Technische Universität München
Fakultät für Informatik
Lehrstuhl für Computer Graphik und Visualisierung

Interactive Virtual Cutting of Elastically Deformable Bodies

Jun Wu

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 27.10.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.12.2014 angenommen.

To my wife Miao

Abstract

This thesis is concerned with computational problems relevant to simulate elastically deformable bodies with changing topology. In particular, I present interactive techniques for the physically-based simulation of cuts in elastic bodies, for the collision detection among these dynamically separating objects, and for the modeling of residual stress in intact soft tissues to simulate the flap shrinkage after cutting. One motivation of this research is to enhance the functionality and performance of surgery simulators, which are becoming accepted in preoperative surgical planning and in surgical skills training, and the potential of which is yet to be fully exploited.

First, I present a coherent summary of the state-of-the-art on virtual cutting, focusing on the distinct geometrical and topological representations of the deformable body, as well as the specific numerical discretizations of the governing equations of motion.

Second, I present a composite finite element (CFE) formulation for interactive virtual cutting based on linear elasticity and the corotational formulation of strain. Constructed from a semi-regular hexahedral finite element discretization, the composite formulation thoroughly reduces the number of simulation degrees of freedom (DOFs) and thus enables to carefully balance simulation performance and accuracy. In addition, a geometric multigrid solver is employed to solve the governing equations, and a dual contouring approach is utilized to reconstruct a high-resolution surface mesh for visual rendering and collision detection.

Third, I present a collision detection algorithm which effectively exploits the specific characteristics of CFEs, i.e., exhibiting a reduced number of DOFs. The results demonstrate that a reduced number of DOFs leads to a much faster collision detection. Furthermore, I present a topology-aware interpolation scheme to accurately interpolate the penetration depth on complicated boundaries. With this efficient collision detection method, real-time force feedback is obtained for the intuitive manipulation and cutting of elastic bodies through a haptic device.

Fourth, to evaluate the virtual cutting simulator in flap surgery planning, I present an interactive method to introduce a physically meaningful residual stress distribution into a patient-specific model. The residual stress field is computed from a sparse set of directional stress strokes on the surface and stress magnitudes at a few locations specified by experienced surgeons. It is then used to simulate the shrinkage of the flap after cutting. The simulation results show a good match with the shrinkage in a real flap surgery.

Zusammenfassung

Diese Arbeit behandelt Berechnungsprobleme die beim Simulieren elastisch deformierbarer Körper mit sich ändernden Topologien auftreten. Im Besonderen präsentiere ich interaktive Techniken für die physikbasierte Simulation von Schnitten in elastischen Körpern, für die Kollisionserkennung zwischen den dynamisch auseinanderdriftenden Objekten, und für die Modellierung der Ruhespannungen im intakten Weichgewebe für die Simulation des Zusammenziehens eines Gewebelappens nach dem Schneiden. Ein Ziel meiner Forschung ist es, die Funktionalität und Performance von Operationssimulatoren zu verbessern, die zunehmend fr die präoperative Operationsplanung und im Operations-Training eingesetzt werden, und deren Möglichkeiten erst noch voll auszuschöpfen sind.

Erstens präsentiere ich einen zusammenhängenden Überblick über den Stand der Technik im virtuellen Schneiden, mit dem Fokus auf den verschiedenen geometrischen und topologischen Repräsentationen von deformierbaren Körpern, sowie den spezifischen numerischen Diskretisierungen der zugrundeliegenden Bewegungsgleichungen.

Zweitens stelle ich eine Composite-Finite-Element (CFE) -Formulierung für das interaktive Schneiden vor, die auf linearer Elastizität und der co-rotierten Formulierung der Dehnung basiert. Ausgehend von einer semi-regulären Hexaeder-Finite-Elemente-Diskretisierung, reduziert die CFE-Formulierung gezielt die Anzahl der Simulationsfreiheitsgrade (degrees of freedom, DOFs) und ermöglicht es damit vorsichtig Simulationsperformance und -genauigkeit auszubalancieren. Zusätzlich wird ein geometrischer Mehrgitterlöser verwendet um die zugrundeliegenden Gleichungen zu lösen, und es wird ein dualer Konturierungsansatz eingesetzt um ein hochaufgelöstes Oberflächennetz für die visuelle Darstellung und die Kollisionserkennung zu rekonstruieren.

Drittens präsentiere ich einen Kollisionserkennungsalgorithmus der effektiv die spezifischen Eigenschaften von CFEs, d.h. das Aufweisen einer geringeren Anzahl an DOFs, ausnutzt. Die Ergebnisse zeigen, dass eine geringere Anzahl an DOFs zu einer viel schnelleren Kollisionserkennung führt. Außerdem präsentiere ich ein Topologieberücksichtigendes Interpolationsschema um die Eindringtiefe bei komplizierten Rändern genau interpolieren zu können. Mit dieser effizienten Kollisionserkennungsmethode erhält man Echtzeit-Force-Feedback für das intuitive Manipulieren und Schneiden von elastischen Körpern mit einem haptischen Gerät.

Viertens, um die virtuelle Schnitt-Simulation in der Planung von Lappenoperationen zu evaluieren, präsentiere ich eine interaktive Methode um eine physikalisch plausible Ruhespannungsverteilung für ein patientenspezifisches Modell zu generieren. Das Ruhespannungsfeld wird aus einer kleinen Menge an der Oberfläche gezeichneter Spannungsrichtungen und aus an wenigen Stellen von erfahrenen Chirurgen spezifizierten Spannungsstärken berechnet. Es wird dann verwendet um das Schrumpfen des Lappens nach dem Schneiden zu simulieren. Die Simulation zeigt eine gute Übereinstimmung mit dem Schrumpfen in einer realen Lappenoperation.

Acknowledgements

I gratefully acknowledge the support of all of the people who made this thesis possible.

First and foremost, I would like to thank my advisor Prof. Dr. Rüdiger Westermann. Having read several of his articles on GPU computing and cloth simulation, I wrote him an email, asking about the possibility to do research in his group. Back then I was a graduate student at Beihang University, Beijing, China, and I was applying for an Erasmus scholarship to study in Europe. Fortunately, I was selected for the scholarship and was accepted to study at TU München. I would like to thank Prof. Westermann for providing me the research opportunity, for his time commitment and many fruitful discussions on diverse research problems.

A special thanks goes to my colleague Dr. Christian Dick. His work on hexahedral finite elements and geometric multigrid solver was the basis for the developments presented in this thesis. Furthermore, I want to thank all of my current and former colleagues who have always been open for discussions: Dr. Stefan Auer, Dr. Kai Bürger, Shunting Cao, Matthäus Chajdas, Mengyu Chu, Ismail Demir, Dr. Christian Dick, Florian Ferstl, Dr. Roland Fraedrich, Dr. Joachim Georgii, Dr. Tiffany Inglis, Dr. Johannes Kehrer, Mihaela Mihai, Dr. Tobias Pfaffelmoser, Marc Rautenhaus, Florian Reichl, Dr. Jens Schneider, Prof. Dr. Nils Thürey, Dr. Marc Treib, and Dr. Mikael Vaaraniemi.

I would like to thank my collaborators in the research group Computer Aided Plastic Surgery, headed by Prof. Dr. med. Laszlo Kovacs, for discussions and medical data: Dr. med. Maximilian Eder, Jalil Jalali, and Stefan Raith.

I would like to thank my advisors in my former study, Prof. Dr. Yuru Zhang and Prof. Dr. Dangxiao Wang at Beihang University, and Prof. Dr. Charlie C. L. Wang at the Chinese University of Hong Kong, for stimulating my interest on research and for supporting me to pursue a degree in Europe.

Many thanks go to my parents, my wife, and my friends. Studying in Munich is a lot of fun for me, but certainly not for my parents, since it means that their young son is ten thousand miles away and has limited time to accompany them. I want to thank them for always supporting me during my long period of study. Since my parents don't understand English, I would like to switch to Chinese.

“爸爸妈妈，谢谢你们一直以来对我的支持和鼓励！”

I am grateful to my brother Feng Wu for taking care of my parents and my grandma. I am enormously thankful to my wife Miao Feng for traveling such a long distance away from her hometown and staying with me in a foreign country. I thank all of you for your love.

Finally, I am very thankful to the Erasmus Mundus TANDEM program funded by the European Commission for providing me the scholarship to start my research at TU München.

Contents

Chapter 1

Introduction

The research presented in this thesis is concerned with physically-based modeling and simulation, especially in the context of surgery simulation.

Physically-based modeling and simulation deals with mathematical modeling and numerical reproduction of physical phenomena, such as the dynamic behavior of clothes, human tissues, and smokes. It is an active research area in computer graphics, with contributions coming from robotics and computational mechanics as well. Robustness, physical accuracy, and computational efficiency are three important concerns which promote the development of advanced simulation techniques. Recent progress in numerical methods has demonstrated very realistic dynamic simulation effects of a wide variety of physical phenomena. These techniques have been widely employed in entertainment applications such as digital visual effects and computer games, and in medical applications such as surgery simulators.

Surgery simulators are useful in computer-aided preoperative surgical planning. Building on patient-specific data acquired by medical imaging (e.g., CT or MRI), the simulator predicts the surgery outcome by performing an accurate mechanical analysis of human tissues. For example, in orthopedics to predict the stress distribution of the bone after inserting an implant [DGBW08], or in plastic surgery to predict the breast shape after augmentation [GEB+14]. Together with advanced visualization techniques, the surgeon can find an optimal solution by examining the results of using different solution parameters.

Surgery simulators are also employed in virtual reality (VR) based surgical skills training. Usually combined with a force feedback device, the simulator provides realistic and real-time visual and force feedback in response to trainees' operations. For example, in laparoscopic surgery to prepare the trainees for real surgery on the robotic surgical system [Inc], or in dental surgery to train eye-hand coordination and dental drilling skills [WWWZ10]. Historically, it has been expected, since two decades ago [Sat93], that VR simulators will revolutionize the training of young surgeons in a way similar to that flight simulators have been doing for pilot training. The demand of advanced training platforms becomes stronger as the surgery becomes less invasive and operations become more sophisticated. The transferability of skills acquired in virtual environments to the operative setting has been demonstrated in a number of experimental studies [GRC+05,SWC+08]. Surgery simulators as a training platform are becoming accepted adjuncts to traditional training models.

Driven by these applications, surgery simulators continue to evolve in both physical accuracy and computational efficiency. On the one hand, while accuracy is the key factor in designing preoperative planning systems, accelerating the computational speed would greatly enhance their usability by allowing interactive steering of simulation parameters. On the other hand, employing detailed geometric models and applying accurate physical models in virtual reality environments would significantly improve the fidelity of training systems, and thus lead to a better skills transfer. To these ends, robust, physically-based yet efficient simulation techniques are highly demanded, and serve as the motivation of this thesis.

1.1 Objective

The primary goal of this thesis is the development of simulation techniques which support interactive cutting of soft tissues. Cutting is the controlled separation of a physical object as a result of an acutely directed force, exerted through a sharp-edged tool. In typical surgery scenarios, the surgeon uses a scalpel to perform cuts in order to access interior tissues, or to remove tumors from organs. While deformable body simulation has been widely investigated and open source frameworks are available, few can support robust cutting. To enhance the functionality and performance of surgery simulators, the conducted research targets interactive virtual cutting of high-resolution elastic bodies. To enhance the interactivity, it is also desired to allow the user to perform cuts through a haptic device, so as to perceive real-time feedback forces.

The second goal is the exploration of the virtual cutting simulator for flap surgery planning. Flap surgery involves cutting out a section of tissue (called a flap) from a donor site in the body and relocating it to a recipient site. The specific surgery which we are working on is to take a flap from the abdomen in order to reconstruct the shape of the breast after mastectomy—a surgery that removes the breast tissue for the treatment of breast cancer. The purpose of the simulation is to support the flap surgery planning procedure by simulating the process of flap cutting, and in particular to predict the shrinkage of the flap after it has been cut out.

1.2 Challenges

From a technical point of view, virtual cutting involves a) the simulation of the deformable body based on a computational model, b) the incorporation of material discontinuities caused by cutting into the computational model, and c) the detection and handling of collisions among deformable bodies, and with the cutting tool.

The major challenge in virtual cutting is the robust and efficient incorporation of cuts into an accurate computational model that is used for the simulation of the deformable body. The accurate representation of geometrical discontinuities requires the modification of the underlying discretization which is difficult to realize in a robust way. For example, incorporating cuts into a tetrahedral discretization usually ends up with ill-shaped elements, which are known to be problematic in numerical simulation. Furthermore, performing these operations at interactive rates is highly challenging, as for the simulation of cutting most simulation components must be updated or re-computed in each time step, with precomputation in general not being applicable. This includes the update of the finite element model, where elements that are touched by the cutting blade have to be split, the update of the corresponding finite element matrices, the re-assembly of the element matrices into the linear system of equations, and the re-initialization of the numerical solver.

Collision detection is an indispensable component in virtual environments. In the context of virtual cutting, collision detection is notably time consuming, since new geometric primitives are created on-the-fly. These newly created geometric primitives include split volumetric elements which are adapted to represent discontinuity in the object, and newly created surface meshes corresponding to the cutting surfaces. Precomputation of acceleration data structures such as bounding volume hierarchies becomes less effective, since the tree structures need a considerable amount of updating to accommodate new elements. Furthermore, as a consequence of cutting, an object may be split into several separated objects. It is therefore necessary to consistently detect both inter- and intra-collisions. Moreover, a quantitative measure of the penetration is desired for robust collision response.

With respect to the application of virtual cutting in flap surgery planning, a specific challenge is the prediction of the flap volume. In breast reconstruction, the question is how much volume should be taken from the abdomen site such that the volume is minimal but sufficient for reconstructing a natural looking breast shape [KZPB04]. The selection of the volume in the intact abdomen is difficult to plan, since after cutting the flap shrinks. In clinical practice, the surgeon usually cuts out the largest possible portion from the abdomen. This prolongs the healing of the donor site. From a mechanical point of view, the shrinkage is related to the residual stress in the human body. Residual stress is the stress which is present in an object in the absence of external forces. After cutting, the tissue shrinks and the wound opens due to the release of residual stress. Therefore, to accurately predict the shrinkage of the flap, patient-specific models of the residual stress need to be derived and included in the simulation.

1.3 Contributions

State-of-the-art report on virtual cutting. We present a coherent summary of the state-of-the-art in virtual cutting of deformable bodies, focusing on the distinct geometrical and topological representations of the deformable body, as well as the specific numerical discretizations of the governing equations of motion. In particular, we discuss virtual cutting based on tetrahedral, hexahedral, and polyhedral meshes, in combination with standard, polyhedral, composite, and extended finite element discretizations. A separate section is devoted to meshfree methods. The report is complemented with an application study to assess the performance of virtual cutting simulators.

A composite finite element framework for virtual cutting. To enable interactive virtual cutting of high-resolution objects, we present a composite finite element formulation which balances computational performance and accuracy. The deformation is based on linear elasticity and the corotational strain formulation. Geometrically, the composite elements are constructed in a bottom-up manner from a semi-regular hexahedral grid which adaptively refines along a cut, building upon previous work on hexahedral finite element simulation [DGW11a]. Particular attention is given to the analysis of the connectivity among the hexahedral elements, in order to (possibly) duplicate composite elements along complicated boundaries to better preserve the topology on the composite level. Numerically, system matrices of the composite elements are assembled from those of the underlying fine level hexahedral elements. The linear system of equations resulting from the implicit time integration of the dynamic system is solved by using a geometric multigrid solver. Furthermore, we employ the dual contouring approach on the fine resolution level to construct a high quality surface that is accurately aligned with the cuts.

Efficient collision detection for composite finite elements. To simulate the contacts among deformable bodies in the virtual environment, we have developed an efficient collision detection method which is specifically tailored for composite finite element simulation of cuts. This collision detection method consists of a broad phase and a narrow phase. In the broad phase, we employ the spatial subdivision approach to find potentially colliding vertex/composite element pairs. In the narrow phase, in order to evaluate penetration depth and direction, we transform the position of the considered vertex in the deformed configuration back to its position in the reference configuration. To address the non-conforming properties of geometric composition and hexahedral discretization, we propose a topology-aware interpolation approach for the sub-grid accurate computation of penetration depth.

Interactive residual stress modeling. We present an interactive method to compute a physically meaningful residual stress distribution for a patient-specific model. In particular, given a sparse set of eigenvectors and eigenvalues of the stress tensor field at a few locations specified by experienced surgeons, we compute a stress tensor field by smoothly propagating these stress values to regions where stress values are unspecified. The smooth propagation is formulated as a Laplace's equation, subjected to the mechanical equilibrium equations which define the concept of residual stress. The computed residual stress field is immediately visualized by simulating the resulting deformation of a set of round shaped incisions on the surface. We demonstrate the potential of our approach for flap surgery planning by comparing the simulated flap with the shrinkage in a real flap surgery.

1.4 List of Publications

The research results presented in this thesis have been originally published in the following peer-reviewed conference papers and journal articles (listed in chronological order):

1. Jun Wu, Christian Dick, and Rüdiger Westermann, *Interactive high-resolution boundary surfaces for deformable bodies with changing topology*, Proceedings of Workshop on Virtual Reality Interaction and Physical Simulation, 2011, Eurographics Association, pp. 29–38.

2. Jun Wu, Kai Bürger, Rüdiger Westermann, and Christian Dick, *Interactive residual stress modeling for soft tissue simulation*, Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine, 2012, Eurographics Association, pp. 81–89.

3. Jun Wu, Christian Dick, and Rüdiger Westermann, *Efficient collision detection for composite finite element simulation of cuts in deformable bodies*, The Visual Computer (Proc. Computer Graphics International 2013) **29** (2013), no. 6-8, 739–749.

4. Jun Wu, Rüdiger Westermann, and Christian Dick, *Real-time haptic cutting of high-resolution soft tissues*, Studies in Health Technology and Informatics (Proc. Medicine Meets Virtual Reality 21) **196** (2014), 469–475.

5. Jun Wu, Rüdiger Westermann, and Christian Dick, *Physically-based Simulation of Cuts in Deformable Bodies: A Survey*, Proceedings of Eurographics (State-of-the-Art Reports), 2014, Eurographics Association, pp. 1–19.

6. Jun Wu, Rüdiger Westermann, and Christian Dick, *A Survey of Physically Based Simulation of Cuts in Deformable Bodies*, Computer Graphics Forum, accepted (30.11.2014).

1.5 Structure of this Thesis

In the next chapter, we present the fundamentals of the linear elasticity theory, the finite element method, collision detection, and haptic rendering. In Chapter 3 we give an overview of virtual cutting techniques proposed in the computer graphics community. In Chapter 4 we introduce our composite finite element formulation to balance speed and accuracy, and the surface reconstruction method to rebuild a high-resolution render surface. In Chapter 5 we explain the collision detection method which is tailored for the specific composite finite element formulation. In Chapter 6 we present the modeling of residual stress for flap shrinkage simulation. In Chapter 7 we demonstrate the performance of the developed haptic cutting system. In Chapter 8 we summarize our results, and conclude the thesis with an outlook on future research challenges.

Chapter 2

Fundamentals

This chapter presents the fundamentals of theories and techniques that are employed in the subsequent chapters. In particular, we introduce the linear theory of elasticity in Section 2.1, the finite element method in Section 2.2, collision detection in Section 2.3, and haptic rendering in Section 2.4.

2.1 Linear Elasticity Theory

The theory of elasticity studies how elastic materials deform due to the action of external forces. It has been applied in engineering for decades, and primarily in an offline, non-interactive setting. Our interactive simulations build on the basics of elasticity described in this section. For a comprehensive understanding of elasticity, we refer the reader to [Sla02, Cia88].

In the theory of elasticity, a strain tensor field describes the material's local changes of shape, and a stress tensor field describes the internal forces acting in the material. These two fields are related by material models, and satisfy equations of motion, leading to partial differential equations describing the deformation behavior of elastic materials.

2.1.1 Deformation, Strain, and Stress

Given an elastic object in the undeformed *reference configuration* $\Omega \subset \mathbb{R}^3$, the deformed object can be modeled by a *displacement vector field* $\boldsymbol{u}(\boldsymbol{x})$, $\boldsymbol{u} : \Omega \to \mathbb{R}^3$, where $\boldsymbol{x}$ denotes the Euclidean coordinates of a material point in the object's reference configuration (see Figure 2.1). The *deformation* of a material point is described by $\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$, yielding the *deformed configuration* $\{\boldsymbol{\varphi}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \Omega\}$.

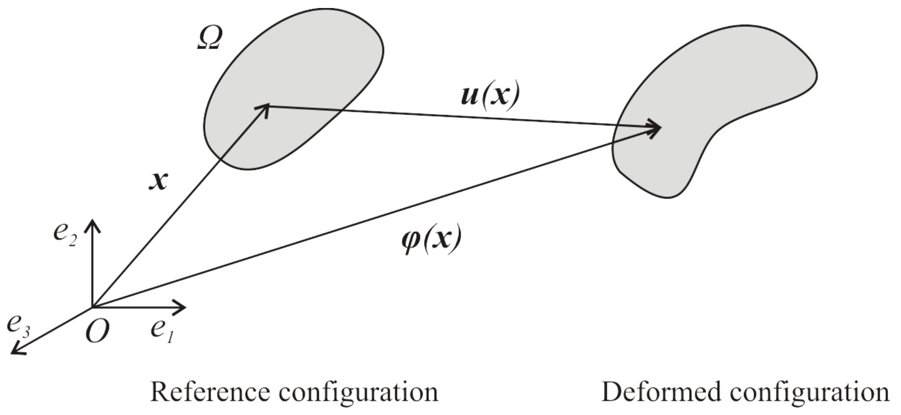


Figure 2.1: *Reference and deformed configuration of an elastic object. The deformation is described for each material point* $\boldsymbol{x}$ *by its deformation* $\boldsymbol{\varphi}(\boldsymbol{x})$*, or interchangeably by its displacement vector* $\boldsymbol{u}(\boldsymbol{x})$*.*

Strain is a measure of deformation, describing the local change of shape. For a spring, the strain can be defined as the ratio of the elongation over the original length. For a 3D object, the strain is represented by a symmetric second-order tensor, known as the *Green-St. Venant strain tensor*,

$$\boldsymbol{E} = \frac{1}{2}\left((\nabla\boldsymbol{\varphi})^{\mathrm{T}}\nabla\boldsymbol{\varphi} - \boldsymbol{I}_{3\times3}\right) = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\mathrm{T}} + (\nabla\boldsymbol{u})^{\mathrm{T}}\nabla\boldsymbol{u}\right), \tag{2.1}$$

where $\nabla\boldsymbol{\varphi}$ is the deformation gradient, and $\boldsymbol{I}_{3\times3}$ is the 3×3 identity matrix. The Green-St. Venant strain tensor is non-linear due to its quadratic term $(\nabla\boldsymbol{u})^{\mathrm{T}}\nabla\boldsymbol{u}$. Its components are

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_{k=1}^{3}\frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j}\right). \tag{2.2}$$

Stress is a measure of internal forces, defined as force per unit area. Given an internal material point represented by an (imaginary) infinitesimal cube, stress can be expressed by forces per unit area acting on faces of the cube, as shown in Figure 2.2. Component σ_{ij} denotes the stress on the i-th face (the face with i-th axis as its normal) along the j-th axis. The stress is represented by a tensor

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}. \tag{2.3}$$

There are actually different formulations of stress depending on in the reference or in the deformed configuration the forces/areas are defined: the Cauchy stress tensor

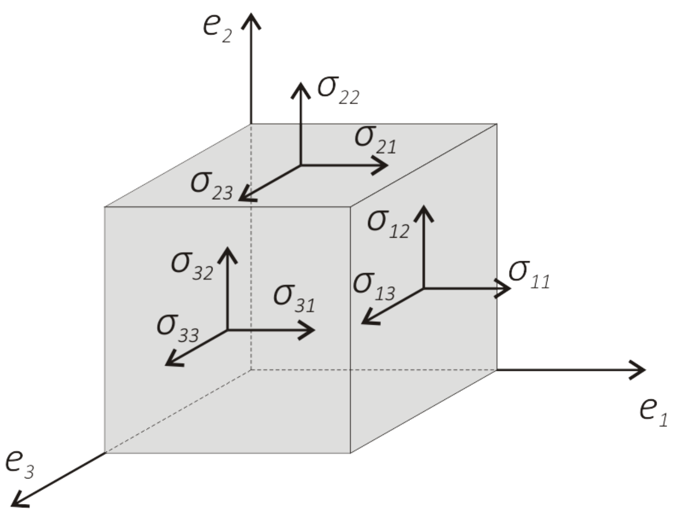


Figure 2.2: *Stress components at an internal material point represented by an (imaginary) infinitesimal cube.*

$\boldsymbol{\sigma}$ (both forces and areas are defined in the deformed configuration), the first Piola-Kirchhoff stress tensor $\boldsymbol{P}$ (forces are defined in the deformed configuration while areas are defined in the reference configuration), and the second Piola-Kirchhoff stress tensor $\boldsymbol{S}$ (both forces and areas are defined in the reference configuration). These formulae can be converted from one to another.

2.1.2 Material Models

The relationship between strain and stress is described by a *material model*. We consider elastic materials, and in particular hyperelastic materials. Elastic materials are materials for which the state of stress depends only on the current deformation. They are an idealization of general materials, i.e., the materials for which the stress depends on as well how the state of deformation is reached (e.g., the strain rate). For elastic materials, the second Piola-Kirchhoff stress is a function of the strain, i.e., $\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{E})$.

Figure 2.3 illustrates stress-strain curves of different material models during loading (i.e., applying a force) and unloading (i.e., removing the force). The elastic model, linear (a) or nonlinear (b), has the same stress-strain relation during loading and unloading. The viscoelastic model (c) has a closed deformation cycle. The area of the closed cycle indicates the energy lost (e.g., heat) during the deformation cycle. The plastic model (d) would have a permanent deformation even after the loading is completely removed. In this thesis, we would assume a linear hyperelastic model, the formulations of which are explained in the next subsection.

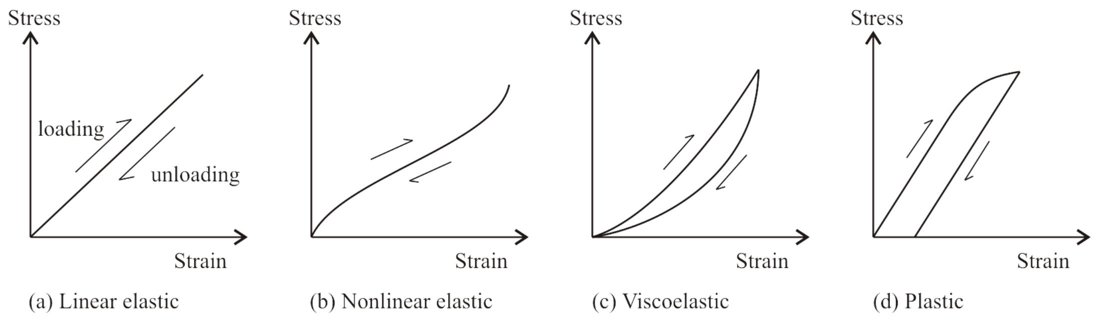


Figure 2.3: *Stress-strain curves of different material models during loading and unloading.*

Hyperelastic materials are a subset of elastic materials for which the elastic strain energy depends only on the current deformation, and does not depend on the deformation history. Strain energy stores (part of) the work done by external forces in deforming the object. For hyperelastic materials, the strain energy density (strain energy per unit volume) is a function of strain, i.e., $W = W(\boldsymbol{E})$. The second Piola-Kirchhoff stress can be derived as $\boldsymbol{S}(\boldsymbol{E}) = \frac{\partial W(\boldsymbol{E})}{\partial \boldsymbol{E}}$.

2.1.3 Linear Elasticity

The linear theory of elasticity is based on the assumption of *small* displacements and a *linear* relationship between stress and strain. This leads to a linear system of equations which can be solved efficiently. By using a corotational strain formulation which removes the per-element rigid body rotation part from the deformation before the strain is computed, the linear theory can also be used to accurately simulate deformations exhibiting large rotations (see Section 2.2.3). Linear elasticity is therefore widely employed in interactive graphics applications.

By assuming that the displacements are small (i.e., $||\nabla \boldsymbol{u}|| \ll 1$), the quadratic term in the Green-St. Venant strain tensor ε_G can be neglected, leading to a linear strain tensor, known as the *infinitesimal strain tensor*

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{T}}\right). \tag{2.4}$$

Under the small deformations assumption, it can be derived that the three stress tensors (the Cauchy stress tensor, and the first and the second Piola-Kirchhoff stress tensor) are equal. We denote the equivalent symmetric second-order stress tensor by $\boldsymbol{\sigma}$.

The linear relationship between stress and strain is described by

$$\boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\varepsilon}, \tag{2.5}$$

where $\boldsymbol{C}$ is the fourth-order *elasticity tensor*. For an *isotropic* material, the deformation is independent of the material's spatial orientation, reducing the stress-strain relationship to the form

$$\boldsymbol{\sigma} = 2\,\mu\,\boldsymbol{\varepsilon} + \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\,\boldsymbol{I}. \tag{2.6}$$

The scalar material parameters λ and μ are referred to as *Lamé constants*. They are related to Young's modulus E and Poisson's ratio ν by

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \; \mu = \frac{E}{2(1+\nu)}. \tag{2.7}$$

Rearranging the entries of the symmetric tensors as vectors, the linear material model can be written as matrix-vector product, e.g., for isotropic materials as

$$\underbrace{\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{23} \end{pmatrix}}_{\bar{\sigma}} = \underbrace{\begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & & & \\ \lambda & 2\mu+\lambda & \lambda & & & \\ \lambda & \lambda & 2\mu+\lambda & & & \\ & & & \mu & & \\ & & & & \mu & \\ & & & & & \mu \end{pmatrix}}_{\bar{C}} \underbrace{\begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{13} \\ 2\varepsilon_{23} \end{pmatrix}}_{\bar{\varepsilon}}. \tag{2.8}$$

2.1.4 Equations of Equilibrium

The static elasticity problem consists of determining a displacement vector field $\boldsymbol{u} : \Omega \to \mathbb{R}^3$ such that at each material point the surplus of the internal body forces—expressed by the divergence of the stress tensor field—balances the external body forces $\boldsymbol{f}_b$ according to

$$-\mathrm{div}\,\boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{u})) = \boldsymbol{f}_b \qquad \text{in} \quad \Omega \setminus \partial\Omega, \tag{2.9}$$

and that the boundary conditions

$$\boldsymbol{u} = \boldsymbol{u}^0 \qquad \text{on} \quad \Gamma_D, \tag{2.10}$$

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{u}))\,\boldsymbol{n} = \boldsymbol{f}_s \qquad \text{on} \quad \Gamma_N, \tag{2.11}$$

on the boundary $\partial\Omega = \Gamma_D \uplus \Gamma_N$ are satisfied, where $\boldsymbol{n}$ is the unit outer normal on $\partial\Omega$. The boundary conditions consist of Dirichlet boundary conditions (Eq. 2.10), which prescribe the displacement $\boldsymbol{u}^0$ on Γ_D, and Neumann boundary conditions (Eq. 2.11), which prescribe surface tractions $\boldsymbol{f}_s$ on Γ_N.

For the simulation of dynamic motion, inertial forces are incorporated into the equilibrium equation. The dynamic elasticity problem is formulated as an initial boundary value problem: Find a time-dependent displacement vector $\boldsymbol{u}(t) : \Omega \to \mathbb{R}^3$ such that

$$\rho\, \ddot{\boldsymbol{u}}(t) - \operatorname{div} \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{u}(t))) = \boldsymbol{f}_b \qquad \text{in} \quad \Omega \setminus \partial\Omega, \tag{2.12}$$

where ρ is the object's density, and $\ddot{\boldsymbol{u}}$ denotes the acceleration. The boundary conditions are analogous to the static case. In addition, initial conditions prescribing the displacement and the velocity of the deformable body at the initial time are required.

2.2 The Finite Element Method

The finite element method (FEM) is one of the most popular approaches to solve the system of partial differential equations arising from the elasticity theory. A detailed explanation of finite element procedures for mechanics can be found in textbooks (e.g., [Bat96]), while a concise introduction of the FEM in medical simulation is given in [BN98].

2.2.1 Weak Formulation of Elasticity Problem

Multiplying the dynamic equilibrium equation (Eq. 2.12) with an arbitrary test function $\boldsymbol{v}$, integrating over the simulation domain Ω, and applying the divergence theorem (integration by parts) leads to the variational formulation of elasticity

$$\int_\Omega \rho \boldsymbol{v} \cdot \ddot{\boldsymbol{u}} \mathrm{d}x + \int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{v}) : \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{u}))\mathrm{d}x - \int_\Omega \boldsymbol{v} \cdot \boldsymbol{f}_b \mathrm{d}x = 0 \quad \forall \boldsymbol{v}. \tag{2.13}$$

For the sake of simplicity, external surface forces are not considered in this continuous formulation (i.e., at this stage we assume $\boldsymbol{f}_s \equiv \boldsymbol{0}$), but can be easily incorporated into the discretized formulation. The formulation for elastostatic simulations can be obtained by omitting the first term corresponding to the inertial forces.

2.2.2 Finite Element Discretization

In the finite element method, the simulation domain Ω is decomposed into a finite set of elements. Typically, triangles or quadrilaterals are employed for discretizing a 2D domain, while tetrahedra or hexahedra are employed in the 3D case.

Building on such a spatial discretization, the FEM approximates the continuous displacement field by means of interpolation of the displacements at the vertices of the finite element grid. Specifically, the *element shape functions* $\phi_i^e(\boldsymbol{x})$ interpolate the displacement field within an element Ω^e from the element's vertices according to

$$\boldsymbol{u}|_{\Omega^e}(\boldsymbol{x}) = \sum_{i=1}^{n_v} \phi_i^e(\boldsymbol{x})\boldsymbol{u}_i^e = \Phi^e(\boldsymbol{x})\boldsymbol{u}^e. \tag{2.14}$$

where n_v is the number of simulation nodes of this element, $\Phi^e(x)$ the element shape matrix

$$\Phi^e(\boldsymbol{x}) = \begin{pmatrix} \phi_1^e(\boldsymbol{x}) & & & & \phi_{n_v}^e(\boldsymbol{x}) & & \\ & \phi_1^e(\boldsymbol{x}) & & \dots & & \phi_{n_v}^e(\boldsymbol{x}) & \\ & & \phi_1^e(\boldsymbol{x}) & & & & \phi_{n_v}^e(\boldsymbol{x}) \end{pmatrix}, \tag{2.15}$$

and $\boldsymbol{u}^e = (\boldsymbol{u}_1^e, \dots, \boldsymbol{u}_{n_v}^e)^{\mathrm{T}}$ the linearization of the displacement vectors at the element's vertices.

We employ trilinear interpolation as shape functions for a hexahedral discretization of 3D objects. Trilinear interpolation fulfills the requirements for finite element simulation: a) Partition of unity: $\Sigma_{i=1}^{n_v}\phi_i(\boldsymbol{x}) = 1$; b) Linear field reproduction: $\Sigma_{i=1}^{n_v}\phi_i(\boldsymbol{x})\boldsymbol{x}_i = \boldsymbol{x}$; and c) Kronecker delta function property:

$$\phi_i(\boldsymbol{x}_j) = \begin{cases} 1.0 & \text{if } j = i, \\ 0.0 & \text{otherwise.} \end{cases} \tag{2.16}$$

Using the shape functions, and expressing the strain by $\overline{\boldsymbol{\varepsilon}}(\boldsymbol{u})|_{\Omega^e}(\boldsymbol{x}) = B^e(\boldsymbol{x})\boldsymbol{u}^e$, where $\boldsymbol{B}^e$ is the *element strain matrix* which contains partial derivatives of the shape function,

$$\boldsymbol{B}^e(x) = \begin{pmatrix} \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_1} & & & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_1} & & \\ & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_2} & & & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_2} & \\ & & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_3} & & & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_3} \\ \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_2} & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_1} & & \dots & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_2} & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_1} & \\ \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_3} & & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_1} & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_3} & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_1} \\ & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_3} & \frac{\partial\phi_1^e(\boldsymbol{x})}{\partial x_2} & & & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_3} & \frac{\partial\phi_{n_v}^e(\boldsymbol{x})}{\partial x_2} \end{pmatrix}, \tag{2.17}$$

it can be derived that the weak formulation Eq. 2.13 leads to an ordinary differential equation describing the dynamics of an element,

$$\underbrace{\int_{\Omega^e} \rho (\Phi^e)^{\mathrm{T}} \Phi^e \mathrm{d}x}_{\boldsymbol{M}^e} \, \ddot{\boldsymbol{u}}^e + \underbrace{\int_{\Omega^e} (\boldsymbol{B}^e)^{\mathrm{T}} \overline{\boldsymbol{C}} \boldsymbol{B}^e \mathrm{d}x}_{\boldsymbol{K}^e} \, \boldsymbol{u}^e = \underbrace{\int_{\Omega^e} (\Phi^e)^{\mathrm{T}} \boldsymbol{f}_b \mathrm{d}x}_{\boldsymbol{f}^e}, \tag{2.18}$$

(*formally* written for a single element). $\boldsymbol{M}^e$, $\boldsymbol{K}^e$, and $\boldsymbol{f}^e$ are called the element mass matrix, the element stiffness matrix, and the element load vector, respectively.

Assembling per-element equations with respect to the global index of the shared nodes yields a linear system of equations for the entire object, i.e., $\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$. It is common to apply a velocity-dependent Rayleigh damping term which leads to

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{D}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}, \tag{2.19}$$

where $\boldsymbol{D}$ is calculated as $\boldsymbol{D} = \alpha \boldsymbol{M} + \beta \boldsymbol{K}$. Non-negative scalars α and β are the mass and the stiffness proportional damping coefficient. Note that Neumann boundary conditions (prescribing tractions) are naturally incorporated into the weak formulation and do not need further treatment. In contrast, Dirichlet boundary conditions (prescribing displacements for specific vertices) must be explicitly enforced by removing these vertices as DOFs from the finite element formulation. To ensure a consistent treatment of free and fixed vertices, these vertices can be later incorporated into the finally resulting linear system of equations by adding the equation $\boldsymbol{I}\boldsymbol{u}_i = \boldsymbol{u}_i^0$ for each fixed vertex i.

2.2.3 Corotational Strain Formulation

The infinitesimal strain tensor works under the assumption of small displacements. It is not rotationally invariant, and interprets rotations as strains, leading to the introduction of artificial stresses. To realistically simulate deformations with large rotations—as they are typically occurring in virtual cutting applications—without resorting to the non-linear Green-St. Venant strain tensor, a rotationally invariant infinitesimal stain formulation [RB86], known as corotational stain formulation, is widely applied in graphics applications [MDM$^+$02] (see Figure 2.4). The idea is to rotate the deformed finite elements back to align with their initial orientation in the reference configuration before the strain is computed. This leads to the equation

$$[\boldsymbol{R}^e] \, \boldsymbol{K}^e \, ([\boldsymbol{R}^e]^{\mathrm{T}}(\boldsymbol{x}^e + \boldsymbol{u}^e) - \boldsymbol{x}^e) = \boldsymbol{f}^e, \tag{2.20}$$

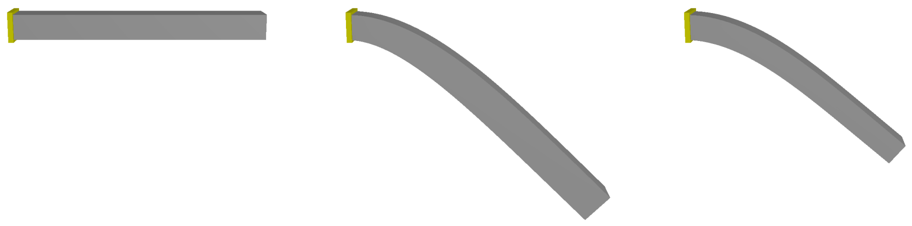


Figure 2.4: *Effects of the corotational strain formulation. A horizontal bar is fixed on its left end (left), and deforms due to gravity. The unrealistic large deformation using the infinitesimal strain tensor (middle) is not found in the simulation using the corotational strain formulation (right).*

(for the static case), where $[\boldsymbol{R}^e]$ denotes a $n_v \times n_v$ block diagonal matrix, and each diagonal entry is the element rotation matrix $\boldsymbol{R}^e$, determined from the current element deformation. The rotation matrix can be computed by minimizing the distance between the rotated deformed and the undeformed configuration [GW08]. Rearranging this formulation for $\boldsymbol{u}^e$ leads to $[\boldsymbol{R}^e]\boldsymbol{K}^e[\boldsymbol{R}^e]^{\mathrm{T}}\boldsymbol{u}^e = \boldsymbol{f}^e + [\boldsymbol{R}^e]\boldsymbol{K}^e(\boldsymbol{x}^e - [\boldsymbol{R}^e]^{\mathrm{T}}\boldsymbol{x}^e)$, which can be incorporated into Eq.2.19. It should be noted that the corotational strain formulation does not change the structure of the system matrix, however, it is required to update the entries at each simulation time step since the element rotations are determined from the deformation of the previous time step.

2.2.4 Time Integration

There exist explicit and implicit time integration schemes to numerically integrate the time-dependent system of ordinary differential equations arising from dynamic simulations of elasticity i.e., Eq. 2.19.

Implicit time integration allows for using a reasonably large time step size dt without introducing numerical problems. Using a finite difference discretization of the time derivatives, the implicit Euler time integration reads

$$\boldsymbol{M}\frac{\boldsymbol{u}^{t+\mathrm{d}t} - 2\boldsymbol{u}^t + \boldsymbol{u}^{t-\mathrm{d}t}}{\mathrm{d}t^2} + \boldsymbol{D}\frac{\boldsymbol{u}^{t+\mathrm{d}t} - \boldsymbol{u}^{t-\mathrm{d}t}}{2\mathrm{d}t} + \boldsymbol{K}\boldsymbol{u}^{t+\mathrm{d}t} = \boldsymbol{f}^{t+\mathrm{d}t}. \tag{2.21}$$

This equation can be rewritten as

$$\tilde{\boldsymbol{K}}\boldsymbol{u}^{t+\mathrm{d}t} = \tilde{\boldsymbol{f}}^{t+\mathrm{d}t}, \tag{2.22}$$

with

$$\begin{aligned}\tilde{\boldsymbol{K}} &= \boldsymbol{K} + \tfrac{1}{\mathrm{d}t^2}\boldsymbol{M} + \tfrac{1}{2\mathrm{d}t}\boldsymbol{D},\\ \tilde{\boldsymbol{f}}^{t+\mathrm{d}t} &= \boldsymbol{f}^{t+\mathrm{d}t} + \boldsymbol{M}\tfrac{2\boldsymbol{u}^t-\boldsymbol{u}^{t-\mathrm{d}t}}{\mathrm{d}t^2} + \boldsymbol{D}\tfrac{\boldsymbol{u}^{t-\mathrm{d}t}}{2\mathrm{d}t}.\end{aligned} \tag{2.23}$$

This linear system of equations can be solved using numerical solvers to be presented in Section 3.5.

Explicit (or forward) Euler time integration scheme approximates the elastic force $\boldsymbol{Ku}$ based on the displacement of previous time step, and thus avoids solving a system of equations. In explicit time integration, the equations of motion is decoupled and each DOF can be evaluated independently. It is written as

$$\boldsymbol{u}^{t+\mathrm{d}t} = \left(\frac{\boldsymbol{M}}{\mathrm{d}t^2} + \frac{\boldsymbol{D}}{2\mathrm{d}t}\right)^{-1}\left(\boldsymbol{f}^{t+\mathrm{d}t} - \boldsymbol{Ku}^t + \boldsymbol{M}\frac{2\boldsymbol{u}^t - \boldsymbol{u}^{t-\mathrm{d}t}}{\mathrm{d}t^2} + \boldsymbol{D}\frac{\boldsymbol{u}^{t-\mathrm{d}t}}{2\mathrm{d}t}\right). \tag{2.24}$$

However, the stability of explicit schemes is only guaranteed for sufficiently small time steps, expressed by the *Courant condition*, which gives an upper limit for the time step size according to

$$\mathrm{d}t < h\sqrt{\frac{\rho}{\lambda + 2\mu}}. \tag{2.25}$$

Here h denotes the smallest distance of two vertices in the reference configuration, ρ is the material density, and λ, μ are the Lamé constants. In general, the stiffer the simulated materials are, the smaller must be the time step size. This implies that a large number of simulation steps are required to advance an interactive simulation. The problem aggravates in the situation of a fine discretization (i.e., a small h). Therefore, from a stability point of view, explicit integration schemes are not well suited for interactive applications which require large time steps (e.g., 10-100 ms).

In our simulations, we employ a second-order accurate Newmark integration scheme. The implicit Newmark integration scheme approximates the time derivatives as

$$\begin{aligned}\dot{\boldsymbol{u}}^{t+\mathrm{d}t} &= \tfrac{2}{\mathrm{d}t}\left(\boldsymbol{u}^{t+\mathrm{d}t} - \boldsymbol{u}^t\right) - \dot{\boldsymbol{u}}^t,\\ \ddot{\boldsymbol{u}}^{t+\mathrm{d}t} &= \tfrac{2}{\mathrm{d}t}\left(\dot{\boldsymbol{u}}^{t+\mathrm{d}t} - \dot{\boldsymbol{u}}^t\right) - \ddot{\boldsymbol{u}}^t \;=\; \tfrac{4}{\mathrm{d}t^2}\left(\boldsymbol{u}^{t+\mathrm{d}t} - \boldsymbol{u}^t - \dot{\boldsymbol{u}}^t\mathrm{d}t\right) - \ddot{\boldsymbol{u}}^t.\end{aligned} \tag{2.26}$$

The dynamic equation can be written in the same form as implicit Euler integration (Eq. 2.22), with

$$\begin{aligned}\tilde{\boldsymbol{K}} &= \boldsymbol{K} + \tfrac{4}{\mathrm{d}t^2}\boldsymbol{M} + \tfrac{2}{\mathrm{d}t}\boldsymbol{D},\\ \tilde{\boldsymbol{f}}^{t+\mathrm{d}t} &= \boldsymbol{f}^{t+\mathrm{d}t} + \boldsymbol{M}\left(\tfrac{4\boldsymbol{u}^t}{\mathrm{d}t^2} + \tfrac{4\dot{\boldsymbol{u}}^t}{\mathrm{d}t^2} + \ddot{\boldsymbol{u}}^t\right) + \boldsymbol{D}\left(\tfrac{2\boldsymbol{u}^t}{\mathrm{d}t} + \dot{\boldsymbol{u}}^t\right).\end{aligned} \tag{2.27}$$

2.3 Collision Detection

The detection of collisions among objects and self-collisions of deformable objects is an essential component in dynamic simulations. In the context of virtual cutting, there are four types of collisions:

i) the collisions between a cutting tool and intact soft tissues,
ii) the collisions between the sweep surface of the cutting front and the internal structures of soft tissues,
iii) the self-collisions of soft tissues, and
iv) the collisions of separated soft tissues.

In this section, we present the basics of some methods which we have employed or extended in the development of our virtual cutting simulator. For an overview of collision detection for deformable objects, we refer to the survey [TKH+05].

With respect to the time dimension, collision detection (CD) is classified into discrete methods and continuous methods. Time discrete CD checks the intersection of objects at a given simulation time t_i, and reports the amount of intersection if any. It is usually employed in simulations which use a constant time step. Collisions are handled during the next simulation time step, for example, by applying a penalty force the magnitude of which is proportional to the amount of intersection. For collisions of type i, iii, and iv listed above, we detect them using discrete methods, and apply penalty forces to resolve collisions.

Time continuous collision detection (CCD) checks the collision of moving objects during the advancement from their configuration at time t_i to their configuration at time t_{i+1}, and reports the specific time $t_i + \delta t$, $(0 \leq \delta t \leq t_{i+1} - t_i)$ at which the contact happens. CCD has a much higher complexity than discrete CD, and is usually employed in offline simulations which adaptively tune the time step in order to accurately resolve collisions. A simplified version of CCD is employed to handle the collisions of type ii: we test static internal structures against the sweep surface of the moving cutting front. The detected intersections are used to disconnect the internal structures and thus to simulate cutting.

Collision detection is usually carried out in two phases: a broad phase which uses inexpensive calculations to cull out collisions than cannot occur, and thus safely reduces the number of elementary tests, and a narrow phase which accurately performs elementary intersection tests. For collisions of type ii, we employ bounding volume hierarchies as the broad phase, and basic line-triangle intersection tests as the narrow

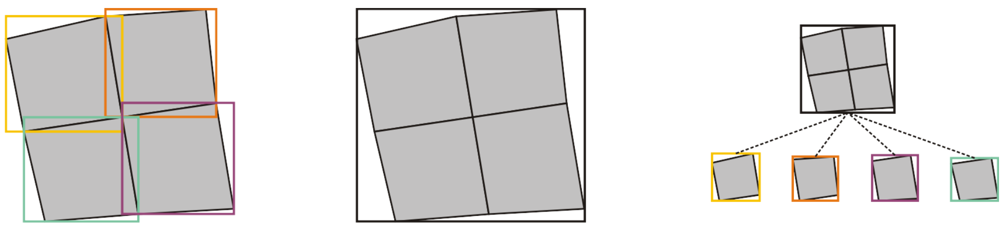


Figure 2.5: *The bounding volume hierarchy for a deformed hexahedral discretization (2D illustration). Left: Axis aligned bounding boxes (colored square frames) for leaf nodes (gray). Middle: The parent bounding box covering a group of leaf nodes. Right: The tree structure encoding the hierarchy.*

phase. For other collision types, we employ a spatial subdivision approach as the broad phase, and rely on a distance field for evaluating the penetration in the narrow phase.

2.3.1 Bounding Volume Hierarchies

Bounding volume hierarchies (BVHs) are an efficient tree data structure for accelerating collision detection in rigid body simulations. The idea of BVHs is to partition the geometric primitives recursively until some leaf criterion is met. The leaf node contains a single primitive (or a small set of primitives), while a non-leaf node represents a tight volume which covers the spatial extend of its children. Collision detection is performed by traversing the tree top-down. Two nodes are checked for overlapping only if their parents do overlap.

For deformable body simulations the BVHs should be updated after deformation. Usually, the tree's structure (i.e., parent-child relation) is maintained the same, while the geometric values of the bounding volumes are updated in a bottom-up manner. On the one hand, the tightness of bounding volumes influences the culling efficiency. On the other hand, for deformable bodies the bounding volumes should be efficiently computed. To this end, we construct a hierarchy using axis aligned bounding boxes (AABBs) for the hexahedral cells (see Figure 2.5 for a 2D illustration), and use it for collision detection against the sweep surface of the cutting front.

2.3.2 Spatial Subdivision

Spatial subdivision approaches partition the 3D space into a number of regions, and check the geometric primitives located in the same region for collision. For example, the 3D space can be subdivided by a uniform Cartesian grid. An interesting approach is to employ a hash function to map 3D grid cells to values in a hash table, rather than

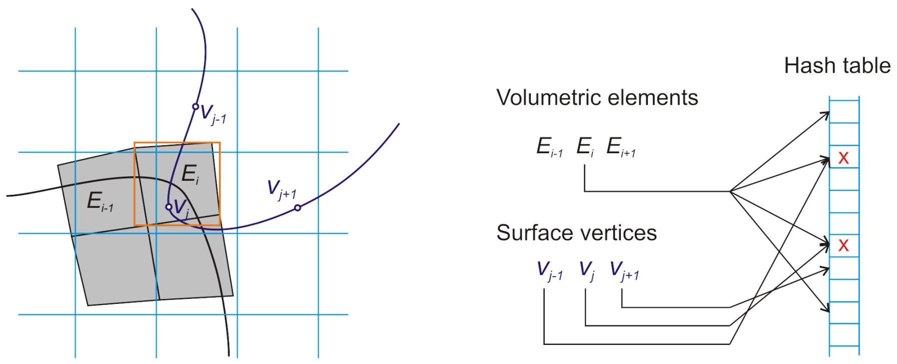


Figure 2.6: *Collision detection using spatial hashing. Volumetric elements (by their bounding boxes) and surface vertices are classified by a uniform grid and mapped to values in a 1D hash table. Elements which are mapped to the same hash value are potentially colliding.*

to maintain a list of all grid cells. This is flexible and memory efficient since this allows for handling potentially infinite regular grids with a sparse distribution of object parts. It was designed for volumetric deformation with tetrahedral elements [THM+03]. We employ this approach since it efficiently detects both collisions and self-collisions in a unified process (see Figure 2.6). The basic algorithm has three passes.

1. All vertices are classified with respect to the 3D grid and mapped to hash values. The hash value for a vertex at the position (x, y, z) is computed as $h : h = hash(\lfloor \frac{x}{l} \rfloor, \lfloor \frac{y}{l} \rfloor, \lfloor \frac{z}{l} \rfloor)$, where l is the length of the uniform classifying grid cell, and $\lfloor \rfloor$ represents the floor function.

2. All volumetric elements (hexahedra in our implementation) are classified with respect to the same grid and mapped to hash values. For a hexahedron the hash values are computed by the same hash function $hash(i, j, k)$ as in the first pass for entries in the set $\{(i, j, k) \mid \lfloor \frac{x_{min}}{l} \rfloor \leq i \leq \lfloor \frac{x_{max}}{l} \rfloor, \lfloor \frac{y_{min}}{l} \rfloor \leq j \leq \lfloor \frac{y_{max}}{l} \rfloor, \lfloor \frac{z_{min}}{l} \rfloor \leq k \leq \lfloor \frac{z_{max}}{l} \rfloor\}$, where the subscripts $_{min}$ and $_{max}$ represent the bounding box values of the hexahedron.

3. For a vertex-hexahedron pair both elements of which are mapped to the same hash value and the vertex does not belong to the hexahedron, a narrow phase performs an accurate penetration test, e.g., by examining the trilinear coordinates of the vertex with respect to the deformed hexahedron.

Parameters of spatial hashing greatly influences the overall performance. We follow the experimental study and analysis conducted in [THM+03] to choose these parame-

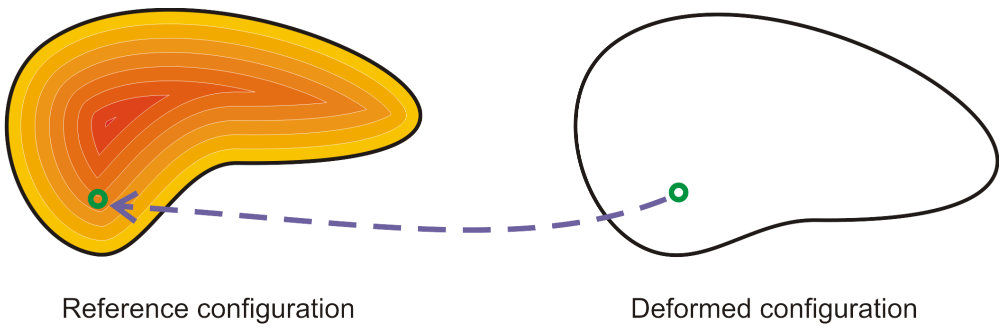


Figure 2.7: *Left: The distance field inside a 2D liver shape is visualized by a color mapping. Right: The distance value at a point in the deformed configuration can be approximated by its material depth, i.e., the distance value in the reference configuration.*

ters. Experiments showed that the hash function

$$h(i, j, k) = (i\,p_1 \text{ xor } j\,p_2 \text{ xor } k\,p_3) \mod n$$

can be efficiently evaluated and have a small number of hash collisions. Additionally, it was found that a good choice of the hash table size n is the number of 3D grid cells, and that an optimal choice of the grid cell length l is the average edge length of all volumetric elements.

2.3.3 Distance Fields

A *distance field* is a scalar field $D : \mathbb{R}^3 \rightarrow \mathbb{R}$ that specifies the minimum distance to the closed surface of an object, where the distance may be signed to distinguish between the inside and outside of the object. The boundary surface is thus implicitly represented as the zero level set, i.e., $S = \{p : D(p) = 0\}$. Given the distance field of a solid object, the evaluation of distances and normals for collision detection and response is very fast and independent of the number of surface triangles.

Material depth is the distance value of points in the interior of an object in its reference configuration. It was proposed to avoid recomputation of distance fields for deformable bodies [HFS+01]. The distance value of a point in the deformed object is approximated by its material depth, i.e., the distance of this material point in its reference configuration (see Figure 2.7).

To generate the distance field of a closed surface mesh, the brute force method is to compute the distance from the center of each cell in a uniform Cartesian grid to the surface mesh. This is accurate but computationally expensive. An alternative and practical solution is the so-called distance transforms [JBS06]. The idea is to estimate the distance value of a cell from the known distance values of its neighbors. In an initialization step, cells which intersect with the surface mesh are assigned with an accurate distance value by directly computing the distance to the surface mesh. In a propagation step, the known distance values at these boundary cells are propagated towards the interior of the object via a cubic template which covers a few cells around the cell to be evaluated.

2.4 Haptic Rendering

Haptic rendering is concerned with conveying information about virtual objects through the sense of touch. The importance of haptic feedback in medical training simulations is well recognized [CMJ11]. Haptic interaction with virtual environments is bidirectional (with respect to the exchange of mechanical energy) via a haptic device. In this thesis, we integrate a haptic device which senses the position and orientation of its stylus-like end-effector which is held by a user, and outputs a 3D force which is computed based on the interaction with simulated objects. This type of devices is called impedance-type force feedback devices. Mechanical impedance refers to the quotient of force and velocity. An ideal impedance-type device should provide an impedance of zero (i.e., no force) when the user moves the stylus in the free space, and an infinitely large impedance when the virtual tool touches a rigid virtual object.

Haptic rendering requires update rates of 1 kHz or higher, in order to achieve (perceived) smooth force feedback [SCB04, OL06a]. It was reported that users can perceive differences at update rates between 500 Hz and 1 kHz [BOYBJK90].

More fundamentally, the update rate is closely related to the stability of the haptic system, i.e., a human-in-the-loop system consisting of three subsystems: the user, the device, and the virtual environment. The virtual environment and the digital control of the mechatronic device are time-discrete. The latency inherent in sampled-data systems can lead to unstable behaviors (e.g., vibrations). According to the passivity-based analysis [CGSS93, CB94], the haptic system maintains stable if all components are passive, i.e., the subsystem does not add energy to the global system. The sufficient condition of passivity for a viscoelastic virtual wall (the simplest virtual environment) is given in [CS97] as

$$\Delta T < \frac{2(b - B)}{K}, \tag{2.28}$$

where ΔT is the sampling period, b is the inherent damping of the device, and K and B are the stiffness and damping of the virtual wall, respectively. This condition implies

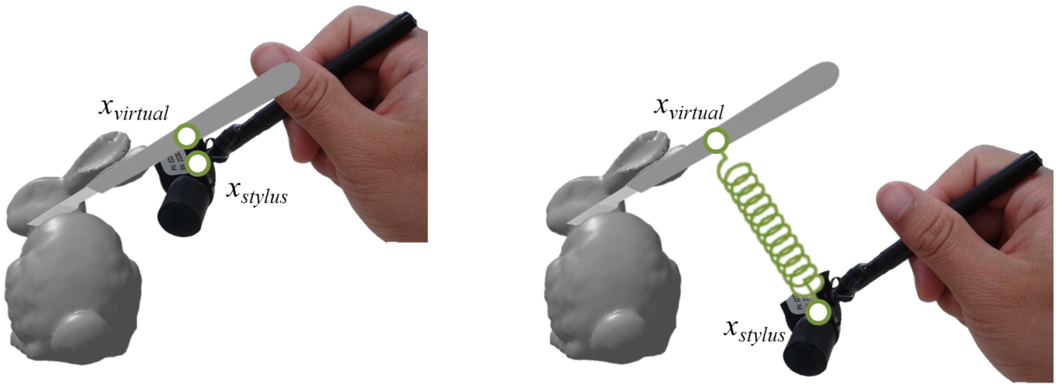


Figure 2.8: *Left: In direct haptic rendering, the position of the virtual tool is mapped directly from the position of the stylus, i.e.,* $\mathbf{x}_{virtual} = \mathbf{x}_{stylus}$*. Right: In simulation-based haptic rendering, the position of the virtual tool is driven by a spring force which tries to align it with the position of the stylus, together with the interaction force exerted by virtual objects. (The displacement between* $\mathbf{x}_{virtual}$ *and* $\mathbf{x}_{stylus}$ *is enlarged for clarity.)*

that a high update rate is very necessary to simulate interaction with stiff virtual objects.

2.4.1 Virtual Coupling

Haptic rendering methods can be classified into direct methods and simulation-based methods [OL06a]. In *direct haptic rendering*, the position of the stylus is mapped directly to a virtual tool (see Figure 2.8 (left)). The interaction force computed based on the contact between the virtual tool and the surrounding virtual environment is directly sent to the haptic device. The direct rendering method is effective if the virtual tool is represented by a single point. Otherwise, there might exist multiple contact regions between the virtual tool and other virtual objects. Since each contact region contributes to the overall interaction force, it is not convenient to modulate the overall stiffness. Actually, there is a stiffness limit for impedance-type haptic devices. Rendering a virtual wall with a stiffness value above this limit results in instability problems of the hardware. For example, the nominal stiffness of the PHANToM Premium 1.5 device used in our experiments is 3.5 N/mm.

In *simulation-based haptic rendering*, the position of the stylus drives the motion of the virtual tool, together with the interaction force between the virtual tool and other virtual objects (see Figure 2.8 (right)). The position of the stylus and the position of the virtual tool are virtually coupled through a spring (and a damper) [MPT99]. The coupling force, instead of the interaction force, is sent to the haptic device. This virtual coupling scheme was originally proposed in [CSB95, AH98] to maintain passivity in virtual manipulation of geometrically complicated and mechanically stiff objects.

With the coupling force $\boldsymbol{f}_{vc}$ which tries to align the position of the virtual tool with the position of the stylus, and the interaction force $\boldsymbol{f}_c$ resulting from contacts with the deformable object, the translational movement of the virtual tool (a virtual scalpel in our simulations) is described by

$$m\,\ddot{\boldsymbol{x}}_{virtual} = \boldsymbol{f}_{vc} + \boldsymbol{f}_c, \tag{2.29}$$

where m is the mass of the virtual tool, and $\boldsymbol{x}_{virtual}$ is the position of the virtual tool. Note that both $\boldsymbol{f}_{vc}$ and $\boldsymbol{f}_c$ are a function of the position of the virtual tool. This differential equation can be numerically integrated using implicit time integration [OL06b, WWWZ10].

The movement of the virtual tool can also be formulated as a static equilibrium problem [WM03]: At each haptic rendering time step, find a position of the virtual tool such that the coupling force balances the interaction force. This approach involves less parameters (e.g., mass, integration time step) and thus is adopted. The first-order approximation of the equilibrium equation [BJ08] is

$$\boldsymbol{f}_{vc} + \boldsymbol{f}_c + \left(\frac{\partial \boldsymbol{f}_{vc}}{\partial \boldsymbol{x}_{virtual}} + \frac{\partial \boldsymbol{f}_c}{\partial \boldsymbol{x}_{virtual}}\right)\Delta\boldsymbol{x}_{virtual} = 0 \tag{2.30}$$

This equation can be efficiently solved for the unknown displacement $\Delta\boldsymbol{x}_{virtual}$ by Gaussian elimination.

Chapter 3

State-of-the-Art Report on Virtual Cutting

Virtual cutting of deformable bodies has been an important and active research topic in physically-based modeling and simulation for more than a decade. A particular challenge in virtual cutting is the robust and efficient incorporation of cuts into an accurate computational model that is used for the simulation of the deformable body.

This report presents a coherent summary of the state-of-the-art in virtual cutting of deformable bodies, focusing on the distinct geometrical and topological representations of the deformable body, as well as the specific numerical discretizations of the governing equations of motion. In particular, we discuss virtual cutting based on tetrahedral, hexahedral, and polyhedral meshes, in combination with standard, polyhedral, composite, and extended finite element discretizations. A separate section is devoted to meshfree methods. Furthermore, we discuss cutting-related research problems such as collision detection and haptic rendering in the context of interactive cutting scenarios. The report is complemented with an application study to assess the performance of virtual cutting simulators.

3.1 Introduction

Physically-based, yet efficient and robust simulation of cutting of deformable bodies (also referred to as virtual cutting) has been an important and active research topic in

This chapter is partially based on material that has been originally published in J. Wu, R. Westermann, and C. Dick, *Physically-based Simulation of Cuts in Deformable Bodies: A Survey*, Proceedings of Eurographics (State-of-the-Art Reports), 2014, pp. 1–19, and on material that has been accepted for publication in J. Wu, R. Westermann, and C. Dick, *A Survey of Physically Based Simulation of Cuts in Deformable Bodies*, Computer Graphics Forum.

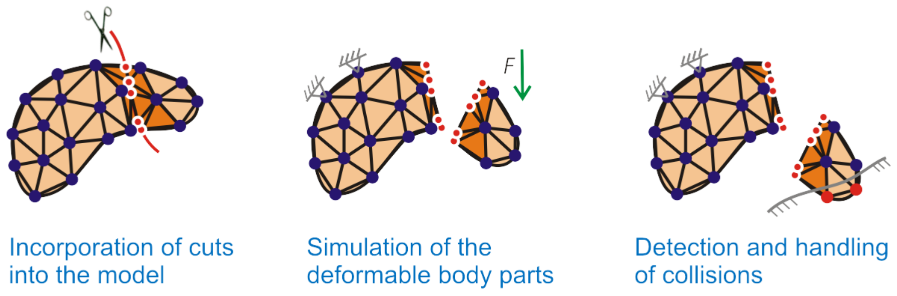

Incorporation of cuts into the model
Simulation of the deformable body parts
F
Detection and handling of collisions

Figure 3.1: *Illustration of the three major tasks involved in mesh-based virtual cutting simulations.*

the computer graphics community for more than a decade. It is at the core of virtual surgery simulators, and it is also frequently used in computer animation. A survey of early cutting techniques has been given 10 years ago by Bruyns et al. [BSM+02], and since then a number of significant improvements with respect to physical accuracy, robustness, and speed have been proposed. Our intention in the current state-of-the-art report is to review the basic concepts and principles underlying these techniques.

Virtual cutting involves three major tasks (illustrated in Figure 3.1): First, the incorporation of cuts into the computational model of the deformable body, i.e., the update of the geometrical and topological representation of the simulation domain as well as the numerical discretization of the governing equations. Second, the simulation of the deformable body based on this computational model. Third, the detection and handling of collisions. Since the basic principles underlying techniques for collision detection in virtual cutting in principle are not different to those used in deformable body simulation, this report summarizes only the particular adaptations that have been proposed in the context of interactive cutting simulation. For a broader overview of the state-of-the-art in this field, including many technical and implementation-specific details, let us refer to the survey by Teschner et al. [TKH+05]. The fracture process driven by cutting tools, on the other hand, is still an open research question, requiring to consider different material properties to predict tissue responses, friction and sliding contacts, as well as accurate force transmission. For a good introduction to the specific problems that have to be addressed to resolve collisions between insertion tools and deformable bodies let us refer to the work by Chentanez et al. [CAR+09].

This report presents a coherent summary of the state-of-the-art in virtual cutting of deformable bodies, focusing on the distinct geometrical and topological representations and the numerical discretizations that have been proposed. The report discusses the different approaches with respect to

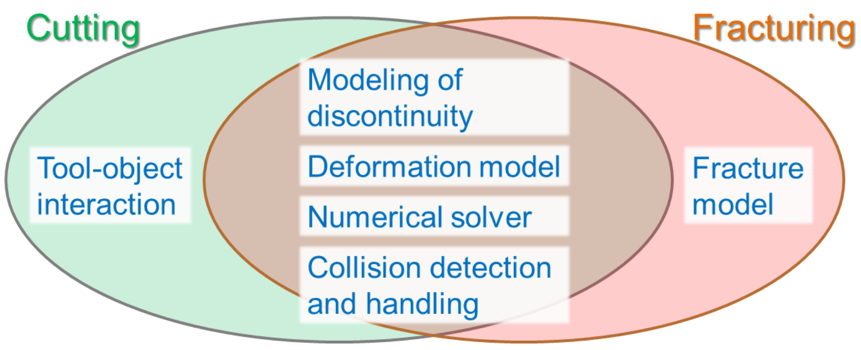


Figure 3.2: *Simulation components in cutting and fracturing. In this report we focus on the components that are common to both simulation tasks.*

- (physical) accuracy, referring to the ability to represent arbitrarily-shaped cuts both in the geometrical and topological representation as well as in the numerical discretization, and to use physically-based simulation to predict the behavior of the cut object;
- robustness, relating to the numerical stability of the involved algorithms in complicated cutting scenarios, such as thin slicing or repeated cutting at the same location; and
- computational efficiency, which is particularly important in real-time applications such as surgery training and planning, where the update of the computational model as well as the deformation computation must be performed within a very limited time budget.

The techniques we discuss in this report are also employed in fracture simulations. While cutting is the controlled separation of a physical object as a result of an acutely directed force, exerted through sharp-edged tools, fracturing refers to the cracking or breaking of (hard) objects, under the action of stress. Fracture simulations build on a fracture model, which determines when and where a crack appears, as well as how the crack propagates through the model. To actually realize the crack, the geometrical and topological representation of the object as well as the numerical discretization of the governing equations have to be updated accordingly, and the dynamics simulation of the cut body has to be performed. The relation between virtual cutting and fracturing is illustrated in Figure 3.2. In this report we focus on reviewing techniques for realizing an actual cut, rather than how the position and shape of a cut is determined. For a thorough introduction to fracture simulation let us refer to [OH99, OBH02], and to a recent survey on fracture modeling [MBP14] which discusses geometry- and image-based approaches as well.

When comparing the individual approaches used for virtual cutting, one of the most apparent classification criteria is the geometrical and topological representation of the simulation domain. In general, this representation is a spatial discretization, as a spatial discretization of the simulation domain—continuously updated according to the introduced cuts—is required for the numerical discretization of the governing equations.

Most approaches are based on a volumetric mesh representation of the object. Early works in the field (e.g., [OH99, BMG99, CDA00, NvdS00, MK00, OBH02]) mainly employ tetrahedral meshes, which offer a high degree of flexibility considering the modeling of cuts by splitting elements or/and snapping element vertices onto the cutting surfaces. Unfortunately, these procedures are prone to producing ill-shaped elements, which are numerically unstable. Recent works address this issue by using regular or semi-regular meshes consisting of hexahedral elements [JBB+10, DGW11a, WDW11, SSSH11]. Some works also consider the use of polyhedral meshes [WBG07, MKB+08]. In addition to mesh-based approaches, meshfree approaches based on particles [MKN+04, PKA+05, SOG06, PGCS09] were proposed.

In order to obtain a physically accurate simulation of the deformable body, the large majority of mesh-based approaches employ the finite element method for the numerical discretization of the governing equations. The straightforward approach is to maintain a 1:1 correspondence between computational elements (finite elements) and geometrical elements (cells) of the underlying mesh. The numerical simulation then is mathematically identical to the simulation of an object without cuts. In particular for interactive applications, however, it is highly desirable to decouple the spatial discretization used for the geometrical and topological modeling of cuts from the spatial discretization employed in the numerical simulation, in order to thoroughly balance speed and accuracy. Approaches that are based on this principle are the extended finite element method [JK09, KMB+09] and the composite finite element method [JBB+10, WDW11].

Using implicit time integration schemes, the numerical discretization leads to a large, sparse linear system of equations in each simulation time step. This system can be solved by using standard black box solvers, such as a conjugate gradient solver. A significantly higher computational efficiency can be achieved by means of problem-specific geometric multigrid solvers [GW06], when these solvers are particularly designed for the efficient treatment of the material discontinuities arising in the context of virtual cutting [DGW11a, WDW11].

A topic beyond the scope of this survey is the realistic texturing of the induced cutting surfaces, such that the body's internal structures are displayed. To this end, 3D solid textures are employed, which can be obtained by texture synthesis [PCOS10] or from slice-based real-world data.

The remainder of this report is organized as follows: The different mesh representations and the respective adaptation strategies used in virtual cutting are discussed in Section 3.2. Finite element methods and meshfree approaches are discussed in Section 3.3 and in Section 3.4 respectively. Numerical solvers are reviewed in Section 3.5. A summary of the surveyed techniques and representative simulation scenarios are presented in Section 3.6, followed by a discussion of techniques for collision detection and haptic rendering in interactive scenarios in Section 3.7. To demonstrate the performance that can be achieved for virtual cutting on desktop PC hardware, we have performed an application study. The results of this study are presented in Section 3.8. The report is concluded in Section 3.9 with a discussion of future research challenges.

3.2 Mesh-based Modeling of Cuts

Virtual cutting of a deformable body is modeled by manipulating the geometrical and topological representation of the simulation domain. In this section, after briefly discussing the modeling of the cutting process, we focus on mesh-based representations, including tetrahedral, hexahedral, and polyhedral meshes, and we discuss the adaptation of these meshes to cuts.

For rendering and collision handling, a surface representation of the object is required. This representation can be directly obtained from a tetrahedral or polyhedral mesh by determining the element faces lying on the surface. For hexahedral meshes, however, a separate surface representation is mandatory to compensate the jagged simulation domain boundary (staircases) resulting from the hexahedral discretization. To this end, cube-based or dual contouring algorithms that reconstruct a smooth surface from the hexahedral mesh were proposed. Sifakis et al. [SDF07] demonstrated how a lower-resolution tetrahedral mesh representing the simulation domain can be combined with a set of given high-resolution surface meshes (original object surfaces and cutting surfaces) for rendering and collision handling.

3.2.1 Geometric Modeling of the Cutting Process

The cutting process is modeled in simulation practice by detecting intersections between the volumetric mesh that represents the deformable object, and a triangulated surface mesh that represents a cutting surface. The cutting surface is generated from the movement of the cutting tool (scalpel). Specifically, element edges, or links between face-adjacent elements are tested against the cutting surface mesh [BMG99, NvdS00, WDW13]. To generate sub-mesh cutting effects such as in polyhedral modeling, element faces are also tested against the cutting mesh [WBG07]. Based on these intersections, elements are split and detached accordingly, as described in the following sections. Since cutting happens locally and advances gradually, a large region of the deformable object can be pruned before elementary intersection tests are performed using bounding volume hierarchies, and a breadth-first traversal of the volumetric mesh starting from previous intersection points is also useful.

The cutting surface normally is the surface swept by the scalpel's cutting edge between two successive simulation frames. Together with 3D spatial interfaces such as a haptic device, this approach enables a natural interaction with the virtual environment. The scalpel may have a complex geometry for visual rendering, comprising a set of triangles. For simplicity, however, the blade that actually cuts the object is usually represented by a single line segment. An open problem is how time-continuous intersection testing between the deformable body and the cutting tool can be realized. In current approaches, the deformable body and the scalpel are moved sequentially within each simulation frame, rather than simultaneously. As a consequence, edges/links might be missed by the cutting tool, especially if the object is moving rapidly.

For non-interactive applications, the cutting surface can also be predefined in the reference configuration [MBF04, KMB+09]. For example, the cutting surface can be constructed from a contour defined on the surface of the deformable object, which is similar to the guide contours defined during preoperative surgery planning [WBWD12]. Prescribing a cutting surface in the reference configuration allows for precisely applying a certain cutting shape, simplifies the intersection tests, and avoids the possible problems with temporally discrete intersection testing. While the simulation of the progressive cutting process is important for animation and interactive applications, in some special cases such as surgery planning, the dynamic process might be of less importance compared to the finally resulting shapes. In these cases the entire cut can be introduced in a single step, which potentially simplifies remeshing operations.

3.2.2 Tetrahedral Meshes

After a brief review of some of the approaches for generating an initial tetrahedral mesh, we introduce and discuss the following techniques for the incorporation of cuts into tetrahedral meshes (see Figure 3.3 for a 2D illustration):

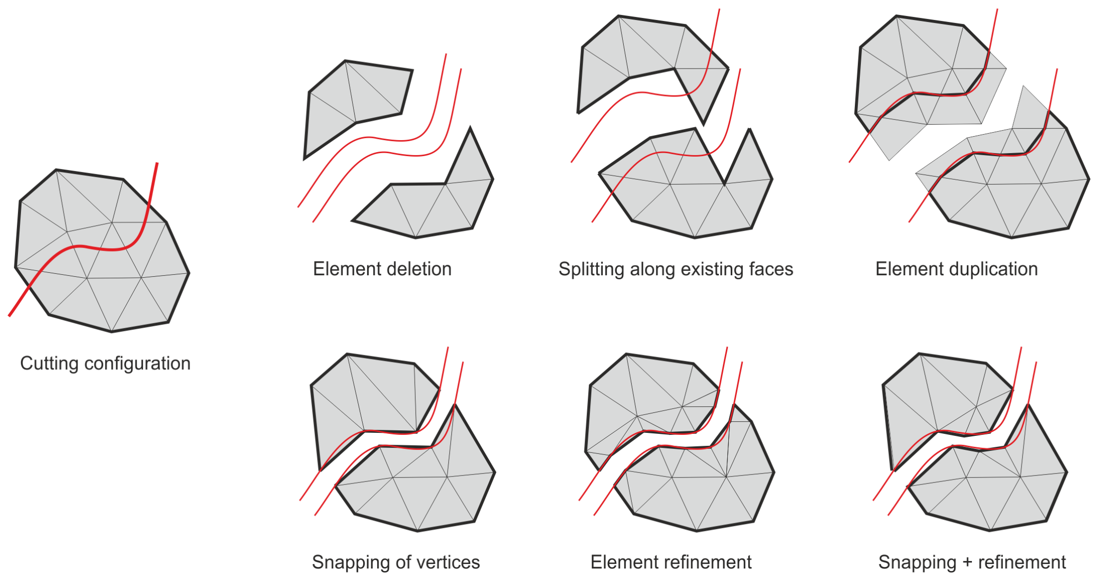


Figure 3.3: *Illustration of different methods for incorporating cuts into a tetrahedral mesh (a triangle mesh in 2D). The red cutting path separates the object into two disconnected parts, which are illustratively displaced to make the discontinuity visible. The surface of the object (bold black line) is given by the set of surface faces of the tetrahedral elements, except for the approach that is based on element duplication, where a separate surface mesh is maintained.*

- Element deletion [CDA00],
- Splitting along existing element faces [NvdS00, MG04, LT07],
- Element duplication [MBF04, SDF07],
- Snapping of vertices [NS01, LJD07],
- Element refinement [BMG99, BG00, MK00, BS01, BGTG04, GCMS00],
- Combined snapping of vertices and element refinement [SHGS06].

One challenge is the accurate representation of arbitrarily-shaped cuts, while avoiding the creation of ill-shaped elements [She02], which lead to numerical instabilities during mesh adaptation and deformation computation. The method of element deletion and the method of splitting along existing element faces maintain the well-shaped elements of the original discretization, but they result in jagged surfaces. By means of snapping of vertices or element refinement, or a combination of both, cuts can be accurately represented. However, since the elements are modified, for these methods it is necessary to prevent ill-shaped elements. The method of element duplication provides a good trade-off between accuracy and robustness by embedding an accurate surface into the duplicated elements in their original shapes.

Tetrahedral Mesh Generation

An initial tetrahedral discretization of the simulation domain can be generated from surface meshes [Si06], medical image data [ZBS05,LZW$^+$14], or level sets [TMFB05]. Quality tetrahedral mesh generation itself remains an active research topic. It is well known that ill-shaped elements (e.g., needle elements, or almost planar sliver elements) lead to numerical instabilities [She02].

Cut Modeling without Creating New Elements

Perhaps the easiest way to incorporate cuts into the deformable body is to separate the material by removing elements that are touched by a cutting tool. While this simple method is widely adopted in real-time simulations (e.g., [CDA00]), it puts severe limitations on the mechanical accuracy and visual quality. First, the newly exposed surface does not conform to the smooth swept surface of a cutting tool, but to the initial discretization of the deformable body, leading to a rather jagged surface. Second, the removal of elements causes a loss of volume, and it leaves unrealistic holes in the object. A remedy to the second problem is to split the object along existing element faces [NvdS00]. This works fine if the cutting surfaces are known a priori to creating the initial discretization [LT07], i.e., the tetrahedralization takes the pre-recorded cutting surface into account. However, for arbitrary cuts, it still results in a jagged surface. To make the newly created surface conforming to cuts, a simple method is to snap the vertices onto the cutting surface before splitting the object along element faces [NS01]. This modification, however, may create ill-shaped elements, which need further treatment afterwards.

Cut Modeling by Element Refinement

To accurately accommodate complex cuts with a reasonable number of initial elements, it is thus necessary to locally refine tetrahedra. Bielser et al. presented a 1:17 subdivision method for tetrahedral decomposition, by generating a vertex on each edge, and a vertex on each triangle face [BMG99]. The exact placement of these vertices depends on the intersection between the cutting tool and the element. Initially, adjacent elements share their vertices. Cutting is modeled by duplicating vertices appropriately. Figure 3.4 (left) illustrates the five topologically different configurations of a tetrahedron

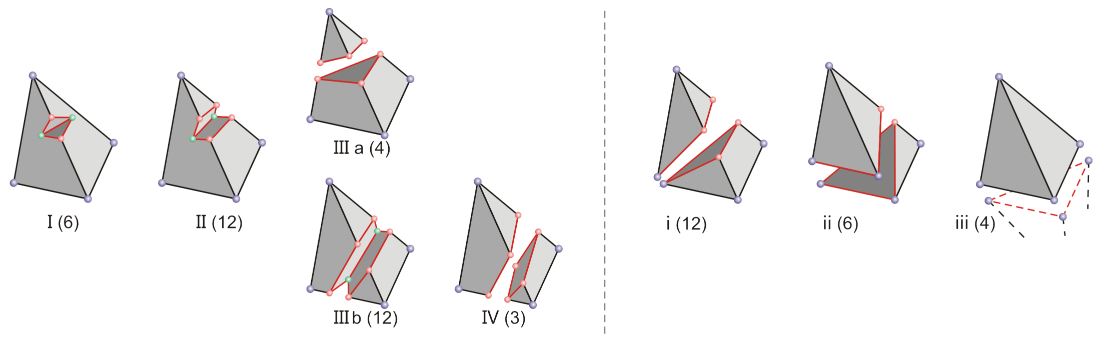


Figure 3.4: *Left: A cut tetrahedron can have five topologically different configurations. The Roman numeral represents the number of disconnected edges. The number in parentheses indicates the number of topologically equivalent configurations by rotation and mirroring operations. Right: In the hybrid cutting approach, three additional topological configurations of a cut tetrahedron are introduced. The small Roman numeral represents the number of existing vertices which are snapped and duplicated.*

after introduction of a cut. Among these five configurations, III a and IV correspond to complete cuts through the tetrahedron, while the other three correspond to partial cuts. For each of these configurations, the information which vertices have to be duplicated in order to generate the respective topological configuration after performing the 1:17 subdivision, is precomputed and stored in a look-up table.

To reduce the number of elements compared to a full 1:17 subdivision, Bielser and Gross subdivided only those edges and faces which are part of the cutting surface [BG00]. Mor and Kanade presented a method for progressive cutting that minimizes the number of newly created elements [MK00], and Ganovelli et al. proposed a multi-resolution approach to reduce the number of elements [GCMS00]. Bielser et al. further proposed a state machine to track the topological configuration of each tetrahedron during progressive cutting [BGTG04].

Considering the decomposition of tetrahedron, if the intersection between an edge and the cutting surface is very close to one of the edge's vertices, ill-shaped elements will occur. Steinemann et al. proposed a combination of snapping of vertices and element refinement to solve this problem [SHGS06]. The idea is illustrated in Figure 3.5 for the 2D case. If a vertex of an intersected edge lies close to the cutting surface (the distance is smaller than a given threshold), the algorithm moves this vertex onto the cutting surface, and separates the material by duplicating the vertex. If the cutting surface intersects an edge close to its midpoint, the edge is split. The method is implemented by extending the set of five topological configurations of a cut tetrahedron, shown in Figure 3.4 (left), by three additional topological configurations, illustrated in Figure 3.4

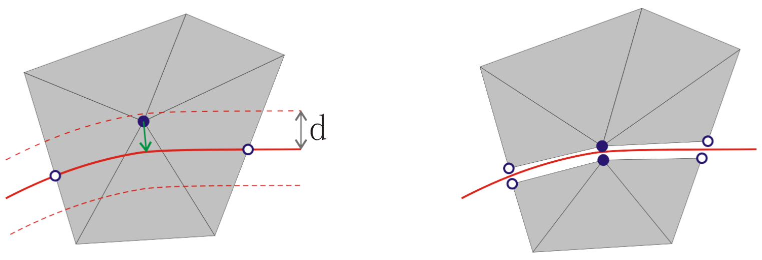


Figure 3.5: *A hybrid cutting approach based on both snapping of vertices and element refinement. If the intersection between an edge and the cutting surface is close to one of the edge's vertices (determined by a threshold d), the vertex is moved onto the cutting surface, in order to prevent the creation of ill-shaped elements. Otherwise the edge is split at the exact intersection point.*

(right). The additional configurations correspond to a complete cut that passes through one, two, or three vertices.

To model a curved cut within a tetrahedral element, given by a sequence of cutting surface triangles, the individual triangles in principle can be successively incorporated into the tetrahedral mesh, leading to a sequence of repeated tetrahedral splits. Since this approach leads to a very large number of tetrahedra along the cut, in practice only a single split of the initial tetrahedron is performed. A curved cut thus is approximated by a only a few tetrahedron faces. The resulting sub-tetrahedra in general are only split if they are intersected by another cut. Also for progressive cutting, the initial tetrahedron is split only once, i.e., when a partial cut is further progressing through a tetrahedron, the current tetrahedral split is undone and replaced by a new split.

Cut Modeling by Element Duplication

Molino et al. proposed the virtual node algorithm to circumvent subsequent numerical problems resulting from ill-shaped elements [MBF04]. The basic idea is to create one or more replicas of the elements that are cut, and to embed each distinct material connectivity component of an element into a unique replica. The replicas comprise both original vertices inside the material (referred to as real nodes) and newly created vertices outside the material (referred to as virtual nodes). Embedding means that the deformation computation is performed on the well-shaped replicas of the original element, and then the displacements of the element's fragments are determined by means of interpolation.

In the initial version of the algorithm, each replica is required to have at least one real node. This was extended by Sifakis et al. to allow for replicas with purely virtual nodes, and thus to support an arbitrary number of fragments within a single tetrahedron [SDF07]. Given a set of triangle surface meshes (original object surfaces and cutting surfaces), enclosed by a tetrahedral mesh that covers the simulation domain, the algorithm first generates a set of non-intersecting polygons from the triangle soup consisting of surface mesh triangles and tetrahedron faces. From these polygons, a polyhedral discretization is determined by examining the connectivity among the polygons. Note that the polyhedra and the tetrahedra per construction do not intersect. Then, for each tetrahedron, the material connectivity components are determined from the polyhedral discretization. For each connectivity component, a duplicate of the tetrahedron is created. In this way, the algorithm enables to combine a lower-resolution tetrahedral mesh for the representation of the simulation domain with high-resolution surface meshes for rendering and collision handling. Note that by means of the duplication of elements, the volumetric representation and the surface representation are topologically consistent.

Wang et al. redeveloped the virtual node algorithm [WJST14]. Their version allows for cuts passing through mesh vertices or lying on mesh edges and faces (without the need of ambiguous perturbation of the cutting surfaces as in the original version), enables multiple cuts per tetrahedron face (at a lower algorithmic complexity than [SDF07]), and includes a mesh intersection routine that is provably robust in the context of floating point rounding errors.

3.2.3 Hexahedral Meshes

A regular or semi-regular hexahedral discretization, generated directly from medical image data [ZBS05] or from polygonal surface meshes by voxelization techniques [ED08, DGBW08], provides an effective means to represent cuts without having to worry about ill-shaped elements [JBB$^+$10, SSSH11]. We discuss the approach of using a linked volume representation, where the connectivity is modeled by links between face-adjacent elements [FG99, DGW11a], and review surface reconstruction techniques to build a smooth surface mesh from the hexahedral grid [WDW11].

Volume Representation

To model cuts in the deformable body, Frisken-Gibson proposed a linked volume representation [FG99]. The basic idea of the linked volume representation is to decompose the object into a set of hexahedral elements, using a uniform hexahedral grid. Face-adjacent elements are connected via links, with six links emanating from each element.

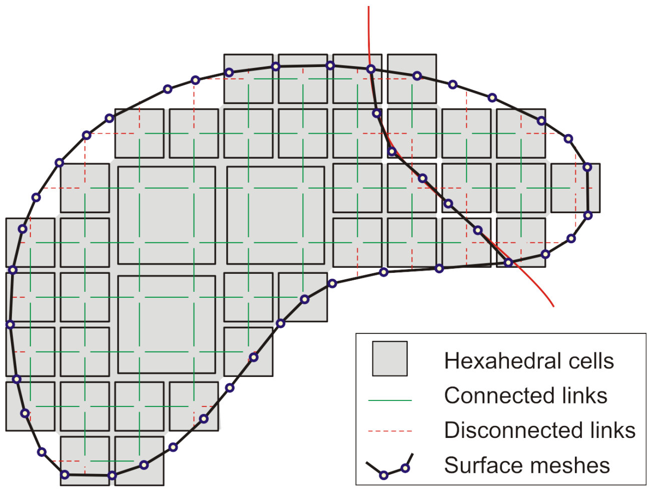


Figure 3.6: *2D illustration of the modeling of cuts in a linked volume representation. The object is discretized by means of an adaptive octree grid (shaded cells). The cells of this grid are connected by links (green, solid). Cutting is modeled by disconnecting links (red, dashed). A surface mesh (black line and dots) is reconstructed from the dual grid of links.*

Cuts are modeled by marking links as disconnected when they are intersected by the virtual cutting blade. Cuts are thus represented at the resolution of the hexahedral grid.

Since the resolution of a uniform grid is in practice limited by simulation time and memory requirements, an adaptive octree grid for virtual cutting was proposed by Dick et al. [DGW11a] (see Figure 3.6), which adaptively refines along cuts, down to a certain finest level. Links are still considered on the uniform grid corresponding to this finest level, but are physically stored only for the elements at the finest level. The adaptive octree grid is constructed by starting from a coarse uniform grid. Whenever a link on the finest level is intersected by the surface of the deformable object, the incident elements (possibly only one element, when both endpoints of the link are lying within the same element) are refined using a regular 1:8 split. At the finest level, links are marked as disconnected when they are intersected by the object's surface. Elements that are lying outside of the object are removed from the representation. To avoid jumps in the discretization, additional splits are performed to ensure that the level difference between elements sharing a vertex, an edge, or a face is at most one (restricted octree). Cuts are modeled analogously to the modeling of the object surface, i.e., elements are adaptively refined along a cut down to the finest level, where links are marked as disconnected (see Figure 3.7 for an example). Material properties such as Young's modulus and density are assigned on a per-element basis. To model inhomogeneous materials, the octree mesh can be refined further.

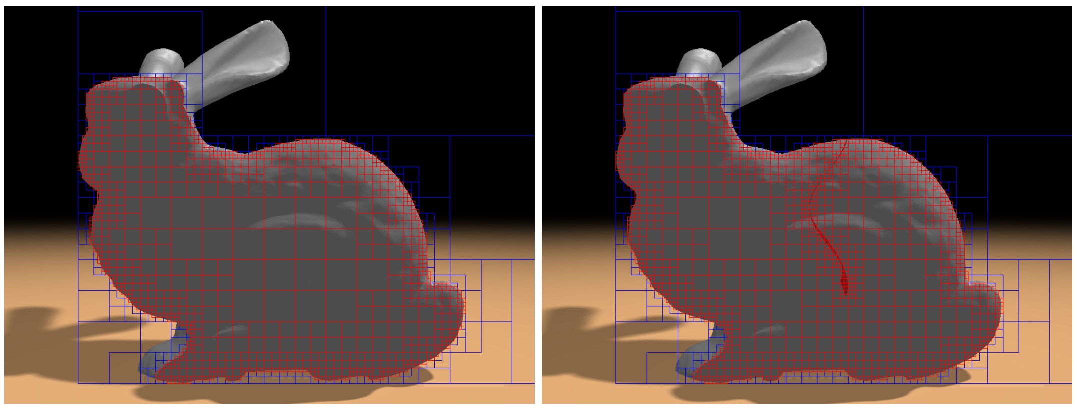


Figure 3.7: *The Stanford bunny model is discretized into a linked octree grid (left), which is refined along the surface and the cuts (right).*

Surface Representation

To render the surface of the deformable object—including the additional surface parts that are generated by cutting—a surface mesh is reconstructed from the volume representation. Wu et al. [WDW11] applied the dual contouring approach [JLSW02] for constructing this surface. Compared to the splitting cubes algorithm [PGCS09], which was used in [DGW11a], dual contouring improves the quality of the generated mesh and reduces the total number of triangles. Dual contouring operates on the (imaginary) grid that is formed by the links between the elements at the finest level. For each link that is cut by the blade, the distances between the intersection point and the link's endpoints as well as the normal of the blade at the intersection point are stored. This information is used to position a surface vertex within each cell that is incident to at least one disconnected link. Since for a cut two surfaces have to be created—one for each material side—this vertex is duplicated, so that for each material component in the cell one vertex exists. The material components in a cell are determined by means of a look-up table, which is indexed by the pattern of connected and disconnected links incident to a cell. After generating the vertices, the surface is spanned by creating two surface patches (2×2 triangles) for each link that is cut. The vertices are finally bound to the nearest element of the respective material part. This binding allows for carrying over the deformation computed at the vertices of the hexahedral simulation mesh to the surface vertices.

In the discussed approaches, the resulting surface is reconstructed from the underlying hexahedral grid. This design thus avoids the explicit cutting of the surface mesh. It has also been adopted for fracturing simulation [HJST13]. A different strategy is to ex-

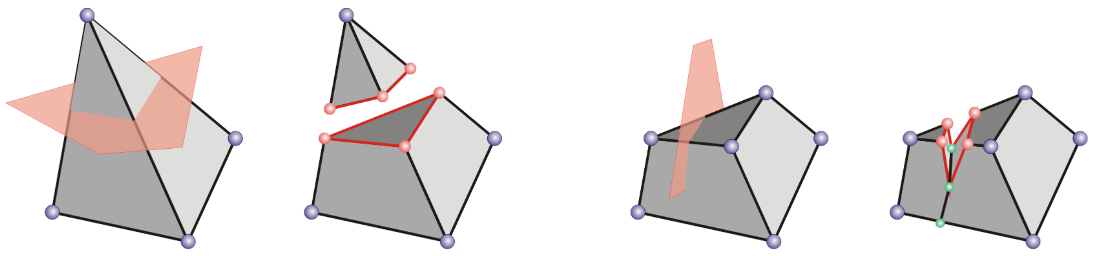


Figure 3.8: *Illustration of cuts in polyhedral elements. Left: A tetrahedron is cut into two parts, resulting in a small tetrahedron and a triangular prism. Right: The triangular prism is partially cut, resulting in two polyhedral elements that are partially connected. Contrary to a tetrahedral discretization, no further subdivision is required.*

plicitly cut the surface mesh, separated from the hexahedral simulation grid. This strategy was followed by Seiler et al. for the simulation of punching operations [SSSH11], which are commonly seen in endoscopic surgery.

3.2.4 Polyhedral Meshes

When cuts are modeled by element refinement, the resulting elements necessarily must be tetrahedra or hexahedra, when a tetrahedral or hexahedral discretization is used. A polyhedral discretization removes this constraint by allowing the creation of general polyhedra. This potentially enables the modeling of cuts by creating a smaller number of new elements [WBG07, MKB+08].

Without loss of generality, polyhedral modeling of cuts can be realized by starting with a purely tetrahedral discretization of the object. To model a complete cut, upon which an element is split into disconnected parts, two new convex elements are created. These resulting elements are composed of vertices of the initial element and the intersections between its edges and the cutting polygon. As illustrated in Figure 3.8 (left), a tetrahedron is split into a small tetrahedron and a triangular prism. Note that no remeshing is needed to decompose the triangular prism into smaller tetrahedra. Figure 3.8 (right) shows the modeling of a partial cut. The intersections between the element's edges and the cutting polygon, and between the element's faces and the cutting polygon's edges are employed as new vertices.

While polyhedral modeling of cuts potentially leads to simplified operations, similar to tetrahedral meshes, there are practical issues with respect to ill-shaped elements. A fundamental problem is that quality criteria of general polyhedral elements are unclear. Wicke et al. found that in particular sliver polyhedra, which are almost planar, lead to numerical problems during simulation, and applied vertex merging and snapping to remove these slivers [WBG07]. Furthermore, to avoid possible numerical problems, it is required to enforce that the elements are convex. These issues and constraints make quality and efficient polyhedralization non-trivial.

3.2.5 Discussion on Discretizations

Avoiding ill-shaped elements is still a major challenge when using a tetrahedral discretization, especially under the constraint of the limited time budget in real-time applications. While a polyhedral discretization offers more flexibility with respect to the shape of individual elements, still special care is required to avoid ill-shaped polyhedra, and also to ensure the convexity of the elements. The virtual node algorithm is superior in this aspect, since it embeds possibly ill-shaped fragments into duplicates of original elements, whose quality can be ensured during preprocessing. Another solution to handle ill-shaped elements is to treat them separately with an alternative numerical strategy, for example with a geometric deformation model. This was studied by Fierz et al. [FSHH12].

Using a semi-regular hexahedral discretization is an effective means to ensure that the elements are well-shaped during dynamic mesh refinement. However, a separate surface representation is required to compensate the jagged nature of the hexahedral grid.

3.3 Finite Element Simulation for Virtual Cutting

In mesh-based cutting approaches, the finite element method is typically used for the numerical discretization of the governing equations of elasticity. The standard approach is to directly employ the spatial discretization that is induced by the mesh, i.e., to create one computational element (finite element) for each geometrical element (cell). The deformable body simulation then is identical to the case without cuts. General simulation of deformable bodies in computer graphics is for example surveyed in [NMK⁺06]. For an introduction of finite elements for elasticity let us refer to Chapter 2[1].

In this section, we discuss three finite element methods that are specialized for simulating cuts in deformable bodies. In particular, we discuss the extended finite element method, the composite finite element method, as well as the polyhedral finite element method. For the first two methods, separate spatial discretizations are employed for the

[1] Unlike in Chapter 2, from this chapter on, for simplicity we do not explicitly distinguish vectors and matrices from scalars by using boldface letters. Nevertheless, the reference is straightforward to interpret from the specific context.

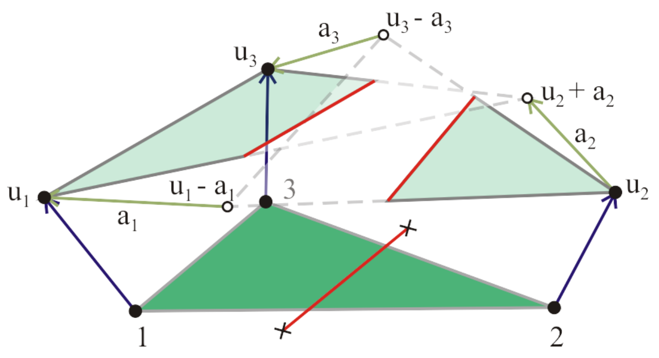


Figure 3.9: *A discontinuous displacement field computed with the extended finite element method. The green triangle domain is divided by a red cut line. The displacements u_i and a_i correspond to the original and added DOFs respectively. Using the shifted Heaviside function as enrichment functions, the displacement field is $u(x) = \Phi^e(x)\,(u_1,\ u_2 + a_2,\ u_3)^{\mathrm{T}}$ on the left side of the cut, and $u(x) = \Phi^e(x)\ (u_1 - a_1,\ u_2,\ u_3 - a_3)^{\mathrm{T}}$ on the right side.*

representation of the simulation domain and for the numerical simulation. This allows for the modeling of complicated-shaped cuts (and also a complicated-shaped original surface of the object), while requiring only a rather small number of computational elements. In this way, these methods enable to carefully balance speed and accuracy, which is particularly important for interactive applications.

For each method, we present its idea and its main components (e.g., the design of shape functions and the construction of element stiffness matrices). In an additional section, we also briefly review the numerical methods for solving the system of equations resulting from finite element discretization and implicit time integration, since the numerical solver is a crucial component considering the overall performance of a cutting application.

3.3.1 The Extended Finite Element Method

The basic idea of the extended finite element method (XFEM) [BB99] is to model material discontinuities introduced by cuts by adapting the basis functions of the finite dimensional solution spaces [BM97, SCB01]. The XFEM was originally invented to accurately simulate material interfaces and crack propagation [MDB99, SMMB00]. The idea was recently utilized for cutting and fracturing deformable objects in graphics applications [LT07, JK09, KMB+09].

In the standard FEM, the displacement within an element is interpolated from the displacements at the element's nodes by using *continuous* shape functions, which are apparently not sufficient to model the discontinuities introduced by cuts. The idea of the XFEM is to introduce *discontinuous* enrichment functions, together with additional degrees of freedom (DOFs) assigned to the original nodes. The displacement field $u(x)$ is computed as

$$u(x) = \Phi^e(x)\, u^e + \underbrace{\Psi^e(x)\Phi^e(x)\, a^e}_{enrichment}, \tag{3.1}$$

where $\Phi^e(x)$ is the standard shape matrix, u^e is the vector of original DOFs, $\Psi^e(x)$ is the element's shape enrichment matrix, which is composed of discontinuous enrichment functions $\psi_i^e(x)$, and a^e represents the newly assigned DOFs.

To make the shape functions fulfill the Kronecker delta property, a good choice for the enrichment functions is the shifted Heaviside function, i.e.,

$$\psi_i^e(x) = \frac{H(x) - H(x_i)}{2}, \tag{3.2}$$

where x_i is the position of the i-th node, and $H(x)$ is the generalized Heaviside function (also known as the sign function)

$$H(x) = \begin{cases} +1 & \text{if } x \text{ is on the cut's left side;} \\ -1 & \text{if } x \text{ is on the cut's right side.} \end{cases} \tag{3.3}$$

As illustrated in Figure 3.9 (for simplicity for a planar element), using the shifted Heaviside function ensures the discontinuity across the cut. Substituting the enrichment functions Eq. 3.2 into Eq. 3.1, in this example the displacement field becomes $u(x) = \Phi^e(x)\,(u_1,\ u_2 + a_2,\ u_3)^{\mathrm{T}}$ on the left side of the cut, and $u(x) = \Phi^e(x)\ (u_1 - a_1,\ u_2,\ u_3 - a_3)^{\mathrm{T}}$ on the right side. Employing the shifted Heaviside function as enrichment functions makes it easy to treat boundary conditions: Since they vanish at the nodes, i.e., $\psi_i^e(x_i) = 0$, the displacement at the position of the i-th node is independent of the additional DOFs a^e. It should be noted that a different selection of the enrichment functions influences the physical meaning of the original and the added DOFs, but leads to the same displacement field.

With the enriched shape functions defined, the enriched element stiffness matrix is computed as

$$^{\mathrm{x}}K^e = \int_{\Omega^e} (^{\mathrm{x}}B^e)^{\mathrm{T}} C (^{\mathrm{x}}B^e) \mathrm{d}x, \tag{3.4}$$

where C represents the material law, and the enriched element strain matrix is composed according to

$$^{\mathrm{x}}B^e = \left(B_1^e,\ \ldots,\ B_{n_v}^e,\ \psi_1^e B_1^e,\ \ldots,\ \psi_{n_v}^e B_{n_v}^e\right). \tag{3.5}$$

The enriched element stiffness matrix has the form

$$^{x}K^{e} = \begin{pmatrix} K^{e,uu} & K^{e,ua} \\ K^{e,au} & K^{e,aa} \end{pmatrix}, \tag{3.6}$$

where the superscripts ^{u} and ^{a} correspond to the original and the added DOFs, respectively. Details that facilitate implementation, as well as enriched element stiffness matrices for the corotational and non-linear strain formulations were derived by Jeřábková et al. [JK09].

While only a single cut is considered above, multiple cuts, in principle, can be supported by further adding more enrichment functions and simulation DOFs. Kaufmann et al. proposed enrichment textures for detailed cutting of shells [KMB+09]. They proposed a harmonic enrichment approach, which uses only one unified kind of enrichment functions to handle multiple, partial, progressive, and complete cuts. While this approach is in general applicable to 3D solids, such a generalization has not been reported yet.

3.3.2 The Composite Finite Element Method

The idea of the composite finite element method (CFEM) [HS97, SW06] is to approximate a high-resolution finite element discretization of a partial differential equation by means of a smaller set of coarser elements. Preusser et al. used the composite finite element method to resolve complicated simulation domains with only a few degrees of freedom, and also to improve the convergence of geometric multigrid methods by an effective representation of complicated object boundaries at ever coarser scales [PRS07, LPR+09]. In computer graphics, Nesme et al. employed the CFEM as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation [NKJF09].

Recently composite finite elements were leveraged in the context of virtual cutting to reduce the number of simulation DOFs [JBB+10, WDW11]. The adoption of the CFEM for cutting simulation is motivated by the following facts. First, using hexahedral discretizations (see Section 3.2.3), an accurate representation of complex cuts typically requires a high-resolution octree grid. Creating a hexahedral simulation element for each octree cell would lead to a very large number of DOFs, exceeding the number of DOFs that can be simulated in real-time. Second, due to its simplicity, the regular or semi-regular hexahedral grid enables an efficient construction of composite finite elements.

A composite finite element is obtained by combining a set of small standard finite elements into a single larger element. In particular, the shape functions of the composite finite element are assembled from the shape functions of the individual elements. The geometrical and topological composition, and the numerical composition of the stiffness matrices are detailed in Section 4.4.

3.3.3 The Polyhedral Finite Element Method

To avoid the remeshing process in standard finite elements, Wicke et al. proposed to directly work on more general convex polyhedral elements [WBG07]. Martin et al. extended this method to support arbitrary convex and concave polyhedral elements with planar (not necessarily triangulated) faces [MKB+08]. Kaufmann et al. further applied the discrete discontinuous Galerkin FEM to arbitrary polyhedra [KMBG08]. These approaches are collectively named here as polyhedral finite element method (PFEM).

Shape functions for polyhedra A key component in the PFEM are valid shape functions defined on the polyhedral domain. They should fulfill the properties of positivity and reproduction of linear polynomials, as required for the convergence of the finite element method [WBG07].

Wicke et al. employed the mean value interpolation function, which is defined as a normalized weight function for each vertex x_i of a convex polyhedron with k vertices according to

$$\phi_i(x) = \frac{w_i(x)}{\sum_{l=1}^{k} w_l(x)}. \tag{3.7}$$

Enumerating x_i's edge-adjacent vertices by x_j, the weight w_i is defined as a weighted sum of ratios of signed tetrahedra volumes by

$$w_i(x) = \sum_j \left(\frac{c_{j,j+1}}{V_{i,j,j+1}} + \frac{c_{i,j} V_{j-1,j+1,j}}{V_{i,j-1,j} V_{i,j,j+1}} \right), \tag{3.8}$$

where $V_{a,b,c}$ represents the volume of the tetrahedron spanned by x_a, x_b, x_c and x, and $c_{a,b}$ is computed as

$$c_{a,b}(x) = \frac{||(x_a - x) \times (x_b - x)||}{6} \arccos\left(\frac{(x_a - x)^{\mathrm{T}}(x_b - x)}{||x_a - x|| \, ||x_b - x||} \right). \tag{3.9}$$

Martin el al. used harmonic shape functions as a generalization of linear tetrahedral shape functions to general polyhedral elements. A shape function is called harmonic if its Laplacian vanishes within the element. With its value fully determined at the nodes (constrained by the Kronecker delta property), the harmonic shape function is uniquely determined. Since closed form expressions for harmonic shape functions do not exist for general polyhedra, they numerically computed the solution of the Laplacian equation using the method of fundamental solutions.

Computation of element matrices In contrast to finite element methods based on tetrahedral and hexahedral elements, an analytical evaluation of the element stiffness matrices for polyhedral elements is non-trivial. To efficiently integrate $(B^e)^{\mathrm{T}} C B^e$ over a polyhedral domain, Wicke et al. approximated the integrals using a small set of sample points p heuristically placed throughout the element, in particular, one integration sample p_i per vertex of the element, and one sample p_f per triangle of the element faces. In their implementation, the per-vertex samples are placed between the element centroid c and the vertex x_i, at $p_i = 0.8x_i + 0.2c$, while the per-triangle samples are located between the element centroid c and the face centroid c_f, at $p_f = 0.9c_f + 0.1c$. Their simulation results show that the exact location of the samples has little influence, and that the difference compared to employing around 10,000 samples per element is subtle.

Integrating over this set of sample points, the element stiffness matrix K^e has the form

$$K^e = \sum_i \frac{\mu_i^e}{2} (B^e(p_i))^{\mathrm{T}} C B^e(p_i) + \sum_f \frac{\kappa_f^e}{2} (B^e(p_f))^{\mathrm{T}} C B^e(p_f). \tag{3.10}$$

Here, μ_i^e and κ_f^e represent the volume fractions associated with the per-vertex integration sample p_i and with the per-triangle sample p_f, respectively. Specifically, enumerating x_i's edge-adjacent vertices by x_j, the volume fraction μ_i^e is defined as

$$\mu_i^e = \frac{\sum_j V(x_i, x_j, x_{j+1}, c)}{3V^e}. \tag{3.11}$$

For a triangle face with vertices x_{j_1}, x_{j_2}, and x_{j_3}, the volume fraction κ_f^e is defined as

$$\kappa_f^e = \frac{V(x_{j_1}, x_{j_2}, x_{j_3}, c)}{V^e}. \tag{3.12}$$

3.3.4 Discussion on Finite Element Methods

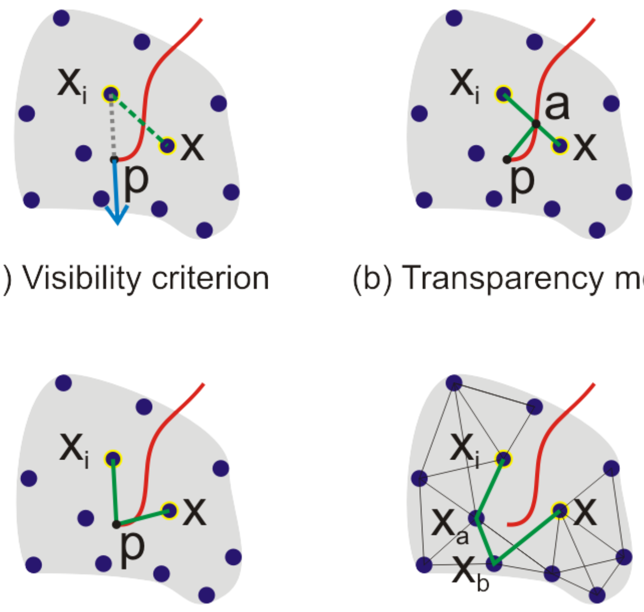
Both the extended finite element method and the composite finite element method are based on using distinct spatial discretizations for the representation of the simulation domain and the numerical discretization. In particular, the spatial discretization that is employed for the numerical discretization does not need to be aligned at the simulation domain boundary. Compared to the standard finite element methods, this enables to reduce the number of computational elements/DOFs along the boundary, and thus to balance speed and accuracy. Both approaches are based on using duplicated DOFs at the same location in order to correctly model the topology of the simulation domain. Whereas the extended finite element method directly duplicates the DOFs at the vertices of the original element, the composite finite element method is based on duplicating elements, which implicitly leads to a duplication of DOFs.

It is worth noting that the virtual node algorithm [MBF04, SDF07] described in Section 3.2.2 is related to these approaches, in that it is also based on the duplication of elements and thus DOFs in order to correctly represent the topology of the simulation domain. However, since in the virtual node algorithm the duplicated elements are assigned the (standard) element matrices of the original elements before cutting, the distribution of the material to the distinct sides of a cut is not modeled accurately. This is in contrast to the extended and the composite finite element method, where the material boundaries (including those resulting from cutting) are accurately represented by using specialized element matrices, i.e., the construction of these matrices takes the exact material boundaries into account.

3.4 Meshfree Methods

In contrast to finite element methods, meshfree methods (also known as meshless methods) do not require a simulation grid. Instead, the material is represented by a set of moving simulation nodes, which interact with each other according to the governing equations of elasticity. From computational mechanics, reviews of meshfree methods for cutting and fracturing can be found in [NRBD08, RBZ10]. The advantage of meshfree methods is that they do not need an explicit encoding of the material topology and can be used even in scenarios where the connectivity of the nodes is difficult to maintain without introducing errors [NTV92]. On the downside, meshfree methods need to compute node-to-node adjacency in every simulation step, making it necessary to maintain and update an additional search data structure.

In computer graphics, Desbrun and Cani are among the first to employ the concepts underlying meshfree methods for deformable body simulation. They animated soft substances that can split and merge by combining particle systems with inter-particle forces [DG95]. Müller et al. proposed point-based animation for a wide spectrum of volumetric objects [MKN+04], the basics of which are summarized in Appendix 3.10. Meshfree methods were also used in offline fracturing [PKA+05] and interactive cut-



(a) Visibility criterion (b) Transparency method

(c) Diffraction method (d) Graph-based diffraction method

Figure 3.10: *Discontinuity modeling in meshfree methods. The object (gray region) is sampled at a set of simulation nodes (blue dots). (a) The visibility criterion assigns a zero value to the shape function since $\overline{x_i x}$ intersects the cut (the red curve). (b) The transparency method enhances the Euclidean distance $\overline{x_i x}$ with the distance from the discontinuity tip to the intersection, $\overline{pa}$. (c) The diffraction method considers the distances from the tip to both nodes, $\overline{px_i}$ and $\overline{px}$. (d) The diffraction distance is approximated by the shortest path in a visibility graph, $\overline{x_i x_a}$, $\overline{x_a x_b}$, and $\overline{x_b x}$.*

ting [SOG06, PGCS09].

In meshfree methods, the object is sampled at a set of simulation nodes x_i. The deformation field is approximated by $u(x) = \sum_i \phi_i(x)u_i$, where u_i are the displacement vectors at the simulation nodes and ϕ_i are shape functions. The shape functions proposed in the computer graphics literature are usually constructed using the moving least squares (MLS) approximation [LS81]. Alternative designs of shape functions for meshfree methods can be found in engineering textbooks (e.g., [FM03]). The shape functions are weighted by a polynomial kernel $w(x, x_i, r_i)$, which rapidly decays with increasing distance between the simulation node x_i and the point x where the function is to be evaluated.

Modeling discontinuity Meshfree methods model the material discontinuities caused by cutting (as well as initial surface concavities) by augmenting the shape functions. A straightforward way is by introducing the visibility criterion [BLG94], i.e., if a point

x is invisible from the simulation node x_i due to the newly created surface, the shape function $\phi_i(x)$ is assigned a value of zero. A drawback of this solution is that it introduces an artificial discontinuity (see Figure 3.10 (a), the two sides of the ray starting from the discontinuity tip p are classified as visible and invisible respectively), which affects negatively numerical convergence and stability. To cope with this, Pietroni et al. extended the visibility criterion by introducing the concept of visibility disk and augmented the shape function by the ratio of the visible region within the disk [PGCS09]. While this approach alleviates the discontinuity caused by the binary-valued visibility criterion, a rigorous definition of the visibility disk has not been proposed so far.

The discontinuity can also be modeled by defining different distance measures for quantification of the distance between x and x_i, which is then considered in the weights of the shape functions. The transparency method [OFTB96] adds to the Euclidean distance between x and x_i a factor that depends on the distance from the discontinuity tip to the intersection of the line segment$\overline{xx_i}$, $\overline{pa}$ in Figure 3.10 (b). This distance measure was used in offline simulations [PKA$^+$05, GLB$^+$06].

The diffraction method [OFTB96], which considers the diffraction of rays around the discontinuity tip, weights the Euclidean distance between x and x_i by their distances to the discontinuity tip, $\overline{px_i}$ and $\overline{px}$ in Figure 3.10 (c). Note that the diffraction and transparency methods were designed for simple 2D domains where the discontinuity tip is well defined. For efficient evaluation of diffraction distances in 3D, Steinemann et al. proposed the use of a visibility graph for estimating the distance along fully visible paths between two points [SOG06]. The distance is chosen as the shortest path in a precomputed visibility graph, see Figure 3.10 (d), $\overline{x_ix_a}$, $\overline{x_ax_b}$, and $\overline{x_bx}$. Upon cutting, the intersected edges of the visibility graph are removed from the graph, and the shortest paths in the graph are updated accordingly.

Boundary surface Meshfree methods do not naturally provide a representation of the boundary. To create new surfaces after cutting, Steinemann et al. proposed to explicitly triangulate the swept surface of a cutting tool [SOG06]. The swept surface is trimmed with respect to the original surface of the object. In the context of fracturing, Pauly et al. explicitly modeled advancing crack fronts by continuously adding surface samples during crack propagation [PKA$^+$05]. Instead of creating new surfaces explicitly, Pietroni et al. maintained a uniform hexahedral grid that is embedded into the simulation domain, and reconstructed from this grid a mesh corresponding to an implicit surface in the volume [PGCS09].

While meshfree methods were intentionally proposed to efficiently handle large deformations and topological changes, more recent works demonstrated performance gains by augmenting meshfree methods with explicitly stored connectivity information. For example, a graph connecting simulation nodes can be employed to evaluate material distance [SOG06] or encode visibility [JL12]. In these two approaches, cutting is modeled by removing corresponding edges in the graph. Another example is the embedding of a uniform hexahedral grid into a meshfree simulation in order to construct the newly exposed cutting surfaces [PGCS09].

3.5 Numerical Solvers

To solve the sparse linear system of equations $Ax = b$ resulting from finite element discretization and implicit time integration, an efficient numerical solver is required. Here it is worth noting that in the particular application of virtual cutting, the initialization time of the solver plays a significant role. Since the system matrices have to be re-assembled in every simulation step due to the use of the corotational strain formulation and the handling of topological changes, the initialization of the solver has to be performed in every simulation step, too.

3.5.1 Direct Solvers

Direct methods determine an exact solution of the linear system of equations by a finite sequence of operations. Targeting static finite element simulations using linear elasticity without corotational strain (i.e., the matrix A changes only if cuts are applied), Zhong et al. [ZWP05] proposed to solve the equation system by precomputing the inverse A^{-1}. In their approach, a cut is modeled by deleting elements, and by adapting the rows and columns of A corresponding to their incident vertices. The manipulation of A is expressed in the form $A + UV^{\mathrm{T}}$, so that the inverse can be updated via the Sherman-Morrison-Woodbury formulae [Hag89]

$$(A + UV^{\mathrm{T}})^{-1} = A^{-1} - A^{-1}U(I + V^{\mathrm{T}}A^{-1}U)^{-1}V^{\mathrm{T}}A^{-1}. \tag{3.13}$$

If the number of affected elements and the number of non-zero entries in the right-hand side vector b are limited by a given constant, the update of A^{-1} and the evaluation of $x = A^{-1}b$ have linear run-time in the number of vertices. Lee et al. [LPO10] made use of condensation [BNC96, WH05] to further accelerate the inversion process. By considering only the surface vertices of the volumetric object in the computation of the inverse, significantly improved computation times can be achieved.

In a similar manner, Lindblad and Turkiyyah updated the inverse of the stiffness matrix in the extended finite element method [LT07]. Using the XFEM, the dimension of A increases due to the newly assigned DOFs (see Eq. 3.6), and A changes to $\begin{bmatrix} A & A^{ua} \\ A^{au} & A^{aa} \end{bmatrix}$, where the superscripts ^{u} and ^{a} correspond to the original and the added DOFs, respectively. The inverse of the new matrix can be computed from the inverse of the original matrix using

$$\begin{bmatrix} A & A^{ua} \\ A^{au} & A^{aa} \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}A^{ua}D^{-1}A^{au}A^{-1} & -A^{-1}A^{ua}D^{-1} \\ -D^{-1}A^{au}A^{-1} & D^{-1} \end{bmatrix}, \tag{3.14}$$

where $D = A^{aa} - A^{au}A^{-1}A^{ua}$.

Turkiyyah et al. proposed to use progressive updates of the Cholesky factorization to simulate cutting of a 2D mesh [TKAN09], and Courtecuisse et al. updated the inverse of the compliance matrix after cutting to model the contact [CJA+10]. Recently, sparse direct solvers were applied to corotational elastodynamics with consistent topology [HLSO12].

3.5.2 Iterative Solvers

Direct solvers using matrix inversion and factorization do not scale well in the number of DOFs because of their extensive memory and computation requirements. Furthermore, such solvers cannot trade accuracy for speed, which is required in interactive applications to guarantee prescribed response rates.

Iterative solvers generate a sequence of increasingly more accurate approximations to the exact solution of a system of equations, and thus are an effective means to balance speed and accuracy by choosing a fixed number of iterations or by specifying a stopping criterion in terms of a threshold for the error reduction. When rating the efficiency of an iterative solver, the major criterion is the achieved convergence rate, i.e., error reduction per computing time.

Nienhuys and van der Stappen [NvdS00, NS01] used a conjugate gradient (CG) solver [She94], which requires a large number of iterations to obtain stable and visually realistic deformations [CJA+10, CAR+09]. On the other hand, since CG solvers involve matrix-vector and vector-vector products they can be parallelized efficiently using OpenMP [CAR+09] and CUDA [CJA+10].

Dick et al. proposed a geometric multigrid solver for cutting simulation, based on a hexahedral discretization of the simulation domain [DGW11a]. The main idea of multigrid is to employ a hierarchy of successively coarser grids, such that successively lower frequency error components can be effectively relaxed on successively coarser grids. Multigrid is optimal in the sense that it exhibits asymptotic linear runtime in the number of unknowns. A detailed comparison of different solvers in the context of virtual cutting has revealed significantly improved convergence rates of multigrid methods compared to a Cholesky solver and a CG solver with Jacobi pre-conditioner. In particular it was demonstrated that the convergence rate of multigrid methods does not depend on the smoothness of the object boundary, which was commonly referred to as a main weakness of multigrid methods.

In multigrid methods, in addition to the relaxation scheme and the coarse grid hierarchy, transfer operators are required to transfer quantities between the grids. Consider a two-level geometric hierarchy where the fine and coarse level are denoted by superscripts h and $2h$, respectively. The linear system on the fine grid has the form $A^h x^h = b^h$. On the coarse grid, the system matrix is approximated by $A^{2h} = R_h^{2h} A^h I_{2h}^h$, where I_{2h}^h is the interpolation operator and $R_h^{2h} = (I_{2h}^h)^{\mathrm{T}}$ the restriction operator. In each iteration, the solver performs the following steps ($\tilde{x}^h$ denotes the current approximate solution):

1. Relax $A^h \tilde{x}^h \approx b^h$,
2. Compute residual $r^h = b^h - A^h \tilde{x}^h$,
3. Restrict residual to the coarse grid: $r^{2h} = R_h^{2h} r^h$,
4. Solve residual equation on the coarse grid: $A^{2h} e^{2h} = r^{2h}$,
5. Interpolate error to the fine grid: $\tilde{e}^h = I_{2h}^h e^{2h}$,
6. Apply correction to the solution: $\tilde{x}^h = \tilde{x}^h + \tilde{e}^h$,
7. Relax $A^h \tilde{x}^h \approx b^h$.

Pre- and post-smoothing (steps 1 and 7) usually involves one or two relaxation iterations. Applying the two-grid method recursively for step 4 leads to a multigrid V-cycle: the relaxation is performed on ever coarser grids $2h$, $4h$, $8h$, ... On the coarsest grid, for step 4 a conjugate gradient solver or a direct solver can be used. In order to adequately represent cuts on the coarser levels, the grid hierarchy can be constructed in a way similar to the hierarchical construction of composite finite elements (see Figure 4.4).

3.6 Summary of Techniques for Cutting Simulation

Since there exist so many different techniques for simulating cuts in deformable bodies, in the following we try to provide a comprehensive overview of these techniques

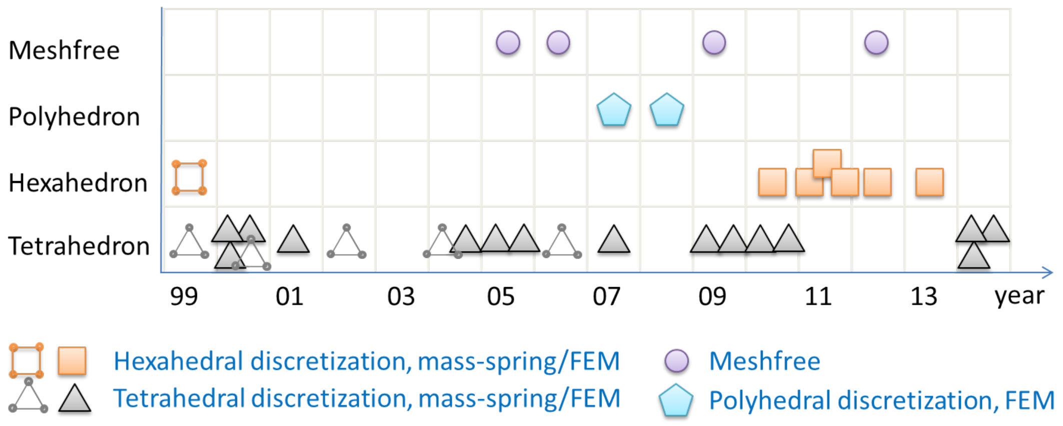


Figure 3.11: *Plot of publications of techniques summarized in Table 3.1 in chronological order.*

according to a few specific categories. The overview is presented in Table 3.1. In particular, we classify techniques according to the discretization and the modeling of cuts (Geometry), the deformable model (Deformation), the time integration and numerical solver (Solver), and the intended application scenario (Scenario). We further give some remarks on specific properties of these techniques.

In Table 3.1 the different techniques are grouped into six categories. In the first category are techniques building on tetrahedral discretizations. It can be observed that these techniques are intended primarily for medical applications. The second and third categories comprise techniques building upon the virtual node algorithm and the XFEM, respectively. In the fourth category are techniques using hexahedral discretizations. Into the fifth and sixth category, respectively, fall papers using polyhedral discretizations and meshfree methods.

Note that considering the specified discretization, in the approaches using composite finite elements, duplication of elements is used on the composite element level, but initially the topological discontinuity is represented by element deletion [JBB$^+$10] or link disconnection after element refinement [WDW11] on the finest level.

Figure 3.11 depicts the chronological appearance of the discussed techniques with respect to the underlying discretization and physical model. From this it can be observed that the majority of publications in the field address mesh-based approaches, and that there seems to be a clear trend towards physically accurate simulations using finite elements. Whereas approaches based on a tetrahedral discretization are constantly used from the beginning, approaches using a hexahedral discretization have emerged more recently.

We show representative simulation results for each group in Figures 3.12 and 3.13.

Reference	Geometry	Deformation	Solver	Scenario	Remark
Bielser et al. [BMG99, BG00, BGTG04]	Tet., refinement	Mass-spring	Explicit/Semi-implicit	Interactive	Basic tet. refinement
Cotin et al. [CDA00]	Tet., deletion	Tensor-mass	Explicit	Interactive	Hybrid elastic model
Mor & Kanade [MK00]	Tet., refinement	FEM	Explicit	Interactive	Progressive cutting
Nienhuys et al. [NvdS00, NS01]	Tet., boundary splitting/snapping	FEM	Static (CG solver)	Interactive	FEM with a CG solver
Bruyns et al. [BSM+02]	Tet., refinement	Mass-spring	Explicit	Interactive	An early survey
Zhong et al. [ZWP05]	Tet., deletion	FEM	Static (direct solver)	Interactive	Static FEM with a direct solver
Wu & Heng [WH05]	Tet., refinement	FEM	Static (CG + direct solver)	Interactive	CG + direct solver
Steinemann et al. [SHGS06]	Tet., refinement + snapping	Mass-spring	Explicit	Interactive (Fig. 3.13 a)	Refinement + snapping
Chentanez et al. [CAR+09]	Tet., refinement	FEM	Implicit (CG solver)	Interactive (Fig. 3.13 d)	Needle insertion
Lee et al. [LPO10]	Tet., deletion	FEM	Static (direct solver)	Interactive	Direct solver + condensation
Courtecuisse et al. [CJA+10, CAK+14]	Tet., deletion/refinement	FEM	Implicit (CG solver)	Interactive (Fig. 3.13 c,e)	Surgery applications
Li et al. [LZW+14]	Tet., refinement	FEM	Explicit	Interactive	Volumetric images
Molino et al. [MBF04]	Tet., duplication	FEM	Mixed explicit/implicit	Offline	Basic virtual node algorithm
Sifakis et al. [SDF07]	Tet., duplication	FEM		Offline (Fig. 3.12 a)	Arbitrary cutting
Wang et al. [WJST14]	Tet., duplication	FEM		Offline	Redevelopment
Jeřábková & Kuhlen [JK09]	Tet.	XFEM	Implicit (CG solver)	Interactive	Introduction of the XFEM
Turkiyyah et al. [TKAN09]	Tri.	2D-XFEM	Static (direct solver)	Interactive	XFEM with a direct solver
Kaufmann et al. [KMB+09]	Tri./Quad.	2D-XFEM	Semi-implicit	Offline (Fig. 3.12 c)	Enrichment textures
Frisken-Gibson [FG99]	Hex., deletion	ChainMail	Local relaxation	Interactive	Linked volume
Jeřábková et al. [JBB+10]	Hex., deletion	CFEM		Interactive	CFEM
Dick et al. [DGW11a]	Hex., refinement	FEM	Implicit (multigrid)	Offline/Interactive (Fig. 3.12 d)	Linked octree, multigrid solver
Seiler et al. [SSSH11]	Hex., refinement	FEM	Implicit	Interactive	Octree, surface embedding
Wu et al. [WDW11, WBWD12, WDW13]	Hex., refinement	CFEM	Implicit (multigrid)	Interactive (Fig. 3.13 b, f)	Residual stress, collision
Wicke et al. [WBG07]	Poly., splitting	PFEM	Implicit	Offline (Fig. 3.12 b)	Basic polyhedral FEM
Martin et al. [MKB+08]	Poly., splitting	PFEM	Semi-implicit	Offline	Harmonic basis functions
Pauly et al. [PKA+05]	Particles, transparency	Meshfree	Explicit	Offline	Fracture animation
Steinemann et al. [SOG06]	Particles, graph-based diffraction	Meshfree		Offline/Interactive (Fig. 3.12 e)	Splitting fronts propagation
Pietroni et al. [PGCS09]	Particles, extended visibility	Meshfree		Interactive	Splitting cubes algorithm
Jung & Lee [JL12]	Particles, connectivity graph	Meshfree	Semi-implicit	Interactive	Dynamic BVHs

Table 3.1: *An overview of cutting techniques outlined in this report.*

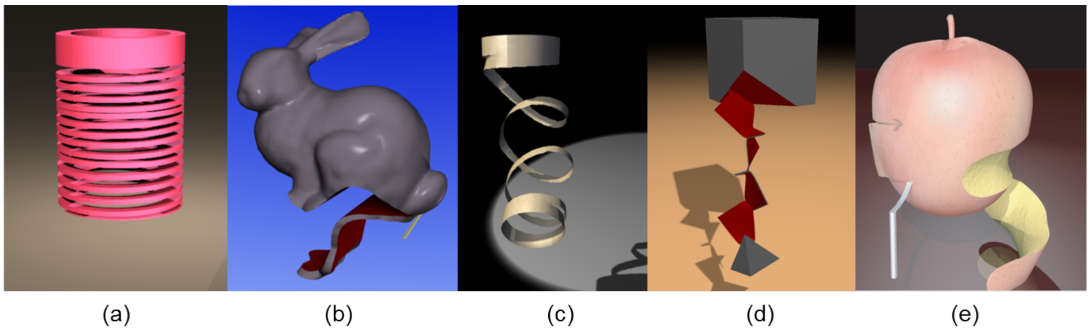


Figure 3.12: *Offline progressive cutting scenarios simulated by (a) the virtual node algorithm on a tetrahedral mesh (image courtesy of Sifakis et al. [SDF07] ©2007 ACM), (b) the polyhedral finite element method (image courtesy of Wicke et al. [WBG07] ©2007 WILEY), (c) the extended finite element method on quads (image courtesy of Kaufmann et al. [KMB+09] ©2009 ACM), (d) the hexahedral finite element method on an adaptive octree grid [DGW11a], and (e) the meshfree method (image courtesy of Steinemann et al. [SOG06] ©2006 Eurographics). Copyrighted materials, image a, b, c, and e, are reprinted with permissions from ACM, WILEY, ACM, and Eurographics, respectively.*

Figure 3.12 shows scenarios simulated offline by (a) the virtual node algorithm, (b) the polyhedral finite element method, (c) the extended finite element method, (d) the hexahedral finite element method on an octree grid, and (e) the meshfree method. Figure 3.13 shows interactive simulation scenarios in various medical contexts, such as (a) ablating a polyp in a hysteroscopy simulator, using element refinement together with snapping of vertices (including a realistic texturing of the cutting surfaces [BZH+05]), (b) virtual soft tissue cutting and shrinkage simulation, by modeling the residual stress in biological tissues, (c) real-time simulation of a brain tumor resection, using an asynchronous pre-conditioner, (d) needle insertion in a prostate brachytherapy simulator with a parallelized CG solver, (e) real-time simulation of laparoscopic hepatectomy dealing with complex contacts, and (f) haptic-enabled real-time virtual cutting of high-resolution soft tissues, using composite finite elements and a multigrid solver.

3.7 Collision Handling and Haptic Rendering

Besides the simulation of deformable bodies, collision handling as well as the haptic rendering of cutting are two important issues in a virtual cutting system. In the following, we briefly cover these topics with the intention to expose challenges of cutting-related research problems. Collision response is analogous to standard deformable body simulation without cutting, i.e., collision forces can be obtained by penalty-based

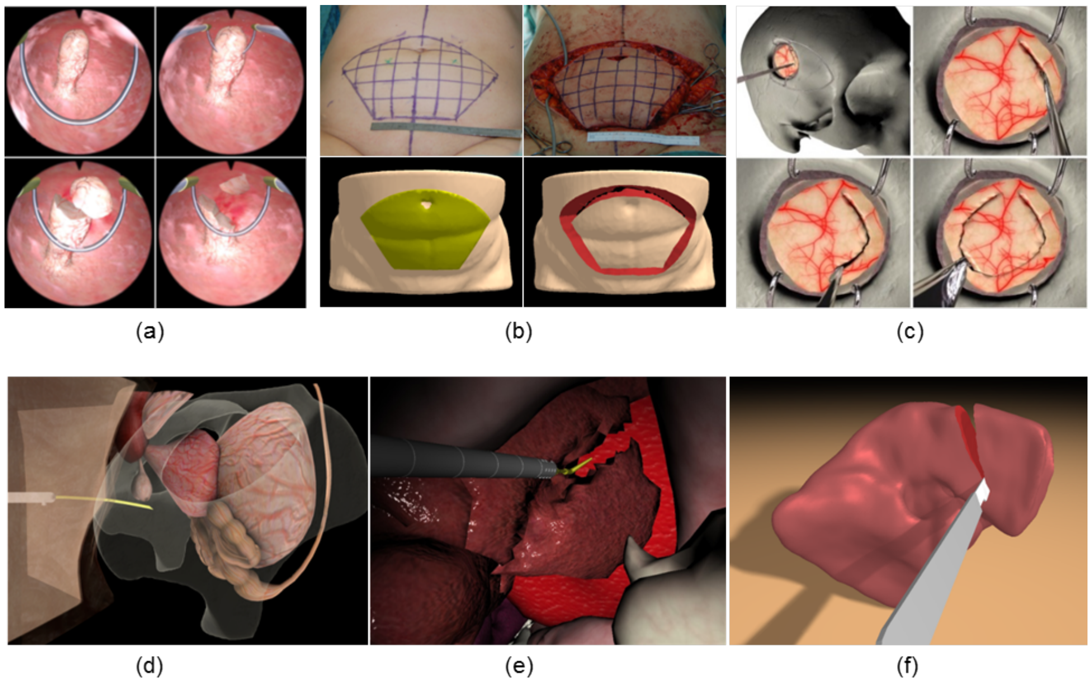


Figure 3.13: *Interactive simulation in medical contexts. (a) Ablating a polyp in a hysteroscopy simulator (image courtesy of Steinemann et al. [SHGS06] ©2006 IEEE). (b) Virtual soft tissue cutting and shrinkage simulation [WBWD12] (abdomen photographs courtesy of Dr. med. Laszlo Kovacs). (c) Real-time simulation of a brain tumor resection (image courtesy of Courtecuisse et al. [CAK+14] ©2014 Elsevier). (d) Needle insertion in a prostate brachytherapy simulator (image courtesy of Chentanez et al. [CAR+09] ©2009 ACM). (e) Real-time simulation of laparoscopic hepatectomy (image courtesy of Courtecuisse et al. [CJA+10] ©2010 Elsevier). (f) Haptic-enabled virtual cutting of high-resolution soft tissues [WWD14]. Copyrighted materials, image a, c, d, and e, are reprinted with permissions from IEEE, Elsevier, ACM, and Elsevier, respectively.*

methods (i.e., by scaling the repulsion force according to penetration depth or penetration volume), or by constraint-based methods (i.e., by enforcing non-penetration by solving a linear complementarity problem [AFC+10]).

3.7.1 Collision Detection

Collision detection for general deformable bodies has been widely studied and an excellent survey is given by Teschner et al. [TKH+05]. These techniques are primarily designed for objects with fixed topology, and most of these techniques need an intensive pre-process to build acceleration data structures. Although, in principle, these methods can also be applied in the context of virtual cutting, the re-initialization of the acceleration structures when topological changes are applied strongly limits their usability. In this section, we discuss the special requirements on collision detection approaches in virtual cutting scenarios, and discuss methods specially designed to meet these requirements.

In virtual cutting simulation, new volumetric elements are created on-the-fly, and new surfaces are exposed. To handle these dynamically created geometric primitives, it is necessary to rebuild or update acceleration data structures such as boundary volume hierarchies. Moreover, as a result of cutting, an object may be incrementally split into several separated objects. It is therefore necessary to consistently detect both inter- and intra-collisions. Consequently, ideal solutions for collision detection for virtual cutting are methods that do not rely on heavy precomputation, detect both self-collisions and collisions between different bodies, and provide a quantitative measure of the penetration for robust collision response.

In several simulators using the SOFA framework [FDD+12] (e.g., [JBB+10,CAK+14]), collision detection is performed by using layered depth images (LDIs) [HZLM01, HTG04, FBAF08]. LDIs do not require preprocessing of surface meshes, and can be efficiently generated in each simulation step by parallel rasterization on the GPU. LDIs sample a closed manifold object by casting a set of parallel rays and enumerating the intersections between each ray and the object. Along each ray, the line segment from an odd intersection (entering the object) to an even intersection (leaving the object) is considered as part of the object. By comparing the LDIs of two objects, the intersection volume and its gradient can be computed and employed in collision response. For detecting self-collisions, the intersections are classified as entering and leaving based on the angle between the forward ray and the surface normal at the intersection point. One open question of LDIs is the representation of thin objects, such as surgical scalpels. Since LDIs do not support non-closed manifold meshes, it would be necessary to use rasterization at extremely high resolution in order to sufficiently represent thin features.

Wu et al. [WDW13] proposed an efficient collision detection algorithm particularly tailored to composite finite element simulation of cuts. In the broad phase, potentially colliding pairs of a deformed volumetric element and a displaced surface vertex are identified by using a spatial hashing approach [THM+03]. All surface vertices in the simulation environment, including vertices of a thin scalpel, are treated in a uniform way. For each potentially colliding element/vertex pair, the surface vertex is back-transformed to its position in the reference configuration using the interpolation weights of the vertex with respect to the volumetric element. The penetration at this position is evaluated from a distance field in the reference configuration (which is locally updated during cutting), and forward-transformed to the deformed configuration for collision response. It was shown that by checking the coarse composite elements, rather than the underlying fine hexahedral elements, a significant performance gain can be obtained. The results also demonstrate that smooth collision response can be achieved for deformable bodies using a hexahedral discretization, despite of the presence of staircase boundaries.

Bounding volume hierarchies (BVHs) are an efficient data structure to accelerate collision detection. However, the tightness of bounding volumes and the culling efficiency degrade significantly in case of large deformations and topological changes. Especially in the context of fracture simulation, several methods have been proposed to optimize the reconstruction/updating of BVHs. To efficiently insert/delete geometric primitives, Otaduy et al. presented a method to dynamically reconstruct BVHs, as opposed to reconstructing them from scratch [OCSG07]. The hierarchies are balanced by simple local operations for progressive fractures. BVHs for large scale fracture simulation was studied in [HSK+10]. Recently, Glondu et al. [GSM+12] demonstrated real-time brittle fracture simulations based on a combination of locally updated distance fields and sphere trees for adaptive collision detection, together with an approximation of modal analysis for fracture simulation [GMD13].

Jung et al. proposed a method to reconstruct BVHs for meshfree simulation of cuts, where an undirected graph is maintained to encode the connectivity of simulation nodes [JL12]. The BVH reconstruction is triggered by the event that an object is completely excised into two pieces. This event is detected by examining the node connectivity: In a breadth-first traversal starting from an arbitrary node, if at least one node of the object is not visited during the traversal, the excision event is reported.

3.7.2 Haptic Rendering of Cutting

Haptics provides an intuitive interface to interact with the simulated deformable bodies and conveys rich information about the dynamics directly to the user. This is especially useful in medical simulations [CMJ11]. While haptics has been mentioned in many cutting simulators, few provide a sufficient description on the implementation and evaluation. Realistic haptic rendering of cutting is a challenging task due to the required high update rates of haptic rendering and the complex physical interaction between the cutting tool and soft tissues.

First, haptic rendering requires update rates of 1 kHz or higher, in order to achieve (perceived) smooth force feedback [SCB04]. It was reported that users can perceive differences at update rates between 500 Hz and 1 kHz [BOYBJK90]. More fundamentally, the update rate is closely related to the stability of the haptic system, i.e., a human-in-the-loop system consisting of the user, the (digitally controlled) haptic device, and the (time-discrete) virtual environment. The latency inherent in time-discrete systems can lead to unstable behaviors (e.g., vibrations). We refer to the haptics literature [CGSS93, CB94] for a detailed explanation. The sufficient condition of passivity and thus stability of the haptic device for a viscoelastic virtual wall (the simplest virtual environment) is given in [CS97] as

$$\Delta T < \frac{2(b - B)}{K}, \tag{3.15}$$

where ΔT is the sampling period, b is the inherent damping of the device, and K and B are the stiffness and damping of the virtual wall, respectively. This condition implies that a high update rate is very necessary to simulate the interaction with stiff virtual objects.

Equation 3.15 states the stability for haptic interaction where the cutting tool is simply represented by a single point. For a general cutting tool which may simultaneously contact the deformable body at several locations, it is necessary to modulate the overall contact stiffness, considering the limited impedance offered by haptic devices. To this end, rather than directly mapping the position of the virtual tool from the position signal read from the haptic device, it is customary to separate these two positions, and virtually couple them by a spring (and a damper). This virtual coupling approach results in the so-called simulation-based haptic rendering: The movement of the virtual tool is driven by the coupling force, which tries to align the virtual tool with the haptic stylus, and the interaction force (e.g., the cutting force) between the virtual tool and deformable bodies. The coupling force, rather than the interaction force is sent to the haptic device. In this way the stability of the device can be easily ensured by tuning the coupler [MPT99]. It also enables that the haptic simulation and the deformation/cutting simulation run at different update rates. We refer to a recent survey on haptic rendering [OGL13] for a detailed explanation of different rendering paradigms. The virtual coupling scheme is successfully employed in bone drilling [WWWZ10] and in soft tissue cutting [WWD14] to compute feedback forces which can be rendered stably.

Second, the physical cutting mechanism (i.e., the fracturing of soft tissues driven by cutting tools) is largely unknown, especially considering the complex material properties and various cutting tools such as needles, blades, scissors, or punchers. Compared to the physical world, where cuts are induced by the internal stresses resulting from the force interaction between the deformable object and the scalpel, the cutting approaches so far are purely geometry-based: The tissues are cut by geometric intersection tests, as discussed in Section 3.2.1. It can be interpreted as modeling an infinitely sharp scalpel, which can induce arbitrary stresses and thus immediately penetrates the object. In contrast, in the physical world, the object would deform under the influence of the increasing force exerted by the scalpel, before the scalpel eventually penetrates. Modeling a more realistic interaction between the object and the scalpel is part of ongoing research, for example in biomedical engineering [CDL07]. Using penalty-force-based collision handling, an initial attempt to simulate this effect is to employ a virtually extended scalpel shape [JBB$^+$10]. The enlarged scalpel penetrates into the deformable object before the real scalpel penetrates, thus leading to a deformation before the object is cut. The enlargement of virtual tools is similarly used in bone drilling simulation [WWWZ10].

An approach to obtain an intuitive feedback force is to employ a velocity-proportional force model [WWD14]. Intuitively, if the user moves the scalpel with a high speed against the deformable object, (s)he feels a large resistant force. The force direction is opposite to the direction of movement, and the force magnitude is proportional to both the speed of movement and the contact volume between the scalpel and the deformable object. Another possible solution is the data-driven approach [HKSH09].

3.8 Application Study on Cutting Simulation

In the following application study we intend to shed light on the performance of physically-based cutting simulation and, by this, to assess the model resolution that can be handled in interactive scenarios requiring update rates of 20-30 Hz. We restrict ourselves to the analysis of one specific simulation approach for which a highly optimized implementation is available. Even though this approach has limitations, we believe that it allows for a very good estimation of the simulation efficiency that can be achieved when the model of linear elasticity is used.

Figure 3.14 shows our experiment setup in which we simulate a cut in a linear elastic liver model. All experiments were performed on a standard desktop PC equipped with an Intel Xeon X5560 processor (a single core was used) and 8 GB main memory.

We analyze three variants of the hexahedral finite element approach proposed by Dick et al [DGW11a]. It uses the corotational formulation of finite elements, which simulates linear dependencies between the components of stress and strain, and considers the geometric non-linearity by respecting per-element rotations in the strain computation. While finite element discretizations enable high accuracy, hexahedral elements

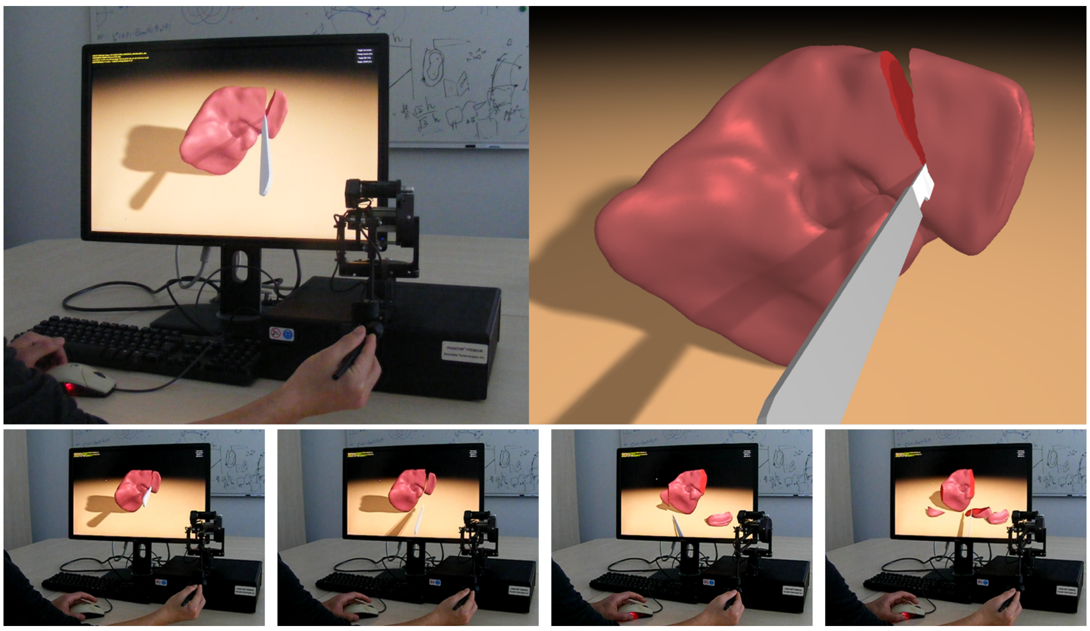


Figure 3.14: *Left: Experiment setup. To cut a liver model the user manipulates a haptic device that is mapped to a scalpel. Right: High quality surface rendering. Bottom: A sequence of images from a live recording, available at http://wwwcg.in.tum.de/research/research/projects/real-time-haptic-cutting.html.*

are well suited for constructing a mesh hierarchy that can be used by a geometric multigrid solver to achieve optimal convergence rates. On the other hand, since hexahedral simulation elements are not aligned with the object boundaries, approaches using unstructured tetrahedral simulation grids might be favorable when smooth boundary-aware discretizations of the simulation domain are required. For instance, to perform accurate collision detection and response. In all of our experiments, a high-resolution surface is generated from the cut object using the dual contouring algorithm on the hexahedral grid, and this surface is used for rendering and collision detection [WDW13].

Our first variant uses finite elements on a uniform hexahedral grid and realizes cuts by simply disconnecting elements along element faces. A high-resolution finite element model consisting of 173,843 hexahedral elements (566,493 DOFs) on a $82 \times 83 \times 100$ uniform grid is used as our reference solution (see Figure 3.15 (left)). We simulate the cut by instantly bringing the cutting tool into its final position and performing all required operations like finding and disconnecting edges in the simulation grid, FE matrix assembly, and numerical multigrid solver execution. In particular, we perform 2 V-cycles each including 2 pre- and 2 post-smoothing Gauss-Seidel relaxation steps, which reduces the error to below 1%. Note that while cutting on a uniform grid does

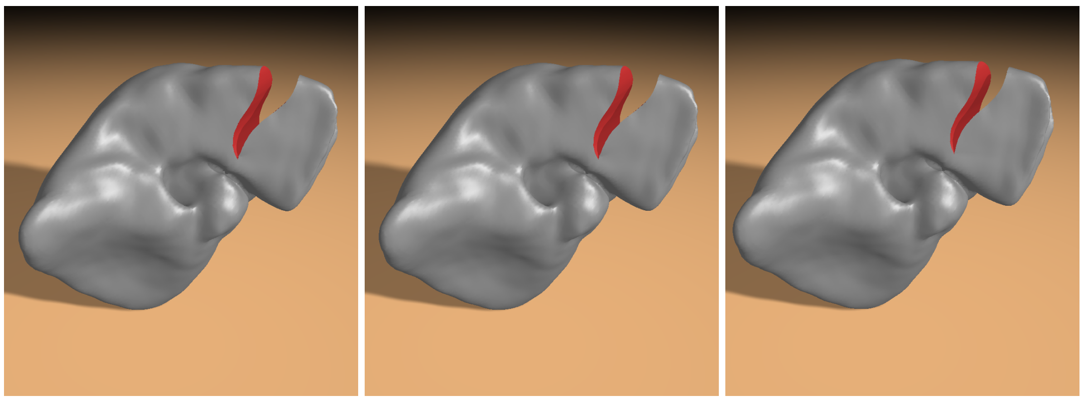


Figure 3.15: *From left to right: Simulation of cuts using finite elements on a uniform hexahedral grid, finite elements on an adaptive octree grid, and composite finite elements on an adaptive octree grid.*

not add new elements, the number of DOFs is increased, since some of the originally shared element vertices become separated vertices due to the cut.

The second variant uses finite elements on an adaptive octree grid. This grid is constructed by starting with a uniform coarse hexahedral grid, which is adaptively refined along the initial object boundary and the cut, until a user-selected level is reached. For this variant we have set the resolution of the initial coarse grid and the refinement depth such that in the refined regions the grid resolution of the first variant is reached. The third variant uses the same adaptive grid as the second one, but instead of standard finite elements, it uses composite finite elements at the resolution of the initial coarse grid [WDW11]. Since the refined elements are used only to correct the coarse grid simulation, considerably higher performance is achieved. The last variant is intended to demonstrate the trade-offs between highest accuracy and speed in interactive scenarios when employing the principle of homogenization for linear elasticity [KMOD09]. Since the adaptive variants restrict the element refinement to a user-selected depth, alternative (offline) approaches like extended finite elements [JK09, KMB+09] can be favorable in applications where cuts should be modeled at sub-grid accuracy.

Figure 3.15 (middle) shows the same cut as before, but now the second variant is used to simulate the cut. We start with a $21 \times 21 \times 25$ uniform grid. Initially, this grid is refined adaptively along the object boundary via two levels of subdivision. When the cut is simulated, the grid is further refined along the cut using the same refinement depth, resulting in 41,676 hexahedral elements (135,600 DOFs).

In the last experiment (see Figure 3.15 (right)) we start with the same initial grid as in the second experiment, and we apply exactly the same adaptive grid refinement

	Uniform	Adaptive	Composite (2 levels)
Coarse resolution		21×21×25	21×21×25
Refined resolution	82×83×100	82×83×100	82×83×100
# Cells (initial)	173 843	40 080	3 439
# DOFs (initial)	566 493	129 162	13 557
# Cells (added due to cut)	0	1 596	39
# DOFs (added due to cut)	2 037	6 438	318
Octree subdivision (t_1)	0	13.29	13.39
Surface meshing (t_2)	1.26	1.26	1.24
FE matrices (t_3)	29.57	7.05	20.99
Multigrid hierarchy (t_4)	40.34	10.09	2.06
Solver (t_5)	2 033.09	581.66	40.61
Simulation per cut ($\sum_{i=1}^{5} t_i$)	2 104.26	613.35	78.29

Table 3.2: *Timings (in milliseconds) for cutting simulations using finite elements on a uniform hexahedral grid, finite elements on an adaptive octree grid, and composite finite elements on an adaptive octree grid.*

along the object boundary and the cut. However, the adaptively generated elements are not considered as DOFs in the simulation, but they are used to assemble the coarse grid matrices according to their contributions. Thus, the simulation is performed using 3,439 composite elements (13,557 DOFs).

Table 3.2 lists the times that are consumed by the different processes in each experiment. It can be seen that even though a large number of DOFs can be simulated in roughly 2 seconds using a uniform simulation grid, the grid resolution has to be reduced about a factor of 4 in each dimension to make the uniform grid suitable for interactive scenarios. Via the adaptive octree grid the cut can be simulated at almost no visual difference to the high-resolution reference solution. Due to the restriction of element refinements along the initial object boundary and the cut, the overall simulation time can be reduced by a factor of 3.5. Using composite finite elements, the number of simulation elements to be considered by the numerical solver can be decreased further, making this approach suitable for interactive scenarios. Despite the low number of DOFs to be solved for, the simulation result is in very good agreement with the results generated by the other variants. It is clear, however, that due to the reduced number of DOFs, the simulated deformations cannot exactly match the high-resolution reference in general.

Table 3.2 further indicates that surface meshing does not have any impact on the overall performance. This is because it works only on the boundary elements. Since the effective resolution of the boundary elements is the same in all three experiments, surface meshing always consumes the same amount of time. It can be seen that in addition to the time consumed by the multigrid solver, especially in the third, interactive variant the adaptive grid refinement (t_1) and the assembly of the finite element matrices (t_3) take up a considerable amount of the overall time. In this variant, the time required to generate the multigrid hierarchy (t_4) is rather low due to the low resolution of the simulation grid. This variant requires a grid hierarchy from the finest level to the coarse simulation level as well for assembling the FE matrices on the simulation grid. The time for updating this part of the hierarchy is counted in t_3 for this variant.

3.9 Discussion and Conclusion

In this report we have reviewed the current state-of-the-art in computer-aided simulation of cuts in deformable bodies. We have discussed distinct geometry and topology representations, specifically-tailored finite element approaches, and meshfree methods, with respect to accuracy, robustness, and computational efficiency.

The analysis of current techniques indicates a clear trend towards physically-based simulations. From our experience this trend is driven by the application domains in which virtual cutting is applied. Especially in virtual surgery simulators, which are used for training and preoperative planning, doctors are more and more demanding for reliable simulations that can accurately predict the behavior of soft tissue undergoing cuts and deformations thereof. Thus, going beyond this STAR we see the urgent need for a benchmark that is tailored to the problem of virtual cutting simulation and that can be used to assess simulation techniques quantitatively.

Furthermore, especially in medical applications the accurate modeling of real-world and patient-specific material is becoming of ever increasing importance. Going beyond the model of linear elasticity and homogeneous material, soft tissues exhibiting non-linear, anisotropic, viscoelastic and even viscoplastic behavior [Hum03] need to be considered by interactive simulators in the future. However, even though it is known in principle how to model such tissue types physically, we see the efficient numerical simulation of these types as one of the most important research questions for the future.

One possibility to achieve interactive cutting simulation even for complex tissue types is the use of dedicated parallelization strategies on multi-core and multi-GPU architectures. On single GPUs, the parallelization of deformable body simulation has already shown significant performance improvements [DGW11b, CJA+10]. The layout of numerical solvers across multiple CPU or GPU nodes, however, is extremely challenging. In particular for the parallelization of an initially sequential solver, typically frequent communication between the nodes is required, letting bandwidth and latency become quickly the bottleneck. It is therefore required to develop parallel solvers that are particularly tailored to such computing architectures by reducing the communication between the nodes. A promising approach that needs further investigation are domain decomposition methods, which divide the problem into subproblems that can be solved independently.

Interactive simulation frame rates can also be achieved by model reduction techniques, as having been demonstrated in a number of engineering [NACC08] and also graphics applications [BJ05]. The idea is to carefully approximate a large system of equations with a much smaller system (i.e., to reduce the number of DOFs), without significantly sacrificing accuracy. A major difficulty of applying these techniques in the context of interactive cutting scenarios is the fact that determining the reduced system is typically very compute-intensive, so that in practice at least the cutting zone is non-reducible and must be tackled fully, without reduction. A possible direction thus is to couple model reduction for reducible zones and full simulation for non-reducible zones [KGRB13, NAG+12].

Another direction of research is the development of approaches for modeling the physical interaction between a scalpel and soft tissues accurately [MH01,CDL07,MRO08]. This can greatly contribute to the realistic haptic rendering of cutting forces. For simplicity, most virtual cutting techniques assume that the material is separated as long as it is swept by a cutting tool. In the physical world, however, we can observe that there is a deformation of soft tissues before a cut happens. The simulation of this kind of tool-object interaction may benefit from general contact resolution techniques [HVS+09, AFC+10, SH12].

Finally, the discussion of techniques in this report has revealed that two major, and somehow opposing, requirements reflect in the design of cutting techniques. On the one hand, one seeks to use unstructured spatial object discretizations to accurately model a cut. This has led to geometric techniques using tetrahedral or polyhedral meshes, which are remeshed irregularly along the cut. On the downside, the remeshing step becomes very elaborate for arbitrary cutting paths, and it increases the number of simulation elements significantly.

On the other hand, to achieve high performance of the numerical solver used to simulate the dynamic behavior of the cut body, structured simulation grids have turned out to be favorable. Scalable solvers exhibiting linear complexity in the number of supporting vertices have been achieved via geometric multigrid methods on semi-regular hexahedral grids. While building geometric multigrid hierarchies on hexahedral grids is simple, on unstructured grids the construction of such hierarchies is extremely complicated and very time-consuming. In general, however, elements in hexahedral grids are not aligned with the object boundaries, introducing modeling inaccuracies along these boundaries.

In our opinion it is one of the most interesting questions whether adaptive spatial discretizations can be found that give rise to efficient numerical solution techniques at the same time allowing for an accurate alignment of simulation elements along the object boundaries.

The research on interactive virtual cutting is not limited to the computer graphics community. In computational mechanics, where research on cutting and fracturing was initially more concerned with accurate material models (e.g., accurate boundary conditions [MBB+11], accurate cutting force models [MMSE11], goal-oriented error estimates [GENR+14]), there is a recent trend towards interactive surgery simulations [NAG+12, JJMW13]. It will be interesting to investigate how such models can be integrated into efficient simulators. We envisage that cross-fertilization with computational mechanics will further advance virtual cutting simulation towards its application in pre-operative planning and surgery training.

3.10 Appendix: Meshfree Methods for Deformable Body Simulation

In contrast to FEMs which operate on a finite element mesh with an explicit connectivity among the nodes (see Section 2.2), meshfree methods maintain node adjacency by defining an influence region for each node, described by a weight kernel. The weight kernel for the node i located at the position x_i can be defined as, for example [MKN+04],

$$\omega(x, x_i, r_i) = \begin{cases} \frac{315}{64\pi r_i^9}(r_i^2 - \|x_i - x\|^2)^3 & \text{if } \|x_i - x\| < r_i, \\ 0 & \text{otherwise,} \end{cases} \tag{3.16}$$

where r_i is the influence radius. The value of the weight kernel rapidly decays with increasing distance between the simulation node x_i and the point x where the function is to be evaluated. The radius of influence r_i should be sufficiently small to adequately discretize displacement gradients. It is typically chosen in such a way that the influence region includes a constant number of neighbors of the node x_i.

Meshfree methods model objects as a set of interacting nodes which carry properties, e.g., mass, volume, and density. The mass m_i carried by a node is initialized as $m_i = s\, r_i^3 \rho$, where ρ is the material density, and s is a scaling constant for all nodes, chosen such that the estimated density $\rho_i = \Sigma_j m_j \omega(x_j, x_i, r_i)$ is close to ρ. The volume covered by a node is calculated by $v_i = m_i/\rho_i$.

In meshfree methods, the displacement field is approximated as $u(x) = \Sigma_i \phi_i(x) u_i$ from the linearized displacement vectors u_i at a set of (nearby) nodes $\{x_i\}$ and the shape functions $\phi_i(x)$ [FM03]. The meshfree shape functions can be approximated by using, for example, the moving least squares (MLS) scheme [LS81]. The MLS is a method to reconstruct continuous functions from a set of sample points based on their distribution, i.e., the connectivity of samples are not required. Given the weight kernel ω, the MLS yields the following shape functions [PKA+05]

$$\phi_i(x) = \omega(x, x_i, r_i) p^{\mathrm{T}}(x)[H(x)]^{-1} p(x_i), \tag{3.17}$$

where p denotes a complete polynomial basis $p(x) = [1 \ x \ ... \ x^n]^{\mathrm{T}}$, and $[H(x)]^{-1}$ is the inverse of the moment matrix defined as

$$H(x) = \sum_i \omega(x, x_i, r_i) p(x_i) p^{\mathrm{T}}(x_i). \tag{3.18}$$

Since the inversion of a matrix is involved in the shape functions, a direct evaluation of their derivative is non-trivial. An alternative is to approximate the gradient ∇u at nodes using a MLS formulation with a linear basis [MKN+04]. Representing the neighbors of the node x_i as $\{x_j\}$, the displacements of neighbor nodes can be approximated by

$$\tilde{u}_j = u_i + \nabla u|_{x_i} x_{ij}, \tag{3.19}$$

where $x_{ij} = x_j - x_i$, and $\nabla u|_{x_i}$ is the displacement gradient at the node x_i. The sum of the squared differences between the approximated values $\tilde{u}_j$ and the known values u_j is

$$e = \sum_j (\tilde{u}_j - u_j)^2 \omega(x_j, x_i, r_i). \tag{3.20}$$

Minimizing this error function by assigning a zero value to its derivative with respect to u_i, it can be derived that

$$\nabla u|_{x_i} = A^{-1} \left(\sum_j (u_j - u_i) x_{ij} \omega(x_j, x_i, r_i) \right), \tag{3.21}$$

where the matrix $A = \Sigma_j x_{ij} x_{ij}^{\mathrm{T}} \omega(x_j, x_i, r_i)$ is a 3×3 matrix.

With the gradient of the displacement field computed, the strain tensor can be evaluated at the node x_i according to the Green-St. Venant (Eq. 2.1) or the infinitesimal (Eq. 2.4) strain formulation. The stress tensor for linear elastic materials can be computed using Eq. 2.5.

Internal forces can be derived as the negative gradient of the strain energy density with respect to the displacement field,

$$f^{int} = -\nabla_u(\frac{1}{2}\epsilon : \sigma). \tag{3.22}$$

By integrating this function over the volume v_i covered by the node x_i, it can be derived that this yields the following form of internal forces [MKN+04],

$$f_i^{int} = 2\, v_i \left(I + (\nabla u|_{x_i})^{\mathrm{T}}\right) \sigma_i \; A^{-1} \left(\sum_j x_{ij}\omega(x_j, x_i, r_i) \right). \tag{3.23}$$

The dynamic simulation problem can then be formulated as

$$f_i^{int} + f_i^{ext} - m_i \ddot{u} = 0, \tag{3.24}$$

where f_i^{ext} denotes the external forces applied to the node x_i.

To animate the boundary surface which is for example represented by a surface mesh, the displacement u of a surface vertex can be computed based on the displacements u_i of its nearby simulation nodes as

$$u(x) = \frac{1}{\sum_i \omega(x, x_i, r_i)} \sum_i \omega(x, x_i, r_i) \left(u_i + \nabla u|_{x_i}(x - x_i)\right). \tag{3.25}$$

Chapter 4

A Composite Finite Element Framework for Virtual Cutting

In this chapter we present a composite finite element framework for virtual cutting simulation. A design goal is to allow for interactive simulation update rates, and meanwhile to support a high-resolution render surface. Key to our approach is a composite finite element formulation, building on a hexahedral discretization of the simulation domain. Starting at a coarse resolution simulation grid, along a cut we perform an adaptive octree refinement of this grid down to a desired resolution and iteratively pull the fine level finite element equations to the coarse level. In this way, the fine level dynamics can be approximated with a small number of degrees of freedom at the coarse level. By embedding the hierarchical adaptive composite finite element scheme into a geometric multigrid solver, and by exploiting the fact that during cutting only a small number of cells are modified in each time step, high update rates can be achieved for high-resolution surfaces at very good approximation quality. To construct a high quality surface that is accurately aligned with a cut, we employ the dual contouring approach on the fine resolution level, and we instantly bind the constructed triangle mesh to the coarse grid via geometric constrains.

4.1 Introduction

In numerical simulations using composite finite elements the basis functions of the finite elements on a coarse grid are assembled via linear combinations of basis functions

This chapter is partially based on material that has been originally published in J. Wu, C. Dick, and R. Westermann, *Interactive high-resolution boundary surfaces for deformable bodies with changing topology*, Proceedings of Workshop on Virtual Reality Interaction and Physical Simulation, 2011, pp. 29–38.

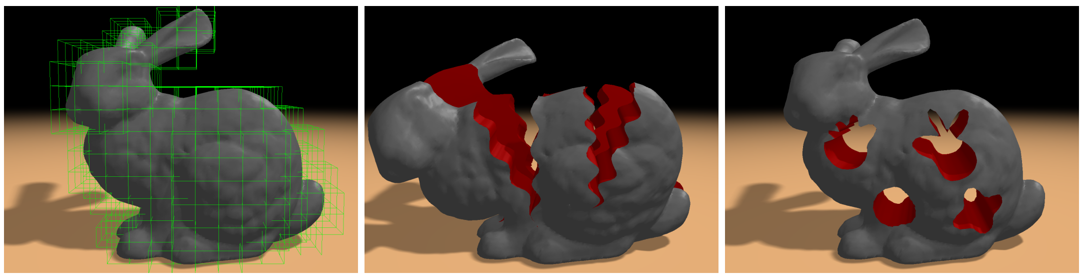


Figure 4.1: *Left: The coarse hexahedral simulation grid and the high-resolution surface that is constructed from the embedded fine grid. Middle and Right: The deformation of the body after complex sin wave and stamp-like cutting have been performed. Cutting, surface construction, and deformation simulation are performed at 12 simulation frames per second.*

on a finer grid. Composite finite elements allow approximating the dynamics of a set of fine level elements by one single coarse element that reflects the portion of material that is covered by the fine elements. Such elements have originally been invented to resolve complicated material micro-structures with only few degrees of freedom (DOFs) and to enable multigrid methods to effectively represent complicated object boundaries at ever coarser scales [HS97, SW06, PRS07, LPR+09]. To respect physically disconnected parts of the simulation domain on the coarse grids, the connectivity between the embedded fine elements can be considered to generate one coarse-grid element for every disconnected part [ABA00, MBF04].

In computer graphics, composite finite elements have been used as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation, i.e., by Nesme et al. [NKJF09], and just recently to model material discontinuities that are caused by cuts and incisions, for instance, by Jeřábková et al. [JBB+10] and Dick et al. [DGW11a]. While Dick et al. took advantage of composite finite elements to improve the convergence of a multigrid solver in situations where material discontinuities are covered by the coarse-grid cells, Jeřábková and coworkers employed the underlying principle to allow simulating complicated topologies at a much coarser resolution than the finest structures.

In this chapter we propose combining the strengths of previous composite finite element methods for cutting deformable bodies. In particular, we present a combination of the adaptive octree refinement with iterative composite element hierarchy to enable simulating high-resolution cuts with a small number of DOFs. To enable sharp creases when cutting into the material, and to generate a high-quality boundary surface along a cut, we propose using the dual contouring approach [JLSW02]. Figure 4.1 demonstrates the quality and speed that can be achieved by using a low-resolution composite finite element simulation grid (left) in combination with an embedded adaptive grid at which a high-resolution surface is constructed.

The specific contributions of this chapter are:

- A combination of adaptive top-down refinement and iterative fine-to-coarse composite finite elements to enable high-resolution cuts and efficient computation of the coarse-grid equations.
- Acceleration techniques to further improve the computational efficiency of equation composition, i.e., usage of an iterative coarsening approach to exploit the fact that many interpolations weights are zero and thus to reduce the number of arithmetic operations, and employment of a look-up-table for the first coarsening step.
- A new approach for constructing a high-resolution, high-quality surface that accurately aligns with a cut and reduces the number of surface elements.
- An incremental update scheme for both the coarse-grid finite elements and the fine-grid render surface to achieve high simulation rates.

The remainder of this chapter is organized as follows: First, we will acknowledge work that is related to ours. Next, we will briefly describe the hierarchical octree refinement of a hexahedral grid along a cut, and the reconstruction of a high-resolution render surface from the fine grid. In Section 4.4, the composite finite element approach in combination with an efficient assembly of the coarse-grid equations from the fine-grid equations is outlined. In Section 4.5, we introduce the multigrid solver for the linear system of equations of composite finite element. Finally, detailed timing and memory statistics are provided, and the chapter is concluded with some ideas for further work.

4.2 Related Work

This section reviews virtual cutting techniques on a hexahedral finite element discretization. For a comprehensive review of virtual cutting using other discretizations let us refer to Chapter 3.

The semi-regular hexahedral finite element discretization has recently been employed to simulate topological changes in deformable bodies [JBB$^+$10,DGW11a,SSSH11]. The hexahedral discretization has several advantages due to its simplicity and regularity. First, each hexahedral element has the same and regular shape: The edges of an element have the same length, and the angles formed by adjacent edges are $\frac{\pi}{2}$. The numerical problem associated with ill-shaped elements in the commonly used tetrahedral discretization is successfully avoided. This allows us to focus on the computational efficiency without worries about numerical instability. Second, the geometric operations on a hexahedral discretization is simple and efficient. For example, the subdivision operation divides a hexahedral element into 2^3 finer elements by simply dividing at the middle point of each edge. Along an opposite direction, 2^3 hexahedral elements can be grouped into one coarser element. This property supports efficient composition of elements to reduce the number of DOFs to be solved for [JBB$^+$10], and to improve the convergence of a multigrid finite element solver [DGW11a].

Based on a hierarchical hexahedral finite element discretization, Jeřábková et al. modeled a cut via element removal on the finest resolution level. The boundary surface along the cut is reconstructed via the Marching Cubes algorithm. This approach, however, limits the resolution at which a cut can be performed and, in general, makes it difficult to construct a surface that accurately aligns with a cut. Dick et al., on the other hand, performed an adaptive octree subdivision along a cut, restricting the high-resolution grid to those regions where it is required. To accurately align the surface with a cut and to allow sharp creases when cutting into stiff material, the splitting cubes algorithm [PGCS09] is employed. Seiler et al. proposed a method that allows decoupling the resolution of a material surface from the resolution of the simulation grid, but since the method is restricted to non-progressive cuts using a volumetric blade, the construction of a boundary surface is extremely simplified and stamping rather than cutting is simulated.

4.3 Geometry and Topology Representation

Our method for cutting deformable objects is particularly designed to allow for interactive simulation update rates and, at the same time, to allow for a detailed modeling of fine cuts, including the adaptation of a high-resolution render surface for a high-quality rendering. To achieve this, we use a high-resolution model to represent the geometry and topology of the deformable body, which is coupled with a lower-resolution finite element model for the simulation of this body. The geometry and topology representation consists of a volume representation and a surface representation, and will be explained in this section. The coupling of the two models will be explained in the Section 4.4.

4.3.1 Volume Representation

To model cuts in the deformable body, we use a linked volume representation as it was initially proposed by Frisken-Gibson [FG99]. The basic idea of the linked volume representation is to decompose the object into a set of hexahedral cells, using a uniform hexahedral grid. Face-adjacent cells are connected via links, with six links emanating from each cell. Cuts are modeled by marking links as disconnected when they are intersected by the virtual cutting blade (see Figure 4.2). Cuts are thus represented at the resolution of the hexahedral grid.

Since the resolution of a uniform grid is in practice limited by memory requirements, we use an adaptive octree grid as proposed by Dick et al. [DGW11a], which adaptively refines along cuts, down to a certain finest level. Links are still considered on the uniform grid corresponding to this finest level, but are physically stored only for the cells at the finest level. The adaptive octree grid is constructed by starting from a coarse uniform grid. Whenever a link on the finest level is intersected by the surface of the deformable object, the incident cells (possibly only one octree cell, when both endpoints of the link are lying within the same cell) are refined using a regular 1:8 split. At the finest level, links are marked as disconnected when they are intersected by the object's surface. Cells that are lying outside of the object are removed from the representation. To avoid jumps in the discretization, additional splits are performed to ensure that the level difference between cells sharing a vertex, an edge, or a face is at most one (restricted octree). Cuts are modeled analogously to the modeling of the object surface, i.e., cells are adaptively refined along a cut down to the finest level, where links are marked as disconnected (see Figure 4.4 for an example). Material properties such as Young's modulus and density are assigned on a per-cell basis. To model inhomogeneous materials, the octree can further be refined. For a more detailed description, we would like to refer the reader to [DGW11a].

4.3.2 Surface Representation

To render the boundary surface of the deformable object—including the additional surface parts that are generated by a cut—a surface mesh is reconstructed from the volume representation. Since the object boundary is represented by the cells at the finest level, the surface reconstruction is performed at this level. The meshing procedure has to consider that sharp surface features are generated by a cut and should be reproduced (see Figure 4.1 for an example). Furthermore, the procedure should be able to efficiently establish a correspondence between the surface vertices and the vertices of the simulation

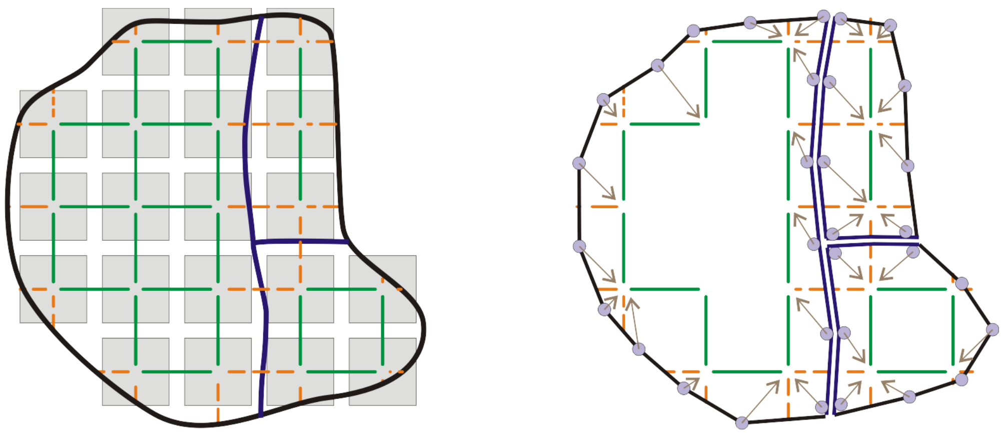


Figure 4.2: *2D illustration of surface reconstruction and binding of the resulting surface vertices. Left: Linked volume representation of an object, consisting a set of simulation nodes (gray) which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary (black) and the newly generated surface due to cutting (blue). Right: Surface reconstruction from the grid that is formed by links. The binding of resulting surface vertices (blue dot) to the simulation nodes is indicated by the brown arrows.*

grid (see Section 4.4). Given this correspondence, the surface vertices can be bound to the simulation vertices and move according to the object deformations, eliminating the need to reconstruct the surface in every frame from the deformed volume configuration.

Surface Meshing

In our implementation the high-resolution surface is reconstructed via the dual contouring approach [JLSW02]. The dual grid we use is the adaptive grid that is formed by the links between the cells at the finest level. For each link that is cut by the blade, the distances between the intersection point to the link's endpoints as well as the normal of the blade at the intersection point are stored. Compared to the splitting cubes algorithm [PGCS09], which was used in [DGW11a], the dual contouring approach reduces the number of triangles by a factor of four. The reason is that the vertices on the links are not used in the triangulation of the boundary surface.

In dual contouring, for each cell that contains a 'feature' a representative vertex is positioned at this feature. In our application a feature is indicated by at least one disconnected link in the dual cell. The position of the representative vertex is where the quadratic function

$$E = \sum_i (n_i \cdot (x - p_i))^2 \tag{4.1}$$

has a minimum. Here, x denotes the vertex position, and p_i and n_i are the positions

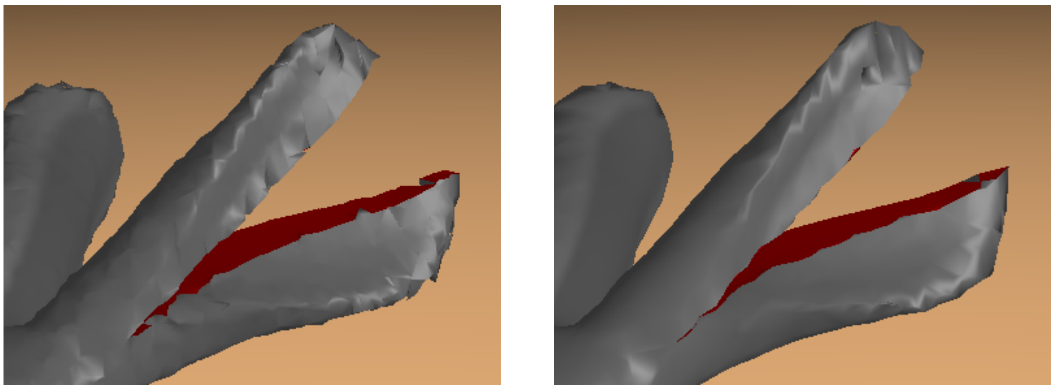


Figure 4.3: *A comparison of mesh quality of the splitting cubes algorithm (left) and the dual contouring approach (right).*

and normals at the intersections of the boundary surface with the links. As described before, the positions of these intersections on a link as well as the normals at these points are stored whenever a link is cut by the blade. Thus, the quadratic function can be evaluated in turn. As proposed in the original dual contouring algorithm, we then connect representative vertices in adjacent cells to construct the surface mesh.

Another advantage of the dual contouring algorithm is the quality of the resulting surface mesh. In the splitting cubes algorithm, the interior vertex is determined for the first cut by averaging face vertices or minimizing a quadratic function, depending on the angle formed by cutting planes. For successive cuts, the interior vertex is projected to a new cut plane. Thus it is processing order dependent. Our implementation results (see Figure 4.3) show that the dual contouring approach produces better surface meshes than the splitting cubes algorithm. Note that to handle singularity situations where cut planes are almost parallel, in our implementation of splitting cubes, the position of the interior vertices is explicitly clipped to ensure it is inside the cell, whereas in dual contouring, the singularity is handled by clipping singular values during solving the minimization of the quadratic function [JLSW02].

So far, the meshing procedure generates one surface part for every cell that is cut. However, a cut divides the material into multiple disconnected parts, for each of which a boundary surface has to be computed. To achieve this, we duplicate the representative vertices in the interior of each cell that is cut, as many times as there are disconnected material parts in this cell. As illustrated in Figure 4.2, each duplicated vertex—associated to a particular disconnected part—is bound to the nearest vertex of the respective part in the dual cell containing this vertex.

To accelerate the meshing procedure, the number of duplicated copies and the connectivity of an interior vertex to a cell's vertices are precomputed and stored in a look-up-table. This table has 2^{12} entries, each of which corresponds to one cutting pattern of the cell. At runtime, for a given representative vertex, a number of disconnected material parts, and a classification of the cell vertices to the parts the binding of the center vertices can be determined efficiently.

Mesh Deformation

To let the reconstructed surface move according to the object deformations, we use the binding that has been established between the surface vertices and the dual grid vertices. Since every dual grid vertex corresponds to exactly one primal grid cell, i.e., the fine-grid cells, every surface vertex is bind to exactly one primal cell. Thus, the deformation of these cells—which is computed from the deformations of the simulation grid as described later on—is carried over to the surface vertices via trilinear interpolation of the primal cell vertices.

4.4 Composite Finite Element Simulation

A major design goal of the presented approach is to achieve interactive simulation update rates. Starting from the adaptive octree grid described in the previous section, one way to create a finite element model would be to create a hexahedral element with trilinear shape functions for each octree cell, with three degrees of freedom (DOFs) located at each non-hanging vertex. However, since we employ a high-resolution octree grid to accurately represent complicated cuts and the boundary of the object, this approach would lead to a very large number of DOFs, far exceeding the number of DOFs that can be simulated at interactive update rates.

In this chapter, we use composite finite elements [HS97, SW06] to reduce the number of DOFs and thus to achieve interactive update rates. A composite finite element is obtained by combining multiple 'standard' elements into a single element. In particular, the shape functions of the composite finite element are assembled from the shape functions of the individual elements.

We use a hexahedral composite finite element model, which is assembled from a 'standard' finite element model with one hexahedral element per octree cell. In the following we will use the term 'composite element' and 'hexahedral element' to refer to the elements of the respective model. Similar to the work by Preusser et al. [PRS07, LPR+09], we use an inverse compositing scheme in that we start with trilinear shape functions on the composite finite elements, and define the trilinear shape functions of the hexahedral elements via restriction of the composite element shape functions to the domains of the individual hexahedral elements. Note that the DOFs of the hexahedral elements remain located at the vertices of the composite elements. Since it is explicitly allowed that a composite element is only partially filled with hexahedral elements, objects with complicated boundaries can be effectively discretized using a small number of elements.

In addition to the handling of complicated boundaries, another challenge is to handle complicated topologies in a coarse simulation model. In the original works by Preusser et al., the composite finite elements are assembled from the hexahedral elements by only considering their spatial location, disregarding the connectivity between the elements. For our application of interactive cutting, this would mean that material parts separated by a cut would possibly be merged into the same composite element, preventing the opening of the cut in the simulation.

We therefore employ a strategy to build a composite finite element model that precisely represents the topology of the object. Our strategy is based on analyzing the connectivity between elements to possibly create more than one composite element at the same location, each representing a separated material part. A similar approach has been used by Nesme et al. [NKJF09] and Jeřábková et al. [JBB$^+$10] to model separated material parts in coarse finite element models, and by Aftosmis et al. [ABA00] and Dick et al. [DGW11a] to represent separated material parts in the multigrid hierarchy of a geometric multigrid solver.

Our simulation is based on the linear theory of elasticity. To accurately simulate deformations with large rotations, we use the corotational formulation of strain.

4.4.1 Construction of the Simulation Model

The composite finite element model is based on a grid which is significantly coarser than the adaptive octree grid used to represent the volume of the deformable object. We build this coarser grid from the adaptive octree grid by successively removing leaf nodes from the associated octree. Note that the current set of leaf nodes corresponds to the current set of grid cells of the adaptive grid. In our implementation, we remove all nodes from octree levels $0, \ldots, \ell-1$, where level number 0 denotes the finest level of the octree. In this way we obtain a coarsened adaptive octree grid for the composite finite element model, where the cells on the finest level coincide with blocks of $(2^\ell)^3$ cells on the finest level of the initial grid. It is worth noting that instead of using this uniform coarsening strategy, it would also be possible to use an adaptive strategy for removing octree nodes, for example to use a higher simulation accuracy in certain regions of

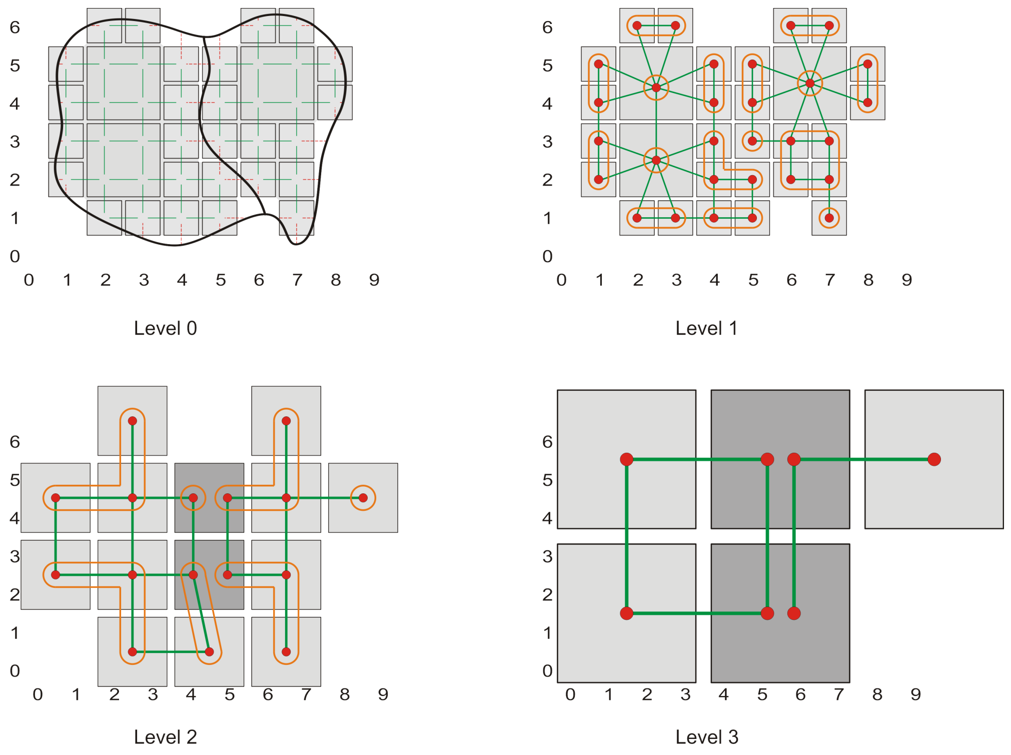


Figure 4.4: *Hierarchical construction of the composite finite element model. Left: 2D illustration of the adaptive linked volume representation, consisting of a set of rendering nodes (gray) which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary (black) and along the newly generated surface due to cutting (blue). Middle left to right: Iterative coarsening of the finite element model. The underlying graph representation is indicated by red vertices and green edges. For each block of* 2^3 *cells on the respectively finest level the connected components (orange) are determined, and the elements of each connected component are replaced by a separate composite finite element.*

interest.

The basic idea underlying the construction of the composite finite element model is to analyze the connectivity of the material within each coarse-grid cell, and to create an individual composite element for each connectivity component. In this way, separated material parts are modeled by different elements, enabling the opening of a cut in the simulation.

The topologies of the composite finite element model and the hexahedral finite element model are each represented by an undirected graph, where the nodes corresponds to the elements, and the edges specify the connectivity between elements. For the hexahedral finite element model, the graph is directly obtained from the linked volume representation (see Figure 4.4).

The construction of the composite finite element model is performed in a two-step process. In the first step, we create the composite finite elements by considering the subgraph of the hexahedral finite element model induced by each coarse-grid cell. We determine the connectivity components of this subgraph by using a depth-first search, and for each connectivity component, we create one composite element, which exactly subsumes the hexahedral elements corresponding to the nodes of this connectivity component. In the second step, the connectivity between the composite finite elements is determined. Two composite elements are connected, if there exists two connected hexahedral elements such that one hexahedral element is merged into the first, and the other into the second composite element. Finally, a shared vertex representation is computed for the composite finite element model. Two connected elements share a common face, and in particular the vertices incident to this face.

Instead of creating the target resolution of the composite finite element model directly from the hexahedral finite element model, we use an iterative coarsening approach, in that we iteratively apply the described scheme by only removing the leaf nodes of the finest octree level in each step (i.e., $\ell = 1$) (see Figure 4.4).

4.4.2 Computation of Element Matrices

We assemble the stiffness and mass element matrices for the composite finite elements from the stiffness and mass element matrices of the underlying hexahedral finite elements. The deformation behavior of the deformable body is described by the physical principal of total potential energy minimization, applied to each single point in time. For a standard hexahedral finite element discretization, the total potential energy E is

$$E(u) = \frac{1}{2}u^{\mathrm{T}}Ku - (f - M\ddot{u})^{\mathrm{T}}u, \tag{4.2}$$

where u is the linearization of the displacement vectors at non-hanging vertices of the hexahedral finite element grid, f is the linearization of the external force vectors applied at these vertices, and K and M denote the global stiffness and mass matrix, respectively. Minimizing this energy via $\frac{\partial}{\partial u}E(u) = Ku - (f - M\ddot{u}) = 0$ leads to the well-known spatially discretized equation of motion.

Using composite finite elements, the DOFs are located at the vertices of these composite elements. The displacements at the vertices of the underlying hexahedral finite elements are determined by trilinear interpolation from the vertices of the composite finite elements. This is described by the equation $u = I\tilde{u}$, where $\tilde{u}$ denotes the linearization of the displacements at the vertices of the composite finite element grid, and the interpolation matrix I expresses the trilinear interpolation from these vertices.

For a composite finite element discretization, the total potential energy E thus is

$$E(I\tilde{u}) = \frac{1}{2}\tilde{u}^{\mathrm{T}} I^{\mathrm{T}} K I \tilde{u} - (f - MI\ddot{\tilde{u}})^{\mathrm{T}} I\tilde{u}. \tag{4.3}$$

This energy is minimized via

$$\frac{\partial}{\partial \tilde{u}} E(I\tilde{u}) = \underbrace{I^{\mathrm{T}} K I}_{\tilde{K}} \tilde{u} - (\underbrace{I^{\mathrm{T}} f}_{\tilde{f}} - \underbrace{I^{\mathrm{T}} M I}_{\tilde{M}} \ddot{\tilde{u}}) = 0. \tag{4.4}$$

Thus, the global stiffness and mass matrix for the composite finite element discretization are obtained via $\tilde{K} = I^{\mathrm{T}} K I$ and $\tilde{M} = I^{\mathrm{T}} M I$, respectively, and the external forces which are specified on the underlying hexahedral finite element discretization are propagated to the composite finite element discretization via $\tilde{f} = I^{\mathrm{T}} f$.

Applying these equations to a single composite finite element c, its element matrices $\tilde{K}^c$ and $\tilde{M}^c$ can be assembled from the element matrices $\tilde{K}^e$ and $\tilde{M}^e$ of the underlying hexahedral elements e that are merged into the considered composite element via

$$\tilde{K}^c_{mn} = \sum_{e \text{ in } c} \sum_{i=1}^{8} \sum_{j=1}^{8} w^{c \to e}_{m \to i} w^{c \to e}_{n \to j} K^e_{ij}, \quad m, n = 1, \ldots, 8 \tag{4.5}$$

($\tilde{M}^c$ is computed analogously). Here, the first sum iterates over the hexahedral elements e that are merged into the composite element c. Note that the element matrices are interpreted as 8×8 matrices with each entry being itself a 3×3 matrix. Thus, the notation K^e_{ij} denotes a 3×3 block of scalar entries.

The trilinear interpolation weights $w^{c \to e}_{m \to i}$ from the vertices $m = 1, \ldots, 8$ of the composite element c to the vertices $i = 1, \ldots, 8$ of the hexahedral element e are defined as

$$w^{c \to e}_{m \to i} = \left(1 - \frac{|x^c_m - x^e_i|}{s^c}\right)\left(1 - \frac{|y^c_m - y^e_i|}{s^c}\right)\left(1 - \frac{|z^c_m - z^e_i|}{s^c}\right), \tag{4.6}$$

where (x^c_m, y^c_m, z^c_m) and (x^e_i, y^e_i, z^e_i) are the coordinates of the vertices, and s^c denotes the edge length of the composite element.

Instead of creating the target resolution of the composite finite element model directly from the hexahedral finite element model, it is also possible to iteratively apply the described scheme by only removing the leaf nodes of the finest octree level in each step (i.e., $\ell = 1$). In this case, many of the interpolation weights $w^{c \to e}_{m \to i}$ are zero, which effectively reduces the number of arithmetic operations needed for assembling the composite finite element matrices, and leads to a speedup of about 1.5 compared to the direct approach.

Considering the element matrices of the underlying hexahedral finite elements, since all elements have the same shape, these elements can be generated from a single element stiffness and mass matrix. Let K^e and M^e denote the element stiffness and mass matrix for a generic element of side length 1, then the element stiffness and mass matrix for an element of side length s is sK^e and $s^3 M^e$, respectively. Furthermore, the element stiffness and mass matrix scales linearly with the Young's modulus and the material density, respectively.

The fixation of the object is also specified on the underlying hexahedral finite element discretization. We propagate this fixation to the composite finite element discretization by fixing exactly those vertices on the composite element grid which are related to a fixed vertex on the underlying hexahedral element grid via a non-zero interpolation weight.

In case of simulating an object consisting of homogeneous material, it is possible to accelerate the assembly of the composite element matrices by employing a look-up table for coarsening of the finest level of the adaptive octree grid. Consider a coarse-grid cell with a domain of 2^3 fine-grid cells, we precompute the composite element matrices for all possible patterns of fine-grid cells being empty or filled with material. Therefore, this look-up table has a size of $2^{(2^3)} = 256$ entries, and a size of 1.1 MB. By using the look-up table, the assembly of the composite finite element matrices is accelerated by a factor of about 2.

In our implementation, the composite finite element matrices are reused between successive simulation time steps. Since a progressive cut in a single time step only affects a very small portion of the model, in each time step only a small number of composite element matrices have to be (re-)assembled, which greatly reduces the computational costs per time step.

4.4.3 Corotational Strain Formulation

To accurately simulate deformations with large rotations using the linear theory of elasticity, we employ the corotational formulation of strain [RB86, HS04]. Since in the linear theory a linear strain tensor is employed, which interprets rotations as strains, the solution significantly diverges from the correct solution with increasing rotations within the deformation. The basic idea underlying the corotational formulation of strain is to remove the rotations before the linear strain is computed, and later re-add these rotations to the resulting stresses. The corotational formulation is based on the finite

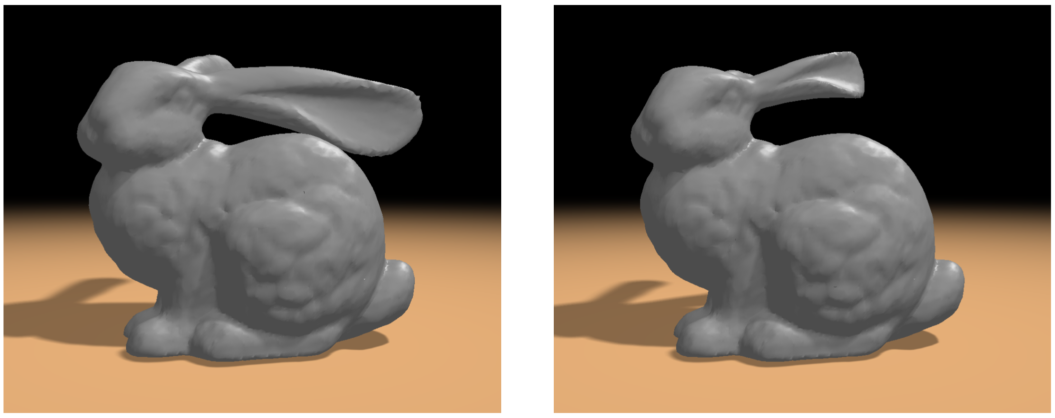


Figure 4.5: *Composite finite element deformations using the linear infinitesimal strain (left) and the corotational strain formulation (right) under the same external forces applied to the bunny ear. Note that the unrealistic large deformation using the infinitesimal strain tensor is not found in the simulation using the corotational strain formulation.*

element discretization by computing a single rotation for each finite element.

In our approach, we employ the corotational strain formulation on the composite finite element discretization. To determine the rotation of a composite finite element c, we first compute the average deformation gradient $\overline{\nabla\varphi}^c$ via

$$\overline{\nabla\varphi}^c = \Big(\sum_{e \text{ in } c} \int_{\Omega^e} \nabla\varphi \, dx\Big) \,/\, \Big(\sum_{e \text{ in } c} \int_{\Omega^e} 1 \, dx\Big), \tag{4.7}$$

where $\varphi(x) = x + u(x)$ denotes the deformation, and Ω^e denotes the domain of element e. The rotation matrix is then obtained by polar decomposition of the average deformation gradient [Hig86]. Note that our formulation of deformation gradient considers that a composite finite element might be only partially filled with hexahedral elements. Otherwise, an incorrect rotation of the composite finite element may lead to numerical instability. The effect of the corotational strain formulation for composite finite elements is shown in Figure 4.5.

4.5 Geometric Multigrid Solver

Applying the Newmark time integration scheme to the spatially discretized equation of motion leads to a linear system of equations in every simulation time step. We solve this system using the geometric multigrid approach proposed by Dick et al. [DGW11a].

A multigrid solver is optimal in the sense that it exhibits asymptotic linear runtime in the number of unknowns. Considering the error between the current approximate solution and the exact solution in an iterative solution scheme, standard relaxation schemes such as Jacobi or Gauss-Seidel relaxation effectively reduce high-frequency error components, but they are inefficient in reducing low-frequency error components. This typically causes the error reduction to stall after a few relaxation steps. The basic idea of multigrid is to improve the convergence by solving the system on a hierarchy of successively coarser grids, such that each frequency range of the error can effectively be reduced on the appropriate scale.

For the considered application of cutting of deformable objects, the challenge is to build a multigrid hierarchy that represents the arising complicated mesh topologies on the coarser levels. In particular, ignoring the topology and simply merging cells based on their spatial location would lead to a significant reduction of the convergence rate compared to the case without cuts. The approach of Dick et al. [DGW11a] uses a strategy for generating a multigrid hierarchy that reflects the cuts on the coarser levels which is similar to the strategy proposed in this chapter for generating the composite finite element grid from the underlying hexahedral element grid. For details we would like to refer the reader to the original work.

In contrast to the original work, where the multigrid solver is applied to an adaptive octree finite element discretization, in this chapter the solver is applied to a composite finite element discretization. We use 3 V-cycles per time step, with 1 pre- and 1 post-smoothing Gauss-Seidel steps, using an over-relaxation parameter of 1.7. On the coarsest level, a Cholesky solver is employed. For the model sizes presented in this chapter, we use 2 or 3 grid hierarchy.

4.6 Results

In the following we analyze the potential of our proposed cutting approach for interactive applications like virtual surgery simulations. The major focus in this analysis is on the performance analysis of the composite finite-element approach and the quality analysis of the generated surfaces. For a thorough analysis of the convergence behavior of the multigrid finite-element approach, including a detailed comparison with alternative solvers, let us refer to [DGW11a]. We present a number of experiments, all of which were run on a standard desktop PC with an Intel Xeon X5560 processor running at 2.80 GHz (a single core is used), 8 GB of RAM, and an NVIDIA GeForce GTX 480 graphics card.

Table 4.1 shows a detailed performance statistics for interactive cutting of the Stanford bunny model (see Figure 4.1), the Armadillo model (see Figures 4.6 and 4.7), and

	Comp.	Hex.	# DOFs of	Contour grid	# Tris	Memory	Time [ms]						Speedup
Model	size	size	simulation	resolution	× 1k	[MB]	Cut	Contour	Compose	Solve	Intpl.	Total	over full
Armadillo	1	1/2	1,794 / 2,052	22×20×26	3	2.45	1.29	0.08	0.48	5.65	0.14	7.64	3.2×
Armadillo	1	1/4	2,205 / 2,439	43×39×51	12	13.48	3.56	0.33	2.55	6.63	1.07	14.14	6.1×
Armadillo	1	1/8	2,256 / 2,514	85×77×101	49	67.02	14.11	1.30	14.94	6.86	6.41	43.62	10.3×
Armadillo	1	1/16	2,289 / 2,547	169×153×201	197	311.68	44.86	5.23	71.23	7.07	28.53	156.92	17.8×
Armadillo	1/2	1/16	8,895 / 10,563	169×153×201	252	385.80	43.01	5.48	69.10	29.37	29.41	176.37	16.0×
Bunny	1	1/16	735 / 1,161	101×78×100	69	112.61	18.41	2.88	43.42	3.37	14.14	82.22	17.2×
Bunny	1/2	1/16	3,051 / 4,485	101×78×100	69	112.61	17.34	2.79	40.27	11.18	14.17	85.75	16.1×
Bunny	1	1/8	735 / 1,149	51×39×50	17	24.49	4.85	1.08	14.04	3.76	3.14	26.87	10.1×
Filigree	1	1/8	9,399 / 9,606	250×38×251	256	341.68	13.04	0.19	35.03	22.07	31.99	102.32	25.2×

Table 4.1: *Performance statistics for different models and different resolutions of the fine-grid finite elements.*

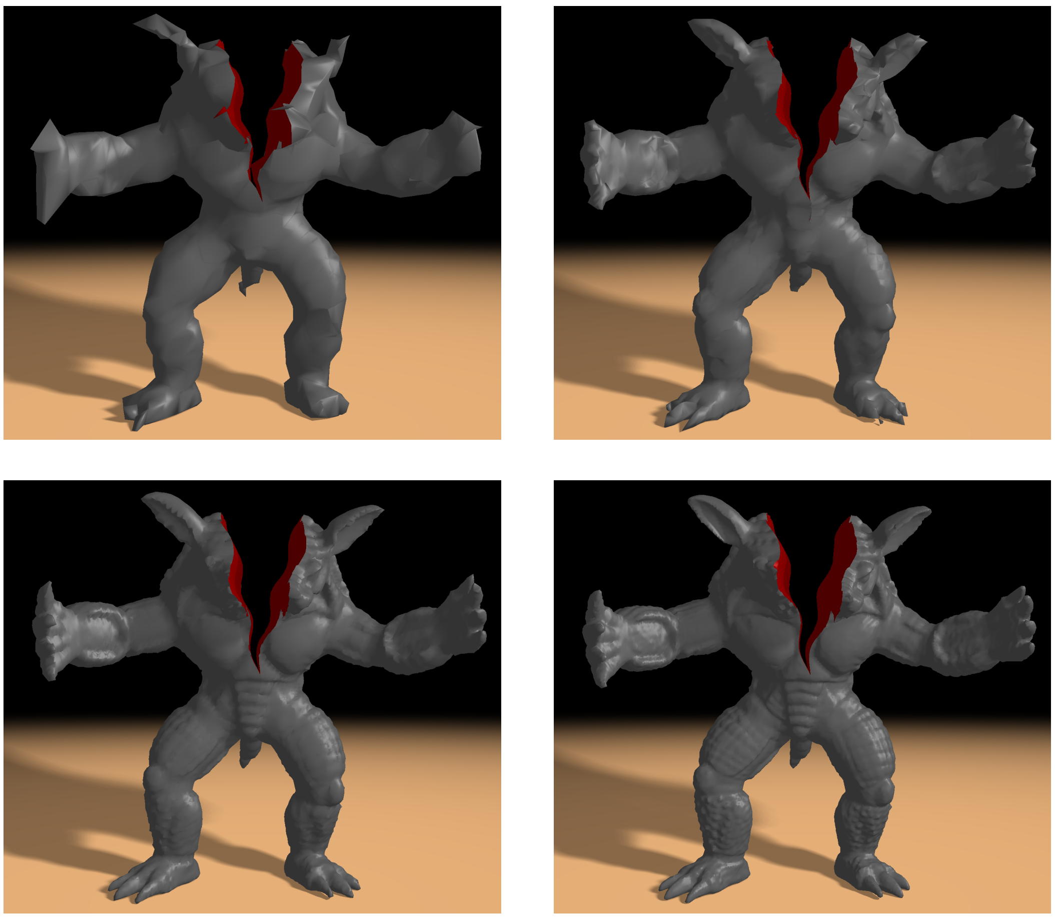


Figure 4.6: *Composite finite-element embedding of the armadillo model. In all images the resolution of the simulation grid is the same, but the hexahedral finite-element representation from which the stiffness matrices are assembled and the surface is reconstructed is adaptively refined along the cuts using 1, 2, 3, and 4 refinement levels, respectively, from top left to bottom right.*

the Filigree model (see Figure 4.8). The second group of columns indicates the size of the composite element and the finest hexahedral element. The size is unified for simplicity. The third group gives information of the models. The first column in this group gives the number of the DOFs we are solving before and after a cut has been performed. The increasing number of DOFs is caused by the duplication of simulation elements that are cut. The next column shows the hexahedral grid resolution to which the adaptive refinement corresponds and from which the surface mesh is extracted. The last two columns give and the number of triangles that are reconstructed, and the memory requirements of our approach.

During cutting a new surface has to be reconstructed from the refined volume rep-

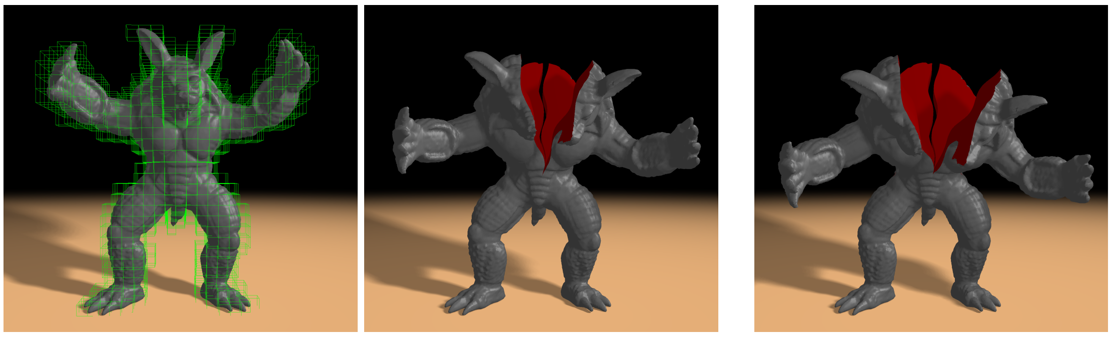


Figure 4.7: *Left: The coarse hexahedral simulation grid and the high-resolution surface that is constructed from the embedded fine grid. Middle: Simulation using composite elements obtained from coarsening the hexahedral finite element model by three levels (10,563 DOFs). Each simulation time step takes 176.37 ms. Right: Simulation of cuts in the Armadillo model using hexahedral finite elements (569,748 DOFs). Each simulation time step takes 2829.87 ms.*

resentation, and the resulting changes in the fine level elements, i.e., their stiffness matrices, have to be propagated to the coarse simulation level to rebuild the stiffness matrices at this level. Since cutting is always performed incrementally, meaning that in every frame the blade is moved only a small distance through the material, the aforementioned update operations have to be performed only locally around the cutting region. The timings required for incrementally performing the cut, reconstructing the surface, and updating the composite finite elements are listed in the first part of group Time. In our examples, the cuts were realized such that in average the number of newly created fine level elements was about 3% of the total number of elements. From our experiences this seems to quite realistically simulate the speed at which a cut is performed in reality. It is clear, on the other hand, that a faster cut can significantly increase the number of affected elements per frame, resulting in vastly increasing simulation times. The second part of this group lists the times required for solving the system of equations on the coarse simulation level and interpolating the computed displacement at the surface vertices. The last column in this group gives the total simulation time per time step.

We compare in the last column the performance of our composite finite elements approach with the full simulation on the finest level as presented in [DGW11a]. In this comparison, the incremental updating of surface and stiffness matrices is also applied to the full simulation. The speedup of overall performance depends on the refinement level. With three levels composition, the speedup of our method is about 10× for Armadillo and Bunny models, and 25× for the Filigree model.

Our statistics indicate that the proposed cutting approach can be used very effec-

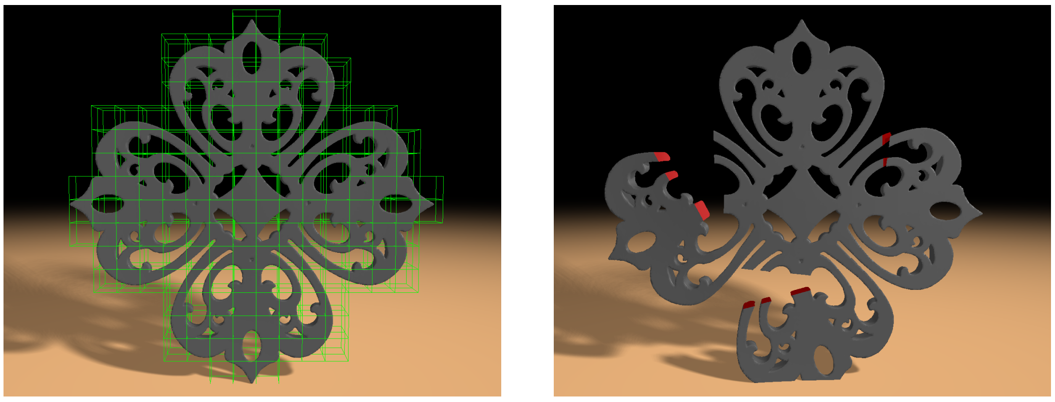


Figure 4.8: *Cutting on a Filigree model with complex topology. Top: The coarse hexahedral simulation grid. Bottom: The deformation of the body after cutting has been performed. Cutting, surface construction, and deformation simulation are performed at 10 simulation frames per second.*

tively in interactive scenarios requiring high update rates and high-resolution boundary surfaces, at reasonable approximation quality. Even the highest resolution Armadillo model can be cut, simulated, and rendered at 5-6 fps on a single core of our target architecture. Especially the possibility to reproduce sharp material features induced by a cut at high quality distinguishes our approach from previous approaches.

To demonstrate the approximation quality of the composite finite-element approach, we have performed one additional test using the Armadillo model. The result of this test is shown in Figure 4.7. In the middle, the model is simulated using composite elements obtained from coarsening the hexahedral finite element model by three levels (10,563 DOFs), while on the right it is simulated using hexahedral finite elements (569,748 DOFs). The simulation time step takes 2829.87 ms for the full simulation and 180.99 ms for simulating the composite elements.

It can be seen that the composite finite element approach yields a slightly different deformation than the full resolution approach, because a much lower number of DOFs is solved. On the other hand, even for the rather large disconnected parts that are generated by the cut, which undergo large displacements due to their high masses, the differences are not too severe. Especially in virtual surgery simulation, where typically only small cuts and incisions are performed by a doctor, these difference become much less significant.

4.7 Conclusion

We have presented several highly effective improvements to existing cutting approaches using composite finite elements, with the goal to efficiently generate a high-resolution boundary surface from a much coarser simulation grid. By using composite finite elements, in combination with an adaptive refinement of an embedded finite-element grid along the cuts, a very high-resolution surface representation is achieved. Since we use the dual contouring approach for reconstructing this surface, sharp features are reproduced, and compared to previous approaches a significantly smaller number of triangles is generated. Even though the simulation accuracy depends on the number of DOFs we solve on the coarse simulation grid, by assembling the stiffness matrices on this grid from the matrices on the adaptively refined volume representation, topological changes can be simulated at very good approximation quality.

Chapter 5

Efficient Collision Detection for Composite Finite Element Simulation of Cuts in Deformable Bodies

In the previous chapter, we have demonstrated the efficiency of composite finite elements for virtual cutting of elastic bodies. In this chapter we present an efficient collision detection method which is specifically tailored for composite elements. Our method exploits the specific characteristics of CFEs, i.e., the fact that the number of simulation degrees of freedom is significantly reduced. We show that this feature not only leads to a faster deformation simulation, but also enables a faster collision detection. To address the non-conforming properties of geometric composition and hexahedral discretization, we propose a topology-aware interpolation approach for the computation of penetration depth. We show that this approach leads to accurate collision detection on complex boundaries. Our results demonstrate that by using our method cutting on high-resolution deformable bodies including collision detection and response can be performed at interactive rates.

5.1 Introduction

The physically-based simulation of cuts in deformable bodies can significantly improve the realism in surgery simulators and computer games. To efficiently compute the de-

This chapter is based on material that has been originally published in J. Wu, C. Dick, and R. Westermann, *Efficient collision detection for composite finite element simulation of cuts in deformable bodies*, The Visual Computer (Proc. Computer Graphics International 2013) **29** (2013), no. 6-8, 739–749. The copyright for this material is owned by Springer.

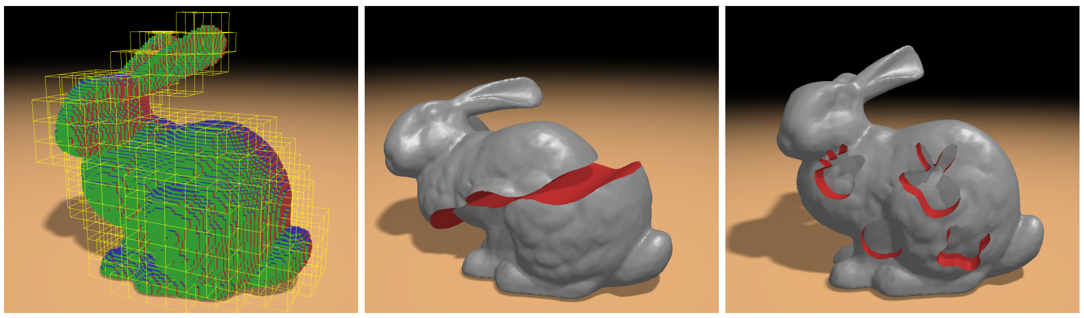


Figure 5.1: *Cutting of the Stanford Bunny model. Left: High-resolution hexahedral elements and coarse composite elements (yellow grid). Middle: The deformable bunny is cut into two parts, resulting in the upper part sliding down. Right: After four stamp-like cuts, the resulting cylinders slide out under tight constraints. Collision detection and finite element simulation are performed at 41 fps. Interactive cutting of the deformable body takes additional 37 ms per simulation frame for updating the surface mesh, the stiffness matrices, and the distance field.*

formation, composite finite elements (CFEs) [HS97] based on a uniform or adaptive hexahedral (octree) discretization of the simulation domain have recently been adopted in the computer graphics community [NKJF09, DGW11a]. The idea is to approximate the finite element discretization of the governing system of partial differential equations on a high-resolution grid by subsuming blocks of finite elements into coarser elements, thereby reducing the number of simulation degrees of freedom (DOFs) and thus trading performance for a moderate decrease of accuracy. Additionally exploiting the fact that during cutting only a small number of elements are modified in each simulation frame, [JBB+10] and [WDW11] have further demonstrated that interactive update rates—12 fps including mesh cutting, surface reconstruction, and deformation computation—can be achieved for simulating progressive cuts in elastic bodies with an effective resolution of 100^3 hexahedral elements.

Besides the pure deformation simulation, efficient collision detection is another essential component in interactive virtual environments. For deformable bodies with changing topology, collision detection is particularly time consuming, since new geometric primitives are created on-the-fly. As a consequence of cutting, an object may be split into several separated objects. It is therefore necessary to consistently detect both inter- and intra-collisions. Moreover, a quantitative measure of the penetration is desired for robust collision response.

Considering the promising results of CFEs, it is highly interesting to investigate collision detection in this specific context. On the one hand, CFEs have the potential to simplify collision detection, due to their nature of reducing the number of finite elements. On the other hand, they present several challenges with respect to the underlying non-conforming geometric structures, i.e., the coarse composite elements (which might be duplicated to represent separated material parts on each side of a cut) versus the underlying hexahedral elements, and the hexahedral discretization versus a triangulated surface mesh representation. The non-conformity is particularly severe in the case of cuts: The gap of a cut has an initial width of zero. In contrast, a naive collision detection approach purely based on hexahedral elements can only achieve an accuracy in the order of the hexahedral grid spacing.

In this chapter we present an efficient collision detection method for interactive CFE simulation of cuts applied to high-resolution deformable bodies. Our method exploits the composition structure of CFEs to speed up the collision query, and addresses the non-conformity of the underlying geometric structures by analyzing the topology of the hexahedral element grid in order to improve collision accuracy. Our method detects both intra-object and inter-object collisions, and also supports application scenarios where one of the objects is not closed (e.g., a thin scalpel).

The specific contributions of this chapter are:

- A collision detection algorithm which exploits the specific characteristics of CFEs. The complexity of our broad phase collision detection depends on the number of simulation DOFs, instead of the number of geometrical primitives. This leads to a speedup factor which scales exponentially with respect to the level of composition.

- A topology-aware interpolation scheme to determine the penetration depth on the non-conforming hexahedral element grid. By flipping the sign of distance values associated with hexahedral elements based on analyzing their topological connectivity, the accuracy of penetration depth interpolation around the boundary is improved.

The remainder of this chapter is organized as follows: In the next section, we summarize work that is related to ours. In Section 5.3, we briefly review CFE simulation of cuts in deformable bodies. In Section 5.4, we then present our collision detection approach. In Section 5.5, we describe the distance field computation and updating. Detailed timing statistics and evaluations are provided in Section 5.6, and the chapter is concluded in Section 5.7.

5.2 Related Work

Composite finite elements approximate a high-resolution finite element discretization of a partial differential equation by means of a small set of coarser elements. Such elements have originally been invented to resolve complicated simulation domains with only a few degrees of freedom, and are also used in the context of geometric multigrid methods to effectively represent complicated object boundaries at ever coarser scales [HS97, SW06, PRS07, LPR+09].

In computer graphics, the idea of CFEs has been used as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation [NKJF09], and just recently to model material discontinuities that are caused by cuts and incisions, for instance, to improve the convergence of a geometric multigrid solver [DGW11a] and to reduce the number of simulation DOFs in order to increase simulation performance [JBB+10, WDW11].

A high quality surface mesh reconstructed after cutting can serve as a good basis for collision handling. Jeřábková et al. [JBB+10] model a cut by removing elements on the finest resolution level, and reconstruct the boundary surface by means of the Marching Cubes algorithm. The element removal approach, in general, makes it difficult to construct a surface that accurately aligns with a cut. Seiler et al. [SSSH11] propose a method that decouples the resolution of the material surface from the resolution of the simulation grid, but the method is restricted to volumetric non-progressive cuts. Dick et al. [DGW11a] model progressive cuts by disconnecting links between hexahedral elements, and reconstruct the surface by a splitting cubes algorithm [PGCS09], taking into account the exact intersections of the cutting blade and the links. Based on this work, Wu et al. [WDW11] use CFEs to reduce the number of simulation DOFs, and propose a dual contouring approach [JLSW02] to construct a high quality surface which is accurately aligned with a cut.

Collision detection for general deformable bodies has been widely studied and an excellent survey is given by Teschner et al. [TKH+05].

To simulate deformable bodies with dynamically created geometric primitives, we are interested in collision detection methods that do not rely on heavy precomputation. Spatial hashing [THM+03] requires no preprocessing of surface meshes, and detects both self-collisions and collisions between different bodies. Layered depth images (LDIs) [HZLM01, HTG04, FBAF08] hold these properties as well, but are limited to closed manifold objects. Bounding volume hierarchies (BVHs) are optimized to handle dynamic topologies [OCSG07, HSK+10], and can be combined with locally updated distance fields to simulate brittle fractures [GSM+12].

For simulating reduced deformations governed by only a few DOFs, James and Pai [JP04] have demonstrated that precomputed BVHs can be updated at a cost proportional to the number of simulation DOFs. Built on intensive precomputation of collision-free certificates, reduced structures have also been exploited for self-collision processing [SGO09,BJ10]. These algorithms are designed for reduced deformable bodies with consistent topology. CFEs can be considered as a special kind of a locally reduced model.

To facilitate dynamically created geometric primitives and to detect both inter- and intra-collisions, for CFEs we propose to utilize spatial hashing in the broad phase of collision detection. The exact penetration information at complicated boundaries is then evaluated with a novel topology-aware interpolation scheme.

To compute the penetration depth for collision response, a practical workflow for interactive applications [HFS+01, FL01, THM+03, MZS+11] is to first find potentially overlapping pairs of a vertex and a volumetric element. The vertex is then transformed, with respect to this volumetric element, from the deformed configuration to the reference configuration, where the penetration depth and direction of the vertex are evaluated. Finally, the penetration information is transformed back to the deformed configuration and utilized to compute the collision force. High update rates can be achieved by using a precomputed distance field in the reference configuration.

5.3 Composite Finite Element Simulation of Cuts

(We reiterate some important properties of CFEs for chapter completeness, with a focus on the implications of composite elements to collision detection. For a detailed explanation of CFEs, we refer the reader to [WDW11] and Chapter 4.)

Our collision detection method is particularly designed for CFE simulation, which has recently gained popularity to simulate cuts in medical applications [JBB+10,SSSH11, WDW11]. In the following, we briefly summarize the main principles of CFE simulation of cuts in deformable bodies.

For CFE simulation, three coupled geometric representations are employed to describe a deformable body (see Figure 5.2 for an illustration).

Hexahedral elements. The deformable body is discretized by means of trilinear hexahedral elements that are aligned on a restricted octree grid. This grid is adaptively refined along the surface (including cutting surfaces) of the object. To simplify the discussion, a uniform Cartesian grid is assumed in the following. Physical properties

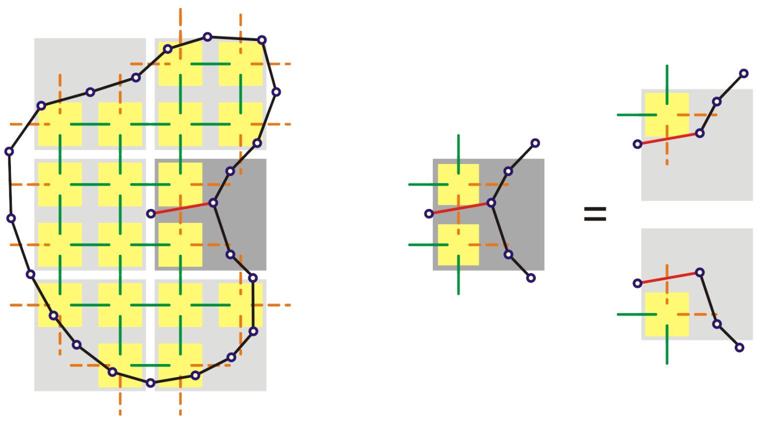


Figure 5.2: *A deformable body is represented by linked hexahedral elements (yellow), from which composite elements (gray) and a surface mesh (black) are constructed. Note that the gaps between elements are purely illustrative, i.e., in the finite element model the elements are directly attached to each other. The dark gray color indiciates that more than one (here: two) composite elements exist at the same location, corresponding to the distinct material parts separated by the cuts. Links are shown in green and orange colors if marked as connected and disconnected, respectively.*

(i.e., Young's modulus and Poisson's ratio) are assigned on a per-element basis. The topology is represented by links between face-adjacent elements. Cuts are modeled by marking links as disconnected. For each link that is cut the intersection point with the cutting blade and the blade's normal at that point are stored. This linked hexahedral element representation is the basis for constructing the following two representations, which are used for efficient deformation computation and high quality visual rendering, respectively.

Composite elements. From the linked hexahedral elements, composite elements are constructed by considering $2^k \times 2^k \times 2^k$ blocks of hexahedral elements. In each block, we analyze the connectivity among the elements, and create one composite finite element for each connectivity component. In this way we create one composite finite element for the distinct material parts separated by cuts, and thus accurately represent cuts in the composite finite element discretization. As a consequence, multiple composite finite elements might exist at the same location. An example for duplicated composite elements is shown Figure 5.2, indicated by a darker cell color. From an implementation point of view, the composition process is performed in k iterations by successively considering $2 \times 2 \times 2$ blocks. While Figure 5.2 illustrates an example of one-level composition ($k = 1$), the composite elements can be several levels coarser than the hexahedral elements, thereby trading speed for a moderate decrease of simulation accuracy. Note that when the underlying hexahedral element grid is an adaptive octree grid, the composite element grid is also an adaptive octree grid.

The deformation computation is performed on the composite elements. From a mathematical point of view, this is achieved by starting with a finite element discretization on the fine hexahedral elements, and substituting the DOFs of the hexahedral elements by means of trilinear interpolation from the DOFs of the composite elements.

Surface mesh. By means of a dual contouring approach, a smooth surface triangle mesh is constructed on the dual grid consisting of the links between hexahedral elements by utilizing the intersection points and the normals of the cutting blade stored at these links [WDW11]. The surface is bound to the hexahedral element model by assigning each vertex to its topologically connected, closest hexahedral element. In this way, the computed deformation of the finite element model can be transferred to the surface mesh.

The three representations are geometrically non-conforming, i.e., a composite finite element may be only partially filled with hexahedral elements, and the surface mesh arbitrarily intersects the hexahedral element grid.

In each simulation time step, the computed per-vertex displacements of the composite elements therefore are propagated to the hexahedral elements and then to the surface mesh vertices, both by means of trilinear interpolation. It is worth noting that a) parts of the triangle mesh associated with the same composite element cannot self-collide, since the coordinates of the surface mesh vertices are trilinearly interpolated from those of the same composite element, and that b) for a given vertex position, inferring between the trilinear interpolation weights with respect to the composite element and the trilinear interpolation weights with respect to an underlying hexahedral element is straightforward.

5.4 Collision Detection for CFE Simulation of Cuts

From an abstract point of view, we consider collision detection as querying whether a given vertex penetrates into a solid object, and if yes, reporting the penetration depth and direction to be used for computing the collision response. Since the querying vertex can be from the same object or another one, this abstraction handles both inter- and intra-collisions. Note that the vertex can also be from a non-closed object, e.g., a thin blade in surgery simulations.

Our collision detection approach consists of a broad and a narrow phase. In the broad phase (Section 5.4.1), potential overlaps between surface vertices and composite elements are detected. This phase is performed in the deformed configuration. Testing coarse composite elements instead of fine hexahedral elements leads to a significant speedup.

In the narrow phase (Section 5.4.2), we determine the penetration depth for each potentially colliding vertex by transforming the composite element/vertex pair determined in the broad phase into the reference configuration. Due to the trilinear interpolation between composite element DOFs and hexahedral element DOFs, we can directly determine the position of the vertex with respect to the hexahedral element grid. The penetration depth is interpolated from a signed distance field. Since distance field voxels on both sides of a cutting surface are classified as inside, i.e., are assigned to a negative distance value, a simple interpolation of the penetration depth for a vertex position close to a cutting surface is not accurate (e.g., the distance value of a vertex directly on the cutting surface should be zero, but since all surrounding voxels carry a negative distance value, a non-zero value would be returned). We address this problem by specifically flipping the sign of distance values based on considering the connectivity between hexahedral elements.

5.4.1 Broad Phase Collision Detection

To identify for each vertex the potentially overlapping composite elements, we employ the spatial hashing approach proposed by Teschner et al. [THM+03]. The basic idea is to subdivide the 3D space using a uniform Cartesian grid. For each geometric primitive (vertices and composite elements), we determine the set of grid cells that are intersected by the primitive. If the sets of grid cells for two primitives are not disjunct, the pair of primitives is classified as potentially colliding and is further examined in the narrow phase. To efficiently represent the 3D grid in main memory, a hash table is employed.

Our broad phase collision detection consists of two passes. In the first pass, we traverse all composite elements. For each composite element, we construct an axis aligned bounding box in the deformed configuration. Since the surface mesh vertices are not strictly placed in the interior of the composite element, e.g., the mesh constructed using the sharp feature preserving dual contouring approach [JLSW02], we compute a bounding box which covers the composite element's eight vertices as well as all surface mesh vertices associated with this composite element (i.e., the vertices of the part of the surface mesh that belongs to the material part described by the composite element). The composite element's ID is stored in each grid cell that is intersected by the bounding box.

In the second pass, we traverse all surface vertices. For each vertex, we determine the grid cell that contains the vertex. The grid cell's list of composite elements ID generated in the first pass directly yields the composite elements that are potentially colliding with the considered vertex. Note that the composite element which the surface mesh vertex is associated with can be excluded from the list, since self-collisions of mesh parts associated with the same composite element cannot occur.

The parameters of spatial hashing significantly influence the performance of collision detection. We follow the optimized spatial hashing approach [THM+03] to assign these parameters. The spacing of the uniform Cartesian grid is chosen equal to the size of the composite elements in the reference configuration. To reduce hash collisions, we employ the XOR hash function to obtain a hash value $H(x, y, z)$ from a grid cell's index (x, y, z):

$$H(x, y, z) = (x\, p_1 \oplus y\, p_2 \oplus z\, p_3) \mod n, \tag{5.1}$$

where $\oplus$ is the XOR operator, and the prime numbers p_1, p_2, p_3 are 73856093, 19349663, 83492791. The size n of the hash table is chosen to be eight times the number of composite elements.

5.4.2 Narrow Phase Collision Detection

To further examine potentially colliding vertex/composite element pairs, we transform the vertex back into the reference configuration. The penetration depth and direction for this vertex are then computed using a topology-aware interpolation scheme based on a signed distance field, and are later utilized for collision response.

The transformation of the vertex is based on its trilinear interpolation weights with respect to the deformed composite element. These can be computed by solving the interpolation equation system using Newton iteration [Tod]. From the trilinear interpolation weights, we can immediately determine the location of the vertex in the reference configuration. All further processing in the narrow phase is performed in the reference configuration.

To determine the penetration depth and direction, we employ a signed distance field of the surface of the deformable body in the reference configuration. Each sample of the distance field consists of the signed scalar distance between the sample point and the nearest surface point, as well as the vector pointing from the sample to the nearest surface point (vector distance). The distance field is sampled at the grid cell centers of a uniform Cartesian grid, which is aligned with the hexahedral element grid (finest octree level). A negative (positive) distance value classifies the respective voxel as inside (outside). The spatial extent of the distance field must be chosen such that there is at least one layer of outside voxels. If in the following a more distant outside voxel is queried, any positive distance value can be returned (e.g., 1.0).

To determine whether the considered vertex penetrates the object, and at the same time obtain the penetration depth and direction for this vertex, a simple approach would be to sample the distance field by means of trilinear interpolation of the adjacent (surrounding) eight voxels, and to consider the vertex as penetrating, iff the obtained distance value is negative. However, this simple interpolation produces artifacts at cutting surfaces, where separated material parts are directly adjacent in the reference configuration. This situation is illustrated in Figure 5.3. Considering vertex v_a or v_b, all adjacent voxels have negative distance values, resulting in a negative interpolated distance value and thus in the (incorrect) classification of the vertex as penetrating. Note that the error can easily be larger than the size of a voxel/hexahedral element, as is the case for vertex v_b. The approach can be extended by determining the hexahedral element where the considered vertex is lying in. If this hexahedral element topologically does not belong to the composite element, the vertex is classified as non-penetrating. Otherwise, the distance field is trilinearly interpolated and the classification is performed based on the sign of the resulting distance value as before. This approach limits the error to half the size of a voxel/hexahedral element, but due to the clamping of the 'inside' region at boundaries of hexahedral elements, it leads to a jagged object surface as seen by the collision handling algorithm (see Figure 5.5). Further results obtained by the described algorithm—in the following referred to as 'simple interpolation approach'—are presented in the results section.

The problem with the simple interpolation approach is that the topology of the deformable object is not considered, i.e., the trilinear interpolation of the distance field incorporates voxels that are 'inside' with respect to some material connectivity component, but 'outside' with respect to the specific material connectivity component that is associated with the considered composite element. Our approach addresses this problem by first classifying the eight interpolation voxels as inside or outside with respect to the material connectivity component that is associated with the considered composite element, and then temporarily switches the sign of the distance value of 'outside' voxels, if this value is negative. This topology-aware interpolation approach virtually completely eliminates errors due to the geometrical non-conformity of the hexahedral grid and the triangle surface mesh, and leads to a smooth object surface as seen by the collision handling algorithm.

The individual steps of our algorithm are as follows:

1. Starting from the eight surrounding interpolation voxels, determine the subset of

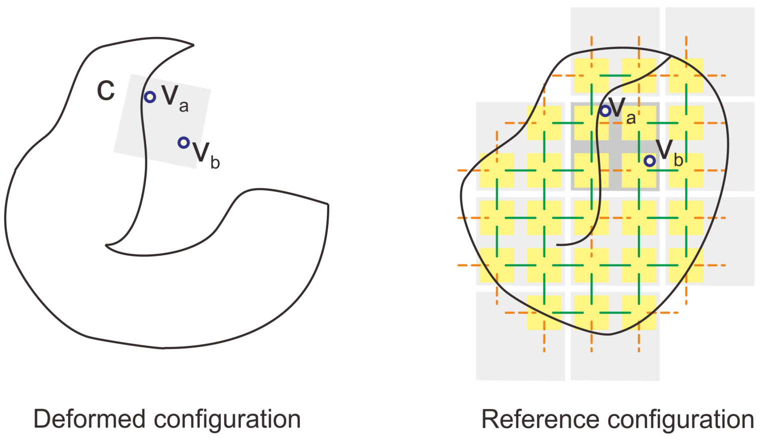


Figure 5.3: *Collision detection artifacts at cutting surfaces. In the deformed configuration (left), the vertices v_a and v_b are outside the object. Transformed back into the reference configuration (right), the vertices lie inside the object, although in a different material part with respect to the original query.*

those interpolation voxels for which the respective hexahedral element is existing and topologically belongs to the composite element.

2. Augment this subset by those interpolation voxels which are topologically connected to the interpolation voxels that are already contained in the subset.

3. The interpolation voxels in the subset are 'inside' voxels, the others are 'outside' voxels. Temporarily switch the sign of the distance value of 'outside' voxels to positive. Remark: The sign of the distance value of 'inside' voxels is negative, since each of these voxels corresponds to a hexahedral element (whose center is located inside of the deformable body). Distance field voxels outside of the simulation domain do not correspond to an hexahedral element, but already carry a positive distance value. Thus, no sign flipping is necessary for such voxels.

4. Trilinearly interpolate the distance value using the voxels' temporary distance values.

An example for the application of the algorithm is given in Figure 5.4. The effect of topology-aware interpolation is visualized in Figure 5.5. For a bunny model after cutting and deformation (left), we show the zero isocontour of the distance field returned by the simple interpolation approach (middle) and our topology-aware interpolation approach (right). Note that the zero isocontour corresponds to the object surface as seen by the collision handling algorithm. It is clearly visible that the topology-aware interpolation approach leads to a much smoother and much more precise isocontour.

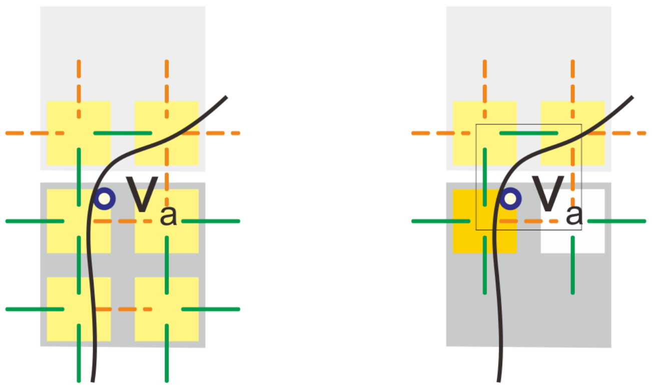


Figure 5.4: *Classification of voxels in our topological-aware interpolation approach. Left: We consider the lower composite element, more precisely, the left material part (note that there are two composite elements at the same location, indicated by the dark gray color). Right: From the four (in 2D) interpolation voxels of vertex v_a, one corresponds to a hexahedral element (orange) that topologically belongs to the considered composite element. Two further interpolation voxels correspond to hexahedral elements (yellow) that are topologically connected to the orange element. These three voxels are 'inside'. The remaining voxel (corresponding hexahedral element is shown in white color) is 'outside'—for this voxel, the sign of the distance value has to be switched to positive.*

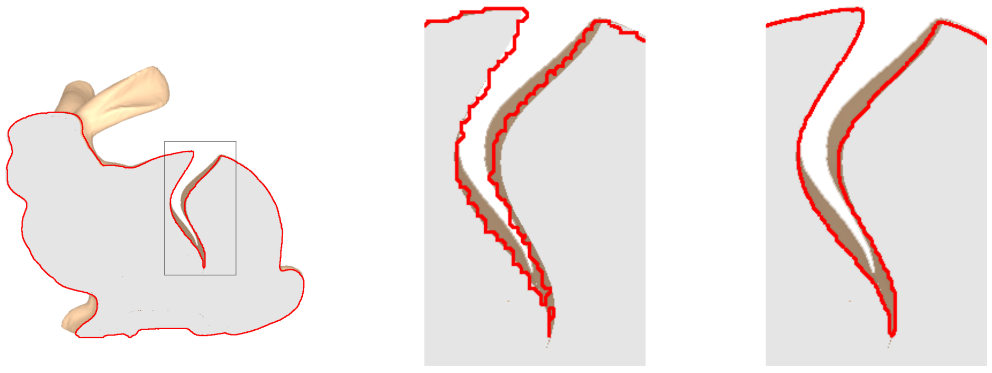


Figure 5.5: *Comparison between the zero isocontours of the distance fields obtained by the simple interpolation approach (middle) and our topology-aware interpolation approach (right).*

To determine the penetration depth and direction in the deformed configuration, the closest surface point indicated by the interpolated vector distance is transformed into the deformed configuration. The penetration depth and direction then are given by the vector between the queried vertex and its transformed closest surface point.

5.5 Distance Field Computation

After each cutting operation, the signed distance field is updated in a region growing manner [JBS06, Cui97], starting from those hexahedral elements which are incident to a link that has been cut. The update is performed as follows. When the link between two hexahedral elements has been disconnected by a cut, we assign new vector distances to the respective voxels, and add these voxels into a fifo queue. After all newly disconnected links are processed, we remove a voxel h_i from the list, and check its six face-adjacent voxels. The vector distance $d(h_j)$ of a neighbor voxel h_j is updated to $d' = \overrightarrow{h_j h_i} + d(h_i)$, if $\|d'\|_2$ is smaller than $\|d(h_j)\|_2$. If the vector distance of the neighbor voxel has been updated and the neighbor voxel is not already contained in the list, it is added to the list.

Note that to obtain an absolutely accurate distance field, a priority queue would have to be used instead of a fifo queue. This, however, would increase the complexity of updating. For a reasonable collision response, an approximately monotone distance field is sufficient to provide fully reasonable results, as has for example been demonstrated in [GSM+12].

Sub-voxel accurate boundary values. Sub-voxel accurate distance values for the boundary voxels improve the accuracy of the distance field over a binary classification [SJ01]. The exact distance value for a boundary voxels can be obtained by determining the distance from the voxel's center to the triangle surface mesh. In our approach, the exact intersection point of a link and the cutting blade as well as the blade's normal is stored when a link is cut. To obtain a sub-voxel accurate distance value for a boundary voxel, we consider the disconnected links that are incident to the corresponding hexahedral element, and compute the minimum distance of the voxel center to the planes spanned by link/blade intersection points and blade normals.

Accuracy of the deformed distance field. Figure 5.6 shows a comparison between an exact distance field, directly computed in the deformed configuration, and an approximate distance field, computed in the reference configuration and transformed into the deformed configuration. The solid green lines represent the isocontours of the exact distance field, whereas the dashed red lines are the isocontours of the approximate distance field. The approximate distance field decreases monotonically from the interior towards the boundary, and it converges to the exact one as the surface of the object is approached. Recomputing the distance field in the deformed configuration would take minutes, even when accelerated by means of a hierarchical structure [LGLM99], whereas locally updating the precomputed distance field in the reference configuration

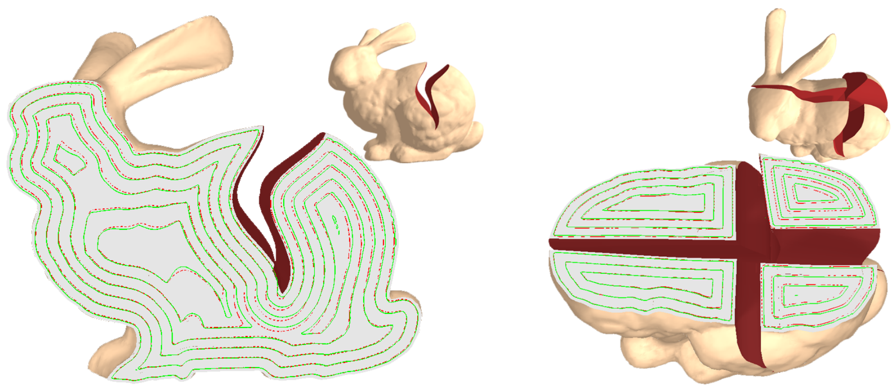


Figure 5.6: *Isocontours of the distance field computed directly in the deformed configuration (solid green), and of the distance field computed in the reference configuration and transformed into the deformed configuration (dashed red).*

and transforming distance values into the deformed configuration can be performed in a few milliseconds.

5.6 Results

In the following, we analyze the potential of our proposed collision detection approach for interactive applications like virtual surgery simulations. Our experiments were run on a standard desktop PC equipped with an Intel Xeon X5560 processor running at 2.80 GHz (a single core is used), 8 GB of RAM, and an NVIDIA GeForce GTX 480 graphics card.

Examples. In the first example, we focus on the collision detection between newly created surfaces after cutting. Figure 5.1 (left) shows a deformable model consisting of hexahedral elements at a resolution of $101 \times 78 \times 100$, and composite elements (yellow grid) that are three levels coarser than the hexahedral elements. In the middle, the bunny, with its bottom fixed to the ground, is cut into two parts by a curved surface, resulting in the upper part sliding down along the cutting surface. On the right, four stamp-like cuts are applied to the bunny. The resulting cylindrical shapes slide out of the bunny's body, under the tight constraints of the holes. Contact forces are computed using a penalty force model, and together with their derivatives are fed into the time-implicit CFE elasticity simulation, which is based on an efficient geometric multigrid solver [DGW11a].

Figure 5.7 demonstrates our collision detection applied to a cylinder shell model, the top of which is fixed. After cutting, the parts fall down, and stack on the ground.

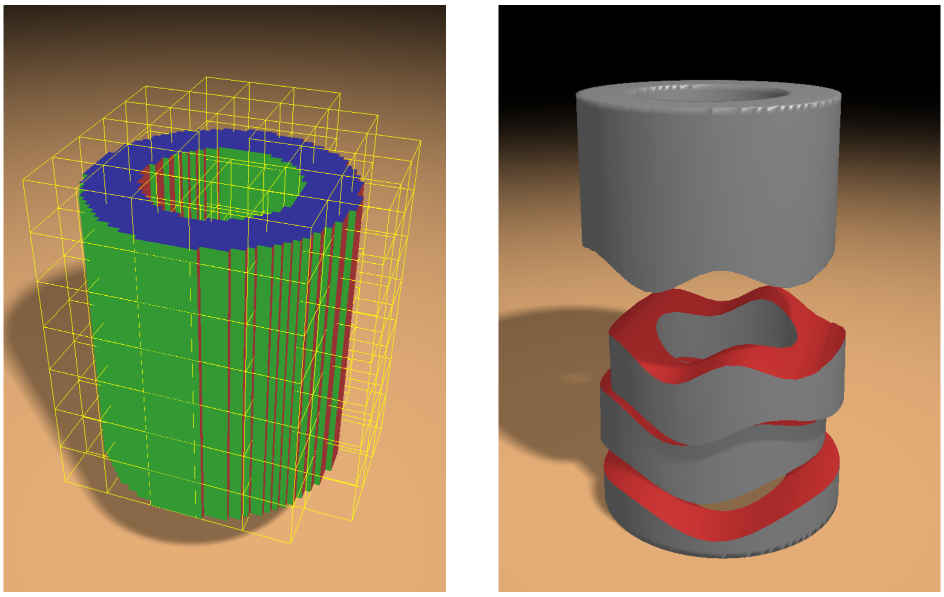


Figure 5.7: *Cutting of a cylinder shell model. Left: High-resolution hexahedral elements and coarse composite elements. Right: After cutting, the parts fall down and stack. Collision detection and finite element simulation are performed at 136 fps, while the per cut updating is finished in 17 ms.*

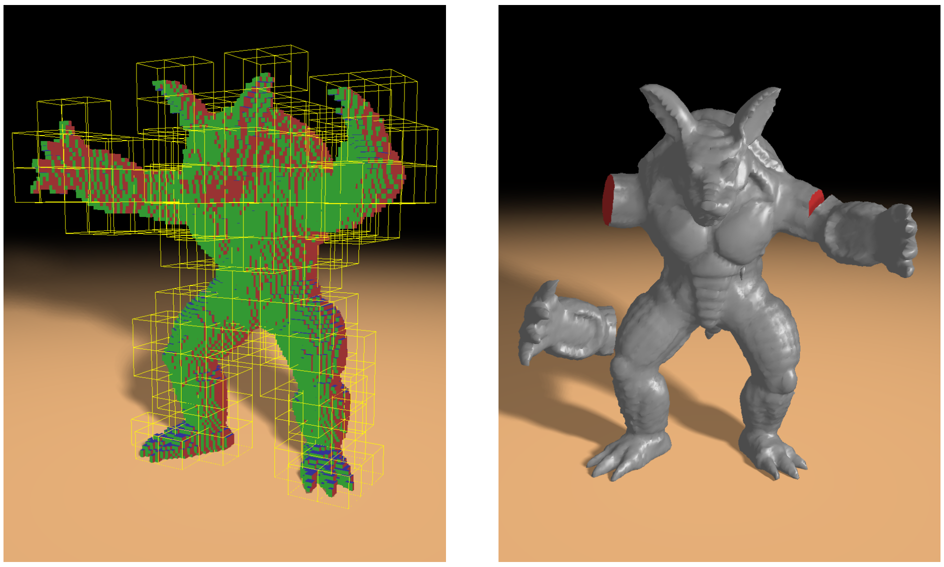


Figure 5.8: *Cutting of the Stanford Armadillo model. Left: High resolution hexahedral elements and coarse composite elements. Right: After cutting, the parts fall down. Collision detection and finite element simulation are performed at 30 fps, while the per cut updating is finished in 32 ms.*

Figure 5.8 shows haptic cutting on the Stanford Armadillo model. For a real-time recording of the dynamic effects, we would like to refer the reader to the video accompanied with [WDW13].

Evaluation of accuracy. To evaluate the sub-voxel accurate collision detection, we analyze the change of the kinematic energy of the simulation objects over time, which is a measure for the smoothness of simulation. Jumps in the kinetic energy can easily lead to visual artifacts and numerical instabilities. Figure 5.9 compares the change of kinetic energy of a bunny model using the simple distance field interpolation approach with our topology-aware interpolation approach. It can be observed that the change of kinematic energy for the topology-aware interpolation is much smoother than for the simple interpolation. Figure 5.10 shows the simulation of a cut in a zero-gravity environment. Since the simple interpolation approach leads to an error in penetration depth computation, the cut opens. In contrast, using our topology-aware interpolation approach, the cut remains closed.

Performance. Table 5.1 shows detailed timing statistics for different models and different resolutions of the hexahedral element grid. The second group of columns gives information about the employed models: the resolution of the hexahedral element grid, the number of simulation DOFs (using a composition level of three), and the number of surface triangles. The third group lists timings for mesh updating (adaptive octree refinements along the cutting surface and creation of the surface mesh), updating of the FEM stiffness matrices, updating of the distance field for collision detection, and finally the total time required for update operations. Note that the time required for updating is dependent on the spatial extent of a cut. In our tests, the cuts were realized such that the number of newly created hexahedral elements (due to adaptive octree refinements) in average was about 2% to 4% of the total number of elements.

The last group gives timings for collision detecting and FEM simulation. The numbers in parentheses indicate the performance gain resulting from using composite elements, compared to performing collision detection and finite element simulation on the underlying hexahedral elements. With no composition, the broad phase takes about 86% of the total time for collision detection. As the composition level l increases, the narrow phase is becoming the bottleneck. To increase the overall performance, in the narrow phase we test every 2^lth vertex rather than all vertices. It is worth noting that the resulting speedup factor in the narrow phase (8 when $l = 3$) is far below the overall speedup factor achieved for collision detection (between 21.3 and 58.4).

Our statistics indicate that the composition not only leads to a significantly faster deformation simulation, but also to a highly efficient collision detection. With three

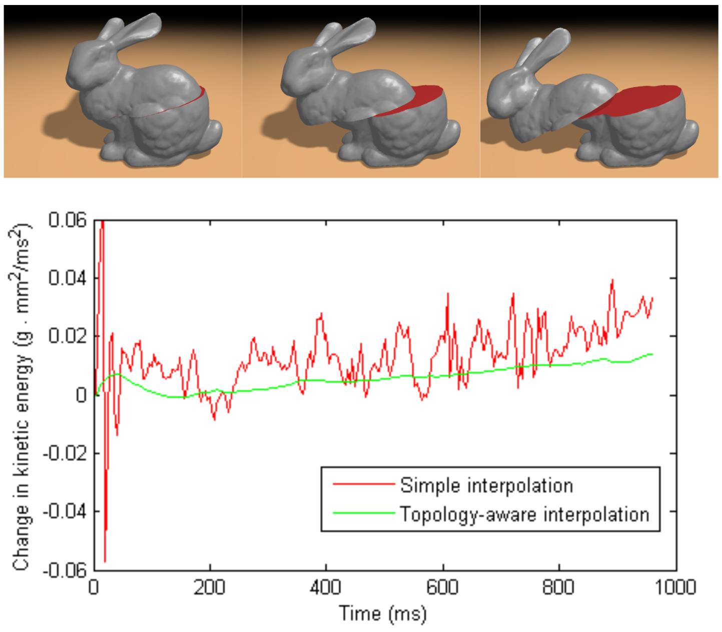


Figure 5.9: *Analysis of the change of kinetic energy over time. Top: After cutting the bunny, the upper part slides down. Bottom: With our topology-aware interpolation, the change of kinetic energy (green) is significantly smoother than when the simple interpolation is used (red).*

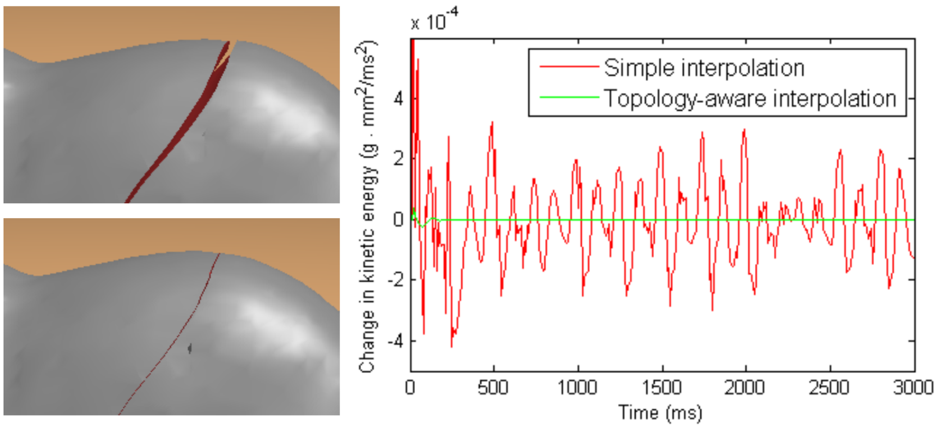


Figure 5.10: *Cutting in a zero-gravity environment. Left top: Using the simple interpolation, the cut opens since the collision detection is not accurate. Left bottom: With out topology-aware interpolation, the width of the cut remains zero. Right: Change of kinetic energy over time.*

	Hexahedral	# Sim.	# Tris	Per Cut Updating [ms]				Simulation [ms] (Speedup)		
Model	Elements	DOFs	× 1k	Mesh	Stiffness	Distance	Total	Col. Det.	FEM	Total
Bunny	101×78×100	3,669	69	8.9	25.6	2.4	36.9	4.6 (43.5×)	19.6 (253.6×)	24.2 (213.2×)
Bunny	51×39×50	933	17	3.1	5.8	0.4	9.3	1.5 (26.8×)	3.7 (93.2×)	5.2 (67.4×)
Liver	82×83×100	2,928	59	11.7	24.4	1.9	38.0	3.2 (58.4×)	23.3 (103.7×)	26.5 (98.2×)
Liver	41×42×51	717	13	2.8	5.6	0.3	8.7	1.2 (21.3×)	4.3 (67.2×)	5.5 (57.0×)
Cylinder	42×42×50	1,338	30	5.7	11.3	0.2	17.2	2.1 (32.5×)	5.2 (118.4×)	7.3 (93.3×)
Armadillo	85×77×100	2,346	50	9.3	22.6	0.4	32.3	3.7 (26.1×)	29.7 (31.0×)	33.4 (30.5×)

Table 5.1: *Performance statistics for different models and different resolutions of the hexahedral element grid. All models are simulated with a composition level of three.*

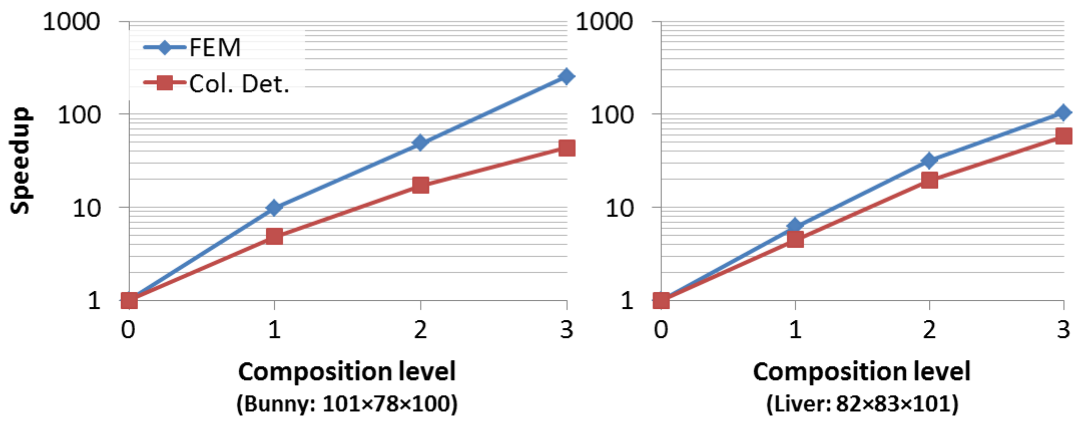


Figure 5.11: *Speedup with respect to different composition levels.*

levels composition, the speedup of the overall performance can reach about two orders of magnitude. Figure 5.11 shows the speedups for different composition levels. Exponential growth of the speedup factor can be observed.

5.7 Conclusion

We have presented an efficient collision detection method for CFE simulation of cuts in deformable bodies. By exploiting the composition structure, we achieve a speedup factor which scales exponentially with respect to the level of composition. By analyzing the topology of underlying hexahedral elements, our approach accurately interpolates the penetration depth for non-conforming geometries. These advances make CFEs an ideal candidate for applications where interactive simulation update rates are required.

In the future, we plan to work on the physically-based modeling of the fracture mechanics between a scalpel and soft tissues, in order to provide a fully realistic haptic feedback for surgical training applications. Another direction of research is constraint-based contact handling for deformable bodies. Here, it would be interesting to investigate the potential of CFEs to accelerate this computationally intensive approach.

Chapter 6

Interactive Residual Stress Modeling for Soft Tissue Simulation

Residual stress is the stress which remains in a deformable body in the absence of external forces. Due to the release of residual stress after cutting, soft tissues will shrink and the wound will open. Thus, to realistically simulate soft tissue deformations due to cutting, a model for the residual stress in a patient body is needed. In this chapter we present an interactive method to compute a physically meaningful patient-specific residual stress distribution. With our method, by using their experience doctors can sketch directional stress strokes and specify stress magnitudes at a few control points on the body surface. The residual stress is then immediately computed from these inputs and visualized by displaying the deformations of a set of control cuts on the body. In a visually guided session, the user can further modify the initial strokes and magnitudes until a satisfactory result is obtained. We demonstrate the potential of the proposed method for virtual cut simulation by showing the variations of wound openings depending on the residual stress distribution, and by comparing the simulation of a flap surgery with the real flap surgery.

6.1 Introduction

The input models used in computer-aided surgery simulation are typically generated from patient-specific scans such as CT or MRI. In these scans, the internal stress distribution of the measured tissue is lost. However, even in the absence of external forces

This chapter is partially based on material that has been originally published in J. Wu, K. Bürger, R. Westermann, and C. Dick, *Interactive residual stress modeling for soft tissue simulation*, Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine, 2012, pp. 81–89.

soft tissue is not stress free, but it is tensioned by the so-called *residual stress*. The residual stress is a non-negligible factor in soft tissue dynamics [Hum03], and its effect is especially apparent when soft tissue is cut. After cutting, the tissue will shrink and the wound will open due to the release of residual stress [CF86]. Moreover, the shrinkage of a flap due to the relaxation of residual stress affects the quality of surgeries, e.g., a breast reconstruction surgery, in which the volume of the flap is of crucial importance [KZPB04]. Therefore, to produce physically realistic cutting simulations for soft tissue, patient-specific models of the residual stress need to be derived and included in the simulation.

However, despite the importance of residual stress for soft tissue dynamics, it is not well addressed in biomedical simulation. This is due to the fact that the residual stress distribution is patient-specific, but no procedure yet exists to derive this distribution directly from non-invasive tissue measurements. The residual stress is valued by a second order tensor which can be decomposed into principal directions (eigenvectors) and principal stresses (eigenvalues). The values of principal stresses vary strongly between different individuals, e.g., tight or loose. Moreover, the principal directions are geometry dependent and vary spatially. Even though models for these distributions in the human body exist, these models cannot easily be parameterized to a particular body.

In this chapter, we present an interactive method to derive a physically meaningful residual stress distribution for a given patient model, and we use this distribution for estimating the opening of cuts in soft tissue (see Figure 6.1). The stress model is interactively designed by a domain expert based on her expertise about the residual stress distribution in human bodies of different shape and consistence. The design process is subject to additional constraints, which enforces that the resulting model respects certain physical laws. The modeling result is visualized immediately, and thus allows interactive control over the residual stress distribution.

The specific contributions of this chapter are:

- *A procedure to interactively model the residual stress distribution in patient specific models.* Following the well-known concept of Langer's lines [Lan78] in anatomy, the user draws several strokes on the surface of soft tissues, thus specifying a coarse skeleton of the distribution of the residual stress direction-field. In addition, the stress magnitudes are specified at a sparse set of control points, enabling the expert to adapt the stress distribution to the patient's tissue consistency. Direction and magnitude parameters are propagated to the whole body via constraint interpolation.

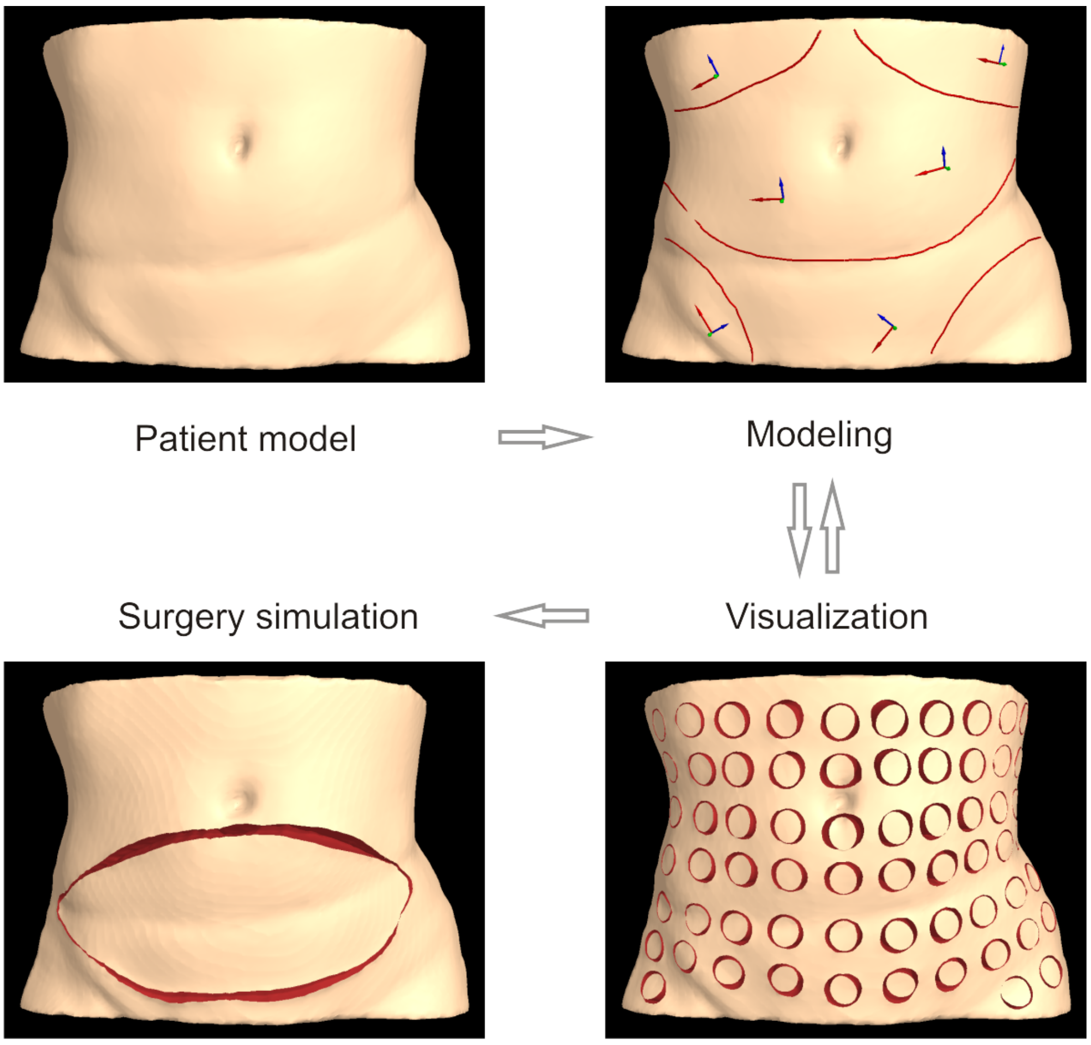


Figure 6.1: *Overview of the residual stress modeling procedure. First, a patient specific volumetric CT scan is used as input model. Next, an experienced surgeon designs a residual stress tensor field by drawing stress directions via strokes on the model surface, and by specifying the stress magnitudes at several sparse locations. A global tensor field is constructed from this information via Laplacian interpolation, while the equilibrium equations serve as a penalty term. The tensor field is immediately visualized by simulating the effects of making many small holes on the surface. Based on the feedback, the surgeon can modify her design until a satisfied tensor field is computed. Finally, this residual stress is used in a biomedical simulation of the opening of wounds due to cuts.*

- *A novel stress tensor visualization method to assist the modeling process.* The visualization method is in spirit similar to how the anisotropic behavior of skin stress is observed. By making many small round incisions on the soft tissues, and simulating the deformation after these topological changes, it is easy to perceive the principle direction of residual stress and to predicate the effect of surgical operations.

- *A demonstration of the potential of the proposed method in cutting simulations.* In a number of preliminary tests we show that the residual stress distributions generated by the semi-automatic modeling process yield very realistic wound openings in virtual cutting simulations. The possibility to interactively adjust the distributions taking into account the observed tissue consistency (e.g., tight or loose) enables cutting simulations that mimic the real-world tissue behavior quite realistically.

The remainder of this chapter is organized as follows: In Section 6.2, we review related work in biomedical engineering and visualization. In Section 6.3, we discuss the properties of residual stress from both a mechanical and a biological perspective, and we lay open the foundation on which our approach is based. In Section 6.4 and Section 6.5, we introduce the modeling procedure and the visualization approach, respectively. Implementation details are given in Section 6.6. We conclude the chapter with some results and ideas for further research in this field.

6.2 Related Work

The accurate measurement of residual stress is still an open problem, due to the fact that the mechanical effect of the residual stress is highly coupled with the nonlinear, heterogeneous, anisotropic behavior of biological tissues. Nevertheless, both *in vivo* and *ex vivo* experiments have been performed to investigate the residual stress in soft tissues such as the skin [JJKG08, FTN11]. These experiments typically capture only the residual stress of a small specimen. However, a consistent residual stress field defined on the whole soft tissue is needed for the purpose of medical simulation. In this chapter, we try to obtain a meaningful stress field by a constructive method based on the parameters modeled by domain experts.

Tensor field modeling. Zhang et al. [ZHT07] presented a topology based approach to design tensor fields on surfaces. An extension of this approach to 3D tensor fields is not trivial, since the topology of 3D tensor fields is much more complicated. Recently, Huang et al. [HTWB11] proposed a method to construct smooth 3D cross-frame fields based on a spherical harmonics representation of frames. The three axes of a cross-frame has no order, and thus the decomposition of a cross-frame field into three consistent eigenvector fields is not straightforward. Arsigny et al. [ACPA06] proposed Log-Euclidean metrics for calculus on diffusion tensors. Our stress tensors differ from the diffusion tensors in that the stress tensor is not positive-definite. Meanwhile, the expensive evaluation of logarithms also makes this sophisticated approach not suitable for interactive modeling. Takayama et al. [TIHN07] proposed a sketch based interface to model myocardial fiber orientation, and extended it to model tensor fields for solid texturing [TOII08]. Also for solid texture synthesis, Zhang et al. [ZDL+11] proposed to use quaternions to represent tensors. Our problem differs from this as the tensor field of residual stress should satisfy some equilibrium equations.

Tensor field visualization. Various tensor field visualization approaches for different applications have been proposed based on glyphs, line/surface tracing, line integral convolution, direct volume rendering, topological features [HLL97], physical effects of the tensor field on the underlying media [ZP02], or a combination of multiple of above techniques [WLY04, DGBW09]. Our visualization is close to the volume deformation based approach [ZP02]. Moreover, we perform topological operations to make the deformation more meaningful for surgery planning. It can be seen as a reproduction of Langer's experiments [Lan78] from which he observed the anisotropic behavior of skins.

6.3 Modeling Foundations

In this section, we shortly review the theoretical foundations of our modeling method. We first specify the equilibrium equations which the residual stress tensor field must satisfy due to the underlying physical principles. We then address the characteristics of residual stresses in the medical context of soft tissue simulation.

6.3.1 Mechanics of Residual Stress

We consider a deformable body with reference configuration $\Omega \subset \mathbb{R}^3$. The 3D second-order residual stress tensor field σ must satisfy the following equilibrium equations as given by Hoger [Hog86]:

$$-\operatorname{div}\sigma = f_{b,0} \equiv 0 \qquad \text{in} \quad \Omega \setminus \partial\Omega, \tag{6.1}$$

$$\sigma\, n_\Gamma = f_{s,0} \equiv 0 \qquad \text{on} \quad \Gamma_N, \tag{6.2}$$

$$\sigma^T = \sigma \tag{6.3}$$

Here, n_Γ is the unit outward normal on $\Gamma = \partial\Omega$. $\Gamma_N \subset \Gamma$ denotes the free surface of the deformable body. $f_{b,0}$ and $f_{s,0}$ denote the residual body forces and surface forces, which are applied in the undeformed state. These forces are zero in our application.

Equations (6.1) and (6.2) correspond to the fact that the residual stress and the residual body and surface forces must be balanced in the undeformed state. In the absence of external forces, these equations mean that the residual stress is self-balanced at each material point, and that on the surface the residual stress perpendicular to the surface must be zero. By integrating these equations over the body's domain and surface, it can then be derived that the mean residual stress within the body is zero [Hog86]. This implies that a non-zero uniform residual stress tensor field cannot exist.

Equation (6.3) specifies that the stress tensor is symmetric. According to the spectral decomposition theorem, a 3D second-order, symmetric tensor can be decomposed into three mutually orthonormal eigenvectors n_i and three eigenvalues λ_i according to

$$\sigma = \sum\nolimits_{i=1}^{3} \lambda_i n_i \otimes n_i, \tag{6.4}$$

where $\otimes$ denotes the tensor product operator. A positive eigenvalue means that the body is tensioned along its corresponding eigenvector direction, whereas a negative eigenvalue means that the body is compressed along this direction. Note that the directions of the eigenvectors are not unique—if v is an eigenvector, then $-v$ also is an eigenvector. Therefore, when we refer to the direction of an eigenvector v, we rather refer to the (non-oriented) direction of the line $\lambda v, \lambda \in \mathbb{R}$.

6.3.2 Residual Stress in Soft Tissues

The decomposition of the stress tensor into eigenvectors and eigenvalues is meaningful in anatomy and physiology. A well-known concept in plastic and reconstructive surgery are *cleavage lines*, which were first described in 1861 by anatomist Karl Langer [Lan61], and thus are also named as *Langer's lines*. Langer's work was initially written in German language, and later translated into English by Gibson [Lan78]. One of the original illustrations by Langer is shown in Figure 6.2. It depicts directions

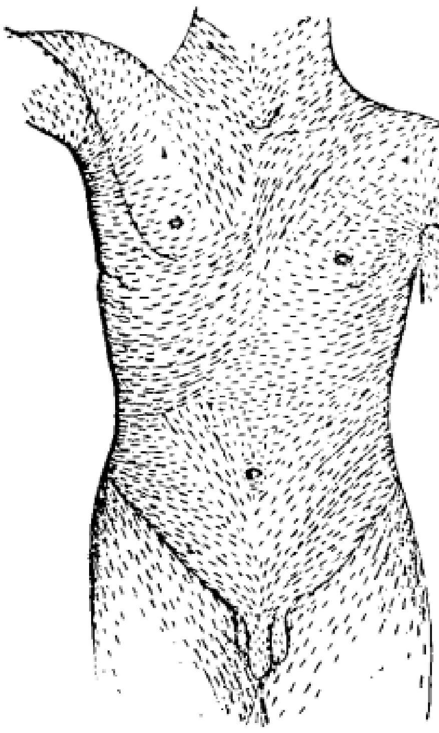


Figure 6.2: *Directions of highest tension within the human skin (illustration by anatomist Langer [Lan61], 1861).*

within the human skin along which the skin has the highest tension—mathematically, this direction at each point is the direction of the eigenvector corresponding to the largest eigenvalue. A surgical cut is usually carried out parallel to Langer's lines, since parallel incisions generally heal better and produce less scarring than those which cut across.

Biologically, Langer's lines correspond to the natural orientation of collagen fibers within the dermis. In general, the residual stresses in soft tissues are highly related to the growth of organs [GO06]. According to the model of kinematic growth [RHM94], residual stresses are developed during the growth of soft tissue in the following way: The original body is fictitiously decomposed into many small stress-free pieces, each of which grows independently. Because the growths in each piece need not be compatible, internal forces are needed to assemble the pieces into a consistent configuration. These internal forces induce residual stresses. Since the growth of soft tissues is slow and gradual, it can be assumed that the residual stresses vary smoothly throughout the tissue.

6.4 Modeling Procedure

Based on the spectral decomposition of the stress tensor, we construct the stress tensor field by modeling eigenvector and eigenvalue fields. Assuming that the residual stress field in biological tissues is smooth, we propose the following approach: The user specifies eigenvectors and eigenvalues at a few, sparsely distributed locations on the surface of the object. These values are smoothly propagated over the entire domain of the body by using Laplacian interpolation, with the equilibrium equations serving as a regularization term. After the residual stress tensor field has been computed, the effect of this field on the underlying media (augmented by a set of cuts) is immediately visualized (see Section 6.5). Following a computational steering approach, the domain expert can then further refine the eigenvector and eigenvalue fields, until the desired result is obtained (see Figure 6.1 for an illustration of this process).

In the modeling process, we first specify the eigenvector fields, followed by the eigenvalue fields. Since the three eigenvectors at each point are mutually orthogonal, it is sufficient to specify two eigenvectors—the third eigenvector can be obtained by computing their cross product. Moreover, the normal on the surface is an eigenvector of the stress tensor (since $\sigma n = 0$ on Γ_N, i.e., n is eigenvector for the eigenvalue 0). Therefore, only one eigenvector field must be specified by the user, corresponding to Langer's lines.

To specify this field for a patient specific geometry model, the user draws a set of strokes on the surface of the body. From these strokes, a smooth vector field in the entire body is constructed and visualized. As the user incrementally draws more strokes, the vector field is further refined. An example of user input strokes and the automatically constructed eigenvector field is shown in Figure 6.3.

The eigenvalue fields are specified in a similar way. Since the eigenvalue corresponding to the surface normal is always zero, up to two eigenvalues can be specified at a point. Considering the user interface, after the user has selected a location on the surface, we show the local tensor frame with colored arrows indicating the direction of surface normal, the Langer's line, and the third vector resulting from their cross product. The length of the arrow corresponds to the magnitude of the eigenvalue as it is specified by the user (see Figure 6.4).

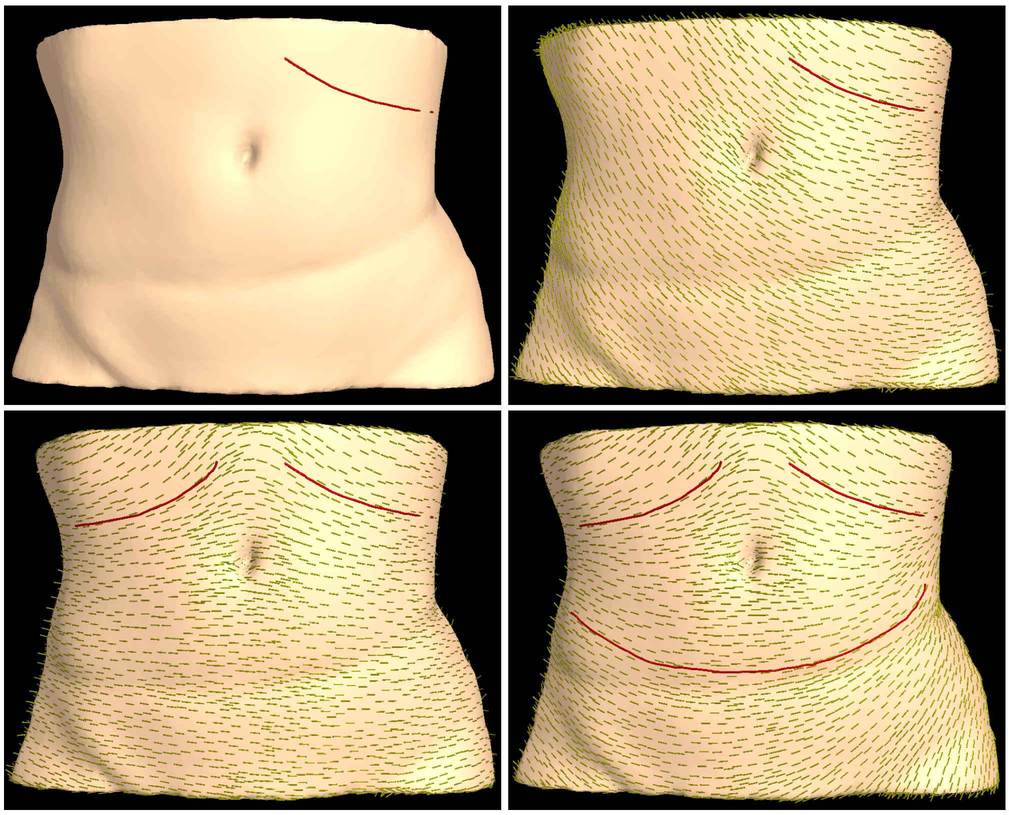


Figure 6.3: *Modeling the eigenvector field by sketching on the surface. The vector field (yellow arrows) is created after the user draws the first stroke (red line), and updated every time after the user draws a new stroke.*

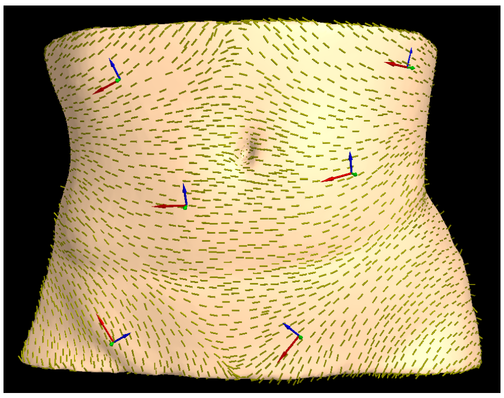


Figure 6.4: *Modeling the eigenvalue fields by adjusting frame-arrow lengths. The local tensor frame (colored arrows) of a user-selected location shows the specified eigenvalue along each direction.*

6.5 Visualization

The residual stress tensor field could be visualized by showing streamlines that are traced in the eigenvector fields and by coloring coding the eigenvalue fields (see Figure 6.5). However, from such a visualization it is rather difficult for the surgeon to interpret the physical behavior of the material after cutting, which depends on not only the residual stresses, but also the material properties, and the depth and the length of an incision. Therefore, for our particular application, we propose a visualization method which directly shows this behavior.

Our method is based on introducing a set of small, round incisions distributed over the surface of the body. Then, the deformation of the body due to the residual stress is computed (see Figure 6.6). Typically, due to the relaxation of residual stress after cutting, the body will shrink, and the wounds will open. If the residual stress is anisotropic, both the outer and inner wound margins become elliptical in shape with their axes mutually orthogonal (see Figure 6.7). The major axis of the outer ellipse indicates the eigenvector direction for the major eigenvalue. On the surface, it corresponds to Langer's lines. The gap between the outer and inner boundary of the wound indicates the magnitude of the stress.

This visualization approach is similar to how Langer observed the anisotropic phenomenon of the residual stress [Lan78]. Langer marked circular outlines at short distances from each other on the skin of cadavers, and then incised around the circles. From the resulting deformation he observed patterns and determined *line directions* by the longer axis of the ellipses.

We provide two coloring approaches of the new cutting surface to indicate some important information, i.e., the depth of cuts and the stress in the direction normal to the cutting surface (Figure 6.6 (middle and right)). The depth field is computed in a way similar to Takayama et al. [TOII08]. We define the skin layer of the abdomen wall as having depth 0 (red color), and the inner layer as having depth 1 (yellow color). Radial based function interpolation is then used to interpolate the depth values.

To better visualize the anisotropic behavior of the deformation, we color the cutting surface by the norm of the normal component of the residual stress, $||\sigma n||_2$, where n represents the normal of the newly created cutting surface, with a linear color map from red for the highest tension to blue for the lowest. These colors are consistent with the color of arrows for eigenvalue editing. As we can see from Figure 6.6 (right), the red area means higher released tension, while blue area means lower tension. The axis passing through red poles indicates the Langer's line.

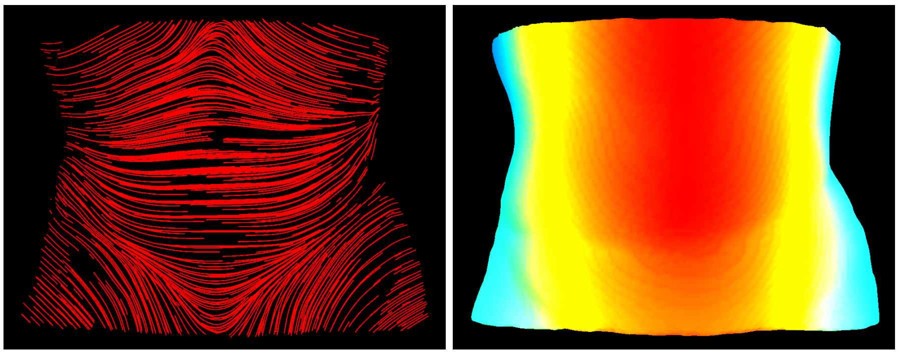


Figure 6.5: *Left: A streamline visualization of the interpolated 3D volumetric eigenvector field. Right: Visualization of a eigenvalue field by color coding.*

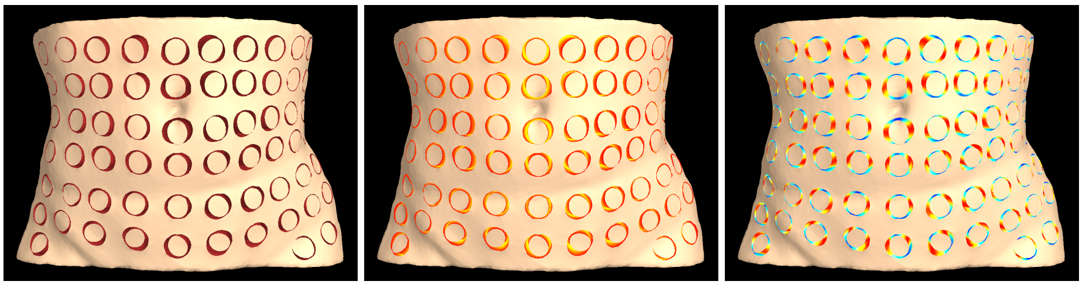


Figure 6.6: *Left: A visualization of the residual stress tensor field by simulating the effect of many small round incisions. The long axis of the outer ellipse corresponds to the major principle direction, while the width of the wound gives a direct visual impression how the tissue behaves when being cut. Middle: The resulting cutting surfaces are colored by the depth. Right: The cutting surfaces are colored by the magnitude of the residual stress component in the direction of the cutting surface normal.*

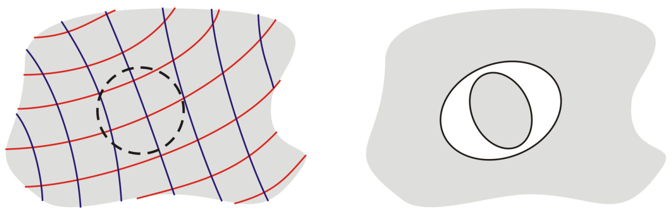


Figure 6.7: *Left: An eigenvector field (red and blue curves) of the residual stress, the eigenvalues along the red curves are larger than those along the blue ones. Right: After a round incision (the dashed circle), the wound gapes.*

6.6 Implementation

In this section, we introduce the interpolation of residual stress tensors (i.e., eigenvector fields and eigenvalues fields), and the dynamic simulation of deformable bodies with residual stresses. The eigenvector fields are interpolated by Laplacian smoothing, while the eigenvalue fields are interpolated by combining the Laplacian equation and the static equilibrium equation in a least square sense. It is arguable that an interpolation approach which treats the whole tensor as a unity may produce more reasonable results. However, since the tensor is preferably represented by eigenvectors and eigenvalues for intuitive modeling inputs, and they are of different units and scales, the resulting formulation of optimization problem in that case is ill conditioned and its convergence is poor. Thus, we choose to generate the tensor field by interpolating eigenvectors first and then eigenvalues.

Our interpolation and simulation are based on a hexahedral finite element discretization using a uniform Cartesian grid, following the approach proposed in [DGW11a] and [WDW11]. The topology of the finite element model is represented by using a linked volume, where face-adjacent elements are connected via links, and these links are marked as disconnected when the respective elements are separated by a cut. A visually smooth render surface that is topologically consistent with the hexahedral finite element model is constructed directly from this model by using a dual contouring approach. For details, we refer the reader to the original works.

For the modeling of the residual stress tensor field, the user paints eigenvectors and eigenvalues onto the smooth render surface. The respective values are propagated to the closest hexahedral finite elements. The computation of the residual stress tensor field is then performed on this grid.

6.6.1 Static Equilibrium Equations

For the discretization of the continuous equilibrium equations, we employ a hexahedral finite element discretization using piecewise constant interpolation for the residual stress tensor field (i.e., we store one stress tensor per element) and trilinear interpolation for the test functions. This leads to a linear system of equations

$$T\overline{\sigma} = 0. \tag{6.5}$$

Here, $\overline{\sigma}$ is a linearization of the per-element stress tensors, each of which is represented as a 6-component vector $\overline{\sigma}^e = (\sigma^e_{11}, \sigma^e_{22}, \sigma^e_{33}, \sigma^e_{12}, \sigma^e_{13}, \sigma^e_{23})^T$ using Voigt notation. T is an $n_v \times n_e$ matrix which maps per-element stress tensors to per-vertex forces, where n_e and n_v denote the number of finite elements and vertices, and each matrix entry is itself a matrix consisting of 3×6 scalars. T is assembled from the per-element matrices

$$T^e = \int_{\Omega^e} (B^e)^T dx,$$

by considering the sharing of vertices between adjacent finite elements. Here, $B^e(x)$ is the element strain matrix, and Ω^e is the domain of the finite element. Note that the matrix T depends only on the geometry and topology of the deformable body; it does *not* depend on the material properties such as Young's modulus or Poisson's ratio.

For eigenvalue interpolation, we employ a formulation of Equation (6.5) in terms of eigenvalues. According to Equation (6.4), it is given by

$$T \frac{d\overline{\sigma}}{d\lambda} \lambda = 0, \tag{6.6}$$

where λ is a linearization of the per-element stress tensors' eigenvalues, and $\frac{d\overline{\sigma}}{d\lambda}$ is assembled from the per-element third-order tensors $\frac{d\sigma^e}{d\lambda^e}$ given by

$$\left(\frac{d\sigma^e}{d\lambda^e}\right)_{ijk} = (n_k^e \otimes n_k^e)_{ij}.$$

6.6.2 Interpolation of Eigenvalues

In the following, we introduce our interpolation scheme for the case of eigenvalue interpolation. The interpolation of the eigenvectors, which is algorithmically performed *prior* the interpolation of the eigenvalues, is performed in a similar way and is described in the following section.

For the formulation of our interpolation scheme, we consider a vector-valued function $\phi(i) = \lambda^i = (\lambda_1^i, \lambda_2^i, \lambda_3^i)^T$, which specifies the three eigenvalues for each finite element $i = 1, ..., n_e$.

Laplacian operator. We use Laplacian smoothing [SCOL⁺04] to propagate the prescribed values over the entire domain of the body. The discrete Laplacian operator $\triangle$ acting on ϕ is defined as

$$(\triangle\phi)(i) = \sum_{j \in N_1(i)} \omega(i, j) \left[\phi(j) - \phi(i)\right],$$

where $N_1(i)$ is the set of elements that are face-adjacent to element i. Since we use a uniform discretization, the weight $\omega(i, j)$ is chosen as $\omega(i, j) = \frac{1}{|N_1(i)|}$. To get a smooth tensor field, we seek for

$$(\triangle\phi)(i) = 0, \qquad i = 1, \dots, n_e.$$

In matrix form, this is written as

$$L\lambda = 0, \tag{6.7}$$

where $\lambda = ((\lambda^1)^T, \dots, (\lambda^n)^T)^T$, and L is an $n_e \times n_e$ matrix defined by

$$L_{ij} = \begin{cases} -I_{3\times3} & \text{if } i = j, \\ \omega(i, j)I_{3\times3} & \text{if } j \in N_1(i), \\ 0 & \text{otherwise.} \end{cases}$$

Boundary condition. The eigenvalues for some elements are prescribed by the user. This is modeled by

$$\phi(i_k) = \lambda^{i_k,0}, \qquad k = 1, \dots, n_p,$$

meaning that for element i_k the eigenvalues $\lambda^{i_k,0}$ are prescribed. n_p denotes the number of these elements. The matrix form of these constraints is given by

$$C\lambda = \lambda^0, \tag{6.8}$$

where $\lambda^0 = ((\lambda^{1,0})^T, \dots, (\lambda^{n_p,0})^T)^T$, and C is an $n_p \times n_e$ matrix defined by

$$C_{k\ell} = \begin{cases} I_{3\times3} & \text{if } \ell = i_k, \\ 0 & \text{otherwise.} \end{cases}$$

We solve the linear systems given by the Laplacian operator Equation (6.7), the boundary constraint Equation (6.8), and the equilibrium constraint Equation (6.6) in a least squares sense, i.e.,

$$\arg\min_{\lambda} \|A\lambda - b\|_2^2, \tag{6.9}$$

where $A = \begin{pmatrix} L & C & T\frac{d\sigma}{d\lambda} \end{pmatrix}^T$, and $b = \begin{pmatrix} 0 & \lambda^0 & 0 \end{pmatrix}^T$. The corresponds the linear system, $A^TA\lambda = A^Tb$, is evaluated by a conjugate gradient solver.

6.6.3 Interpolation of Eigenvectors

For the eigenvector interpolation, which is performed component-wise, the formulation of the Laplacian operator and user-specified constraints is analogous, with the exception that the equilibrium equation is not considered in the minimization problem.

Note that due to the component-wise interpolation of the eigenvectors, it is finally required to orthonormalize the resulting fields in order to obtain a set of mutually perpendicular eigenvector fields. In the orthonormalization process, the eigenvector along the normal direction is fixed, while the one along Langer's line is clipped and normalized.

6.6.4 Dynamic Simulation

The incorporation of the residual stresses into the simulation of the deformable body leads to the spatially discretized Lagrangian equation of motion

$$T\overline{\sigma} + M\ddot{u} + D\dot{u} + Ku = f. \tag{6.10}$$

Here M, D, and K are the mass, damping, and stiffness matrix, and u and f are the linearized vertex displacements and per-vertex forces. Note that the matrices in this equation have to be reassembled whenever the topology of the object has changed, i.e., whenever the object has been cut. To enable the accurate simulation of large deformations using linear elasticity, we employ the corotational formulation of the linear strain tensor. The linear system of equations resulting from applying the implicit Newmark time integration scheme is solved by using an efficient geometric multigrid solver [DGW11a].

6.7 Results and Discussion

For our experiments, the Young's modulus and Poisson's ratio are chosen as 80 kPa and 0.4 respectively, which are comparable to those of an abdominal wall [SAF+06]. The tension along the Langer's lines is typically prescribed as 20 kPa, while along the cross direction it is 10 kPa [JJKG08]. Our experiments were run on a standard desktop PC equipped with an Intel Core Q9450 processor running at 2.66 GHz (a single core is used), 4 GB of RAM, and an NVIDIA GeForce GTX 280 graphics card.

Figure 6.8 shows two cuts, the left cut crosses Langer's lines, while the right one follows Langer's lines. It can be seen that the wound caused by the cross cut is wider.

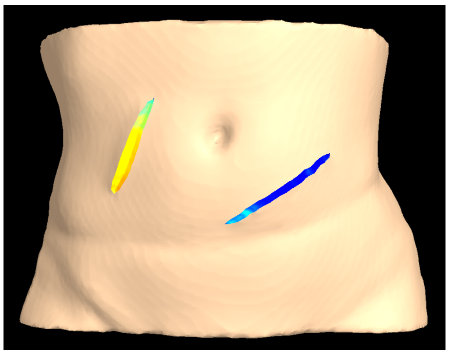


Figure 6.8: *The left cut crosses the Langer's lines, while the right cut follows Langer's lines (the lines are shown in Figure 6.2). The wound caused by the cross cut is wider.*

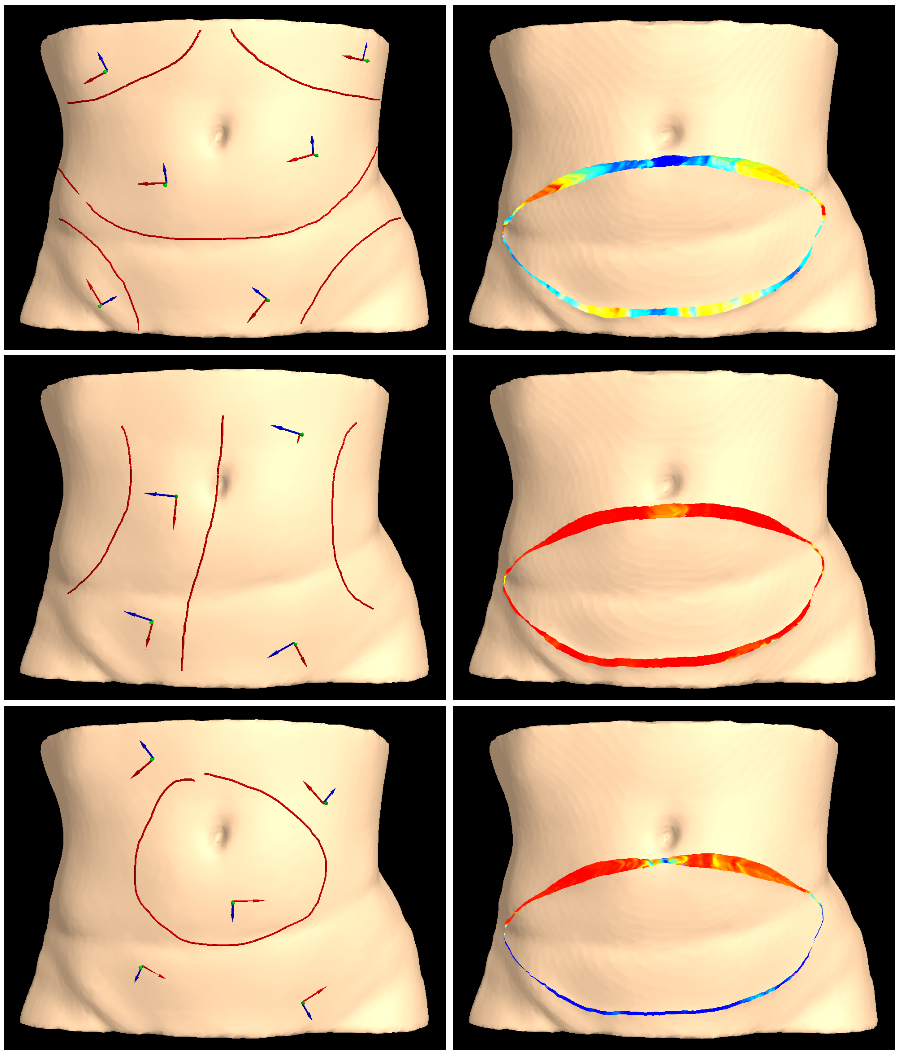


Figure 6.9: *Left: Three different specifications of eigenvectors and eigenvalues. Right: Flap surgery simulations with residual stresses computed from different specifications. The cutting surfaces are colored by the magnitude of the residual stress component in the direction of the cutting surface normal. The first row corresponds to a modeling of the residual stress in the spirit of Langer's lines.*

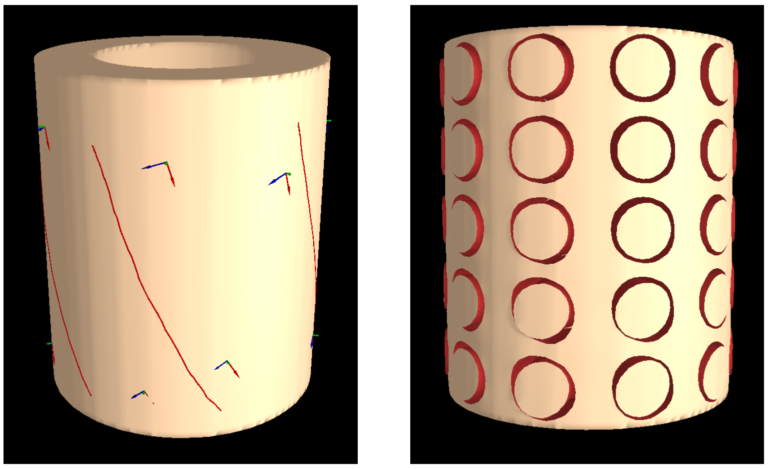


Figure 6.10: *Modeling and visualization on a cylinder model. The stresses in the upper part are set isotropic, while that in the lower part are set anisotropic. The resulting shapes of round incisions are circular and elliptical respectively.*

Figure 6.9 the first row shows the simulation of a flap surgery using the residual stress distribution computed from a reasonable modeling. Dynamic simulation is shown in the video accompanied with [WBWD12]. The width of the wound is typically $5 \sim 20$ *mm*. So far, the doctors were not able to clearly differentiate the Langer's lines for different types of bodies. The doctors, however, found our simulated cuts to be in line with what they were expecting for the particular body type presented in this chapter.

Flap surgery simulations with residual stresses computed from different eigenvector and eigenvalue specifications are shown in Figure 6.9. This demonstrates the variations of wound openings depending on the residual stress distribution.

Figure 6.10 shows our modeling and visualization techniques applied to a cylinder model. In the upper part of the cylinder, we assign the same eigenvalue on the Langer's direction and the cross-Langer's direction. In the lower part, the eigenvalue on the cross-Langer's direction is reduced by half. As a consequence, in the upper part the shape of incisions is still circular, while those in the lower part become ellipses.

6.7.1 Evaluation

In collaboration with plastic surgeons, we carried out an initial evaluation of the residual stress modeling procedure specifically for flap surgery. The medical objective of this flap surgery is to reconstruct a breast: The flap taken from the abdomen site would be trimmed, reshaped, and relocated to form a natural-looking breast shape. This plastic surgery is usually performed in the situation that a breast is removed due to the

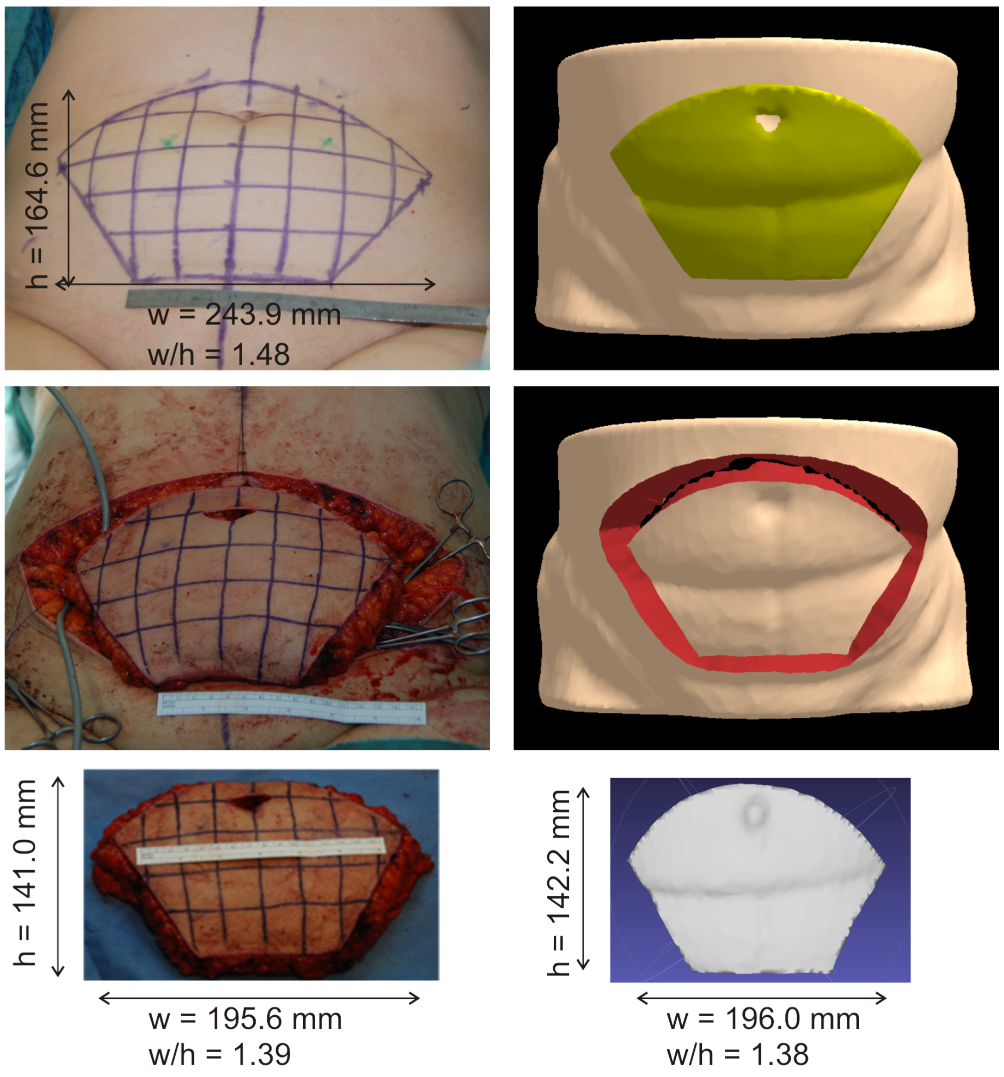


Figure 6.11: *A comparison between the real surgery scenarios and the simulations. First row: A patient abdomen with a preoperative surgical plan drawn on the skin (left) and the corresponding 3D model of the abdomen superimposed with the planned flap contour (right). Second row: The abdomen after excising the flap in the real surgery (left) and the simulated abdomen (right). Third row: The shrunken flap shape in the real surgery (left) and in the simulation (right).*

	Width	Height	Width/height
Preoperative	243.9 mm	164.6 mm	1.48
Intra-operative	195.6 mm	141.0 mm	1.39
Simulation	196.0 mm	142.2 mm	1.38
Relative error	0.20 %	0.85%	0.64 %

Table 6.1: *Measurements of the flap.*

treatment of breast cancer. The flap volume is a crucial factor in this plastic surgery. In particular, the shrinkage of the flap after excision makes the preoperative planning a challenging task.

In this evaluation, the residual stress modeling procedure is applied to introduce a stress tensor field into a patient abdomen model, and then the flap shrinkage is simulated using the finite element method on a uniform hexahedral grid. A comparison between the real surgery scenarios and the simulations is given in Figure 6.11. The second row shows that the simulation of the abdomen is very similar to that in the real surgery.

The measurements of the flap in preoperative, intra-operative, and in simulation are given in Table 6.1. The relative errors of the simulation results compared to the intra-operative results are smaller 1%. This suggests that the simulation has a high potential to provide useful information to assist surgeons in surgery planning.

6.7.2 Performance

Table 6.2 shows the performance of our system for different models and different resolutions. The second group of columns gives information about the models, i.e., the resolution, the number of hexahedral finite elements, and the number of simulation vertices. The third group shows the timing statistics for each modeling step. The first column gives the time for the eigenvector interpolation, which includes the interpolation of the user-specified Langer's lines and the surface normals, orthonormalization and the computation of the third eigenvector. The next column shows time for the interpolation of the user-specified eigenvalues. The last column gives the time for a single step of the dynamic deformation simulation. The statistics demonstrate that for models of a few thousands of cells, the model can be computed in one second. For models consisting of forty thousand cells, the computations take a few seconds.

				Time [ms]		
Model	Resolution	# Cells	# Vertices	Vector	Value	Sim.
Cylinder	21×21×26	5,550	7,540	261.4	215.2	83.3
Cylinder	42×42×51	44,000	51,714	5,517	8,951	637.4
Abdomen	63×21×42	5,734	10,189	155.6	106.1	98.6
Abdomen	126×41×84	45,313	62,481	3,046	3,607	707.8

Table 6.2: *Performance statistics for different models and different resolutions. The last three columns give times for the eigenvector interpolation, the eigenvalue interpolation, and the dynamic simulation, respectively.*

6.8 Conclusion and Future Work

We have presented methods to design and visualize a patient-specific residual stress tensor field with respect to physical constraints. The design is based on the mechanics of residual stress in soft biological tissues. We proposed a method for interactively designing tensor fields via a sketch-based interface. A graphical depiction of the residual stress distribution is achieved by visualizing the simulated deformations of a set of small round incisions. In a number of examples we demonstrated the potential of our approach in patient-specific surgery simulation, where a consistent residual stress tensor field is desired.

Our method is well suited for modeling the residual stress distribution on shell-like structures, which are common in biological tissues. For complex-shaped structures, it would be interesting to allow the user to draw stress directions inside the body, if the user has *prior* knowledge of the interior residual stress. Meanwhile, as a modeling tool, the accuracy of its results depends heavily on the correctness of user's inputs. Thus it would be beneficial to incorporate into the modeling framework accurate measurements of some parameters as a complement to surgeon's experience. Another issue is that the user-sketched stroke has a specific direction, while the eigenvector of a tensor is actually bidirectional. This specification may produce some artifacts in vector field generation. Further investigation is needed to address this open problem.

In the future we will also investigate the parallelization of our approach on GPU and multi-core architectures. Even though our method already achieves quite satisfying update rates, it can still take several seconds to update a residual stress field with respect to the user input. In order to achieve good scalability on a parallel system, we will in particular address the integration of iterative solution methods into our approach. Moreover, we will further evaluate the confidence of our results with respect to real-world measurement on more patient models.

Chapter 7

The Haptic Cutting System

This chapter presents the integrated system of virtual cutting simulation, collision detection, and haptic interaction. In Section 7.1, we give an overview of the components and the data flow between them. In Section 7.2, we explain the employed haptic rendering algorithm. Our system delivers a real-time performance of 15 simulation frames per second for haptic soft tissue cutting of a deformable body at an effective resolution of 170,000 finite elements.

7.1 Overview

To employ our method for virtual surgery training, we have integrated haptic feedback into the simulation. The haptic cutting system is composed of three computational components: CFE simulation, collision detection, and haptic simulation. These three components form two simulation loops (see Figure 7.1). The deformation simulation loop (blue) and the haptic simulation loop (orange) are updated at different frequencies. The components in the deformation loop (i.e., CFE simulation and collision detection) have been presented in Chapter 4 and Chapter 5. The haptic simulation is to be presented in Section 7.2.

The input to the CFE simulation is the closed boundary surface mesh of the deformable object, material properties (Young's modulus and Poisson's ratio), and some simulation specific parameters (e.g., the time step and the composition level). The surface mesh is first rasterized into a uniform Cartesian grid, from which a linked octree is then adaptively computed. The composite elements are constructed from the linked

The result section of this chapter is based on material that has been originally published in J. Wu, R. Westermann, and C. Dick, *Real-time haptic cutting of high-resolution soft tissues*, Studies in Health Technology and Informatics (Proc. Medicine Meets Virtual Reality 21) **196** (2014), 469–475.

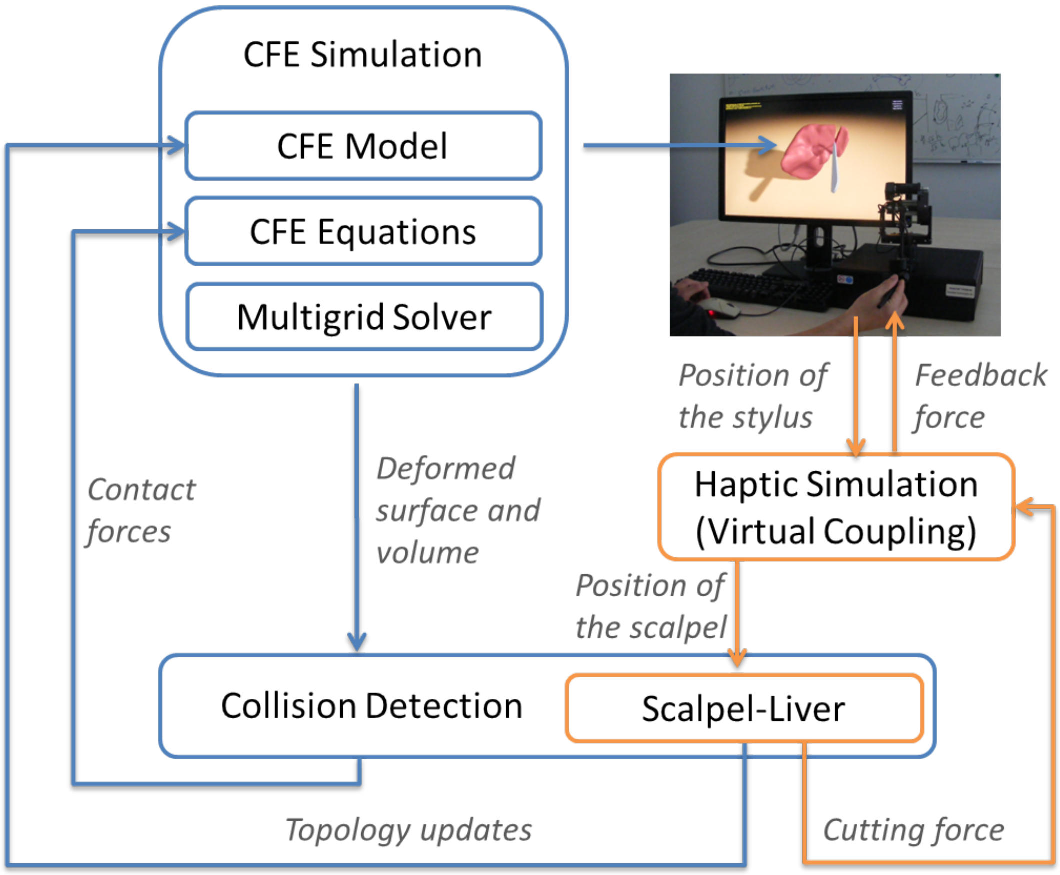


Figure 7.1: *The haptic cutting system maintains a deformation simulation loop (blue) and a haptic simulation loop (orange) that are updated at different frequencies.*

octree representation (Section 4.3). Typically, we employ a composition level of 2 or 3, depending on the resolution of the object. The finite element matrices are computed for each composite element by considering its underlying octree cells (Section 4.4). We simulate the dynamics of linear elastic materials with the corotational strain formulation. The linear system resulting from the implicit time integration is solved using a geometric multigrid solver (Section 4.5). The computed deformation of the composite elements is propagated to the underlying octree cells, and then to the surface mesh which is constructed from the linked octree representation (Section 4.3). The deformed high-resolution surface mesh is used for visual rendering and serves as an input for the collision detection.

The collision detection checks for collisions between surface vertices and deformed octree cells. First, a spatial subdivision approach determines potentially colliding pairs consisting of a surface vertex and a composite element (Section 5.4.1). Second, a narrow phase evaluates for each pair the penetration depth and direction from a signed distance field. Special attention is paid to accurately interpolate the penetration depth along complicated boundaries (Section 5.4.2). From the collision detection results, contact forces are computed based on a penalty-based contact force model. The contact forces are forwarded to the CFE simulation as updated boundary conditions.

The collision detection also checks the intersection between the sweep surface of a scalpel and the links connecting the octree cells. The detected intersections are used to modify the topology of the simulation grid. Note that since the haptic loop has a much higher update rate compared to the deformation loop, in the collision detection with a moving scalpel, we assume the object is static during two deformation cycles.

7.2 Haptic Rendering

To obtain an intuitive feedback force, we use a velocity-proportional force model. Intuitively, if the user moves the scalpel with a high speed against the deformable object, (s)he feels a large resistant force. The force direction is opposite to the direction of movement, and the force magnitude is proportional to both the speed of movement and the contact volume between the scalpel and the deformable object. To compute the force, the scalpel is discretized into a number of sample points. We first compute an elementary cutting force for each sample. The overall cutting force is then obtained as the sum of these elementary forces.

The velocity of the manipulated scalpel is estimated from the position signal provided by the haptic device. This data is known to be noisy, and may lead to vibration of the device. To improve the stability of the velocity-proportional force model, we apply the virtual coupling approach (see Section 2.4.1), an artificial spring coupling the motion of the haptic device in the physical world and the motion of the scalpel in the virtual environment.

The *static* virtual coupling approach computes the position of the virtual scalpel as formulated in Eq. 2.30. At this position of the virtual scalpel, the virtual coupling force balances the cutting force, corresponding to a static equilibrium status. Using a spring force model with zero rest length, the virtual coupling force is calculated by

$$\boldsymbol{f}_{vc} = k_{vc}\left(\boldsymbol{x}_{stylus} - \boldsymbol{x}_{virtual}\right), \tag{7.1}$$

where k_{vc} is the virtual coupling stiffness, which in practice is limited by the maximum renderable stiffness of a haptic device. The position of the stylus $\boldsymbol{x}_{stylus}$ is read from the

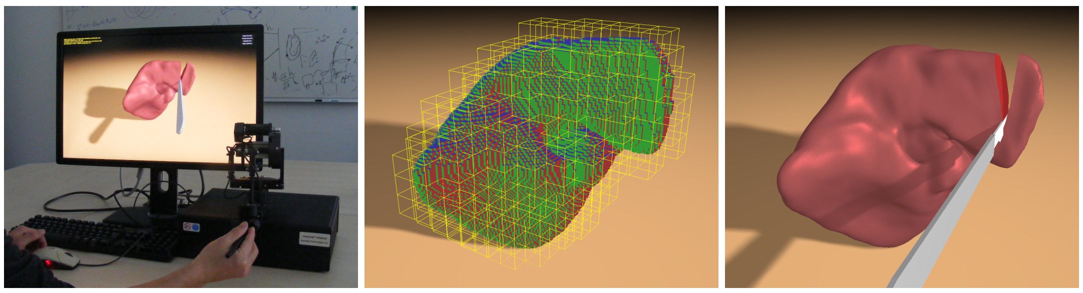


Figure 7.2: *Haptic cutting of a liver model. Left: Setup. Middle: High-resolution hexahedral elements and coarse composite elements. Right: High quality surface mesh used for visualization. The collision detection between the scalpel and the deformable liver model is performed at 1 kHz haptic rates.*

haptic device. The coordinate system of the device is transformed to be aligned with the coordinate system in the virtual world.

The cutting force is calculated as

$$\boldsymbol{f}_c = -n_c \, b_c \frac{\Delta \boldsymbol{x}_{virtual}}{\Delta T}, \tag{7.2}$$

where n_c is the number of samples which penetrate into the object, b_c is a scalar parameter, and $\Delta T = 1\,ms$ is the haptic rendering time step.

Besides haptic rendering of the cutting force, we also support the haptic rendering of contact simulation. The user switches between the contact mode and the cutting mode by pressing/releasing a button on the stylus. The contact force is calculated using a conventional penalty force model which is proportional to the penetration depth, and points to the closest point on the object's surface.

While the position of the virtual tool is computed based on the virtual coupling scheme, we directly map the orientation of the stylus to the orientation of the virtual tool, i.e., no virtual coupling is applied for the rotational motion. This is intuitive if the device has no torque feedback. Let us refer to [BJ08] if virtual coupling for the rotational motion is desired.

7.3 Results

We demonstrate the performance of haptic cutting on a volumetric liver model. The adaptive octree finite element model of the liver consists of 40,080 hexahedral elements, corresponding to 173,843 elements on a $82 \times 83 \times 100$ uniform grid. Applying three levels of composition ($8^3 : 1$) leads to 647 composite elements with 2,928 DOFs (see Figure 7.2). The surface mesh reconstructed from the hexahedral grid has 58,920 triangles. At that resolution, very fine details are visible on the cutting surface. In each simulation time step, the composite finite element simulation takes 23.3 ms. The detection of both inter- and intra-collisions takes 3.2 ms. In total, the simulation runs at 38 simulation frames per second.

Cutting takes additional time for octree subdivision and re-creation of the coarse grid hierarchy and the render surface (12 ms), reassembly of element matrices (25 ms), as well as updating the signed distance field for collision detection (2 ms). In total, during cutting, the simulation is running at 15 simulation frames per second.

Note that the haptic rendering loop is decoupled from the visual rendering loop: the collision detection between the scalpel and the organ as well as the computation of the feedback force are performed at 1 kHz, whereas soft tissue cutting is performed at 15 Hz. In our example, the blade is represented by a set of triangles with a total of 30 vertices. The collision detection between this small set of vertices and the organ takes less than 1 ms.

For a demonstration of the quality of our haptic cutting approach, please see the accompanying video at http://wwwcg.in.tum.de/research/research/projects/real-time-haptic-cutting.html. This live recording shows cutting of a liver model by manipulating a haptic device, which is mapped to a virtual scalpel in the virtual environment. Figure 7.3 shows a sequence of images from this recording. Feedback forces are depicted in Figure 7.4.

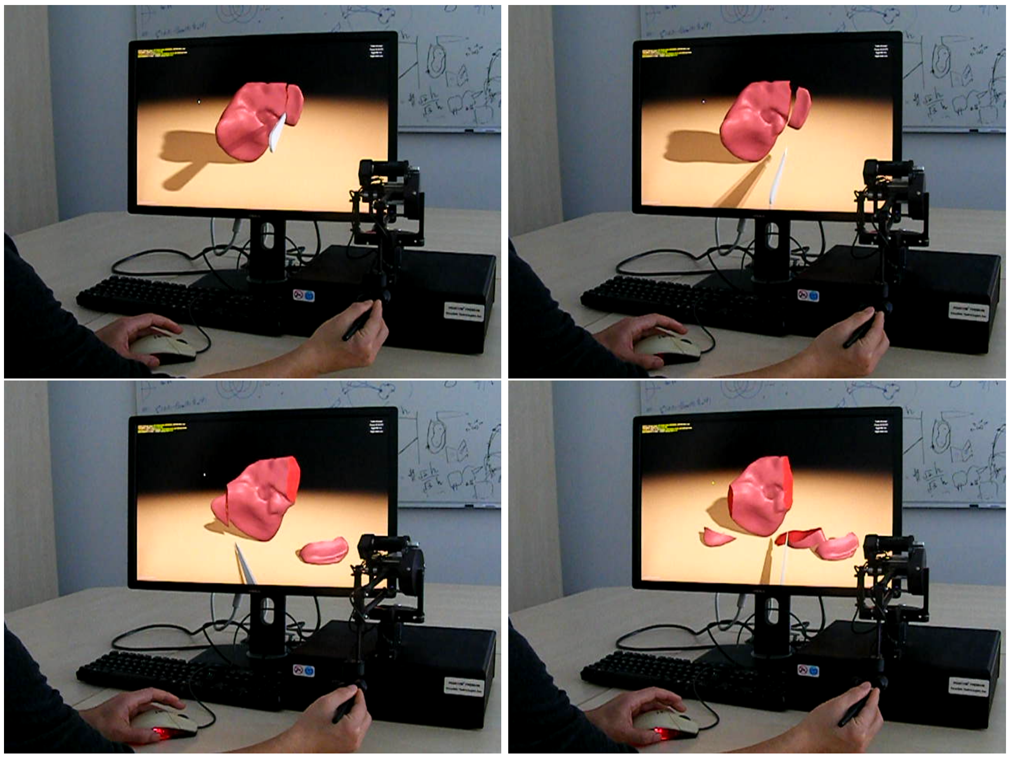


Figure 7.3: *A sequence of images from a live recording of haptic cutting, available at http://wwwcg.in.tum.de/research/research/projects/real-time-haptic-cutting.html.*

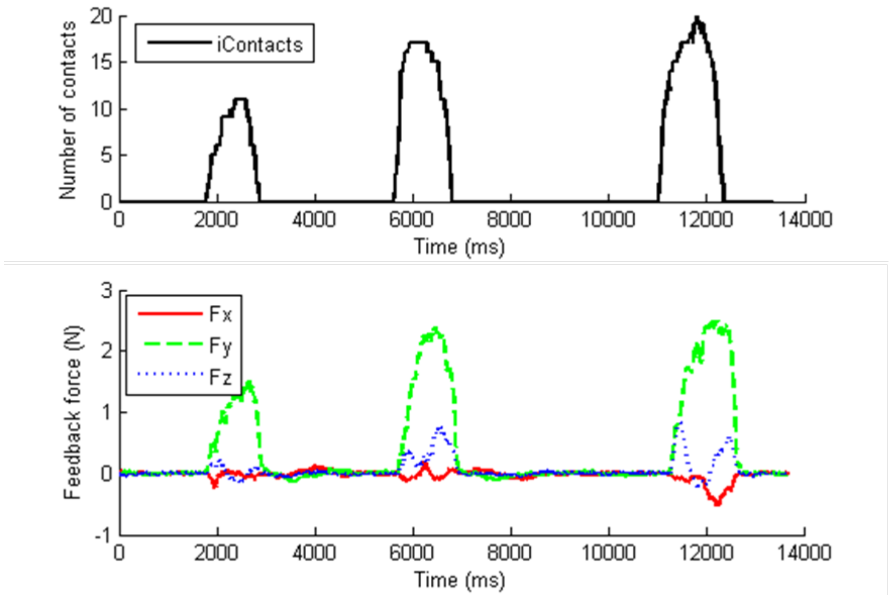


Figure 7.4: *Top: Number of contacts between the scalpel and the liver model. Bottom: Computed feedback forces while making three cuts.*

Chapter 8

Conclusion

In this thesis, we have presented novel computational models for simulating deformable bodies with changing topology. To the best of our knowledge, the achieved quality and performance of the virtual cutting simulator has not been reported in the literature. Considering that our approach increases the finite element resolution for physically accurate interactive virtual cutting simulation by an order of magnitude, our technique has a high potential to significantly improve the realism of surgery simulators.

The key to this advancement is a composite finite element formulation, which effectively reduces the number of the simulation degrees of freedom, while maintaining a good approximation of the mechanical properties in the full resolution discretization. The composite formulation based on a hexahedral discretization involves only local operations. This allows for real-time processing of progressive cuts in the simulation domain. In addition, an efficient geometric multigrid solver is employed to solve the system of equations resulting from the implicit numerical time integration of linear elasticity with the corotational strain formulation.

In addition to the deformation simulation, we have presented a collision detection method which is specifically tailored for the proposed composite elements. The results have demonstrated that the composite formulation not only leads to a faster deformation simulation, but also enables a faster collision detection. We have further proposed a method to improve the accuracy of penetration depth evaluation, alleviating the staircases of the hexahedral discretization along the new surfaces introduced by cuts.

Furthermore, we have presented the first interactive method for modeling a physically meaningful residual stress distribution into a patient-specific model. It has been demonstrated that the residual stress significantly influences the deformation of soft tissues during cutting. The incorporation of the residual stress distribution into the considered flap surgery simulation leads to a realistic simulation of the shrinkage of the flap. A comparison between the simulated flap and the real flap on the same patient shows a promising match.

8.1 Future Work

The research presented in this thesis opens up many possibilities. We highlight the following research topics which we plan to conduct in the near future.

An open research problem is the physical interaction between a cutting tool and deformable materials. In current virtual cutting practice, the topological changes are introduced by purely geometric intersection tests. It can be interpreted as moving an infinitely sharp tool which cuts the material as long as the tool sweeps, without any force exchange between the tool and the material. In contrast, in the physical world, the material would deform under the influence of the increasing force exerted by the cutting tool, before the tool eventually penetrates. To accurately simulate this physical interaction, we need to further research on the contact force model and the fracture model. The contact force model computes the interaction force between the tool and the material. The fracture model determines when and where the topological change happens, based on the computed contact force, possibly together with the deformation status and the deformation rate. Both the contact force model and the fracture model in the context of soft tissue cutting simulation are largely unclear.

Contact force models for deformable bodies is a research area which we have not yet fully studied. Under the constraint of interactive simulation rates, we currently resolve the collisions by penalty forces. This efficient method produces penetration artifacts, and involves several parameters which affect the simulation results. An alternative is the constraint-based approach, which formulates the unilateral constraint of non-penetration as a linear complementarity problem (LCP). The solution of this LCP are contact forces which accurately resolve the collisions. It would be interesting to design multigrid solvers to efficiently solve this LCP.

Fracture simulation on octree grids is another interesting direction. In the literature there exist different fracture models, some of which have been studied in computer animation based on a tetrahedral discretization. Typically, the topological change occurs if an eigenvalue of the stress tensor of an element exceeds a given threshold. However, this criterion alone would result in shattering artifacts, i.e., many elements in a small region are disconnected. To revolve this problem, methods are proposed to suppress shattering and to adaptively refine tetrahedral elements. It is unclear how to efficiently implement these models in an adaptive hexahedral discretization.

In addition, we will apply the efficient finite element analysis in engineering design tasks. An emerging application is 3D printing, which enables flexible fabrication of complicated shapes with a substantially reduced cost compared with traditional manufacturing methods. However, the analysis of mechanical properties of these complicated shapes presents a new challenge, since such detailed shapes would require an extremely high-resolution discretization. From a computational point of view, the composite finite element approach developed in this thesis would be a promising tool to support the mechanical analysis for 3D printing.

Bibliography

[ABA00] M. Aftosmis, M. Berger, and G. Adomavicius. A parallel multigrid method for adaptively refined cartesian grids with embedded boundaries, AIAA 2000-0808. In *38th AIAA Aerospace Sciences Meeting and Exhibit*, 2000.

[ACPA06] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A log-euclidean framework for statistics on diffeomorphisms. In Rasmus Larsen, Mads Nielsen, and Jon Sporring, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2006*, volume 4190 of *Lecture Notes in Computer Science*, pages 924–931. Springer Berlin Heidelberg, 2006.

[AFC$^+$10] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph.*, 29(4):82:1–82:10, July 2010.

[AH98] R.J. Adams and B. Hannaford. A two-port framework for the design of unconditionally stable haptic interfaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1254–1259 vol.2, Oct 1998.

[Bat96] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, 1996.

[BB99] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, 45(5):601–620, 1999.

[BG00] Daniel Bielser and Markus Gross. Interactive simulation of surgical cuts. In *Pacific Graphics*, pages 116–125, 2000.

[BGTG04] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. *Graphical Models*, 66(6):398 – 417, 2004.

[BJ05] Jernej Barbič and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM SIGGRAPH*, pages 982–990, 2005.

[BJ08] Jernej Barbič and Doug L. James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Trans. Haptics*, 1(1):39–52, January 2008.

[BJ10] J. Barbič and D.L. James. Subspace self-collision culling. *ACM Trans. Graph.*, 29(4):81, 2010.

[BLG94] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

[BM97] I. Babuška and J. M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997.

[BMG99] Daniel Bielser, Volker A. Maiwald, and Markus H. Gross. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum*, 18(3):31–38, 1999.

[BN98] Morten Bro-Nielsen. Finite element modeling in surgery simulation. *Proceedings of the IEEE*, 86(3):490–503, 1998.

[BNC96] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, 1996.

[BOYBJK90] Frederick P. Brooks, Jr., Ming Ouh-Young, James J. Batter, and P. Jerome Kilpatrick. Project gropehaptic displays for scientific visualization. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 177–185, New York, NY, USA, 1990. ACM.

[BS01] Cynthia D. Bruyns and Steven Senger. Interactive cutting of 3D surface meshes. *Computers & Graphics*, 25(4):635 – 642, 2001.

[BSM$^+$02] Cynthia D. Bruyns, Steven Senger, Anil Menon, Kevin Montgomery, Simon Wildermuth, and Richard Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The Journal of Visualization and Computer Animation*, 13(1):21–42, 2002.

[BZH$^+$05] Daniel Bachofen, János Zátonyi, Matthias Harders, Gábor Székely, P Fruh, and Markus Thaler. Enhancing the visual realism of hysteroscopy simulation. *Studies in health technology and informatics*, 119:31, 2005.

[CAK$^+$14] Hadrien Courtecuisse, Jérémie Allard, Pierre Kerfriden, Stéphane P.A. Bordas, Stéphane Cotin, and Christian Duriez. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis*, 18(2):394 – 410, 2014.

[CAR$^+$09] Nuttapong Chentanez, Ron Alterovitz, Daniel Ritchie, Lita Cho, Kris K. Hauser, Ken Goldberg, Jonathan R. Shewchuk, and James F. O'Brien. Interactive simulation of surgical needle insertion and steering. *ACM Trans. Graph.*, 28(3):88:1–88:10, July 2009.

[CB94] J. Edward Colgate and J. Michael Brown. Factors affecting the z-width of a haptic display. In *IEEE International Conference on Robotics and Automation*, pages 3205–3210. IEEE, 1994.

[CDA00] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[CDL07] T. Chanthasopeephan, J.P. Desai, and A.C.W. Lau. Modeling soft-tissue deformation prior to cutting for surgical simulation: Finite element analysis and study of cutting parameters. *IEEE Transactions on Biomedical Engineering*, 54(3):349–359, 2007.

[CF86] Cheng-Jen Chuong and Yuan-Cheng Fung. On residual stresses in arteries. *J. Biomech. Eng.*, 108(2):189–192, 1986.

[CGSS93] J. Edward Colgate, Paul E. Grafing, Michael C. Stanley, and Gerd Schenkel. Implementation of stiff virtual walls in force-reflecting interfaces. In *IEEE Virtual Reality Annual International Symposium*, pages 202–208. IEEE, 1993.

[Cia88] Philippe G Ciarlet. *Mathematical elasticity, volume I: Three-dimensional elasticity*. Elsevier, 1988.

[CJA$^+$10] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D.Y. Lee, and S. Cotin. GPU-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology*, 103(2-3):159 – 168, 2010.

[CMJ11] T.R. Coles, D. Meglan, and N. John. The role of haptics in medical training simulators: A survey of the state of the art. *IEEE Transactions on Haptics*, 4(1):51–66, Jan 2011.

[CS97] J. Edward Colgate and Gerd G. Schenkel. Passivity of a class of sampled-data systems: Application to haptic interfaces. *Journal of robotic systems*, 14(1):37–47, 1997.

[CSB95] J.E. Colgate, M.C. Stanley, and J.M. Brown. Issues in the haptic display of tool use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 140–145, Aug 1995.

[Cui97] Olivier Cuisenaire. Region growing euclidean distance transforms. In *The 9th International Conference on Image Analysis and Processing*, pages 263–270, 1997.

[DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of SIGGRAPH*, pages 287–290, 1995.

[DGBW08] Christian Dick, Joachim Georgii, Rainer Burgkart, and Rüdiger Westermann. Computational steering for patient-specific implant planning in orthopedics. In *Proceedings of Visual Computing for Biomedicine '08*, pages 83–92, 2008.

[DGBW09] Christian Dick, Joachim Georgii, Rainer Burgkart, and Rüdiger Westermann. Stress tensor field visualization for implant planning in orthopedics. *IEEE Trans. Vis. & Comput. Graph. (Proc. IEEE Visualization)*, 15(6):1399–1406, 2009.

[DGW11a] Christian Dick, Joachim Georgii, and Rüdiger Westermann. A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Trans. Vis. & Comput. Graph.*, 17(11):1663–1675, 2011.

[DGW11b] Christian Dick, Joachim Georgii, and Rüdiger Westermann. A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Simulation Modelling Practice and Theory*, 19(2):801–816, 2011.

[ED08] Elmar Eisemann and Xavier Décoret. Single-pass GPU solid voxelization for real-time applications. In *Proceedings of Graphics Interface*, pages 73–80, 2008.

[FBAF08] François Faure, Sébastien Barbier, Jérémie Allard, and Florent Falipou. Image-based collision detection and response between arbitrary volume objects. In *SCA '08: Symposium on Computer Animation*, pages 155–162. EG, 2008.

[FDD+12] Franois Faure, Christian Duriez, Herv Delingette, Jrmie Allard, Benjamin Gilles, Stphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stphane Cotin. SOFA: A multi-model framework for interactive physical simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer Berlin Heidelberg, 2012.

[FG99] Sarah F. Frisken-Gibson. Using linked volumes to model object collisions, deformation, cutting, carving, and joining. *IEEE Trans. Vis. & Comput. Graph.*, 5(4):333–348, 1999.

[FL01] Susan Fisher and Ming C. Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Eurographic workshop on Computer animation and simulation*, pages 99–111, 2001.

[FM03] Thomas-Peter Fries and Hermann G Matthies. Classification and overview of meshfree methods. *Technical University of Braunschweig*, 2003.

[FSHH12] B. Fierz, J. Spillmann, I.A. Hoyos, and M. Harders. Maintaining large time steps in explicit finite element simulations using shape matching. *Visualization and Computer Graphics, IEEE Transactions on*, 18(5):717–728, May 2012.

[FTN11] Cormac Flynn, Andrew Taberner, and Poul Nielsen. Mechanical characterisation of in vivo human skin using a 3d force-sensitive micro-robot and finite element analysis. *Biomechanics and Modeling in Mechanobiology*, 10:27–38, 2011.

[GCMS00] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum*, 19(3):271–281, 2000.

[GEB+14] J. Georgii, M. Eder, K. Burger, S. Klotz, F. Ferstl, L. Kovacs, and R. Westermann. A computational tool for preoperative breast augmentation planning in aesthetic plastic surgery. *IEEE Journal of Biomedical and Health Informatics*, 18(3):907–919, May 2014.

[GENR+14] O.A. González-Estrada, E. Nadal, J.J. Ródenas, P. Kerfriden, S.P.A. Bordas, and F.J. Fuenmayor. Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Computational Mechanics*, 53(5):957–976, 2014.

[GLB+06] Xiaohu Guo, Xin Li, Yunfan Bao, Xianfeng Gu, and Hong Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE Transactions on Visualization and Computer Graphics*, 12:375–385, 2006.

[GMD13] Loeiz Glondu, Maud Marchal, and Georges Dumont. Real-time simulation of brittle fracture using modal analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):201–209, February 2013.

[GO06] A. Guillou and R.W. Ogden. Growth in soft biological tissue and residual stress development. In Gerhard A. Holzapfel and Ray W. Ogden, editors, *Mechanics of Biological Tissue*, pages 47–62. 2006.

[GRC$^+$05] Anthony G Gallagher, E Matt Ritter, Howard Champion, Gerald Higgins, Marvin P Fried, Gerald Moses, C Daniel Smith, and Richard M Satava. Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. *Annals of surgery*, 241(2):364, 2005.

[GSM$^+$12] L. Glondu, S.C. Schvartzman, M. Marchal, G. Dumont, and M.A. Otaduy. Efficient collision detection for brittle fracture. In *SCA '12: Symposium on Computer Animation*, pages 285–294, 2012.

[GW06] Joachim Georgii and Rüdiger Westermann. A multigrid framework for real-time simulation of deformable bodies. *Computer & Graphics*, 30:408–415, 2006.

[GW08] Joachim Georgii and Rüdiger Westermann. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, pages 11–19, 2008.

[Hag89] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[HFS$^+$01] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Proc. the Computer Animation*, pages 136 –254, 2001.

[Hig86] Nicholas J. Higham. Computing the polar decomposition—with applications. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1160–1174, 1986.

[HJST13] Jan Hegemann, Chenfanfu Jiang, Craig Schroeder, and Joseph M. Teran. A level set method for ductile fracture. In *SCA '13: Symposium on Computer Animation*, pages 193–201. ACM, 2013.

[HKSH09] R. Hover, G. Kosa, G. Szekely, and M. Harders. Data-driven haptic rendering: From viscous fluids to visco-elastic solids. *Haptics, IEEE Transactions on*, 2(1):15–27, Jan 2009.

[HLL97] Lambertus Hesselink, Yuval Levy, and Yingmei Lavin. The topology of symmetric, second-order 3D tensor fields. *IEEE Trans. Vis. & Comput. Graph.*, 3(1):1–11, 1997.

[HLSO12] Florian Hecht, Yeon Jin Lee, Jonathan R. Shewchuk, and James F. O'Brien. Updated sparse cholesky factors for corotational elastodynamics. *ACM Transactions on Graphics*, 31(5):123:1–13, October 2012.

[Hog86] Anne Hoger. On the determination of residual stress in an elastic body. *Journal of Elasticity*, 16:303–324, 1986.

[HS97] W. Hackbusch and S.A. Sauter. Composite finite elements for the approximation of pdes on domains with complicated micro-structures. *Numerische Mathematik*, 75:447–472, 1997.

[HS04] Michael Hauth and Wolfgang Straßer. Corotational simulation of deformable solids. In *Proceedings of WSCG*, pages 137–145, 2004.

[HSK^{+}10] Jae-Pil Heo, Joon-Kyung Seong, DukSu Kim, Miguel A. Otaduy, Jeong-Mo Hong, Min Tang, and Sung-Eui Yoon. Fastcd: fracturing-aware stable collision detection. In *SCA '10: Symposium on Computer Animation*, pages 149–158, 2010.

[HTG04] B. Heidelberger, M. Teschner, and M. Gross. Detection of collisions and self-collisions using image-space techniques. *Journal of WSCG*, 12(3):145–152, 2004.

[HTWB11] Jin Huang, Yiying Tong, Hongyu Wei, and Hujun Bao. Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph.*, 30(6):143:1–143:8, 2011.

[Hum03] J.D. Humphrey. Review paper: Continuum biomechanics of soft biological tissues. *Proc. Royal Society of London. Series A*, 459(2029):3–46, 2003.

[HVS^{+}09] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. Asynchronous contact mechanics. *ACM Trans. Graph.*, 28(3):87:1–87:12, July 2009.

[HZLM01] Kenneth E. Hoff, III, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast and simple 2d geometric proximity queries using graphics hardware. In *I3D '01: Symposium on Interactive 3D graphics*, pages 145–148. ACM, 2001.

[Inc] Intuitive Surgical Inc. da Vinci skills simulator. `http://www.intuitivesurgical.com/products/skills_simulator/`. [Online; accessed 17-June-2014].

[JBB^{+}10] Lenka Jeřábková, Guillaume Bousquet, Sbastien Barbier, Franois Faure, and Jrmie Allard. Volumetric modeling and interactive cutting of deformable bodies. *Progress in Biophysics and Molecular Biology*, 103(2-3):217 – 224, 2010.

[JBS06] M.W. Jones, J.A. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Trans. Vis. & Comput. Graph.*, 12(4):581 –599, july-aug. 2006.

[JJKG08] Emmanuelle Jacquet, Gwendal Josse, Fouad Khatyr, and Camille Garcin. A new experimental method for measuring skin's natural tension. *Skin Research and Technology*, 14(1):1–7, 2008.

[JJMW13] Xia Jin, GrandRoman Joldes, Karol Miller, and Adam Wittek. 3D algorithm for simulation of soft tissue cutting. In Adam Wittek, Karol Miller, and Poul M.F. Nielsen, editors, *Computational Biomechanics for Medicine*, pages 49–62. 2013.

[JK09] Lenka Jeřábková and Torsten Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Comput. Graph. Appl.*, 29(2):61–71, 2009.

[JL12] Hoeryong Jung and Doo Yong Lee. Real-time cutting simulation of meshless deformable object using dynamic bounding volume hierarchy. *Computer Animation and Virtual Worlds*, 23(5):489–501, 2012.

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, 2002.

[JP04] D.L. James and D.K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. In *ACM Trans. Graph.*, volume 23, pages 393–398. ACM, 2004.

[KGRB13] P. Kerfriden, O. Goury, T. Rabczuk, and S.P.A. Bordas. A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 256(0):169–188, 2013.

[KMB$^+$09] Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph.*, 28(3):1–10, 2009.

[KMBG08] Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. Flexible simulation of deformable models using discontinuous galerkin FEM. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 105–115, 2008.

[KMOD09] Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph.*, 28:51:1–51:8, July 2009.

[KZPB04] Laszlo Kovacs, Alexander Zimmermann, Nikolaos A. Papadopulos, and Edgar Biemer. Re: factors determining shape and symmetry in immediate breast reconstruction. *Annals of Plastic Surgery*, 53(2):192–194, 2004.

[Lan61] Karl Langer. Zur anatomie und physiologie der haut. über die spaltbarkeit der cutis. *Sitzungsbericht der Mathematisch-naturwissenschaftlichen Classe der Wiener Kaiserlichen Academie der Wissenschaften Abt*, page 44, 1861.

[Lan78] Karl Langer. On the anatomy and physiology of the skin. *British journal of plastic surgery*, 31:3–8, 93–106, 1978. Translated by T. Gibson.

[LGLM99] Eric Larsen, Stefan Gottschalk, Ming C Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical report, Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[LJD07] Yi-Je Lim, Wei Jin, and Suvranu De. On some recent advances in multimodal surgery simulation: A hybrid approach to surgical cutting and the use of video images for enhanced realism. *Presence: Teleoper. Virtual Environ.*, 16(6):563–583, December 2007.

[LPO10] Bryan Lee, Dan C. Popescu, and Sbastien Ourselin. Topology modification for surgical simulation using precomputed finite element models based on linear elasticity. *Progress in Biophysics and Molecular Biology*, 103(2-3):236 – 251, 2010.

[LPR$^+$09] Florian Liehr, Tobias Preusser, Martin Rumpf, Stefan Sauter, and Lars Ole Schwen. Composite finite elements for 3d image based computing. *Computing in Visualization and Science*, 12(4):171–188, 2009.

[LS81] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981.

[LT07] Alex Lindblad and George Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3D continuum models. In *Proceedings of ACM symposium on Solid and Physical Modeling*, pages 421–429, 2007.

[LZW$^+$14] Shuai Li, Qinping Zhao, Shengfa Wang, Aimin Hao, and Hong Qin. Interactive deformation and cutting simulation directly using patient-specific volumetric images. *Computer Animation and Virtual Worlds*, 25(2):155–169, 2014.

[MBB$^+$11] M. Moumnassi, S. Belouettar, É. Béchet, S. P.A. Bordas, D. Quoirin, and M. Potier-Ferry. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering*, 200(5–8):774 – 796, 2011.

[MBF04] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.*, 23(3):385–392, 2004.

[MBP14] Lien Muguercia, Carles Bosch, and Gustavo Patow. Fracture modeling in computer graphics. *Computers & Graphics*, 2014.

[MDB99] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999.

[MDM$^+$02] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–54, 2002.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface*, pages 239–246, 2004.

[MH01] M. Mahvash and V. Hayward. Haptic rendering of cutting: A fracture mechanics approach. *Haptics-e*, 2(3):1–12, 2001.

[MK00] Andrew B. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 598–607, 2000.

[MKB$^+$08] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum*, 27(5):1521–1529, 2008.

[MKN$^+$04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, 2004.

[MMSE11] Jason Z. Moore, Kostyantyn Malukhin, Albert J. Shih, and Kornel F. Ehmann. Hollow needle tissue insertion force model. {*CIRP*} *Annals - Manufacturing Technology*, 60(1):157 – 160, 2011.

[MPT99] William A. McNeely, Kevin D. Puterbaugh, and James J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 401–408, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[MRO08] Sarthak Misra, K. T. Ramesh, and Allison M. Okamura. Modeling of tool-tissue interactions for computer-based surgical simulation: A literature review. *Presence: Teleoper. Virtual Environ.*, 17(5):463–491, October 2008.

[MZS$^+$11] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.*, 30(4):37:1–37:12, July 2011.

[NACC08] Siamak Niroomandi, Icíar Alfaro, Elías Cueto, and Francisco Chinesta. Real-time deformable models of non-linear tissues by model reduction techniques. *Computer Methods and Programs in Biomedicine*, 91(3):223–231, 2008.

[NAG$^+$12] S. Niroomandi, I. Alfaro, D. Gonzlez, E. Cueto, and F. Chinesta. Real-time simulation of surgery by reduced-order modeling and X-FEM techniques. *International Journal for Numerical Methods in Biomedical Engineering*, 28(5):574–588, 2012.

[NKJF09] Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph.*, 28(3):52:1–52:9, July 2009.

[NMK$^+$06] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[NRBD08] Vinh Phu Nguyen, Timon Rabczuk, Stphane Bordas, and Marc Duflot. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation*, 79(3):763 – 813, 2008.

[NS01] Han-Wen Nienhuys and A. Frank van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 145–152, 2001.

[NTV92] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics*, 10(5):307–318, 1992.

[NvdS00] Han-Wen Nienhuys and A. Frank van der Stappen. Combining finite element deformation with cutting for surgery simulations. In *Proceedings of Eurographics Short Presentations '00*, pages 43–52, 2000.

[OBH02] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH*, pages 291–294, 2002.

[OCSG07] M.A. Otaduy, O. Chassot, D. Steinemann, and M. Gross. Balanced hierarchies for collision detection between fracturing objects. In *VR '07: IEEE Virtual Reality Conference*, pages 83 –90, march 2007.

[OFTB96] D. Organ, M. Fleming, T. Terry, and T. Belytschko. Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Computational Mechanics*, 18(3):225–235, 1996.

[OGL13] M.A. Otaduy, C. Garre, and M.C. Lin. Representations and algorithms for force-feedback display. *Proceedings of the IEEE*, 101(9):2068–2080, Sept 2013.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH*, pages 137–146, 1999.

[OL06a] Miguel A Otaduy and Ming C Lin. High fidelity haptic rendering. *Synthesis Lectures on Computer Graphics and Animation*, 1(1):1–112, 2006.

[OL06b] Miguel A Otaduy and Ming C Lin. A modular haptic rendering algorithm for stable and transparent 6-dof manipulation. *IEEE Transactions on Robotics*, 22(4):751–762, 2006.

[PCOS10] N. Pietroni, P. Cignoni, M.A. Otaduy, and R. Scopigno. Solid-texture synthesis: A survey. *Computer Graphics and Applications, IEEE*, 30(4):74–89, July 2010.

[PGCS09] Nico Pietroni, Fabio Ganovelli, Paolo Cignoni, and Roberto Scopigno. Splitting cubes: a fast and robust technique for virtual cutting. *The Visual Computer*, 25(3):227–239, 2009.

[PKA$^+$05] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3):957–964, July 2005.

[PRS07] Tobias Preusser, Martin Rumpf, and Lars Ole Schwen. Finite element simulation of bone microstructures. In *Proceedings of the 14th Workshop on the Finite Element Method in Biomedical Engineering, Biomechanics and Related Fields*, pages 52–66. University of Ulm, July 2007.

[RB86] C.C. Rankin and F.A. Brogan. An element-independent co-rotational procedure for the treatment of large rotations. *ASME J. Pressure Vessel Tchn.*, 108:165–174, 1986.

[RBZ10] T. Rabczuk, S. Bordas, and G. Zi. On three-dimensional modelling of crack growth using partition of unity methods. *Computers & Structures*, 88(23-24):1391–1411, 2010.

[RHM94] Edward K. Rodriguez, Anne Hoger, and Andrew D. Mcculloch. Stress-dependent finite growth in soft elastic tissues. *J. Biomech.*, 27(4):455–467, 1994.

[SAF$^+$06] Chengli Song, Afshin Alijani, Tim Frank, George Hanna, and Alfred Cuschieri. Elasticity of the living abdominal wall in laparoscopic surgery. *Journal of Biomechanics*, 39(3):587 – 591, 2006.

[Sat93] Richard M Satava. Virtual reality surgical simulator: The first steps. *Surgical endoscopy*, 7(3):203–205, 1993.

[SCB01] T. Strouboulis, K. Copps, and I. Babuska. The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190(32-33):4081–4193, 2001.

[SCB04] Kenneth Salisbury, Francois Conti, and Federico Barbagli. Haptic rendering: introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.

[SCOL$^+$04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM Press.

[SDF07] Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 73–80, 2007.

[SGO09] Sara C. Schvartzman, Jorge Gascón, and Miguel A. Otaduy. Bounded normal trees for reduced deformations of triangulated surfaces. In *SCA '09: Symposium on Computer Animation*, pages 75–82, 2009.

[SH12] Jonas Spillmann and Matthias Harders. Robust interactive collision handling between tools and thin volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1241–1254, August 2012.

[She94] Jonathan R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[She02] Jonathan R. Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). Technical report, University of California at Berkeley, 2002.

[SHGS06] D. Steinemann, M. Harders, Markus Gross, and G. Szekely. Hybrid cutting of deformable solids. In *Virtual Reality Conference*, pages 35–42, 2006.

[Si06] Hang Si. *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, 2006. http://tetgen.org.

[SJ01] R. Satherley and M.W. Jones. Hybrid distance field computation. In *Proc. Volume Graphics*, pages 195–209, 2001.

[Sla02] William S Slaughter. *The linearized theory of elasticity*. Springer, 2002.

[SMMB00] N. Sukumar, N. Moës, B. Moran, and T. Belytschko. Extended finite element method for three-dimensional crack modelling. *International Journal for Numerical Methods in Engineering*, 48(11):1549–1570, 2000.

[SOG06] Denis Steinemann, Miguel A. Otaduy, and Markus Gross. Fast arbitrary splitting of deforming objects. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 63–72, 2006.

[SSSH11] M. Seiler, D. Steinemann, J. Spillmann, and M. Harders. Robust interactive cutting based on an adaptive octree simulation mesh. *Vis. Comput.*, 27(6):519–529, 2011.

[SW06] SA Sauter and R. Warnke. Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing*, 77(1):29–55, 2006.

[SWC+08] Lana P Sturm, John A Windsor, Peter H Cosman, Patrick Cregan, Peter J Hewett, and Guy J Maddern. A systematic review of skills transfer after surgical simulation training. *Annals of surgery*, 248(2):166–179, 2008.

[THM+03] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H. Gross. Optimized spatial hashing for collision detection of deformable objects. In *the Vision, Modeling, and Visualization Conference*, pages 47–54, 2003.

[TIHN07] Kenshi Takayama, Takeo Igarashi, Ryo Haraguchi, and Kazuo Nakazawa. A sketch-based interface for modeling myocardial fiber orientation. In *Proc. the 8th international symposium on Smart Graphics*, pages 1–9, 2007.

[TKAN09] George Turkiyyah, Wajih Bou Karam, Zeina Ajami, and Ahmad Nasri. Mesh cutting during real-time physical simulation. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, pages 159–168, 2009.

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, 2005.

[TMFB05] J. Teran, Neil Molino, R. Fedkiw, and R. Bridson. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers*, 21(1):2–18, 2005.

[Tod] Don Todd. b4wind user's guide - trilinear interpolation. `www.grc.nasa.gov/WWW/winddocs/utilities/b4wind_guide/trilinear.html`. [Online; accessed 30-July-2012].

[TOII08] Kenshi Takayama, Makoto Okabe, Takashi Ijiri, and Takeo Igarashi. Lapped solid textures: Filling a model with anisotropic textures. *ACM Trans. Graph.*, 27(3):53:1–53:9, 2008.

[WBG07] Martin Wicke, Mario Botsch, and Markus Gross. A finite element method on convex polyhedra. *Computer Graphics Forum*, 26(3):355–364, 2007.

[WBWD12] Jun Wu, Kai Bürger, Rüdiger Westermann, and Christian Dick. Interactive residual stress modeling for soft tissue simulation. In *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 81–89, 2012.

[WDW11] Jun Wu, Christian Dick, and Rüdiger Westermann. Interactive high-resolution boundary surfaces for deformable bodies with changing topology. In *Proceedings of 8th Workshop on Virtual Reality Interaction and Physical Simulation*, pages 29–38, 2011.

[WDW13] Jun Wu, Christian Dick, and Rüdiger Westermann. Efficient collision detection for composite finite element simulation of cuts in deformable bodies. *The Visual Computer*, 29(6-8):739–749, 2013.

[WH05] Wen Wu and Pheng Ann Heng. An improved scheme of an interactive finite element model for 3d soft-tissue cutting and deformation. *The Visual Computer*, 21(8-10):707–716, 2005.

[WJST14] Yuting Wang, Chenfanfu Jiang, Craig Schroeder, and Joseph Teran. An adaptive virtual node algorithm with robust mesh cutting. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 77–85. The Eurographics Association, 2014.

[WLY04] Burkhard C. Wünsche, Richard Lobb, and Alistair A. Young. The visualization of myocardial strain for the improved analysis of cardiac mechanics. In *Proc. Computer graphics and interactive techniques in Australasia and South East Asia*, pages 90–99, 2004.

[WM03] Ming Wan and W.A McNeely. Quasi-static approach approximation for 6 degrees-of-freedom haptic rendering. In *Visualization, 2003. VIS 2003. IEEE*, pages 257–262, Oct 2003.

[WWD14] Jun Wu, Rüdiger Westermann, and Christian Dick. Real-time haptic cutting of high-resolution soft tissues. *Studies in Health Technology and Informatics (Proc. Medicine Meets Virtual Reality 21)*, 196:469–475, 2014.

[WWWZ10] Jun Wu, Dangxiao Wang, Charlie C. L. Wang, and Yuru Zhang. Toward stable and realistic haptic interaction for tooth preparation simulation. *Journal of Computing and Information Science in Engineering*, 10(2):021007:1–9, 2010.

[ZBS05] Yongjie Zhang, Chandrajit Bajaj, and Bong-Soo Sohn. 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5083–5106, 2005.

[ZDL$^+$11] Guo-Xin Zhang, Song-Pei Du, Yu-Kun Lai, Tianyun Ni, and Shi-Min Hu. Sketch guided solid texturing. *Graphical Models*, 73(3):59 – 73, 2011.

[ZHT07] Eugene Zhang, James Hays, and Greg Turk. Interactive tensor field design and visualization on surfaces. *IEEE Trans. Vis. & Comput. Graph.*, 13(1):94–107, 2007.

[ZP02] Xiaoqiang Zheng and Alex Pang. Volume deformation for tensor visualization. In *Proc. IEEE Visualization*, pages 379–386, 2002.

[ZWP05] Hualiang Zhong, Mark P. Wachowiak, and Terry M. Peters. Adaptive finite element technique for cutting in surgical simulation. In Jr. Robert L. Galloway and Kevin R. Cleary, editors, *Medical Imaging 2005: Visualization, Image-Guided Procedures, and Display*, volume 5744 of *Proc. SPIE*, pages 604–611. 2005.