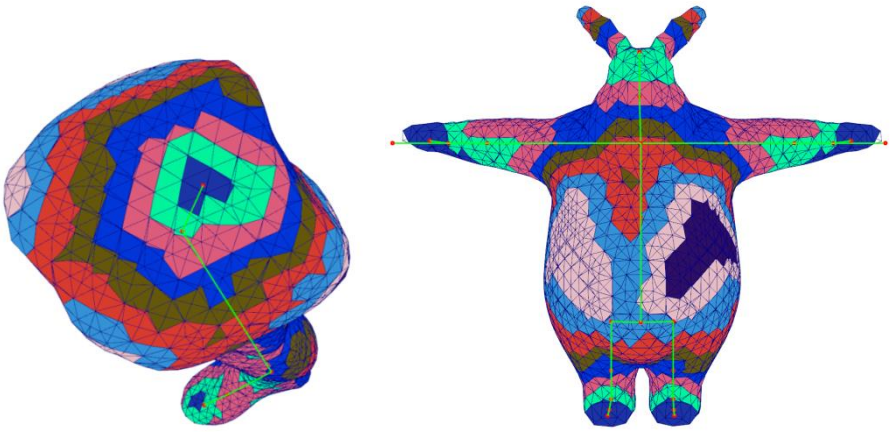
$$c(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx c(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} c \cdot \Delta\boldsymbol{x} = 0, \tag{6.5}$$

where $\nabla_{\boldsymbol{x}} c = \frac{\partial c}{\partial \boldsymbol{x}}$ is the gradient of c with respect to $\boldsymbol{x}$. Then by restricting the direction of $\Delta\boldsymbol{x}$ as $\Delta\boldsymbol{x} = \lambda \nabla_{\boldsymbol{x}} c$ for the conservation of the linear and angular momentum, we have

$$\Delta\boldsymbol{x} = k \left(-\frac{c(\boldsymbol{x})}{|\nabla_{\boldsymbol{x}} c|^2} \right) \nabla_{\boldsymbol{x}} c, \tag{6.6}$$

where k is the stiffness defining the strength of the constraint. By considering the nodal weight $w_i = m_i^{-1}$, the nodal correction vector is finally computed as

$$\Delta\boldsymbol{x}_i = w_i \, k \left(-\frac{c(\boldsymbol{x})}{\sum_{j=0}^{3} w_j \left| \nabla_{\boldsymbol{x}_j} c \right|^2} \right) \nabla_{\boldsymbol{x}_i} c, \tag{6.7}$$



(a) Leg model (b) Bunny model

Fig. 6.2: TET elements are re-ordered in different layers, so that the constraints can be solved layer-wisely (layer rendered in different colors).

The strength of a certain constraint can be defined by a corresponding stiffness k, thus the final position correction is taken as $k\Delta\boldsymbol{x}$. This projection process is usually done by a predefined number of iterations in a Gauss-Seidel manner, i.e., it solves the constraint element-wisely, and the resulting positional correction of each element immediately affects following updates. The constraint solver actually propagates the deformations from place to place across the whole TET mesh. It converges to an equilibrium state if an infinite number of iterations are taken.

There are limitations with the constraint solving scheme. First, it converges slowly, meaning that it requires many iterations to converge, especially for high resolution TET meshes (i.e., it takes longer time to propagate a deformation across a mesh with a higher resolution). This also means that the mechanical behaviors depend on mesh resolution. Second, the order of constraints also affects convergence, meaning that if the constraints of elements are solved in a different order, it will produce different deformation behaviors (i.e., show different material stiffness).

In our animation system, the constrained nodes moves with the skeleton and the rest free nodes are updated by the constraint solver. Based on the fact that

deformations are propagated from the constrained nodes to the free nodes, we reorder the elements into layers according to the distances from the skeleton, such that each bone is surrounded by multi-layers of elements, as illustrated in Fig. 6.2 for two TET meshes (a *leg* model and a *bunny* model). Then, the solver solves the constraints upon elements layer-by-layer.

In order to further improve the convergence rate at an iteration, during the first few iterations, we artificially increases the nodal masses of this layer to consolidate the previous updates after updating nodal positions of each layer (named *consolidating iterations* here) , and reset the masses to the original configuration for later iterations. This *layered constraint solver* requires much less iterations for each animation frame than the un-ordered scheme used in previous works. Moreover, it relieves the dependency on different mesh resolutions by increasing the number of consolidating iterations for a higher resolution mesh. More implementation details and results are discussed later.

6.4.4 Frame-Field Augmented Anisotropic Model

Soft tissues of a character often exhibit anisotropic elastic behaviors, thus it is necessary to define a spatially varying frame-field to help specify material properties in local frames, as shown in Fig. 6.1d. Here we use the interactive tool devised in the previous chapter (refer to Section **5.4**) to generate the frame-field.

In order to augment the simulation with the frame-field control, the deformation gradient (as in Eq. (6.2)) of each element should be computed in the local frames as

$$F' = (Q_i^T D_s)(Q_i^T D_m)^{-1}, \tag{6.8}$$

where Q_i is the local frame matrix in Eq. (5.4). The inverse matrix of the second term can be pre-computed.

Therefore, the constraint solver yields a local correction vector $\Delta \boldsymbol{x}_i'$, which needs to be transformed back to a global correction vector $\Delta \boldsymbol{x}_i = Q_i \Delta \boldsymbol{x}_i'$.

6.4.5 Discussion

A decoupled shear constraint $C_{ij}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \frac{\sqrt{S_{ij}}}{|S_{ij}|} = 0, i < j$ was also proposed in [Müller '14], which decouples shear from stretch and penalizes only the angular deformation. However, this decoupled shear constraint is nonlinear along the projection direction. It causes overshooting problems, leading to simulation failure (which happens in our experiments) when large bending angles occur at the joints. Thus, non-decoupled shear constraint, which affects the stretch of an element, is applied here. Combining it with the volume constraint might result in an over-constrained system, producing jittering artifacts. This issue can be solved by decreasing the stiffness of the volume constraint.

Another issue with the PBD framework is that material stiffness depends on not only the constraint coefficients, but also the time-step size and the number of iterations in the constraint solver. Here we set the time step to be consistent with the frame-rate of the skeleton motion, and set a constant number of iterations.

6.5 Computational Results

In the skeletal animation system, a high resolution surface mesh is embedded in a coarse tetrahedral mesh for the final rendering, and is deformed by barycentric interpolation of the simulated mesh. A coarse volumetric mesh is useful to boost

performance speed. Kinematic motion capture (Mocap) data is obtained from the open Mocap databases [CMU , Huang '06] and [NUS] in *BVH* format (refer to [Biovision] for specification). And we use Blender software [Blender] to edit the skeletal structure in order to match it to a character model. We implemented the system in C++, and all the algorithms are CPU-based.

6.5.1 Setting of Constrained Elements

Initially, we constrained all the skeleton-intersected elements (as shown in Fig. 6.3a, constrained nodes are rendered as yellow points), which turned out to produce undesirable artifacts with the skinning results. If the nodes of a constrained element are attached to different bones, outliers are produced due to the volume constraint. *Rumman et al.* [Rumman '14, Rumman '15] solved this problem by adding an additional bind constraint, which maintains the distance between a node and a related bone; however, it adds more computational cost.

Here, we establish a simple rule that tetrahedra intersected by more than one bone (i.e., around the joints) should not be constrained, i.e., there is only a one-to-one relationship between constrained elements and attached bones. Fig. 6.3d shows the corrected constrained nodes, and Fig. 6.3e and 3f show correct deformation behaviors around all the joints. This strategy makes the *bind constraint* unnecessary, which saves computational cost.

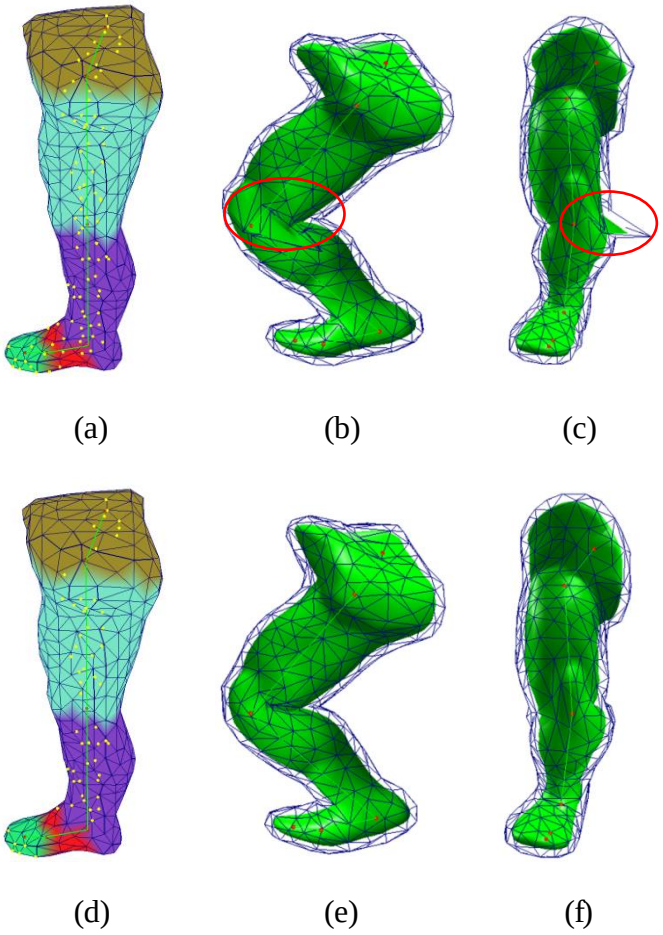
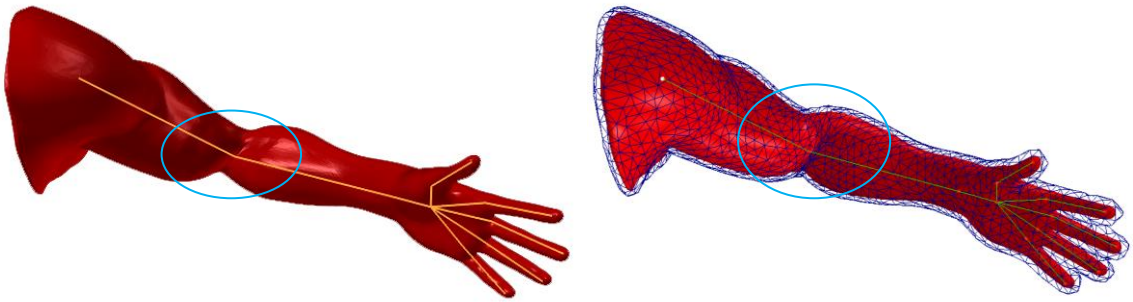


Fig. 6.3: Constrained elements by the skeleton. In (a), all the elements intersected by the skeleton are constrained, producing outliers shown in (b)(c). In (d), bone-shared elements are excluded, and correct deformations are produced shown in (e) and (f).

6.5.2 Comparison with Conventional Skinning Methods

Here we compare our method with the popular LBS and DQS in different scenarios. As shown in Fig. 6.4a, the LBS method suffers from a well-known candy-wrapper artifact when performing a twisting motion. In Fig. 6.5b, the DQS suffers from a bugling-joint artifact with a bending motion, whereas the LBS produces a better deformation (Fig. 6.5a); a textured-cylinder model is used here to provide a better visual reference for comparison. Therefore, none of these methods can produce satisfactory deformations in all scenarios. Since our skinning method works with physically-based constraints, realistic deformations can be produced without suffering

from these artifacts. Volume is well preserved, as shown in Fig. 6.4b and Fig. 6.5c. Furthermore, there is no need of a weight-painting refinement process as is used in the previous methods.



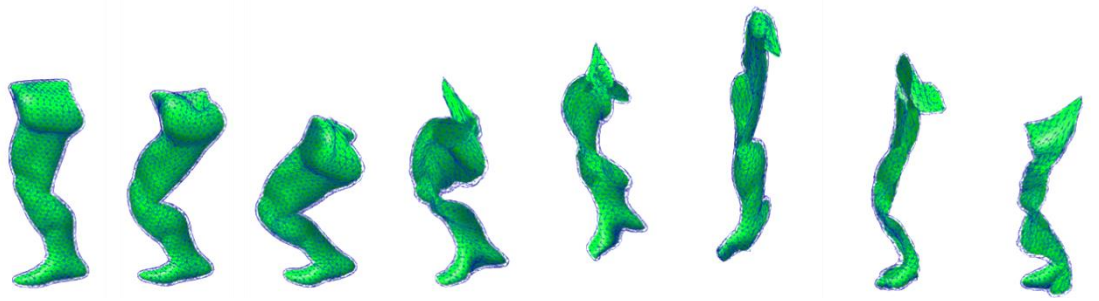
(a) (b)

Fig. 6.4: Comparison with the LBS method: (a) LBS skinning with a candy-wrapper artifact; (b) Skinning by our method.

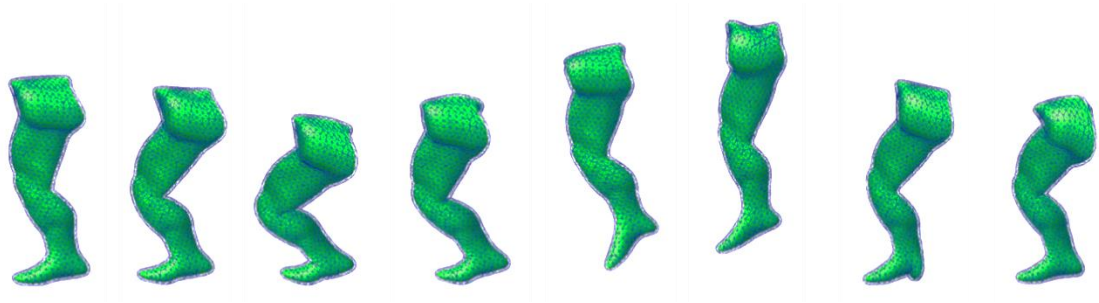
(a) (b) (c)

Fig. 6.5: Comparison with the LBS and DQS. (a) LBS skinning; (b) DQS skinning with a bulging-joint artifact; (c) Skinning by our method.

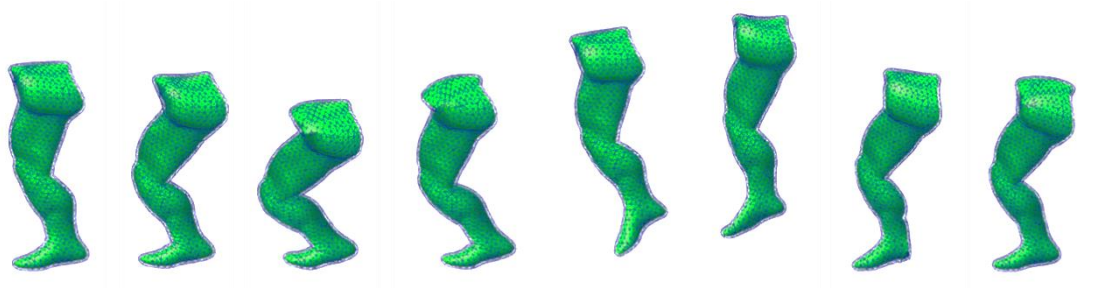
6.5.3 Comparison with Unordered Constraint Solver



(a) Unordered solver with 10 iterations



(b) Unordered solver with 50 iterations

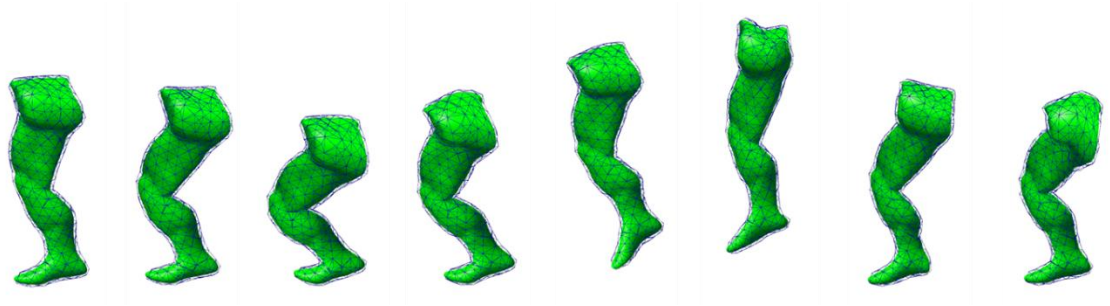


(c) Layered constraint solver with 10 iterations (contains 6 consolidating iterations)

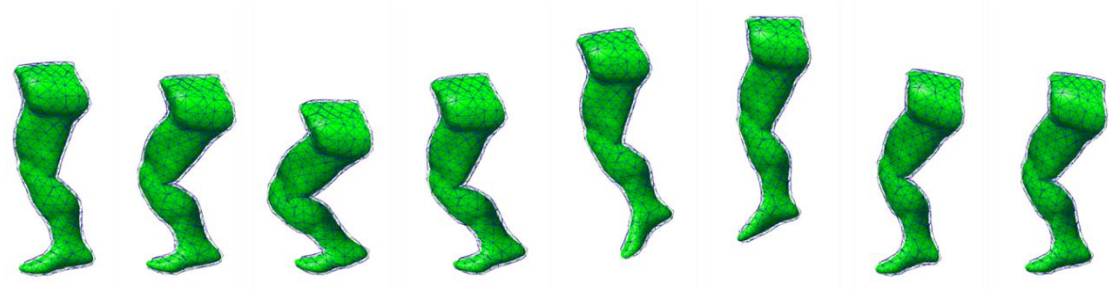
Fig. 6.6: Comparison between unordered solver and our layered constraint solver

Here we compare the results of unordered constraint solver and our layered constraint solver. As shown Fig. 6.6, a jumping leg model (with 24387 TET elements) is simulated, with all the three rows are simulated by the same constraint coefficients and time-step size. The unordered solver converges slowly, as shown in Fig. 6.6a, that 10 iterations are far from convergence, but it needs 50 iterations to obtain satisfactory behaviors as in as shown Fig. 6.6b. In contrast, the reordered solver in as shown Fig. 6.6c takes only 10 iterations (with the first 6 as consolidating iterations) to achieve

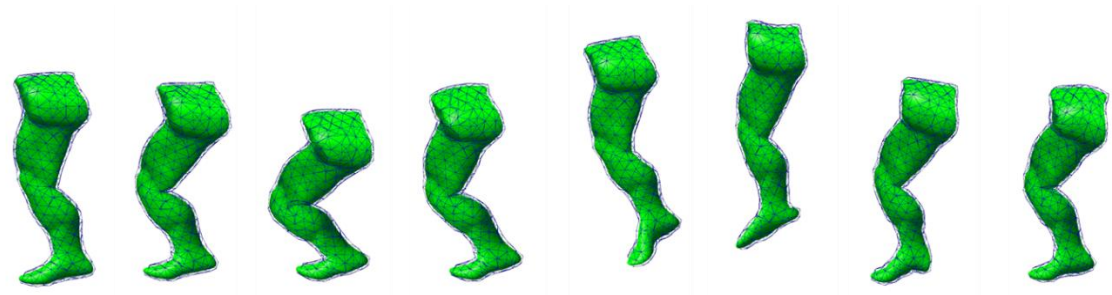
similar results. This validates the fast convergence rate of the proposed ordered solving scheme, which greatly reduces computational cost for solving the constraints.



(a) Unordered solver with 10 iterations



(b) Layered constraint solver with 10 iterations (contains 6 consolidating iterations)



(b) Layered constraint solver with 10 iterations (contains 2 consolidating iterations)

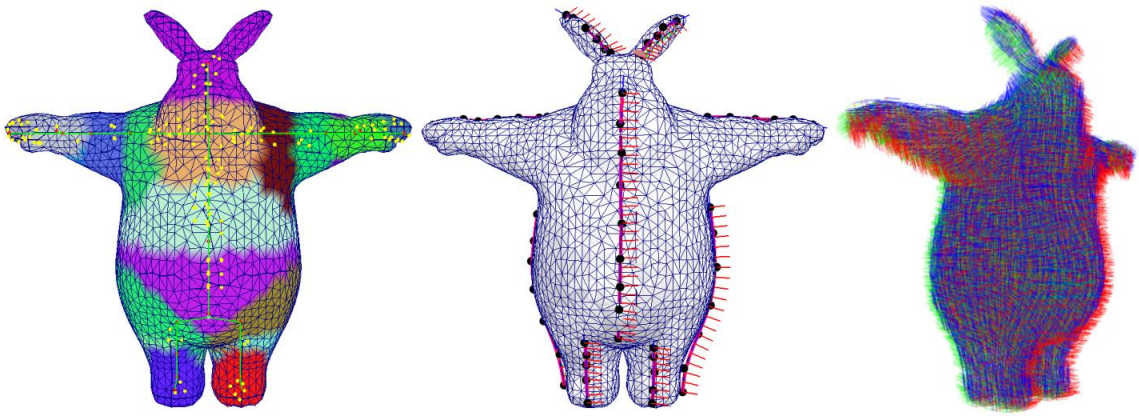
Fig. 6.7: Comparison the results of unordered solver and our layered constraint solver, for a low-resolution mesh

For the unordered solver, there is a large difference of convergence rates among meshes with different resolutions. In Fig. 6.7, a lower-resolution leg model (with 3452 TET elements) is simulated with the same constraint coefficients and time-step size as in Fig. 6.6. As shown in Fig. 6.7a, the unordered solver with 10 iterations converges much faster with the lower-resolution mesh than the high-resolution one in Fig. 6.6a. This poses difficulty for deformation control of meshes with different resolutions, due

to large performance differences among them. The ordered solver larges alleviates this problem. As shown in Fig. 6.7b, with the same setting with Fig. 6.6c, the results show stiffer material properties than the former one; it can be adjusted by decreasing the number of consolidating iterations (2 consolidating iterations as in Fig. 6.7c) to achieve similar deformation behaviors with Fig. 6.6c. Therefore, deformation control can be done at similar performance level by using the ordered solver.

6.5.4 Comparison of Isotropic and Anisotropic Deformations

Here, we demonstrate the simulation of a frame-field augmented simulation of a fat bunny model (obtained from Blender open movie project [Foundation '08]). Fig. 6.8a shows a skeleton-constrained TET mesh; Fig. 6.8b shows the NURBS curves and associated RMFs for generation of the frame-field in Fig. 6.8c.

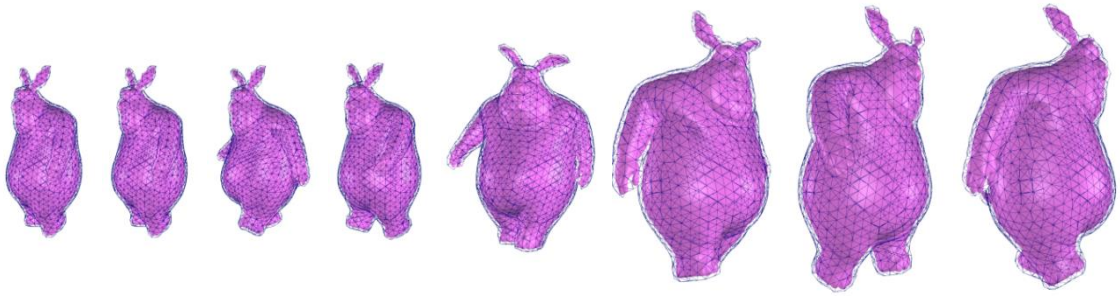


(a) (b) (c)

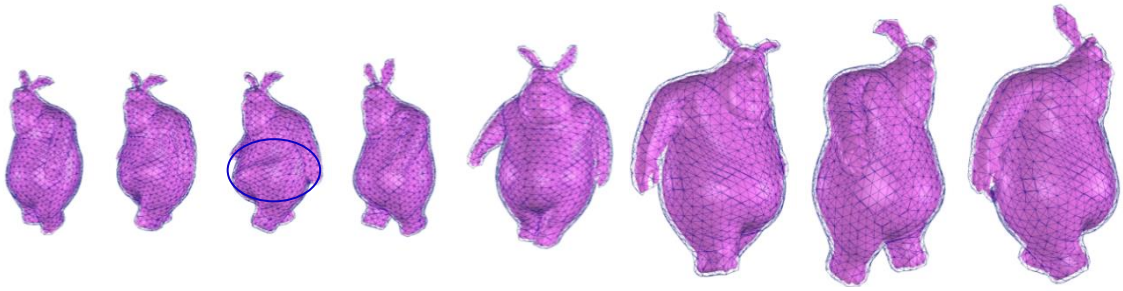
Fig. 6.8: (a) Skeletal constraint; (b) NURBS curves and associated RMFs; (c) The generated frame-field.

In Fig. 6.9, we show the deformations of models with isotropic and anisotropic materials. Without a reference frame-field, we can only define isotropic materials: the strain-based constraints are solved in a single global frame, and there are only two strain-based coefficients (i.e., one stretch stiffness and one shear stiffness). Isotropic

deformations are shown in Fig. 6.9a. With the help of the frame-field, where each frame is represented by three axes ($\boldsymbol{r}, \boldsymbol{g}, \boldsymbol{b}$) rendered in red, green and blue colors respectively, totally six constraint coefficients can be defined: three stretch coefficients and three shear coefficients along three directions of each local frame for each TET element. Here we adjust six parameters to define an anisotropic material. The anisotropic setting exhibits more complicated and expressive deformations than the isotropic one, such as the wrinkles formed at its belly and more flexible behaviors at the ears, as shown in Fig. 6.9b. Previously, using geometrical constraints as in previous works (as in [Rumman '14, Rumman '15]) is difficult to define anisotropic materials, especially for a TET mesh with an irregular mesh grid.



(a) Isotropic deformations



(b) Anisotropic deformations by using the frame-field

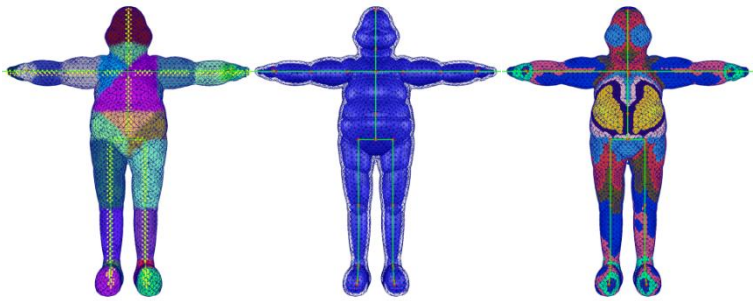
Fig. 6.9: Comparison between isotropic and anisotropic deformations

6.5.5 Comparison with Previous Skeletal Animation using PBD

In a mostly related work of skeletal animation by *Rumman et al.* [Rumman '14, Rumman '15], a straightforward *two-pass* deformation scheme was proposed: first, the

LBS scheme is applied on a whole TET mesh; second, all the vertex positions are corrected by a PBD simulation with respect to three geometrical constraints (stretch, volume and bind constraints).

Compared with their work: (1) our method avoids the LBS computation, and no weight function needs to be defined; instead, only a simple and fast rigging is required to update the constrained nodes; (2) Strain-based constraints instead of geometrical constraints are exploited, which makes the definition of material properties independent of mesh grid, and anisotropic materials can be defined; no bind constraint is needed to main the rest distance between a node and its projection on the skeleton. (3) With the layered constraint solving scheme, our solver converges much faster, which has more computational efficiency.



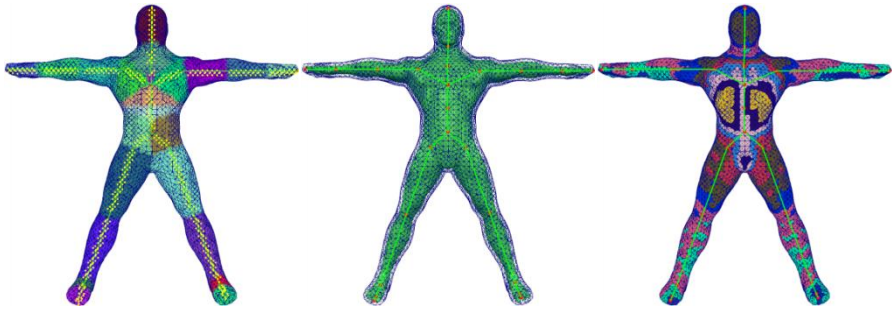
(a) (*from left to right*): a rigged model; a surface mesh embedded in a TET mesh; multiple layers of elements in different colors

(b) Some animation frames

Fig. 6.10: Some animation frames (The surface model is courtesy of *Kim* [Kim '11a])

6.5.6 Performance Analysis

More simulation results are shown in Fig. 6.10 and Fig. 6.11, which demonstrates the effectiveness and robustness of our method against large ranges of motions.



(a) (*from left to right*): a rigged model; a surface mesh embedded in a TET mesh; multiple layers of elements

(b) Some animation frames

Fig. 6.11: Skeletal animation of a male model

An un-optimized implementation of the system is done with CPU-based algorithms in C++. All the examples are executed on a desktop with Intel Core CPUs (i7-4770 @3.40GHz). The time-step size of the constraint solver is set in consistent with the frame-rate of the Mocap data (120Hz with the male model, and others with 30Hz or 25Hz). Statistics and simulation performance of the examples are shown in Table 6.1. Simulation performance with different models. It shows that the frame-field augmented simulation only costs a small fraction of time more than the one without a

frame-field, which involves local-frame computations. All the simulations are done within 10 iterations for the constraint solver, thanks to the layered solving scheme. Therefore, real-time performance can be achieved.

Table 6.1. Simulation performance with different models

Model	**#Total elements**	**#Unconstrained elements**	**Time-step size (*s*)**	**Number of Iterations** (total/consolidating)	**Simulation Time (*ms*)**	
					without frame-field	with frame-field
Cylinder	2214	2169	0.042	3/1	2.3	2.8
Leg (low resolution)	3452	3373	0.033	10/2	11.6	14.6
				10/6	11.8	14.6
Arm	10760	10433	0.042	6/2	21.5	27.1
Leg (high resolution)	24837	24670	0.033	10/6	87.5	108.5
Bunny	27350	27136	0.033	4/1	37.8	48.3
Male	74288	73647	**0.008**	10/1	258.5	335.1
Fatman	89876	89306	0.033	10/2	317.7	408.5

6.6 Conclusion

We have presented a real-time system that exploit the advantages of both dynamics simulation of deformable objects and skeletal animation. A strain-based PBD framework is used the geometrically based skinning techniques such as LBS and DQS. This turns out to be effective and efficient for animating characters consisting of soft tissues. Natural secondary motion of soft tissues can be produced, while anisotropic deformation behaviors can be intuitively controlled by using a user-designed frame-field. Stretch and shear coefficients can be easily tuned to change material stiffness

along different directions. The constraint solver converges fast with an ordered layered solving scheme. Stable and real-time performance can be achieved, thanks to the PBD framework.

There are still rooms for improvement in the future. First, volume can be well preserved for each element; however, without dealing with self-contact, self-penetration artifacts can occur. In the future, self-collision constraint will be exploited in order to obtain more accurate deformation behaviors. Second, currently there is only a one-way interaction between the skeleton and the simulated mesh, that is, motion of the mesh is driven by the skeleton. Furthermore, a two-way coupling approach can be incorporated into the system. The motion of the soft tissues can also affect the movements of the pre-defined motion of skeleton. This would be helpful to apply a same Mocap data to characters of different sizes and weights, and re-generate different locomotion that respects their different characteristics. Third, better tetrahedralization algorithm can also be developed to generate TET mesh with qualified layers of elements, which would improve the performance of the layered constraint solver.

Chapter 7. Conclusion

This thesis presents our research on dynamics simulation of deformable models applied in computer graphics field. Both geometrically-based and physically-based approaches have been studied in our work.

We focus on deformable objects of anisotropic materials, which are less exploited than those of isotropic materials in existing work. Deformable objects of anisotropic materials are commonly seen in the real world, which exhibit more complicated and flexible mechanical behaviors. Therefore, compared with isotropic models, anisotropic models bring new challenges for designing stable and efficient simulation approaches, and for developing deformation control methods.

7.1 Major Contributions

Our main goals are that: 1) developing stable and efficient approaches for dynamics simulation of anisotropic models, which can be applied in interactive or real-time applications; 2) developing control methods to achieve directable deformation behaviors of anisotropic models. Our major contributions are as follows:

- Firstly, we have studied transversely isotropic materials for simulation of deformable objects with fibrous structures. A novel fiber-field incorporated CLFEM model is developed, which is computationally efficient and allows large deformations. Transversely isotropic deformations can be intuitively controlled by a user input fiber-field.
- Secondly, we have further investigated deformable objects of orthotropic materials, which have more complicated material properties and more flexible mechanical behaviors. A novel interactive tool is designed to produce a smooth frame-field inside a volumetric mesh, and a quaternion Laplacian smoothing algorithm is

developed in order to generate such a frame-field. We have formulated a CLFEM model augmented with an orthotropic frame-field. Robust dynamics simulation is achieved by a stable tuning of material properties and dealing with inverted and degenerate deformations.

- Thirdly, we have developed a novel method of physically-based skeletal character animation. Deformations of a deformable character are driven by the motion of an embedded skeleton, combining conventional skeletal animation with dynamics simulation. To further improve the efficiency and stability of simulation, we exploit the PBD framework instead of the rigorous FEM model. A strain-based constrain is employed instead of geometric constraint used in previous works. A new layered constraint solving scheme is devised for fast convergence. This bridges the gap between PBD and rigorous physical approaches. The frame-field used in the FEM model can also be incorporated in the strain-based constraint; anisotropic deformation behaviors can be controlled intuitively and stably by setting different constraint coefficients with respect to the local frames of the frame-field.

The proposed approaches can be well applied in many graphics applications, such as soft tissues simulation in virtual surgery, computer games and animations.

7.2 Future work

In computer graphics research, it has always been a goal to simulate real-world phenomenon. Dynamics simulation of deformable objects is still an active research area, with the goal of pursuing more stable, efficient and physically-realistic results. There are some issues that are not investigated in our current work, but they are also of

great importance to a practical simulation framework. We might further investigate the following issues in the future:

- To improve the simulation performance of our current implementation, parallel computing algorithms need to be designed, either by using multi-threading on CPUs, or by harnessing the computing power of modern GPUs.
- Collision detection and response are still two challenging problems for deformable objects simulation. They are important for simulating phenomenon of self-interaction and interactions with environments.
- Since real-world deformable objects are not ideally elastic models, mechanical behaviors such as elastoplastic deformations and fractures also need to be considered for improved realism.
- In a simulation system, besides deformable objects (with different dimensions such as 1D ropes, 2D shells and 3D solids), there are often other object types such as rigid bodies, gases, and liquids. A unified physics framework has become popular in visual effects in recent years (e.g., a unified PBD framework is shown in Fig. 7.1 [Macklin '14]). With a unified dynamics solver, different types of objects that were previously simulated with different solvers can interact with each other in a fully-coupled way. However, there are still challenges for object representations, efficient solvers, modelling methods with physics support, etc.

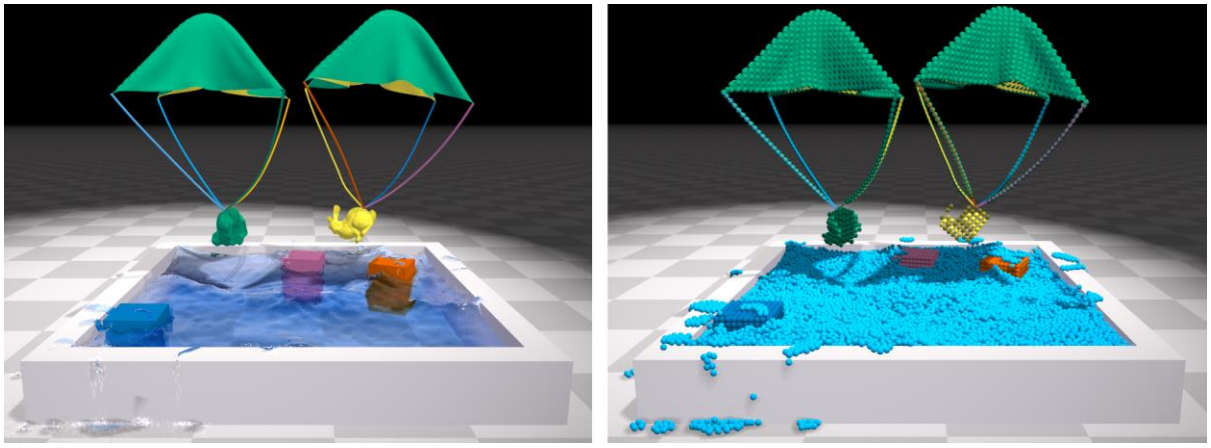


Fig. 7.1 Unified Particle Physics for Real-Time Applications [Macklin '14]

References

Abell, Martha L and James P Braselton (2014). Introductory Differential Equations: with Boundary Value Problems, Elsevier.

Abu Rumman, Nadine and Marco Fratarcangeli (2015). "Position-Based Skinning for Soft Articulated Characters." Computer Graphics Forum **34**(6): 240-250.

Alexa, Marc (2003). "Differential coordinates for local mesh morphing and deformation." The Visual Computer **19**(2): 105-114.

Allard, J, S Cotin, F Faure, P. J Bensoussan, F Poyer, C Duriez, H Delingette and L Grisoni (2007). SOFA - an Open Source Framework for Medical Simulation. Medicine Meets Virtual Reality 15. J. D. Westwood, R. S. Haluck, H. M. Hoffman et al. **125:** 13-18.

Allard, Jérémie, Hadrien Courtecuisse and François Faure (2011). Implicit FEM Solver on GPU for Interactive Deformation Simulation. GPU Computing Gems Jade Edition. W. H. Wen-mei, Elsevier**:** 281-294.

An, Steven S, Theodore Kim and Doug L James (2008). Optimizing cubature for efficient integration of subspace deformations. ACM Transactions on Graphics (TOG), ACM.

Baraff, David and Andrew Witkin (1998). Large steps in cloth simulation. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM**:** 43-54.

Barbič, Jernej (2007). Real-time reduced large-deformation models and distributed contact for computer graphics and haptics, Carnegie Mellon University. **PhD**.

Barbič, Jernej and Doug L. James (2005). Real-Time subspace integration for St. Venant-Kirchhoff deformable models. ACM SIGGRAPH 2005 Papers. Los Angeles, California, ACM**:** 982-990.

Barbič, Jernej and Jovan Popović (2008). Real-time control of physically based simulations using gentle forces. ACM Transactions on Graphics (TOG), ACM.

Barbič, Jernej, Marco da Silva and Jovan Popović (2009). "Deformable object animation using reduced optimal control." ACM Trans. Graph. **28**(3): 1-9.

Barbič, Jernej, Funshing Sin and Eitan Grinspun (2012). "Interactive Editing of Deformable Simulations." ACM Trans. on Graphics (SIGGRAPH 2012) **31**(4).

Barbič, Jernej and Yili Zhao (2011). Real-time large-deformation substructuring. ACM SIGGRAPH 2011 papers. Vancouver, British Columbia, Canada, ACM**:** 1-8.

Bender, Jan, Dan Koschier, Patrick Charrier and Daniel Weber (2014a). "Position-based simulation of continuous materials." Computers & Graphics **44**: 1-10.

Bender, Jan, Matthias Müller and Miles Macklin (2015). "Position-Based Simulation Methods in Computer Graphics." In Tutorial Proceedings of Eurographics.

Bender, Jan, Matthias Müller, Miguel A. Otaduy, Matthias Teschner and Miles Macklin (2014b). "A Survey on Position-Based Simulation Methods in Computer Graphics." Computer Graphics Forum **33**(6): 228-251.

Biovision BVH File Specifications page, http://www.character-studio.net/bvh_file_specification.htm.

Blender "Blender, free and open source 3D creation suite." https://www.blender.org.

Bloomenthal, Mark (1988). "Approximation of sweep surfaces by tensor product B-splines." Tech Reports UUCS-88-008, University of Utah.

Bonet, J. and R.D. Wood (2008). Nonlinear continuum mechanics for finite element analysis, 2nd Ed., Cambridge university press.

Botsch, Mario, Leif Kobbelt, Mark Pauly, Pierre Alliez and Bruno Lévy (2010). Polygon mesh processing, CRC press.

Bouaziz, Sofien, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise and Mark Pauly (2012). Shape‐Up: Shaping Discrete Geometry with Projections. Computer Graphics Forum, Wiley Online Library.

Bouaziz, Sofien, Sebastian Martin, Tiantian Liu, Ladislav Kavan and Mark Pauly (2014). "Projective dynamics: fusing constraint projections for fast simulation." ACM Transactions on Graphics (TOG) **33**(4): 154.

Bower, Allan F (2009). Applied mechanics of solids, CRC press.

Capell, Steve, Matthew Burkhart, Brian Curless, Tom Duchamp and Zoran Popović (2007). "Physically based rigging for deformable characters." Graphical Models **69**(1): 71-87.

Capell, Steve, Seth Green, Brian Curless, Tom Duchamp and Zoran Popović (2002). "Interactive skeleton-driven dynamic deformations." ACM Trans. Graph. **21**(3): 586-593.

Chao, Isaac, Ulrich Pinkall, Patrick Sanan and Peter Schröder (2010). A simple geometric model for elastic deformations. ACM Transactions on Graphics (TOG), ACM.

Chen, Zhili, Miaojun Yao, Renguo Feng and Huamin Wang (2014). "Physics-inspired adaptive fracture refinement." ACM Transactions on Graphics (TOG) **33**(4): 113.

Choi, Hon Fai and Silvia S Blemker (2013). "Skeletal muscle fascicle arrangements can be reconstructed using a laplacian vector field simulation." PloS one **8**(10): e77576.

Choi, Min Gyu and Hyeong-Seok Ko (2005). "Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation." IEEE Transactions on Visualization and Computer Graphics **11**(1): 91-101.

CMU "CMU Graphics Lab Motion Capture Database." (http://mocap.cs.cmu.edu/).

Cook, Robert D (2007). Concepts and applications of finite element analysis, John Wiley & Sons.

Coros, S., S. Martin, B. Thomaszewski, C. Schumacher, R. Sumner and M. Gross (2012). "Deformable objects alive!" ACM Transactions on Graphics (TOG) **31**(4): 69.

Coumans, Erwin (2010). "Bullet physics engine." Open Source Software: http://bulletphysics. org **1**.

Crane, Keenan, Fernando de Goes, Mathieu Desbrun and Peter Schröder (2013). Digital geometry processing with discrete exterior calculus. ACM SIGGRAPH 2013 Courses, ACM.

Davis, Timothy A (2004). "UMFPACK-an unsymmetric-pattern multifrontal method with a column pre-ordering strategy." ACM Trans. Math. Software **30**(204): 196-199.

Deng, Bailin, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, Yuliy Schwartzburg and Mark Pauly (2015). "Interactive design exploration for constrained meshes." Computer-Aided Design **61**: 13-23.

Deul, Crispin and Jan Bender (2013). "Physically-Based Character Skinning." VRIPHYS **13**: 25-34.

Diziol, R., J. Bender and D. Bayer (2011). Robust real-time deformation of incompressible surface meshes. Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Vancouver, British Columbia, Canada, ACM**:** 237-246.

Fackler, P.L. (2005). "Notes on matrix calculus." North Carolina State University.

Fasshauer, Greg (2007). "Course Notes: Numerical Methods for Differential Equations / Computational Mathematics II, http://www.math.iit.edu/~fass/478_578_handouts.html."

Faure, François, Benjamin Gilles, Guillaume Bousquet and Dinesh K. Pai (2011). "Sparse meshless models of complex deformable solids." ACM Trans. Graph. **30**(4): 1-10.

Felippa, C.A. (2004). "Introduction to finite element methods." University of Colorado, Boulder, http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/.

Foundation, Blender (2008). "Big Buck Bunny." https://peach.blender.org/.

Frâncu, Mihai and Florica Moldoveanu (2015). "Cloth Simulation Using Soft Constraints." Journal of WSCG (Cumulative issue) **Vol.23, No.1-3, ISBN 978-80-86943-64-0**: 9-18.

Gast, Theodore F. and Craig Schroeder (2014). Optimization integrator for large time steps. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Copenhagen, Denmark, Eurographics Association**:** 31-40.

Georgii, Joachim and Rüdiger Westermann (2008). "Corotated Finite Elements Made Fast and Stable." VRIPHYS **8**: 11-19.

Gibson, S.F.F. and B. Mirtich (1997). "A survey of deformable modeling in computer graphics." Technical Report TR-97-1 **9**.

Gilles, Benjamin, Guillaume Bousquet, Francois Faure and Dinesh K. Pai (2011). "Frame-based elastic models." ACM Trans. Graph. **30**(2): 1-12.

Gilles, Benjamin, François Faure, Guillaume Bousquet and Dinesh K Pai (2013). Frame-based interactive simulation of complex deformable objects. Deformation Models, Springer**:** 145-166.

Hahn, Fabian, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros and Markus Gross (2012). "Rig-space physics." ACM Trans. Graph. **31**(4): 1-8.

Hahn, Fabian, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner and Markus Gross (2013). Efficient simulation of secondary motion in rig-space. Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Anaheim, California, ACM**:** 165-171.

Halic, T., S. Kockara, C. Bayrak, R. Rowe and B. Chen (2009). Soft Tissue Deformation and Optimized Data Structures for Mass Spring Methods. Bioinformatics and BioEngineering, 2009. BIBE '09. Ninth IEEE International Conference on.

Harmon, David and Denis Zorin (2013). "Subspace integration with local deformations." ACM Transactions on Graphics (TOG) **32**(4): 107.

Hauser, Kris K, Chen Shen and James F O'Brien (2003). Interactive Deformation Using Modal Analysis with Constraints. Graphics Interface.

Hildebrandt, Klaus, Christian Schulz, Christoph Von Tycowicz and Konrad Polthier (2011). "Interactive surface modeling using modal analysis." ACM Transactions on Graphics (TOG) **30**(5): 119.

Hildebrandt, Klaus, Christian Schulz, Christoph von Tycowicz and Konrad Polthier (2010). Eigenmodes of surface energies for shape analysis. Advances in Geometric Modeling and Processing, Springer**:** 296-314.

Hildebrandt, Klaus, Christian Schulz, Christoph von Tycowicz and Konrad Polthier (2012). "Modal shape analysis beyond Laplacian." Computer Aided Geometric Design **29**(5): 204-218.

Huang, Jin, Xiaohan Shi, Xinguo Liu, Kun Zhou, Baining Guo and Hujun Bao (2006). "Geometrically based potential energy for simulating deformable objects." The Visual Computer **22**(9-11): 740-748.

Huang, Jin, Yiying Tong, Kun Zhou, Hujun Bao and Mathieu Desbrun (2011). "Interactive Shape Interpolation through Controllable Dynamic Deformation." IEEE Transactions on Visualization and Computer Graphics **17**(7): 983-992.

Hunter, Peter and Andrew Pullan (2001). "Fem/bem notes." Department of Engineering Science, The University of Auckland, New Zeland.

Ijiri, T., T. Ashihara, N. Umetani, T. Igarashi, R. Haraguchi, H. Yokota and K. Nakazawa (2012). "A Kinematic Approach for Efficient and Robust Simulation of the Cardiac Beating Motion." PloS one **7**(5): e36706.

Irving, G., J. Teran and R. Fedkiw (2004). Invertible finite elements for robust simulation of large deformation. Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation. Grenoble, France, Eurographics Association**:** 131-140.

Irving, G., J. Teran and R. Fedkiw (2006). "Tetrahedral and hexahedral invertible finite elements." Graph. Models **68**(2): 66-89.

Jakobsen, Thomas (2001). Advanced character physics. Game Developers Conference.

Jernej, Barbič, Shing Sin Fun and Schroeder Daniel (2012). Vega FEM Library: http://www.jernejbarbic.com/vega.

Jernej Barbič, Fun Shing Sin, Daniel Schroeder (2012). Vega FEM Library: http://www.jernejbarbic.com/vega.

Jiménez, Pablo, Federico Thomas and Carme Torras (2001). "3D collision detection: a survey." Computers & Graphics **25**(2): 269-285.

Kavan, Ladislav, Steven Collins, Carol O'Sullivan and Jiri Zara (2006). "Dual quaternions for rigid transformation blending." Trinity College Dublin, Tech. Rep. TCD-CS-2006-46.

Kavan, Ladislav, Steven Collins, Jiří Žára and Carol O'Sullivan (2007). Skinning with dual quaternions. Proceedings of the 2007 symposium on Interactive 3D graphics and games, ACM.

Kavan, Ladislav, Steven Collins, Jiří Žára and Carol O'Sullivan (2008). "Geometric skinning with approximate dual quaternion blending." ACM Transactions on Graphics (TOG) **27**(4): 105.

Kavan, Ladislav and Jiří Žára (2005). Spherical blend skinning: a real-time deformation of articulated models. Proceedings of the 2005 symposium on Interactive 3D graphics and games, ACM.

Kelly, Piaras Solid Mechanics Lecture Notes. Department of Engineering Science, University of Auckland, http://homepages.engineering.auckland.ac.nz/~pkel015/SolidMechanicsBooks/index.html.

Kharevych, Liliya, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E Marsden, Peter Schröder and Matthieu Desbrun (2006). Geometric, variational integrators for computer animation. Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association.

Kim, Junggon and Nancy S. Pollard (2011a). "Fast simulation of skeleton-driven deformable body characters." ACM Trans. Graph. **30**(5): 1-19.

Kim, Theodore and Doug L. James (2011b). Physics-based character skinning using multi-domain subspace deformations. Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Vancouver, British Columbia, Canada, ACM: 63-72.

Kim, YoungBeom and JungHyun Han (2014). "Bulging‐free dual quaternion skinning." Computer Animation and Virtual Worlds **25**(3-4): 321-329.

Koyama, Yuki, Kenshi Takayama, Nobuyuki Umetani and Takeo Igarashi (2012). Real-time example-based elastic deformation. Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation, Eurographics Association.

Lehoucq, Richard B, Danny C Sorensen and Chao Yang (1998). ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods, Siam.

Li, Siwang, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao and Mathieu Desbrun (2014a). "Space-time editing of elastic motion through material optimization and reduction." ACM Transactions on Graphics **33**(4): Art. No. 108.

Li, Yijing, Jernej Barbic, Vladlen Koltun and Eftychios Sifakis (2014b). Stable Orthotropic Materials. Eurographics/ACM SIGGRAPH Symposium on Computer Animation, The Eurographics Association.

Lipman, Yaron, Raif M Rustamov and Thomas A Funkhouser (2010). "Biharmonic distance." ACM Transactions on Graphics (TOG) **29**(3): 27.

Liu, Ning, Xiaowei He, Yi Ren, Sheng Li and Guoping Wang (2012). Physical material editing with structure embedding for animated solid. Proceedings of Graphics Interface 2012. Toronto, Ontario, Canada, Canadian Information Processing Society**:** 193-200.

Liu, Tiantian, Adam W Bargteil, James F O'Brien and Ladislav Kavan (2013). "Fast simulation of mass-spring systems." ACM Transactions on Graphics (TOG) **32**(6): 214.

Macklin, Miles, Matthias Müller, Nuttapong Chentanez and Tae-Yong Kim (2014). "Unified particle physics for real-time applications." ACM Transactions on Graphics (TOG) **33**(4): 153.

Magnenat-Thalmann, Nadia, Richard Laperrire and Daniel Thalmann (1988). Joint-dependent local deformations for hand animation and object grasping. In Proceedings on Graphics interface'88, Citeseer.

Martin, Sebastian, Bernhard Thomaszewski, Eitan Grinspun and Markus Gross (2011). "Example-based elastic materials." ACM Transactions on Graphics (TOG) **30**(4): 72.

Micky Kelager, Sarah Niebe, Kenny Erleben (2010). "A Triangle Bending Constraint Model for Position-Based Dynamics." Proceedings of VRIPHYS'2010. pp.31~37.

Mollemans, W., F. Schutyser, J. Van Cleynenbreugel and P. Suetens (2003). Tetrahedral mass spring model for fast soft tissue deformation. Surgery Simulation and Soft Tissue Modeling, Proceedings. N. Ayache and H. Delingette. **2673:** 145-154.

Morin, David (2008). Introduction to classical mechanics: with problems and solutions, Cambridge University Press.

Movania, Muhammad Mobeen (2011). OpenCloth: A new open source cloth simulation library**:** http://code.google.com/p/opencloth/.

Müller, M. and N. Chentanez (2011). "Solid simulation with oriented particles." ACM Transactions on Graphics (TOG) **30**(4): 92.

Müller, Matthias, Nuttapong Chentanez, Tae-Yong Kim and Miles Macklin (2014). Strain based dynamics. Proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation (SCA), Copenhagen.

Müller, Matthias, Julie Dorsey, Leonard McMillan, Robert Jagnow and Barbara Cutler (2002). Stable real-time deformations. Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM.

Müller, Matthias and Markus Gross (2004). Interactive virtual materials. Proceedings of Graphics Interface. London, Ontario, Canada, Canadian Human-Computer Communications Society**:** 239-246.

Müller, Matthias, Bruno Heidelberger, Marcus Hennix and John Ratcliff (2007). "Position based dynamics." J. Vis. Comun. Image Represent. **18**(2): 109-118.

Müller, Matthias, Bruno Heidelberger, Matthias Teschner and Markus Gross (2005). Meshless deformations based on shape matching. ACM Transactions on Graphics (TOG), ACM.

Müller, Matthias, Jos Stam, Doug James, Nils Th and rey (2008). Real time physics: class notes. ACM SIGGRAPH 2008 classes. Los Angeles, California, ACM**:** 1-90.

Nealen, Andrew, Matthias Mueller, Richard Keiser, Eddy Boxerman and Mark Carlson (2006). "Physically Based Deformable Models in Computer Graphics." Computer Graphics Forum **25**(4): 809-836.

Nocedal, Jorge and Stephen Wright (2006). Numerical optimization, Springer Science & Business Media.

NUS NUS Mocap Database. National University of Singapore, http://animation.comp.nus.edu.sg/nusmocap.html, Created with funding from NUS AcRF R-252-000-429-133.

Ogden, RW (2003). "Nonlinear elasticity, anisotropy, material stability and residual stresses in soft tissue." COURSES AND LECTURES-INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES: 65-108.

Panozzo, Daniele and Alec Jacobson (2015). "libigl tutorial notes." SGP Graduate School 2015.

Pentland, A. and J. Williams (1989). "Good vibrations: modal dynamics for graphics and animation." SIGGRAPH Comput. Graph. **23**(3): 207-214.

Picinbono, G., H. Delingette and N. Ayache (2001). Nonlinear and anisotropic elastic soft tissue models for medical simulation, IEEE.

Picinbono, Guillaume, Hervé Delingette and Nicholas Ayache (2003). "Non-linear anisotropic elasticity for real-time surgery simulation." Graph. Models **65**(5): 305-321.

Rivers, Alec R. and Doug L. James (2007). "FastLSM: fast lattice shape matching for robust real-time deformation." ACM Trans. Graph. **26**(3): 82.

Rohmer, D., A. Sitek and G.T. Gullberg (2007). "Reconstruction and visualization of fiber and laminar structure in the normal human heart from ex vivo diffusion tensor magnetic resonance imaging (DTMRI) data." Investigative radiology **42**(11): 777.

Rumman, Nadine Abu and Marco Fratarcangeli (2014). Position based skinning of skeleton-driven deformable characters. Proceedings of the 30th Spring Conference on Computer Graphics, ACM.

Rumman, Nadine Abu and Marco Fratarcangeli (2015). Position - Based Skinning for Soft Articulated Characters. Computer Graphics Forum, Wiley Online Library.

San-Vicente, Gaizka, Iker Aguinaga and Juan Tomas Celigueta (2012). "Cubical Mass-Spring Model Design Based on a Tensile Deformation Test and Nonlinear Material Model." IEEE Transactions on Visualization and Computer Graphics **18**(2): 228-241.

Schumacher, Christian, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner and Markus Gross (2012). Efficient simulation of example-based materials. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association.

Selle, Andrew, Michael Lentine and Ronald Fedkiw (2008). "A mass spring model for hair simulation." ACM Trans. Graph. **27**(3): 1-11.

Shabana, Ahmed A (1996). Theory of vibration: an introduction, Springer.

Shabana, Ahmed A (2009). Computational dynamics, John Wiley & Sons.

Shoemake, Ken (1985). Animating rotation with quaternion curves. ACM SIGGRAPH Computer Graphics, ACM.

Sifakis, Eftychios and Jernej Barbic (2012). FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. ACM SIGGRAPH 2012 Courses, ACM.

Sin, F., Y. Zhu, Y. Li, D. Schroeder and J. Barbic (2011). Invertible Isotropic Hyperelasticity using SVD Gradients. Posters and Demos, 2011 ACM SIGGRAPH/Eurographics symposium on Computer animation

Sorkine, Olga and Marc Alexa (2007). As-rigid-as-possible surface modeling. Symposium on Geometry processing.

Sorkine, Olga, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl and H-P Seidel (2004). Laplacian surface editing. Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM.

Sosnovik, D.E., R. Wang, G. Dai, T.G. Reese and V.J. Wedeen (2009). "Diffusion MR tractography of the heart." Journal of Cardiovascular Magnetic Resonance **11**(1): 1-15.

Steinemann, Denis, Miguel A. Otaduy and Markus Gross (2008). Fast adaptive shape matching deformations. Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Dublin, Ireland, Eurographics Association**:** 87-94.

Stern, Ari and Mathieu Desbrun (2006). Discrete geometric mechanics for variational time integrators. ACM SIGGRAPH 2006 Courses, ACM.

Stomakhin, A., R. Howes, C. Schroeder and J.M. Teran (2012). Energetically Consistent Invertible Elasticity. Eurographics/ACM SIGGRAPH Symposium on Computer Animation, The Eurographics Association.

TAKAYAMA, K., T. ASHIHARA, T. IJIRI, T. IGARASHI, R. HARAGUCHI and K. NAKAZAWA (2008). "A Sketch-Based Interface for Modeling Myocardial Fiber Orientation that Considers the Layered Structure of the Ventricles." The journal of physiological sciences: JPS **58**(7): 487-492.

Takayama, K., T. Igarashi, R. Haraguchi and K. Nakazawa (2007). A sketch-based interface for modeling myocardial fiber orientation. Smart Graphics, Springer.

Teran, J., S. Blemker, V. Ng Thow Hing and R. Fedkiw (2003). Finite volume methods for the simulation of skeletal muscle. Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. San Diego, California, Eurographics Association**:** 68-74.

Teran, Joseph, Eftychios Sifakis, Geoffrey Irving and Ronald Fedkiw (2005). Robust quasistatic finite elements and flesh simulation. Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. Los Angeles, California, ACM**:** 181-190.

Terzopoulos, Demetri, John Platt, Alan Barr and Kurt Fleischer (1987). "Elastically deformable models." SIGGRAPH Comput. Graph. **21**(4): 205-214.

Teschner, M., B. Heidelberger, M. Muller and M. Gross (2004). A Versatile and Robust Model for Geometrically Complex Deformable Solids. CGI '04: Proceedings of the Computer Graphics International, IEEE Computer Society.

Teschner, Matthias, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M - P Cani, François Faure, Nadia Magnenat - Thalmann and Wolfgang Strasser (2005). Collision detection for deformable objects. Computer Graphics Forum, Wiley Online Library.

Ting, T.C.T. (1996). Anisotropic elasticity: theory and applications, Oxford University Press, USA.

Tournier, Maxime, Matthieu Nesme, François Faure and Benjamin Gilles (2014a). Seamless adaptivity of elastic models. Proceedings of the 2014 Graphics Interface Conference, Canadian Information Processing Society.

Tournier, Maxime, Matthieu Nesme, François Faure and Benjamin Gilles (2014b). "Velocity-based adaptivity of deformable models." Computers & Graphics **45**: 75-85.

Tu, Xiaoyuan (1999). Artificial animals for computer animation: biomechanics, locomotion, perception, and behavior, Springer Science & Business Media.

von Tycowicz, Christoph (2014). Concepts and Algorithms for the Deformation, Analysis, and Compression of Digital Shapes, Freie Universität Berlin.

von Tycowicz, Christoph, Christian Schulz, Hans-Peter Seidel and Klaus Hildebrandt (2013). "An efficient construction of reduced deformable objects." ACM Transactions on Graphics (TOG) **32**(6): 213.

Wang, Wenping, Bert Jüttler, Dayue Zheng and Yang Liu (2008). "Computation of rotation minimizing frames." ACM Transactions on Graphics (TOG) **27**(1): 2.

Ward, Joseph Patrick (1992). Solid mechanics: an introduction, Springer Science & Business Media.

Wood, WL (1990). Practical time-stepping schemes, Clarendon Press Oxford, UK.

Wriggers, Peter and Tod A Laursen (2006). Computational contact mechanics, Springer.

Yang, Yin, Weiwei Xu, Xiaohu Guo, Kun Zhou and Baining Guo (2013). "Boundary-Aware Multi-Domain Subspace Deformation." IEEE Transactions on Visualization and Computer Graphics.

Zhang, Shaoting, Junzhou Huang and Dimitris N Metaxas (2011). "Robust mesh editing using Laplacian coordinates." Graphical Models **73**(1): 10-19.

Zhang, Wenjing, Jianmin Zheng and Nadia Magnenat Thalmann (2015). "Real-Time Subspace Integration for Example-Based Elastic Material." Computer Graphics Forum **34**(2): 395-404.

Zhou, Kun, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo and Heung-Yeung Shum (2005). Large mesh deformation using the volumetric graph laplacian. ACM Transactions on Graphics (TOG), ACM.

Appendix A. Linear FEM Formulation

Material point: $\boldsymbol{X} = (X, Y, Z)$.

Deformed mapping: $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X})$.

Deformation gradient: $F = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{X}} = \frac{\partial (\boldsymbol{X}+\boldsymbol{u})}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + I \in \mathbb{R}^{3\times 3}$.

Strain tensor: $\varepsilon_{mat} \in \mathbb{R}^{3\times 3}$.

A strain energy density function (μ and λ are the *Lam écoefficients*):

$$\psi(F) = \mu\, \varepsilon_{mat} : \varepsilon_{mat} + \frac{1}{2}\lambda\, tr^2(\varepsilon_{mat}).$$

A.1 Strain and Stress

A.1.1 Linear Strain Tensor

Linear strain tensor: $\varepsilon_{mat} = \frac{1}{2}(F + F^{\mathrm{T}}) - I = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}}\right)^{\mathrm{T}} \in \mathbb{R}^{3\times 3}$.

Since $\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}$, we obtain the linear strain tensor as

$$\varepsilon_{mat} := \begin{pmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{31} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) & \frac{\partial v}{\partial y} & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) & \frac{\partial w}{\partial z} \end{pmatrix}.$$

Thus $\varepsilon_{mat} \in \mathbb{R}^{3x3}$ is symmetric.

A.1.2 Strain and Stress in Contracted (Vector) Form

Suppose that

$$\boldsymbol{\varepsilon} = (\varepsilon_{11}\ \varepsilon_{22}\ \varepsilon_{33}\ \gamma_{23}\ \gamma_{31}\ \gamma_{12})^T = (\varepsilon_{11}\ \varepsilon_{22}\ \varepsilon_{33}\ 2\varepsilon_{23}\ 2\varepsilon_{31}\ 2\varepsilon_{12})^T, \tag{7.1}$$

which is the *contracted* form of the strain tensor, then $\psi(F)$ is a quadratic function of the components of $\boldsymbol{\varepsilon}$, and can be re-written as

$$\psi(F) = \frac{1}{2}\boldsymbol{\varepsilon}^T G\boldsymbol{\varepsilon} = \frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\ , \tag{7.2}$$

where the stress vector

$$\boldsymbol{\sigma} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{23}, \sigma_{31}, \sigma_{12})^T = G\boldsymbol{\varepsilon}.$$

A.1.3 Material Stiffness Matrix

Here $G \in \mathbb{R}^{6\times 6}$ is called the ***material stiffness matrix*** (or, *elasticity tensor*), which can be derived as the Hessian matrix of ψ with respect to the strain vector,

$$G = \frac{\partial^2\psi}{\partial\boldsymbol{\varepsilon}^2} = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}.$$

Since we have $\lambda = \frac{\kappa\upsilon}{(1+\upsilon)(1-2\upsilon)}$, $\mu = \frac{\kappa}{1+\upsilon}$, where κ is the Young's modulus, and υ is the Poisson's ratio, then we have

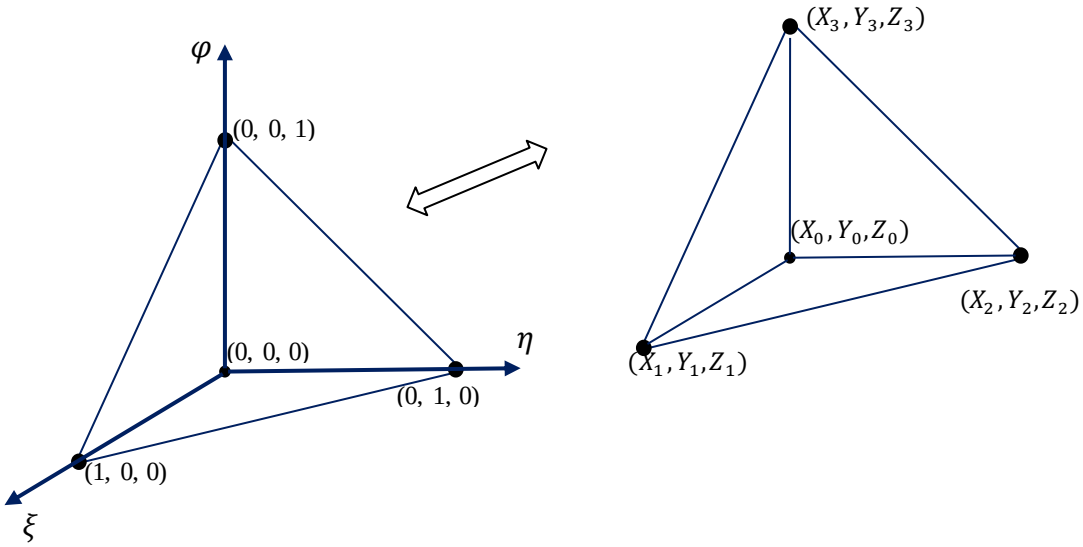
$$G = \frac{\kappa\upsilon}{(1+\upsilon)(1-2\upsilon)}\begin{pmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\upsilon & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\upsilon & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\upsilon \end{pmatrix}.$$

A.2 FEM Discretization: Stiffness Matrix Computation

A tetrahedral element has undeformed position vector $(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3) \in \mathbb{R}^{12\times 1}$, and the displacement vector is $(\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3) \in \mathbb{R}^{12\times 1}$.

A.2.1 Method1: By Strain-Displacement Matrix

A.2.1.1 Element Shape Functions



Isoparametric mapping for a linear tetrahedral element

Natural shape functions $\{N_0, N_1, N_2, N_3\}$:

$$\begin{cases} N_0(\xi,\eta,\varphi) = \varsigma = 1 - \xi - \eta - \varphi \\ N_1(\xi,\eta,\varphi) = \xi \\ N_2(\xi,\eta,\varphi) = \eta \\ N_3(\xi,\eta,\varphi) = \varphi \end{cases}$$

And $0 \le N_i \le 1$, $\sum_{i=0}^{3} N_i = 1$

A.2.1.2 Geometric Interpolation

$$\boldsymbol{X} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \sum_{i=0}^{3} N_i \boldsymbol{X_i} = \begin{pmatrix} N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 \\ 0 & N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 \\ 0 & 0 & N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 \end{pmatrix} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ X_1 \\ Y_1 \\ Z_1 \\ X_2 \\ Y_2 \\ Z_2 \\ X_3 \\ Y_3 \\ Z_3 \end{pmatrix}$$

$$= \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \sum_{i=0}^{3} N_i \boldsymbol{u_i}$$

$$= \begin{pmatrix} N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 \\ 0 & N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 \\ 0 & 0 & N_0 & 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 \end{pmatrix} \begin{pmatrix} u_0 \\ v_0 \\ w_0 \\ u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ u_3 \\ v_3 \\ w_3 \end{pmatrix} = N\boldsymbol{d}\,.$$

Finite elements constructed in this way are known as ***isoparametric finite element***.

A.2.1.3 Linear Strain Vector

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \frac{\partial}{\partial X} & 0 & 0 \\ 0 & \frac{\partial}{\partial Y} & 0 \\ 0 & 0 & \frac{\partial}{\partial Z} \\ \frac{\partial}{\partial Y} & \frac{\partial}{\partial X} & 0 \\ 0 & \frac{\partial}{\partial Z} & \frac{\partial}{\partial Y} \\ \frac{\partial}{\partial Z} & 0 & \frac{\partial}{\partial X} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = L\boldsymbol{u}$$

$$\boldsymbol{\varepsilon} = LN\boldsymbol{d} = B\boldsymbol{d}$$

L: strain operator.

B: strain-displacement matrix.

A.2.1.4 Computation of Strain-Displacement Matrix B

$$B = \begin{pmatrix} \frac{\partial N_0}{\partial X} & 0 & 0 & \frac{\partial N_1}{\partial X} & 0 & 0 & \frac{\partial N_2}{\partial X} & 0 & 0 & \frac{\partial N_3}{\partial X} & 0 & 0 \\ 0 & \frac{\partial N_0}{\partial Y} & 0 & 0 & \frac{\partial N_1}{\partial Y} & 0 & 0 & \frac{\partial N_2}{\partial Y} & 0 & 0 & \frac{\partial N_3}{\partial Y} & 0 \\ 0 & 0 & \frac{\partial N_0}{\partial Z} & 0 & 0 & \frac{\partial N_1}{\partial Z} & 0 & 0 & \frac{\partial N_2}{\partial Z} & 0 & 0 & \frac{\partial N_3}{\partial Z} \\ \frac{\partial N_0}{\partial Y} & \frac{\partial N_0}{\partial X} & 0 & \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial X} & 0 & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial X} & 0 & \frac{\partial N_3}{\partial Y} & \frac{\partial N_3}{\partial X} & 0 \\ 0 & \frac{\partial N_0}{\partial Z} & \frac{\partial N_0}{\partial Y} & 0 & \frac{\partial N_1}{\partial Z} & \frac{\partial N_1}{\partial Y} & 0 & \frac{\partial N_2}{\partial Z} & \frac{\partial N_2}{\partial Y} & 0 & \frac{\partial N_3}{\partial Z} & \frac{\partial N_3}{\partial Y} \\ \frac{\partial N_0}{\partial Z} & 0 & \frac{\partial N_0}{\partial X} & \frac{\partial N_1}{\partial Z} & 0 & \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial Z} & 0 & \frac{\partial N_2}{\partial X} & \frac{\partial N_3}{\partial Z} & 0 & \frac{\partial N_3}{\partial X} \end{pmatrix},$$

Thus we need to compute $\frac{\partial N_i}{\partial X}, \frac{\partial N_i}{\partial Y}, and\ \frac{\partial N_i}{\partial Z}$. Since that

$$\begin{pmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \varphi} \end{pmatrix} = \begin{pmatrix} \frac{dX}{d\xi} & \frac{dY}{d\xi} & \frac{dZ}{d\xi} \\ \frac{dX}{d\eta} & \frac{dY}{d\eta} & \frac{dZ}{d\eta} \\ \frac{dX}{d\varphi} & \frac{dY}{d\varphi} & \frac{dZ}{d\varphi} \end{pmatrix} \begin{pmatrix} \frac{\partial N_i}{\partial X} \\ \frac{\partial N_i}{\partial Y} \\ \frac{\partial N_i}{\partial Z} \end{pmatrix} = J \begin{pmatrix} \frac{\partial N_i}{\partial X} \\ \frac{\partial N_i}{\partial Y} \\ \frac{\partial N_i}{\partial Z} \end{pmatrix},$$

we have

$$\begin{pmatrix} \frac{\partial N_i}{\partial X} \\ \frac{\partial N_i}{\partial Y} \\ \frac{\partial N_i}{\partial Z} \end{pmatrix} = J^{-1} \begin{pmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \varphi} \end{pmatrix}.$$

Since that

$$\begin{aligned} N_0 = \varsigma = 1 - \xi - \eta - \varphi \quad & \rightarrow \frac{\partial N_0}{\partial \xi} = -1, \quad \frac{\partial N_0}{\partial \eta} = -1, \quad \frac{\partial N_0}{\partial \varphi} = -1 \\ N_1 = \xi \quad & \rightarrow \frac{\partial N_1}{\partial \xi} = 1, \quad \frac{\partial N_1}{\partial \eta} = 0, \quad \frac{\partial N_1}{\partial \varphi} = 0 \\ N_2 = \eta \quad & \rightarrow \frac{\partial N_2}{\partial \xi} = 0, \quad \frac{\partial N_2}{\partial \eta} = 1, \quad \frac{\partial N_2}{\partial \varphi} = 0 \\ N_3 = \varphi \quad & \rightarrow \frac{\partial N_3}{\partial \xi} = 0, \quad \frac{\partial N_3}{\partial \eta} = 0, \quad \frac{\partial N_3}{\partial \varphi} = 1 \end{aligned}$$

We have

$$J = \begin{pmatrix} \frac{dX}{d\xi} & \frac{dY}{d\xi} & \frac{dZ}{d\xi} \\ \frac{dX}{d\eta} & \frac{dY}{d\eta} & \frac{dZ}{d\eta} \\ \frac{dX}{d\varphi} & \frac{dY}{d\varphi} & \frac{dZ}{d\varphi} \end{pmatrix} = \begin{pmatrix} \frac{\partial N_0}{\partial \xi} & \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} \\ \frac{\partial N_0}{\partial \eta} & \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} \\ \frac{\partial N_0}{\partial \varphi} & \frac{\partial N_1}{\partial \varphi} & \frac{\partial N_2}{\partial \varphi} & \frac{\partial N_3}{\partial \varphi} \end{pmatrix} \begin{pmatrix} X_0 & Y_0 & Z_0 \\ X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{pmatrix}$$

$$= \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_0 & Y_0 & Z_0 \\ X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \end{pmatrix} = \begin{pmatrix} X_1 - X_0 & Y_1 - Y_0 & Z_1 - Z_0 \\ X_2 - X_0 & Y_2 - Y_0 & Z_2 - Z_0 \\ X_3 - X_0 & Y_3 - Y_0 & Z_3 - Z_0 \end{pmatrix}.$$

From the above we know that J is constant and thus B is constant. And we can obtain $\frac{\partial N_i}{\partial X}, \frac{\partial N_i}{\partial Y}, and\ \frac{\partial N_i}{\partial Z}$ by

$$\begin{pmatrix} \frac{\partial N_i}{\partial X} \\ \frac{\partial N_i}{\partial Y} \\ \frac{\partial N_i}{\partial Z} \end{pmatrix} = J^{-1} \begin{pmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \varphi} \end{pmatrix},$$

For example, $\begin{pmatrix} \frac{\partial N_0}{\partial X} \\ \frac{\partial N_0}{\partial Y} \\ \frac{\partial N_0}{\partial Z} \end{pmatrix} = J^{-1} \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}.$

A.2.1.5 Internal Forces and Stiffness matrix

1) Strain energy:

$$U = \int_\Omega \frac{1}{2}(B\boldsymbol{d})^T G(B\boldsymbol{d})\, d\Omega = \frac{1}{2}\boldsymbol{d}^T \int_\Omega B^T GB\, d\Omega\ \boldsymbol{d}.$$

2) Element internal force:

$$\boldsymbol{f}_{int} = \frac{\partial U}{\partial \boldsymbol{d}} = \int_\Omega B^T GB d\Omega\ \boldsymbol{d} = K^e \boldsymbol{d}.$$

3) Element Stiffness Matrix:

$$K^e = \int_\Omega B^T GB\, d\Omega = V^e (B^T GB).$$

And the volume of undeformed element: $V^e = \frac{1}{6}(\boldsymbol{x}_1 - \boldsymbol{x}_0, \boldsymbol{x}_2 - \boldsymbol{x}_0, \boldsymbol{x}_3 - \boldsymbol{x}_0)$

A.2.2 Method2: By Approximate Deformation Gradient

Deformed position vector $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \in \mathbb{R}^{12\times 1}$

1. Approximated deformation gradient

Frome $d\boldsymbol{x} = F d\boldsymbol{X}$, we have

$$\begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix} = F \begin{pmatrix} X_1 - X_0 & X_2 - X_0 & X_3 - X_0 \\ Y_1 - Y_0 & Y_2 - Y_0 & Y_3 - Y_0 \\ Z_1 - Z_0 & Z_2 - Z_0 & Z_3 - Z_0 \end{pmatrix}.$$

Thus we get

$$F = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix} \begin{pmatrix} X_1 - X_0 & X_2 - X_0 & X_3 - X_0 \\ Y_1 - Y_0 & Y_2 - Y_0 & Y_3 - Y_0 \\ Z_1 - Z_0 & Z_2 - Z_0 & Z_3 - Z_0 \end{pmatrix}^{-1} = D_s \, D_m^{-1}.$$

2. Linear strain

From $\varepsilon_{mat} \coloneqq \frac{1}{2}(F + F^{\mathrm{T}}) - I$, we get the contracted form $\boldsymbol{\varepsilon}$.

3. Strain energy density

$$\psi(F) = \frac{1}{2}\boldsymbol{\varepsilon}^T G \boldsymbol{\varepsilon}$$

4. Strain energy

$$U = V^e \, \psi(F)$$

5. Internal force (on nodes) and stiffness matrix

$$\boldsymbol{f}_{int} = \frac{\partial U}{\partial(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)} \in \mathbb{R}^{12\times 1}$$

$$K^e = \frac{\partial \boldsymbol{f}_{int}}{\partial(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3)} \in \mathbb{R}^{12\times 12}$$

A.2.3 Remarks

Method 1 is more suitable for linear FEM, since the stiffness matrix is constant and independent of the deformed configuration, and thus can be pre-computed; Method 2 depends on the deformed configuration, thus the stiffness matrix cannot be precomputed, which is more suitable for nonlinear FEM, i.e., with the nonlinear Green tensor.

Appendix B. Constitutive Models of Isotropic Elastic Materials

Right Cauchy-Green tensor: $C = F^T F$

Eigenvalues of F: $\lambda_1, \lambda_2, \lambda_3$, which are the principle stretches;

Eigenvalues of C: $\lambda_1^2, \lambda_2^2, \lambda_3^2$, which are the principle directions of C;

Invariants of C:

$$I_1 = tr(C) = C:I = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$$

$$I_2 = tr(CC) = C:C = \lambda_1^4 + \lambda_2^4 + \lambda_3^4$$

$$I_3 = det(C) = J^2 = \det(F)^2 = \lambda_1^2\lambda_2^2\lambda_3^2$$

B.1 The Compressible Neo-Hookean Material

Strain energy density function [Bonet '08]:

$$\psi(I_1, J) = \frac{\mu}{2}(I_1 - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2,$$

Or, equivalently

$$\psi(I_1, I_3) = \frac{\mu}{2}(I_1 - \ln J - 3) + \frac{\lambda}{8}(\ln I_3)^2.$$

B.2 The Mooney-Rivlin Material

Strain energy density function [Bonet '08]:

$$\psi = \mu_{10}(I_1 - 3) + \frac{1}{2}\mu_{01}(I_1^2 - I_2 - 6).$$

Appendix C. Reduced Internal Force and Tangent Stiffness Matrix

C.1 Reduced internal force:

$$\hat{\boldsymbol{f}}(\boldsymbol{q}) = \frac{\partial W}{\partial \boldsymbol{q}} = \frac{\partial}{\partial \boldsymbol{q}} \int_\Omega \Psi \, d\Omega = \int_\Omega \frac{\partial \Psi(\boldsymbol{X})}{\partial \boldsymbol{q}} d\Omega \approx \sum_{i=1}^{n} \omega_i \, g(\boldsymbol{X}_i),$$

where $g = \frac{\partial \Psi(\boldsymbol{X})}{\partial \boldsymbol{q}}$, and the reduced internal force is approximated by n cubature points $\boldsymbol{X}_i$. And we have

$$g = \frac{\partial \Psi(\boldsymbol{X})}{\partial \boldsymbol{q}} = g = \frac{\partial \Psi}{\partial F}\frac{\partial F}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}} = \frac{\partial \Psi}{\partial F}\frac{\partial F}{\partial \boldsymbol{u}} U.$$

Thus finally we get

$$\hat{\boldsymbol{f}}(\boldsymbol{q}) \approx \sum_{i=1}^{n} \omega_i \frac{\partial \Psi(\boldsymbol{X}_i)}{\partial F}\frac{\partial F}{\partial \boldsymbol{u}} U.$$

C.2 Reduced stiffness matrix:

$$\hat{K} = \frac{\partial}{\partial \boldsymbol{q}} \hat{\boldsymbol{f}}(\boldsymbol{q}) \approx \sum_{i=1}^{n} \omega_i \frac{\partial g(\boldsymbol{X}_i)}{\partial \boldsymbol{q}},$$

and we have

$$\frac{\partial g(\boldsymbol{X})}{\partial \boldsymbol{q}} = \frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \Psi}{\partial \boldsymbol{q}}\right) = \frac{\partial}{\partial F}\left(\frac{\partial \Psi}{\partial \boldsymbol{q}}\right)\left(\frac{\partial F}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{q}}\right) = \left(\frac{\partial^2 \Psi}{\partial F^2}\frac{\partial F}{\partial \boldsymbol{q}}\right)^T \left(\frac{\partial F}{\partial \boldsymbol{u}} U\right) = \left(\frac{\partial F}{\partial \boldsymbol{u}} U\right)^T \frac{\partial^2 \Psi}{\partial F^2}\left(\frac{\partial F}{\partial \boldsymbol{u}} U\right).$$

Thus finally we get

$$\hat{K} \approx \sum_{i=1}^{n} \omega_i \left(\frac{\partial F}{\partial \boldsymbol{u}} U\right)^T \frac{\partial^2 \Psi(\boldsymbol{X}_i)}{\partial F^2}\left(\frac{\partial F}{\partial \boldsymbol{u}} U\right).$$