Topological Changes in Simulations of Deformable Objects

Christoph Joachim Paulus

N° d'ordre: 3686

UNIVERSITÉ DE STRASBOURG

Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

THESIS

presented for the grade of

Docteur de l'Université de Strasbourg

Mention: Informatique

by

Christoph Joachim PAULUS

Topological Changes in Simulations of Deformable Objects

Defended publicly on the **April 3, 2017**

Members of the jury:

M. Hervé Delingette ..... *Reviewer*
Research director at Inria, Sophia Antipolis, France

M. Fabrice Jaillet ..... *Reviewer*
Assistant Professor University of Lyon, France

M. Matthias Harders ..... *Examiner*
Professor at the University of Innsbruck, Austria

M. Jun Wu ..... *Examiner*
Assistant Professor at the University of Delft, The Netherlands

M. David Cazier ..... *Advisor*
Professor at the University of Strasbourg

M. Stéphane Cotin ..... *Advisor*
Research Director at Inria, Strasbourg, France

Abstract

Topological Changes in Simulations of Deformable Objects

Virtual cutting of deformable objects is at the core of many applications in interactive simulation and especially in computational medicine. The ability to simulate surgical cuts, soft tissue tearing or fractures, is essential for augmenting the capabilities of existing or future simulation systems.
In this thesis, we present a new remeshing algorithm based on the finite element method. For tetrahedral elements with linear shape functions, we combined remeshing algorithm with the movement of the nodes to the cutting surface, called *snapping* in the literature. Our approach shows benefits when evaluating the impact of cuts on the number of nodes and the numerical quality of the mesh. When applying our remeshing algorithm to quadratic shape functions, we observe similar results. Due to the curved surfaces of the elements, when using quadratic shape functions, the snapping of nodes entails higher challenges. Thus, to investigate into the snapping, experience has been gathered on triangular shell elements, simulating fractures.
Beyond the simulation of fractures, our remeshing approach for tetrahedral elements is generic enough to support a large variety of applications. In this work, we are the first to present results on the detection of topological changes, such as fractures, tearing and cutting, from a monocular video stream. Examples with highly elastic silicone bands, in-vivo livers and ex-vivo kidneys show the robustness of our detection algorithm, which is combined with the remeshing approach, in different scenarios. Finally, the augmentation of internal organ structures highlights the clinical potential and importance of the conducted work.

Keywords: Cutting, tearing, fracture, topological changes, Quadratic shape functions, shells, augmented reality, deformation, internal structures

Résumé

Changements Topologiques dans des Simulations des Objets Deformable

La découpe virtuelle d'objets déformables est au cœur de nombreuses applications pour la simulation interactive. Nous présentons un nouvel algorithme de remaillage permettant la simulation de découpes avec la méthode des éléments finis. Nous avons combiné notre algorithme avec la méthode du snapping déplaçant les noeuds à la surface de coupe, pour des tétraèdres linéaires. Notre approche permet de maîtriser le nombre de noeuds et de la qualité numérique du maillage durant les coupes. Elle donne des résultats similaires pour les fonctions de forme quadratiques. Dans ce cadre, nous avons évalué le snapping pour simuler la fracture de surfaces triangulées. Nous avons appliqué nos résultats en 3D pour l'assistance aux gestes chirurgicaux, en étant les premiers à présenter des résultats sur la détection de déchirures dans un flux vidéo monoculaire. La robustesse de notre algorithme et l'augmentation des structures internes des organes souligne l'intérêt clinique de notre méthode.

Acknowledgements

"All success has its secrets, all failure its causes." Joachim Kaiser

In this work, inspiring individuals were a great support and encouragement for me, working towards common goals and walking through failures – whatever the cause – *together*. I consider these individuals my secret of success and dedicate the following pages to returning thanks.

I owe thanks to my advisors Stéphane Cotin and David Cazier, that made this scientific work possible, financed by the research institute INRIA and conducted with the university of Strasbourg. I am grateful for your constructive criticism and the investment of time, that did not only shape my career, but as well my person. Stéphane for communicating the passion for research to me, for encouraging exchange and opening instead of closure, for the effort invested into creating a good working environment and for the security of my contract at the end of this work. David for the structured way of thinking, the mathematical approach to problems, the possibility to teach at the IUT Haguenau and for your valuable personal advices.

I would like to express my gratitude to Matthias Harders, Jun Wu, Hervé Delinguette and Fabrice Jaillet for having accepted to attend my PhD defense as jury. Particularly I thank Hervé Delinguette and Fabrice Jaillet for reading my thesis and performing the review, despite their busy schedule.

I am indebted to Stefan Suwelack, the advisor of my diploma thesis, for his advises on the writing of the extended finite element paper, the quadratic tetrahedral elements, the supervision of interns and for the always positive attitude communicated in our calls. For the experiments conducted with ex-vivo organs, I would like to thank Seong-Ho and Renato, that showed an admirable patience in redoing the experiments if something went wrong.

In the course of my doctorate, I had the chance to guide several internship candidates: I worked with Sabrina Izcovich on the cutting of quadratic tetrahedral elements, with Andrea Mendizabal on the smoothed finite element method, Arnaud Bonnet on the fracture of curved shells and Roland Maier on the immersed boundary method. Not having led interns before, this experience was particularly enriching for me, I hope you enjoyed the time and learned as much as I did. Particularly for you, Roland, I was grateful to have an exchange on a very similar level, with all your feedback on my PhD thesis. Similarly I liked my teaching activities, first at the IUT Haguenau and then at the university of Strasbourg. For the time teaching at the university

of Strasbourg, I would like to include a special note of thanks to Stéphane Marseglia, who shared his experience teaching Maple and made the preparation and correction of exams much easier. For teaching the masters course in C++, I appreciated the advises of my colleagues Etienne, Marc and Bruno, that helped me to provide examples for the students and prevented several mistakes from my side.

I truly enjoyed being part of the research team Mimesis, I liked that people are open to support each other at (almost) any time and the relaxing team athmosphere often relieved the stress before deadlines. In our team, we have important support from Isabelle Blanchard and Anne Aubry for administrative questions. Thank you Isabelle for doing your part of the work even when I sometimes struggled to keep the deadlines you set. A strength of our team is the constant support when writing papers. Stéphane, David, Lionel Untereiner, Hadrien Courtecuisse, Nazim Haouchine, Bruno Marques and Roland Maier, thank you for not letting me work alone the nights before the deadlines. Lionel, I appreciate, that you always encourage me to take my breaks and to recover from the deadline sprints, acknowledging and anticipating the limits of my capacities.
I would as well like to recognize the time investment of Etienne Schmitt, Marc Legendre and Bruno teaching me a lot about C++ and programming in general, while Hugo and Fred helped me with many questions related to the usage of the open source framework Sofa. I particularly thank Bruno, who assisted me (almost) from the beginning until the end of my doctorate with his knowledge in computer science, installing libraries, operating systems and much more. Know that, I really enjoyed all the lunch breaks we had and the discussions, even if most of our fun ideas will probably never be realized, well, since we already will make enough money with a part of them ;) Marc, thank you for your patience when correcting my french, I hope you will continue enjoying your PhD, let me know if I can help you with anything. Hugo, thank you for having introduced me into the team, you were the link between Karlsruhe and Strasbourg and I might have not started the PhD without having had such an honest ”insider“. Remi, thank you for supporting for continuing the work of Arnaud, I like your calmness and the good combination of leaving me the possibility to work together when I want, but still advancing alone and being able to work harder if necessary.
Beyond that, I am thankful for the discussions on the theoretical background of this work with Rosalie Plantefeve, Jean-Nicolas Brunet, Roland and Igor Peterlik. Igor, thank you for your advises when being challenged with moral questions around the work and publications. Rosalie, it was awesome to have you as lab-mate, thank you that you never got bored when I was talking about rowing, thank you for having an open ear for all the other subjects. I really liked having you around with your joyful nature! Jean-Nicolas, thank you for

the weekends you spent at work as well, for the numerous breaks and for never having rejected me when I was coming to your office ranting.

I also would like to recognize the persons, that I haven't mentioned above, but played an important role in my three years doctorate: Raffaella, Myriam, Jaime, Yinoussa, Guillaume, Madame Dannhauser, François Jourdes and many others.

Finally, I would like to address my deepest thanks to my friends, particularly Pierrick and Marie, my siblings with their partners and most importantly my parents Ina and Ludwig. Thank you, for encouraging me to take my path, but always letting me know that it is my person that matters, not my work. Thank you, for helping me to stick to my faith, that kept me upright when nobody else could have helped.

Contents

Introduction

1 Motivation, challenges and contributions

In the last decades, minimally invasive interventions replaced more and more open surgery, to alleviate the impact of the surgical act on the patient.

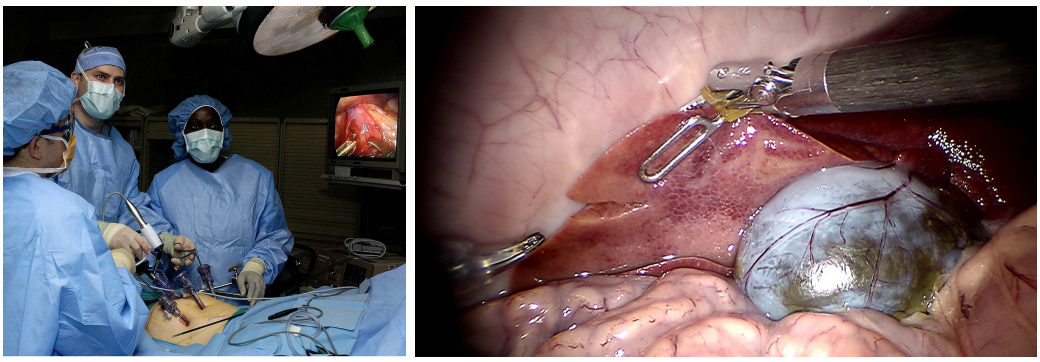


Figure 1: Laparoscopic surgery: Setup *(left)*, per-operative view (*right*)

A minimally invasive surgery for the digestive and pelvic tract is called *laparoscopy* and was first applied in 1987. This recent technique uses the insufflation of carbon dioxide inside the abdominal cavity to create a working space around the organs. Trocars are placed in small incisions inside the

abdominal wall, to allow the insertion of a camera and some elongated surgical instruments. Then the surgical intervention is performed with an indirect view using the laparoscopic camera. Preventing open surgery with large incisions, this approach reduces the post-operative morbidity, the pain, the size of the scars and thus results in a shorter recovery time [121].

Similarly to this technique, trocars serve for an insertion of a light source and a micro surgical instrument into the eye. With these tools, surgical interventions treating a damaged retina can be performed by looking through the sclera. In this context, due to the pressure inside of the eye, open surgery is out of question, which underlines the importance of the minimally invasive approach.

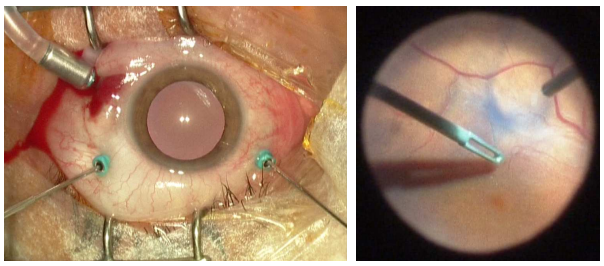


Figure 2: Minimally invasive surgery on the eye: *left* the trocars are set for the pressure control, the light and one instrument, *right:* removal of the epiretinal membrane

The positive effects of minimally invasive interventions on the patients go along with challenging conditions for the surgeons. These include a reduced field of view, the loss of direct manipulations and restricted movements of the surgical instruments by the position of the trocars. For laparoscopy, surgeons additionally need to be able to apply pre-operative information, retrieved by imaging techniques like CT or MRI, to the intraoperative setting that includes the movement and deformation of the targeted tissues. Moreover, due to the camera view that can differ from a direct view a surgeon might have in open surgery, the hand-eye coordination is unnatural and difficult. To conclude, surgeons struggle to locate tumors, to define resection margins and the minimally invasive interventions result in an increased operating time [121].

Effects of these challenges include manipulation errors resulting in an increased risk of tissue damage. In the United States, similar medical errors occur in around 3 percent of hospitalizations and yield costs estimated between 17 and 29 billion US dollars [49]. Still, extensive training of the surgeons is rarely performed to prevent dangerous situations for the patient and the elevated costs. Similar to the developement for the flying industry in the past, simulations and computer assistance could be used for the training, planning, support

and guidance of surgeons. For neurosurgery and orthopedic surgery, computer assisted surgery (CAS) methods are already well known as navigation systems and show an improved patient outcome [35, 130]. With augmented reality (AR) systems, information is overlaid on the view of the surgical field, giving a virtual transparency to a patient for surgical guidance [67]. To provide the additional information, automatic augmented reality is split into two steps: the initial alignment of organs to the image [90], also called registration, and the correct following of deformations induced by manipulations [40]. Algorithms for the guidance of surgeons rely on (pre-operative) imaging systems and intraoperative sensor systems, e.g. ultrasound, the laparoscopic camera, the microscope used for the intervention on the eye. Training systems can't rely on per-operative information and need to provide realistic models for the manipulations performed in a surgical intervention. There are only a few systems existing, that go towards the simulation of a complete procedure. One of the most advanced system is currently developed for the cataract surgery and uses, beside high-fidelity haptical devices and visual models, a physics-based approach to simulate manipulations on the eye.

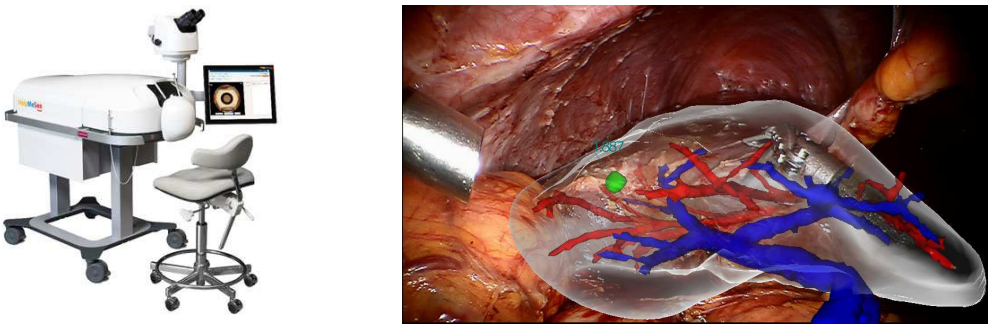


Figure 3: Computer assistance for the training of cataract surgery on the eye [109] *(left)* and the abdominal cavity showing the internal structures of the liver as augmented reality [55] *(right)*

Physics-based approaches are very promising in terms of realism, but challenging since many different phenomena need to be modeled. In laparoscopic surgery, organ manipulations include large deformations, cutting and cauterization. When removing the membrane in eye surgery, it undergoes strong deformations and is torn apart to be removed. Due to the curved surfaces of the organs and the eye, physics-based methods need to provide accurate geometrical models, preferable with curved boundaries, undergoing large deformations [118]. Moreover, the ability to simulate cuts, soft tissue tearing or micro-fractures, is essential in a surgical context. For the training and the per-operative application, proposed solutions need to run stable in real-time and with high accuracy. Combining all the points mentioned in this chapter is particularly challenging and makes this topic an active area of research.

Most solutions result in compromises as a higher accuracy directly reduces the capability to simulate in real-time.

Virtual cutting of deformable objects is at the core of other applications, such as interactive sculpting and thus proposed techniques can be applied in a broader context. Note, that topological changes like cuts, tearing or fractures usually are performed in two steps: the detection of the separation and the update of the mechanical model. Fractures, for example, use a fracture model to detect the location where a crack appears and then propagate the crack to the model.

This thesis focuses on the propagation of topological changes in a mechanical model. Hereby, we argue the correct choice of the mechanical model, to preserve real-time capabilities through the complete phase of the simulation. Different detections of topological changes, based on fracture criteria and on informations from an image are discussed. Issues in the stability and the real-time aspect of the simulation are addressed.

Many physics-based methods for the simulation of deformations exist in the literature, e.g. mass-spring methods, meshless methods and the finite element method. In their conventional form, these methods can't handle topological changes like cuts and thus improvements or adaptations were proposed. This work discusses the different propositions and we present arguments for our choice: the finite element method.

We present the following contributions:

−A robust and efficient algorithm to cut linear tetrahedra, based on the snapping of nodes and a novel re-meshing techniques
−An application to augmented reality during surgery showing topological changes overlaid on an organ that is cut, torn or fractured. Internal structures are overlaid onto the intraoperative view.
−An extention of our re-meshing algorithm to quadratic tetrahedra to obtain smooth curved surfaces
−Preliminary results on the fracture of quadratic shell elements

2 Outline

This manuscript is organized around the publications (see section 3 for an overview) and the future work of this three years doctorate. The aim of this thesis is on the one hand to give further details, explanations and discussions related to these publications. This will allow to better understand and reproduce the obtained results, and thus to build other work upon it. On the other hand, a thorough description of the fundamentals allows to better analyse the results and choices made.

The thesis is devided into four parts: The **first part** discusses the background: the **real-time simulation of soft bodies**.

Chapter 1 gives a short insight into the state of the art related to real-time simulation of soft bodies. Then we point at all the approximations that are made for the sake of simulating in real-time.

Chapter 2 introduces the main notions to explain object deformation in the real world: forces yield to strain, strain is related to the stress by the constitutive equation and finally, balance equations help formulate a problem as a differential equation, the strong formulation.

Chapter 3 transforms the differential equation to an integral equation, the weak formulation. Then discretizations in space and time yield a non-linear system of equations, that is solved using iterative methods relying on different types of linear system solvers. Numerical aspects important for topological changes are examined and accuracy and convergence are discussed.

The **second part** presents the main results considering the algorithms applied for the **topological changes** of the mechanical model.

Chapter 4 outlines previous attempts to solve the problem of incorporating topological changes inside of a mechanical model. The finite element problem is formulated particularly adapted to the cutting problem. As the conventional finite element method can't deal with cuts, we present a solution that extends the function space to allow for cuts inside of the elements. Then we lay the foundations for other approaches, that introduce new elements with boundaries aligned to the cut.

Chapter 5 presents a cutting algorithm used for linear volumetric elements, that has been published at a conference and in a journal. The chapter underlines the real-time compability and the stability of the approach.

Chapter 6 extends the linear cutting algoritm to quadratic volumetric elements and discusses the major advantages and disadvantages. Then we present our work on the fracture of quadratic shell elements.

The **third part** is dedicated to **augmented reality**, the main application of our work.

Chapter 7 introduces the main principles applied for augmented reality and describes the possibilities to assess the errors. Moreover, it describes the link of the mechanical object to the image and then the detection of topological changes. We first present the results on silicone data and then the application of our technique in the surgical setting: on *ex vivo* and *in vivo* porcine examples. The accuracy of internal structures is compared to the state of the arts and shows a major improvement.

In the **fourth part** we summarize the work and point to potentially interesting directions for the future.

3 List of publications

Journal papers

- [82] Christoph Joachim Paulus, Nazim Haouchine, Seong-Ho Kong, Renato Vianna Soares, David Cazier, and Stéphane Cotin. Handling topological changes during elastic registration: Application to augmented reality in laparoscopic surgery. *International Journal of Computer Assisted Radiology and Surgery (IJCARS)*, 2016
- [80] Christoph J. Paulus, Lionel Untereiner, Hadrien Courtecuisse, Stéphane Cotin, and David Cazier. Virtual cutting of deformable objects based on efficient topological operations. *The Visual Computer*, 31(6-8):831–841, 2015

Conference papers

- [79] Christoph J Paulus, Nazim Haouchine, David Cazier, and Stéphane Cotin. Surgical augmented reality with topological changes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015
- [78] Christoph J Paulus, Nazim Haouchine, David Cazier, and Stéphane Cotin. Augmented reality during cutting and tearing of deformable objects. In *Mixed and Augmented Reality (ISMAR)*, pages 54–59, 2015
- [81] Christoph J. Paulus, Lionel Untereiner, Hadrien Courtecuisse, Stéphane Cotin, and David Cazier. Virtual cutting of deformable objects based on efficient topological operations. In *Computer Graphics International (CGI)*, 2015
- [58] Andrea Mendizabal, Rémi Bessard Duparc, Huu Phuoc Bui, Christoph J. Paulus, Igor Peterlik, and Stéphane Cotin. Towards a face-based smoothed finite element method for real-time simulation of the brain shift. 2017

Other

- [77] Christoph Paulus, Stefan Suwelack, Nicolai Schoch, Stefanie Speidel, Rüdiger Dillmann, and Vincent Heuveline. Simulation of complex cuts in soft tissue with the extended finite element method (x-fem). *Preprint Series of the Engineering Mathematics and Computing Lab*, (02), 2014

Part I

Real-time simulation of soft bodies

Chapter 1

Introduction

Contents

1.1 Deformable models

The main motivation of this work is to simulate the physical behaviour of a deformable object undergoing topological changes. For the application in surgeries, approaches need to be accurate, stable, account for topological changes like cuts and deliver solutions at an interactive frame rate. Responding to the versatility of physical phenomena, researchers in computer graphics provided various methods, which address and focus on particular requirements and challenges. In the following we briefly discuss several of these methods, pointing towards the features relevant in our context. For a broader overview of physically based methods, please refer to [66].

One of the most intuitive and simplest approaches, the mass-springs method, replaces the to be simulated object by points, which have a mass and are connected by springs. With this discrete model, deformations are calculated based on Newton's second law. Topological changes like cuts can easily be incorporated into the representation by deleting the springs that lie on the path of the cut. While this method allows for real-time simulations, the main drawback is the strong dependency of the deformation on the chosen connectivities between the mass points. Due to this dependency, mass springs may result in an unrealistic behaviour, since material behaviours are difficult to represent correctly.

In contrary to the mass-spring method, mesh-free methods are based on the theory of continuum mechanics, resulting in an increased reality of the simulation. The object is replaced by points with shape functions, that are weighted to have a limited domain of influence. Real-time computation rates are accomplished for various different physical phenomena, including melting and very strong deformations, wherefore the method often is applied in the computer game industry. Topological changes are taken account of by cropping the domain of influence of the shape functions using different criteria. The drawbacks and challenges coming with the use of the method include the need to update neighborhood information of the nodes every time step; physically plausible, but potentially physically incorrect deformations; and the necessity of an additional mechanism to create and maintain object surface, particularly when topological changes occur. We consider the main difficulty in the correct choice of the integration of the shape functions, particularly since an exact integration comes along with a deteriorated efficiency.

The finite element method (FEM) discretizes an object using nodes connected by elements. Positions inside the elements are the interpolation of the element nodes and for each element continuum mechanics yield a stiffness matrix that relates nodal displacements to nodal forces. In order to use the conventional form of the FEM for cuts, nodes either need to be moved to or introduced on the cutting surface. Besides the conventional form, other adapted approaches of the finite element method exist that account for topological changes. FEM has many benefits such as accuracy and robustness, which make it relevant in our context and the method of our choice. However, it also has important limitations: computation time and sensitivity to the mesh resolution and mesh quality are the most relevant for real-time simulations. These points are summarized in the following subsection, while further details are presented in the next two chapters.

1.2 Summary of applied approximations and assumptions

Simulations in real-time only become possible with approximations, in many cases linearizations. Moreover, assumptions need to be made, to simplify the considerations. The correct placement of the approximations is key, to address the relevant parts for the computation time. Note, that we consider small scale problems, that consist of less than 10000, but rather close to 1000 degrees of freedom. We chose this size of our problems, such that, without parallelization, real-time computation times can be achieved. Detailed theories on the approximations however, are in most of the cases also applicable to greater systems with potential adaptions and improvements.

While this chapter serves as a summary of the approximations and assumptions, details are given in the referenced sections and chapters in brackets:

- Isothermic problem, i.e. there is only the deformation variable (see 2)
- Macroscopic scale, i.e. few parameters quantify a set of particles
- The object is stress free in the initial state (see 2.1.1 and 3.2)
- The object has no speed at the beginning (see 2.1.1)
- Conservation of mass (see 2.1.2)
- Balance of linear and angular momentum (see 2.1.2,2)
- Cauchy's stress theorem, to relate stress and traction tensor (see 2.1.3)
- Balance of energy (see 2.2.1)
- Corotated strain (see 2.2.2)
- Hyperelastic material behaviour (see 2.2), that allows deriving isotropic material behaviour and a linearized Strain Stress relation (see 2.2.4)
- Spatial discretization (see 3.2)
  - Discretization by elements
  - Interpolation using polynomial shape functions
  - Linearization of internal energy resulting in the stiffness matrix
  - Approximation of the integral using Gauss integration
- Boundary Conditions (see end of 2.1.3)
  - Dead loads - independent of the deformation
  - Propagated to the nodes (see 3.2)
  - In most cases time independent
  - Dirichlet boundary conditions are in most cases 0
- Time discretization scheme (see 3.3)
- Only one step of the Newton-Raphson method is used (see 3.3)
- Errors due to the solver of the linear equation system (see 2.1.1)
  - Round-off and truncation errors for direct methods (see 2.1.1)
  - Instabilities for a high condition number (see 2.1.1)

1.3 Notation

In this section, we briefly describe the notation used in our work. Some of the terms in the description might not be familiar to the reader and are therefore explained in the subsequent chapters. The standard notation conventions are adapted to our work, avoiding repetitive usage of the same notation. Hereby, we mainly follow a combination of the notations proposed by [42, 116, 125].

Our work operates on the vector space of real numbers $\mathbb{R}$ and in our applications we also use the space of natural numbers $\mathbb{N} = \{..., -2, -1, 0, 1, 2, ...\}$. The current time is notated as $t \in \mathbb{R}_+$ and a time step as Δt. We use different sets, such as $\Omega, \Omega(t), B, B(t), X, Y, ... \subset \mathbb{R}^3$, but also $X, Y \subset \mathbb{N}^2$. For a set Ω

we define the interior Ω^0, the closed set $\overline{\Omega}$ and the boundary $\partial\Omega = \overline{\Omega}\backslash\Omega^0$. If the set X has a finite number of elements, the number of the elements inside of the set is $n(X)$. We use the space of continuous functions $C(\Omega)$, of n-times differentiable functions $C^n(\Omega)$ and of polynoms of degree n denoted as $P_n(\Omega)$.

Bold letters are vectors or tensors potentially of higher orders (up to order four) operating in most cases $\mathbb{R}^3$. Upper case letters, such as $\mathbf{X}, \boldsymbol{U}, V, ...$, are used for the reference (or material) configuration, while lower case letters, such as $\mathbf{x}, \boldsymbol{u}, v, ...$, are used for the current (or spatial) configuration.

An *italic or tilted* variable, such as $\boldsymbol{X}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{V}, \boldsymbol{v}, ...$, is usually a function dependent on (a point in) $\mathbb{R}^3$. Straight variables, such as $\mathbf{X}, \mathbf{x}, \mathbf{u}, ...$, are independent of a point.

Partial time derivatives of functions are written as $\frac{\partial}{\partial t}$ and $\frac{\mathrm{D}}{\mathrm{D}t}$ (see subsection 2.1.1). Derivatives in space are expressed e.g. as $\nabla_{\mathbf{X}}\boldsymbol{x} = d\boldsymbol{x}/d\mathbf{X}$. Note, that $\nabla_{\mathbf{X}}, \nabla_{\mathbf{x}} \in \mathbb{R}^{1\times 3}$, i.e. derivatives can be considered as a vector vector or a vector matrix product.

Distances between two entities X, Y are notated as $d(X, Y)$ with subscripts indicating which distance measure has been used.

Indices are expressed with the mathematical notation, starting with the first index 1. For directions, we use i, j and the subscript e, e.g. in $\boldsymbol{x}_\mathrm{e}, \boldsymbol{X}_\mathrm{e}, \boldsymbol{U}_\mathrm{e}$, $\boldsymbol{F}_\mathrm{e}, ...$, indicates for the finite element method a variable, which is related to an element with the element id e. The connectivity, i.e. the nodes related to an element, is written as c_e, where $k = c_\mathrm{e}(l)$ is a global node index using the local node index l. To differentiate between different cell types, the index e is used for edges, f for faces and v for volumes. Cells around other cells, such as volumes around an edge are denoted as $v(e)$. In an element, at this point we exemplarily use an edge with the id e, so called local shape functions $N_{\triangle_e,l}$ and a reference domain $\triangle_e$ interpolate between the element nodes $\mathbf{P}_{c_e(l)}$ and help to describe the region of the particular element

$$\Omega_e \coloneqq \left\{ \boldsymbol{x} = \sum_{l=1}^{n(c_e)} \mathbf{P}_{c_e(l)} N_{\triangle_e,l}(\boldsymbol{\xi}) | \boldsymbol{\xi} \in \triangle_e \right\} \tag{1.1}$$

Using the size h of the smallest element in the finite element mesh, approximative functions defined for the finite element method are written with a subscript h, e.g. $\boldsymbol{U}_h$.

For the discretization of the differential equations we notate matrices with capital letters, such as $\mathbb{M}, \mathbb{K}$, and vectors with lower case letters, such as $\mathbb{u}, \mathbb{f}^{int}$.

Chapter 2

Introduction to continuum mechanics

Contents

In order to construct a mechanical model, we explain the physical phenoma considered in this work using the theory of continuum mechanics. This theory considers an object on a macroscopic scale, i.e. a lot of particles are replaced by a few parameters quantifying the set of particles. For that, we first discuss the motion and deformations of an object in different configurations. We state the important balancing principles for this work, introducing different representations of the stress. With appropriate boundary conditions, these allow to formulate a boundary value problem. Then we discuss how the stress can be related to the deformation using the concept of strain, that simplifies the analysis. At the end of this chapter, constitutive equations are discussed closing this chapter on the continuous formulation of the problem. Then, in the following chapter, we address the discretized formulation.

In this work we solve isothermal problems, which describe the material behaviour only using the displacement or a position. Thus, thermal effects e.g. the heating due to compression, are not considered. This consideration stands in contrast to e.g. adiabatic problems, that include thermal effects, but do not dissipate heat over the boundary of the problem. Moreover, we assume *deterministic* behaviour, i.e. examples can be repeated with the same outcome. We restrict the behaviour to material functions that are only pointwise dependent on the deformation gradient, which is also called *local action* principle. Based on this principle equations can be stated without working on domains or with integrals. As the stress inside an object should not change while being rotated or translated, we assume an *invariance to rigid body motions* of the material functions.

2.1 Displacements, deformation and stress

In the derivations of this section, we follow closely the derivations of [42, 116], but were also inspired by [48, 125].

2.1.1 Kinematics

We consider a continuum body B, that moves and deforms in space, changing its configuration $B(t)$ dependent on the time t. Hereby, B is the reference (or undeformed) configuration and $B(t)$ is the current (or deformed) configuration. In this work, the reference configuration is the initial configuration at time $t = 0$. Points in the reference configurations are denoted as $\mathbf{X}$, while points in the current configurations are written as $\mathbf{x}$. The components of these vectors are called material (or referential) coordinates for $\mathbf{X}$ and spatial (or current) for $\mathbf{x}$, both using the same coordinate system oriented using the right-hand rule. We assume, that there exists a uniquely inversible vector field χ, called motion, that relates each position in the reference to a position in the current configuration:

$$\boldsymbol{x} \coloneqq \boldsymbol{x}(\mathbf{X}, t) = \chi(\mathbf{X}, t) \tag{2.1}$$

$$\chi : B \to B(t) =: \chi(B, t) \tag{2.2}$$

In the following we consider regions of this body $B \supset \Omega(t) = \chi(\Omega, t) \subset B(t)$, that do not necessarily fill the complete body. With the inverse of the vectorfield, the material coordinates of each spatial coordinate can be retrieved:

$$\boldsymbol{X}(\mathbf{x}, t) = \chi^{-1}(\mathbf{x}, t) \tag{2.3}$$

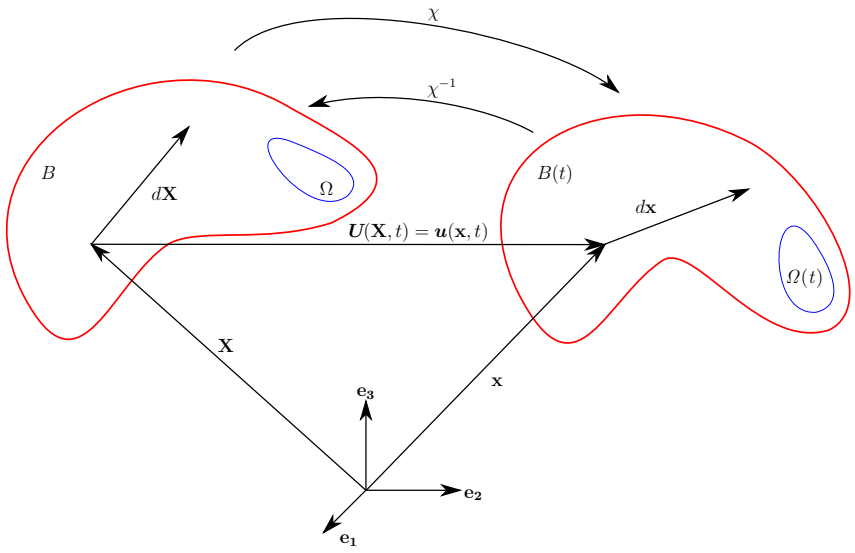


Figure 2.1: Motion of a continuum body with deformation of the object

In the literature, describing a system with the material coordinates is referred to as the *Langrangian* description, while the spatial coordinates are part of the *Eulerian* description. The displacement field can be written in lagrangian

$$\boldsymbol{U}(\mathbf{X},t) \coloneqq \boldsymbol{x}(\mathbf{X},t) - \mathbf{X} \tag{2.4}$$

and in eulerian form:

$$\boldsymbol{u}(\mathbf{x},t) \coloneqq \mathbf{x} - \mathbf{X}(\mathbf{x},t) = \boldsymbol{U}(\chi^{-1}(\mathbf{x},t),t) \tag{2.5}$$

The last equation shows, that the two forms have the same values, but are dependent on different arguments. Note, due to the assumed equality between the undeformed and the initial configuration, the displacement vanishes for $t = 0$.

With the time derivatives, we can write the velocity and the acceleration similarly with lower and upper case letters that depend on the arguments:

$$\boldsymbol{V}(\mathbf{X},t) \coloneqq \frac{\partial \boldsymbol{x}(\mathbf{X},t)}{\partial t} = \boldsymbol{V}(\boldsymbol{x}(\mathbf{X},t),t) =: \boldsymbol{v}(\mathbf{x},t) \tag{2.6}$$

$$\boldsymbol{A}(\mathbf{X},t) \coloneqq \frac{\partial^2 \boldsymbol{x}(\mathbf{X},t)}{\partial t^2} = \boldsymbol{A}(\boldsymbol{x}(\mathbf{X},t),t) =: \boldsymbol{a}(\mathbf{x},t) \tag{2.7}$$

When a spatial vector field depends on spatial coordinates, we use the notation $\frac{\mathrm{D}}{\mathrm{D}t}$ to relate to the material time derivative. Then, using the chain law we obtain:

$$\frac{\mathrm{D}\boldsymbol{v}(\mathbf{x},t)}{\mathrm{D}t} = \left(\frac{\partial \boldsymbol{v}(\chi(\mathbf{X},t),t)}{\partial t}\right)_{\mathbf{X}=\chi^{-1}(\mathbf{x},t)} = \frac{\partial \boldsymbol{v}(\mathbf{x},t)}{\partial t} + \left(\nabla_{\mathbf{x}} \boldsymbol{v}(\mathbf{x},t)\right) \boldsymbol{v}(\mathbf{x},t) \tag{2.8}$$

where the first term $\frac{\partial \boldsymbol{v}(\mathbf{x},t)}{\partial t}$ is the spatial time derivative and the second term $(\nabla_{\mathbf{x}} \boldsymbol{v}(\mathbf{x},t))\, \boldsymbol{v}(\mathbf{x},t)$ is the convective rate of change of $\boldsymbol{v}$. For spatial vector fields, the material derivative and partial derivative by the time are the same

$$\frac{\mathrm{D}\boldsymbol{x}(\mathbf{X},t)}{\mathrm{D}t} = \boldsymbol{V}(\mathbf{X},t), \quad \frac{\mathrm{D}\boldsymbol{V}(\mathbf{X},t)}{\mathrm{D}t} = \boldsymbol{A}(\mathbf{X},t) \tag{2.9}$$

And, thus the time derivatives of the displacement are

$$\frac{\mathrm{D}\boldsymbol{U}}{\mathrm{D}t} = \frac{\partial \boldsymbol{U}}{\partial t} = \boldsymbol{V}, \quad \frac{\mathrm{D}^2\boldsymbol{U}}{\mathrm{D}t^2} = \frac{\partial^2 \boldsymbol{U}}{\partial t^2} = \frac{\partial \boldsymbol{V}}{\partial t} = \boldsymbol{A} \tag{2.10}$$

Besides the quantification of the motion and its time derivatives, we characterize the behaviour of the motion in a neighborhood of a point. For that, we use the deformation gradient

$$\boldsymbol{F}(\mathbf{X},t) \coloneqq \frac{d\mathbf{x}}{d\mathbf{X}} = \nabla_{\mathbf{X}} \boldsymbol{x} = \nabla_{\mathbf{X}} (\boldsymbol{U} + \boldsymbol{X}) = \nabla_{\mathbf{X}} \boldsymbol{U} + \mathbf{I} \in \mathbb{R}^{3\times 3} \tag{2.11}$$

which relates a material line segment $d\mathbf{X}$ starting at point $\mathbf{X}$ to a spatial line segment $d\mathbf{x}$ starting at point $\boldsymbol{x}(\mathbf{X},t)$ as a linear transformation

$$d\mathbf{x} = \boldsymbol{F} d\mathbf{X} \tag{2.12}$$

Similarly, for three linearly independent positively oriented material line segments $d\mathbf{X}, d\mathbf{Y}, d\mathbf{Z}$ at the point $\mathbf{X}$, one can identify the spatial line segments $d\mathbf{x} = \boldsymbol{F} d\mathbf{X}, d\mathbf{y} = \boldsymbol{F} d\mathbf{Y}, d\mathbf{z} = \boldsymbol{F} d\mathbf{Z}$ at the point $\mathbf{x} = \boldsymbol{x}(\mathbf{X},t)$. The volume of the parallelepiped between the line segments is

$$dV = (d\mathbf{X} \times d\mathbf{Y}) \cdot d\mathbf{Z} = \det(d\mathbf{X}, d\mathbf{Y}, d\mathbf{Z}) \tag{2.13}$$

where $(d\mathbf{X}, d\mathbf{Y}, d\mathbf{Z})$ is a matrix with $d\mathbf{X}, d\mathbf{Y}$ and $d\mathbf{Z}$ in the columns. Then, the deformation gradient allows to link the volumes of the parallelepipeds in the two configurations

$$dv = \underbrace{\det(d\mathbf{x}, d\mathbf{y}, d\mathbf{z})}_{=\det(\boldsymbol{F}d\mathbf{X}, \boldsymbol{F}d\mathbf{Y}, \boldsymbol{F}d\mathbf{Z})} = \det(\boldsymbol{F}) \det(d\mathbf{X}, d\mathbf{Y}, d\mathbf{Z}) =: J dV \tag{2.14}$$

Due to the invertibility of $\boldsymbol{F}$ and the positivity of dv and dV, the jacobian J is positive. Thus

$$J(\mathbf{X},t) \coloneqq \det(\boldsymbol{F}(\mathbf{X},t)) = |J(\mathbf{X},t)| = |\det(\nabla_{\mathbf{X}} \chi(\mathbf{X},t))| \tag{2.15}$$

and for incompressible materials, the jacobian equals 1 describing the volume preservation (also called isochoric behaviour). The jacobian relates the vector of infinitesimally small areas in the material $dA\boldsymbol{N}$ with the spatial configuration $da\boldsymbol{n}$:

$$J dA \boldsymbol{N} \cdot d\mathbf{X} = J dV \overset{(2.14)}{=} dv = da\boldsymbol{n} \cdot d\mathbf{x} = da\boldsymbol{n} \cdot \boldsymbol{F} d\mathbf{X} = \boldsymbol{F}^T da\boldsymbol{n} \cdot d\mathbf{X} \tag{2.16}$$

And with the arbitrary choice of $d\mathbf{X}$ we obtain *Nanson's formula*

$$da\boldsymbol{n} = J \boldsymbol{F}^{-T} \boldsymbol{N} dA \tag{2.17}$$

2.1.2 Balance principles

In continuum mechanics, the conservation of mass, the balance of linear and angular momentum and the balance of energy are the fundamental balance principles. In this subsection, we describe the conservation of mass and the balance of linear momentum, while the balance of angular momentum is briefly mentioned in section 2 and the balance of mechanical energy in subsection 2.2.1.

Every continuum possesses a mass, which can't be produced or destroyed and is assumed to stay the same during a motion

$$M(\Omega) = m(\Omega(t)) = \text{const.} > 0 \tag{2.18}$$

Since the mass is independent of the occupied region, the equation in differential form gives

$$dM(\mathbf{X}) = dm(\mathbf{x}, t) \tag{2.19}$$

with infinitessimal mass elements dM and dm. Defining the material densities in material P and spatial ρ form, this equation becomes

$$P(\mathbf{X})dV = dM(\mathbf{X}) = \rho(\mathbf{x}, t)dm(\mathbf{x}, t) = \rho(\mathbf{x}, t)dv \tag{2.20}$$

and with the integration over the domains we obtain the *conservation of mass* stated as

$$\int_{\Omega(t)} \rho(\mathbf{x}, t)dv = \int_{\Omega} P(\mathbf{X})dV = \text{const.} > 0 \tag{2.21}$$

Note, when considering equations (2.14), (2.20) the motion allows relating the two representations of the densities

$$P(\mathbf{X}) = \rho\left(\chi(\mathbf{X}, t), t\right) J(\mathbf{X}, t) \tag{2.22}$$

which is called the *continuity mass equation* in the literature. This result can as well be obtained using integration by substitution, with the transformation χ between the two regions.

The linear momentum in the inital and current configuration is given by

$$\mathbf{L}(t) \coloneqq \int_{\Omega(t)} \rho(\mathbf{x}, t)\boldsymbol{v}(\mathbf{x}, t)dv = \int_{\Omega} P(\mathbf{X})\boldsymbol{V}(\mathbf{X}, t)dV \tag{2.23}$$

And the balance of linear momentum reads *the change of linear momentum in time is equal to the sum of all external forces acting on body B* and thus we can write

$$\mathbf{F} = \frac{\mathrm{D}\mathbf{L}}{\mathrm{D}t} = \int_{\Omega(t)} \rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} dv = \int_{\Omega} P \frac{\mathrm{D}\boldsymbol{V}}{\mathrm{D}t} dV \tag{2.24}$$

In this equation, the continuity mass equation (2.22) allows to move the material time derivative into the integral, without being applied to the time dependent density ρ. To obtain this result several non-trivial steps have to be performed, which are not stated at this point. For further details, please refer to [42], page 140.

Our object is subject to time-varying and space-varying forces: the body forces $\boldsymbol{b} = \rho\mathbf{g}$ and surface forces expressed as the traction $\boldsymbol{t}$. Then the external force $\mathbf{F}(t)$ is given by the integral over $\Omega(t)$ and its surface $\partial\Omega(t)$:

$$\mathbf{F} = \int_{\partial\Omega(t)} \boldsymbol{t} da + \int_{\Omega(t)} \boldsymbol{b} dv \tag{2.25}$$

Thus the linear momentum applied to the continuum mechanics can be written as

$$\int_{\partial\Omega(t)} \boldsymbol{t} da + \int_{\Omega(t)} \boldsymbol{b} dv = \int_{\Omega(t)} \rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} dv \tag{2.26}$$

2.1.3 Cauchy's equation of motion

With Cauchy's stress theorem, the Cauchy traction vector $\boldsymbol{t}(\mathbf{x}, t, \boldsymbol{n})$ exerted on the outward normal $\boldsymbol{n}(\mathbf{x}, t)$, can be related to Cauchy's (or true) stress tensor

$$\boldsymbol{t}(\mathbf{x}, t, \boldsymbol{n}) = (\boldsymbol{t}_1\ \boldsymbol{t}_2\ \boldsymbol{t}_3) \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \boldsymbol{\sigma}(\mathbf{x}, t)\boldsymbol{n}(\mathbf{x}, t) \tag{2.27}$$

And for $\mathbf{x} \in \partial\Omega(t)$ with the normal $\boldsymbol{n}$ of the boundary, we can write:

$$\boldsymbol{t}(\mathbf{x}, t, \partial\Omega(t)) := \boldsymbol{t}(\mathbf{x}, t, \boldsymbol{n}) \tag{2.28}$$

Similarly, we introduce the the Piola-Kirchhoff traction vector:

$$\boldsymbol{t} da = \boldsymbol{t} J dA =: \boldsymbol{T} dA \tag{2.29}$$

With the divergence theorem we can transform the surface integral of the balance of linear momentum to a volume integral:

$$\int_{\partial\Omega(t)} \boldsymbol{t}(\mathbf{x}, t, \partial\Omega(t)) da \overset{(2.27)}{=} \int_{\partial\Omega(t)} \boldsymbol{\sigma}(\mathbf{x}, t)\boldsymbol{n} da \overset{(A.29)}{=} \int_{\Omega(t)} \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma} dv \tag{2.30}$$

Inserting this result into the balance of linear momentum yields

$$\int_{\Omega(t)} \left(\nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma} + \boldsymbol{b} - \rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} \right) dv = 0 \tag{2.31}$$

As this equality holds for arbitrary volumes $\Omega(t)$, the integrand has to be zero and we can write Cauchy's equation of motion (CEM) in eulerian form

$$\nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} \tag{2.32}$$

Note, that based on this equation and the balance of angular momentum, the symmetry of Cauchy's stress tensor is enforced (see section 2).

Cauchy's equation of motion can't be resolved, as Cauchy's stress $\boldsymbol{\sigma}$, the body force $\boldsymbol{b}$, the velocity $\boldsymbol{v}$ and the density ρ in spatial configuration are unknown. In order to transform Cauchy's stress to the material configuration, we introduce the Piola-Kirchhoff stress tensor (PKST):

$$\boldsymbol{t}da = \boldsymbol{\sigma}\boldsymbol{n}da = \boldsymbol{\sigma} J \boldsymbol{F}^{-T} \boldsymbol{N} dA =: \boldsymbol{P}\boldsymbol{N}dA \tag{2.33}$$

Beyond the Cauchy stress tensor and the Piola-Kirchhoff stress tensor, other stress tensors are used in the literature, such as the symmetric purely mathematical Second Piola-Kirchhoff stress tensor:

$$\boldsymbol{S} \coloneqq \boldsymbol{F}^{-1}\boldsymbol{P} = J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T} = J(\boldsymbol{\sigma}\boldsymbol{F}^{-T})^T\boldsymbol{F}^{-T} = \boldsymbol{P}^T\boldsymbol{F}^{-T} = \boldsymbol{S}^T \tag{2.34}$$

In order to derive the equation of motion in the Lagrangiang form, we come back to the balance of linear momentum (2.26). Using Nanson's formula we transform the first term of the left hand side:

$$\int_{\partial\Omega(t)} \boldsymbol{t}da \overset{(2.27)}{=} \int_{\partial\Omega(t)} \boldsymbol{\sigma}\boldsymbol{n}da \overset{(2.17)}{=} \int_{\partial\Omega} \boldsymbol{\sigma} J \boldsymbol{F}^{-T}\boldsymbol{N}dA \tag{2.35}$$

$$= \int_{\partial\Omega} \boldsymbol{P}\boldsymbol{N}dA \overset{(A.29)}{=} \int_{\Omega} \nabla_{\mathbf{X}} \cdot \boldsymbol{P} dV \tag{2.36}$$

Similar to the derivation in 2.1.2, integration by substitution helps to transform the second term of the external forces:

$$\int_{\Omega(t)} \boldsymbol{b}(\mathbf{x},t)dv \overset{(A.30)}{=} \int_{\Omega} \boldsymbol{b}\left(\chi(\mathbf{X},t),t\right) \underbrace{|\det(\nabla_{\mathbf{X}}\chi(\mathbf{X},t))|}_{\overset{(2.15)}{=} J(\mathbf{X},t)} dV = \int_{\Omega} \boldsymbol{B}(\mathbf{X})dV \tag{2.37}$$

where the body forces in the two configurations are related using the jacobian of the deformation tensor:

$$\boldsymbol{B}(\mathbf{X}) = \boldsymbol{b}\left(\chi(\mathbf{X},t),t\right) J(\mathbf{X},t) \tag{2.38}$$

In summary, we obtain Cauchy's equation of motion formulated in reference configuration in the integral form:

$$\int_\Omega \nabla_{\mathbf{x}} \cdot \boldsymbol{P} dV + \int_\Omega \boldsymbol{B} dV = \int_\Omega P \frac{\mathrm{D}\boldsymbol{V}}{\mathrm{D}t} dV \tag{2.39}$$

Again, this integral equation can be derived for arbitrary volumes, thus the local representation equals

$$\nabla_{\mathbf{X}} \cdot \boldsymbol{P} + \boldsymbol{B} = P \frac{\mathrm{D}\boldsymbol{V}}{\mathrm{D}t} = P \frac{\partial \boldsymbol{V}}{\partial t} = P \frac{\mathrm{D}^2 \boldsymbol{U}}{\mathrm{D}t^2} \tag{2.40}$$

This equation is valid on the complete body B and only contains geometric or physical terms. With the appropriate boundary conditions, the infinite space of solutions to this differential equation can be limited. We impose boundary conditions like displacements $\boldsymbol{U}_D$ on $\Gamma_D \subseteq \partial B$ – *Dirichlet or essential boundary conditions* – and surface forces or surface traction $\boldsymbol{T}_N$ on $\Gamma_N \subseteq \partial B$ – *Neumann or natural boundary conditions*. Where boundary conditions have to be imposed on the complete boundary, i.e. $\partial B = \Gamma_D \cup \Gamma_N$, and can not overlap, i.e. $\Gamma_D \cap \Gamma_N = \emptyset$. Then the boundary conditions can be formulated as

$$V_D(\Gamma_D, \boldsymbol{U}_D) \coloneqq \{\boldsymbol{x} | \boldsymbol{x}(\mathbf{X}, t) = \mathbf{X} + \boldsymbol{U}_D(\mathbf{X}, t) \quad \forall \mathbf{X} \in \Gamma_D\}, \tag{2.41}$$

$$V_N(\Gamma_N, \boldsymbol{T}_N) \coloneqq \{\boldsymbol{x} | \boldsymbol{\sigma}\boldsymbol{n} = \boldsymbol{T}_N(\mathbf{X}, t) \quad \forall \mathbf{X} \in \Gamma_N\} \tag{2.42}$$

And thus we can formulate the boundary value problem of non-linear elasticity at a time t as
Find $\chi(\cdot, t) \in C^2(\Omega(t)) \cap C^1\left(\overline{\Omega(t)}\right) \cap V_D(\Gamma_D, \boldsymbol{U}_D) \cap V_N(\Gamma_N, \boldsymbol{T}_N)$ solving (2.40).

2.2 Constitutive theory - material laws and behaviours

In order to solve the differential equations stated in the last section, forces need to be related to the stress. Therefore, in this section, we introduce the constitutive equation, describing the macroscopic material behaviour. These help to simplify the complexity of the material behaviours and to cover the observed effects of experimental investigations.

In Cauchy-elastic or simple elastic materials, the stress is independent of the deformation history. Thus with the same final load, the same stress is expected, even with different deformation paths. For these materials, the constitutive equation uses a so called response function $\mathcal{G}$ to connect the stress to the deformation gradient

$$\boldsymbol{\sigma}(\mathbf{x}, t) = \mathcal{G}(\boldsymbol{F}(\boldsymbol{X}, t), \mathbf{X}), \quad \text{where } \mathbf{X} = \boldsymbol{X}(\mathbf{x}, t) \tag{2.43}$$

The following sections explain the hyperelasticity or Green elasticity, which is a Cauchy-elastic material behaviour. Rubberlike materials or human tissues belong to this category (see [48], page 198). This approach postulates the existence of the *strain energy density function* ψ, which is used to derive the stress strain relationship.

Note that one state of deformation can have multiple different deformation paths. While the stress for this state is the same for material behaviours based on the equation above, the internal or strain energy of Cauchy-elastic materials – in contrary to hyperelasticity – is potentially dependent on the deformation history. This can have an effect on the uniqueness of the boundary problem, introduced in 2.1.3. Further information about this active area of research is provided by [21, 41, 56].

In the first subsection, we derive the balance of mechanical energy for Cauchy's equation of motion in the material and eulerian configuration using different stress tensors. The derivation reveals that the strain energy density function can be used to describe the internal energy of a system. In order to specify this function better, the strain is introduced in subsection 2.2.2 followed by further information on possible simplifications using linearizations. Finally, subsection 2.2.3 uses the constitutive equation (2.43) to link the strain with the stress using the strain energy density function.

2.2.1 Work rate and balance of mechanical energy

In this subsection, we formulate the balance equation for the mechanical energy in the material configuration, used for the derivation of the weak formulation of the finite element method in 3.1. As a byproduct, the change of internal energy is expressed using different stress tensors. This expression is used to relate the stress and the strain in subsection 2.2.3.

Multiplying Cauchy's equation of motion in eulerian configuration (2.32) with the velocity $\boldsymbol{v}$, we obtain the differential equation

$$(\nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}) \cdot \boldsymbol{v} + \boldsymbol{b} \cdot \boldsymbol{v} = \rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} \cdot \boldsymbol{v} \tag{2.44}$$

We introduce the spatial velocity gradient $\boldsymbol{l} \coloneqq d\boldsymbol{v}/d\mathbf{x}$ and represent it by the *rate of deformation tensor* $\boldsymbol{d}$ and the *spin tensor* $\boldsymbol{w}$:

$$\boldsymbol{l} = \underbrace{\frac{1}{2}(\boldsymbol{l} + \boldsymbol{l}^T)}_{\coloneqq \boldsymbol{d}} + \underbrace{\frac{1}{2}(\boldsymbol{l} - \boldsymbol{l}^T)}_{\coloneqq \boldsymbol{w}} = \boldsymbol{d} + \boldsymbol{w} \tag{2.45}$$

With this decomposition, we change the first term of (2.44) with the product rule (A.27) to

$$(\nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}) \cdot \boldsymbol{v} \overset{(A.27)}{=} \nabla_{\mathbf{x}} \cdot (\boldsymbol{\sigma}\boldsymbol{v}) - \boldsymbol{\sigma} : \boldsymbol{l} \overset{(A.24)}{=} \nabla_{\mathbf{x}} \cdot (\boldsymbol{\sigma}\boldsymbol{v}) - \boldsymbol{\sigma} : \boldsymbol{d} \tag{2.46}$$

For the right hand side, the chain rule yields

$$\frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} \cdot \boldsymbol{v} = \frac{1}{2}\frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} \cdot 2\boldsymbol{v} = \frac{1}{2}\frac{\mathrm{D}(\boldsymbol{v}^2)}{\mathrm{D}t} \tag{2.47}$$

and insertion into equation (2.44) gives

$$\nabla_{\mathbf{x}} \cdot (\boldsymbol{\sigma}\boldsymbol{v}) - \boldsymbol{\sigma} : \boldsymbol{d} + \boldsymbol{b} \cdot \boldsymbol{v} = \rho\frac{1}{2}\frac{\mathrm{D}(\boldsymbol{v}^2)}{\mathrm{D}t} \tag{2.48}$$

After integration over the current configuration $\Omega(t)$ one can write

$$\int_{\Omega(t)} \nabla_{\mathbf{x}} \cdot (\boldsymbol{\sigma}\boldsymbol{v}) dv - \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{d} dv + \int_{\Omega(t)} \boldsymbol{b} \cdot \boldsymbol{v} dv = \int_{\Omega(t)} \rho\frac{1}{2}\frac{\mathrm{D}(\boldsymbol{v}^2)}{\mathrm{D}t} dv \tag{2.49}$$

With the divergence theorem, the balance equation for the mechanical energy in the spatial configuration can be written as:

$$\frac{1}{2}\int_{\Omega(t)} \rho\frac{\mathrm{D}\boldsymbol{v}^2}{\mathrm{D}t} dv + \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{d} dv = \int_{\partial\Omega(t)} \boldsymbol{t} \cdot \boldsymbol{v} da + \int_{\Omega(t)} \boldsymbol{b} \cdot \boldsymbol{v} dv \tag{2.50}$$

where this equation can be decomposed into the different parts: the time derivative of the kinetic energy

$$\Pi_{\mathrm{kin}}(t) \coloneqq \frac{1}{2}\int_{\Omega(t)} \rho\boldsymbol{v}^2 dv \tag{2.51}$$

the change of internal energy

$$\dot{\Pi}_{\mathrm{int}}(t) \coloneqq \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{d} dv \tag{2.52}$$

and the mechanical power due to surface and volume loads

$$\dot{\Pi}_{\mathrm{ext}}(t) \coloneqq \int_{\partial\Omega(t)} \boldsymbol{t} \cdot \boldsymbol{v} da + \int_{\Omega(t)} \boldsymbol{b} \cdot \boldsymbol{v} dv \tag{2.53}$$

With these definitions, the balance of mechanical energy in the current configuration can be stated as

$$\dot{\Pi}_{\mathrm{kin}} + \dot{\Pi}_{\mathrm{int}} = \dot{\Pi}_{\mathrm{ext}} \tag{2.54}$$

Since the current configuration might not be given, it is desirable to relate the balance to the the initial configuration. Pursuing this purpose, the velocity gradient allows to express the time derivative of the deformation gradient:

$$\dot{\boldsymbol{F}} = \frac{\mathrm{D}\boldsymbol{F}}{\mathrm{D}t} = \frac{\partial \boldsymbol{F}}{\partial t} = \frac{\partial}{\partial t}\frac{d\mathbf{x}}{d\mathbf{X}} = \frac{\partial}{\partial t}\frac{d\mathbf{x}}{d\mathbf{x}}\frac{d\mathbf{x}}{d\mathbf{X}} = \frac{d}{d\mathbf{x}}\frac{\partial \mathbf{x}}{\partial t}\frac{d\mathbf{x}}{d\mathbf{X}} = \boldsymbol{l}\boldsymbol{F} \tag{2.55}$$

Thus we have the following equality

$$\boldsymbol{l} = \dot{\boldsymbol{F}}\boldsymbol{F}^{-1} \tag{2.56}$$

and the internal energy transforms to

$$\dot{\Pi}_{\mathrm{int}}(t) = \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{l} dv = \int_{\Omega(t)} \boldsymbol{\sigma} : \dot{\boldsymbol{F}}\boldsymbol{F}^{-1} dv \overset{(\mathrm{A.22})}{=} \int_{\Omega(t)} \boldsymbol{\sigma}\boldsymbol{F}^{-T} : \dot{\boldsymbol{F}} dv \tag{2.57}$$

Then, with integration by substitution and the definition of the Piola-Kirchhoff stress tensor we have

$$\dot{\Pi}_{\mathrm{int}}(t) = \int_{\Omega} J\boldsymbol{\sigma}\boldsymbol{F}^{-T} : \dot{\boldsymbol{F}} dV = \int_{\Omega} \boldsymbol{P} : \dot{\boldsymbol{F}} dV \tag{2.58}$$

With similar arguments and by using the first Piola-Kirchhoff traction vector (2.29) we get

$$\dot{\Pi}_{\mathrm{kin}}(t) = \frac{1}{2}\int_{\Omega(t)} \rho(\mathbf{x},t)\frac{\mathrm{D}\boldsymbol{v}^2(\mathbf{x},t)}{\mathrm{D}t} dv = \frac{1}{2}\int_{\Omega} \rho(\mathbf{X})\frac{\mathrm{D}\boldsymbol{V}^2(\mathbf{X},t)}{\mathrm{D}t} dV \tag{2.59}$$

$$\dot{\Pi}_{\mathrm{ext}}(t) = \int_{\partial\Omega(t)} \boldsymbol{t}(\mathbf{x}, \partial\Omega(t)) \cdot \boldsymbol{v}(\mathbf{x},t) da + \int_{\Omega(t)} \boldsymbol{b}(\mathbf{x},t) \cdot \boldsymbol{v}(\mathbf{x},t) dv \tag{2.60}$$

$$= \int_{\partial\Omega} \boldsymbol{T}(\mathbf{X}) \cdot \boldsymbol{V}(\mathbf{X},t) dA + \int_{\Omega} \boldsymbol{b}(\mathbf{X}) \cdot \boldsymbol{V}(\mathbf{X},t) dV \tag{2.61}$$

Finally the balance equation for the mechanical energy in the material configuration gives

$$\frac{1}{2}\int_{\Omega} \rho\frac{\mathrm{D}\boldsymbol{V}^2}{\mathrm{D}t} dV + \int_{\Omega} \boldsymbol{P} : \dot{\boldsymbol{F}} dV = \int_{\partial\Omega} \boldsymbol{T} \cdot \boldsymbol{V} dA + \int_{\Omega} \boldsymbol{b} \cdot \boldsymbol{V} dV \tag{2.62}$$

In conclusion, we can express the change of internal energy in the initial and the current configuration

$$\dot{\Pi}_{\mathrm{int}}(t) = \int_{\Omega} \boldsymbol{P} : \dot{\boldsymbol{F}} dV = \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{l} dv \overset{\boldsymbol{\sigma}=\boldsymbol{\sigma}^T}{=} \int_{\Omega(t)} \boldsymbol{\sigma} : \boldsymbol{d} dv \tag{2.63}$$

In the context of pure hyperelastic materials, the internal or strain energy can as well be expressed in terms of the strain energy density function ψ, that has to be integrated over the undeformed initial geometry (see [48], page 180)

$$\Pi_{\text{int}} = \int_{\Omega} \psi dV \tag{2.64}$$

This shows the relevance of the strain energy function for the derivation of the material laws. To link the different representations of the internal energy, we introduce the strain tensors in the following.

2.2.2 Strain tensors

When a material is deformed, energy is converted from external (e.g. kinetic or potential) energy to internal – so called elastic or strain – energy. In the context of elastic material behaviour, it is assumed, that a strained object can regain its initial state upon the removal of the load, releasing the complete strain energy into other representations of energy. In order to quantify the elastic or strain energy, we postulate the existence of the elastic potential or strain energy function ψ.

The strain energy depends on the deformation of the object and does not change under rigid-body motions. Therefore, the strain energy function is invariant to arbitrary orthogonal tensors $\mathbf{Q}$:

$$\psi(\boldsymbol{F}) = \psi(\mathbf{Q}\boldsymbol{F}) \tag{2.65}$$

Using the transpose of the orthogonal tensor from the polar decomposition of the deformation $\boldsymbol{F} = \boldsymbol{R}\mathbf{U}$, the deformation can be decomposed into a pure rotation matrix $\boldsymbol{R}$ and a symmetric pure stretch matrix $\mathbf{U}$. Note, the stretch matrix $\mathbf{U}$ depends on the material coordinates $\mathbf{X}$. However, to prevent confusion with the displacement $\boldsymbol{U}(\mathbf{X}, t)$, we do not note this variable in italic. This allows to remove the dependency of the strain energy function from the rotation:

$$\psi(\boldsymbol{F}) \overset{(2.65)}{=} \psi(\boldsymbol{R}^T\boldsymbol{F}) = \psi(\boldsymbol{R}^T\boldsymbol{R}\mathbf{U}) = \psi(\mathbf{U}) \tag{2.66}$$

However, to obtain this formulation, a polar decomposition has to be performed at every point $\mathbf{X}$ of our object and at every time t. This makes the deformation $\boldsymbol{F}$ a disadvatageous candidate for a strain tensor in the first place, as it changes under rigid body motions.

In the literature, other deformation tensors have been proposed, which we use to derive the relevant finite strain tensors in the following. If we consider two infinitessimally small vectors $d\mathbf{X}, d\tilde{\mathbf{X}}$ starting from a point $\mathbf{X}$, we can determine deformed vectors $d\mathbf{x}, d\tilde{\mathbf{x}}$ and the starting point $\boldsymbol{x}(\mathbf{X}, t)$. Then the scalar product in the deformed state can be described in the undeformed state using the right Cauchy-Green deformation tensor (RCGDT):

$$\begin{aligned} d\mathbf{x} \cdot d\tilde{\mathbf{x}} &= (\boldsymbol{F}(\mathbf{X}, t)d\mathbf{X}) \cdot (\boldsymbol{F}(\mathbf{X}, t)d\tilde{\mathbf{X}}) \\ &= (\boldsymbol{F}(\mathbf{X}, t)d\mathbf{X})^T(\boldsymbol{F}(\mathbf{X}, t)d\tilde{\mathbf{X}}) \\ &= d\mathbf{X}^T \underbrace{\boldsymbol{F}(\mathbf{X}, t)^T \boldsymbol{F}(\mathbf{X}, t)}_{=\boldsymbol{F}^T\boldsymbol{F}=:\boldsymbol{C}} d\tilde{\mathbf{X}} \end{aligned}$$

Note that the right Cauchy-Green deformation tensor is invariant to the rotations in the deformation gradient

$$\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F} = (\boldsymbol{R}\mathbf{U})^T\boldsymbol{R}\mathbf{U} = \mathbf{U}^T\boldsymbol{R}^T\boldsymbol{R}\mathbf{U} \overset{\boldsymbol{R}^T\boldsymbol{R}=\mathbf{I}}{=} \mathbf{U}^T\mathbf{U} \overset{\mathbf{U}^T=\mathbf{U}}{=} \mathbf{U}\mathbf{U} \tag{2.67}$$

and for equal angles, it equals the identity matrix:

$$d\mathbf{x} \cdot d\tilde{\mathbf{x}} = d\mathbf{X} \cdot d\tilde{\mathbf{X}} \quad \Longrightarrow \quad \boldsymbol{C} = \mathbf{I} \tag{2.68}$$

To obtain the Green-Lagrange strain tensor, which is zero for rotations, we substract the scalar products

$$d\mathbf{x} \cdot d\tilde{\mathbf{x}} - d\mathbf{X} \cdot d\tilde{\mathbf{X}} = d\mathbf{X}^T\boldsymbol{C}d\tilde{\mathbf{X}} - d\mathbf{X}^T d\tilde{\mathbf{X}} \tag{2.69}$$

$$= d\mathbf{X}^T \underbrace{(\boldsymbol{C} - \mathbf{I})}_{=:2\boldsymbol{\varepsilon}_{gl}} d\tilde{\mathbf{X}} \tag{2.70}$$

With the previous derivations we can express the Green-Lagrange strain tensor dependent of the displacement:

$$\boldsymbol{\varepsilon}_{gl} = \frac{1}{2}(\boldsymbol{C} - \mathbf{I}) = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \mathbf{I}) \tag{2.71}$$

$$\overset{(2.11)}{=} \frac{1}{2}((\nabla_{\mathbf{X}}\boldsymbol{U} + \mathbf{I})^T(\nabla_{\mathbf{X}}\boldsymbol{U} + \mathbf{I}) - \mathbf{I}) \tag{2.72}$$

$$= \frac{1}{2}(\nabla_{\mathbf{X}}\boldsymbol{U}^T\nabla_{\mathbf{X}}\boldsymbol{U} + \nabla_{\mathbf{X}}\boldsymbol{U}^T + \nabla_{\mathbf{X}}\boldsymbol{U}) \tag{2.73}$$

Due to the symmetry and rotational invariance of the identity matrix, this tensor is as well symmetric and rotation invariant.

For small displacements, the quadratic term of the strain tensor vanishes

$$\frac{du_j}{dX_K} \ll 1 \quad \Longrightarrow \quad \nabla_{\mathbf{X}}\boldsymbol{U}^T\nabla_{\mathbf{X}}\boldsymbol{U} \approx 0 \tag{2.74}$$

and the linearized Green-Lagrange strain tensor can be defined:

$$\boldsymbol{\varepsilon}_{gl,l} = \frac{1}{2}(\nabla_{\mathbf{X}}\boldsymbol{U}^T + \nabla_{\mathbf{X}}\boldsymbol{U}) \approx \boldsymbol{\varepsilon}_{gl} \tag{2.75}$$

Similarly, the assumption of small displacements helps to justify the equality between the material and spatial gradient of the displacement

$$\nabla_{\mathbf{X}}\boldsymbol{U} = \frac{d\boldsymbol{U}}{d\mathbf{X}} = \frac{d\boldsymbol{u}}{d\mathbf{x}}\frac{d\mathbf{x}}{d\mathbf{X}} = (\nabla_{\mathbf{x}}\boldsymbol{u})\boldsymbol{F} = \nabla_{\mathbf{x}}\boldsymbol{u}\nabla_{\mathbf{X}}\boldsymbol{U} + \nabla_{\mathbf{x}}\boldsymbol{u} \approx \nabla_{\mathbf{x}}\boldsymbol{u} \tag{2.76}$$

Thus, the linearized Green-Lagrange strain tensor for small displacements can be expressed either using the material or the spatial derivative

$$\boldsymbol{\varepsilon}_{gl,l} = \frac{1}{2}(\nabla_{\mathbf{X}}\boldsymbol{U}^T + \nabla_{\mathbf{X}}\boldsymbol{U}) \approx \frac{1}{2}(\nabla_{\mathbf{x}}\boldsymbol{u}^T + \nabla_{\mathbf{x}}\boldsymbol{u}) \tag{2.77}$$

With $\nabla_{\mathbf{X}}\boldsymbol{U} = \boldsymbol{F} - \mathbf{I}$, the linearization of the Green-Lagrangian strain yields a direct dependency on the deformation gradient

$$\boldsymbol{\varepsilon}_{gl,l} = \frac{1}{2}((\boldsymbol{F} - \mathbf{I}) + (\boldsymbol{F} - \mathbf{I})^T) = \frac{1}{2}(\boldsymbol{F} + \boldsymbol{F}^T) - \mathbf{I} \tag{2.78}$$

As a result, rigid body motion like rotations $\chi(\boldsymbol{X}, t) = \boldsymbol{R}\boldsymbol{X}$ produce a nonzero strain $\boldsymbol{\varepsilon}_{gl,l} = \frac{1}{2}\boldsymbol{R} + \frac{1}{2}\boldsymbol{R}^T - \mathbf{I}$. We obtain a non-zero stress and thus an internal force that might yield a deformation – so called *ghost forces*. Thus, the theory of linear elastic materials designed for small deformations is as well restricted to small displacements. To allow applying this theory while avoiding an unrealistic behaviour due to rotations, the literature introduces a rotationfree, so called corotated, strain tensor. The polar decomposition $\boldsymbol{F} = \boldsymbol{R}\mathbf{U}$ of the deformation gradient, separates the pure rotations $\boldsymbol{R}$ from the strain tensor. Thus, based only on the pure stretch tensor $\mathbf{U}$, we can define the corotated strain tensor

$$\boldsymbol{\varepsilon}_{CR} = \frac{1}{2}(\mathbf{U} + \mathbf{U}^T) - \mathbf{I} \tag{2.79}$$

For the subsequent chapters, we consider the time derivative of the non-linearized tensor. We obtain with the product rule:

$$\dot{\boldsymbol{\varepsilon}}_{gl} = \frac{1}{2}\left(\dot{\boldsymbol{F}}^T\boldsymbol{F} + \boldsymbol{F}^T\dot{\boldsymbol{F}}\right) \tag{2.80}$$

Which can be expressed using the symmetric spatial velocity gradient $\boldsymbol{d}$ to get

$$\dot{\boldsymbol{\varepsilon}}_{gl} \overset{(2.56)}{=} \frac{1}{2}\left(\boldsymbol{F}^T\boldsymbol{l}^T\boldsymbol{F} + \boldsymbol{F}^T\boldsymbol{l}\boldsymbol{F}\right) = \boldsymbol{F}^T\frac{1}{2}\left(\boldsymbol{l}^T + \boldsymbol{l}\right)\boldsymbol{F} \overset{(2.45)}{=} \boldsymbol{F}^T\boldsymbol{d}\boldsymbol{F} \tag{2.81}$$

Based on this equation, the time derivative of the strain tensor is referred to as *pull back* of the spatial velocity gradient.

Moreover, for the variation of the strain tensor in the material configuration we obtain

$$\delta\boldsymbol{\varepsilon}_{gl} = \frac{d}{d\alpha}\frac{1}{2}\left[\nabla_{\mathbf{X}}(\boldsymbol{x}(\mathbf{X},t)+\alpha\boldsymbol{\eta})^T\nabla_{\mathbf{X}}(\boldsymbol{x}(\mathbf{X},t)+\alpha\boldsymbol{\eta})-\mathbf{I}\right]\Big|_{\alpha=0} \tag{2.82}$$

$$= \frac{1}{2}\frac{d}{d\alpha}\left[(\boldsymbol{F}+\alpha\nabla_{\mathbf{X}}\boldsymbol{\eta})^T(\boldsymbol{F}+\alpha\nabla_{\mathbf{X}}\boldsymbol{\eta})-\mathbf{I}\right]\Big|_{\alpha=0} \tag{2.83}$$

$$= \frac{1}{2}\left(2\alpha\nabla_{\mathbf{X}}(\boldsymbol{\eta})^T\nabla_{\mathbf{X}}\boldsymbol{\eta}+\boldsymbol{F}^T\nabla_{\mathbf{X}}\boldsymbol{\eta}+(\nabla_{\mathbf{X}}\boldsymbol{\eta})^T\boldsymbol{F}\right)\Big|_{\alpha=0} \tag{2.84}$$

$$= \frac{1}{2}\left(\boldsymbol{F}^T\nabla_{\mathbf{X}}\boldsymbol{\eta}+(\nabla_{\mathbf{X}}\boldsymbol{\eta})^T\boldsymbol{F}\right) \tag{2.85}$$

To conclude, in this subsection, we derived different measures for the deformation and strain: the right Cauchy-Green deformation tensor $\boldsymbol{C}$, the Green-Lagrange strain tensor $\boldsymbol{\varepsilon}_{gl}$, the linearized Green-Lagrange strain tensor $\boldsymbol{\varepsilon}_{gl,l}$ and the corotated strain tensor $\boldsymbol{\varepsilon}_{CR}$.

Note, that due to the invariance under rotations, the strain energy function usually is expressed in terms of the right Cauchy-Green deformation tensor or the Green-Lagrange strain tensor. However, the following paragraph points out, that using the standard deformation gradient $\boldsymbol{F}$ is accepted for the constitutive equation of the Piola-Kirchhoff stress tensor.

2.2.3 Constitutive equation - relation between strain and stress

In this subsection, we first summarize the results of the two previous subsections relevant for the material behaviour and then explain the assumption of pure hyperelasticity.

Similar to the derivations in (2.57) and (2.58) expressing the change of internal energy in terms of the Piola-Kirchhoff stress tensor, it can be expressed with the Second Piola-Kirchhoff stress tensor. Using integration by substition we can write

$$\dot{\Pi}_{\text{int}}(t) \overset{(2.63)}{=} \int_{\Omega}\boldsymbol{P}:\dot{\boldsymbol{F}}dV \overset{(2.63)}{=} \int_{\Omega(t)}\boldsymbol{\sigma}:\boldsymbol{d}dv = \int_{\Omega}J\boldsymbol{\sigma}:\boldsymbol{d}dV \tag{2.86}$$

With the definition of the Second Piola-Kirchhoff stress tensor and the *pull back* of the symmetric spatial velocity gradient derived in (2.81) we have

$$J\boldsymbol{\sigma}:\boldsymbol{d} \overset{(2.34)}{=} \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T:\boldsymbol{d} \overset{(A.21)}{=} \boldsymbol{S}\boldsymbol{F}^T:\boldsymbol{F}^T\boldsymbol{d} \overset{(A.22)}{=} \boldsymbol{S}:\boldsymbol{F}^T\boldsymbol{d}\boldsymbol{F} \overset{(2.81)}{=} \boldsymbol{S}:\dot{\boldsymbol{\varepsilon}}_{gl} \tag{2.87}$$

Taking the time derivative of the internal energy based on the strain energy function, we obtain

$$\dot{\Pi}_{\text{int}} = \frac{\mathrm{D}\Pi_{\text{int}}}{\mathrm{D}t} \overset{(2.64)}{=} \frac{\mathrm{D}\int_\Omega \psi dV}{\mathrm{D}t} = \int_\Omega \frac{\mathrm{D}\psi}{\mathrm{D}t} dV = \int_\Omega \dot{\psi} dV \tag{2.88}$$

Since this equation and equation (2.86) are valid for arbitrary domains Ω, the integrands equate each other

$$\dot{\psi} = J\boldsymbol{\sigma} : \boldsymbol{d} = \boldsymbol{P} : \dot{\boldsymbol{F}} = \boldsymbol{S} : \dot{\boldsymbol{\varepsilon}}_{gl} \tag{2.89}$$

This equation relates the derivative of the strain energy density function with different stress tensors. In the following paragraph, the strain energy density function serves as a link to the strain, closing the circle between stress and strain.

In hyperelastic materials the *stress can be obtained by differentiating the strain energy density with respect to strain* [48], page 184. Dependent on the stress tensor, appropriate measures for the deformation or strain have to be applied. Usually, the Piola-Kirchhoff stress tensor is directly expressed using the deformation gradient

$$\boldsymbol{P} = \boldsymbol{\sigma} J \boldsymbol{F}^{-T} = \mathcal{G}(\boldsymbol{F}) J \boldsymbol{F}^{-T} =: \mathcal{H}_{\boldsymbol{P}}(\boldsymbol{F}) \tag{2.90}$$

For the Second Piola-Kirchhoff stress tensor, the general response function (2.43) can be written in terms of the Green-Lagrange strain tensor $\boldsymbol{\varepsilon}_{gl}$ which defines the response function $\mathcal{H}_{\boldsymbol{S}}$ for this stress tensor:

$$\boldsymbol{S} = \boldsymbol{F}^{-1} \boldsymbol{\sigma} J \boldsymbol{F}^{-T} = \boldsymbol{F}^{-1} \mathcal{G}(\boldsymbol{\varepsilon}_{gl}) J \boldsymbol{F}^{-T} =: \mathcal{H}_{\boldsymbol{S}}(\boldsymbol{\varepsilon}_{gl}) \tag{2.91}$$

In order to express the response function $\mathcal{H}_{\boldsymbol{P}}$ using the strain energy density function, we take the time derivative and use the chain rule:

$$\dot{\psi} = \underbrace{\frac{\partial \psi}{\partial \boldsymbol{F}}}_{=\mathcal{H}_{\boldsymbol{P}}} : \dot{\boldsymbol{F}} = \underbrace{\frac{\partial \psi}{\partial \boldsymbol{\varepsilon}_{gl}}}_{=\mathcal{H}_{\boldsymbol{S}}} : \dot{\boldsymbol{\varepsilon}}_{gl} \tag{2.92}$$

With (2.89) we obtain

$$\left(\frac{\partial \psi}{\partial \boldsymbol{F}} - \boldsymbol{P} \right) : \dot{\boldsymbol{F}} = 0 \tag{2.93}$$

and

$$\left(\frac{\partial \psi}{\partial \boldsymbol{\varepsilon}_{gl}} - \boldsymbol{S} \right) : \dot{\boldsymbol{\varepsilon}}_{gl} = 0 \tag{2.94}$$

for arbitrary deformation gradients $\boldsymbol{F}$ and strain tensors $\boldsymbol{\varepsilon}_{gl}$. Finally we can write the constitutive equation for the Piola-Kirchhoff stress tensor

$$\boldsymbol{P} = \mathcal{H}_{\boldsymbol{P}}(\boldsymbol{F}) = \frac{\partial \psi}{\partial \boldsymbol{F}} \tag{2.95}$$

and for the Second Piola-Kirchhoff stress tensor

$$\boldsymbol{S} = \mathcal{H}_{\boldsymbol{S}}(\boldsymbol{\varepsilon}_{gl}) = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}_{gl}} \tag{2.96}$$

for appropriate strain energy density functions ψ adapted to the material.

2.2.4 Isotropic elastic material laws

In this subsection we derive the main material laws using the strain energy density function introduced in the previous chapters. The theory described before is not restricted to these few material laws, but for this work, we do not need further considerations. Beyond other works, the interested reader might refer to the work of S. Suwelack [116].

If a material behaves identically in all directions, it is called isotropic. Essentially this means, that the function ψ has to be independent of rotations and translation of the used deformation or strain measure. Mathematically speaking, for an arbitrary rotation matrix or orthogonal tensor $\mathbf{Q}$ we can write

$$\psi(\boldsymbol{C}) = \psi(\mathbf{Q}\boldsymbol{C}\mathbf{Q}^T) \tag{2.97}$$

There exist three invariants under these rotations, that allow to construct the strain energy density function for every material based on experimental data. Further information can be found in the reference mentioned above, here we want to focus on St. Venant-Kirchhoff material [21], with the strain energy density function

$$\psi(\boldsymbol{\varepsilon}) = \frac{\lambda}{2}(\mathrm{tr}(\boldsymbol{\varepsilon}))^2 + \mu \mathrm{tr}(\boldsymbol{\varepsilon}^2) \tag{2.98}$$

that can as well be written as

$$\psi(\boldsymbol{\varepsilon}) = \frac{1}{2}\boldsymbol{\varepsilon} : \mathbf{D} : \boldsymbol{\varepsilon} = \frac{1}{2}\boldsymbol{\varepsilon}_{ij} D_{ijkl} \boldsymbol{\varepsilon}_{kl} \tag{2.99}$$

with the fourth-order tensor $D_{ijkl} \coloneqq \lambda \delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) = D_{klij}$.

The Lamé parameters λ and μ are material-dependent quantities and are of theoretical nature for hyperelastic materials. Commonly, the Young's modulus

E and the Poisson's ratio ν are defined in terms of the Lamé parameters

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \qquad \nu = \frac{\lambda}{2(\lambda + \mu)} \tag{2.100}$$

This set of parameter better reflects the physical behaviour of the material. The Young's or elastic modulus is used to define the stress-strain relationship, i.e. a higher value results in a stiffer object. The Poisson's ratio quantifies the change of the volume, i.e. the expansion of an object perpendicular to a compression or stretch. While the Young's modulus can take any positive values, the Poisson's ratio is inferior to 0.5 and incompressible materials have a Poisson's ratio close to 0.5.

To derive the constitutive equation, it is simpler to calculate the derivative of the definition using the fourth-order tensor

$$\boldsymbol{S}_{mn} = \frac{\partial \psi(\boldsymbol{\varepsilon})}{\partial \boldsymbol{\varepsilon}_{mn}} = \frac{1}{2}\left(\frac{\partial \boldsymbol{\varepsilon}_{ij}}{\partial \boldsymbol{\varepsilon}_{mn}} D_{ijkl}\boldsymbol{\varepsilon}_{kl} + \boldsymbol{\varepsilon}_{ij} D_{ijkl}\frac{\partial \boldsymbol{\varepsilon}_{kl}}{\partial \boldsymbol{\varepsilon}_{mn}}\right) \tag{2.101}$$

$$= \frac{1}{2}\left(\delta_{im}\delta_{jn} D_{ijkl}\boldsymbol{\varepsilon}_{kl} + \boldsymbol{\varepsilon}_{ij} D_{ijkl}\delta_{km}\delta_{ln}\right) \tag{2.102}$$

$$= \frac{1}{2}\left(D_{mnkl}\boldsymbol{\varepsilon}_{kl} + \boldsymbol{\varepsilon}_{ij} D_{ijmn}\right) = D_{mnkl}\boldsymbol{\varepsilon}_{kl} \tag{2.103}$$

where the second equality is by the product rule and the last equality is due to the symmetry of $\mathbf{D}$. Summarizing, we obtain a linear constitutive relation written using the fourth-order constitutive tensor for isotropic materials

$$\boldsymbol{S} = \mathbf{D} : \boldsymbol{\varepsilon} \tag{2.104}$$

expressed using the standard notation

$$\boldsymbol{S} = \lambda \mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I} + 2\mu\boldsymbol{\varepsilon} \tag{2.105}$$

The St. Venant Kirchhoff material is often seen as a simple extension of linear elastic materials. For large strains, not many materials show a linear stress-strain relation, which restricts the linear elastic and St. Venant Kirchhoff model to small strains. However, linear elastic materials use the linearized green-lagrange strain tensor $\boldsymbol{\varepsilon}_{gl,l}$, while the St. Venant Kirchhoff materials use the Green-Lagrange strain tensor in its normal form $\boldsymbol{\varepsilon}_{gl}$. Consequently, the St. Venant Kirchhoff model allows for big displacements and rotations, while the linear elasticity is restricted to small displacements, as mentioned in 2.2.2.

In conclusion, we now can relate a force to the stress using the strain. With that, we now have the complete continuous theory and address the discretization using the finite element method in the following chapter.

Chapter 3

The finite element method

Contents

In this section, we introduce the main notions of the Finite Element Method (FEM), that replaces an object by a finite number of nodes connected by so-called finite elements that use the material laws derived in chapter 2.2. The finite element method can not be directly applied to the boundary value problem stated in the last section. As the first step, an integral equation – the weak formulation – is derived based on the strong and variational formulation. In order to discretize the continuous problem, the space is represented by points and elements, that interpolate the positions between the points. Then, these discretizations are inserted inside of the weak formulation approximating the balance equation at a time t. For the discretization in the domain of the time, section 3.3 describes different time integration schemes. This allows to formulate discrete non-linear problem, whose solution is obtained using linear equation solvers 3.4. We conclude with meshing aspects that are important in the context of topological changes.

The ideas of this chapter can be found with more details in [116], [48] and [125]. These references are used as guidance for this chapter and additional information is given addressing topological changes.

3.1 The weak formulation

The weak formulation is usually obtained in two different ways, either based on the strong formulation or using variational principles. Similar to the previous sections, the partial differential equations are derived differently in the initial and current configuration, while the main ideas are similar. In this work, we focus on the initial configuration, as the boundary conditions mentioned in (2.42) are given in this setting. The interested reader can find further details in [125], chapter 3.4.

The weak formulation is based on the Cauchy's equation of motion mentioned in (2.40), multiplying with time independent test functions $\boldsymbol{\eta} \in C^\infty(\Omega) \cap V_D(\Gamma_D, \mathbf{0})$, which are zero on the Dirichlet boundary. We get

$$(\nabla_{\mathbf{X}} \cdot \boldsymbol{P} + \boldsymbol{B})\, \boldsymbol{\eta} = \rho \frac{\mathrm{D}^2 \boldsymbol{U}}{\mathrm{D}t^2} \boldsymbol{\eta} \tag{3.1}$$

and after the integration over the initial domain

$$\int_\Omega \nabla_{\mathbf{X}} \cdot \boldsymbol{P} \cdot \boldsymbol{\eta} dV + \int_\Omega \boldsymbol{B} \boldsymbol{\eta} dV = \int_\Omega \rho \frac{\mathrm{D}^2 \boldsymbol{U}}{\mathrm{D}t^2} \boldsymbol{\eta} dV \tag{3.2}$$

We transform the first integrand using the product rule

$$\nabla_{\mathbf{X}} \cdot \boldsymbol{P} \cdot \boldsymbol{\eta} = \nabla_{\mathbf{X}} \cdot \boldsymbol{P}\boldsymbol{\eta} - \boldsymbol{P} : \nabla_{\mathbf{X}} \boldsymbol{\eta} \tag{3.3}$$

and replace the Piola-Kirchhoff stress tensor by the Second Piola-Kirchhoff stress tensor

$$\boldsymbol{P} : \nabla_{\mathbf{X}} \boldsymbol{\eta} \; = \boldsymbol{F}\boldsymbol{S} : \nabla_{\mathbf{X}} \boldsymbol{\eta} \overset{(\mathrm{A.21})}{=} \boldsymbol{S} : \boldsymbol{F}^T \nabla_{\mathbf{X}} \boldsymbol{\eta} \tag{3.4}$$

$$\overset{(\mathrm{A.23})}{=} \boldsymbol{S} : \frac{1}{2} \left(\boldsymbol{F}^T \nabla_{\mathbf{X}} \boldsymbol{\eta} + (\nabla_{\mathbf{X}} \boldsymbol{\eta})^T \boldsymbol{F} \right) \overset{(2.85)}{=} \boldsymbol{S} : \delta \boldsymbol{\varepsilon} \tag{3.5}$$

Then the divergence theorem gives

$$\int_\Omega \nabla_{\mathbf{X}} \cdot \boldsymbol{P} \cdot \boldsymbol{\eta} dV = \underbrace{\int_\Omega \nabla_{\mathbf{X}} \cdot \boldsymbol{P}\boldsymbol{\eta} dV}_{= \int_{\partial\Omega} \boldsymbol{T} \cdot \boldsymbol{\eta} dA} - \int_\Omega \boldsymbol{S} : \delta\boldsymbol{\varepsilon} dV \tag{3.6}$$

Thus we can write the weak form of the Cauchy's equation of motion in the material configuration

$$\underbrace{\int_\Omega \boldsymbol{S} : \delta\boldsymbol{\varepsilon} dV}_{=: a(\chi, \boldsymbol{\eta}) = a(\boldsymbol{U}, \boldsymbol{\eta})} \underbrace{- \int_{\partial\Omega} \boldsymbol{T} \cdot \boldsymbol{\eta} dA - \int_\Omega \boldsymbol{\eta} \cdot \boldsymbol{B} dV}_{=: l(\boldsymbol{\eta})} \underbrace{- \int_\Omega \boldsymbol{\eta} \cdot \rho \frac{\mathrm{D}^2 \boldsymbol{U}}{\mathrm{D}t^2} \boldsymbol{\eta} dV}_{=: b(\chi, \boldsymbol{\eta}) = b(\ddot{\boldsymbol{U}}, \boldsymbol{\eta})} = 0 \tag{3.7}$$

where we defined the bilinear forms a, b and the linear form l, used in the derivation of the FEM in subsection 3.2. Similarly one can obtain the solution for the static case

$$\int_\Omega \boldsymbol{S} : \delta\boldsymbol{\epsilon} dV - \int_{\partial\Omega} \boldsymbol{T}\boldsymbol{\eta} dA - \int_\Omega \boldsymbol{\eta}\boldsymbol{B} dV = 0 \tag{3.8}$$

Based on the integral equation, the problem can be discretized in space, using nodes that are connected by elements on which we apply the theory discussed in previous sections. This discretization is addressed in the following section.

3.2 Discretization and integration in space

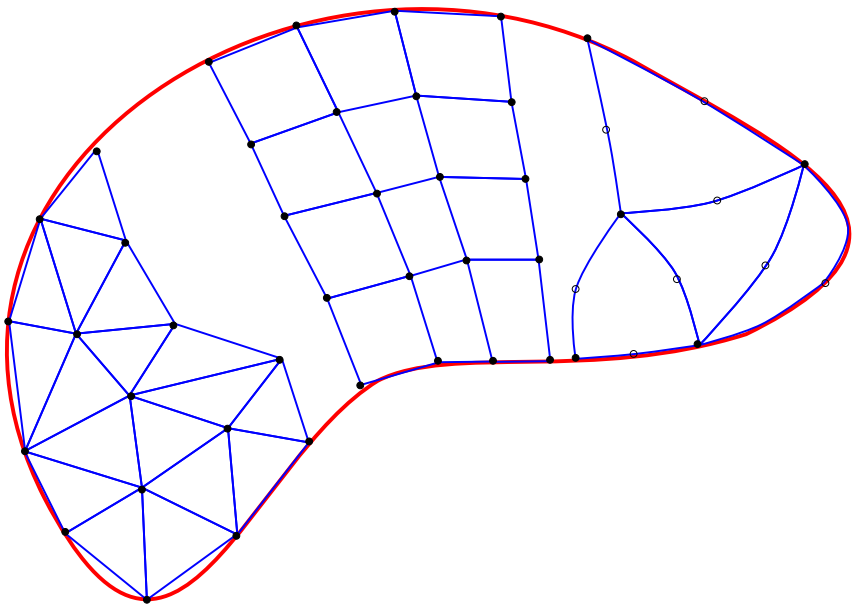


Figure 3.1: Body B discretized using different element types, *from left to right*: linear triangles, linear rectangles and quadratic triangles

In this subsection, the integral equation of the weak form is discretized using $n(\mathcal{E})$ so called finite elements $\mathcal{E}$. In the following, we present briefly the general idea, while details are given right thereafter. At the end of the subsection we describe the update of the system due to topological changes, when the standard form of the finite element method is used.

First of all, the geometry of a body in the initial configuration B is approx-

imated by a set of regions

$$B \approx B_h = \bigcup_{\mathrm{e}=1}^{n(\mathcal{E})} \Omega_\mathrm{e} \tag{3.9}$$

In each of the regions Ω_e, shape functions interpolate the geometry using the discrete values $\mathbf{P}_k$. Additionally shape functions help to approximate the primary variable $\boldsymbol{U}$ of our system using $\mathbf{u}_k$. The shape functions help to describe as well other variables like the vectorfield χ_h and, as a result, can transform every element to a reference element $\Omega_\triangle$. In the reference element, discrete integration methods yield an approximation of the integral using integration points $\boldsymbol{\xi}^g$. Summarizing, an integral over a body B can be transformed as follows

$$\int_B \star \, dV \approx \int_{B_h} \star \, dV = \bigcup_{\mathrm{e}=1}^{n(\mathcal{E})} \int_{\Omega_\mathrm{e}} \star \, dV = \bigcup_{\mathrm{e}=1}^{n(\mathcal{E})} \int_{\Omega_\triangle} \star' \, d\triangle \approx \bigcup_{\mathrm{e}=1}^{n(\mathcal{E})} \sum_{i=1}^{n(\boldsymbol{\xi}^g)} \star' \tag{3.10}$$

Where the union of the integrals over the elements is used to underline, that this is not a standard sum, but an assembling process. Similarly, the idea works on the boundary, since $\partial B \approx \partial B_h$, where ∂B_h can be expressed by $\partial \Omega_\mathrm{e}$. In the applied Galerkin method, test functions are approximated using the shape functions, that interpolate the values of the test functions at particular positions $\boldsymbol{\eta}_k$. Inserting the approximations of the displacement function and the test functions in (3.10) simplifies the integrals to

$$a(\boldsymbol{U}, \boldsymbol{\eta}) \approx \boldsymbol{\eta}_k \cdot \mathbf{f}_k^{int}(\mathbf{u}), \quad b(\ddot{\boldsymbol{U}}, \boldsymbol{\eta}) \approx \boldsymbol{\eta}_{k_1} \cdot \mathrm{M}_{k_1 k_2} \ddot{\mathbf{u}}_{k_2}, \quad l(\boldsymbol{\eta}) \approx \boldsymbol{\eta}_k \cdot \mathbf{f}_k^{ext} \tag{3.11}$$

Due to the arbitrary choice of the discrete test functions, the terms $\boldsymbol{\eta}_k$ cancel out. With the lexicographic order $(\mathbb{u}_{3k+1}, \mathbb{u}_{3k+2}, \mathbb{u}_{3k+3}) \coloneqq \mathbf{u}_k$, where $\mathbb{u} \in \mathbb{R}^{3n(\mathcal{P})\times 1}$ and similar replacements of $\mathbf{f}_k^{int}$ by $\mathbb{f}^{int}$, $\ddot{\mathbf{u}}_k$ by $\ddot{\mathbb{u}}$, $\mathrm{M}_{k_1 k_2}$ by $\mathbb{M}$ and $\mathbf{f}_k^{ext}$ by $\mathbb{f}^{ext}$, we can write the problem as a system of potentially non-linear equations

$$\mathbb{M}\ddot{\mathbb{u}} + \mathbb{f}^{int}(\mathbb{u}) = \mathbb{f}^{ext} \tag{3.12}$$

Linearization of the term $\mathbb{f}^{int}(\mathbb{u})$ and application of the constitutive equation in every element gives

$$\mathbb{M}\ddot{\mathbb{u}} + \mathbb{K}\mathbb{u} = \mathbb{f}^{ext} \tag{3.13}$$

Finally, a damping term containing the damping matrix $\mathbb{D}$ for the dissipation of the energy is added

$$\mathbb{M}\ddot{\mathbb{u}} + \mathbb{D}\dot{\mathbb{u}} + \mathbb{K}\mathbb{u} = \mathbb{f}^{ext} \tag{3.14}$$

In this subsection, we derive and write all the matrices of this equation explicitly. The calculation of the velocity $\dot{\mathbb{u}}$ and the acceleration $\ddot{\mathbb{u}}$ is explained in subsection 3.3 and the solution of the system in subsection 3.4.

For the finite element approach, an approximation B_h for the geometry of the body B has to fullfill the following conditions

- The intersection of two elements with ids e_1 and e_2 is either
  - Empty, i.e. $\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2} = \emptyset$,
  - A point, i.e. $\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2} = \Omega_k$,
  - An complete edge, i.e. $\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2} = \Omega_e$,
  - A complete face, i.e. $\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2} = \Omega_f$ or
  - A complete volume, i.e. $\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2} = \Omega_v$
- Inside of the body B_h are no gaps between the elements

To quantify the difference between the discrete and the real body, we use

$$h \coloneqq \max_{\mathrm{e} \in \mathcal{E}} \left(\max_{\mathbf{X}_1, \mathbf{X}_2 \in \Omega_\mathrm{e}} d(\mathbf{X}_1, \mathbf{X}_2) \right) \tag{3.15}$$

with the euclidean distance $d(\mathbf{X}_1, \mathbf{X}_2) = \|\mathbf{X}_1 - \mathbf{X}_2\|_2$ and we can write $B_h \overset{h \to 0}{\longrightarrow} B$.

In fact, the finite elements connect the positions $\mathbf{P}_{c_\mathrm{e}(l)} \in B_h$ and respectively the unknown positions $\mathbf{p}_{c_\mathrm{e}(l)} \in \chi(B_h)$ with $l \in \{1, ..., n(c_\mathrm{e})\}$, where the connectivity c_e contains the global numbers of the element nodes. Usually, a reference element $\Omega_\triangle$ helps charaterize arbitrary element geometries and local shape functions $N_{\triangle,l} : \Omega_\triangle \to \mathbb{R}$ are defined for every element type. Different element types with the reference element and their respective local shape functions are summarized in the appendix section 3. Inside a finite element the interpolation of the local shape functions can transform the reference element $\Omega_\triangle$ to the initial or current configuration of the element

$$\boldsymbol{X}_\mathrm{e}(\boldsymbol{\xi}) \coloneqq \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{P}_{c_\mathrm{e}(l)} N_{\triangle,l}(\boldsymbol{\xi}) \tag{3.16}$$

$$\boldsymbol{x}_\mathrm{e}(\boldsymbol{\xi}) \coloneqq \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{p}_{c_\mathrm{e}(l)} N_{\triangle,l}(\boldsymbol{\xi}) \tag{3.17}$$

In this definition, we assume that the transformations $\boldsymbol{X}_\mathrm{e} : \Omega_\triangle \to \Omega_\mathrm{e}$ and $\boldsymbol{x}_\mathrm{e} : \Omega_\triangle \to \Omega_\mathrm{e}(t)$ are bijective – every point in the element is reached exactly once. Thus we can express the local shape functions and the current position in an element as a function of the initial configuration, i.e. $N_{\mathrm{e},l}(\mathbf{X}) \coloneqq N_{\triangle,l}(\boldsymbol{X}_\mathrm{e}^{-1}(\mathbf{X}))\ \forall \mathbf{X} \in \Omega_\mathrm{e}, N_{\mathrm{e},l}(\mathbf{X}) = 0,\ \forall \mathbf{X} \in B_h \backslash \Omega_\mathrm{e}$ and

$$\boldsymbol{x}_\mathrm{e}(\mathbf{X}) \coloneqq \sum_{l=1}^{n(c_\mathrm{e})} N_{\mathrm{e},l}(\mathbf{X}) \mathbf{p}_{c_\mathrm{e}(l)} \tag{3.18}$$

Until not stated differently, the notation of the Euler sum is not used. Similarly, one can define local shape functions $\tilde{N}_{\mathrm{e},l}(\mathbf{X})$ to express the elementary discretized displacement function

$$\boldsymbol{U}_{\mathrm{e}}(\mathbf{X}) \coloneqq \sum_{l=1}^{n(c_{\mathrm{e}})} \tilde{N}_{\mathrm{e},l}(\mathbf{X}) \underbrace{(\mathbf{p}_{c_{\mathrm{e}}(l)} - \mathbf{P}_{c_{\mathrm{e}}(l)})}_{=:\mathbf{u}_{c_{\mathrm{e}}(l)}} \tag{3.19}$$

In our work we use the isoparametric approach that uses the same shape functions for the geometry and the primary variables, i.e. $\tilde{N} = N$. In the previous chapter, we have seen that $\boldsymbol{U}(\mathbf{X}, t) = \boldsymbol{u}(\mathbf{x}, t)$ when $\mathbf{x} = \boldsymbol{x}(\mathbf{X}, t)$. As the discrete displacement for a node with an id $k = c_{\mathrm{e}}(l)$ is neither dependent on the reference, nor dependent on the current configuration, we choose the lower case letter $\mathbf{u}_{c_{\mathrm{e}}(l)}$ to denote it in the future.

Since the elementary discrete displacement function $\boldsymbol{U}_{\mathrm{e}}$ is a sum of differentiable functions in $\boldsymbol{X}$, it is differentiable as well. Based on the definitions

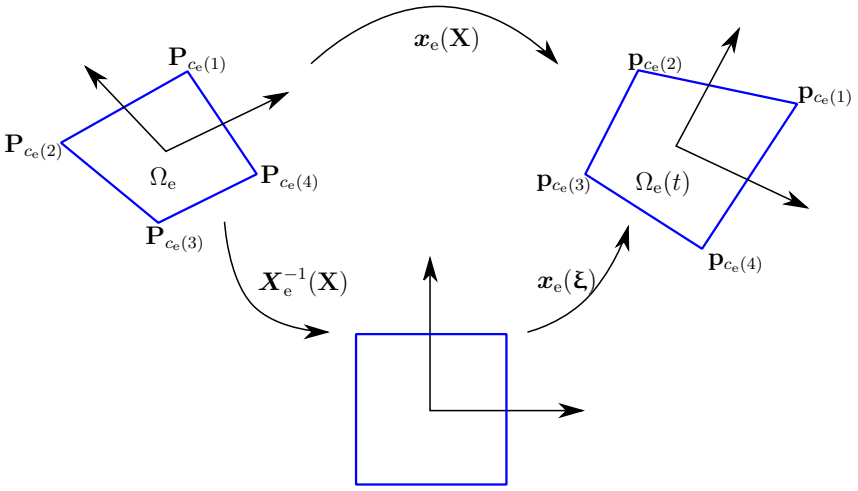


Figure 3.2: Isoparametric mapping of the deformation of a finite element

above, the isoparametric mapping displayed in figure 3.2 can be written as

$$\boldsymbol{F}_{\mathrm{e}} \coloneqq \boldsymbol{j}_{\mathrm{e}} \boldsymbol{J}_{\mathrm{e}}^{-1} \tag{3.20}$$

where the local shape functions serve to express the gradients $\boldsymbol{j}_{\mathrm{e}}$ and $\boldsymbol{J}_{\mathrm{e}}$

$$\boldsymbol{J}_{\mathrm{e}} \coloneqq \nabla_{\boldsymbol{\xi}} \boldsymbol{X}_{\mathrm{e}} = \sum_{l=1}^{n(c_{\mathrm{e}})} \mathbf{P}_{c_{\mathrm{e}}(l)} \nabla_{\boldsymbol{\xi}} N_{\triangle,l} \tag{3.21}$$

$$\boldsymbol{j}_{\mathrm{e}} \coloneqq \nabla_{\boldsymbol{\xi}} \boldsymbol{x}_{\mathrm{e}} = \sum_{l=1}^{n(c_{\mathrm{e}})} \mathbf{p}_{c_{\mathrm{e}}(l)} \nabla_{\boldsymbol{\xi}} N_{\triangle,l} \tag{3.22}$$

Note that, in these definitions, we used the outer product of vectors, that means e.g. for $\mathbf{P}_{c_\mathrm{e}(l)} \in \mathbb{R}^{3\times1}, \nabla_{\boldsymbol{\xi}} N_{\triangleright,l} \in \mathbb{R}^{1\times3}$ we have $\mathbf{P}_{c_\mathrm{e}(l)} \nabla_{\boldsymbol{\xi}} N_{\triangleright,l} \in \mathbb{R}^{3\times3}$. Since the outer product can be considered as a standard matrix multiplication, in the following declarations, the outer product is used without further comments. Hence the gradients in the material and spatial configuration can be stated as

$$\nabla_{\mathbf{X}} \boldsymbol{U}_\mathrm{e} = \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{u}_{c_\mathrm{e}(l)} \nabla_{\mathbf{X}} N_{\mathrm{e},l} = \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{u}_{c_\mathrm{e}(l)} \left(\nabla_{\boldsymbol{\xi}} N_{\triangleright,l} \, \boldsymbol{J}_\mathrm{e}^{-1} \right) \tag{3.23}$$

$$\nabla_{\mathbf{x}} \boldsymbol{U}_\mathrm{e} = \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{u}_{c_\mathrm{e}(l)} \nabla_{\mathbf{x}} N_{\mathrm{e},l} = \sum_{l=1}^{n(c_\mathrm{e})} \mathbf{u}_{c_\mathrm{e}(l)} \left(\nabla_{\boldsymbol{\xi}} N_{\triangleright,l} \, \boldsymbol{j}_\mathrm{e}^{-1} \right) \tag{3.24}$$

Note, that for linear local shape functions $N_{\triangleright,l}$, the derivative $\nabla_{\boldsymbol{\xi}} N_{\triangleright,l}$ is constant and thus $\boldsymbol{j}_\mathrm{e}, \boldsymbol{J}_\mathrm{e}, \nabla_{\mathbf{x}} \boldsymbol{U}_\mathrm{e}, \nabla_{\mathbf{X}} \boldsymbol{U}_\mathrm{e}$ are constant.

The global shape functions expressed using the Euler sum

$$\Phi_k(\boldsymbol{X}) \coloneqq \sum_{\mathrm{e}\in\mathcal{E}} \sum_{l=1}^{n(\mathrm{e})} \delta_{c_\mathrm{e}(l)k} N_{\mathrm{e},l}(\boldsymbol{X}) = \delta_{c_\mathrm{e}(l)k} N_{\mathrm{e},l}(\boldsymbol{X}) \tag{3.25}$$

allow for an extension of the observations from the elements to the complete discretized body B_h

$$\boldsymbol{U}_h \coloneqq \mathbf{u}_k \, \Phi_k = \mathbf{u}_k \delta_{c_\mathrm{e}(l)k} N_{\mathrm{e},l} = \mathbf{u}_{c_\mathrm{e}(l)} N_{\mathrm{e},l} \tag{3.26}$$

In this equation, we used first the representation of the discrete displacement dependent on the points and then on the elements. Both of these representations are important, the first is used to visualize the complete object, while the second is necessary for the derivation on which details are presented in the following. Commonly, the local shape functions form a basis of the function space of the finite elements and thus, with $B_h \overset{h\to0}{\longrightarrow} B$ we have $\boldsymbol{U}_\mathrm{e}(\mathbf{X}) \overset{h\to0}{\longrightarrow} \boldsymbol{U}(\mathbf{X}) \; \forall \mathbf{X} \in \Omega_\mathrm{e}$ and $\boldsymbol{U}_h(\boldsymbol{X}) \overset{h\to0}{\longrightarrow} \boldsymbol{U}(\mathbf{X}) \; \forall \mathbf{X} \in B_h$. Similar to the above, we can write

$$\nabla_{\mathbf{X}} \boldsymbol{U}_h = \mathbf{u}_{c_\mathrm{e}(l)} \nabla_{\mathbf{X}} N_{\mathrm{e},l} = \mathbf{p}_{c_\mathrm{e}(l)} \nabla_{\mathbf{X}} N_{\mathrm{e},l} - \mathbf{I} \tag{3.27}$$

$$\ddot{\boldsymbol{U}}_h \coloneqq \ddot{\mathbf{u}}_{c_\mathrm{e}(l)} N_{\mathrm{e},l} \tag{3.28}$$

Note, that these definitions differ from e.g. (3.23), as we sum over the elements using the Euler notation.

Applying the Galerkin method, the test functions can be discretized by interpolating values at specific points $\boldsymbol{\eta}_k$ using the shape functions

$$\boldsymbol{\eta}_h \coloneqq \boldsymbol{\eta}_k \, \Phi_k = \boldsymbol{\eta}_{c_\mathrm{e}(l)} N_{\mathrm{e},l} \tag{3.29}$$

$$\nabla_{\mathbf{X}} \boldsymbol{\eta}_h = \boldsymbol{\eta}_k \nabla_{\mathbf{X}} \Phi_k = \boldsymbol{\eta}_{c_\mathrm{e}(l)} \nabla_{\mathbf{X}} N_{\mathrm{e},l} \tag{3.30}$$

In the next steps we insert the results of this chapter in the weak form to obtain the equation system briefly mentioned in (3.45). The linear form of equation (3.7) becomes with the definitions above and the compact support of $N_{\triangle,l}$

$$l(\boldsymbol{\eta}) \approx l(\boldsymbol{\eta}_h) = \int_{\partial B_h} \boldsymbol{T} \cdot \boldsymbol{\eta}_h \; dA + \int_{B_h} \boldsymbol{B} \cdot \boldsymbol{\eta}_h \; dV \tag{3.31}$$

$$= \int_{\partial B_h \cap \Omega_\mathrm{e}} \boldsymbol{T} N_{\mathrm{e},l} \cdot \boldsymbol{\eta}_{c_\mathrm{e}(l)} \; dA + \int_{\Omega_\mathrm{e}} \boldsymbol{B} N_{\mathrm{e},l} \cdot \boldsymbol{\eta}_{c_\mathrm{e}(l)} \; dV \tag{3.32}$$

$$= \boldsymbol{\eta}_{c_\mathrm{e}(l)} \cdot \left(\int_{\partial B_h \cap \Omega_\mathrm{e}} \boldsymbol{T} N_{\mathrm{e},l} \; dA + \int_{\Omega_\mathrm{e}} \boldsymbol{B} N_{\mathrm{e},l} \; dV \right) \tag{3.33}$$

$$=: \boldsymbol{\eta}_{c_\mathrm{e}(l)} \cdot \mathbf{f}_{\mathrm{e},l}^{ext} = \boldsymbol{\eta}_k \cdot \delta_{c_\mathrm{e}(l)k} \mathbf{f}_{\mathrm{e},l}^{ext} =: \boldsymbol{\eta}_k \cdot \mathbf{f}_k^{ext} =: \eta_{ki} \mathbb{f}_{3k+i}^{ext} \tag{3.34}$$

where we defined the external forces $\mathbf{f}^{ext}$ in the elemental and nodal representation $\mathbf{f}_k^{ext} = \delta_{c_\mathrm{e}(l)k} \mathbf{f}_{\mathrm{e},l}^{ext}$. For the bilinear forms we progress similarly

$$b(\ddot{\boldsymbol{U}}, \boldsymbol{\eta}) \approx b(\ddot{\boldsymbol{U}}_h, \boldsymbol{\eta}_h) = \int_{B_h} \boldsymbol{\eta}_h \cdot \rho \ddot{\boldsymbol{U}}_h \; dV \tag{3.35}$$

$$= \int_{\Omega_{\mathrm{e}_1} \cap \Omega_{\mathrm{e}_2}} N_{\mathrm{e}_1,l_1} \boldsymbol{\eta}_{c_{\mathrm{e}_1}(l_1)} \cdot \rho N_{\mathrm{e}_2,l_2} \ddot{\mathbf{u}}_{c_{\mathrm{e}_2}(l_2)} dV \tag{3.36}$$

$$= \boldsymbol{\eta}_{c_{\mathrm{e}_1}(l_1)} \cdot \delta_{\mathrm{e}_1\mathrm{e}_2} \int_{\Omega_{\mathrm{e}_1}} \rho N_{\mathrm{e}_1,l_1} N_{\mathrm{e}_2,l_2} dV \ddot{\mathbf{u}}_{c_{\mathrm{e}_2}(l_2)} \tag{3.37}$$

$$=: \boldsymbol{\eta}_{c_{\mathrm{e}_1}(l_1)} \cdot \mathrm{M}_{\mathrm{e}_1 l_1 \mathrm{e}_2 l_2} \ddot{\mathbf{u}}_{c_{\mathrm{e}_2}(l_2)} =: \boldsymbol{\eta}_{k_1} \cdot \mathrm{M}_{k_1 k_2} \ddot{\mathbf{u}}_{k_2} \tag{3.38}$$

$$= \eta_{k_1 i_1} \mathbb{M}_{3k_1+i_1 3k_2+i_2} \ddot{\mathbb{u}}_{3k_2+i_2} \tag{3.39}$$

Note that M usually is a sparse matrix, as it only contains values at the positions $k_1 = c_\mathrm{e}(l_1), k_2 = c_\mathrm{e}(l_2)$, this property has been underlined in the previous equation using $\delta_{\mathrm{e}_1\mathrm{e}_2}$ and comes from $\Omega_{\mathrm{e}_1}^0 \cap \Omega_{\mathrm{e}_2}^0 = \emptyset$. Moreover, we have $\mathbf{f}_k^{ext} \in \mathbb{R}^3$, while $\mathrm{M}_{k_1 k_2} \in \mathbb{R}$, i.e. the matrix $\mathbb{M}_{3*k_1+i,3*k_2+i} \coloneqq \mathrm{M}_{k_1 k_2}, i \in \{1,2,3\}$ contains 3×3 blocks with the same constant on the diagonal for the node combinations k_1, k_2. Additionally, the node combinations k_1, k_2 and k_2, k_1 yield the same results, thus the matrix $\mathbb{M}$ is symmetric. In real-time simulations, the mass matrix often is simplified to the lumped mass matrix $\mathbb{M}'$ which is only non-zero on the diagonal. The simplest approach is to divide the total mass m by the number of nodes $n(\mathcal{P})$ and to write these values on the diagonal $\mathbb{M}'_{ii} = \mathrm{m}/n(\mathcal{P})$. Another idea is to sum the values of the lines and write the result on the diagonal $\mathbb{M}'_{ii} = \sum_j \mathbb{M}_{ij}$. While the first approach is faster to calculate, the second approach is more accurate as it takes account of the real mass around a node, e.g. in the case of a boundary node or a high resolution of nodes in a region. Now, for the bilinear form dependent on the discrete

displacement, we can write

$$a(\boldsymbol{U},\boldsymbol{\eta}) \approx a(\boldsymbol{U}_h,\boldsymbol{\eta}_h) = \int_{B_h} \boldsymbol{S} : \delta\boldsymbol{\varepsilon}\ dV \overset{(3.4)}{=} \int_{B_h} \boldsymbol{P} : \nabla_{\mathbf{X}}\boldsymbol{\eta}_h\ dV \tag{3.40}$$

$$= \int_{B_h} \boldsymbol{P} : \boldsymbol{\eta}_{c_\mathrm{e}(l)} \nabla_{\mathbf{X}} N_{\mathrm{e},l}\ dV \overset{(A.26)}{=} \boldsymbol{\eta}_{c_\mathrm{e}(l)} \cdot \int_{B_h} \underbrace{\boldsymbol{P}\nabla_{\mathbf{X}} N_{\mathrm{e},l}}_{=:\boldsymbol{f}^{int}_{\mathrm{e},l}(\mathbf{u},\mathbf{X})}\ dV \tag{3.41}$$

$$=: \boldsymbol{\eta}_k \cdot \int_{B_h} \boldsymbol{f}^{int}_k(\mathbf{u},\mathbf{X})\ dV = \boldsymbol{\eta}_k \cdot \mathbf{f}^{int}_k(\mathbf{u}) = \eta_{ki} \mathbb{f}^{int}_{3k+i}(\mathbb{u}) \tag{3.42}$$

Inserting above results inside of the weak form (3.7), we obtain the equality

$$\boldsymbol{\eta}_k \cdot (\mathrm{M}_{kk_2}\ddot{\mathbf{u}}_{k_2} + \mathbf{f}^{int}_k(\mathbf{u}) - \mathbf{f}^{ext}_k) = 0 \tag{3.43}$$

Due to the arbitrary choice of the test function $\boldsymbol{\eta}_h$ and with that the arbitrary choice of the values $\boldsymbol{\eta}_k$ at the nodes k, the discretized, potentially non-linear system of ordinary differential equations in vectorial form simplifies to

$$\mathrm{M}_{kk_2}\ddot{\mathbf{u}}_{k_2} + \mathbf{f}^{int}_k(\mathbf{u}) - \mathbf{f}^{ext}_k = 0 \quad \forall k \tag{3.44}$$

and after lexicographic reordering of the variables we can write the non-linear system of ordinary differential equations

$$\mathbb{M}\ddot{\mathbb{u}} + \mathbb{f}^{int}(\mathbb{u}) = \mathbb{f}^{ext} \tag{3.45}$$

where the first term with the mass matrix vanishes for a static problem.

Now, we consider the Taylor expansion around $\mathbf{u} = \mathbf{u}_T = \mathbf{0}$ of the internal force

$$\mathbf{f}^{int}_k(\mathbf{u}) \approx \mathbf{f}^{int}_k(\mathbf{u}_T) + \frac{\partial \mathbf{f}^{int}_k(\mathbf{u}_T, \dot{\mathbf{u}}_T)}{\partial \mathbf{u}}\mathbf{u} \tag{3.46}$$

If the reference configuration is taken as a natural configuration, i.e. one in which the stress vanishes, then the internal force vanishes $\mathbf{f}^{int}_k(\mathbf{u}_T) = \mathbf{0}\ \forall k$. And the tangent stiffness matrix can be calculated as

$$\mathbf{K}_{k_1k_2} \coloneqq \frac{\partial \mathbf{f}^{int}_{k_1}}{\partial \mathbf{u}_{k_2}} = \int_{B_h} \frac{\partial \boldsymbol{f}^{int}_{k_1}}{\partial \mathbf{u}_{k_2}}\ dV = \delta_{c_\mathrm{e}(l_1)k_1}\delta_{c_\mathrm{e}(l_2)k_2} \underbrace{\int_{\Omega_\mathrm{e}} \frac{\partial \boldsymbol{f}^{int}_{\mathrm{e},l_1}}{\partial \mathbf{u}_{c_\mathrm{e}(l_2)}}\ dV}_{=:\mathbf{K}_{\mathrm{e},l_1l_2}} \tag{3.47}$$

Thus the global stiffness matrix in vectorial form $\mathbf{K}_{k_1k_2}$ is assembled as a sum of element matrices $\mathbf{K}_{\mathrm{e},l_1l_2}$. The element matrices are calculated as the integral over the elements and using the constitutive equation derived in section 2.2.3 for linear elasticity, are explicitly written as (please refer to section 4 of the appendix for details):

$$\mathbb{K}_{\mathrm{e},3l_1+i_1\ 3l_2+i_2} = \int_{\Omega_\mathrm{e}} \lambda N_{l_2,i_2}N_{l_1,i_1} + \mu\delta_{i_1i_2}N_{l_2,i_3}N_{l_1,i_3} + \mu N_{l_2,i_1}N_{l_1,i_2}\ dV \tag{3.48}$$

where $(N_{l,i})_i = (N_{l,1}, N_{l,2}, N_{l,3}) = \nabla_{\mathbf{X}} N_{\mathrm{e},l}$. After the assembling process, the global stiffness matrix $\mathbb{K}$ is sparsely filled using the arguments explained in detail for the mass matrix. However, in contrast to the mass matrix $\mathbb{M}$, the blocks for the nodes k_1, k_2 are filled completely. Equation (3.48) shows, that the local matrix $\mathbb{K}_{\mathrm{e}}$ and thus the global stiffness matrix $\mathbb{K}$ are symmetric.

The derived stiffness matrix is based on the linear strain $\boldsymbol{\varepsilon}_{gl,l}$, and rigid body motions produce a non-zero strain, a non-zero stress and thus an internal force that might yield a deformation – so called *ghost forces*. To prevent that, one can use the corotated strain tensor. Inserting the corotated strain tensor into the constitutive equation used in the linear elasticity (2.105), similar to the calculations above, a corotated tangent stiffness matrix is calculated as

$$\mathbf{K}_{CR,k_1k_2} \coloneqq \delta_{c_{\mathrm{e}}(l_1)k_1}\delta_{c_{\mathrm{e}}(l_2)k_2} \int_{\Omega_{\mathrm{e}}} \boldsymbol{R} \frac{\partial \boldsymbol{f}^{int}_{\mathrm{e},l_1}}{\partial \mathbf{u}_{c_{\mathrm{e}}(l_2)}} \boldsymbol{R}^T \, dV \tag{3.49}$$

Note, that dependent on the polynomial degree and the integration method, several rotation matrices need to be calculated for every element, e.g. for every integration point. As a result, compared to the calculation for pure linear elasticity, additional computation time is necessary for the polar decompositions. However, the computational costs are reasonable when compared to the advantages in the results of the simulations, see figure 3.3. Therefore, we choose to use the corotated finite element method in this work. In the following, we use $\mathbb{K}$ interchangeably for the linear and the corotational approach.

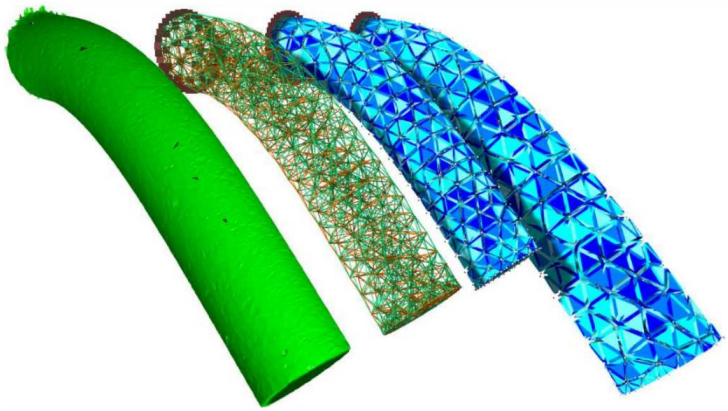


Figure 3.3: Comparison of different solutions of the bending problem; *from left to right* reference solution, mass-spring system, corotational FEM, and linear FEM models [2]

Coming back to the system of ordinary equations, for the linear and corotational approach, the internal forces do not depend upon the velocity $\dot{\mathbf{u}}$, so we have that

$$\mathbb{M}\ddot{\mathbf{u}} + \mathbb{K}\mathbf{u} = \mathbb{f}^{ext} \tag{3.50}$$

again, without the first term for a static problem. This equation system captures the main observations of experiments in the real world, but does not take account of e.g. viscous effects in the material or internal frictions that often occur combined. Assuming, that the damping is proportional to the velocity [125], page 207, we artifically add a term using a damping matrix $\mathbb{D}$, that can be used to simulate the dissipation of energy from the system:

$$\mathbb{M}\ddot{\mathbb{u}} + \mathbb{D}\dot{\mathbb{u}} + \mathbb{f}^{int}(\mathbb{u}) = \mathbb{f}^{ext} \tag{3.51}$$

where $\mathbb{f}^{int}(\mathbb{u}) = \mathbb{K}\mathbb{u}$ for linear elasticity and the corotational strain tensor. Usually, the so called Raleigh or modal damping is used [133]

$$\mathbb{D} = \alpha\mathbb{M} + \beta\mathbb{K} \tag{3.52}$$

where the parameters α and β have to be chosen experimentally [20]. Due to the symmetry of $\mathbb{M}$ and $\mathbb{K}$, the damping matrix is symmetric as well. The non-linear differential equation can be rewritten and is used in the following as

$$\mathbb{M}\ddot{\mathbb{u}} = \mathbb{f}(\mathbb{u}, \dot{\mathbb{u}}) \tag{3.53}$$

where the force $\mathbb{f}$ contains the internal forces, the damping term and the external forces.

To calculate the matrices $\mathbb{M}, \mathbb{K}, \mathbb{D}$ and the vectors $\mathbb{f}^{int}(\mathbb{u}), \mathbb{f}^{ext}$, integrals over the discretized domain B_h need to be computed. As it can be seen in the equations (3.33), (3.37) and (3.48), these integrals can be subdivided to integrals over the elements Ω_e or element boundaries $\partial B_h \cap \Omega_\mathrm{e}$. The integrands include the shape functions, derivatives of the shape functions and multiplications of the two of them, i.e. the integrands are polynomials with a maximal degree of two times the polynom degree of the local shape functions. The general approach to calculate the integrals is a transformation back to the reference elements $\Omega_\triangleleft$ using the transformation theorem. In the reference elements, *Gauss* integration points $\boldsymbol{\xi}^g$ allow for an exact integration of the polynomial integrand and thus we can write:

$$\int_{\Omega_\mathrm{e}} f \; dV = \int_{\Omega_\triangleleft} f \det(\boldsymbol{J}_\mathrm{e}) \; d\triangleleft = \sum_{i=1}^{n(\boldsymbol{\xi}^g)} w(\boldsymbol{\xi}_i^g) f(\boldsymbol{\xi}_i^g) \det(\boldsymbol{J}_\mathrm{e}(\boldsymbol{\xi}_i^g)) \tag{3.54}$$

For further details on the choice of the integration points, please refer to [125].

The conventional finite element method only allows for a separation along the element faces. Therefore, for an emerging cut, the method needs to update the element regions Ω_e intersecting with the cut. These regions are constructed based on the current positions $\mathbf{p}_k$, that either move or are inserted on the cut introducing new element regions. These movements or insertions in the spatial

configuration can be expressed as barycentric coordinates of the existing finite element mesh and are transformed to the material configuration in order to calculate the stiffness, mass and damping matrix and the internal and external forces. Note, that the choices for the calculation of these matrices have an important impact on the simulation of the deformation: For example an equal distribution of the mass to all nodes results potentially increased mass in the cut region, as it contains more nodes after being remeshed. And with the corotational finite element stiffness matrix, the rotations need to be recomputed for every updated or newly inserted region Ω_e.

To summarize this subsection, we transformed the integral equation of the weak formulation at a fixed time to a non-linear system of equations, described by matrices that are computed as the integral over elementary domains that discretize the space. Then we mentioned necessary updates related to an emerging cut in the conventional finite element method. Now, we would like to address simulations with a variation of the time and discretize the space.

3.3 Time discretization and integration

Beyond the displacement in the initial and current position – the static problem – we would like to know the intermediate states of our body. That means, that the time needs to be discretized similarly to the space, i.e. with the same order of accuracy. To meet this requirement, the linear momentum, angular momentum and the mechanical energy should possibly be conserved over time. We choose an isochronal discretization, i.e. with the time step Δt and the start time t_0 of our simulation, time is discretized recursively as $t_{n+1} = t_n + \Delta t$. Aiming on real-time simulations means, that the calculation of the deformation in a time step Δt has to take less time than the time step Δt.

A time integration method is defined to relate the derivatives of the discretized displacement $\mathbb{u}, \dot{\mathbb{u}}$ and $\ddot{\mathbb{u}}$ in the discretized space of time. The time integration method adds the link necessary to transform the system of second order ordinary differential equations (3.51) to a system of equations.

Explicit time integration methods only can enforce (3.51) at the beginning of the time step Δt, while *implicit* methods enforce it as well but not necessarily only at the end. Explicit methods are fast, parallelisable, but only conditionally stable dependent on the size of the time step. Implicit methods are slower per time step but usually allow the choice of greater time steps while preserving the stability and are therefore used more frequently [3]. Further in-

formations are given in the following paragraphs that describe the explicit euler and implicit euler method exemplarily, pointing at the advantages and disadvantages. In the following, we indicate to which time step the variables belong to with the superscripts t and $t + \Delta t$ for an arbitrary t.

The explicit euler method can be expressed as

$$\mathbb{M}\ddot{\mathbb{u}}^{t+\Delta t} = \mathbb{f}(\mathbb{u}^t, \dot{\mathbb{u}}^t) \tag{3.55}$$

$$\dot{\mathbb{u}}^{t+\Delta t} = \dot{\mathbb{u}}^t + \Delta t\ \ddot{\mathbb{u}}^t \tag{3.56}$$

$$\mathbb{u}^{t+\Delta t} = \mathbb{u}^t + \Delta t\ \dot{\mathbb{u}}^t \tag{3.57}$$

calculating the displacements, velocities and accelerations at the time $t + \Delta t$ based on the known values at the time t. Thus the non-linear behaviour of equation (3.51) transforms to a linear system of equations. If mass lumping is used, the mass matrix is sparsely filled and even diagonal and thus the degrees of freedom can be decoupled. As a result, the solution of the linear system of equations – and with that the discretized differential equation – is calculated very quickly and can be parallelized [23]. However, the size of the time step is limited by material parameters

$$\Delta t < \frac{h}{\sqrt{E/\rho}} \tag{3.58}$$

to maintain the stability of the system [34]. For stiff materials, the increased Young's modulus yields a smaller time step to ensure the stability of the system. A smaller time step results in more calculation to advance in time particularly for stiff systems [115]. Therefore, the method is rather used for the simulation of soft tissues, such as the brain tissues [45]. If the condition for the time step is violated in a time step, the solution might not show an "obvious" instability, but only a significant solution error. Such an error is particularly dangerous, as it can be accumulated over several time steps [6]. As a result, this research area stays active improving the general applicability of the method, e.g. by artifical augmentation of the mass matrix for stiff materials [44].

In contrary to the explicit euler method, the implicit euler method

$$\mathbb{M}\ddot{\mathbb{u}}^{t+\Delta t} = \mathbb{f}(\mathbb{u}^{t+\Delta t}, \dot{\mathbb{u}}^{t+\Delta t}) \tag{3.59}$$

$$\dot{\mathbb{u}}^{t+\Delta t} = \dot{\mathbb{u}}^t + \Delta t\ \ddot{\mathbb{u}}^{t+\Delta t} \tag{3.60}$$

$$\mathbb{u}^{t+\Delta t} = \mathbb{u}^t + \Delta t\ \dot{\mathbb{u}}^{t+\Delta t} \tag{3.61}$$

results in a non-linear equation system, as the unknown variables $\mathbb{u}^{t+\Delta t}$ and $\dot{\mathbb{u}}^{t+\Delta t}$ are inside of the potentially non-linear function $\mathbb{f}$. With $\mathrm{d}\mathbb{u} = (\mathbb{u}^{t+\Delta t} - \mathbb{u}^t)$ and respectively $\mathrm{d}\dot{\mathbb{u}} = (\dot{\mathbb{u}}^{t+\Delta t} - \dot{\mathbb{u}}^t)$, the Taylor approximation of first order

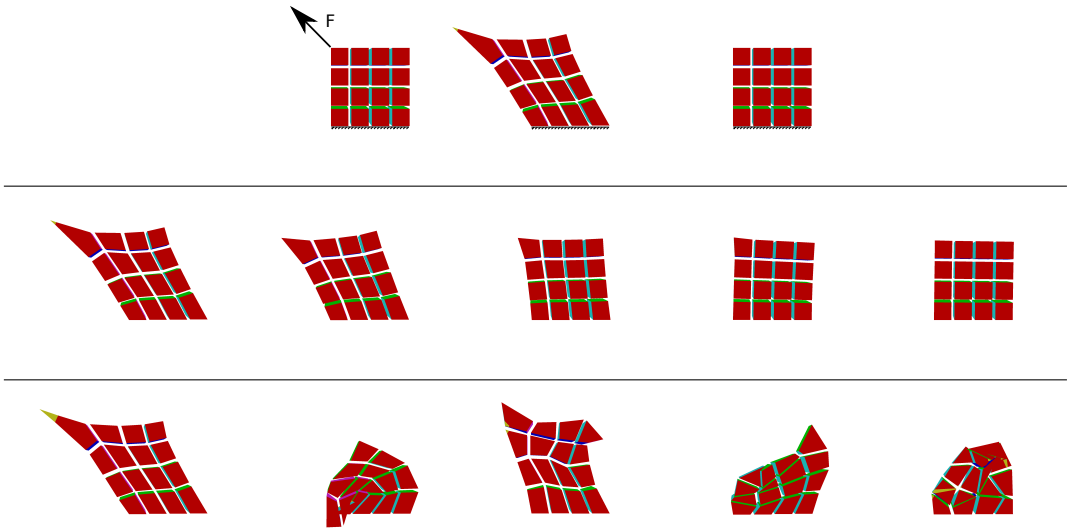


Figure 3.4: *First line*: Considered example *left* rest position with applied forces, *middle* deformed position when force is set to zero, *right* expected final position; Relaxation process shows different behaviour when using an implicit (*second line*) and explicit euler method (*third line*) that is unstable and explodes at the end

gives

$$\mathbb{f}(\mathbb{u}^{t+\Delta t}, \dot{\mathbb{u}}^{t+\Delta t}) \approx \mathbb{f}(\mathbb{u}^t, \dot{\mathbb{u}}^t) + \frac{\partial \mathbb{f}(\mathbb{u}^t, \dot{\mathbb{u}}^t)}{\partial \mathbb{u}} \mathrm{d}\mathbb{u} + \frac{\partial \mathbb{f}(\mathbb{u}^t, \dot{\mathbb{u}}^t)}{\partial \dot{\mathbb{u}}} \mathrm{d}\dot{\mathbb{u}} \tag{3.62}$$

$$= \mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}\mathrm{d}\mathbb{u} + \mathbb{D}\mathrm{d}\dot{\mathbb{u}} \tag{3.63}$$

In this setting, the stiffness matrix $\mathbb{K}$ (and the damping matrix $\mathbb{D}$) are so called tangent matrices that express the changement of the forces in the direction of the variable $\mathbb{u}$ (and respectively $\dot{\mathbb{u}}$). Inserting the implicit euler method transforms the equation above to

$$\mathbb{M}\ddot{\mathbb{u}}^{t+\Delta t} = \mathbb{f}(\mathbb{u}^{t+\Delta t}, \dot{\mathbb{u}}^{t+\Delta t}) \tag{3.64}$$

$$\approx \mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}(\mathbb{u}^{t+\Delta t} - \mathbb{u}^t) + \mathbb{D}(\dot{\mathbb{u}}^{t+\Delta t} - \dot{\mathbb{u}}^t) \tag{3.65}$$

$$= \mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}\Delta t\dot{\mathbb{u}}^{t+\Delta t} + \mathbb{D}\Delta t\ \ddot{\mathbb{u}}^{t+\Delta t} \tag{3.66}$$

$$= \mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}(\Delta t\dot{\mathbb{u}}^t + \Delta t^2\ddot{\mathbb{u}}^{t+\Delta t}) + \mathbb{D}\Delta t\ \ddot{\mathbb{u}}^{t+\Delta t} \tag{3.67}$$

$$= \mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}\Delta t\dot{\mathbb{u}}^t + (\Delta t^2\mathbb{K} + \mathbb{D}\Delta t)\ddot{\mathbb{u}}^{t+\Delta t} \tag{3.68}$$

Assuming the equality, the Newton-Raphson algorithm helps to solve the non-linear equation system (3.59). The Newton-Raphson algorithm is the primary solution scheme for non-linear finite element equations. It uses the Taylor approximation mentioned above to iteratively approach the solution of the non-linear system of equations. Please find more information on the Newton-Raphson algorithm in [48], section 2.2.1, [6], section 6.1 and [125], section

5.1.1. To minimize the computation time, simulations aiming on real-time performance often perform only one iteration of the Newton-Raphson algorithm. For fully non-linear material behaviours, this approach can raise stability issues, but the chosen method for this work, the corotated finite element method, stays stable. Then the acceleration $\ddot{\mathbb{u}}^{t+\Delta t}$ is the solution of the equation system

$$\underbrace{(\mathbb{M} - \Delta t\mathbb{D} - \Delta t^2\mathbb{K})}_{=:\mathbb{A}} \underbrace{\ddot{\mathbb{u}}^{t+\Delta t}}_{=:\mathbb{x}} = \underbrace{\mathbb{f}^{int}(\mathbb{u}^t) + \mathbb{f}^{ext} + \mathbb{K}\Delta t\dot{\mathbb{u}}^t}_{=:\mathbb{b}} \tag{3.69}$$

Then using the definition of the implicit euler method, we obtain the velocity and the displacements at the current time step.

When cutting occurs in the conventional finite element method, regions Ω_e intersecting with the cut are updated and the mesh size h potentially decreases. Since this has a direct impact on the size of the time step (see (3.58)) for explicit methods, we apply the implicit euler method for our simulation.

Solving the equation system is dependent on the invertibility of the matrix $\mathbb{A}$ defined in the equation above. In the next subsection, solving this equation system is discussed briefly pointing out the main types of methods used.

3.4 Solving systems of linear equations

In the last chapter the non-linear equation system presented for the implicit euler method has been solved using one step of the Newton-Raphson method. In a more general setting, other iterative solvers for non-linear equation systems are applied, e.g. the quasi-Newton method. Iterative solvers for non-linear equations systems usually linearize the system and construct very large linear equation systems for every iteration. The process of resolving these linear equation systems takes a considerable amount of the time resolving the complete finite element system and thus linear equation solvers play an important role when considering the real-time aspect of the finite element method. This chapter is devoted to give a brief insight into the different approaches to solve systems of linear equation.

We consider the general case of a full rank square matrix $\mathbb{A}$ and the right hand side $\mathbb{b}$, aiming on obtaining a vector $\mathbb{x}$ solving $\mathbb{A}\mathbb{x} = \mathbb{b}$. Due to the symmetry of the mass, damping and stiffness matrix, the equation system presented in (3.69) is using a symmetric matrix $\mathbb{A}$. Furthermore, the systems of equations constructed in the finite element method are sparse due to the sparseness of the mass and the stiffness matrix. Thus, we restrict our description to symmetric sparse matrices $\mathbb{A}$, please see figure 3.5 for a simple

example. For the matrix $\mathbb{A}$ used in the linear system like the one above, a

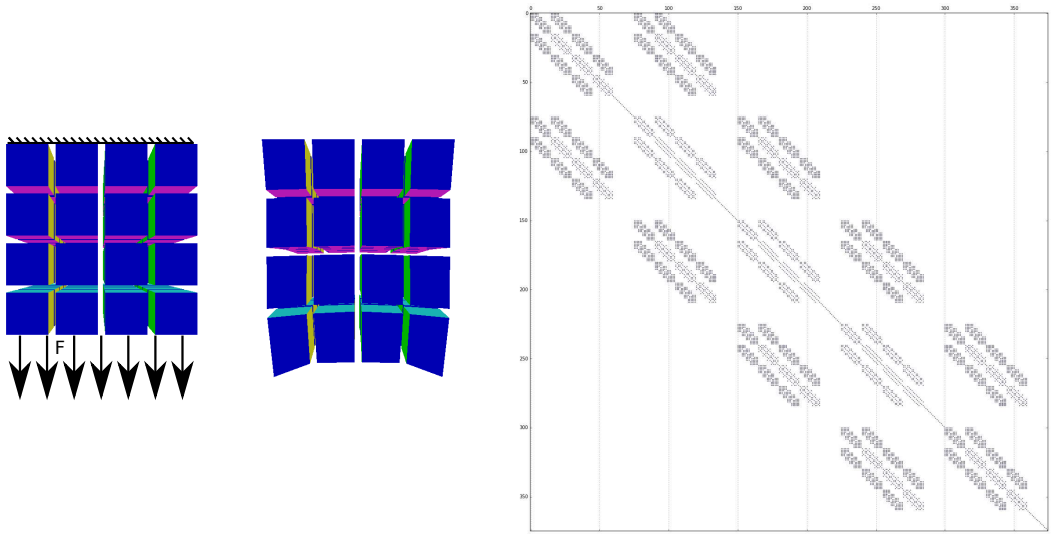


Figure 3.5: Sparsity of a stiffness matrix in an example of a hexahedral beam: *left* example in initial configuration with forces on bottom (*black arrows*) and fixed displacement conditions on top; *middle* deformed configuration; *right* entries in the stiffness matrix

condition number κ can be defined as

$$\kappa(\mathbb{A}) = \frac{\lambda_{\max}}{\lambda_{\min}} = \|\mathbb{A}\|\|\mathbb{A}^{-1}\| \geq 1 \tag{3.70}$$

where $\lambda_{\max}$ is the maximal and $\lambda_{\min}$ the minimal eigenvalue of the matrix $\mathbb{A}$. A system is ill-conditioned iff the condition number κ has a very high value.

Coming back to the finite element problem, to address the static and the dynamic system at once, we consider in the following only the eigenvalues of the stiffness matrix $\mathbb{K}$. To avoid an ill-conditioned matrix $\mathbb{K}$, it is valuable to provide a lower limit for the minimal eigenvalue λ_{min} and an upper limit for the maximal eigenvalues λ_{max}. While the minimal eigenvalue λ_{min} is closely tied to the smallest size $\|\Omega_{\mathrm{e,min}}\|$ (area or volume) and the material parameters of the elements, the largest eigenvalue λ_{max} can be made by a single badly shaped element [**?**]. That means, the condition number is proportional to the largest eigenvalue of the element stiffness matrices [32]. This proportionality allows to simplify the consideration to single elements: for triangular elements, which approach a very small angle, and for tetrahedral elements with a small dihedral angle, the maximal eigenvalue of the element stiffness matrix can go towards infinity. If there are no badly shaped elements in a tetrahedral mesh, the condition number is proportional to the maximal edge length $l_{\max}$ in the mesh and it can be stated $\kappa \in \mathcal{O}(l_{\max}/\|\Omega_{\mathrm{e,min}}\|)$. Thus, to prevent obtaining

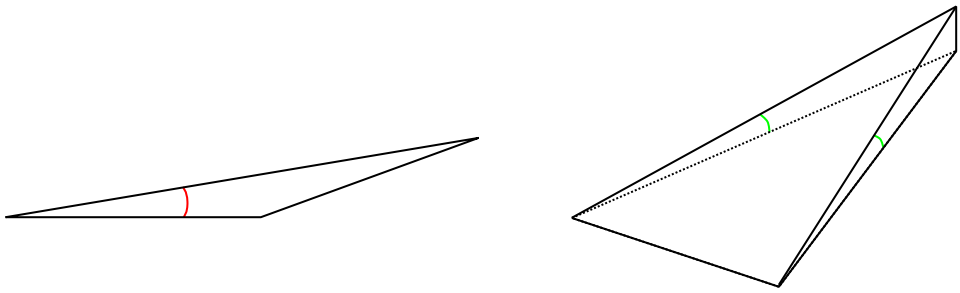


Figure 3.6: Triangle and tetrahedra with high eigenvalue λ_{max}, due to a small angle of the triangle *(in red)* and a small dihedral angle of the tetrahedron *(in green)*

an ill-conditioned system in the first place, badly shaped elements need to be avoided and there should not exist a big ratio between very small and very big elements, i.e. a uniform mesh is advantageous. For further arguments, e.g. related to the impact of material properties on the condition number, see [**?**]. Note, that topological changes induced from remeshing methods that replace one element by several smaller elements, are prone to changing the $\|\Omega_{\mathrm{e,min}}\|$ and thus augmenting the condition number. Therefore, it is important to observe and control the change of the condition number when introducing a new remeshing algorithm.

There exist two different approaches to solve a linear system of equations: direct and iterative solvers. Direct solvers calculate an exact solution in a finite number of steps [22], even for ill-conditioned or negative definite system equations, if round-off errors do not affect the solution. Methods of this class are prone to using a lot of memory in the resolution process particularly for large equation systems. Iterative solvers approximate and eventually converge to the solution, are easier to implement, faster and use less memory [4]. Real-time simulation rates can be achieved by limiting the number of iterations, which might result in an incorrect behaviour when the iterative method did not yet approximate the solution enough. An ill-conditioned system has a great impact on the convergence rate of these methods and without the calculation of the so called preconditioning matrix, the convergence rate might be extremely low.

Direct solvers have been an important research area for several decades, as they can provide trustworthy solutions even under unfortunate conditions, e.g. when a system is ill-conditioned. Well known from linear algebra, the Gauss elimination uses multiplications and adding of rows to transform the linear equation system in a upper triangular matrix $\mathbb{U}$, which is then used in the backward substitution to find the solution $\mathbb{x}$ of the system. Another type of direct method factorizes the matrix $\mathbb{A}$ into other matrices with particular properties, that are used to calculate the solution straightforwardly. For the

symmetric matrix $\mathbb{A}$, the *LU decomposition* constructs the same upper triangular matrix $\mathbb{U}$ as the Gauss elimination and additionally a lower triangular matrix $\mathbb{L}$, s.t. $\mathbb{A} = \mathbb{L}\mathbb{U}$. The solution is then computed by first solving $\mathbb{L}\mathbb{y} = \mathbb{b}$ by forward substitution and then $\mathbb{U}\mathbb{x} = \mathbb{y}$ by backward substition. In both techniques mentioned above the sparseness of $\mathbb{A}$ might not be conserved and e.g the matrix $\mathbb{U}$ is dense, resulting in a high usage of memory. For our example of figure 3.5, the global stiffness matrix $\mathbb{K}$ has 10785 entries, while $\mathbb{L}$ and $\mathbb{U}$ have together 38340 entries. To prevent this effect, sparse solvers have

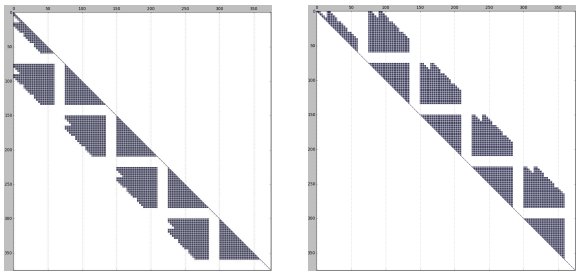


Figure 3.7: LU decomposition of the example of figure 3.5: *left* lower matrix $\mathbb{L}$ and *right* upper matrix $\mathbb{U}$

been developed, reducing the memory usage and thus improving the performance. Open source implementations of sparse direct solver are available, e.g. PARDISO shows very good performance [99]. Due to their robustness, direct sparse methods often get applied in the early phase of the development of an algorithm: e.g. for the implementation of a new method, as a first step the static problem is solved using direct methods to prevent instabilities originating from the solution process of the linear equation system(s). However, for large systems, even for sparse direct solvers, the computation times take a relevant amount of time endangering simulations with real-time framerates, as resolving the system uses a number of operations of order three. Moreover direct methods are subject to truncation and round-off errors, that are more likely for systems with a large condition number [6]. These errors propagate to future steps and thus are endangering the accuracy of the solution over time.

Iterative solvers, have several advantages over direct solvers, that are explained in detail in the following using the example of the conjugate gradient. With the symmetry of $\mathbb{A}$, solving the equation system $\mathbb{A}\mathbb{x} = \mathbb{b}$ can be expressed as $\frac{1}{2}\mathbb{A}^T\mathbb{x} + \frac{1}{2}\mathbb{A}\mathbb{x} = \mathbb{b}$ and thus a solution of the linear equation system minimizes the so called total potential

$$\Pi(\mathbb{x}) = \frac{1}{2}\mathbb{x}^T\mathbb{A}\mathbb{x} - \mathbb{x}^T\mathbb{b} \tag{3.71}$$

Given an approximation $\mathbb{x}^{(s)}$, the conjugate gradient aims on efficiently calculating a new approximation $\mathbb{x}^{(s+1)}$, that decreases the total potential rapidly.

For that, the method generates a space of linearly independent vectors, satisfying particular orthogonality properties, in which it calculates the minimal possible solution. Every time step, the norm of the residual $\mathbb{r}^{(s)} = \mathbb{b}^{(s)} - \mathbb{A}\mathbb{x}^{(s)}$ is compared to a convergence tolerance, that is set dependent on the scenario. Beyond that, additional criterions on $\|\mathbb{x}^{(s)}\|$ can be used to evaluate the convergence of the method. For the calculation of the approximations $\mathbb{x}^{(s)}$, the conjugate gradient only uses matrix-vector and vector-vector products. As a result, the implementation of this algorithm is very simple. Moreover, the orthogonality properties and the linear independency of the vector space mentioned above ensure the calculation of the solution in n iterations, where n is the size of $\mathbb{b}$. When the n iterations are performed, the conjugate gradient can be considered as a direct linear solver. In contrary to the direct solvers of the last paragraph, the conjugate gradient does not construct new matrices and uses much less memory than direct methods. Furthermore, the algorithm has the complexity $\mathcal{O}(N)$. However, the speed of convergence highly depends upon the conditioning of our linear equations system: in the worst case, the residual only reduces by a factor of $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ in every iteration [72]. The negative impact of an ill-conditioned problem can be prevented by using preconditioners: these constructed matrices are close to the inverse of the system matrix $\mathbb{A}$ and by increasing the condition of the matrix, the convergence improves. Due to the proximity to the inverse of the system matrix, preconditioners can take an important amount of the calculation time of resolving the linear equation system and compromises are necessary between a good preconditioner and a rapid construction time. Despite the valuable improvement using preconditioners, the construction of this tool needs to be adapted to the situation and thus iterative method can not be viewed as a general tool that works in all circumstances. Still, even without preconditioners, the solving process can be stopped when the aimed accuracy is achieved and thus the use of iterative solvers in real-time simulations is advantageous. For further details on the conjugate gradient, please refer to [103].

In conclusion, we like to underline, that an ill-conditioned system has negative impacts on both types of linear equation solvers: Direct solvers may yield inaccurate solutions due to round-off errors [6] and iterative solvers may endanger the real-time aspect of the simulation because of a deteriorated convergence rate or the computationally intensive construction of an adequate preconditioner.

Note, that for the corotational finite element method, the rotation matrices change the condition number of the element stiffness matrix and yield a higher stability when resolving the linear system of equations for the Newton-Raphson algorithm.

Thus the corotated finite element approach keeps the simplicity of the stress-deformation relationship in linear materials, allowing for geometric, but not for material nonlinearities. Despite of its non-linearity, the corotated finite element method does not inherit disturbing behaviours, e.g. the monotony in compression often resulting in a break down under extreme compressions, from methods like the St. Venant Kirchhoff material. Therefore, the corotational finite element method is an interesting candidate for real-time simulations and is applied in the context of this work.

The arguments mentioned in this section related to the condition number point towards the meshing aspect of the problem. Further important aspects are explained in the following.

3.5 Meshing aspects

In this section, we discuss two aspects relevant for this work: we begin with the choice of the element shape and shape functions, and pursue with the handling of hanging nodes that can occur due to topological changes.

Section 3.2 is formulated for different element types, usually tetrahedra or hexahedra with linear shape functions $N_{\triangle}$. The choice of the element shape and the shape functions e.g. their polynomial degree is of high importance as it influences the convergence rate and the weakness to numerical misbehaviours. In the following, we briefly discuss the element shape, comparing triangular elements with rectangular elements, to introduce the term *volumetric locking*. An example on tetrahedral elements underlines the importance of discussing this issue. Then, further informations are given on the impact of elevated polynomial degrees on the accuracy of our approximated solution.

As mentioned in previous sections, internal constraints such as nearly incompressible behaviour can yield a stiff behaviour which is called *volumetric locking* and occurs particularly with linear tetrahedra [122].

At this point, we illustrate the behaviour on the 2D example displayed in figure 3.8. If we consider nearly incompressible behaviour, the boundary triangles need to keep the same area and with that the same height. Thus the free nodes of boundary triangles are restricted to vertical or horizontal movements. Due to the boundary conditions on the left-hand side and the bottom, these restricted movements yield that the free point closest to the left bottom corner can not move. With similar arguments, all points of all boundary elements are fixed. That means, the boundary condition propagates

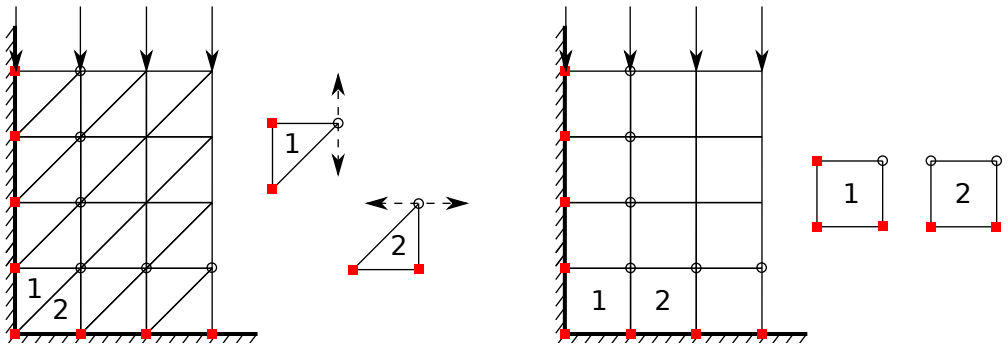


Figure 3.8: Example of locking behaviour on triangles *(left)* (idea taken from [133], Fig 11.4) and rectangles *(right)*, with fixed nodes *(red squares)*, loads *(black arrows)* and possible movements *(black arrows with dashed lines)* for the triangles; considering the rectangles for incompressible materials, the two nodes of rectangle two in the *right* figure are free to move, but the rectangle has to keep the same area

through the boundary elements and there is a new line of elements with a fixed boundary condition. Applying the arguments above repeatedly yields that the complete object is fixed.

For the rectangular elements, the free nodes of the boundary elements that are not in the corner can move keeping the same area, while the fourth node of the element in the corner is fixed for nearly incompressible materials. Again, repetead application of this argument yields a completely fixed mesh. However, in contrary to the triangular boundary elements the free nodes of the rectangular boundary elements are not restricted to a horizontal movement. Thus a decreased incompressibility has a greater effect on rectangular elements and the volumetric locking vanishes faster in the example above. Figure 3.9,

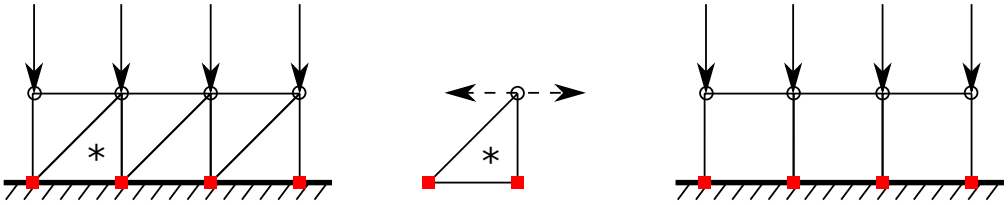


Figure 3.9: Example of locking behaviour on triangles *(left)* and rectangles *(right)*, with fixed nodes *(red squares)*, loads *(black arrows)* and possible movements *(black arrows with dashed lines)*; for incompressible materials, the triangles are locked, but the rectangles can move

displays the advantage of rectangular elements on a simple example with one boundary instead of two: triangular elements are locked, while rectangular elements can move.

Figure 3.10, shows how the first example can be expanded to the 3D case with tetrahedral elements: a beam that is fixed for all points that either have $x = 0$, $y = 0$ or $z = 0$. In order to maintain the volume of the tetrahedral

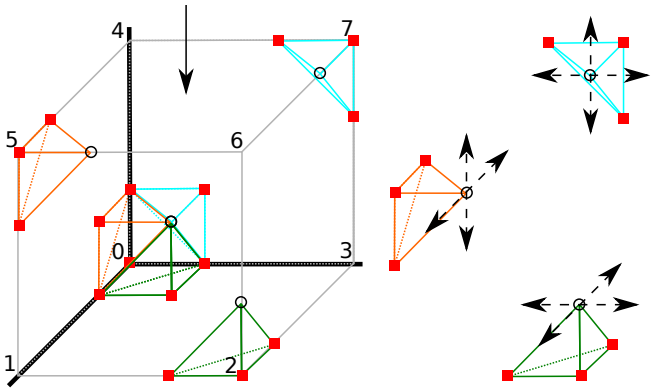


Figure 3.10: Example of locking behaviour in 3D: cube *(grey lines)* is fixed for all faces containing the node *0* and undergoes load *(black arrow)* equally distributed on top face *(face with nodes 4,5,6,7)*; plot of three different tetrahedra types with fixed nodes *(red squares)* and possible movements *(black arrows with dashed lines)*

elements at the boundary, the free node can only move in a plane. In the corner of the beam, that is fixed by three sides in this example, this restricted movement forces again a fixed node and a propagation fixing the complete mesh. We relinquish to expand the example to hexahedra, but would like to underline, that a boundary hexahedron has four free nodes, while a boundary tetrahedron only has one free node. Thus linear tetrahedra tend stronger to volumetric locking than linear hexahedra. Similarly, the second example can be expanded.

These examples are obviously constructed for the explanation of volumetric locking. In standard settings, volumetric locking decreases the rate of convergence or yields unrealistic behaviours, particularly for non-symmetric structured meshes. An example of unrealistic behaviour is displayed in figure 3.11: we consider an object in the shape of an hourglass that is fixed on the bottom and a surface force is applied at the top pulling straight up. Since there are several tetrahedral elements with three fixed nodes (surrounded by *red lines* in figure 3.11) on one side of the object, volumetric locking occurs asymmetrically and the object tilts to the front. This minimalistic, but relatively general example uses a Poisson's ratio of 0.4, showing issues when using linear tetrahedral elements. Note, that the asymmetric behaviour is observed due to the non-symmetric mesh, the upper node of the locked elements are all at the front of our hourglass – this can be easily seen when taking a look at the second image of figure 3.11. In a more general case, the locked elements are distributed symmetrically and instead of an asymmetric behaviour, the object

behaves generally stiffer and the convergence rate deteriorates.

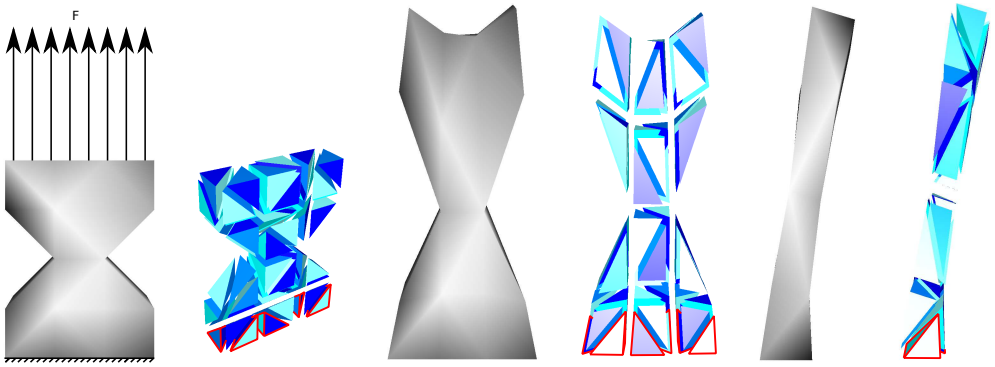


Figure 3.11: Example for an unrealistic behaviour due to volume locking: locked elements are surrounded by *red lines*; *from left to right:* initial configuration with applied forces on top and fixed boundary condition on bottom, mesh in initial configuration, final configuration front view, mesh in final configuration front view, final configuration side view, mesh in final configuration side view

In this example we used the sparse direct solver PARDISO proposed by [99]. When using an iterative solver like the conjugate gradient, a limited number of iterations prevents the solver to converge and as a result the tilting behaviour may not occur. Increasing the number of iterations yields the same tilting. Therefore, it is better to use the error-based criterium to stop the iterations of the solver. But similar arguments as for the number of iterations apply to the error-based criterium. Thus, to prevent errors comming from the linear solver and with that to ensure the correct interpretation of results, direct methods are the better choice for the early phase of implementing a new method. For further explanations related to linear solvers, please refer to section 3.4.

Another argument for the improved behaviour of linear rectangles compared to linear triangles is the enlarged space of shape functions: the forth node adds a linearly independent polynomial to the space of shape functions in the reference element. The same applies for the three dimensional case, linear hexahedra have a greater space of shape functions than tetrahedra and thus, beyond the argument of the volumetric locking, the approximation is improved. To obtain further insight, we consider the example displayed in figure 3.12, which has been presented in [19], using a Poisson's ratio of 0.3 that avoids the effect of volumetric locking. It can be observed, that the convergence rate for linear shape functions on tetrahedra and hexahedra is for both of degree 1, but there is a minor difference between the errors from the beginning favoring hexahedral elements. Beyond this result, the experiment shows, that the convergence rate is rather dependent on the polynomial degree of the shape function, e.g. elements of order two have a convergence rate of two,

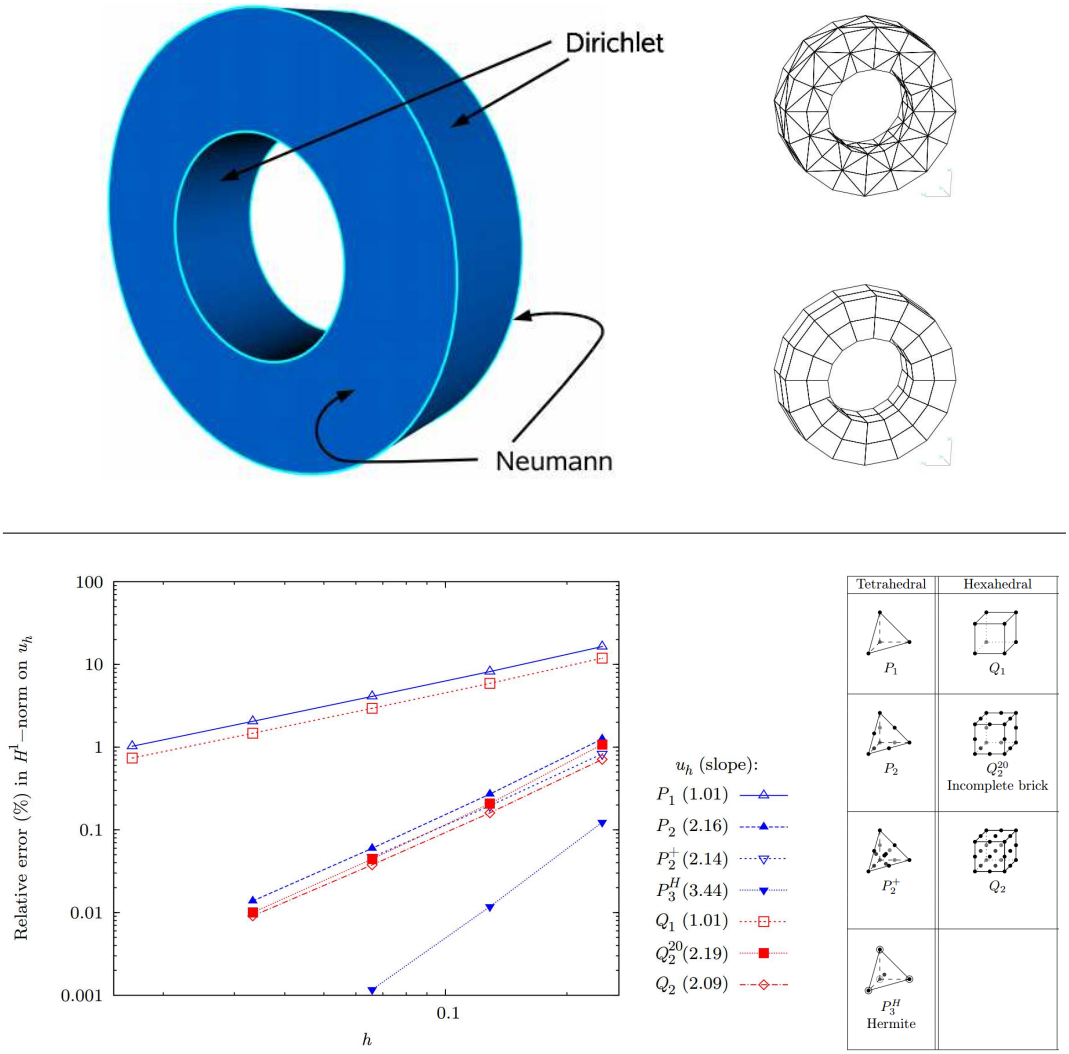


Figure 3.12: Comparison of different element types from [19] *top* setup of the example and an example of tetrahedral and hexahedral meshes; *bottom* convergence of the different element types

while the element type of order three has a convergence rate of three. Note, that elements with higher polynomial order generally speaking result in higher computational cost for each degree of freedom and each element, as the spatial integration process has a higher complexity and the system matrices show a denser behaviour. In conclusion, due to the improved accuracy combined with the simplicity second order elements are considered to be a good compromise.

In the example displayed in figure 3.13, the hollow black node only is represented in the small elements and not in the big. A case like this one can occur, when remeshing algorithms do not perform the remeshing process prop-

erly and as a result the new mesh does not fulfill the conditions for the spatial discretization presented in 3.2. According to the stiffness matrices and the theory introduced beforehand, this node could move independently of the big element, as displayed in the right figure. Thus, even if there is no gap inside an

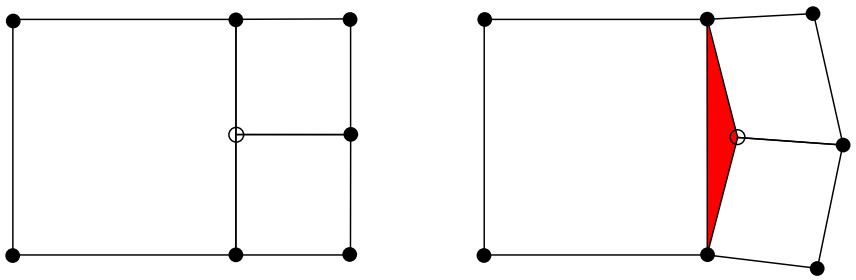


Figure 3.13: Hanging node: *Left*: initial configuration, the *hollow black dot* is only present in the smaller elements; *Right*: when staying untreated, the node in the middle *(hollow black dot)* can move freely introducing a gap between elements *(red triangle)*

object directly after a topological change, after some deformations the current discretized object may have gaps inside of the mesh and might not comply with the conditions. The literature describes these discretizations by the term *non-conformal* mesh, with so called *hanging* or *incompatible* nodes.

To avoid the behaviour of appearing gaps inside of the object due to hanging nodes, different ideas can get applied, two of them are presented in the following.
The hanging nodes need to be incorporated into the existing linear system of equations, such that forces can be propagated from the hanging nodes to the standard nodes of our mesh and furthermore, that displacements are propagated from the standard nodes to hanging nodes. This style of propagation is usually called master-slave configuration. Lagrangian multipliers provide this configuration, but increase the total number of degrees of freedom and can lead to ill-conditioned systems.
Therefore at this point we work with the standard (reduced) representation to calculate new positions, and with a representation that also contains all the hanging nodes [108]. Linking these two representations can be mathematically described using a transformation matrix $\mathbb{T}$, that contains barycentric coordinates of the hanging nodes dependent on the standard mesh nodes and relates the system matrices, the forces and the displacements by matrix-matrix or matrix-vector multiplications. The computational cost of this method remains low, as only the reduced system needs to be solved, but with a denser matrix than the initial one. Beyond that, the linking between the initial configuration with the hanging nodes needs to be performed, which goes along with additional computational costs. For further information, please refer to the detailed description in [17].

Part II

Topological changes in physics based simulation

Chapter 4

Introduction
Simulation of virtual cuts

Contents

In this chapter, we give an introduction to virtual cutting, that is used in the following chapters about linear and quadratic cutting. In the first section, we give a short state of the art, presenting the main ideas to solve this problem. The second section presents the eXtended finite element method, a method we worked on in the past. We conclude with the combinatorial maps, a particular representation of topologies that is valuable for topological changes in real-time.

4.1 Physically-based simulation of cuts in deformable bodies

This section serves as an introduction to the relevant work in the fields of topological changes resulting from cutting or fracture. In reality, cutting includes the excertion of a force and potentially a deformation before the object is separated [114]. Cuts in virtual reality are usually described as a controlled separation process performed through a directed path, while fracturing or tearing refers to the cracking or breaking of an object under the action of stress. These two processes lead to the determination of a cutting or fracture path, requiring topological updates and thus a change of the physical behaviour of the object. The following descriptions use the term virtual cutting interchangeably for cuts and fractures in simulations. Thus in this state of the art we address the topological update of deformable models without further notice on the source of the change. Virtual cutting either takes place on rigid or deformable solids, but some key challenges are emphasized in the context of deformations and surgical simulations.

We provide the reader with few essential landmarks for the cutting algorithms presented by Wu et al. [128] and Bruyns et al. [16]. Moreover, we describe more recent works, that might have importance for the developement of the research field. Our focus hereby is on the update of the topology and the simulation of deformations after cuts. For further information on different techniques for collisions after topological changes, we refer to [128, 127]. We attach importance to accurate, robust and efficient methods, with particular interest in methods that allow computations in real-time. Note, that an exact definition of the term real-time is dependent on the application: when targeting at what we call visual real-time, it is commonly referred to be above 25Hz, while real-time for haptic applications is related to 300Hz until 1000Hz. One simulation can have two components running at different speeds, to address both real-time for visual and for haptic feedback with a high efficiency [43]. In the context of this work, we are mainly interested in the real-time aspect related to visual feedback. But even with this restriction, it is difficult to evaluate the speed of different methods, as similar approaches implemented differently, e.g. on the CPU or GPU, might result in greatly differing frame rates. To prevent wrong inferences, in this overview, with computation times we provide the reader with the implementation technique, if it has been mentioned in the references.

The important building blocks for virtual cuts are: the mechanical model and material laws, the method to simulate the deformations, and the algorithm proposed for the topological update. Section 1 details our choice for the

mechanical model and the material laws. However, to prevent a restriction to this particular choice, this section describes methods for virtual cutting, which are based on different deformation models. The main deformation models used for the simulation of virtual cuts on deformable objects are the finite element method, mass spring systems and mesh-less approaches such as position based dynamics, e.g. recently combined with metaballs [76]. To adapt to topological updates, different improvements of the conventional form of these methods have been proposed and are presented in this section. Let us note that the similarity between methods like the finite element method and mass spring systems allow using the same idea for virtual cuts for both mechanical models. Moreover, re-meshing techniques like the ones presented in subsection 4.1.1 can be combined with mesh-less approaches.

The literature on virtual cuts shows that tetrahedral elements were used from the beginning until now, e.g. [69, 14, 24, 68, 63, 107, 43, 123], while virtual cuts on hexahedral elements were introduced more recently [129, 26, 126, 101]. For polyhedral meshes, there has been only a few works [124, 57]. And mesh-less methods have an increasing importance [65, 83, 88, 104], particularly combined with other approaches or as hybrid ideas [73, 74, 75, 92, 93].

4.1.1 Re-meshing approaches

Subdivisions of hexahedral elements yield in most of the cases non-conformal meshes (see subsection 3.5 for the definition) and thus re-meshing approaches for the conventional finite element method are mostly related to other element types. Thus in this chapter we present ideas proposed for tetrahedral elements, but also for polyhedral elements and finally – using the T-junctions introduced in subsection 3.5 – subdivisions of hexahedral elements [59].

When working with the conventional finite element method, ill-shaped elements lead to numerical instabilities (see the work of Shewchuck et al. [102] and subsection 3.4) and an elevated number of created nodes potentially reduces the ability of an algorithm to calculate solutions in real-time. Thus challenges of re-meshing algorithms include controlling the number of created nodes, and maintaining the quality of the inserted or updated elements. This is particularly difficult for partial, multiple and repeated cuts occurring in the same elements. Mesh creation methods, which are faced with similar challenges, remain an active research topic [105, 131, 119] and solutions are in some cases worth being considered.

One of the first works addressing virtual cutting proposes to remove the

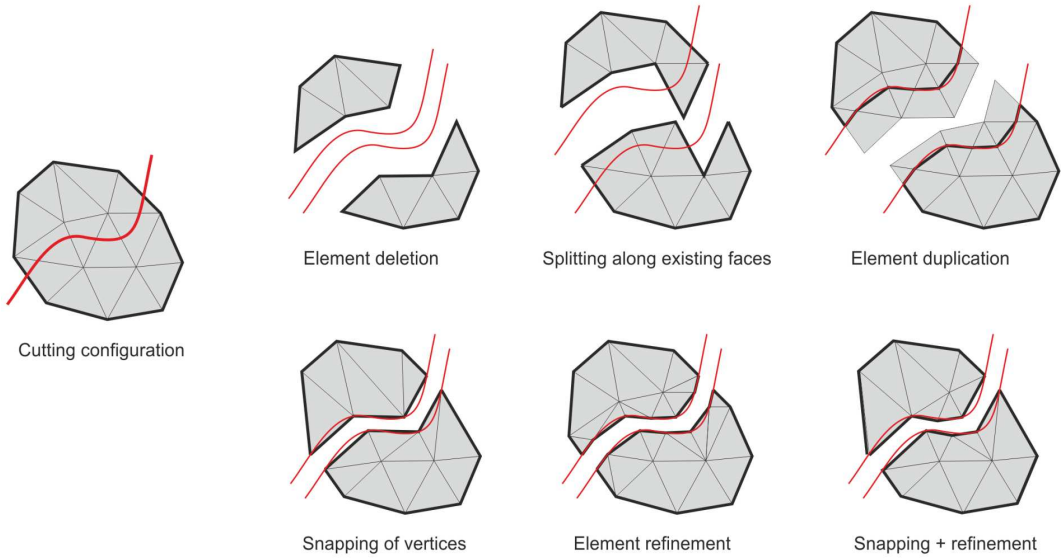


Figure 4.1: Two dimensional illustration of the different methods for the incorporation of a cut *(red curves)* into a mesh, with the boundary depicted as a *black bold line* (figure extracted from [128])

elements traversed by the cut [24]. The removed elements are in a strip of material around the cut surface, which leads to jagged cut surfaces particularly when a coarse mesh is used. Moreover the object looses volume, contradicting the conservation of mass and thus the balance of energy in the system. The approach allows for real-time simulation of the cut since no degrees of freedom are introduced and no subdivision process is required.

The snapping method presented in [68] moves the vertices of the original grid to the separation surface and disconnects the object at these vertices. It results in a close approximation of the cutting surface inside of the tetrahedral mesh adding a low number of degrees of freedom. But, since the topology remains the same, separations are restricted by the topology of the original mesh and partial cutting of an element is not possible. Moreover, the method may lead to ill-shaped elements and moving points on the surface may change the volume. Dependent on the applied technique to calculate the mass, thus contradicting the preservation of mass.

The refinement or re-meshing methods – like the ones proposed in [69, 63, 33, 13, 50] – replace a cut tetrahedron by several other tetrahedra that have their nodes on the separation surface and are disconnected at the desired location. Some of these re-meshing techniques allow for partial cuts, but usually introduce a lot more of nodes. Most of these existing methods are either restricted to planar cuts or enforce a further re-meshing when a second cut

appears. As further re-meshing potentially results in added nodes, it can slow down the simulation. In addition, separations close to the nodes may introduce ill-shaped elements, compromising the stability of the simulation.

A combination of snapping and refinement can alleviate some of the weaknesses of the methods [112]. However, since snapping algorithms do not allow for partially cut elements, partial separations have to rely on the existing re-meshing techniques.

When used in combination with the conventional FEM, re-meshing operations need to build conformal meshes. Please keep in mind, that for conformal meshes, the intersection of two elements is a sub-element of both elements - either a complete face, a complete edge or a vertex. Except from [13, 50], most the methods proposed in the literature do not meet this requirement after the topological operations. But, when augmenting the conventional FEM with T-junctions as described in subsection 3.5, these methods can be used with additional computational cost.

Similarly, multiresolution approaches for hexahedra work with T-junctions [31]. A hexahedron is subdivided using a one-to-eight split, replacing one hexahedron by eight small hexahedra that have half of the edge length of the original hexahedron and each small hexahedron contains an original node. For the small hexahedra new nodes for each edge, face and the volume are inserted. As these nodes may not be present in the neighboring elements, the non-conformal mesh needs to be treated with the theory of T-junctions. Repeating the one-to-eight split, e.g. restricted to maximally two subdivisions, these methods allow for adaptive refinement in regions where higher details in deformations are desired. Tetrahedral elements can as well be subdivided into eight, inserting new nodes on each edge, new small tetrahedra have again half of the edge length of the original tetrahedron. Four of the new tetrahedra contain one original node, while the other four only contain new nodes. Thus this operation allows to construct tetrahedral multiresolution meshes. For occuring cuts, the mesh can be repeatedly split until the highest level and then a separation along the face closest to the cut instrument [59]. Results show for the implementation on a CPU, that the computational cost for a single simulation step before topological changes and the cost for topological changes itself stay very low. However, to our understanding, no information is given for computation times after cuts, and due to a lot of added degrees of freedom, we expect these to be very high.

The re-meshing approaches presented so far start with a mesh that consists of one element type, in our cases mainly tetrahedra, but also hexahedra, and the topological changes are only allowed to introduce new elements of that

type. This limitation can yield on the one side to ill-shaped elements and thus to instable simulations or on the other side to non-conformal meshes and thus a necessity to treat the T-juctions with additional computational cost. To create a smaller number of elements, the literature proposes to remove this constraint by allowing the creation of polyhedra [124, 57]. These approaches potentially yield simplified operations, but polyhedral elements need to stay convex. Let us note, that there exist no integration schemes for polyhedral domains and the integrals necessary for the building of the stiffness matrices are only approximated. Furthermore, similar to tetrahedral elements, ill-shaped elements negatively impact the stability of the simulation after virtual cuts. But contrary to tetrahedral elements, quality measures for polyhedral elements are not existent for general cases and thus numerical instabilities are difficult to prevent.

In conclusion, the main challenge for the re-meshing approaches is to control the quality of newly inserted elements and thus to ensure the stability of the simulation after cuts.

4.1.2 Finite element methods adapted to topological changes

Each computational element (finite element) of the conventional FEM approach, interpolates the element nodes using the shape functions to obtain each position in the discretized element. Hence, due to the continuity of the shape functions, there is no possibility for material discontinuities – or separations – inside of an element. In order to allow for virtual cuts, the last section proposed re-meshing approaches, that do not separate inside the elements, but on the boundary faces of the elements. In this section we present modified finite element methods, which allow for separations inside of the elements. Hereby the differentiation between geometrical element (cell) and computational element (finite element) plays an important role as it helps to balance between speed and accuracy, while preserving the stability of the simulation. We first present the virtual node algorithm, an approach known for its stability in complex cutting scenarios, that dublicates cut elements by reusing the existing stiffness matrices. This approach is followed by the composite finite element (CFEM), an approach that uses a coarse and a fine grid and that is particularly interesting for simulating very detailed geometries and complex cuts. CFEM provides real-time frame rates and an adjustable accuracy of the simulation. Finally, we present briefly the eXtented Finite Element Method (X-FEM), while details will be provided in subsection 4.2.

The virtual node algorithm first presented in the context of topological

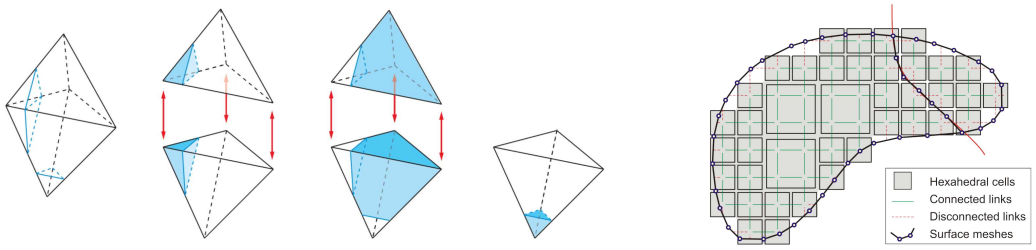


Figure 4.2: *Left*: virtual node algorithm (figure extracted from [107]) for two cuts on tetrahedral elements; *Right*: composite finite element method (figure extracted from [128]), with the coarse and the fine rectangular elements and a cut *(red curve)* separating the links *(green lines)* between the elements

changes in [62] proposes to cut an element by creating a copy (or copies) of the geometric elements (cells) and by assigning a portion of the material to each copy of the cells. Sliver elements or, in general, ill-shaped computational finite elements are avoided, as the computational element is the same as the original element. While Molino et al. [62] only allow to separate a tetrahedron into four independent parts, each containing one original node of the tetrahedron, Sifakis et al. [107] improved the method to split a tetrahedron into an arbitrary number of sub tetrahedra. And [123] extended the method to be able to pass trough the tetrahedron mesh vertices, edges and faces and the applied algorithm has a lower complexity than [107]. In surgical simulations the virtual node algorithm has been used recently by [43]. Collision and cutting are handled on the CPU, while deformations are processed by the GPU. Similarly to [123], it allows passing through sub-elements, but uses additionally a reversed node-snapping, i.e. the cutting blade is snapped to the nodes of the mesh. The implementation runs with high framerates and results in visually pleasing simulation result, allowing for a lot of different simulation scenarios. With the improved use of computational power, when cutting arises, frame rates are still in real-time, around 40Hz for the visual feedback. However, despite these obvious advantages, the virtual node algorithm copies the stiffness matrices of cut elements and thus does not simulate physically correct behaviour and is not suitable when aiming at a high accuracy.

Another approach, the composite finite element method (CFEM) [37, 97] uses a coarse uniform grid for the simulation of deformations and a fine grid for the visualization and collision. The work introduced in [26] and improved by [126, 129] proposes to connect elements in the fine grid by links, that are deleted if a separation takes place. An element of the coarse simulation grid is duplicated as soon as a separation of the complete fine grid inside of the coarse element takes place. In contrary to the virtual node algorithm, the computational elements are not a copy of the initial element, but only contain a portion of the initial element. As a result, a higher physical accuracy can be

expected when virtual cuts are performed on meshes with similar resolutions. The method is very efficient and well-adapted to visually pleasing real-time simulations. However, since a cut of a coarse element is based on the complete separation of the fine grid inside the coarse element, partial cuts are not supported. Moreover, star cuts are difficult to handle, as the links to neighboring elements is restricted by the number of hexahedral faces.

The eXtended finite element method [8] uses different representations for the geometry and the computation of the deformation: the space of the standard shape function can be extended by discontinuous shape functions. When an element is cut, additional degrees of freedoms related to these particular shape functions propagate the discontinuity inside of the element. With different enlargements of the shape function space, completely and partially cut elements with high complexity can be simulated. Let us note, that the application of boundary conditions on the cut surface needs particular treatment, as this surface is inside of the element and does not contain the nodes of the object. Further informations on the method are presented in subsection 4.2.

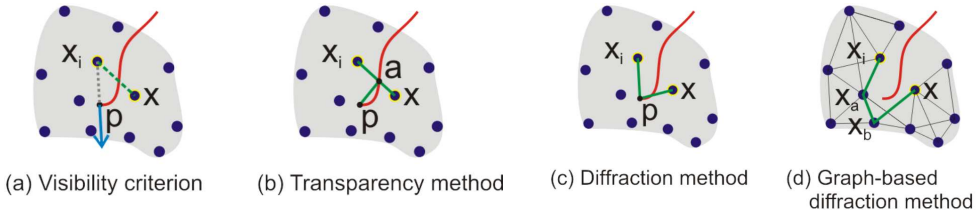
4.1.3 Mesh-free methods

In contrary to mesh-based methods, neighborhood informations for mesh-less methods [9] are computed node- and not saved element-based. Approximation functions, as well called shape functions, are defined for each node and use so called *kernel, weight* or, in the case of smooth particle hydrodynamics (SPH), *smoothing functions*. The kernel function of a node determines a domain of influence of each evaluation point dependent on its distance to the node. That means every other node in this domain influences the node in consideration and the influence decays rapidly with increasing distance using a *local support function*. When deformations occur, the domain of influence stays the same, but it is possible, that nodes go in and out of this domain. Still, for elastic behaviour, the shape functions keep the object together and the nodes can't mix arbitrarily, dependent on the grade of elasticity.

In order to account for topological changes, there are two possibilities: first, the labeling of nodes prevents to take account of nodes on the other side of the cut and second the neighborhood information and thus the kernel functions can be adapted. In the literature, the second proposition has been mainly discussed: first techniques before 1996 are described in high detail in [9], while more recent approaches were discussed in detail by [128]. We give a brief insight into the general ideas of cutting methods based on the adaption of the kernel functions, and mention recent propositions, to label nodes when cuts

occur afterwards.

The *visibility criterion* [10] sets the kernel function to zero for areas, that are not visible from the simulation node. This straight forward approach introduces an artificial discontinuity for points that are just before and behind the separation tip p, that results in a nonconvex boundary and yields negative effects on the applied Galerkin method. As a result, the convergence and stability decrease. Improvements suggested by [88] use a visibility disk, that guarantees a smoother adaptation by a continuous decay around the cut.



(a) Visibility criterion (b) Transparency method (c) Diffraction method (d) Graph-based diffraction method

Figure 4.3: Modelling of a discontinuity *(red curve)* in meshless methods (figure extracted from [128]): an object *(gray)*, sampled by simulation nodes *(blue dots) with an example node* x_i *and an evaluation point* x

Differently to the visibility criterion, the *transparency method* [71, 84, 36] considers that points are visible through a crack. Hereby, the degree of transparency is related to the distance from the crack tip to the point of intersection a between the crack and the connection of the node x_i with the evaluation point x, see figure 4.3 (b). This degree of transparency is incorporated into the weighted function and thus changes the adjancy information between two nodes.

In parallel to the transparency method, Organ et al. [71] proposed the *diffraction method*, which uses a distance that "wraps around" the cut: it is the sum of the distances from the node to the crack tip and from the crack tip to the evaluation node. Both methods presented by Organ et al are implemented only for two dimensional domains. Steinemann et al. [113] combine the approach with a visibility graph and extend it to work for three dimensionsal objects. The visibility graph contains edges between visible nodes, that are removed when intersecting with a virtual cut. Then the distance is measured along the visibile edges and inserted inside the kernel functions.

Note that, besides the good summary of Wu et al. [128], the comparison of the visibility criteria, the transparency and diffraction method in plots makes the work of Pietroni et al. [88] particularly interesting for the reader.

Recently, Shrivatava et al. [104] proposed to label nodes close to appearing cuts and to use the labeling to not take into account of an adjacency of two

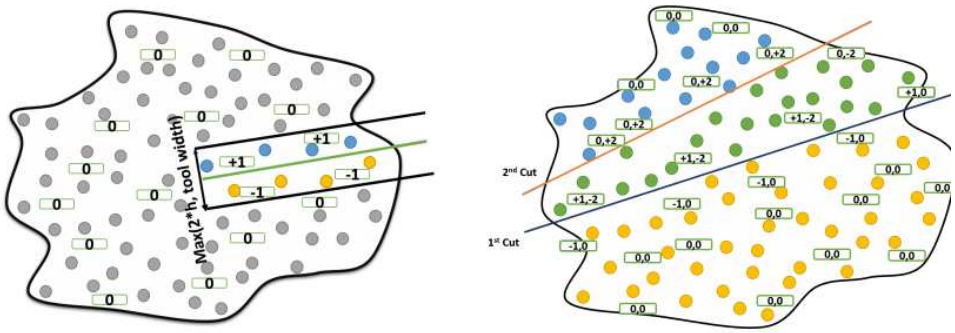


Figure 4.4: Labeling of the nodes *((pair of) numbers close to nodes)* when virtual cuts occur (figure extracted from [104])

nodes lying on different sides of the virtual cut. Different to the approaches above, the cutting plane is surrounded by a a bounding volume, whose choice is adapted to the scenario and the paper proposes to use a cylinder. Every time such a cutting tool enters the deformable object, it is assigned a number. Nodes inside the bounding volume and above (below) the cutting plane formed by the tool are assigned the positive (negative) value of this id. These labels are stored in a vector, that is compared before considering an adjacency between two nodes and thus a separation between these nodes can take place. In contrary to the visibility criterion this introduction of a virtual cut does not allow artificial discontinuity to creep in, but the work only considers planar cuts.

Since meshless methods are node-based, the surface of the simulated objects is not represented. To visualize the deformation not only at the nodes but also at the boundary of the object, the approaches have been applied together with meshes for the geometry of the object. When virtual cuts occur, new surfaces need to be constructed. In order to address this issue, the methods in their form presented above, are often combined with other mesh-based approaches [113, 84, 88]. Interesting combinations with other approaches are discussed in the following.

4.1.4 Combined approaches and other ideas

The previous chapters reveal several shortcomings or weaknesses of the presented methods, e.g. illshaped elements and thus instabilities for the remeshing techniques or no visualization of surfaces for the meshless approaches. Adapted methods or extensions often potentially alleviate these issues, some have been presented in subsection 4.1.2 and briefly at the end of subsec-

tion 4.1.3. Smart application of two ideas can combine the strengths of the methods, while circumventing the shortcomings, without a considerable amount of additional computational resources. In this section, we present some of the most recent approaches, further combinations, particularly between meshless methods for the dynamics and surface meshes, are listed in [128].

Pan et al. [73, 74, 92, 93, 75] propose to use position based dynamics (PBD) for the deformations. This is either combined with a coarse tetrahedral mesh [73, 74, 92, 93] to enforce a volume and an energy constraint [11] or combined with metaballs used as a second dynamic system taking account of the stretching potential energy [75]. To link the approaches, the PBD updates the positions of the tetrahedral mesh nodes for the first and the sphere centers of the metaballs for the second approach. For both ideas a high-resolution surface mesh visualizes the object and is updated by subdivision when a cut occurs.

Virtual cuts in [73, 74, 92, 93] result in a re-meshing of the coarse tetrahedral elements based on the scheme proposed e.g. in [63], which essentially represents the five possible subdivisions of a tetrahedral element. The barycentric mapping between the volume and the surface mesh is updated and constraints are applied on the newly inserted elements. Incision surfaces are identified by the intersections between the existing boundary surface mesh, the sweeping surface of the cutting tool and the volume mesh.

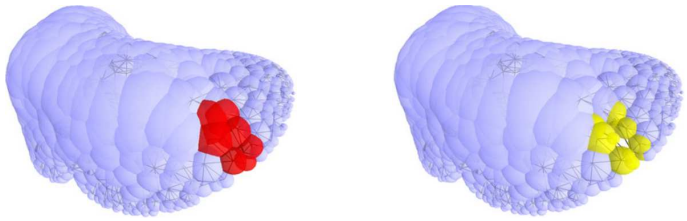


Figure 4.5: Update of the metaballs when a cut occurs: *left*: before the cut, with deleted metaballs in *red*; *right*: after the cut leaves the object, with created metaballs in *yellow* (figure extracted from [75])

In [75], intersected metaballs are replaced by their sphere centers, that is simulated by the PBD until the instrument leaves the object and new metaballs are packed into the area respecting the adapted topology, see figure 4.5. Similarly to the approach above, the mapping between the volumetric representation by the metaballs and the surface representation updates the position of visualization for the continuation of the simulation. A constraint delaunay triangulation provides the cutting surface.

4.2 Adapted function space – the extended finite element method

The eXtended finite element method is particularly interesting, as it provides tools to handle elements that are completely and partially cut with complex geometries. In this section, we discuss briefly our work on this topic, while further details can be found in [77].

For simplicity, we only consider one cut, we define the volume above the cut $\Omega_+ \subset B_h$ and below the cut $\Omega_+ \subset B_h$. Then the idea of the eXtended FEM can be stated as follows: Elements intersected with the cutting surface S are identified and classified as completely and partially cut. Nodes of partially cut element get *branch* enriched, while nodes completely cut elements are *sign* enriched.

If a node of a sign enriched element is already branch enriched by another element, the sign enrichment is not performed. An enrichment of a node results in new degrees of freedom, equivalent to three degrees of freedom, or one node, for a sign enriched and twelve degrees of freedom, or four nodes, for a branch enriched node. These degrees of freedom are attached to the existing displacement vector, first inserting the n(S) sign enriched and then the n(B) branch enriched nodes.Then the connectivities c_e of the elements are updated by adding first the sign enriched node ids and then the branch enriched node ids. Note, that a sign enrichment of a node is restricted to the sign enriched element and thus only changes the connectivity of the sign enriched element itself. In contrary, the branch enrichment of a node k has an effect on all adjacent elements $v(k)$ and even the connectivity of adjacent unenriched elements needs to be updated. Thus a topological representation needs to supply this neighborhood information. Now we can formulate the shape functions for a sign enriched node with the global id $k = c_\mathrm{e}(l), l > 4$ inside a completely cut element e as

$$\Phi_k(\boldsymbol{X}) = N_{\mathrm{e},l \bmod 4}(\boldsymbol{X})\psi_l(\boldsymbol{X}), \quad \forall \boldsymbol{X} \in \Omega_\mathrm{e}, \tag{4.1}$$

$$\text{where } \psi_l(\boldsymbol{X}) = \frac{H(\boldsymbol{X}) - H_l}{2} \quad \text{with}$$

$$H(\boldsymbol{X}) = \begin{cases} 1, & \boldsymbol{X} \in \Omega_+ \\ -1, & \boldsymbol{X} \in \Omega_- \end{cases}, \text{ and } H_l = H(\mathbf{P}_{c_\mathrm{e}(l)}). \tag{4.2}$$

In the literature, H is called the *Heaviside function* and ψ_l the *shifted Heaviside function*. For more information on the choice of the enrichment function ψ, please refer to Schoch et al. [100].

Now, defining the distance of a point $\boldsymbol{X}$ to the cut front by the radius r

and the angular difference to the direction of the cut by θ, the shape function for branch enriched nodes with the id $k = c_{\mathrm{e}}(l), l > 4$ is defined as

$$\Phi_k(\boldsymbol{X}) = N_l(\boldsymbol{X}) = F_n(r,\theta) N_{\mathrm{e},l \bmod 4}(\boldsymbol{X}) \tag{4.3}$$

$$F(r,\theta) = \sqrt{r}\left(\sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\sin(\theta), \cos\left(\frac{\theta}{2}\right)\sin(\theta)\right)^T \tag{4.4}$$

Here, we use the id $n = (k - n(\mathcal{P}) - n(\mathrm{S})) \bmod 4$ of the *asymptotic crack tip function* (ACTF) F.

With these definitions, the discretized displacement function writes

$$\boldsymbol{U}_h = \mathbf{u}_k \Phi_k = \mathbf{u}_{c_{\mathrm{e}}(l)} N_{\mathrm{e},l} \tag{4.5}$$

When using the derivation as in subsection 3.2, using the linear elasticity, the stiffness matrix is expressed as a sum over the element domain and the derivatives of the shape functions. Most implementations of the conventional FEM apply the fast and stable *Gauss integration*. The Gauss integration is advantageous for the conventional FEM, since an exact numerical integration of polynomials with low degrees over domains like a tetrahedron can be achieved. The ACTFs are trigonometric functions wherefore the Gauss integration can only approximate the solution of their integrals. Furthermore, the discontinuity of the Heaviside function yields to inaccuracy, since an integration over arbitrary three dimensional domains is necessary. In order to overcome the negative side affects, we expand the idea of the Gauss integration to replacing the integral by a sum of function values multiplied with integration weights.

The main contributions of our work [77] are:

- Application of the X-FEM to the simulation of cutting in soft tissue – with almost arbitrary cuts
- Capability of simulating completely and partially cut elements
- An evaluation that demonstrates the superior accuracy in comparison with the conventional FEM
- Very precise and application-related description of the X-FEM [77].
- An open source X-FEM implementation[1]

[1]Code hosting platform bitbucket - https://bitbucket.org.
Download command: git clone https://chrijopa@bitbucket.org/chrijopa/xfem-in-cpp.git

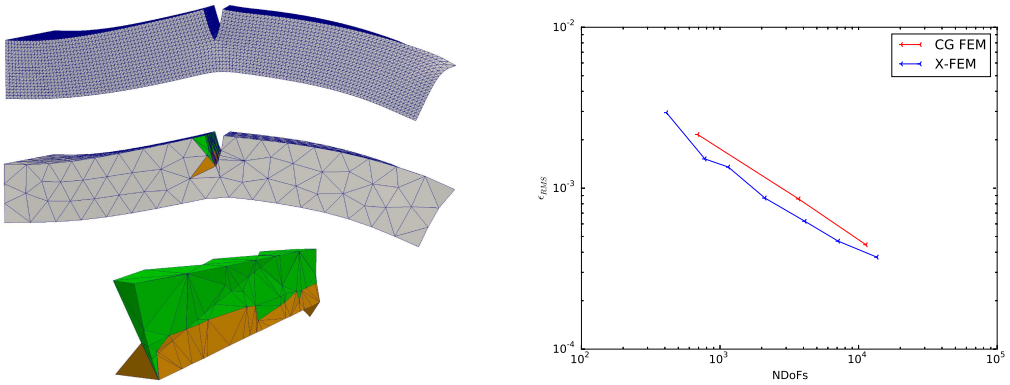


Figure 4.6: Convergence analysis displaying the root mean square error *(right)* of the eXtended FEM compared to a perfectly aligned conventional FEM solution on an example of a cut in a beam (figure extracted from [77]): reference solution *(left top)*, solution with 1810 elements and 595 nodes *(left middle)*, zoom on partially *(orange)* and completely *(green)* cut elements *(left bottom)*

The implementation has been evaluated on an example of a beam, that is cut half way through, see figure A.24. Solutions using the conventional finite element method are perfectly aligned to the cut surface and can be considered as the best possible approximation of the method in this form. The extended finite element method uses a randomly created mesh and thus is not optimized for the cut geometry. Figure A.24, shows that even in this disadvantageous scenario for the X-FEM, the method gives better results.

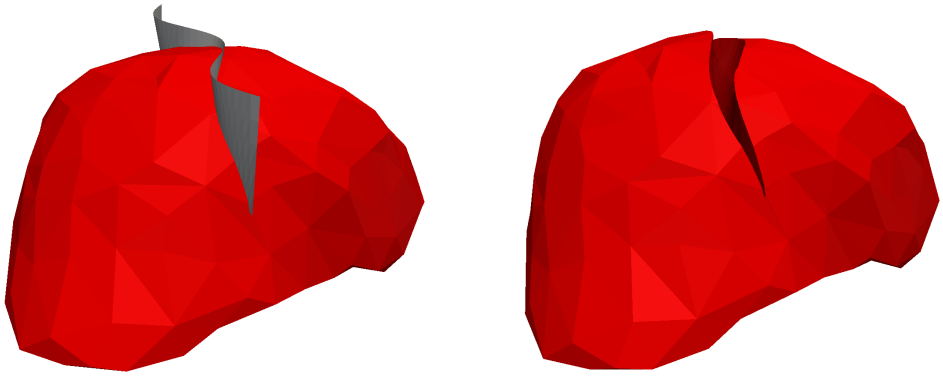


Figure 4.7: A liver with 888 linear tetrahedral elements is cut by a sinusoidal blade (figure extracted from [77]): undeformed with the cutting surface *(left)*, and deformation after the cut *(right)*

Furthermore we applied the method to a medical example, cutting a volumetric mesh of a liver with a complex cut, see figure 4.7.

While the implementation provided with our publication [77] showed the

advantage of the approach in the examples above, no evaluations on the speed of the method are provided, as we only consider static simulations. As mentioned before, the shape functions defined in equation (4.3) are not polynomials with a finite degree and the integration domains for completely and partially cut elements are potentially complex due to the complexity of cuts such as in figure 4.7. While the integration in completely cut elements can be handled with algorithms such as [61] accurately, the integrals in partially cut elements can only be approximated. Our expansion of the Gauss integration has high computational cost and we do not expect such an implementation to run in real-time. Moreover, the cut surfaces inside a cut element do not maintain the Kronecker delta property, and thus a treatment of boundary conditions on these surfaces needs particular methods. Similarly, multiple cuts need an improved method, that already has a high complexity on two dimensional examples [47]. Therefore, despite the promising results, our work was directed towards an implementation of a re-meshing algorithm (see chapter 5), that can use standard integration methods, boundary conditions and does not need new methods to deal with multiple cuts.

4.3 Mesh topology for virtual cutting

The overview of the last sections helps to define the main properties that a topology confronted with appearing cuts should have. In this section, we briefly state these properties and we show, that the conventional representation with the connectivity c_e of each element e falls short to meet the requirements. Then we describe the combinatorial map topology and show how it fullfills the needs for topological changes and point to different implementation of this topological representation.

In the first part on real-time simulations of soft bodies and the previous overview of cutting methods, several different representations of element shapes have been used: triangles for surfaces, tetrahedra, hexahedra and even polyhedra for volumes. In finite element simulations these element shapes exist side by side and even arise mixed in one mesh. Applying different degrees of polynomial approximations on the element shapes yield different topological structures, e.g. control points of quadratic shape functions can be assigned to an edge of a quadratic triangle or tetrahedron (see chapter 6). A separation of the topology, i.e. the connectivity and neighborhood information, and the information on an entity or cell of the topology is valuable. Informations like a control point on an edge are attached to every entity (or cell) of the topology, including e.g.

−Vertex: Position $\mathbf{P}_k$, displacement $\mathbf{u}_k$, velocity $\dot{\mathbf{u}}_k$, ...

–Edge: cut or uncut, control points for quadratic shape functions, ...
–Face: T-junctions [59], links between two hexahedral elements [129], ...
–Volume: cut or uncut (see subsection 4.2), ...

Similarly to the informations on cells, the neighborhood information between the different dimensions of the cells are necessary. Examples for necessary neighborhood informations include, but are not restricted to

–Volumes around a vertex $v(k)$: for the update of the connectivity c_e of all the elements around a branch enriched node (see section 4.2)
–Volumes around an edge $v(e)$: to obtain all cut elements from a cut edge
–Volumes around a face $v(f)$: for the T-junctions used by [59] and e.g. for the visualization of an objects boundary

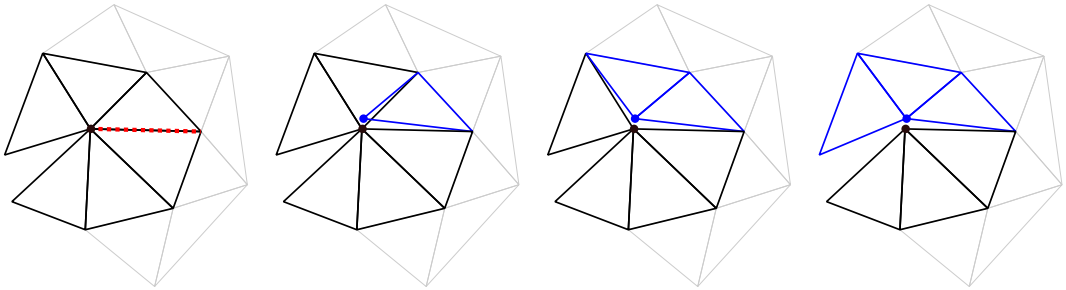


Figure 4.8: Separation process on the topological level for linear elements: intended separation direction *(red dotted line/curve)* and node to be duplicated *(black dot)*, duplication of node *(blue dot)*: before topological change *(left)*, at node duplication and the update of direct neighbors of the edge *(middle left)* and the update of elements sharing the node *(middle right and right)*

When topological changes arise, an indication of a face inside a volumetric mesh and an edge in a surfacic mesh should be sufficient, such that the duplication of a node and the update of the element connectivities are performed, see figure 4.8. The requirements mentioned above should be updated automatically in the case of topological operations, i.e. the entities themselves, the data on the entities and particularly the connectivity information. Where, from an implementation point of view, it is advantageous, if topological operations on a particular element shape can be applied for any degree of polynomial approximation without major changes.

The conventional topological representation, based on the connectivity c_e of every element, usually saves informations like vertex positions, velocity and others separately in vectors related to the id of the vertices. For a volumetric mesh, edges and faces need to be introduced using the connectivity of the volumetric element. Then, necessary neighborhood informations similar to the ones above are calculated. In simulations without topological changes, the conventional representation of the topology only has high computational

costs for the generation of these structures at the beginning of a simulation. However, already simple scenarios like the one presented in figure 4.8 can be cumbersome to implement. It is challenging to know, when a node has to be duplicated and to which elements the duplicate is attached, particularly in the volumetric case. Moreover, when topological changes occur as a result of virtual cuts, the costs of reinitializing the topological structure endanger the real-time aspect of the simulation. And implemented topological operations are difficult to reuse, as the conventional topological representation only has limited information on the element shape.

Introduced by Edmonds [27], the combinatorial representation uses so called darts and bindings between the darts to construct a topology. In *oriented combinatorial maps*, the combination of the bindings allows retrieving all the informations on the topology, including the vertex, edge, face and volume ids of and around any other entity. For a simple separation like the one in figure 4.8, a face (or the edge for surfaces) where a separation takes place has to be provided to the topology. Then the topological structure deletes the bindings between the two volumes (or respectively two faces for surfaces), adds automatically a node if necessary and updates the connectivity information. That means, the structure is maintained after topological operations, thus costly reconstructions of neighborhood informations are prevented. Arbitrary element shapes can be represented and topological operations are defined dependent on the element shape. Saving informations on the polynomial degrees, e.g. the control nodes for quadratic elements on the data of an edge, thus allows reusing these operations without major implementation times.

The CGoGN[2] library provides a very efficient implementation of combinatorial maps, due to the fast processing times and versatility of implemented operations [51]. Data structures are saved as attributes related to each cell and updated efficiently, e.g. when a cell is deleted, the entry in the attributes is skipped and when a new cell of the same dimension is added, this entry is reused. This implementation has been used for the main contributions of this work cutting volumetric elements, that are presented in the following chapters: first we address the simulation of cuts on linear tetrahedral elements in chapter 5. The topological operations are based on the theory of combinatorial maps, that allow a smart extension to quadratic elements presented in chapter 6.

[2]http://cgogn.unistra.fr/

Chapter 5

Cutting linear elements

Contents

In this chapter, we present a re-meshing method that introduces a separation surface inside a mesh of linear tetrahedra with two main benefits: First, the number of inserted nodes (vertices) and elements (tetrahedra) is kept as low as possible to maintain real-time performances. Secondly, the quality of the generated mesh is controlled, i.e. the shape of the introduced elements do not hinder the numerical aspects of the FEM method. Our re-meshing algorithm creates smart approximations of the cutting surface, can deal with complex cutting scenarios and is compatible with classical snapping approaches.

The work described in this chapter has been published as a conference [81] and a journal paper [80]. These publications serve as the main orientation for the chapter.

5.1 Virtual cutting of deformable objects based on efficient topological operations

As virtual cuts occur in the simulation, the mesh supporting the FEM model has to be adapted so that the simulation takes account of the expected phenomena and reproduces what we call the *mesh separation*. The *separating surface* S may be defined as the trajectory of a cutting tool or computed by a fracture algorithm describing the occurrences of tearing or shearing in the material. The tetrahedral elements traversed by this surface have to be cut, refined or rearranged to reflect the new physical state.

The first step of our method is the sampling or detection of the separation surface at the level of the edges of the FEM mesh. For each edge e of the mesh, we compute or estimate a cutting or breaking point and the normal of the separation surface at this point. The surface S may also pass through some of the vertices of the mesh. This special case is addressed in section 5.1.3.

The second step consists in a local re-meshing of the model. Several refinement have been proposed in the computer graphics field. Most of them insert too many vertices [12, 94] or introduce non tetrahedral elements [98]. The idea we defend here consists in the building of a set of triangles τ_j that approximate the separation surface. These triangles are emerging from the initial mesh after splits and flips on the tetrahedral elements.

The re-meshing algorithm we present is an extension of the $\sqrt{3}$-subdivision scheme introduced by [18]. This approach soundly and efficiently handles partial cuts, the emergence and propagation of cracks and the existence of overlapping or crossing between separation surfaces.

5.1.1 Approximating the separation surface

An edge that crosses the separation surface indicates that incident volumes are cut by the surface S. To take account of that, we introduce vertices in the tetrahedra incident to crossed edges. The newly introduced vertices, denoted $\mathbf{p}_k$, are positioned on the surface S. The next steps proceed with the insertion of edges between the $\mathbf{p}_k$ and finally the insertion of the triangles τ_j connecting the newly introduced vertices.

Inserting vertices: The first step consists in subdividing every tetrahedron that is crossed, even partially, by the separation surface S. We use the *1-4 split*, replacing the initial tetrahedron by four new tetrahedra sharing the inserted vertex $\mathbf{p}_k$ (see figure 5.1).

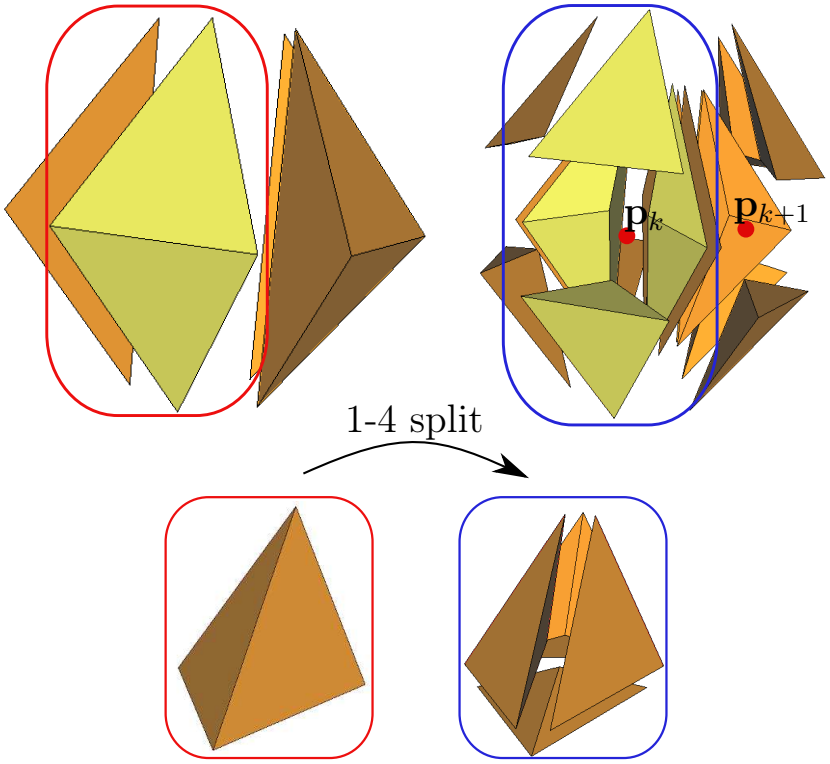


Figure 5.1: A set of tetrahedra incident to an edge before *(top left)* and after *(top right)* the 1-4 split; *(bottom)* 1-4 split of a tetrahedron.

The 1-4 split allows for an arbitrary choice of the vertex position inside the volumes. The intersections of the edges with the separation surface are computed: If the separation surface crosses three or more edges e on a tetrahedron, we insert the new vertex at the barycenter of the intersections. If only one or two edge(s) of a tetrahedron are crossed – in case of partial cut – we use intersecting points between the edges of the tetrahedron and the extension of the separation surface to compute the barycenter. The position of the inserted point belongs to one of the most important parts of our technique and is discussed in detail in section 5.3. In case of a further cut of the tetrahedron, the vertices are moved to the updated barycenter following the same rules.

Inserting edges: The second step aims at creating edges between the inserted vertices. Let us consider the tetrahedra generated by the 1-4 splits around the initial edges e. They form a sequence of pairwise adjacent tetrahedra. Each pair of tetrahedra v_k,v_{k+1} share a face f, incident to e, and thus three vertices. Their fourth vertices are respectively $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$.

We perform a 2-3 flip around the face f, replacing two adjacent tetrahedra by three tetrahedra with the same vertices. The shared face is deleted, but the three new tetrahedra share the edge $\{\mathbf{p}_k, \mathbf{p}_{k+1}\}$. This way, the faces initially incident to e are replaced by a sequence of edges. These edges form a closed polygon lying on the surface S (see figure 5.2).

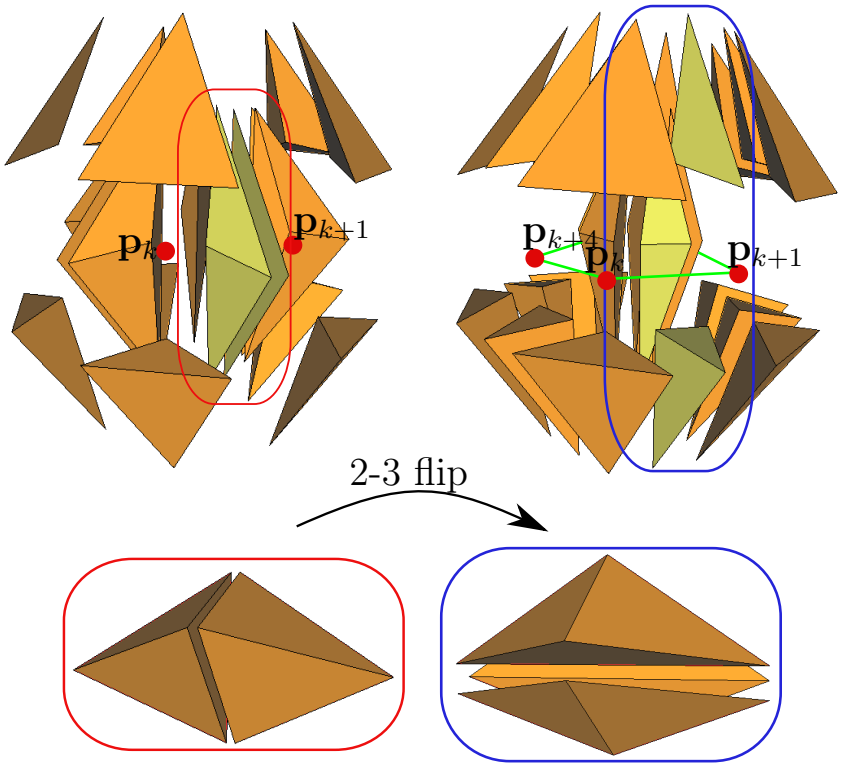


Figure 5.2: A set of tetrahedra before *(top left)* and after *(top right)* the 2-3 flips. Flipping the faces incident to e creates a closed polygon $\{\mathbf{p}_k, \ldots, \mathbf{p}_{k+n(v(e))}\}$ *(green)*; *(bottom)* 2-3 flip of two tetrahedra.

In order to perform the 2-3 flip, the connection between $\{\mathbf{p}_k, \mathbf{p}_{k+1}\}$ has to be completely inside the two neighboring tetrahedra. The vertices $\{\mathbf{p}_k, \mathbf{p}_{k+1}\}$ inserted by the 1-4 splits both depend on the intersection of the separation surface with the tetrahedras' edges. This choice helps to ensure that the 2-3 flip can be performed.

Inserting triangles: Each crossed edge e is now surrounded by a set of $n(v(e))$ tetrahedra that contain the vertices of the crossed edge and two points of the polygon $\{\mathbf{p}_k, \ldots, \mathbf{p}_{k+n(v(e))}\}$. The last step builds a set of $(n(v(e)) - 2)$ triangles τ_j by triangulating the polygon $\{\mathbf{p}_k, \ldots, \mathbf{p}_{k+n(v(e))}\}$.

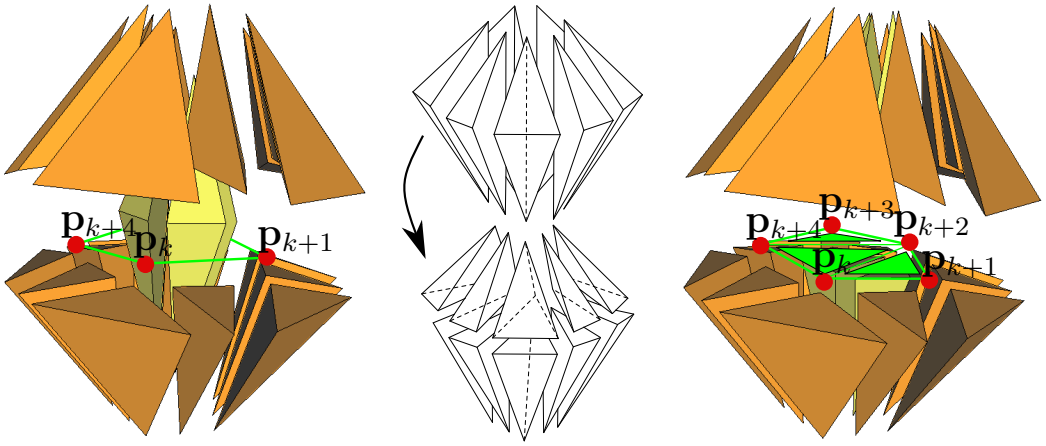


Figure 5.3: *(left)* tetrahedra surrounding the cut edge *(yellow)*; *(middle)* general flip; *(right)* triangulation of the separation surface *(green triangles)*, showing the tetrahedra below the cut

As the vertices $\{\mathbf{p}_k\}$ are placed to sample the separation surface, the triangles τ_j approximate the separation surface by construction. The $n(v(e))$ tetrahedra surrounding the edge are replaced by $2(n(v(e)) - 2)$ tetrahedra defined by the three vertices of each τ_j and by one of the vertices of e (figure 5.3). In the literature, this operation is called a *general flip* or an *edge removal*.

5.1.2 Consideration of the boundary

When the separation surface crosses the boundary of the simulated object, a specific refinement of the boundary tetrahedra is needed to build this *cut line*. The additional refinement combines the previous tetrahedral re-meshing with an extended $\sqrt{3}$-refinement of the boundary triangles.

Re-meshing the boundary surface: In this paragraph, we describe the refinement of triangles of the boundary surface, ignoring the tetrahedra behind them (figure 5.4). The edges that are crossed by the cut line are first selected. New vertices are inserted on the cut line in the adjacent triangles that are split into three. Then the selected edges, except the boundary ones (at the left),

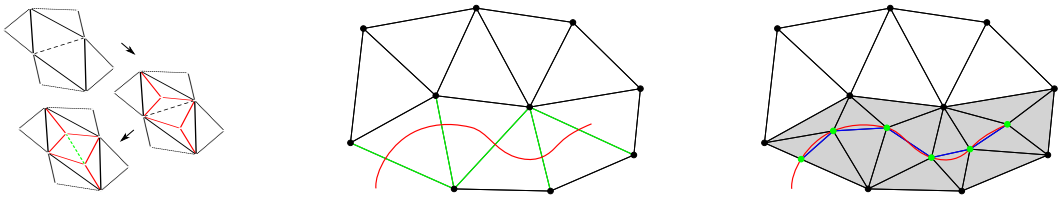


Figure 5.4: *(left)* an outline of the $\sqrt{3}$-refinement on two triangles; *(middle)* the edges crossed by the cut line are selected; *(right)* the adjacent triangles are refined to approximate the curve. The extremities of the cut line are obtained with the insertion of vertices on the boundary edges.

are flipped to link the new vertices. The flipped edges define a polygonal line that smartly approximates the cut line. The boundary edges are finally split at the crossing point of the cut line with an edge with a feature.

Re-meshing the boundary tetrahedra: The refinement of boundary tetrahedra follows this scheme, but uses volumetric operators replacing the operators of the two dimensional case. Similar to the case for edges inside the object, a crossed edge induces a 1-4 split. Then we deal separately with the re-meshing of the boundary tetrahedra: The 1-3 split of triangles is replaced by a 1-3 split extended to tetrahedra shown in figure 5.5 (left). We obtain a polygon of $\{\mathbf{p}_k\}$ with vertices on the separation, but in contrast to the inner edges, that polygon is opened, between the points inserted by the 1-3 split.

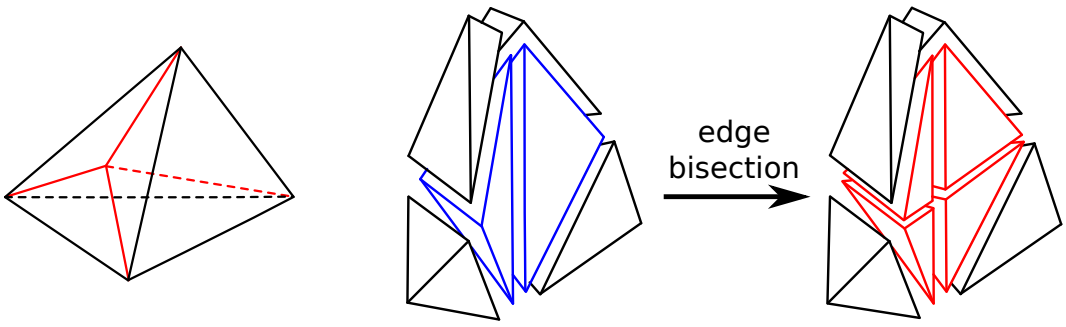


Figure 5.5: *(left)* 1-3 split of a tetrahedron: a vertex is inserted in the boundary triangle; *(right)* bisection of a boundary edge: the incident tetrahedra are split.

Since more than two tetrahedra may be incident to a boundary edge, the replacement of the edge flip on the boundary faces is not straightforward. We use the general edge flip illustrated in figure 5.3 (left) to replace the $n(v(e))$ tetrahedra incident to the boundary edge with $2(n(v(e))-2)$ tetrahedra. This operator inserts an edge that closes the polygon between the vertices inserted

into the two boundary triangles as shown in figure 5.6 (middle top) and approximates the separation surface by a set of triangles τ_j.

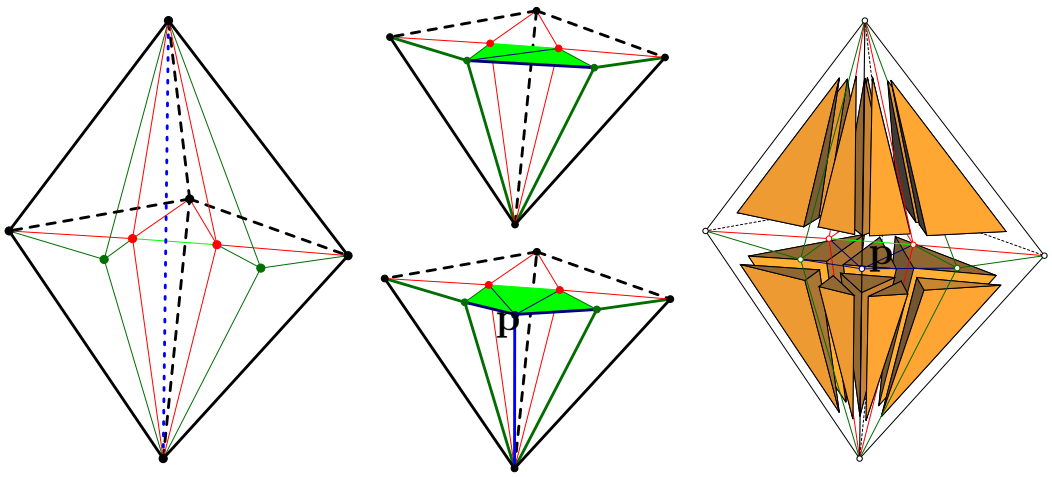


Figure 5.6: Re-meshing around a boundary edge *(dotted blue line)*. *(left)* 1-4 splits are applied to all tetrahedra *(new edges are plot as red lines)*, next 1-3 splits are applied on the boundary tetrahedra only *(new edges are plot as green lines)*; then, the boundary edge is replaced by a general flip *(middle top)* or split to maintain a desired feature *(middle bottom)*; *(right)* the final re-meshing on the boundary of the object after a split of the edge.

The general flip removes the initial boundary edge, which may locally smooth the surface. If this edge belongs to an important feature of the mesh this could be undesirable. The dihedral angle of the boundary faces incident to the cut edge helps to determine, whether the edge belongs to an important feature: for angles close to 180° a general flip can be performed without negative impact, while a general flip on smaller angles changes the surface and thus the volume of the object. In that case, we replace the general flip by the bisection of the boundary edge illustrated on figure 5.5 (right). The bisection introduces a new vertex $\mathbf{p}$ at the intersection of the edge with the separation surface. The new vertex $\mathbf{p}$ is used to close the polygon. Again, we end up with tetrahedra, that approximate the separation surface with a set of triangles τ_j. This case is illustrated in figure 5.6 (middle bottom) and (right).

5.1.3 Disconnection of the mesh

At this point, the separation surface S is approximated with inserted vertices, edges and triangles built from the set of crossed edges. This approach is tailored to the case where S crosses the edges far away from their extremities. When the surface S passes near vertices of the FEM mesh, it may be desir-

able to move these vertices to the surface – or *snap* them – instead of inserting unnecessary elements and vertices. The presented re-meshing approach is compatible with classical snapping strategies, like the one presented by [68]. Their combination is presented hereafter, before the disconnection step is discussed.

Snapping to the separation surface: Intersections of the separation surface S with an edge $\Omega_e(t)$ either yield a re-meshing or result in a movement of one of the existing vertices $k(e)$ to the separation surface. In order to decide which alternative is chosen, we introduce a parameter ϵ.

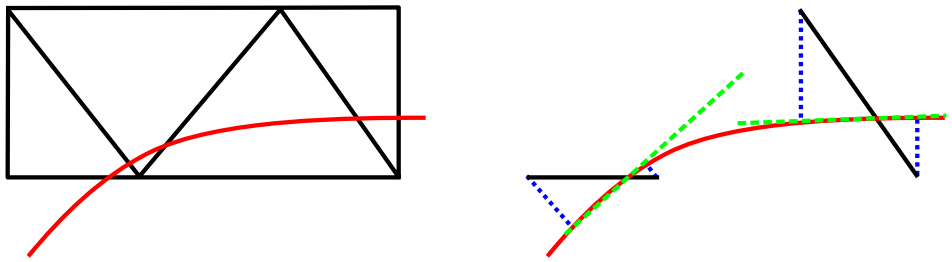


Figure 5.7: *Left:* Cut *(red curve)* of a triangular mesh *(black lines)*; *Right:* Intersected edges *(black lines)*, with the slope of the cut curve at the intersection *(dotted green lines)*, and the distances to the edge vertices *(dotted blue lines)*

If an intersection occurs, we obtain the position $\mathbf{p}_{S\cap\Omega_e(t)}$ of the intersection and we save the normal $\mathbf{n}_S(\mathbf{p}_{S\cap\Omega_e(t)})$ of the separation surface S at the intersection. Then we calculate the distance of the vertices of e to the plane through the point $\mathbf{p}_{S\cap\Omega_e(t)}$ with the normal $\mathbf{n}_S(\mathbf{p}_{S\cap\Omega_e(t)})$. We compare the distance to the threshold ϵ multiplied by a mesh parameter – in our case the average length of the edges in the rest position of the mesh. Finally, a vertex is snapped – or moved – to $\mathbf{p}_{S\cap\Omega_e(t)}$, if its distance is smaller than the distance of the other vertex on the edge and is below the value mentioned above. We discuss the choice of the threshold in the section on the results.

Unsewing over the separation surface: To allow for a fine management of the topological relations linking cells (vertices, edges, faces, volumes) in the mesh, we use the idea of combinatorial maps presented in section 4.3. An efficient implementation of the data structure based on this model is proposed in the CGoGN library [51]. It provides tools to manipulate the topology of meshes and an attribute manager for the different cells.

An important property of combinatorial maps is that they explicitly represent the volumetric binding between the two faces of adjacent tetrahedra. In order to separate our mesh at the separation surface S we unsew that link as

soon as a face f is known to be on the separation surface S. To track this information during the re-meshing, we use markers in the way described in the following paragraph.

Each vertex introduced on the separation surface (by the previously presented operators) is marked. The snapped vertices are marked the same way. As soon as the three vertices of a face are marked, its volumetric binding is deleted. This operations results in a hole between the two elements sharing this face, but, at this stage, their vertices are still connected. The faces are disconnected one after the others what makes the holes grow in a greedy manner.

Finally, as soon as a cycle of adjacent faces, incident to a same vertex, are disconnected, the vertex is separated into two vertices (i.e. it is duplicated) triggering an update of the FEM part. This last condition is automatically checked by the CGoGN library, see section 4.3 for further information.

5.1.4 Handling successive cuts

After carrying out the topological changes introduced in the preceding subsections, we obtain a volumetric mesh naturally supporting an occurrence of new separations. This allows our method to separate initial and newly inserted edges several times.

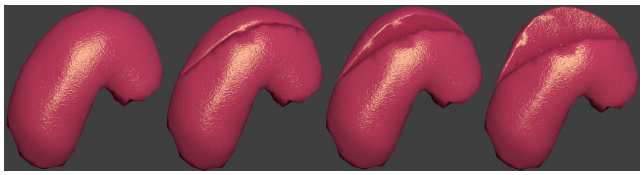
However, each additional separation of an edge reduces the edge length – which can result in ill-shaped elements or edges with greatly differing lengths. Our algorithm prevents this negative impact using the threshold ϵ introduced in the last subsection: as soon as the edge length decreases beyond the threshold multiplied by the average length of the edges, the snapping of vertices is performed, avoiding a further subdivision and additional computational cost.

Beyond that, our algorithm can be used iteratively to incorporate a separation surface into a volumetric mesh with higher detail: This can be achieved by using the operations mentioned beforehand repetitively – but inserting the nodes of the 1-4 split at the barycenter of the tetrahedron – in the elements that intersect with the separation surface. Finally, at the highest level of refinement, the vertices of the 1-4 split are inserted on the separation surface, allowing a detailed representation of the separation surface inside the volumetric mesh. Our approach is specifically interesting in the context of iterative refinement, since a refinement around an edge only has an impact on the tetrahedra around the edge, but not on their neighbors – an advantage that has

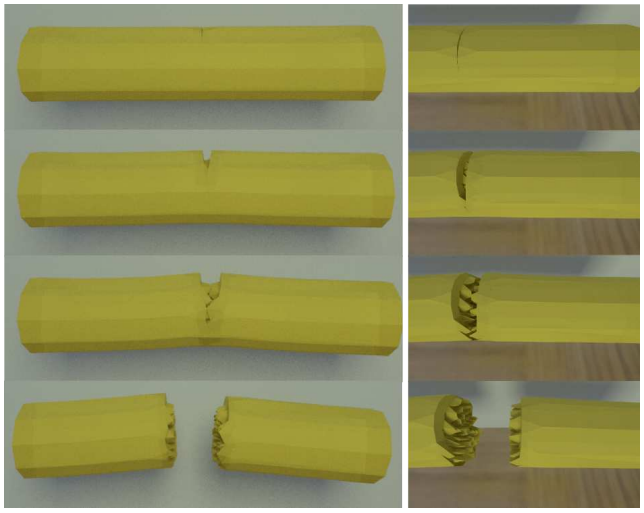
also been presented in [50]. However the nodes inserted by the iterative refinement that are not on the separation surface go along with an additional computational cost, that can endanger the real-time aspect of our approach. Therefore, we use the iterative approach in the context of this work.

5.2 Results and impact on numerical stability

(a) Cut in knee surgery, with the opening of the wound



(b) Advancement of a cut in liver surgery



(c) A cut that results in a propagating crack

Figure 5.8: Virtual cuts produced with our method.

We implemented different examples, based on the proposed method, using the open source framework SOFA [29]. We perform interactive cuts on soft tissues models: for example on a knee in figure A.26a or on a liver in figure A.26b.

In addition, we show that a combination of cutting and breaking of a cylinder can be realized using our method, see figure A.26d.

The quality of the re-meshing – regarding real-time FEM simulation – can be measured with the number of nodes and the *condition number* of the system that has to be solved (for further information of the condition number on the system, see section 3.4). Thus, we use these two measures to demonstrate the benefits of our method.

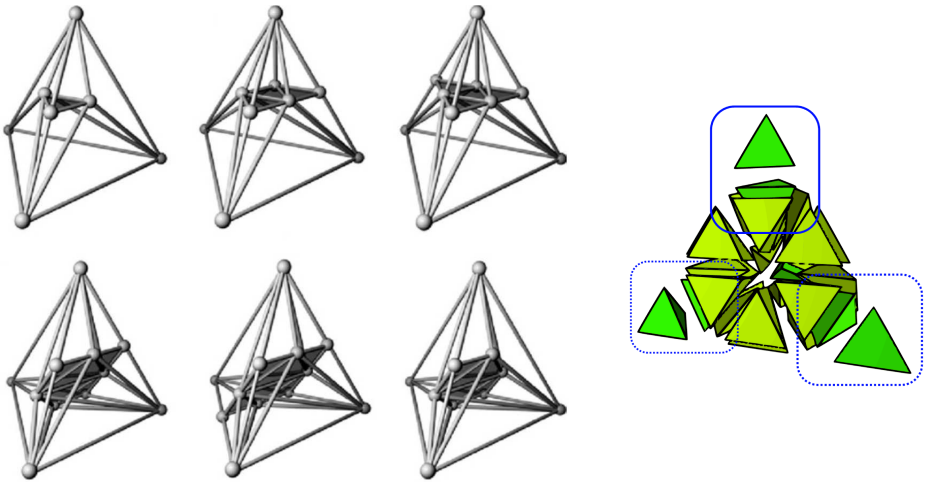


Figure 5.9: Reference re-meshing algorithms: *Left:* 1:17 subdivision (figure extracted from [13]); *Right:* one step of the re-meshing algorithm proposed by [50] (figure extracted from [50])

We choose to compare our approach with the approach followed by Bielser and al. [13, 14] based on the 1:17 subdivision of the tetrahedra traversed by the separation surface (see figure 5.9 (left)). Introduced in the framework of mass-spring systems, this approach has the advantage of being compatible with the FEM based simulations as the built meshes are conformal. It supports the same range of cutting scenarios than our approach.

Moreover we compare our approach to the recent work of Koschier [50] which aims at enhancing the granularity of a mesh to generate detailed fractures without a limitation of the inserted nodes. The beginning of one refinement step of this method (1-4 splits followed by 2-3 flips) is similar to our approach and has its origin in [18]. However, the last steps differ: The work of Koschier inserts two nodes on each cut edge, where our approach avoids a node insertion and limits the number of nodes. The completed subdivision of one element is displayed in figure 5.9 right. Since this is an important property to maintain real-time performance, it will be considered in the following.

Hereafter, we give a theoretical analysis of the number of nodes introduced

by Bielser's, Koschier's and our re-meshing methods. Then, we present experimental results showing the evolution of the numerical quality of the meshes during a simulation considering a cut on a deformable beam.

5.2.1 Theoretical analysis

The re-meshing impacts the FEM simulation in two ways: first, a local stiffness matrix must be computed for each modified element, and second, solving the resulting linear system depends (at best) linearly on the number of nodes in the new mesh. In the following we evaluate these quantities for our re-meshing method and for the method proposed by Bielser and Koschier.

Let us consider a tetrahedral mesh and a cut surface that traverses $n(v)$ adjacent tetrahedra. For that, let $n(k)$ be the number of added nodes and $n(v')$ the number of tetrahedra that replace the initial $n(v)$ tetrahedra during the re-meshing process. To simplify the calculations, we assume that an average of 5 tetrahedra are adjacent to each edge and that the cutting plane intersects all tetrahedra at three (case 1) or four edges (case 2). The reality lies between these two cases.

Let us evaluate $n(e)$ the number of cut edges and $n(f)$ the number of cut faces. In the case 1, 3 edges are cut for each tetrahedron. It follows that $n(e) = \frac{3}{5} \times n(v)$ and $n(f) = \frac{3}{2} \times n(v)$. In the case 2, 4 edges are cut for each tetrahedron resulting in $n(e) = \frac{4}{5} \times n(v)$ and $n(f) = \frac{4}{2} \times n(v)$.

In the approach followed by Bielser, a node is inserted on the middle of each cut edge and face. Thus, in the case 1, $n(k) = n(e) + n(f) = \frac{21}{10} \times n(v)$ and in the case 2, $n(k) = n(e)+n(f) = \frac{28}{10} \times n(v)$. In one refinement step of the method presented by Koschier, one vertex is inserted for each cut tetrahedron and two vertices on each cut edge. We obtain $n(k) = n(v) + 2n(e) = \frac{11}{5} \times n(v)$ for case 1 and $n(k) = n(v)+2n(e) = \frac{13}{5} \times n(v)$ for case 2. With our method, in either case, only one vertex by tetrahedron is inserted, during the $1-4$ split, and $n(k) = n(v)$.

Note, that we presented the number of nodes, when the mesh is remeshed to incorporate the cut surface into the mesh. In order to separate the mesh at the cutting surface, all the nodes on the surface need to be duplicated. As all the nodes of Bielsers and our approach are inserted on the cut surface, the number of added nodes needs to be multiplied by two. For the method presented by Koschier, only one of the two nodes inserted on an edge is duplicated, thus we obtain for case 1 that $n(k) = 2n(v) + 3n(e) = \frac{19}{5} \times n(v)$ and for case 2 we have

$n(k) = 2n(v) + 3n(e) = \frac{22}{5} \times n(v)$.

The number of tetrahedra introduced by the $1:17$ subdivision is, $n(v') = 10 \times n(v)$, in the case 1, and $n(v') = 16 \times n(v)$, in the case 2. On average $n(v') \approx 13 \times n(v)$. In the Koschier method the number of tetrahedra is $n(v') \approx 14 \times n(v)$ (see figure 5.9).

With our method, in the case 1, the $n(v)$ cut tetrahedra become $4 \times n(v)$ tetrahedra after the 1-4 split, $n(v)+3\times n(v)\times \frac{3}{2}$ after the 2-3 flip, and $n(v)+3\times n(v) \times \frac{3}{2} \times \frac{2\times 5-4}{5}$ after the last general flip. Thus, $n(v') = 6.4 \times n(v)$ after these operations. In the case 2, the $n(v)$ cut tetrahedra become $4 \times n(v)$ tetrahedra after the 1-4 split, $4 \times n(v) \times \frac{3}{2}$ after the flip 2-3, and $4 \times n(v) \times \frac{3}{2} \times \frac{2\times 5-4}{5}$ after the last step. Thus, $n(v') = 7.2 \times n(v)$ after these operations.

Our method is far more efficient as it introduces ≈ 2 times less tetrahedra and ≈ 2 times less nodes than Bielser or Koschier. This makes it a better match for real-time simulations what will be demonstrated with the experimental results provided hereafter.

5.2.2 Experimental results

This section shows the results we obtained in experiments performed to evaluate and compare the proposed method with the approach proposed by Bielser. In this scenario a deformable beam is progressively cut. The beam we consider initially has 371 tetrahedral elements and 131 nodes. The cut advances step by step, separating the beam in two pieces (see figure 5.10). We chose a cut plane close to the initial nodes in order to challenge the methods that we compare.

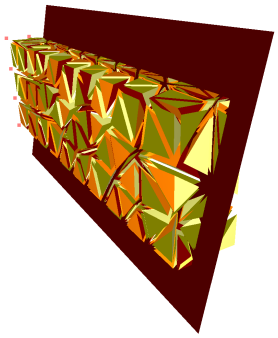


Figure 5.10: Cutting a deformable beam

To make a sound statement about our method, we combined the approach

of Bielser with snapping. The snapping of nodes is performed, based on the threshold ϵ, introduced and described in detail in section 5.1.3. We evaluated the two re-meshing methods with different values for ϵ. The results are plotted in figure 5.11. In these figures, the x-axis is the simulation step and is directly related to the number of tetrahedra that have been cut by the moving plane.

Figure 5.11 left shows the evolution of the total number of nodes in the mesh. As expected, our method introduces less vertices than Bielser, even when the snapping is enforced.

Figure 5.11 right shows the evolution of the condition number of the systems. The condition number is a measure to evaluate the numerical quality of the systems. A higher condition number results in more iterations to solve a numerical system, negatively impacting the performance of the simulation. The plot shows that the decreased quality of the elements introduced along the separation surface is intimately involved in the evolution of the condition number.

With the Bielser method, the condition number increases and with that the numerical quality deteriorates quickly. An increase of the snapping threshold ϵ delays that deterioration, but the final performance remains poor. The benefits of our method are clearly visible here – the numerical quality of the mesh is much better controlled. The snapping clearly improves the performance, as it introduces less nodes and the condition number remains lower. But even if the number of nodes has been doubled, the simulation still run with real-time performance. The example demonstrates that our approach allows the system to stay well-conditioned even in case of large cuts.

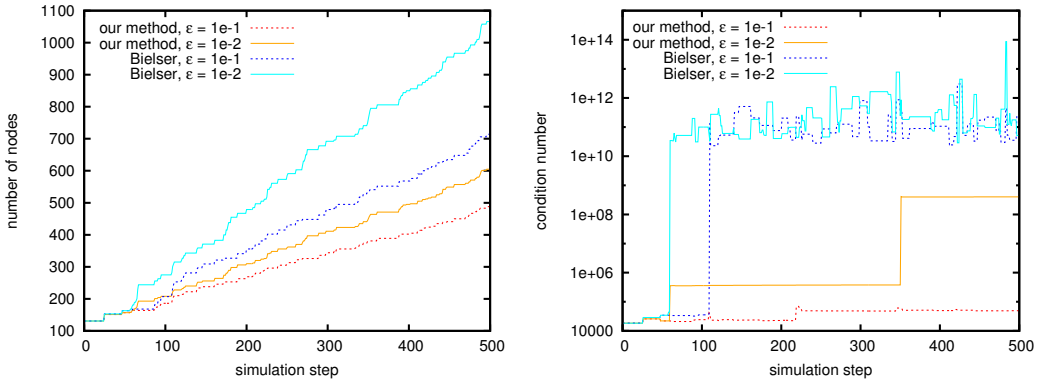


Figure 5.11: Evolution of the the number of nodes *(left)* and condition number *(right)*

Figure 5.12, displays the built separation surface using both re-meshing methods. Please keep in mind, that the smallest element size leads to a small minimal eigenvalue $\lambda_{\min}$, while the biggest eigenvalue $\lambda_{\max}$ is in most cases

related to illshaped elements. In figure 5.12 potential candidates for both can be identified and are surrounded by a red circle for the minimal and red dashed ellipses for the maximal eigenvalue. Thus this figure helps to identify potential source of the difficulties of the Bielser method. Our method displays globally a more homogeneous behaviour.

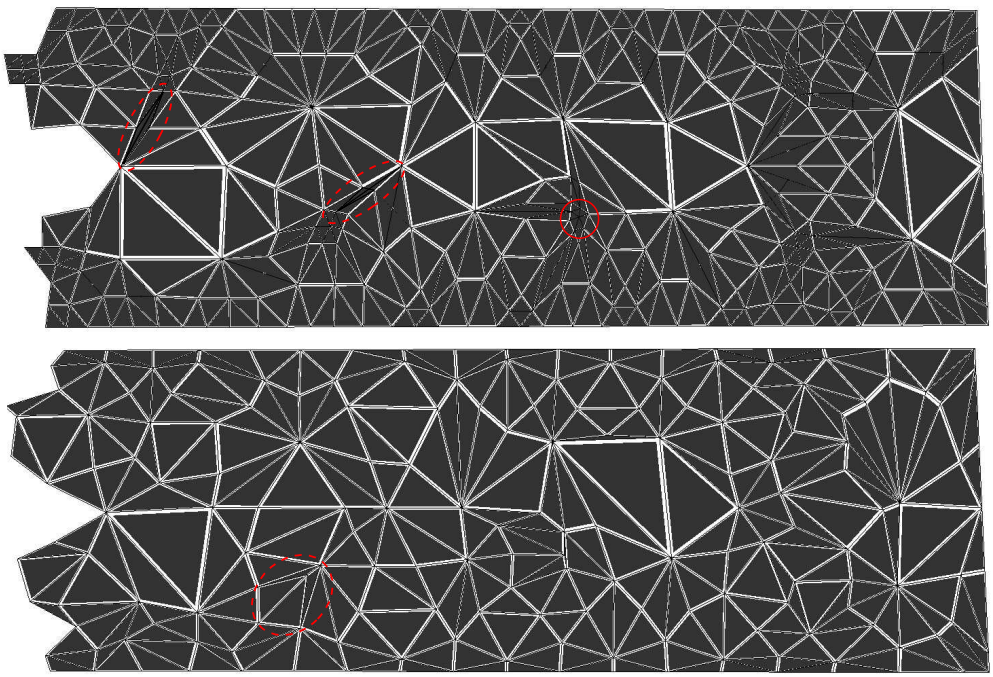


Figure 5.12: The separation surface: *(top)* with Bielsers approach; *(bottom)* with our re-meshing method; small triangles and thus potentially small tetrahedra are surrounded by a *(red circle)*, sliver triangles and thus potetially illshaped tetrahedra are surrounded by *(red dashed ellipses)*

5.3 Discussion and conclusion

In this section, we give details on the positioning of middle points for complete and partial cuts and we discuss important features of the topological operations that we use. Then we close with a summary of the work presented in the chapter.

The position of inserted points on the separation surface is crucial for our method: on the one hand for the correct approximation of the cut surface and on the other hand to obtain well-shaped elements.

In order to approximate the cut, usual re-meshing algorithms sample a

cutting surface using the intersections with the edges (and in three dimensions potentially with faces) of the mesh, i.e. the boundary of the elements. This is displayed in figure 5.9 for the 3D case and figure 5.13 first row for a 2D case. For our re-meshing approach, points are not inserted on the boundary of the elements, but inside the volume. Thus, a cut is sampled by points in the volume, resulting in a higher freedom of the choice for their positions. Currently these positions are calculated based on geometric constraints, that are detailed and discussed in the following.

For completely cut elements our algorithm inserts the middle point at the barycenter of the intersections between the cut surface S and the edges. That means, the cut is sampled in the volume of the element using positions of the cut on the element boundary. Figure 5.13 displays this scenario in the second row and reveals, that this choice potentially leads to an inaccurate approximation of the cut. It is more natural, to set the middle point positions on the

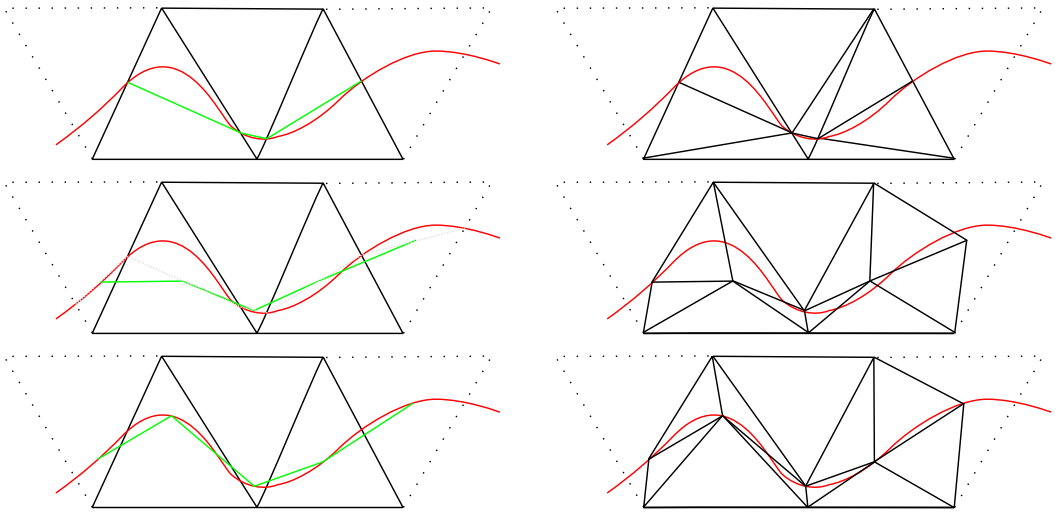


Figure 5.13: Approximation of a cut *(red curve)* inside a mesh *(black lines)* using different sampling and re-meshing ideas, before the re-meshing *(left)* and after the re-meshing *(right)*: *Top row:* sampling and re-meshing on the edges; *Middle row:* sample nodes chosen on the edges but placed inside the element *Bottom row:* sampling and re-meshing inside the element

cut inside of the element. This idea results in the two dimensional example to a better approximation, see figure 5.13 third row. However, this improvement only shows relevance for cuts that have strong changes in direction inside an element and thus will be adressed in the future.

The position of the middle point inserted in the split 1 to 4 operation in partially cut elements is likewise not restricted to intersections of the tetrahedral element boundary and the cut front, but can be displaced along the cutting

front inside of the element. As the re-meshing of partially cut elements is prone to yield illshaped elements (see figure 5.14 (second column)), we choose to place the point at the intersection with the edges between a point in the barycenter and the corners of the element. While this improves the condition number, it has a negative impact on the approximation of the cut surface, as the partial cut is performed either too far or not far enough, see figure 5.14 (middle right). Thus, in the three dimensional case, the cut front potentially has a zick-zack shape, as it can be seen on the left part of figure 5.12. In

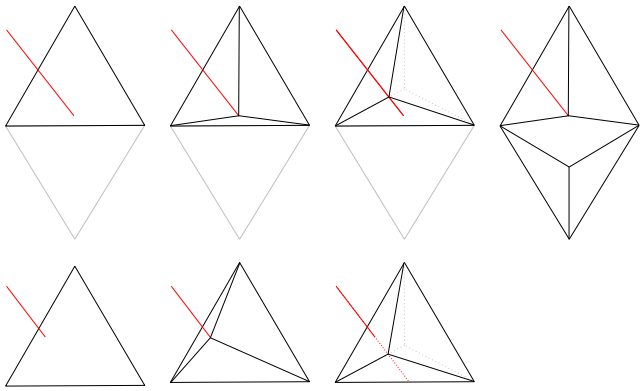


Figure 5.14: Partial cut *(red line)* on triangle, before topological change *(left)* and after topological change using different ideas

order to improve this behaviour while preventing flat elements, a split 1 to 4 of neighboring elements followed by a flip 2 to 3 is potentially interesting. In the two dimensional case the equivalent is represented by a split 1 to 3 followed by a flip 2 to 2, see figure 5.14 (top right). Since this suggestion can not be applied for boundary elements, partial cuts either should be performed as soon as they have a certain depth or progressed to that depth, to prevent negative impact on the stability of the simulation. As this idea has not yet been implemented, it will be addressed in our future work.

Beyond the importance for a correct approximation of the cut surface and the prevention of illshaped elements, the correct positioning of the middle points is critical for the topological operations that follow a 1 to 4 split. In rare cases, inverted elements occur as a result of the 2 to 3 flip and respectively for the 2 to 2 flip in the two dimensional case, see figure 5.15. It can be seen, that the triangle spans over the boundary of the original triangles. The figure reveals, that the inversion is due to the nonconvexity of neighboring elements (triangles in 2D, tetrahedra in 3D). Thus an initial check of the mesh and potential adaptation of the point positions could prevent the scenario.

The intersection of several cuts is one of the main challenges for cutting algorithms. The virtual node algorithm allows for an arbitrary number of po-

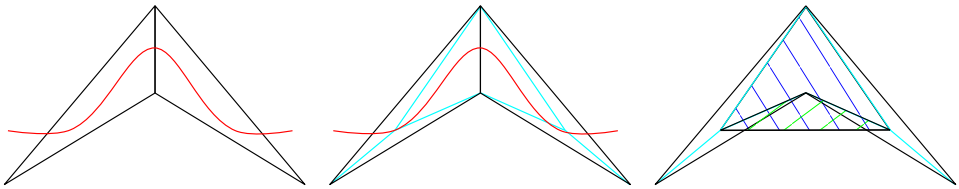


Figure 5.15: Rare case of cutting *(red curve)* a noncomplex set of triangles *(black lines)*: *Left:* before topological change; *Middle:* after split 1 to 3 *(new edges in turquoise)*; *Right:* after flip 2 to 2, with the new triangles *(hatched in green and blue)*

tentially intersecting cuts inside one element (see figure 5.16 (top)), but the behaviour after the cuts might not be physically correct (further details can be found in section 4.1.2). In contrary to this approach, our algorithm is restricted to the initial topology of the mesh. One tetrahedron can maximally be separated into 24 subtetrahedra and one triangle can maximally be separated into six subtriangles (see figure 5.16 (bottom)). This is an acceptable behaviour, when compared to the best cutting algorithms in the literature: For example the work of Wu et al [126, 129] allows in the two dimensional case for a separation into eight subelements (see figure 5.16 (middle row)). However, using a cutting algorithm, similar to their work, with links between two elements to triangles, would yield to a maximal separation into six subelements. Similar ideas apply to the three dimensional case. Note, that the figures display

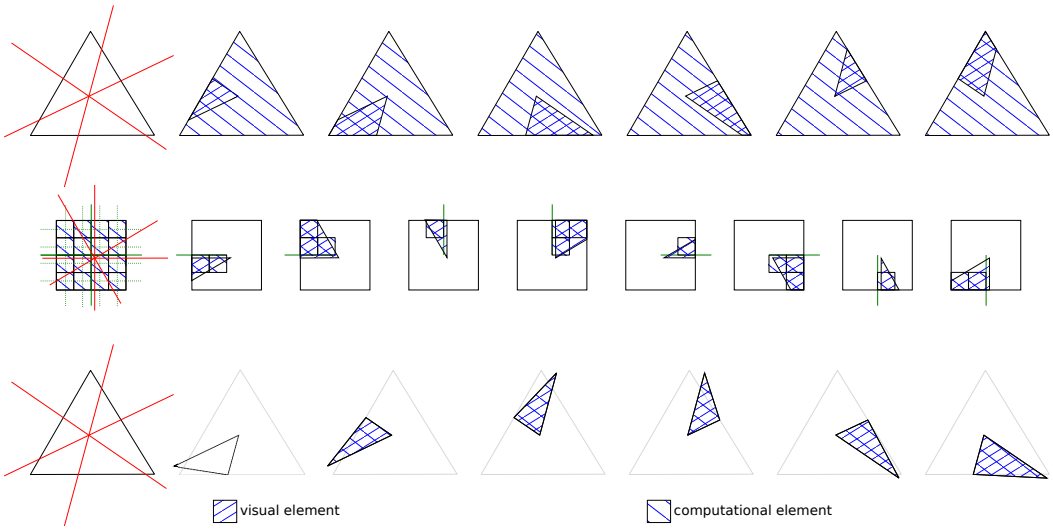


Figure 5.16: Comparison of different cutting algorithms when crossing cuts *(red lines)* occur; *First row:* virtual node algorithm; *Second row:* Composite finite element method, links in *green*; *Thrid row:* our re-meshing approach combined with the snapping of the nodes ToDraw: improve the visualization of visual and computational elements (later)

best case scenarios: Our algorithm and the proposition of Wu are particularly challenged with cuts, that have small (dihedral) angles between them.

In the example above, we tested the abilities of our re-meshing algorithm. In our work, we combine it with the snapping of nodes and thus allows for separations along element boundaries. Therefore, when considering several elements, our approach can handle a higher variety of crossed cuts (see figure 5.17). However, this variety is dependent on the position of the intersection of two cuts. For the two dimensional example in figure 5.17 (first row), the intersection is located close to a node of the mesh and an intersection of twelve cuts can be handled – for triangles with a 60°, the best case scenario. When the intersection of the cut is dislocated, i.e. not close to a node of the mesh, our algorithm has to move a node of the mesh to the cut intersection, to allow for a similar scenario. Resulting meshes are prone to contain small or illshaped elements and do not represent all cuts, as displayed in figure 5.17 (second row). Moreover, in the three dimensional case, the intersection of the cut can be a line or a point, yielding to more challenges. Thus, to prevent small or illshaped elements, current implementations only consider a case where a node is close to the intersection of the cuts. Results for the three dimensional case are displayed in figure 5.17 (bottom).

As it has been mentioned in section 5.1.4, our algorithm has very interesting properties for an iterative or multiscale refinement of a mesh in order to incorporate the separation surface into the volumetric grid. Therefore, it could be very interesting to evaluate these properties in a wider scale and to allow for real-time performance in the context with the iterative or multiscale approach. For crossing cuts, even only one refinement to place a node on the intersection of the cuts already could yield very promising results.

In this chapter we addressed and discussed in detail the challenge of efficient cutting simulation in the context of real-time soft tissue modeling. Our approach relies on a new re-meshing algorithm, that introduces well-shaped elements that produce well-conditioned system matrices. During the cutting process, the number of added degrees of freedom is well controlled. For cuts close to nodes, we improve the performance and stability of our re-meshing method with the snapping of vertices. The combination of the ideas has been implemented on top of a real-time finite element method. The implementation keeps performances of the simulation over the time. The re-meshing method has the ability to create an adaptive approximation of the cutting or tearing surface, which gives the ability to tune between the flexibility and speed of the re-meshing.

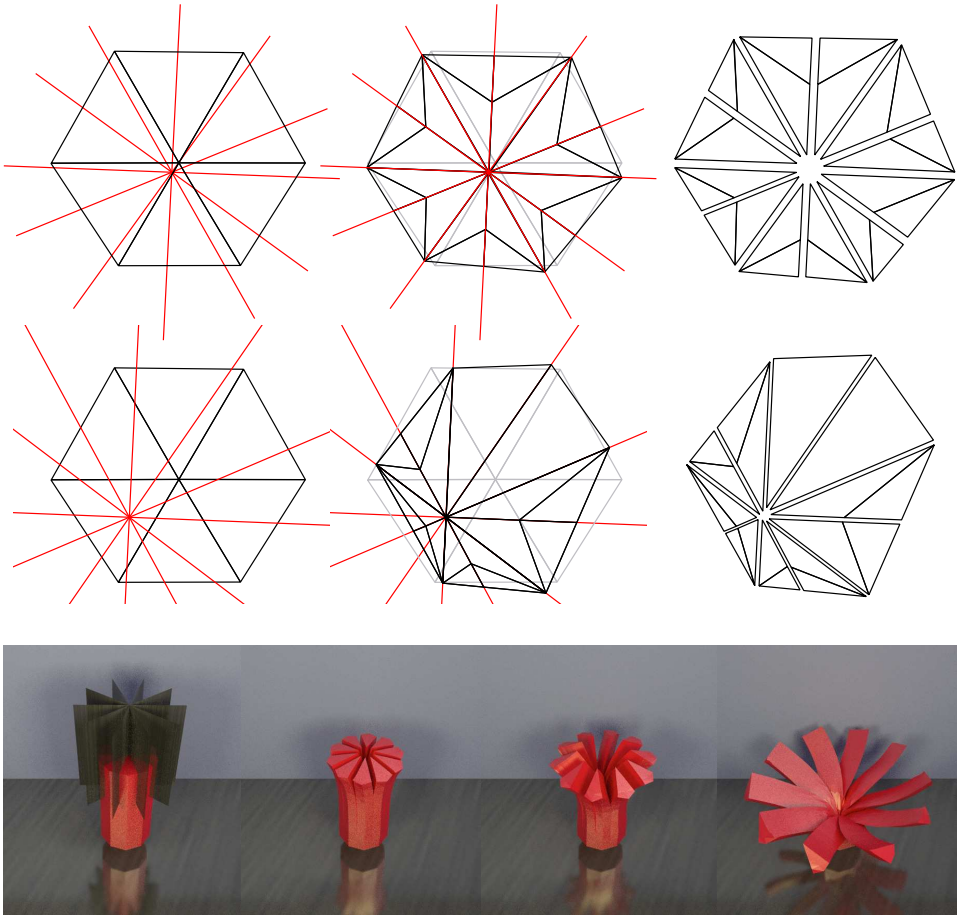


Figure 5.17: Crossing cuts *(red lines)* over several elements *(black lines)* using our re-meshing approach combined with the snapping of nodes; intersection of cuts close *(top row)* and far *(middle row)* to mesh node; results in three dimensional case *(bottom row)*

Chapter 6

Topological changes on quadratic elements

Contents

A geometrical representation that uses linear tetrahedra is sufficient for applications in engineering, as the objects often have sharp corners. But for the curved surfaces of organs or other structures like the membrane in the eye, many linear tetrahedra need to be used to represent the smooth surface. In this particular context, cutting and cauterizing movement trajectories are smooth and thus difficult to represent by the straight lines and planar surfaces of the linear tetrahedra.

As described in section 3.5, linear tetrahedra suffer severe locking for incompressible materials and thus an unnaturally stiff behaviour is observed.

Suwelack et al [118] propose to use quadratic tetrahedra as a robust and accurate model for real-time soft tissue simulations. The paper gives a detailed numerical analysis on the improved accuracy of quadratic shape functions – as already pointed out in section 3.5 – and even shows an improved efficiency for the same accuracy.

In this chapter, we expand the re-meshing algorithm presented in chapter 5 to quadratic tetrahedral elements, using the work of Suwelack et al [118]. To our knowledge, we present the first algorithm re-meshing quadratic tetrahedral elements in real-time. We close with our ongoing work about the fracture simulation of quadratic shells.

6.1 Cutting quadratic tetrahedra

Based on the encouraging results for our novel re-meshing algorithm for linear tetrahedra presented in section 5.2, this chapter applies the re-meshing approach using quadratic shape functions. Subsection 6.1.1 describes how the combinatorial maps are used to represent tetrahedra with a higher polynomial degree. It details how meshes are generated and how the smooth surface of an element is visualized in the simulation. Then, we explain the necessary adaptions of the re-meshing method to apply them for quadratic elements in subsection 6.1.2, i.e. the way nodes on the edges have to be introduced and moved. Results are presented in subsection 6.1.3 as a theoretical analysis and in subsection 6.1.4 on a few examples. The section closes with a brief discussion in subsection 6.1.5.

6.1.1 Topological representation and visualization

Reusing the topological operations provided for the re-meshing algorithm presented in section 5.1 is particularly interesting to save implementation time. Remember, that this work is based on the combinatorial maps topology described in subsection 4.3 and topological operations are related to linear tetrahedral elements. In order to account for the additional node on each edge of a quadratic element, we use attributes on the edges, that save the ids of the nodes on the edges, see figure 6.1 (left) for a two dimensional example. With the node ids, the quadratic triangle can be expressed in its continuous form using the interpolation equations (1.1), (3.16), (3.17), see figure 6.1 (middle). For visualization, tesselations are used, that discretize the boundary of the elements by a finite number of straight elements, see figure 6.1 (right). Same ideas apply for quadratic tetrahedra and can be extended to shape function of any polynomial degree using additional node ids on faces and volumes. But this work only investigates into the analysis of the quadratic case. Based on the description above, the topological operations of section 5.1 can be applied for quadratic tetrahedra as well. The correct insertion and duplication of new points is discussed in section 6.1.2.

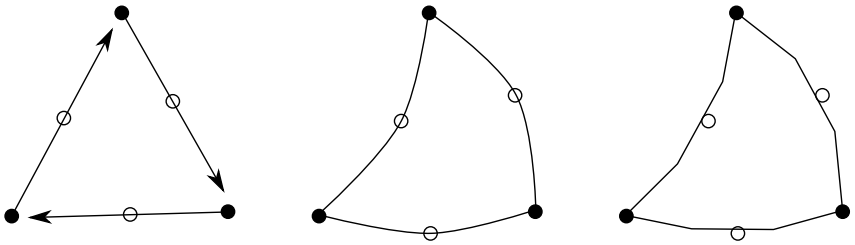


Figure 6.1: Quadratic triangle representations: *Left:* topological, with vertex ids saved as edge attributes; *Middle:* continuous representation; *Right:* visualization using tesselation

Compared to the generation in the linear case, quadratic tetrahedral meshes are difficult to obtain: for a given smooth surface, standard mesh generators often fail generating a coarse quadratic tetrahedral mesh. Mezger et al [60] propose to first produce a coarse triangular mesh from the smooth surface, which serves as an input to generate coarse linear tetrahedra. Then edge middle points are inserted at the barycenter between the tetrahedral nodes and adjusted to the surface, see figure 6.2. In our work, we follow this idea

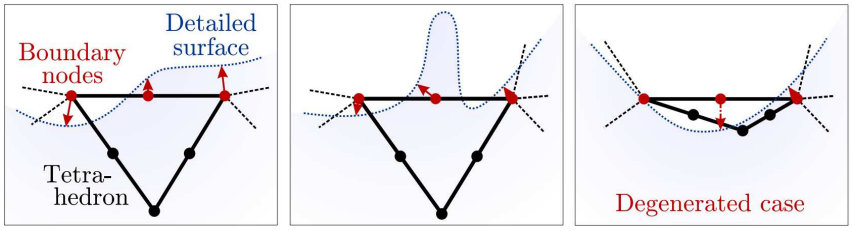


Figure 6.2: Movement of nodes to the smooth surface (figure extracted from [60])

with a modified last step: when moving the middle points on the surface, we use a simulation based on linear elasticity to prevent sliver elements.

6.1.2 Extension of re-meshing idea from linear tetrahedra

In this subsection, we extend the re-meshing algorithm described in chapter 5. At this point the topological operations presented in the previous chapter are not repeated in their entity, since we focus on particular aspects. Therefore it is recommended to first read section 5.1 before continuing this subsection.

As mentioned in the previous subsection, our specific topological representation allows for a reuse of the topological operations. Since quadratic elements have additional degrees of freedoms on the edges, particular attention has to be paid for the correct movement of old and the correct placement of newly

inserted nodes on the edges. The nodes on the edges result in an increased computational cost for every element and further details on this subject can be found in subsection 6.1.3. In the following we address all the steps of the re-meshing algorithm, we adapt it to the quadratic case and highlight the differences.

Similarly to the linear case, the location of the virtual cut is sampled using the edges of the mesh. For example for a cutting surface S, the intersections with the curved edges are calculated and stored alongside with the normal of the separation surface at this point.

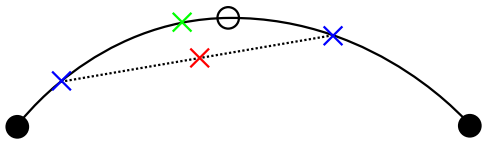


Figure 6.3: Two points *(blue crosses)* on curve *(black)* with straight or direct barycenter *(red cross)* and interpolated edge barycenter *(green cross)*

When placing nodes in quadratic tetrahedra, two cases can be distinguished (see figure 6.3): on a, what we call *straight* or *direct barycenter*, i.e. at the average of all the points. Or on what we call in the following an *interpolated element barycenter*: averaging all the barycentric coordinates of the points and then inserting this averaged barycentric coordinate into the interpolation inside the element (an edge, a triangle or a tetrahedron). For newly inserted edges inside of the object, we assume that edges can be straight without leaving the tetrahedron. That means that the edge nodes can lie on the middle of the two edge points. Therefore, if not specified otherwise, points are inserted on the straight barycenter of the two edge nodes. Thus the split 1 to 4, flip 2 to 3 and the general flip for edges inside the object are performed like in the linear case with the insertion of edge points at the straight barycenter. This choice may yield to inverted elements and an adapted boundary, as it is discussed in subsection 6.1.5.

For quadratic shape functions, the position of the middle point is as essential as the edge points themselves. When displacing e.g. the middle point along the curve of the edge, the continuous representation changes greatly, see figure 6.4 (left). Thus when cutting a boundary edge attention has to be paid to set the points in the interpolated edge barycenter between the cut point and the edge points 6.4 (right).

The middle points on the edges of a quadratic triangles and tetrahedra have the same importance and change the elements shape if not correctly placed. Therefore cut boundary triangles need a special treatment for the split 1 to

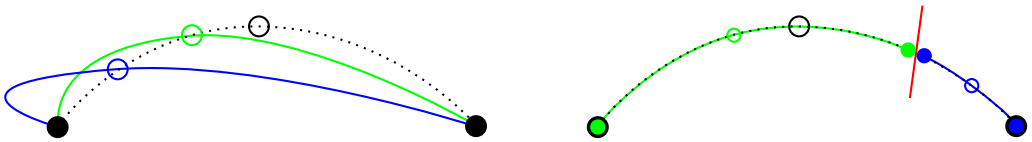


Figure 6.4: *Left:* Effect of displaced middle point: original edge *(dashed curve)* and displaced middle points with resulting curves *(other colors)*; *Right:* Cut *(red line)* of edge *(dotted curve)* and placement of new edge nodes *(green and blue dots)* with new edges *(green and blue curve)*

3 operation: The split 1 to 3 point $\mathbf{p}_{13}$ is inserted at the intersection of the cut with the curve connecting the interpolated triangle barycenter with the nodes 6.4 (left). Then edge points are inserted as the interpolated triangle barycenter between $\mathbf{p}_{13}$ and the triangle corner points, see figure 6.5 (middle right). For a split 1 to 3 of a tetrahedral element, an additional edge point has

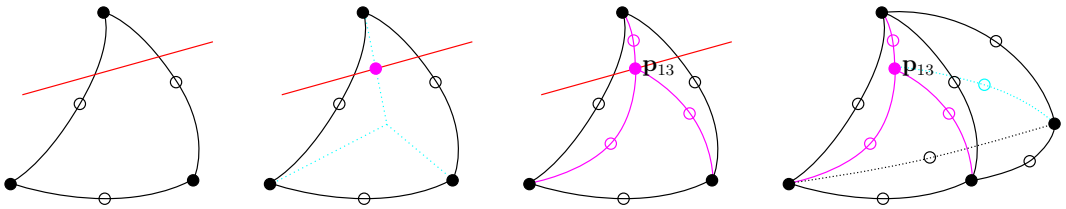


Figure 6.5: Split 1 to 3 *(black dots and curves, dashed when not visible directly)*: *Left:* Cut scenario; *Middle left:* Insertion of the split 1 to 3 point; *Middle right:* Split 1 to 3 on surface, new edges and points in *pink*; *Right:* Split 1 to 3 on tetrahedron with additional internal edge *(turquoise)*

to be inserted as the interpolated tetrahedron barycenter between $\mathbf{p}_{13}$ and the tetrahedron node that does not lie on the boundary of the object.

The split 1 to 3 is followed by a bisection of the boundary edges intersecting with the cut. Similarly to the description above on the bisection of a quadratic edge, an intersection point $\mathbf{p}_{S\cap\Omega_e(t)}$ is inserted on the edge. Then the tetrahedron is split into two by inserting a node at the interpolation edge barycenter between $\mathbf{p}_{S\cap\Omega_e(t)}$ and the nodes of the cut edge. New edges are inserted between the other nodes of the tetrahedron and the intersection point $\mathbf{p}_{S\cap\Omega_e(t)}$. These new edges use the intersection between the curve between the two existing edge middle points on the surfaces, that are adjacent to the cut edge, see figure 6.6.

If the cut boundary edge does not represent an important feature of the mesh, then a general flip is of advantage, since it does not insert nodes on the cut edge. Again, this operation needs minor adjustments to account for the quadratic case: as most newly inserted edges lie inside of the object, we insert

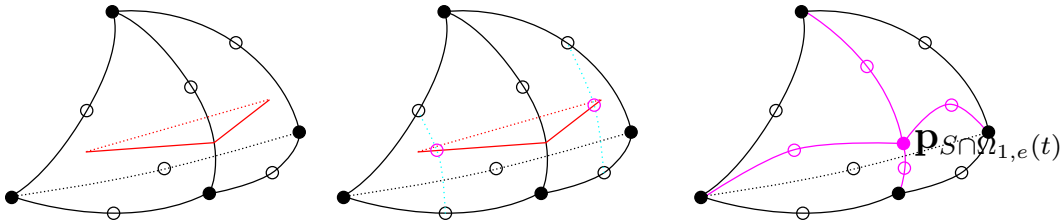


Figure 6.6: Bisect edge on quadratic tetrahedron *(black curves (dotted in the back) and dots)*: *Left:* cut scenario; *Middle:* determination of edge middle points; *Right:* After bisection of edge, new edges and points in *pink*

the edge points on the straight barycenter of the two edge points.

Note, that the topological changes result in standard quadratic tetrahedra, i.e. the visualization mentioned in the previous chapter can be used without further updates.

6.1.3 Theoretical analysis

Similarly to the linear case, this chapter contains a theoretical analysis of the number of nodes added in order to allow for a separation along a cut surface. To our knowledge, there exist no re-meshing techniques, that refine a quadratic tetrahedral element into quadratic tetrahedral elements. Thus we reuse the algorithms of Bielser et al [13, 14] and Koschier et al [50] and compare their extension to quadratic elements with our re-meshing technique. We denote these extended algorithms in the following with a superscript ^{2}, i.e. as Bielser^2, [13, 14]^2, Koschier^2, [50]^2, our and [81, 80]^2. The number of created tetrahedral elements is independent of the polynomial degree and we refer to the derivation for linear elements in section 5.2.1. The following theoretical analysis helps to determine the impact of the re-meshing algorithm on the FEM simulation, as solving the resulting linear system depends (at best) linearly on the number of nodes in the new mesh. Moreover, we compare to the linear case and the reader can decide whether the linear or quadratic re-meshing algorithm is better suited for a particular applications.

We consider a tetrahedral mesh and a cut surface that traverses $n(v)$ adjacent tetrahedra. For that, we denote the number of added nodes on the cutting surface as $n(k_S)$ and the added nodes that are not on the cutting surface as $n(k)$. This distinction is necessary to allow for a derivation of the number of nodes, when the two parts above and below the cut are separated. To simplify the calculations, we assume that an average of 5 tetrahedra are adjacent to

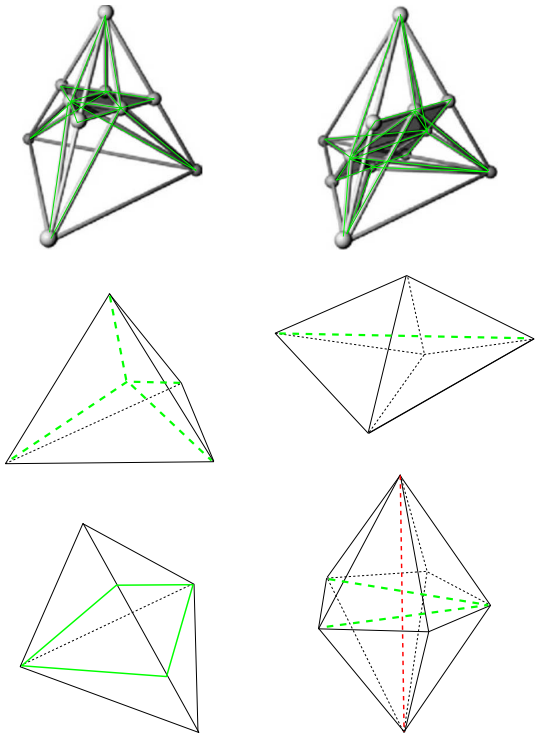


Figure 6.7: Topological operations used in the theoretical analysis with newly inserted edges in *green* and deleted edges in *red*: *Top:* Bielser three cut edges and four cut edges (figures taken from [13, 14]); *Middle:* Split 1 to 4 and flip 2 to 3; *Bottom:* 2 node insertion (applied in [50]) and general flip

each edge and that the cutting plane intersects all tetrahedra at three (case 1) or four edges (case 2). The reality lies between these two cases.

Let us evaluate $n(e)$ the number of cut edges and $n(f)$ the number of cut faces. In the case 1, 3 edges are cut for each tetrahedron. It follows that $n(e) = \frac{3}{5} \times n(v)$ and $n(f) = \frac{3}{2} \times n(v)$. In the case 2, 4 edges are cut for each tetrahedron resulting in $n(e) = \frac{4}{5} \times n(v)$ and $n(f) = \frac{4}{2} \times n(v)$.

In contrast to linear tetrahedra, an evaluation in text form is difficult to understand and we use figures and tables. Quadratic tetrahedra use additional nodes on every edge. Figure 6.7 displays the operations for the different methods and highlights the inserted and deleted edges. Based on the observations in the figures, the number of newly inserted nodes can be calculated, see table 6.1. Compared to the linear tetrahedra, the theoretical analysis on quadratic tetrahedra shows, that there are at least 5 times more nodes inserted, see table 6.2. For the same accuracy, the number of elements is greatly reduced when using quadratic instead of linear shape functions. Similarly, the number of cut edges is greatly reduced. If we consider e.g. the subdivision

Method	case	$n(k_S)$	$n(k)$	$n(k_S)+n(k)$	$2n(k_S)+n(k)$
Bielser^2	1	$n(e)+n(f)+[2n(f)+4n(v)]$	$[n(e)+3n(f)]$	$14.2n(v)$	$23.3n(v)$
[13, 14]^2	2	$n(e)+n(f)+[2n(f)+6n(v)]$	$[n(e)+3n(f)]$	$19.2n(v)$	$32.4n(v)$
Koschier^2	1	$n(v)+n(f)+n(e)+2n(f)$	$[4n(v)]+[3n(e)+2n(f)]$	$14.9n(v)$	$21n(v)$
[50]^2	2	$\underbrace{n(v)}_{\text{split14}}+\underbrace{n(f)}_{\text{flip23}}+\underbrace{n(e)+2n(f)}_{\text{2 node insertion}}$	$\underbrace{[4n(v)]}_{\text{split14}}+\underbrace{[3n(e)+2n(f)]}_{\text{2 node insertion}}$	$18.2n(v)$	$26n(v)$
Our	1	$n(v)+n(f)+n(e)$	$[4n(v)]$	$7.1n(v)$	$10.1n(v)$
[81, 80]^2	2	$\underbrace{n(v)}_{\text{split14}}+\underbrace{n(f)}_{\text{flip23}}+\underbrace{n(e)}_{\text{general flip}}$	$\underbrace{[4n(v)]}_{\text{split14}}$	$7.8n(v)$	$11.6n(v)$

Table 6.1: Re-meshing quadratic tetrahedra with different methods: the number of added nodes for the re-meshing $n(k_S)+n(k)$ and for the re-meshing with separation $2n(k_S)+n(k)$; *nodes added on the edges, are surrounded by []*

of a quadratic tetrahedral element into eight linear tetrahedral elements, then there is potentially one additional edge cut per tetrahedra and two per face. That means, when cutting $n(e)$ quadratic edges we can expect $2n(f)+n(v)$ cut linear edges. Thus, for the first case we get $2*3/2n(v)+n(v)=4n(v)$ and for the second case $2*2n(v)+n(v)=5n(v)$ cut linear edges. This factor is below the factors of inserted nodes, but it has to be underlined that a quadratic tetrahedron, which has the same number of degrees of freedom like the eight linear tetrahedra might have a higher accuracy (see [19, 118] and figure 3.12). Therefore, the theoretic results presented in table 6.2 are comparable to the results in the linear case. Comparing our re-meshing approach

		Linear		Quadratic		Quadratic/Linear	
Method	case	$n(k_S)+n(k)$	$2n(k_S)+n(k)$	$n(k_S)+n(k)$	$2n(k_S)+n(k)$	$n(k_S)+n(k)$	$2n(k_S)+n(k)$
Bielser^2	1	$2.1n(v)$	$4.2n(v)$	$14.2n(v)$	$23.3n(v)$	6.8	5.5
[13, 14]^2	2	$2.8n(v)$	$5.6n(v)$	$19.2n(v)$	$32.4n(v)$	6.9	5.8
Koschier^2	1	$2.2n(v)$	$3.8n(v)$	$14.9n(v)$	$21n(v)$	6.8	5.5
[50]^2	2	$2.6n(v)$	$4.4n(v)$	$18.2n(v)$	$26n(v)$	7	5.9
Our	1	$n(v)$	$2n(v)$	$7.1n(v)$	$10.1n(v)$	7.1	5.1
[81, 80]^2	2	$n(v)$	$2n(v)$	$7.8n(v)$	$11.6n(v)$	7.8	5.8

Table 6.2: Re-meshing tetrahedra: comparison between linear and quadratic case

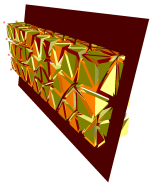
to other re-meshing methods for linear tetrahedra applied used with quadratic shape functions, we introduce more than two times less nodes and thus our method is the best suited method as well for quadratic tetrahedra.

Method	case	$n(k_S)+n(k)$	$2n(k_S)+n(k)$
Bielser^2	1	2	2.3
[13, 14]^2	2	2.5	2.8
Koschier^2	1	2.1	2.1
[50]^2	2	2.3	2.2

Table 6.3: Re-meshing quadratic tetrahedra: comparison of methods of literature applied to quadratic tetrahedra^2 with our method

6.1.4 Experimental results

In order to confirm the theoretic results of the last subsection, we conducted the same example as in the linear case, which is well suited due to the high number of cut elements (see table 6.4 (left)). For three different beam resolutions, table 6.4 (right) displays the number of tetrahedral elements $n(v)$ intersecting with the cut and the number of added nodes using our re-meshing algorithm to allow for a separation. In this example boundary elements are cut and the



Beam elements	$n(v)$	$2n(k_S) + n(k)$
371	124	$2050 - 739 = 1311 \approx 10.6n(v)$
712	215	$3033 - 1334 = 2199 \approx 10.2n(v)$
4621	982	$16786 - 7334 = 9452 \approx 9.6n(v)$

Table 6.4: *Left:* Example of the cut *(red plane)* inside of a beam with 371 elements; *Right:* Evaluation on other resolutions of beams

split 1 to 3 (and the bisect edge) add nodes for cut boundary elements, that are not considered in the theoretical analysis. Beams with a low resolutions have a higher ratio of cut boundary elements when compared to cut elements inside the volume. Thus, we expect more nodes added per cut element for cuts of low resolution beams. Table 6.4 (right) confirms this expectation, but also reveals, that the theoretical values are above our results in the experiment. We explain this behaviour by a lower number of tetrahedra adjacent to an edge than in the theoretical analysis.

A cutting example on a low resolution quadratic mesh of a liver is displayed in figure A.28. Our re-meshing algorithm shows visually pleasing results at real-time frame rates without parallelization or any other attempts to improve computation time. It can be observed, that the spatial discretization with quadratic elements is smooth inside the elements, but not on the element boundaries. Thus sharp feature of an object can be represented, but, on the other hand, the representation with quadratic elements does not preserve the smoothness of the objects, even without cuts.

Table 6.5 helps to evaluate the impact of the cut on the size of the model and with that on the potential computation times. Every split 1 to 4 inserts five new nodes: one on the cut and four on the edges, see table 6.1. Thus for the cut of the liver we have $n(v) = 45/9 = 5$ tetrahedral cut elements and we get for the number of added nodes $2n(k_S) + n(k) = (278 - 36)/5 = 48.4n(v)$. This is equivalent to a factor close to five, when comparing to theoretic results. Note, that the presented case could be considered as worst case scenario: due

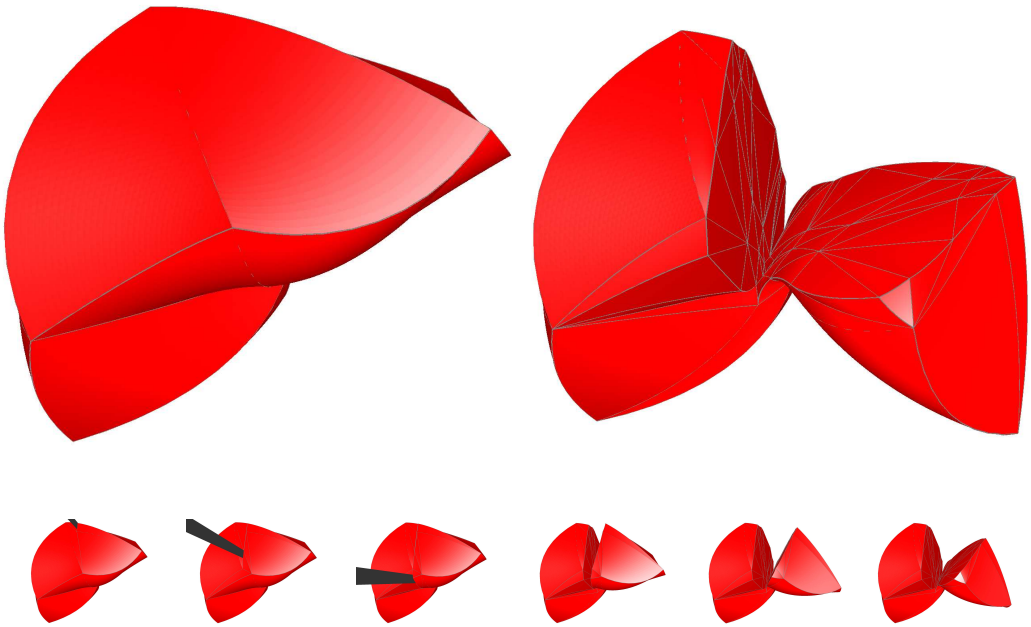
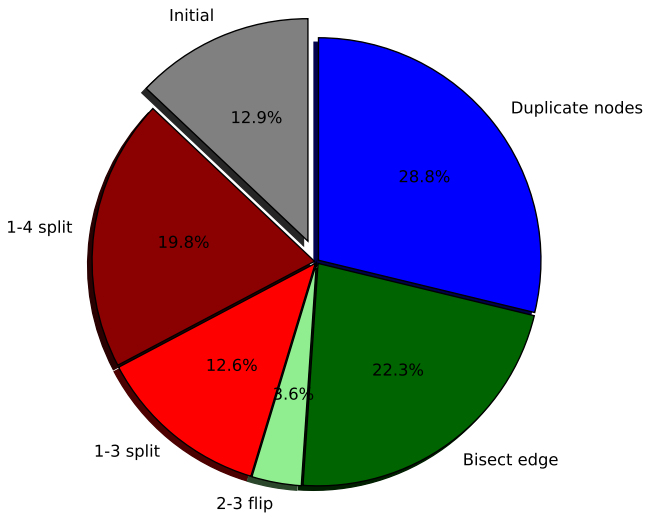


Figure 6.8: Cut of quadratic liver with a smooth surface *(red)* and curved edges *(grey)*: *Top:* initial and final configuration; *Bottom:* intermediate steps from left to right *(cutting tool in dark grey)*

to the low resolution most cut elements are boundary elements and since their surface is curved, the general flip can not get applied. Further important points



	Number of nodes
Initial mesh	36
Split 1 to 4	81
Split 1 to 3	126
Flip 2 to 3	136
Bisect edge	198
Duplicate nodes	278

Table 6.5: Cut of quadratic liver (see figure A.28), impact of the different topological operations on the number of nodes: *Left:* pie chart; *Right:* table with the added nodes after the subsequent operations

are considered and discussed in the next subsection.

6.1.5 Discussion

In this subsection we present and summarize advantages and disadvantages occuring when extending our re-meshing algorithm from linear to a quadratic tetrahedra.

In subsection 6.1.2, the split 1 to 4 inserts the tetrahedron (and the edge) middle point(s) on the direct barycenters of the four (respectively two) points of the tetrahedron (respectively edge). With this choice, nodes can be placed outside of the original element, risking to change the geometric and mechanical representation of the object. A better choice is the placement of the nodes on the interpolated barycenters, that prevents the problems mentioned above (see figure 6.10). However, as the examples presented in the previous chapters do not contain non-convex tetrahedral elements, we use the standard placement of the nodes.

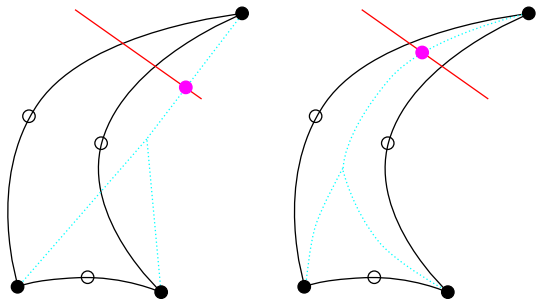


Figure 6.9: Different placements of the inserted point from the split 1 to 3 using the direct barycenters *(left)* and the interpolated barycenters *(right)*

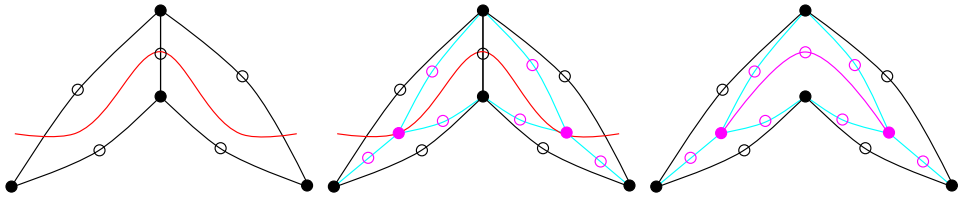


Figure 6.10: Rare cutting scenario of two non-convex neighboring triangles: *Left:* Initial configuration *Middle:* Split 1 to 3; *Right:* Flip 2 to 2

Due to the quadratic representation of the object, less elements can be used while the same accuracy of approximation is maintained. A negative side effect of the lower resolution is, that partially cut elements occur more often and may be represented in a worse fashion than for linear elements.

To conclude, this section presented the extension of the re-meshing algorithm from the linear to the quadratic case (see subsection 6.1.2). The theoretical analysis reveals that our method introduces less nodes, when compared to

other re-meshing algorithms proposed in the literature for linear elements and extended to quadratic elements (see subsection 6.1.3). Furthermore, results on a beam with different resolutions (see subsection 6.1.4) show that while the number of nodes per element increases by a factor of $10/4 = 2.5$, the number of added nodes increases by a factor around 5 (see table 6.2). This is reasonable, due to the higher accuracy per degree of freedom of and the lower number of cut edges for quadratic elements. Still, on an example on a liver represented by coarse quadratic tetrahedra (see subsection 6.1.4), an unexpected high number of nodes has been inserted. Combining the re-meshing algorithm with the snapping of the nodes could alleviate, but not completely remove this effect. Remember, that the snapping or movement of a node changes the interpolation of the element and thus its mechanical and geometrical representation. For quadratic elements, the effect of a snapped moved point on the mesh is more difficult to estimate, as it might get necessary to move the edge middle points as well.

Thus, in order to investigate into the snapping of nodes to the cut, while looking at other ideas of re-meshing algorithms, we consider in the following section the separation of simpler topological structures: triangular shells.

6.2 Fracture of quadratic shell elements

A cut at a microscopic scale is similar to a fracture around and due to sharp-edged devices. Beyond this, fracture or tearing are relevant in surgical interventions, such as the removal of the epiretinal membrane mentioned in the introduction. In this example, only very thin and surfacic structures are torn apart. Based on these arguments, in this section, we present our ongoing work on fracture phenomena using triangular shell elements.

Quadratic shape functions are particularly interesting in this context, as they allow representing the curved surfaces and they improve the stress computation, see figure 6.11 and [60] and further details below.

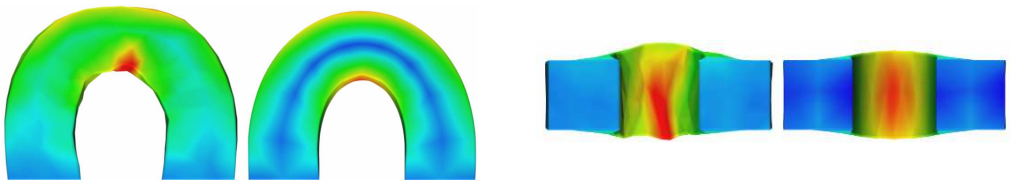


Figure 6.11: Von Mises stress distribution in tetrahedral elements with linear *(left subimages)* and quadratic *(right subimages)* shape functions (figure extracted from [60])

From the computational point of view, a fracture has different stages: First, a failure of a material has to be identified by evaluating the stress and comparing it to the strength of the mechanical model. As soon as a failure is detected, the topological change to react to the failure needs to be calculated and propagated to a mechanical model that simulates the deformations of the object after the failure.

With the Cauchy stress tensor, it is not directly possible to calculate the position and the direction of an emerging fracture. The common approach is to transform the stress tensor into another coordinate system by rotation with the angle

$$\theta_P = \arctan\left(\frac{2\sigma_{12}}{\sigma_{11} - \sigma_{22}}\right) \tag{6.1}$$

such that the shear stress in the new coordinate system is zero. Here we used σ_{ij} for the entries of the Cauchy stress tensor calculated for a triangular element. We define the principal stress

$$\sigma_{P,\pm} \coloneqq \frac{\sigma_{11} + \sigma_{22}}{2} \pm \sqrt{\left(\frac{\sigma_{11} - \sigma_{22}}{2}\right)^2 + \sigma_{12}^2} \tag{6.2}$$

If the maximal principal stress $\sigma_{P,+}$ exceeds the ultimate tensile strength σ_f of the object, i.e.

$$\sigma_{P,+} > \sigma_f \tag{6.3}$$

we assume a material failure in the direction of the rotation angle θ_P. Material failure is commonly classified in plasticity and breakage. For plasticity, the *von Mises* yield criterion based on the von Mises stress is applied. In our work we consider material failure only in form of a breakage and not as a permanent plastic deformation. Thus, our object deforms elastically until it breaks when reaching the ultimate tensile strength.

Linear shape functions yield constant derivatives $\nabla_{\mathbf{X}}\boldsymbol{u}$ (see the explanation after (3.24)), a constant linearized Green Lagrange strain tensor $\boldsymbol{\varepsilon}_{gl,l}$ and thus a constant stress tensor in every element. When using quadratic shape functions, the stress tensor changes non-linearly inside the element. This has a positive effect on the correct calculation of the stress inside of the elements [60]. We calculate the stress tensor at the Gauss integration points and extrapolate the values to the six element nodes. Since the stress depends upon the spatial derivative of the shape functions and these are only continuous and not differentiable at the element boundaries, for both kinds of shape functions, there are jumps of the stress tensor values between neighboring elements. But with quadratic shape functions, due to the improved reality of the stress calculation inside the elements, the difference of the stress tensors between neighboring elements is smaller than for linear shape functions. Therefore, in this work we

choose to simulate the deformation and to calculate the stress using quadratic shape functions.

In order to smooth the jump between neighboring elements, the stress tensor of the element node around a global node that has the highest principal stress $\sigma_{P,+}$ is assigned to the global node. Figure 6.12 displays this approach on two examples: an L-shaped and a precut sheet. Note, the calculation of the stress tensors and thus the principal stresses depend on the number of elements attached to a node. Since edge middle nodes are only attached to two elements, it is difficult to correctly calculate their smoothed strain and it might be interesting to use another smoothing methods that weighs all the stress values of the attached elements.

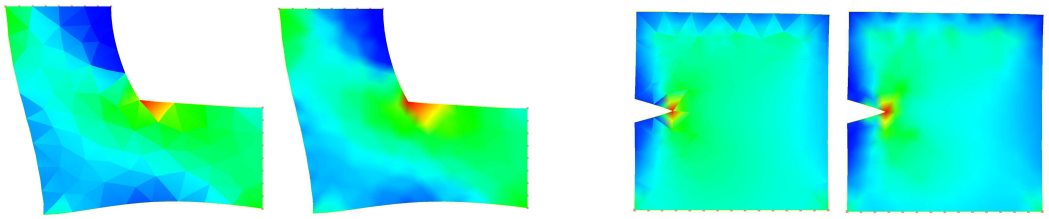


Figure 6.12: L-shape *(two images left)* and precut sheet *(two images right)* with the interpolated principal stress σ_P: elementwise *(left subimages)* and smoothed by choosing the maximal nodal element value *(right subimages)*

In order to separate the triangular elements, we identify the direction of the principal stress and simply disconnect existing egdes that follow the best the direction of the fracture. We choose the minimal angle between the fracture direction and the vectors from the point with the maximal stress to the end points of the edges. Thus we choose the approach depicted in *dark green* in figure 6.13 (left). Similarly to the calculation of the stress, the number of edges attached to the node with the maximal principal stress determines the possibility of the mesh to react to a fracture direction.

The separation at the edge can be combined with the snapping of the edge nodes to the fracture direction, where the boundary nodes are not snapped to maintain the geometry of the object. Figure 6.13 (right) reveals, that our choice for the edge maintains a better mesh quality than figure 6.13 (middle). If the object is deformed, one has to ensure, that the moving of nodes to the fracture direction does not change the stress inside the object. In our implementation, the strain and the stress are calculated based on the difference of the undeformed position $\mathbf{P}_k$ and the current position $\mathbf{p}_k$. Thus to keep the same stress inside the object, we calculate the barycentric coordinates for a point on the fracture direction and transform it to the undeformed configuration using the shape functions. Note that the calculation of the barycentric

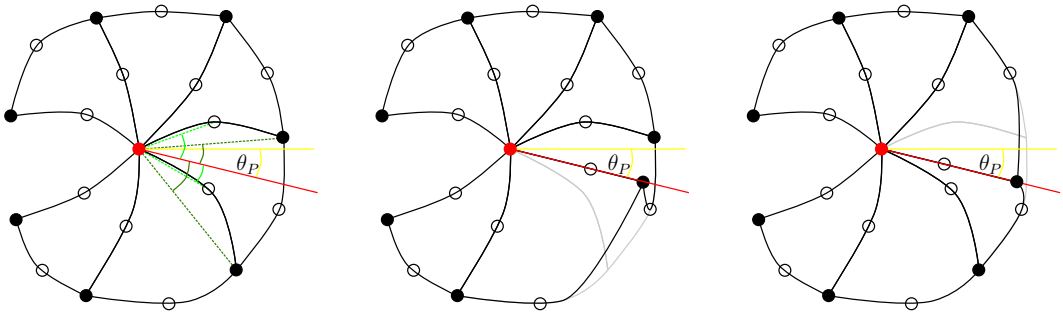


Figure 6.13: Snapping of quadratic triangles to a fracture direction *(red line)* that starts from a node with the maximal principal stress $\sigma_{P,+}$ *(red dot)*: *Left:* comparison of angles between edges and fracture direction *(different approaches in green)*; Snapping of lower *(middle)* (and respectively upper *(right)*) edge, based on criteria presented in *light green* (and respectively *dark green*) on the *left*

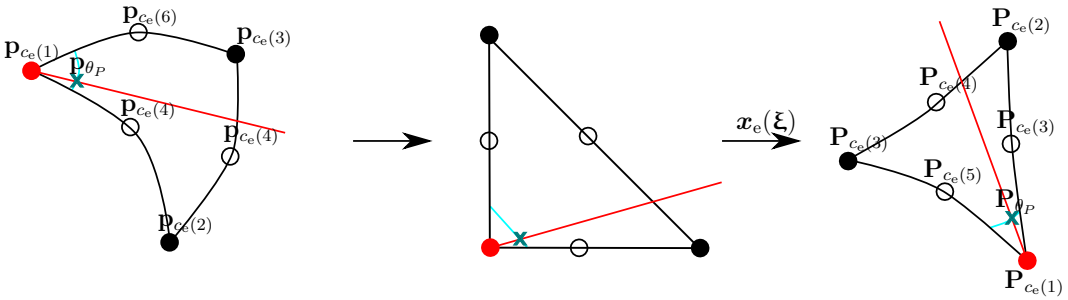


Figure 6.14: Calculation of the fracture direction in the material configuration: *Left:* calculation of the intersection between a curve inside an element with the fracture direction in spatial configuration to obtain the barycentric coordinates of the point *(middle)*; *Right:* fracture direction in material configuration

coordinates results in a system of quadratic functions. We use the intersection $\mathbf{p}_{\theta_P}$ of the fracture direction with a curve inside an element simplifying the problem to one quadratic equation. Then the undeformed positions are snapped on the fracture direction that is calculated using the position $\mathbf{P}_{\theta_P}$ in the undeformed configuration, see figure 6.14.

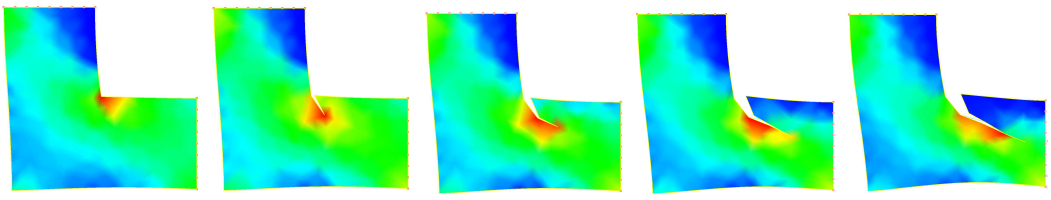


Figure 6.15: Fracture on the L-shape without using snapping

We present preliminary results on a L-shaped membrane (see figure 6.15) and precut sheet in the following. Observations reveal, that the snapping

results in a deterioration of the stress calculation due to deformed elements and finally in instabilities (see figure 6.16 (middle)). While the effect is reduced when using the transformation of the fracture direction to the undeformed configuration (see figure 6.16 (right)), the approach without snapping has the highest stability. However, this approach can only follow the fracture path using existing edges and thus is inaccurate. Beyond that, when considering

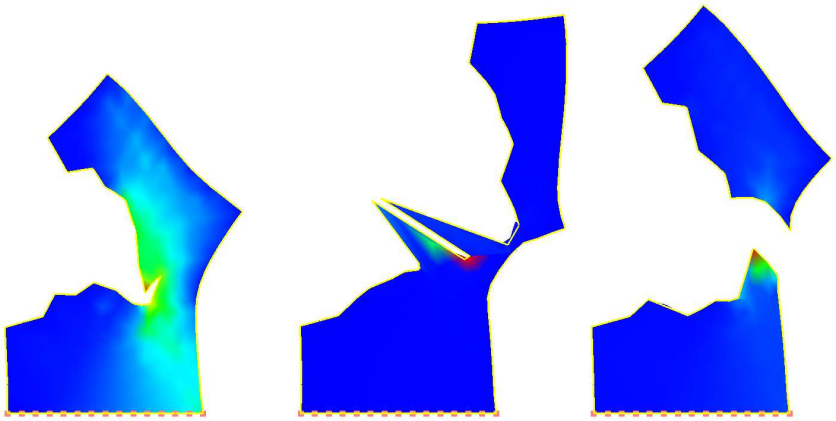


Figure 6.16: Fracture of a thin sheet without snapping *(left)* and with snapping *(middle, right)*. Without *(middle)* and with *(right)* transforming the fracture direction in the undeformed configuration

curved instead of plane geometries, snapping the nodes potentially changes the topology (see figure 6.4).

To conclude, in this section we presented an approach for the simulation of fractures on triangular shell elements. Theoretical considerations and preliminary results on the stress calculation confirm the interest of using quadratic shape functions. The separation is performed along the existing edges and the accurate following of the fracture direction is improved using node snapping. However, due to greatly deformed elements, the stability is deteriorated.

Chapter 7

Augmented reality with topological changes

Contents

Augmented reality has shown significant promise in overcoming certain visualization and interaction challenges in various domains such as medicine, construction, advertising, manufacturing, and gaming. Despite the promise of augmented reality and its successful application to many domains, significant research challenges remain. Among these challenges is the augmentation of non-rigid structures, which can undergo topological changes, such as fracture, tearing or cutting. This is for instance the case in minimally invasive surgery, which has gained popularity and became a well-established procedure thanks to its benefits for the patient, in particular with shortened recovery times. This surgery remains complex from a surgical point of view, mainly because of the reduced field of view which considerably impacts depth perception and surgical navigation (for further arguments see chapter 1). The introduction of augmented reality during surgery has the potential to significantly improve these limitations [67]. Despite such recent improvements, only few studies have investigated in the impact of cutting or resection actions performed during the operation. Given that these are essential steps of any surgical procedure, it is obvious that if the meshes of the underlying mechanical model are not correctly modified, significant errors are generated in the registration and consequently in the estimation of internal structures or tumor localization.

The motivation of our work is to address these issues. We describe the context and begin with a brief state of the arts, which contains the main ideas used for augmented reality applications. Followed by a detailed description of the placement of our work stating our main contributions, we continue with the error assessment, that can be used in our setting. Then, we proceed with the theory of our approach addressing the detection of topological changes from images and a mechanical model. We apply the approach on cut and torn silicone bands and present results for cut organs.

Note, that the work presented in this chapter closely follows our published work [78, 79, 82]. However, additional informations are provided allowing for an improved understanding and an easier identification of strengths and weaknesses coming with our approach.

7.1 Related works

In the context of non-rigid objects, two main material behaviors can be identified: inextensible ones such as clothes and elastic ones such as soft tissues. First, inextensible surfaces have been considered with the exploitation of dis-

tance constraints. The surface deformation is essentially computed through parametric geometrical models [5, 85, 89, 132]. Approaches based on learning methods have also been considered. The solution is estimated from a representative sample of possible shapes using a dimensionality reduction process [95, 96].

As regards elastic behaviors, geometrical properties can not be exploited. The closed-form solution constrained by shading information [64] can capture stretching surfaces and yields good results. But the method assumes a Lambertian surface with a single point light source what greatly limits the possible applications.

More recently, Agudo *et al* [1] describe elastic shapes with physical models and combine the finite element method with an extended Kalman filter. Other approaches try to minimize a stretching energy [54] that unifies geometric and mechanical constraints using a projective camera and locally linear deformations. In a similar way, non-linear elastic models [38] have been used to augment highly elastic objects. The model is constrained by features extracted from a single view camera and a set of boundary points.

The aforementioned methods cope with the registration or augmentation of surfaces. Augmenting 3D objects requires a more advanced acquisition process. For instance, a stereoscopic visual tracking [39] can be associated with a physical model to augment a volumetric object. A regularization mechanism makes the method more robust to errors in the tracking process. In a similar way, Leiza *et al* [52] exploit a depth camera to capture a 3D deformed shape.

The approach presented by Petit *et al* [86] fits the point cloud provided by an RGB-D sensor with a tetrahedral mesh using a geometrical point-to-point coupling. Linear elastic external forces are applied on an adapted corotational finite element model increasing the weight of the forces applied to the contour of the object. The method supports occlusion of the object and strong or highly elastic deformations. It has been extended, but only for non-elastic objects, to cope with fractures [87] that appear when the maximal eigenvalue of the stress tensor exceeds the toughness of the material.

Tsoli et al [120] use an RGB-D sensor for 3D tracking of deformable surfaces with dynamic topology. After a manual initial registration of the surface template, the approach is restricted to inextensible materials and therefore can remove overstretched edges of the template to account for topological changes. Based on geometrical considerations, the approach is independent of material behaviours and works with similar parameters for different scenarios, e.g. multiple or intersecting cuts.

In the context of Computer Assisted Surgery, patient-specific biomechanical models demonstrated their relevance for volume registration. They take into account anisotropic elastic deformations to infer in-depth structure motion [40, 117]. Pratt *et al* use a 4D scan [91] of the heart and a biomechanical model to couple the surface motion with external forces that emanate from camera data. This method uses the cyclic pattern of the heart deformations to improve the registration. A local tuning is used to propagate the surface deformation to in-depth invisible structures. In the context of liver surgery, [40] used a heterogeneous model that takes into account the vascular network to improve the soft tissue behavior while real-time performance is obtained using adequate mesh resolution and pre-computed solvers. In [117], a physics-based shape matching approach is proposed. Non-rigid registration between the pre-operative elastic model and the intra-operative organ shape is modelled as an electrostatic-elastic problem. The elastic model is electrically charged to slide into an oppositely charged organ shape representation.

Cuts and resection are essential in surgical procedures: The virtual model on which the augmented view relies has to be updated to allow a sound localization of the internal structures or tumors. In the context of image-guided neurosurgery, Ferrant *et al.* [30] handle the registration issues induced by tumor resection using the removal of the elements of the brain model that contains the resected tumor and the surrounded area.

In conclusion, providing information using augmented reality for topological changes like cutting and tearing of elastic objects has not yet been addressed in the literature to our knowledge. Moreover, the existing approaches that consider topological changes rely on RGB-D sensors or other sophisticated imaging techniques [30] when detecting the topological changes. In order to compare our approach to the state of the art, the following section specifies the setting in which we place our work and thus allows to better identify our contributions.

7.2 Setting of our work and contributions

The planification phase of laparoscopic surgeries includes the examination of the patient with imaging systems such as computer tomography (CT) or magnetic resonance imaging (MRI). Based on these systems, the three dimensional geometry of an organ can be retrieved yielding an improved analysis. In the intervention, most surgeons rely on a monoscopic camera, resulting in two dimensional images, that show the potentially deformed organ from another point of view.

In order to allow for a general applicability of our augmented reality solutions, we propose to use a monoscopic video stream. Moreover, we use a mechanical model, the virtual representation, which is retrieved from one of the imaging systems mentioned in the previous paragraph. We assume the model to be correctly aligned to the monoscopic image at the beginning of our simulation.

Our main contribution addresses the identification of topological changes, like cuts and tearing, by comparing the motion of the object tracked in the video and its virtual representation. Hereby, our method does not rely on computer vision techniques to detect if an instrument is generating the cut, making this approach also suitable for detecting the tearing of an object.

The proposed idea has been evaluated on highly elastic silicone bands, on *in vivo* livers and *ex vivo* kidneys. Beyond the update due to topological changes of the volumetric objects, internal structures of the organs are updated and overlaid on the view, giving important additional information. Pre-cut, cut and torn objects have been considered and results are visually satisfying.

In order to correctly evaluate our approach, the following section introduces error measures, that are used in the following chapters.

7.3 Detection of topological changes – overview

In the following sections we describe our idea to detect topological changes, that are applied to the mechanical model using the method presented in chapter 5. Figure A.29 displays the workflow of our approach when applied in the medical setting, which can be applied to other settings as well. The figure gives a valuable overview of our method by providing links to the subsections. The work presented in the following closely follows our published work, mainly using [82], but also based on [78, 79].

In the following, we use $\mathrm{X}_{\mathrm{E}}^{\mathrm{S}}$ to name a variable X of the entity E in the simulation state S. The variable X can denote either one point $\mathbf{p}$, a set of points $\mathcal{P}$, the connectivity c of one FEM element or the connectivity $\mathcal{C}$ of a set of elements. The name E denotes the entity the variable belongs to. In the case of a 3D object we use V for the volume, ∂V for the surface of the volume, I for the interior or internal structures and F for the features. In a 2D image, we use f for the features. In order to calculate the 3D features based on the 2D image features we use the position of the camera $\mathbf{v}$ and different projections. Note, that different representations of the 3D features exist and

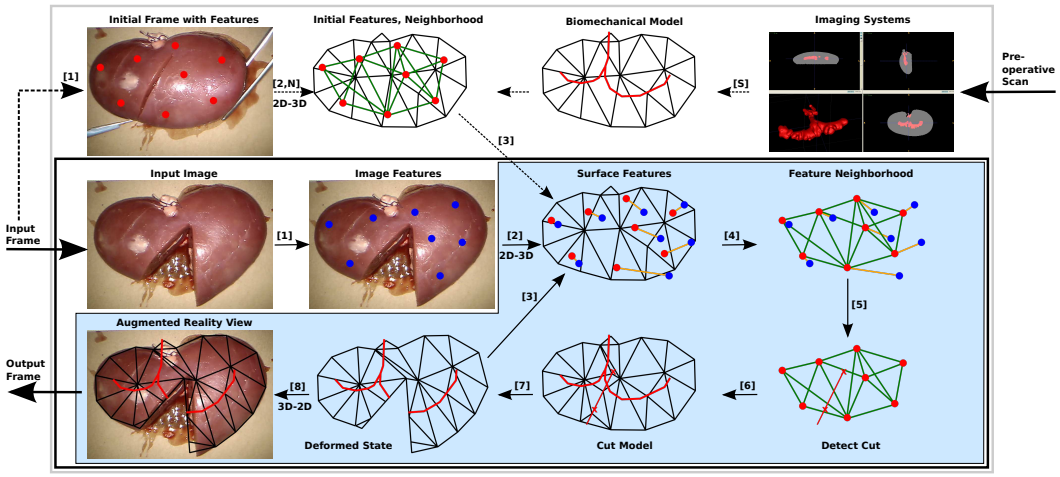


Figure 7.1: Pipeline of our method in the medical setting: [S] Segmentation, [1] identify image features $\mathbf{p}_{l,f}$ (see subsection 7.4.1), [2] map image features on model (see subsection 7.4.2), [N] construct feature neighbourhood $\mathcal{C}_F^0$ (see subsection 7.5.2), [3] use initial/updated model $(\mathcal{P}_V, \mathcal{C}_V)$ and compare image $\mathbf{p}_{l,f}$ and surface features $\mathbf{p}_{l,F}$ by [4] calculating the measure μ_{lm} on the neighborhood $\mathcal{C}_F^t$, [5] detect outliers, insert cut points $\mathbf{p}_c$, expand cut points to cut lines, [6] expand cut line to cut surface S (see subsection 7.5.3), update topology of model and internal structures (see 5 and 7.5.4), [7] solve minimization problem (7.2), [8] display model on output frame; the dashed flashes are performed at initialization; one frame with image features can be used several times, repeating the steps [2]-[8] in the blue area.

are presented in 7.4.2.

Finally, the status S can be either the initial state 0, the current state t or the target state 1. Let us point out that the *targets* denote here the features extracted from the images in the video stream that the virtual model tries to follow. Thus the targets change with each video frame. In addition, the current state is related to the actual position of the virtual model while it moves towards the target.

Before a surgical intervention, the organ and its internal structures are segmented from the pre-operative images. Separate meshes are built for each of the considered 3D objects. The virtual organ is discretized in a set of tetrahedra $\mathcal{C}_V^0$ whose elements are denoted c_V^0. Those elements connect the vertices $\mathcal{P}_V^0$ of the volume mesh that models the organ. The internal structures are discretized as a surface mesh $\mathcal{C}_I^0$ that connects the vertices $\mathcal{P}_I^0$. To geometrically bind the two initial sets of points, the $\mathcal{P}_I^0$ are expressed as barycentric coordinates of the $(\mathcal{P}_V^0, \mathcal{C}_V^0)$. Finally, constitutive laws and a set of parameters are chosen to approximate the elastic behavior of the organ and internal

structures.

Features points $\mathbf{p}^0_{l,f} \in \mathcal{P}^0_f \subsetneq \mathbb{N}^2$ of the real organ are identified in the laparoscopic view and are registered to the virtual model in its initial position. We denote this set as $\mathcal{P}^0_F = \{\mathbf{p}^0_{l,F}\} \subsetneq \mathbb{R}^3$. During the surgical intervention, the detected features are tracked in the video stream $\mathbf{p}^0_{l,f} \in \mathcal{P}^1_f \subsetneq \mathbb{N}^2$ and they form the target points $\mathcal{P}^1_F = \{\mathbf{p}^1_{l,F}\} \subsetneq \mathbb{R}^3$. The real and virtual organs are coupled, by means of these two sets of features points: the tracked features $\mathcal{P}^1_F$ and their initial registration $\mathcal{P}^0_F$, that move according to the deformation of the virtual model and whose current positions are $\mathcal{P}^t_F$.

Our method captures the deformations, and detects cuts and tears, through an analysis of the displacement field of these two point clouds. Detected cuts are reproduced on the virtual organ and the internal structures are updated providing additional information for the surgeon during the advancement of the surgical procedure.

7.4 Linking a model with a monoscopic image

To move the mechanical model, a virtual representation, according to the real object, we introduce spring forces that connect features in the image to features glued to the mechanical model. For the correct interpretation of the two dimensional points in the three dimensional space, the virtual representation of the object is assumed to be registered to the image at the beginning.

Features are identified and tracked in the video stream (see subsection 7.4.1), transformed to three dimensional features (see subsection 7.4.2) and deform the virtual representation (see subsection 7.4.3).

7.4.1 Identification and tracking of features in the 2D image

We rely on a simple combination of the speeded-up robust features (SURF) detector [7] and the iterative Lucas-Kanade optical flow [15, 53]. This combination shows effective results tracking highly elastic objects [38, 40].

From the first frame of a monocular video stream in a single view position we extract 2D features in their initial positions $\mathbf{p}^0_{l,f} \in \mathcal{P}^0_f$ using the speeded-up feature detector. In our scenarios the object does not cover the complete images of the video stream and masks are used to only consider features in the

region of interest, see figure 7.2. For the frame-to-frame tracking the Lucas-Kanade optical flow yields 2D features $\mathbf{p}^1_{l,f} \in \mathcal{P}^1_f$.

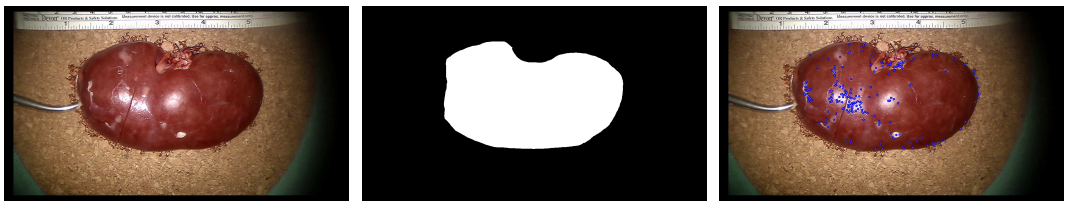


Figure 7.2: *From left to right*: Image from laparascopic view, applied mask, identified features *(blue dots)*

To ensure a trustworthy and robust detection of cuts, unrealistically tracked features – that jump or slide in improbable directions – need to be filtered out. For that, we compare the evolution of the features to determine outliers, looking for spatio-temporal discontinuities. Precisely, we analyze the evolution of the euclidean norm between features during a time step, i.e. we consider the measure $\|\mathbf{p}^1_{l,f}(t) - \mathbf{p}^1_{l,f}(t + \Delta t)\|$, and determine outliers comparing to the measures' average. If a feature $\mathbf{p}^1_{l,f}$ has been identified as an outlier, then it is not used any more.

7.4.2 Transformation of 2D to 3D features

Each feature point $\mathbf{p}_{l,f}$ is associated with a virtual feature point $\mathbf{p}_{l,F} \in \mathbb{R}^3$. Dependent on the scenario, the virtual feature points can be calculated in two different ways, projecting the feature points either onto the plane against which the video leans back in 3D or on the surface of the mechanical model. For the first idea, we calculate the initial virtual feature points $\mathbf{p}^0_{l,F_P}$ and the target virtual feature points $\mathbf{p}^1_{l,F_P}$ in the same way. Using the second idea, we first project the features on the surface of the object to retrieve the initial virtual surface feature points $\mathbf{p}^0_{l,F_S}$ and an initial set of target virtual surface feature points $\mathbf{p}^1_{l,F_S}$. A direct projection of the updated target features on the surface of the object is not possible: when the real object moves, its boundary goes beyond the boundary of the virtual object and a ray to the feature possibly does not hit the object, see figure 7.3 *(right)*. Thus for each updated set of target feature points, we update the target virtual surface feature points $\mathbf{p}^1_{l,F_S}$ using the intercept theorem and the virtual target points $\mathbf{p}^1_{l,F_P}$:

$$\mathbf{p}^1_{l,F_S}(t + \Delta t) = \mathbf{p}^1_{l,F_S}(t) + \left(\mathbf{p}^1_{l,F_P}(t + \Delta t) - \mathbf{p}^1_{l,F_P}(t)\right) \frac{d(\mathbf{p}^1_{l,F_P}(t), \mathbf{v})}{d(\mathbf{p}^1_{l,F_S}(t), \mathbf{v})} \tag{7.1}$$

Figure 7.3 and figure 7.4 display the calculations on several examples.

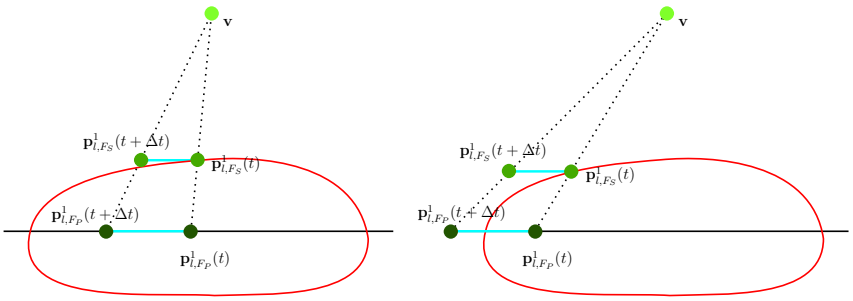


Figure 7.3: Calculation of the virtual features using the intercept theorem, with the projection plane *(black line)*, the object *(red curve)*, the camera position $\mathbf{v}$ and the rays defining the virtual features *(dashed black lines)*

Since the first idea applies feature displacements at the position of the projection plane, they might not be interpreted correctly, see figure 7.4. While this is not a problem for the projection on the surface, the object might undergo unrealistic deformations, as the spring forces introduced in the next subsection only affect surface elements. Obviously, since the 2D image does not give information on the depth, both ideas are challenged with movements of the objects that go into the view direction.

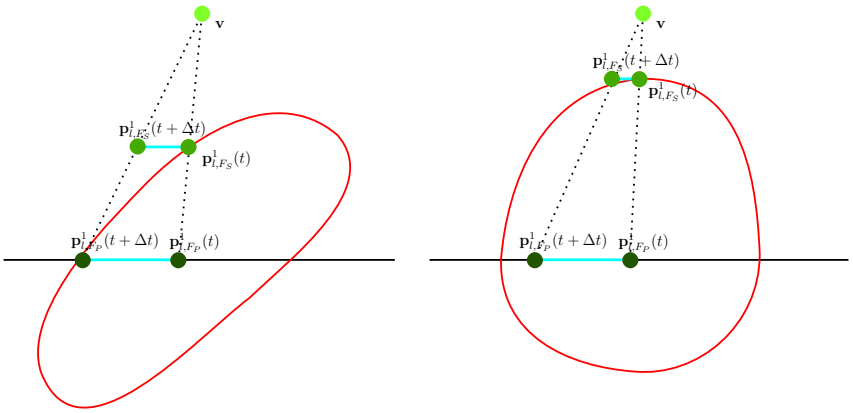


Figure 7.4: Virtual features on thin object non-perpendicular to the camera *(left)* and a thick object *(right)*, with the projection plane *(black line)*, the object *(red curve)*, the camera position $\mathbf{v}$ and the rays defining the virtual features *(dashed black lines)*

The figures reveal, when the difference between image plane of the recorded object is placed close to the surface of the virtual object, then the difference of the target virtual feature points $\mathbf{p}^1_{l,F_P}$ and surface feature points $\mathbf{p}^1_{l,F_S}$ vanishes. This is particularly true for flat objects, since the slope of the surface is close to the slope of the plane, and for thin objects, since a placement of the image plane inside of the object directly means, that it is close to the surface.

The idea applied has to be chosen dependent on the scenario, in the following, we use $\mathbf{p}^0_{l,F} \in \mathcal{P}_{\partial V}$ and $\mathbf{p}^1_{l,F}$. If a stereoscopic view is used, these features are directly retrieved from the video stream, but in our work, we only used the monoscopic view allowing for an application in many different settings.

7.4.3 Mechanical linking of the image and the object

In the simulation, an additional set of virtual feature points are used. The *current* virtual features points $\mathbf{p}^t_F$ are initialized at the positions of the initial virtual feature points $\mathbf{p}^0_F$ and barycentric coordinates in the FEM mesh of the object are calculated.

The choice of the constitutive law determines the set of deformations that can be represented while discriminating non-plausible configurations that could be induced by wrong target surface feature point cloud $\mathcal{P}^1_F$. We apply the corotational linear elasticity that secures the rotational invariance and thus nonlinear characteristics, while keeping the simplicity of the stress-deformation relationship in linear materials [106]. For more information, please refer to section 3.2.

The coupling of the real and virtual models is obtained with the introduction of spring forces between each surface feature point $\mathbf{p}^t_{l,F}(\mathcal{P}^t_V)$ and target surface feature point $\mathbf{p}^1_{l,F}$, which accumulate to the stretching potential energy:

$$\Pi_S(\mathcal{P}^t_F(\mathcal{P}^t_V), \mathcal{P}^1_F) = \sum_l \frac{1}{2} k_l \; d_F(\mathbf{p}^t_{l,F}(\mathcal{P}^t_V), \mathbf{p}^1_{l,F})^2 \tag{7.2}$$

The parameters k_l are experimentally chosen and are in the same order of magnitude than the Young's modulus of the deformable object. The updated set of vertices $\mathcal{P}^t_V$ is obtained by solving the minimization problem between internal elastic potential and stretching potential energy

$$\underset{\mathcal{P}^t_V}{\operatorname{argmin}}(\Pi_{\text{int}}(\mathcal{P}^t_V, \mathcal{C}^0_V) + \Pi_S(\mathcal{P}^t_F(\mathcal{P}^t_V), \mathcal{P}^1_F)) \tag{7.3}$$

Further information on the calculation of the internal energy are detailed in [116], page 73-76. External energies, such as boundary and body forces, could be added, but are not necessary for our application, as the linking to the image already takes account of these forces. For further information on the internal energies and their derivations refer to the subsections 3.1 and 2.2.1. The surface feature points $\mathcal{P}^t_F(\mathcal{P}^t_V)$ and the internal structures $\mathcal{P}^t_I(\mathcal{P}^t_V)$ are updated applying their initial barycentric coordinates to the new positions of the $\mathcal{P}^t_V$.

Since we use a dynamic simulation, there is no need for boundary conditions to obtain a stable problem. We apply the Euler implicit method for the time integration, see section 3.3, and the minimization problem is solved every time step using the conjugate gradient method, see section 3.4.

7.5 Detecting topological changes in the image

The ideas applied in the previous section allow to move a virtual object according to the movement of a real object in the image. However, when topological changes as cuts or tears occur, the mechanical object is restricted to its initial topology, thus preventing to move correctly. In this section, we formulate the problem we are facing (in subsection 7.5.1) and identify the effect of topological changes in the image on the virtual object (in subsection 7.5.2). Subsection 7.5.3 discusses the detection and subsection 7.5.4 the propagation of topological changes to the internal structures of the mechanical model.

7.5.1 Problem formulation

We assume that the initial positions of the surface feature points $\mathcal{P}_F^0$, the virtual organ $\mathcal{P}_V^0$, the internal structures $\mathcal{P}_I^0$ and the target positions of the surface feature points $\mathcal{P}_F^1$ are given. The tetrahedral elements $\mathcal{C}_V^0$ – respectively the triangular ones $\mathcal{C}_I^0$ – connect $\mathcal{P}_V^0$ (and respectively $\mathcal{P}_I^0$) to the virtual organ (and the internal structures).

Our objective is to update the shape (geometry and topology) of the virtual organ $(\mathcal{P}_V^0, \mathcal{C}_V^0)$, based on the position of the target surface feature points $\mathcal{P}_F^1$. This includes the deformation and update of the topology of the model $(\mathcal{P}_V^t, \mathcal{C}_V^t)$ and the internal structures $(\mathcal{P}_I^t, \mathcal{C}_I^t)$. With the previous notation the problem can be formulated as:

$$\text{Given } \mathcal{P}_F^0, (\mathcal{P}_V^0, \mathcal{C}_V^0), (\mathcal{P}_I^0, \mathcal{C}_I^0) \text{ and } \mathcal{P}_F^1 \tag{7.4}$$

$$\text{Find } (\mathcal{P}_V^t, \mathcal{C}_V^t) \text{ solving (7.3)} \tag{7.5}$$

$$\text{and thus minimizing } d_F(\mathcal{P}_F^t(\mathcal{P}_V^t), \mathcal{P}_F^1) \tag{7.6}$$

When a ground truth of the surface of the volumetric model $(\mathcal{P}_{\partial V}^1, \mathcal{C}_{\partial V}^1)$ or of the internal structures $(\mathcal{P}_I^1, \mathcal{C}_I^1)$ are given, the problem could be complemented

by:

$$\text{Minimize } d_{\partial V}(\mathcal{P}^t_{\partial V}(\mathcal{P}^t_V), \mathcal{C}^t_{\partial V}), (\mathcal{P}^1_{\partial V}, \mathcal{C}^1_{\partial V})) \tag{7.7}$$
$$\text{Minimize } d_I((\mathcal{P}^t_I, \mathcal{C}^t_I), (\mathcal{P}^1_I, \mathcal{C}^1_I)) \tag{7.8}$$

The choice of the distances depends on the application and is discussed in the subsection 7.6. To solve the problem, the displacement field between the current features $\mathcal{P}^t_F(\mathcal{P}^t_V)$ and target features $\mathcal{P}^1_F$ is analyzed. Discontinuities in this field reveal an inconsistency between the biomechanical model $(\mathcal{P}^t_V, \mathcal{C}^t_V)$ and the target surface feature points $\mathcal{P}^1_F$ (see the previous subsection). Such an inconsistency can be triggered by a cut of the real model that moves the target surface feature points $\mathcal{P}^1_F$. The detection of a cut or other topological changes is discussed in the following subsection.

7.5.2 Effect of topological changes in the image on the virtual object

When the organ deforms smoothly, the biomechanical model $(\mathcal{P}^t_V, \mathcal{C}^0_V)$, and with it the surface feature points $\mathcal{P}^t_F(\mathcal{P}^t_V)$, can properly follow the target surface feature points $\mathcal{P}^1_F$. As soon as a cut occurs, observations of the surface feature points are twofold:

First, the uncut model $(\mathcal{P}^t_V, \mathcal{C}^0_V)$ becomes unable to adapt to the motion enforced by the stretching energy. In consequence, the vector between the current and the target positions $\mathbf{d}_l(\mathcal{P}^t_V, \mathcal{P}^1_F) = \mathbf{p}^t_{l,F}(\mathcal{P}^t_V) - \mathbf{p}^1_{l,F}$ diverges around the cut.

Secondly, the distance between two target feature points $\mathbf{p}^1_{l,F}$ and $\mathbf{p}^1_{m,F}$ that lie on two different sides of the cut increases much more than the average distance between feature points.

To exploit the observations, we call two feature points $\mathbf{p}^0_{l,F}$ and $\mathbf{p}^0_{m,F}$ neighbors iff their initial Euclidean distance $\delta^0_{\mathrm{lm},F} = \|\mathbf{p}^0_{l,F} - \mathbf{p}^0_{m,F}\|$ is smaller than a given radius $r_{\mathcal{P}_F}$, that is related to the density of the features. The Euclidian distances $\delta^t_{\mathrm{lm},F}$ and $\delta^1_{\mathrm{lm},F}$ are defined respectively. The neighborhood information is stored in the graph $\mathcal{C}^0_F = \{(l,m) | \delta^0_{\mathrm{lm},F} < r_{\mathcal{P}_F}\}$ (see figure A.29.[N]).

We conclude from the first observation: in the region of a cut, the physical model prevents the surface feature points $\mathcal{P}^t_F(\mathcal{P}^t_V)$ from moving towards their targets $\mathcal{P}^1_F$ and the vectors $\mathbf{d}_l(\mathcal{P}^t_V, \mathcal{P}^1_F)$ and $\mathbf{d}_m(\mathcal{P}^t_V, \mathcal{P}^1_F)$ – simply denoted $\mathbf{d}_l$ and $\mathbf{d}_m$ in the following – point in two different directions (compare figure A.30*(top right)* with *(bottom right)*). This difference is quantified with

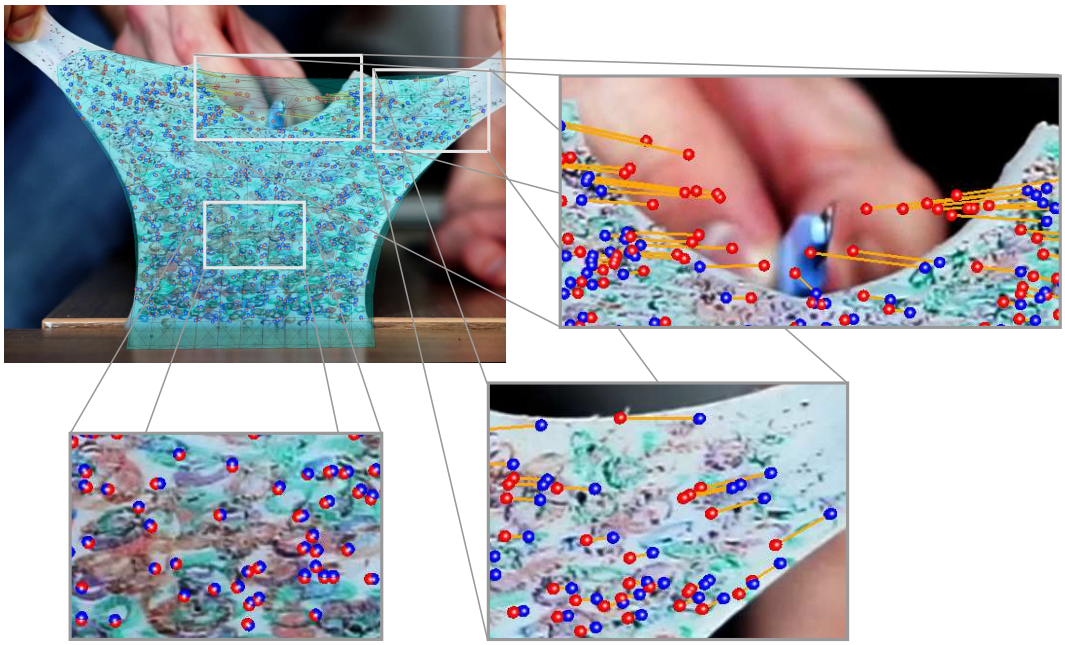


Figure 7.5: Effects of topological changes in image on virtual object *(turquoise)* with target *(blue dots)* and current features *(red dots)* connected by springs *(orange lines)*: *Top left*: complete object with overlayed virtual object; *Top right*: cut region; *Bottom left*: region without strong deformation; *Bottom right*: region with strong deformation

$$\mu_{lm,1} = d(\mathbf{d}_l, \mathbf{d}_m) \tag{7.9}$$

where d is the euclidean distance.

For the second observation, we evaluate the ratio

$$\mu_{lm,2} = \delta^t_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F} \quad \text{or} \quad \mu_{lm,2} = \delta^1_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F} \tag{7.10}$$

of the initial and current/final distance of neighboring surface feature points. An increase indicates either an elongation of the object or a separation due to a cut or a tear. These two cases are distinguished by comparing this ratio with its average in the neighborhood, see the next subsection.

While [78] relies on $\delta^t_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}$, [82] uses $\delta^1_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}$. The choice of [78] yields a higher stability for the simulation as the underlying mechanical model regularizes or smooths strong movements or jumps of the surface feature points. Applying $\delta^1_{\mathrm{lm},F}$ is closer to what happens with the real object and $\mu_{lm,2}$ is independent of the choice of the virtual model. Finally, the choice is dependent on the application and whether the surface feature points are trustworthy.

Based on these observations and introduced notations, we now can identify cut regions as described in the following subsection.

7.5.3 Interpretation of incompabilities between real and virtual object

Identification of a cut region: With the observations of the last subsection, we define the measure

$$\mu_{lm} = \mu_{lm,1}\mu_{lm,2}, \quad \forall (l,m) \in \mathcal{C}_F^0 \tag{7.11}$$

and we denote $\mu_\emptyset$ the mean value over $\mathcal{C}_F^0$. In a normal configuration, the deformations of the real object smoothly diffuse in the vicinity of each feature. Thus neighboring features should have motions that are similar in size and direction. In that case, the measure μ_{lm} remains small. In the case of a normal elongation, neighboring features are stretched. If the deformation stays below the mechanical strength of the virtual object, their associated virtual features controlled by the FEM mesh move the same way – with a possible latency. Thus the norm of the difference between $\mathbf{d}_l$ and $\mathbf{d}_m$ remain small and even if the two vectors have opposite directions, the measure μ_{lm} remains in the magnitude of $\mu_\emptyset$. In the case of a cut or a tear, the motion of the virtual features cannot follow the real features. Therefore, the vectors $\mathbf{d}_l$ and $\mathbf{d}_m$ elongate in opposite direction and the measure μ_{lm} takes values that are far beyond $\mu_\emptyset$. Outliers – i.e. pairs of divergent features – are identified as pairs $(l,m) \in \mathcal{C}_F^0$ such that $\mu_{lm} > \tau\mu_\emptyset$, with a threshold τ that depends on the scenario. Figure 7.6 (left) depicts the increase of the measure on an example of two neighboring features.

The measure defined in (7.11) contains parts that use informations purely from the features, e.g. for $\mu_{lm,2} = \delta^1_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}$, and from the features combined with the information from the mechanical model, e.g. $\mu_{lm,1} = d(\mathbf{d}_l, \mathbf{d}_m)$ or $\mu_{lm,2} = \delta^t_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}$. A third part of the measure that purely depends on the virtual model, e.g. using the stress or the strain inside the mechanical model, can allow for a better detection of a fracture, but has not yet been used.

Insertion of cut points: To detect the cuts, we first search for couples of neighboring feature points (l,m) that are outliers for the measure μ_{lm}. For two outliers (l_0,m_0), (l_1,m_1) with $\mu_{l_0m_0} > \mu_{l_1m_1} > \tau\mu_\emptyset$ the cut should be closer to the (l_0,m_0) couple. As soon as the number of outliers exceeds a given threshold $n(\{(l,m)|\mu_{lm} > \mu_\emptyset\})$, we insert a cut point $\mathbf{p}_c$ at the averaged barycenter of the outliers weighted with their respective measure.

Then, to avoid the insertion of another cut point in the next time step at the same location, we delete from the set $\mathcal{C}_F^t$ of neighbors the couples that cross the sphere of the radius $r_{\mathbf{p}_c}$ around the inserted cut point (see figure 7.7 left, middle). After the update of the neighbors, the measure defined in (7.11) is only evaluated on $\mathcal{C}_F^t$.

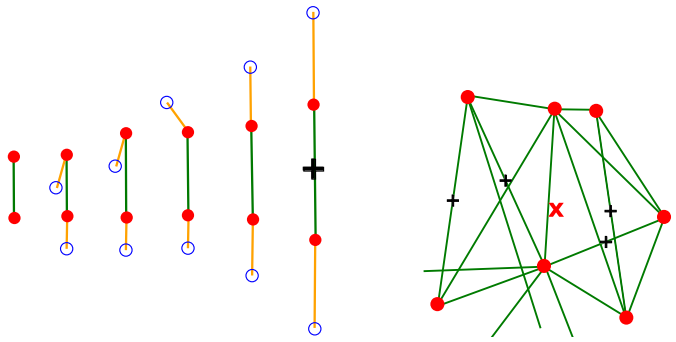


Figure 7.6: *Left*: Neighboring features: *From left to right* initial position, increasing measure, an identified outlier *(black cross)*; *Right*: Initialization of a cut point as the weighted barycenter of the outliers *(black crosses)*; Target *(blue dots)* and current features *(red dots)* are connected by springs *(orange lines)*, the neighborhood information between two features displayed in *green*

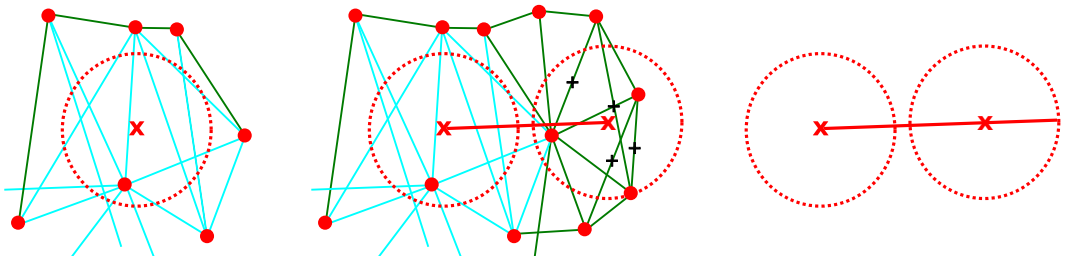


Figure 7.7: *Left*: removed connectivity information *(turquoise)* in circle around new cut point *(red cross)*; *Middle*: insertion of a second cut point yields a cut line (polygon) *(red line)*; *Right*: cut polygon is extended until the end of its radius $r_{\mathbf{p}_c}$

With the first insertion of a cut point, we introduce a sequence of cut points $\{\mathbf{p}_{i_0,c}, \ldots, \mathbf{p}_{i_{n(\mathbf{p}_c)},c}\}$ with $i_0 = i_{n(\mathbf{p}_c)}$, that extends to a cut polygon continuously, i.e. new cut points can be inserted before the first and after the last extremity. New cut points are inserted before or after the nearest extremity (figure 7.7 middle) and the polygon fits the widening of the cut.

Our first publication [79] uses cut lines that advance in two directions instead of cut points that are connected to form lines. An angular restriction in which the cut can go replaces the removal of connectivity information mentioned above. This approach showed nice results, but when the objects move or deform, the direction of the line sticked to the object changes and results in unrealistic behaviours or no detections of cuts due to the angular restriction.

Optimizations: The parameters $n(\{(l,m)|\mu_{lm} > \mu_\emptyset\})$ and $r_{\mathbf{p}_c}$ allow to balance between the precision of the cut and the robustness of the cut detection.

Setting these parameters on a high value, i.e. aiming rather at the robustness than at the precision of the cut detection, reduces the ability of the detection to react instantly on an emerging cut. That means, the detection has the tendency to lag behind. To overcome this negative side effect, we insert another cut point located at the intersection between the line connecting the last and the current cut point and the sphere around the current cut point (figure 7.7 right).

The surface feature points do not go beyond the boundary, i.e. inserted cut points stop before the boundary of the object, preventing the occurrence of complete cuts. The preventive step of the last paragraph results in a cut over the boundary which alleviates the problem without adding a parameter.

For the publications [78, 79], the additional cut point has not been inserted. Thus the boundary is treated differentely: we check after each cut point insertion whether there are still neighboring virtual features in front of the cut. If this is not the case, the cut line is lengthened to cross the boundary of the FEM mesh and the limit of real object.

Expansion to cut surface and propagation to mechanical model: Using the direction of the camera or a predefined vector, the separation polygon can be extruded to a separation surface S, which can be forwarded to any cutting algorithm [128]. In our work we use the approach presented in chapter 5, which is a combination with a node snapping technique [80].

When the detected cutting plane crosses springs, a target surface point $\mathbf{p}_{l,F}^1$ potentially pulls the current surface point $\mathbf{p}_{l,F}^t$ on the other side of the cut and thus closes the topological change performed on the mechanical model. To prevent this behaviour, we introduce a margin r around the cut surface, that deletes springs connected to a current surface point with a distance smaller than the margin, see figure 7.8.

7.5.4 Handling internal structures and closing remarks

In our work we use the biomechanical model $(\mathcal{P}_V^t, \mathcal{C}_V^t)$ of the virtual organ to update internal structures given by a preoperative scan. For that, the initial positions $\mathcal{P}_I^0$ and the initial triangular surface connectivity $\mathcal{C}_I^0$ represent the internal structures like vessels, the urinary system, tumors or others. At initiation, barycentric coordinates are computed for each point $\mathbf{p}_{l,I}^0 \in \mathcal{P}_I^0$ dependent on the shape functions used in the finite element formulation.

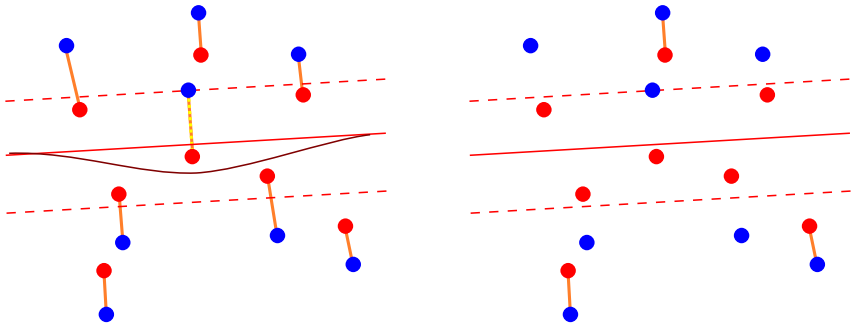


Figure 7.8: *Left*: Example of a detected cut *(red line)* that would be closed due to a spring intersecting with the cut *(yellow dashed line)*, the actual cut is plot in *dark red*, target features *(blue dots)* pull the current features *(red dots)* with the springs *(orange lines)* away from the cut; *Right*: springs are removed in margin *(dashed red lines)* around the detected cut

Similar to the surface feature points, this helps to express the current position of the internal structures dependent on the biomechanical model, i.e. $\mathbf{p}_{l,I}^t = \mathbf{p}_{l,I}^t(\mathcal{P}_V^t, \mathcal{C}_V^t) \in \mathcal{P}_I^t(\mathcal{P}_V^t, \mathcal{C}_V^t)$.

When the cut or the tearing of the organ yields a detection of a separation surface S as being described in subsection 7.5.3, we aim on propagating this change to the internal structures. In order to correctly update the internal structures, we update the connectivity $\mathcal{C}_I^t$ by deleting the triangles that intersect with the separation surface S, i.e.

$$\mathcal{C}_I^t = \{(c_{1,I}, c_{2,I}, c_{3,I}) | [\mathbf{p}_{c_{1,I},I}^t, \mathbf{p}_{c_{2,I},I}^t, \mathbf{p}_{c_{3,I},I}^t] \cap S = \emptyset\} \tag{7.12}$$

Thus with this update, our detection algorithm described in this chapter combined with our cutting algorithm [80] allows to apply topological changes to a biomechanical model and its internal structures. These topological changes update the mechanical model to a new representation of the mesh $(\mathcal{P}_V^t, \mathcal{C}_V^t)$, thus the barycentric mapping for the internal structures $\mathbf{p}_{l,I}^t(\mathcal{P}_V^t, \mathcal{C}_V^t) \in \mathcal{P}_I^t(\mathcal{P}_V^t, \mathcal{C}_V^t)$ needs to be reinitialized. This reinitialization is only of local nature, since in the genereal setting only a part of the mesh is cut.

Proposed ideas have been implemented using the open source framework SOFA [28] for the simulation of the deformation of the object. In the following, we give further details on, and we demonstrate the potential of, our approach to detect topological changes by the analysis of the motion obtained from a monocular video and to incorporate these changes in a virtual volumetric deformable model used for augmented reality. Different scenarios are possible for ongoing topological changes in the video stream: either the object is precut and we record a deformation opening the cut, or the object is undergoing topological changes as cutting and tearing while being recorded.

Our algorithm can address both scenarios, even with the increased degree of difficulty in the second case, see section 7.7 for results on silicone bands. Then, section 7.8 presents cuts on *in-vivo* livers and *ex-vivo* kidneys to show the clinical relevance of our work and to evaluate the accuracy.

The results presented in this chapter are part of our publications [79, 78, 82], which are used as the main landmarks. Since the ideas evolved in these publications, table 7.1 gives an overview on the applied ideas.

Example	**Reference**	**Advancement by...**	**Virtual features $\mathbf{p}_{l,F}$**	**Measure μ_{lm}**
Silicone cut left	[79, 82]	Cut lines	$\mathbf{p}_{l,F_P}$	$d(\mathbf{d}_l, \mathbf{d}_m)$
Silicone cut middle	[79, 82]	Cut lines	$\mathbf{p}_{l,F_P}$	$d(\mathbf{d}_l, \mathbf{d}_m)$
Silicone v-example	[78]	Cut points	$\mathbf{p}_{l,F_P}$	$\delta_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}\ d(\mathbf{d}_l, \mathbf{d}_m)$
Silicone s-example	[78]	Cut points	$\mathbf{p}_{l,F_P}$	$\delta_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}\ d(\mathbf{d}_l, \mathbf{d}_m)$
In-vivo liver 1	[79]	Cut lines	$\mathbf{p}_{l,F_P}$	$d(\mathbf{d}_l, \mathbf{d}_m)$
In-vivo liver 2		Cut points	$\mathbf{p}_{l,F_P}$	$\delta_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}\ d(\mathbf{d}_l, \mathbf{d}_m)$
Ex-vivo kidney 1	[82]	Cut points	$\mathbf{p}_{l,F_S}$	$\delta^1_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}\ d(\mathbf{d}_l, \mathbf{d}_m)$
Ex-vivo kidney 2	[82]	Cut points	$\mathbf{p}_{l,F_S}$	$\delta^1_{\mathrm{lm},F}/\delta^0_{\mathrm{lm},F}\ d(\mathbf{d}_l, \mathbf{d}_m)$

Table 7.1: Summary of the applied ideas for the examples of this chapter

7.6 Error assessment

In the context of our work, we can compare our results either by overlaying them on the two dimensional image or by direct comparison to a three dimensional reference solution.

For the comparison based on the image of the size $[0, n(x)] \times [0, n(y)] \subset \mathbb{N}\times\mathbb{N}$, the object in the image is identified and classified as a set $X \subseteq [0, n(x)]\times [0, n(y)]$ and similarly, the overlay of the virtual object in the image is classified as a set $Y \subseteq [0, n(x)] \times [0, n(y)]$. Then the *Dice's coefficient* compares the intersection of two sets X and Y with the sum of the two sets:

$$d_D(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \tag{7.13}$$

where $|X|$ is the number of elements in X. This number is finite, as we have a finite number of pixels in each image $[0, n(x)] \times [0, n(y)] \subset \mathbb{N} \times \mathbb{N}$. A few examples for this coefficient are displayed in figure 7.9. Our scenarios are in most cases similar to the second image from the right, as we do not remove a part of the object.

If a reference solution in the three dimensional space exists, the surface of the objects or internal structures is particularly important, as it is visible when being overlaid onto the image. Defining X as the surface of the real or

Figure 7.9: Example for dice coefficients *from left to right*: $0, 0.5, 0.75, 1, 2*15/(16+15) = 0.968, 2*15/(16+16) = 0.9375, 2*14/(16+14) = 0.933$

reference solution and Y as the surface of its virtual simulated solution, we can define the *Hausdorff distance* as

$$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\} \tag{7.14}$$

An example for the calculation of the Hausdorff distance is displayed in figure 7.10 (left). As it can be seen, the two parts of the Hausdorff distance yield different results. Therefore in many cases it is valuable to consider the two parts separately in order to better identify the areas where errors occur, see figure 7.10 (right) for an example. Beyond that, average values give added information, as they can clarify whether the maximal value is an outlier.

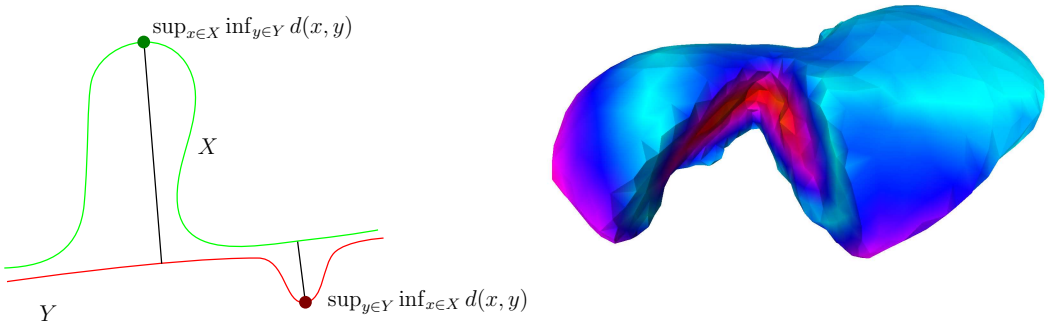


Figure 7.10: Example for the calculation of the Hausdorff distance in two dimensions *(left)* and example of a color plot for $\inf_{y\in Y} d(x,y)\ \forall x \in X$ *(right: low values in blue, high values in red/orange)*

7.7 Results on silicone data

Our algorithm was applied to several scenarios involving highly elastic silicone bands which are cut and torn. A video is recorded while manipulated to induce deformations showing the topological changes. Both scenarios are presented: the precut (see subsection 7.7.1) and the general case (see subsection 7.7.2), detecting source independent topological changes. Results are quantitatively

evaluated using the classical dice coefficient on the two dimensional domain of the video data (see subsection 7.6 for more details). The feature tracking, deformable model update, cut detection and topological changes are performed in real-time. The majority of the computational cost is dedicated to solving the minimization problem mentionned in section 7.4.3. Our separation detection algorithm only adds a minor computational load in most examples 1% of the complete computational cost, which is equivalent to an average of about 0.20 and 0.25 ms. We consider this as very promising, since the implementations have not been optimized and could be parallized if necessary. This makes our approach specifically interesting as added functionality for any simulation framework that can deal with topological changes.

7.7.1 Precut objects

In surgical interventions, the cutting process might not be visible, due to instruments and fog in the field of view. Thus it is interesting to update the deformation and the topology of a biomechanical model based on a precut object, after the action of cutting itself. In order to use our algorithm for similar scenarios, surgeons need to first perform the cut and then close it. The mechanical model has to be registered to the organ in its initial position. When recording an opening the cut, our algorithm allows for an automatic udate of the model and thus an additional information coming from the three dimensional biomechanical model for the surgeon.

For results on surgical interventions please refer to 7.8, in this subsection, we discuss results on two precut silicone bands undergoing strong deformations [79]. We consider an object cut from the side and in the middle, thus cuts with and without intersection of the boundary of the object in the video stream. Note, that the results use the first version of our algorithm, where two cut lines advance to form the cut [79] (see subsection 7.5.3). Our algo-

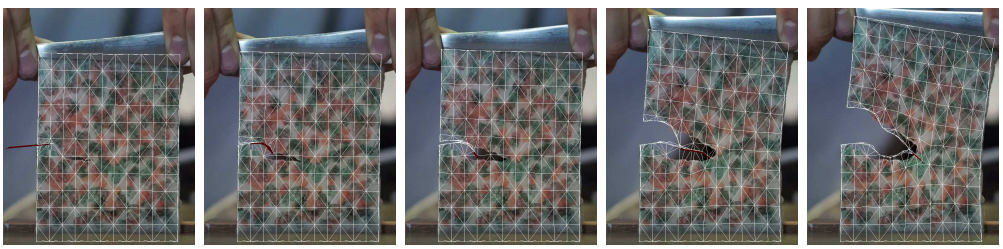


Figure 7.11: Advancement of the cut *(dark red)* inside the virtual object *(edges are plot as white lines)*

rithm potentially detects several cut lines in one step of the simulation, that

are extruded to cut planes to separate the object, see figure 7.11. After the detection of cut lines, the object undergoes strong deformations without further detections (see figure 7.12), demonstrating the ability to distinguish between large strains and cuts. Even with the loss of features in the example of a cut from the left, since part of the object is not visible at the end of the video, the updated mechanical model shows a convincing overlay with the real silicone bands. For the second case, the object returns to its initial state and the mechanical object follows without further cut detections.

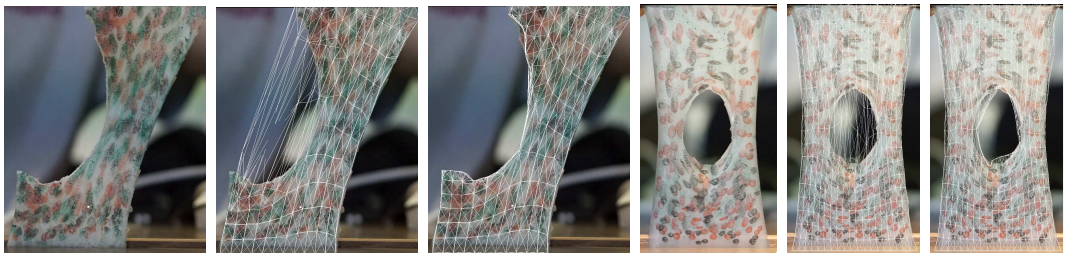


Figure 7.12: Detection and simulation of cuts in silicone bands demonstrating the ability of our approach to distinguish between large strains (186% and 166%) and cuts, augmented with an uncut/cut model *(edges are plot as white lines)*

Evaluations confirm the positive outcome of the visual observations: The results for the first case scenario (object cut on the side) are 0.815 when not accounting for topological changes, and 0.952 when using our algorithm. Results for the second case scenario (object cut in the middle) are 0.900 for the uncut mesh, and 0.964 when applying our method.

7.7.2 Source independent detection

If an object is under tension, initialized cuts potentially result in tearing or fractures. In this section we consider this process on two other silicone bands, that first undergo great deformations, and then are cut and torn.

V-Example: In the first example – displayed in figure 7.13 – we stretch a soft, highly extendable silicone cuboid in its length and width, while the cuboid is fixed at its stem. Almost at the maximal level of elongation before tearing, we perform a vertical cut, that is proceeded by a separation due to tearing. The rectangular shape changes until it resembles the letter "v" – this is why we call this example in the following the v-example. The real cuboid is 10cm in width and height and has a thickness of 8mm, while the virtual duplicate consists of 288 degrees of freedom and 605 tetrahedral elements.

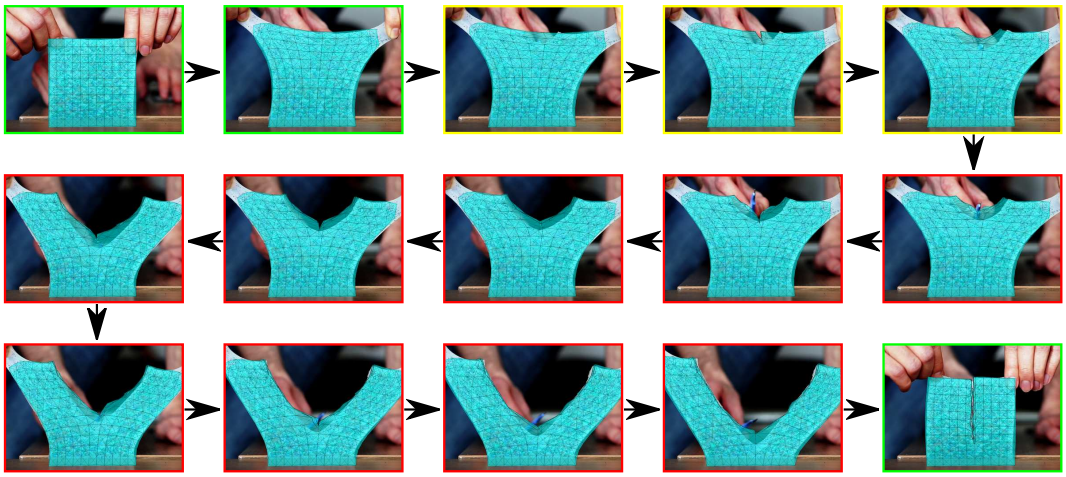


Figure 7.13: V-Example: Detection of the combination of strong deformations *(green boxes)*, cutting *(yellow boxes)* and tearing *(red boxes)*

Figure 7.14 illustrates the workflow of the proposed method on this example: From the input video stream we extract features $\mathbf{p}_{l,f}$ and track them over

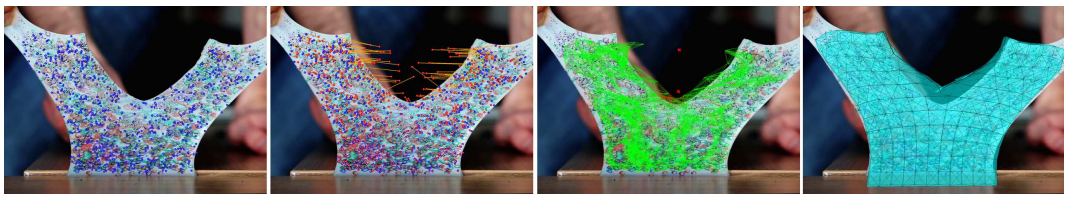


Figure 7.14: *From left to right*: raw image with image features *(blue dots)*; image features and their corresponding point on the deformable model *(red dots)* and the *orange line* showing the distance between real and virtual features; connectivity graph *(green lines)* between virtual features with the endpoints of the cutting path *(red crosses)*; resulting updated mesh overlayed on the image.

time - for the v-example, we extracted 531 features (figure 7.14 (left)). We deform the volumetric object by connecting the real features $\mathbf{p}_{l,f}$ to the virtual features $\mathbf{p}_{l,F}$ (figure 7.14 (left middle)). Using the neighborhood information $\mathcal{C}_F$ of the virtual features, the measure μ_{lm} can be calculated (figure 7.14 (right middle)). Outliers compared to the average of the measure, define a cut region in which we put the cut points, that are illustrated in figure 7.14 (right middle) as well. At every update, a cut line between an old and a new cut point is extended to a cut plane, which then introduces the topological change in the volumetric model (figure 7.14 (right)).

The v-example uses 6795 neighboring virtual features for the detection of an emerging separation. Due to the topological change, the neighborhood information $\mathcal{C}_F$ is updated until we have 6049 neighboring features in the final

frame of the video. A topological change such as a cut or a fracture yields more freedom of motion for the object. In this example, the volumetric model makes use of additional 149 vertexes to take account of the appearing separation. Despite the additional computational cost of to the added degrees of freedom the computation remains real-time, above 25 frames per second.

S-Example: The second example considers another cuboid, that is made of a silicone ten times stiffer than the silicone of the first example. Again, the object is deformed by elongation, but this time, we cut horizontally, with a complex cut geometry – similar to a part of the sinusoidal function, the s-example. After the cut is performed, the object is separated into two parts

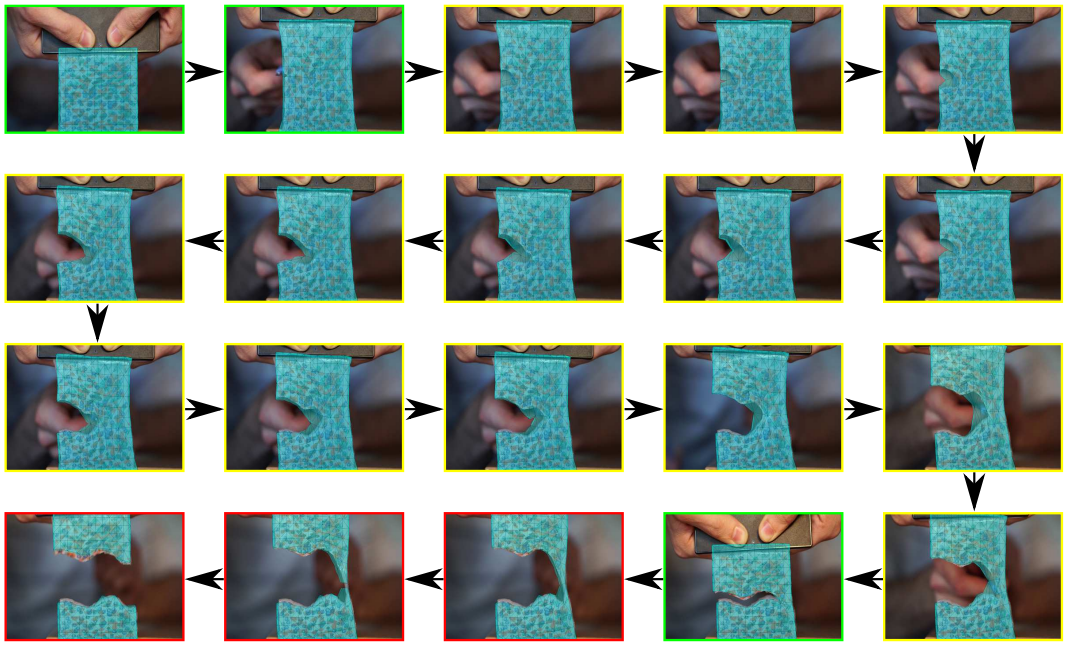


Figure 7.15: S-Example: Detection of the combination of strong deformations *(green boxes)*, cutting *(yellow boxes)* and tearing *(red boxes)*

by tearing. Between the three actions – deformation/elongation, the cut and the tearing – the object returns several times to a position with less stress, disturbing the detection of a cut. The objects dimensions are 8cm in width and height, but the object is thicker 1.2cm. Like in the first example, we apply a virtual model with 288 degrees of freedom and 605 tetrahedral elements. The video stream yields 693 features, that are initially connected by 9225 pairs of neighboring virtual features. In the course of the simulation the number of neighboring virtual features $\mathbf{p}_{l,F}$ reduces to 8023 while the number of vertices in the volumetric model increases to 315. Compared to the v-example, the increase is relatively low, as we used the snapping algorithm and not the re-meshing algorithm. Similar to the v-example, the impact on the computation

time is negligible, the real-time computation is not endangered.

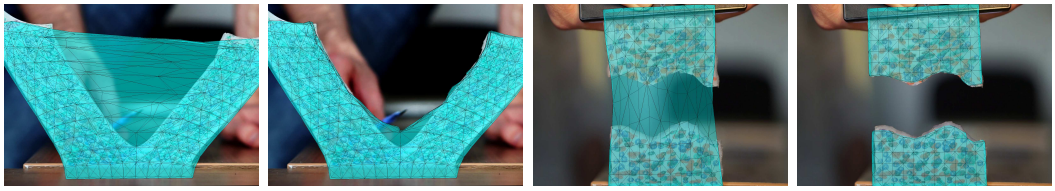


Figure 7.16: Comparison of an augmentation without and with our method the v-example *(left)* and the s-example *(right)*

Accuracy: Our examples yield very good results for the dice measure, in particular when compared to an augmented reality that would continue using a volumetric model not accounting for topological changes. Visual evaluation can be performed by looking at figure A.31. In the first test case – the vertical cut from the top – the dice measure in the final frame using our method is **0.951**, whereas an approach based on an uncut model only has a dice measure of **0.740**. The second example delivers similar results: our method has a dice measure of **0.964** in contrary to an uncut model that can only react incorrectly to the topological change yielding a dice measure of **0.819**.

Based on these promising results, we address the application in a surgical setting in the following.

7.8 Application in surgical interventions

Visualizing internal structures of organs in minimally invasive surgery is an important avenue for improving the perception of the surgeon, or for supporting planning and decision systems. Our physics-based non-rigid augmented reality is particularly interesting for that, as singularities caused by topological changes are detected and propagated to the pre-operative model and thus models can be used throughout the complete surgery.

In subsection 7.8.1 we assess our algorithm on a video of the deformation showing the cut of a porcine liver's lobe in minimal invasive surgery [79, 82] and we briefly discuss an example of the handling of sliding or jumping nodes. The chapter closes with the most recent results on augmenting internal structures [82], which has not yet been addressed in the literature to our knowledge. Note, that the scenarios of this section are using precut organs, refer to 7.7.1 for further details.

7.8.1 Validation on an *in-vivo* liver

We evaluated our approach on video clips involving a deformation of porcine liver lobes that have been cut before starting recording. The initial and the final frames of the first example are shown in figures 7.17 (left) and (right). In this example, the algorithm extracts 438 features $\mathbf{p}_{l,f}$ from the video stream (see figure 7.17 (left)), 32 are identified to be outliers in the advancement of the video (as it has been explained at the end of subsection 7.4.1). The features deform a volumetric mesh, using the spring energy $\Pi_S(\mathcal{P}_F^t(\mathcal{P}_V), \mathcal{P}_F^1)$. Figure 7.17 (middle left) illustrates the initial configuration of the vertices $\mathcal{P}_V$. We calculate the measure μ on 18503 pairs of features and lose 1478 of these pairs in the course of the simulation due to identified outliers – the initial neighborhood information $\mathcal{C}_F$ is displayed in figure 7.17 (middle right).

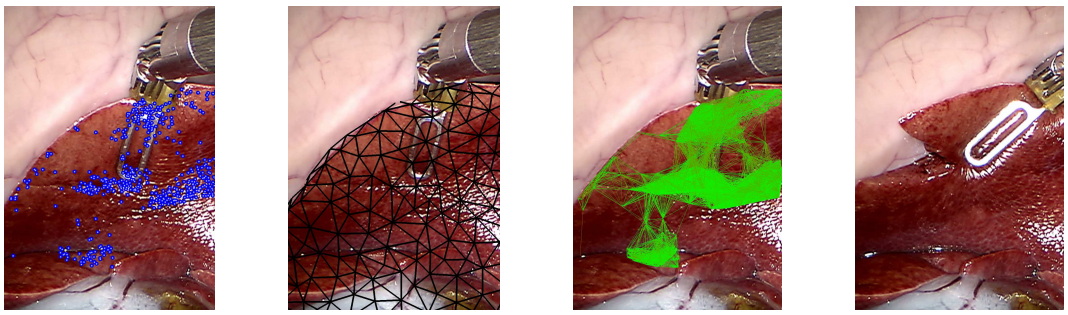


Figure 7.17: *From left to right*: Detected features $\mathbf{p}_{l,f}$ *(blue dots)*; (b) FEM mesh of the virtual organ *(black lines are the edges)*; Computed neighborhood $\mathcal{C}_F$ *(green lines)*; Organ manipulation after a cut

When applying our method, the measure μ_{ij} is calculated on the neighboring pairs of features in $\mathcal{C}_F$ and a pair of features is identified to be cut using a threshold $\tau = 7.0$. The resulting three-dimensional representation of the liver, as illustrated in figures 7.18 (right), is very similar to the actual organ shape.

A tumor has been inserted into the virtual representation of the organ. With no topology update, the augmented view of the tumor is distorted and misplaced, whereas with our cut detection and simulation, the tumor stays undeformed and correctly located below the cut.

To analyze our results, we compute the dice measure comparing the surfaces of the uncut pre-operative mesh and the cut mesh obtained with our method. The dice coefficient associated with our result is 0.963, whereas it was 0.906 for the uncut object, confirming the benefits of the proposed method.

Compared to the evaluation on silicone bands, retrieving and tracking the features in the laparoscopic video stream is cumbersome around the cut and

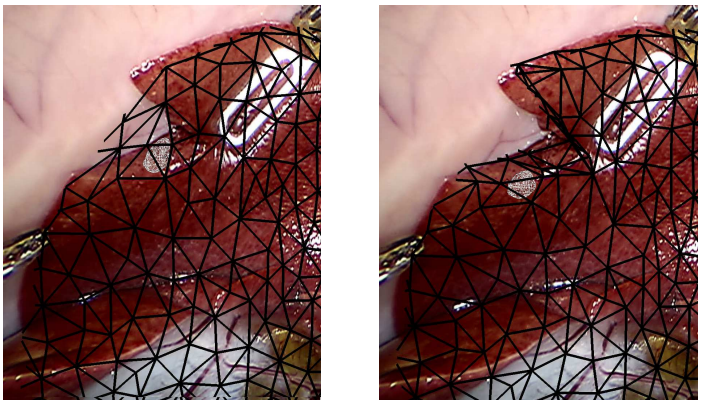


Figure 7.18: *Top*: Augmented reality on cut and deformed liver overlaid by the virtual organ and a tumor; with a normal elastic model *(left)* and with our method *(right)*

due to the reflection of the light of the camera. Moreover, the instrument on the left hand side of the image is going a little bit over the liver lobe and features tend to stick to it when the liver or the instrument move. Thus, before using the two dimensional features $\mathbf{p}^1_{l,f}$, we filter out jumping or sliding features using the idea discussed in subsection 7.4.1. A second example on an *in vivo* liver presented in figure 7.19, shows the neglected points on the left and the final augmentation on the right side.

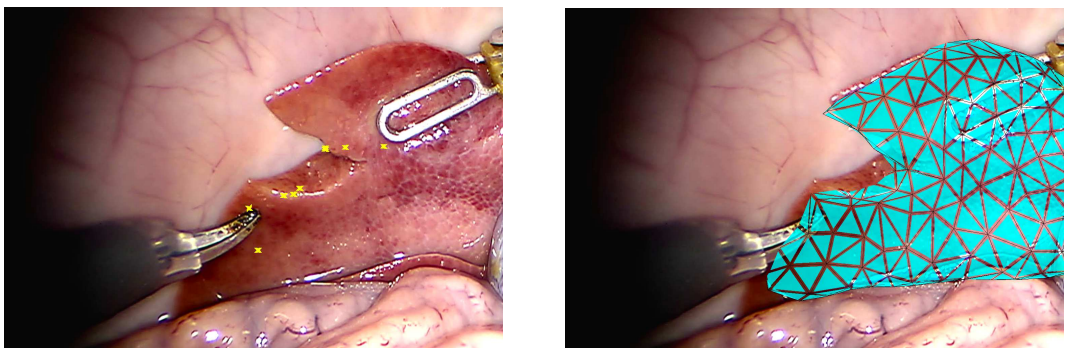


Figure 7.19: Augmentation of endoscopic images using a 3D liver model: *Left*: identified outliers of the detected features *(yellow crosses)*; *Right*: The 3D model topology is updated according to the cut of the liver lobe

While figure 7.18 displays the motivation for an augmentation with internal structures, it does not give information of the accuracy of the estimated position of internal structures. This topic is addressed in the following subsection, on *ex-vivo* kidneys.

7.8.2 Validation on *ex-vivo* kidneys with internal structures

The experiments presented in this subsection involve the cutting of an *ex-vivo* kidney. It provides a challenging evaluation of our method, as the internal structures of the kidney (the calyces – a part of the urinary system) have a complex geometry and cover a large part of the organ.

Ground truth

A trustworthy ground truth for the kidney and its internal structures is obtained as two CT scans were performed before and after the manipulation. To ensure a good visualization of the internal structures in the CT images and to avoid a loss of volume when the cut occurs, we filled the calyces with a gel that solidifies. The gel contains $BaSO_4$ microparticles, showing high contrast in CT images [25].

The kidneys are cut perpendicularly to the long axis in the middle of the parenchyma, incising a part of the calyx system. The cut is widened by stretching the kidney along its longitudinal axis. The organs and internal structures have been segmented using active contour techniques (Snakes) [46]. The resulting surface meshes, displayed in the following, have been smoothed using nearest neighbor smoothing algorithm.

In order to quantify the obtained results, we measure the (sampled) Hausdorff distance H between the surface and the internal structures of the kidney comparing the simulated solution – with and without cutting – with the goal positions provided by the final CT scan. For the surface feature points we use the Euclidean norm:

$$\|\mathbf{p}_{l,F}^{t}(\mathcal{P}_V^t) - \mathbf{p}_{l,F}^{1}\|_F = \|\mathbf{p}_{l,F}^{t}(\mathcal{P}_V^t) - \mathbf{p}_{l,F}^{1}\|_2 \tag{7.15}$$

Averages and maximal values for the measures are collected in table 7.2.

Our results

Kidney 1 measures 102x27x55mm, the internal structures 58x11x27mm. We extracted 225 feature points, that are connected using a radius $r_{\mathcal{P}_F} = 15$mm. We applied our algorithm with a threshold $\tau = 8$ to identify outliers when

			Average (mm)		*Max (mm)*	
Object	***Entity***	**Measure**	**Without cutting**	**With cutting**	**Without cutting**	**With cutting**
Kidney 1	$\mathcal{P}_F^t(\mathcal{P}_V^t)$	$\|\dots\|_F$	1.69	0.52	11.2	5.4
	$(\mathcal{P}_{\partial V}^t(\mathcal{P}_V^t), \mathcal{C}_{\partial V}^t)$	H	1.76	1.49	13.1	10.1
	$(\mathcal{P}_I^t(\mathcal{P}_V^t), \mathcal{C}_I^t)$	H	1.10	0.82	8.9	7.6
Kidney 2	$\mathcal{P}_F^t(\mathcal{P}_V^t)$	$\|\dots\|_F$	2.45	1.28	12.9	5.6
	$(\mathcal{P}_{\partial V}^t(\mathcal{P}_V^t), \mathcal{C}_{\partial V}^t)$	H	1.96	1.22	13.1	6.9
	$(\mathcal{P}_I^t(\mathcal{P}_V^t), \mathcal{C}_I^t)$	H	1.69	1.04	7.4	4.6

Table 7.2: Evaluation on *ex vivo* kidney data. Our method results in a higher accuracy than standard approaches without cutting.

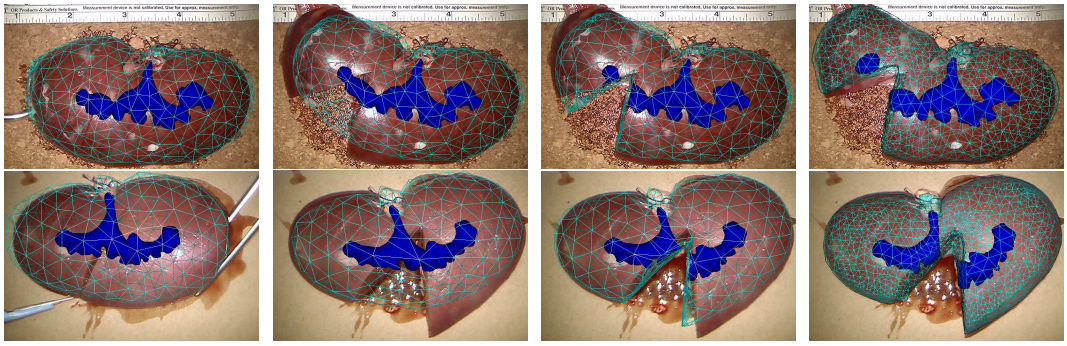


Figure 7.20: Augmented reality on cut and deformed kidney 1 *(top)* and 2 *(bottom)* overlaid by the virtual organ, the initial registration *(left)*, final registrations: uncut *(middle left)*, cut *(middle right)* and reference registration *(right)*.

comparing to the average of the measure, inserting a new cut point $\mathbf{p}_c$ as soon as we have $n = 8$ outliers and updating the neighborhood information using $r_{\mathbf{p}_c} = 7\text{mm}$.

Kidney 2 measures 96x27x56mm, the internal structures 52x12x32mm. We identified and tracked 219 feature points in the video stream and applied our method with $r_{\mathcal{P}_F} = 20\text{mm}$, $\tau = 10, n = 3, r_{\mathbf{p}_c} = 13\text{mm}$.

In both cases, the simulations were performed at minimally 25 fps, while on average the simulations run at 35 fps using a single CPU computer. The algorithms proposed in this work add less than 3% computational costs to a normal elastic FEM implementation, which is reflected in average by 0.15ms for the detection of a cut and 0.15-0.25ms for the update of the internal structures every time step.

For a visual comparison we refer to the overlaid images in figure A.32. More details can be retrieved from figure 7.21, which displays the values of the sampled Hausdorff distance overlaid on the meshs. Table 7.2 summarizes the

results by giving average and maximal values for the measures mentioned in section 7.8.2.

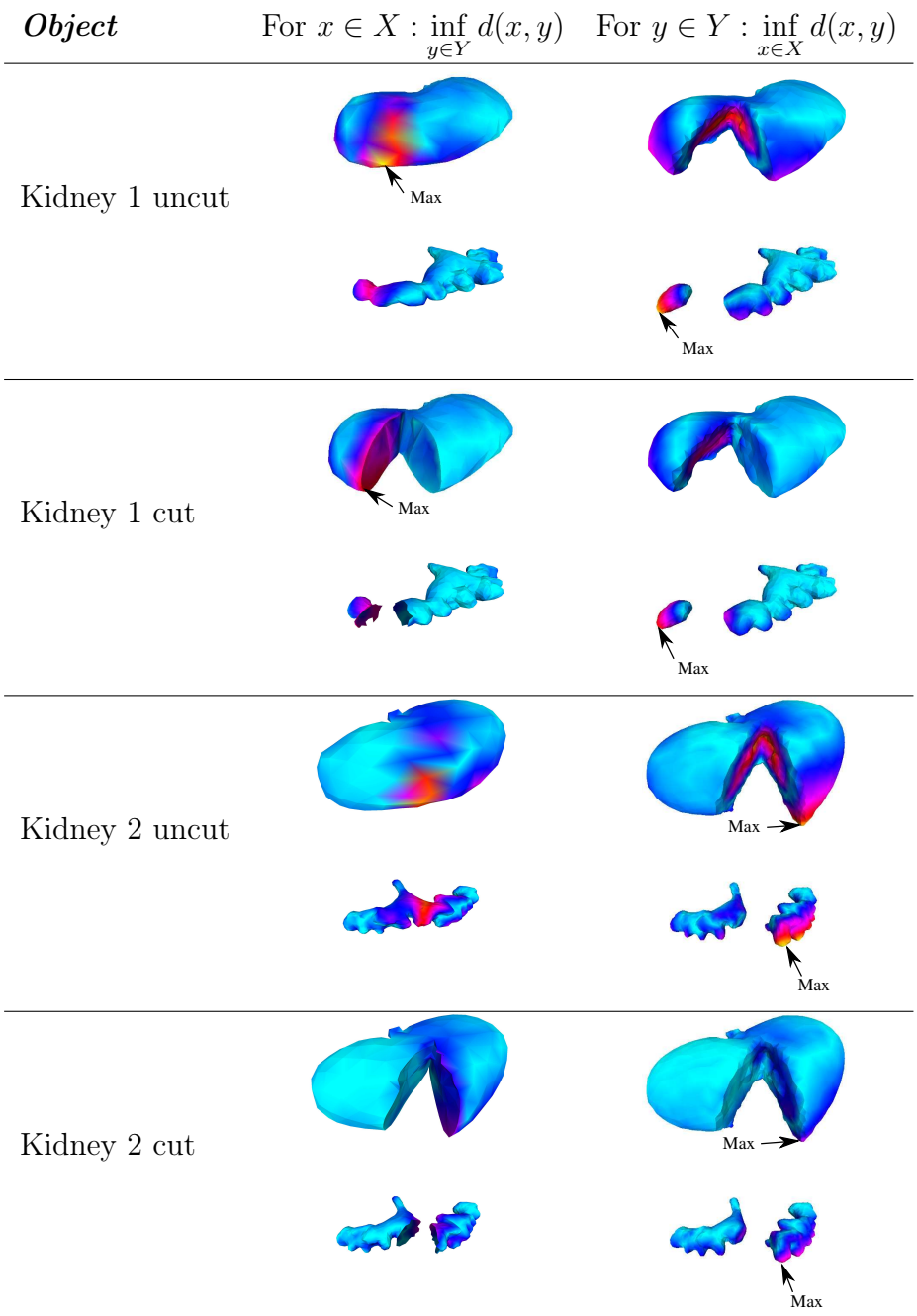


Figure 7.21: Visualization of the two parts found in the Hausdorff distance, surfaces X of the simulated and Y the reference solution; low values in *blue*, high values in *red/orange*.

Short discussion and additional information

In the presented results, our method shows a clear advantage over existing approaches that do not account for (surgical) cuts. Particularly interesting is the impact of the parameters, that have to be adapted for the different scenarios – for example the radius $r_{\mathbf{p}_c}$: for a small $r_{\mathbf{p}_c}$ (e.g. kidney 1), the cut advances in several steps (a progressing cut), while for a greater $r_{\mathbf{p}_c}$ (e.g. kidney 2), cut lines are longer, more robust, but less precise.

The results are highly dependent on the deformation. Subsections 7.7.1, 7.7.2 and previous work [78] shows the potentially positive impact of strongly deformed objects on the measure. Nevertheless, in the medical examples, we refrain from using strong deformations, to stay close to the medical workflow.

Part III

Conclusions and perspectives

Simulations for surgical interventions remain challenging as the surgical act includes many different physical behaviours while interactive simulation rates need to be provided. During surgery, organs may undergo large deformations and thus when a cut is performed, the resulting updated biomechanical model needs to maintain the stability and the consistency of the simulation.

In this PhD thesis we mainly considered the cutting and fracture of deformable objects in real-time. Furthermore we presented results for the detection of cutting and tearing from a monoscopic video stream with a visualization in the intraoperative view showing internal structures of an organ. Deformations are based on elasticity theory and computed using the finite element method.

Cuts based on the eXtended finite element method: First we presented results using the eXtended Finite Element method, which adds new nodes that are related to adapted shape functions. These shape functions allow for a separation inside elements without changing the original representation of the mesh. The convergence analysis reveals that the method outperforms the best possible solution of a conventional finite element method in terms of solution accuracy. Moreover, visually pleasing results are obtained when performing a complex cut in sinusoidal shape on a model of a liver. Our implementation is publicly available[1] and can be added to existing finite element implementations. However, due to the high computational costs of the integration algorithms required for partially cut elements, our implementation does not provide real-time performance. Also it currently only considers static problems.

Re-meshing tetrahedral elements: Therefore, in chapter 5 we presented another cutting approach based on a novel re-meshing technique. Remeshing algorithms replace cut tetrahedra by tetrahedra with nodes on the cut surface, therefore allowing for an opening of the mesh. Our re-meshing approach is based on a series of efficient topological changes that are combined with a moving of nodes to the cut surface. Our ideas have been implemented in the open source framework SOFA, which aims at real-time simulations, and has been combined with CGOGN, an open source framework that handles topological changes efficiently. Several theoretical and experimental examples illustrate the great advantage of our method compared to the literature. In particular our method allows for a limited increase of the number of added nodes during the cutting process. Also, thanks to our method, the condition number of the linear system of equations, arising from the finite element method, changes much less than with the approaches proposed in the litera-

[1]Code hosting platform bitbucket - https://bitbucket.org.
Download command: git clone https://chrijopa@bitbucket.org/chrijopa/xfem-in-cpp.git

ture and we our algorithm maintains real-time computation frame rates after the cuts.

One of the problems we encountered in our simulations was a jagged cut front (see figure 5.12) and potentially an incorrect following of the cut inside of the completely cut elements (see figure 5.13). Thus, a short term perspective is an improved placement of the middle points that control and could alleviate both behaviours. However, the improved placement of the nodes might come along with a deteriorated element quality, particularly for partially cut elements (see figure 5.14). Therefore, in a second, medium term step, we propose to remesh also the neighborhood of the cut elements in order to improve the condition number of the system.

Based on the positive results for tetrahedral elements with linear shape functions, the re-meshing approach has been extended to quadratic shape functions. Due to the improved convergence of quadratic tetrahedra, less nodes yield the same accuracy. Additionally, the smooth surfaces of these elements better fit the curved surface of an organ. To our knowledge, our re-meshing algorithm is the first to be applied to quadratic tetrahedra. Our theoretical analysis proves that our method inserts less nodes when comparing to the extention of re-meshing algorithms proposed in the literature for linear tetrahedra. Experimental results underline the theoretical observations and an example cutting a coarse mesh of a liver gives visually pleasing and physically reasonable results.

While our current implementation allows for curved object surfaces, the cut surfaces in our results are planar. Thus as a short term perspective, we propose to allow for curved cut surfaces, e.g. provided from the smoothed movement of a mouse. For linear shape functions on tetrahedral elements, the snapping of nodes to the cut surface helps to prevent ill-shaped elements and reduces the number of added nodes. We expect similar improvements in case of quadratic shape functions and we will, as a long term perspective, allow for the snapping of the nodes to the cut surface. This is more difficult to realize due to two reasons: First, in order to maintain the same boundary, the movement of a corner node needs to be followed by the movement of the edge nodes. Second, with the update of the current positions we also need to update undeformed positions to prevent introducing artificial stress. Thus, we considered the snapping of nodes on simpler elements with quadratic shape functions, briefly discussed in the following.

Fracture of quadratic shells: Given that quadratic elements allow for a better, more localized, computation of the stress, we proposed to use such elements in the context of fracture simulation. We applied this idea to triangular

shell elements and used the principal stress to calculate the fracture direction in the deformed configuration. The separation of the shells along the crack is performed along the element edges closest to the fracture direction, which are then snapped along the crack direction. This limits the introduction of new nodes, ensuring stability and real-time performance. Preliminary results on an L-shaped sheet show a first implementation of our approach.

For future work, we will improve the snapping method by splitting the fractured triangular elements into four elements, constructed from the six nodes of the quadratic triangle. This will lead to new elements and new nodes in the fractured region, making it easier to adapt the mesh to the actual fracture direction, while maintaining the original geometry of the shells. In order to limit the number of added nodes and to prevent non-conformal meshes, we propose to use T-junctions for the new nodes on existing mesh edges. Then we plan the application to the simulation of removing the epiretinal membrane in the eye. This work might also give interesting insights for an extension to the fracture of tetrahedral elements.

Augmented reality with topological changes: In the context of registration and augmented reality, physically-based methods have been proposed to overlay information coming from a three-dimensional pre-operative model, such as the location of tumors or vessels, onto the laparoscopic image of the organ. With these approaches, additional information coming from the three dimensional model, such as the location of tumors or vessels, can be overlaid on the view of the surgeon. While the literature contains very promising ideas and approaches, to our knowledge, our work has been the first to detect topological changes, that include cutting and tearing, on a highly elastic object from a monoscopic video stream. This work and our recent publication [82] present moreover, beyond the update of the three dimensional mechanical model, the update of the internal structures and thus important additional information could be provided to surgeons in the future. In order to incorporate detected topological changes into the model, we use the re-meshing approach briefly presented beforehand. The evaluation on highly elastic silicone bands, an in-vivo porcine liver and an ex-vivo porcine kidney show the improvement compared to the state of the art, that does not update the initial mechanical model.

While our results are very promising and we are the first addressing the topological changes detected from a monocular video stream in the literature, there are several shortcomings, that we would like to address: The camera observing the scene, the object itself are fixed and we can not handle occlusions. To formulate it differently, our algorithm can not recover or reidentify features. Thus, when a cut occurs and the cut surface is visible, the deformation of the volumetric object is still only based on the features of the initial surfaces,

which might not give enough information for the correct deformation of the volumetric object. This issue will be considered as a next step and compared to our results without recovering features. Additionally, the identification of the cutting instruments [111] would improve the detection of a separation. Then, we propose to address partial cuts in the direction of the camera and we expect the detection to improve due to the newly identified features on the cut surface. For the detection of partial cuts the depth of view has higher importance, as it determines the cut depth. Therefore, for the partial cuts we would use stereoscopic video streams and compare the results between to the results obtained from a monoscopic video stream.

We are confident that the methods presented in this PhD thesis will find their application in the simulation of surgical interventions or in the support of surgeons in their daily routine. While our work is to most part based on the conventional finite element method, specialized methods for real-time simulation of complex structures might gain relevance. We believe in and will investigate into the immersed boundary method, that uses a high resolution surfacic mesh embedded into a coarse mesh, that is used for the simulation.

Part IV

Appendix

1 Basics

1.1 Matrix manipulations

In our work we use the Euler sum, which indicates a sum over indices that appear twice in an equation

$$a_i b_i \coloneqq \sum_i a_i b_i \tag{A.16}$$

The upper and the lower limit of the sum usually is apparent from the context. Note, that as soon as the indice appear only once on the left hand side of the equation, then the Euler sum does not get applied

$$c_i = a_i b_i \neq \sum_i a_i b_i \tag{A.17}$$

With this notation, the multiplication of two matrices or tensors $\mathbf{A}$ and $\mathbf{B}$ writes as

$$\mathbf{AB} = (\mathbf{A}_{ik}\mathbf{B}_{kj})_{ij} \tag{A.18}$$

For matrices of same size, we define the inner product as

$$\mathbf{A} : \mathbf{B} \coloneqq \mathbf{A}_{ij}\mathbf{B}_{ij} \tag{A.19}$$

We obtain the following equalities:

$$\mathbf{A} : \mathbf{B} = \mathbf{A}_{ij}\mathbf{B}_{ij} = \mathbf{A}_{ji}\mathbf{B}_{ji} = \mathbf{B} : \mathbf{A} \tag{A.20}$$

$$\mathbf{AB} : \mathbf{C} = \mathbf{A}_{ik}\mathbf{B}_{kj}\mathbf{C}_{ij} = \mathbf{B}_{kj}\mathbf{A}^T_{ki}\mathbf{C}_{ij} = \mathbf{B} : \mathbf{A}^T\mathbf{C} \tag{A.21}$$

$$\mathbf{A} : \mathbf{BC} = \mathbf{A}_{ij}\mathbf{B}_{ik}\mathbf{C}_{kj} = \mathbf{A}_{ij}\mathbf{C}^T_{jk}\mathbf{B}_{ik} = \mathbf{AC}^T : \mathbf{B} \tag{A.22}$$

For symmetric matrices $\mathbf{A} = \mathbf{A}^T$, the transose can be forwarded to the other matrix

$$\mathbf{A} : \mathbf{B} = \mathbf{A}^T : \mathbf{B} = \mathbf{A}_{ij}\mathbf{B}_{ji} = \mathbf{A}_{ij}\mathbf{B}^T_{ij} = \mathbf{A} : \mathbf{B}^T \tag{A.23}$$

and if we use additionally a skewsymmetric matrix $\mathbf{B} = -\mathbf{B}^T$, the inner product is zero

$$\mathbf{B} = -\mathbf{B}^T \Longrightarrow \mathbf{A} : \mathbf{B} = \mathbf{A} : \mathbf{B}^T = -\mathbf{A} : \mathbf{B} \Longrightarrow \mathbf{A} : \mathbf{B} = 0 \tag{A.24}$$

When we define the outer product of two vectors $\mathbf{b} \in \mathbb{R}^{m\times 1}, \mathbf{c} \in \mathbb{R}^{n\times 1}$ as

$$\mathbf{bc} = (\mathbf{b}_i\mathbf{c}_j)_{ij} \in \mathbb{R}^{m\times n} \tag{A.25}$$

we can combine inner product of the matrices with the outer vector product:

$$\mathbf{A} : \mathbf{bc} = \mathbf{A}_{ij}\mathbf{b}_i\mathbf{c}_j = \mathbf{b}_i\mathbf{A}_{ij}\mathbf{c}_j = \mathbf{b} \cdot \mathbf{Ac} \tag{A.26}$$

1.2 Derivatives and integral equations

If we consider a smooth vector field $\boldsymbol{v}$ and a smooth tensor field $\boldsymbol{A}$, then we have

$$\nabla_{\mathbf{x}}(\boldsymbol{A}^T\boldsymbol{v}) = (\nabla_{\mathbf{x}}\boldsymbol{A})\boldsymbol{v} + \nabla(\boldsymbol{A}) : \nabla_{\mathbf{x}}\boldsymbol{v} \tag{A.27}$$

Considering integral equations, we state the divergence theorem: for a compact region Ω and a piecewise smooth boundary $\partial\Omega$ it gives

$$\int_{\partial\Omega} \boldsymbol{v}\boldsymbol{n}da = \int_{\Omega} \nabla_{\mathbf{x}} \cdot \boldsymbol{v}dv \tag{A.28}$$

$$\int_{\partial\Omega} \boldsymbol{A}\boldsymbol{n}da = \int_{\Omega} \nabla_{\mathbf{x}} \cdot \boldsymbol{A}dv \tag{A.29}$$

and the integration by substitution: for a continuous vector field $\boldsymbol{v}$ and an open set Ω, linked by a bijective function $\chi \in C^1, \chi^{-1} \in C^1$, we have

$$\int_{\chi(\Omega)} \boldsymbol{v}(\mathbf{x})dv = \int_{\Omega} \boldsymbol{v}(\chi(\mathbf{X}))|\det(\nabla_{\mathbf{X}}\chi(\mathbf{X}))|dV \tag{A.30}$$

2 Symmetry of the Cauchy stress tensor

Based on the angular momentum and similarly to the derivation of Cauchy's equation of motion, a second differential equation can be obtained (for further information please refer to [110], page 52)

$$\nabla_{\mathbf{x}} \cdot \left(\boldsymbol{x} \times \boldsymbol{\sigma}^T\right) + \boldsymbol{x} \times \boldsymbol{b} = \rho \left(\boldsymbol{x} \times \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t}\right) \tag{A.31}$$

where

$$\boldsymbol{x} \times \boldsymbol{\sigma}^T = \times_{ijk} x_j \sigma_{lk} \tag{A.32}$$

with the third order permutation tensor $\times_{ijk}$ that is zero for all entries other than

$$\times_{123} = \times_{231} = \times_{312} = -\times_{132} = -\times_{321} = -\times_{213} = 1 \tag{A.33}$$

Using the product rule, we can write

$$\left[\nabla_{\mathbf{x}} \cdot \left(\boldsymbol{x} \times \boldsymbol{\sigma}^T\right)\right]_i = \frac{\partial}{\partial x_l}(\times_{ijk} x_j \sigma_{lk}) = \left[\boldsymbol{x} \times \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}^T\right]_i + \times_{ijk} \delta_{jl} \sigma_{lk} \tag{A.34}$$

Then the linearity of the cross-product operator and a rearranging the differential equation above gives

$$0 = \left[\boldsymbol{x} \times \underbrace{\left(\rho \frac{\partial \boldsymbol{v}}{\partial t} - \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}^T - \boldsymbol{b}\right)}_{\overset{(2.32)}{=} 0}\right]_i = \times_{ijk} \delta_{jl} \sigma_{lk} = \times_{ijk} \sigma_{jk} \tag{A.35}$$

As a result we obtain the symmetry of the CST:

$$\sigma_{jk} = \sigma_{kj} \tag{A.36}$$

3 Shape functions and spatial integration

In this section we list the shape functions and the integration points for different element shapes and polynomial degrees.

Element type				$\Omega_\triangle$	Shape functions	$\boldsymbol{\xi}^g$	w
Triangle	lagrange	1		$\{\boldsymbol{\xi} \in [0,1]^2 \mid \sum_i \xi_i \leq 1\}$	$N_{\triangle,0}(\boldsymbol{\xi}) = 1 - \xi_0 - \xi_1$ $N_{\triangle,1}(\boldsymbol{\xi}) = \xi_0$ $N_{\triangle,2}(\boldsymbol{\xi}) = \xi_1$	$(\frac{1}{2}, \frac{1}{2})$	$\frac{1}{2}$
Triangle	lagrange	2		$\{\boldsymbol{\xi} \in [0,1]^2 \mid \sum_i \xi_i \leq 1\}$	$N_{\triangle,0}(\boldsymbol{\xi}) = \lambda(2\lambda - 1)$ $N_{\triangle,1}(\boldsymbol{\xi}) = \xi_0(2\xi_1 - 1)$ $N_{\triangle,2}(\boldsymbol{\xi}) = \xi_1(2\xi_2 - 1)$ $N_{\triangle,3}(\boldsymbol{\xi}) = 4\xi_0\lambda$ $N_{\triangle,4}(\boldsymbol{\xi}) = 4\xi_0\xi_1$ $N_{\triangle,5}(\boldsymbol{\xi}) = 4\xi_1\lambda$ where $\lambda = 1 - \sum_i \xi_i$	$(\frac{1}{2}, \frac{1}{2})$ $(0, \frac{1}{2})$ $(\frac{1}{2}, 0)$	$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$
Tetrahedron	lagrange	1		$\{\boldsymbol{\xi} \in [0,1]^3 \mid \sum_i \xi_i \leq 1\}$	$N_{\triangle,0}(\boldsymbol{\xi}) = 1 - \xi_0 - \xi_1 - \xi_2$ $N_{\triangle,1}(\boldsymbol{\xi}) = \xi_0$ $N_{\triangle,2}(\boldsymbol{\xi}) = \xi_1$ $N_{\triangle,3}(\boldsymbol{\xi}) = \xi_2$	$(\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	$\frac{1}{6}$
Tetrahedron	lagrange	2		$\{\boldsymbol{\xi} \in [0,1]^3 \mid \sum_i \xi_i \leq 1\}$	$N_{\triangle,0}(\boldsymbol{\xi}) = \lambda(2\lambda - 1)$ $N_{\triangle,1}(\boldsymbol{\xi}) = \xi_0(2\xi_1 - 1)$ $N_{\triangle,2}(\boldsymbol{\xi}) = \xi_1(2\xi_2 - 1)$ $N_{\triangle,3}(\boldsymbol{\xi}) = \xi_2(2\xi_3 - 1)$ $N_{\triangle,4}(\boldsymbol{\xi}) = 4\xi_0\lambda$ $N_{\triangle,5}(\boldsymbol{\xi}) = 4\xi_0\xi_2$ $N_{\triangle,6}(\boldsymbol{\xi}) = 4\xi_1\lambda$ $N_{\triangle,7}(\boldsymbol{\xi}) = 4\xi_2\lambda$ $N_{\triangle,8}(\boldsymbol{\xi}) = 4\xi_0\xi_2$ $N_{\triangle,9}(\boldsymbol{\xi}) = 4\xi_1\xi_2$ where $\lambda = 1 - \sum_i \xi_i$	(α, β, β) (β, α, β) (β, β, α) (β, β, β) where $\alpha = 0.58541020$ and $\beta = 0.13819660$	$\frac{1}{24}$ $\frac{1}{24}$ $\frac{1}{24}$ $\frac{1}{24}$

Table A.3: Different element types with reference domain, shape functions and Gauss integration points and weights

4 Derivation of the stiffness matrix in linear elasticity

To simplify the notations of the following derivations, we use

$$\mathbf{u}_{c_\mathrm{e}(l)} =: (\mathrm{u}_{li})_i = (\mathrm{u}_{l1}, \mathrm{u}_{l2}, \mathrm{u}_{l3}) \tag{A.37}$$

$$\boldsymbol{U}_\mathrm{e} =: (u_i)_i = (u_1, u_2, u_3) \tag{A.38}$$

$$\nabla_\mathbf{X} \boldsymbol{U}_\mathrm{e} = (\nabla_\mathrm{X} u_{ij})_{ij} = \frac{\partial \boldsymbol{U}_\mathrm{e}}{\partial \mathbf{X}} = \left(\frac{\partial u_i}{\partial \mathrm{X}_j} \right)_{ij} \tag{A.39}$$

$$N_l \coloneqq N_{\mathrm{e},l}, \quad N_{l,j} \coloneqq \frac{\partial N_l}{\partial \mathrm{X}_j} \tag{A.40}$$

$$\boldsymbol{\varepsilon}_{gl,l} =: (\varepsilon_{j_1 j_2})_{j_1 j_2} \tag{A.41}$$

With these definitions, we can write

$$\frac{\partial \nabla_\mathrm{X} u_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} = \frac{\partial \left(u_{l_3 i_1} N_{l_3, i_3} \right)}{\partial \mathrm{u}_{l_2 i_2}} = \delta_{i_1 i_2} \delta_{l_2 l_3} N_{l_3, i_3} = \delta_{i_1 i_2} N_{l_2, i_3} \tag{A.42}$$

Insertion inside of the linearized Green-Lagrange strain tensor (2.75) gives

$$\frac{\partial \varepsilon_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} = \frac{1}{2}\frac{\partial}{\partial \mathrm{u}_{l_2 i_2}}\left(\nabla_{\mathrm{X}} u_{i_1 i_3} + \nabla_{\mathrm{X}} u_{i_3 i_1}\right) = \frac{1}{2}\left(\delta_{i_1 i_2} N_{l_2,i_3} + \delta_{i_2 i_3} N_{l_2,i_1}\right) \tag{A.43}$$

According to [70], chapter 6.1.6, the stress increments associated with different conjugate variables coincide in the natural configuration for linear elastic materials. Thus $\boldsymbol{P} = \boldsymbol{\sigma}$ and with the constitutive equation one can obtain

$$\frac{\partial \boldsymbol{P}_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} = \frac{\partial \sigma_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} = \lambda \delta_{i_1 i_3} \frac{\partial \varepsilon_{i_4 i_4}}{\partial \mathrm{u}_{l_2 i_2}} + 2\mu \frac{\partial \varepsilon_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} \tag{A.44}$$

$$= \frac{\lambda \delta_{i_1 i_3}}{2}\left(\delta_{i_2 i_4} N_{l_2,i_4} + \delta_{i_2 i_4} N_{l_2,i_4}\right) + \mu\left(\delta_{i_1 i_2} N_{l_2,i_3} + \delta_{i_2 i_3} N_{l_2,i_1}\right) \tag{A.45}$$

$$= \lambda \delta_{i_1 i_3} N_{l_2,i_2} + \mu \delta_{i_1 i_2} N_{l_2,i_3} + \mu \delta_{i_2 i_3} N_{l_2,i_1} \tag{A.46}$$

and finally, the internal force density can be calculated

$$\frac{\partial f^{int}_{\mathrm{e},l_1 i_1}}{\partial \mathrm{u}_{l_2 i_2}} = \frac{\partial (\boldsymbol{P}_{i_1 i_3} N_{l_1,i_3})}{\partial \mathrm{u}_{l_2 i_2}} = \frac{\partial \boldsymbol{P}_{i_1 i_3}}{\partial \mathrm{u}_{l_2 i_2}} N_{l_1,i_3} \tag{A.47}$$

$$= (\lambda \delta_{i_1 i_3} N_{l_2,i_2} + \mu \delta_{i_1 i_2} N_{l_2,i_3} + \mu \delta_{i_2 i_3} N_{l_2,i_1}) N_{l_1,i_3} \tag{A.48}$$

$$= \lambda \delta_{i_1 i_3} N_{l_2,i_2} N_{l_1,i_3} + \mu \delta_{i_1 i_2} N_{l_2,i_3} N_{l_1,i_3} + \mu \delta_{i_2 i_3} N_{l_2,i_1} N_{l_1,i_3} \tag{A.49}$$

$$= \lambda N_{l_2,i_2} N_{l_1,i_1} + \mu \delta_{i_1 i_2} N_{l_2,i_3} N_{l_1,i_3} + \mu N_{l_2,i_1} N_{l_1,i_2} \tag{A.50}$$

To resume, we can write the elementary stiffness matrix as

$$\mathbb{K}_{\mathrm{e},3l_1+i_1\ 3l_2+i_2} = \int_{\Omega_\mathrm{e}} \frac{\partial f^{int}_{\mathrm{e},l_1 i_1}}{\partial \mathrm{u}_{l_2 i_2}}\, dV \tag{A.51}$$

that can be assembled to obtain the global stiffness matrix

$$\mathbb{K}_{3k_1+i_1\ 3k_2+i_2} = \delta_{k_1 c_\mathrm{e}(l_1)} \delta_{k_2 c_\mathrm{e}(l_2)} \mathbb{K}_{\mathrm{e},3l_1+i_1\ 3l_2+i_2} \tag{A.52}$$

Note, that the element matrices $\mathbb{K}_\mathrm{e}$ are symmetric and as a result, after the assembling process described in the last equation, we obtain a symmetric global stiffness matrix $\mathbb{K}$.

A similar derivation is not possible for the non-linear St. Venant Kirchhoff material since this material behaviour uses the non-linear Green-lagrange strain tensor: Calculating a derivative $\frac{\partial}{\partial u_{li}}$ yields a linear function in u_{li} and thus the internal force $\mathbb{f}^{int}(\mathrm{u})$ can not replaced by a matrix vector product $\mathbb{K}\mathrm{u}$.

Part V

Brief summary in French

1 Introduction et motivation

Dans les dernières déciennes, la chirurgie mini-invasive remplace de plus en plus les opérations ouvertes, pour atténuer les effets de l'acte chirugical sur le patient.

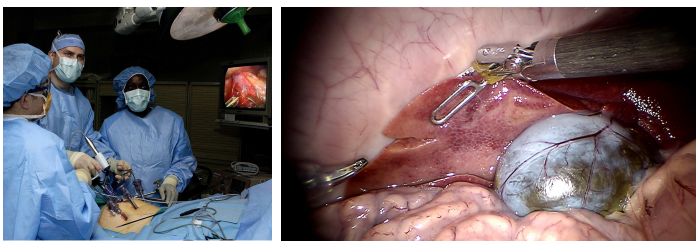


Figure A.22: Laparoscopie: Configuration *(gauche)*, vue peroperatoire *(droite)*

La laparoscopie est la technique utilisée pour les appareils digestif et pelvien en chirgurgie en chirurgie mini-invasive, elle a éte appliquée la première fois en 1987. Cette technique récente utilise l'insufflation de gaz carbonique dans la cavité abdominale pour créer un espace de travail autour des organes. Des trocarts sont placés dans des petites ouvertures dans la paroi abdominale, permettant d'insérer une camera et des instruments chirurgicaux allongés. L'invervention chirurgicale est faite avec une vue indirecte par camera laparoscopique. Évitant les operations ouvertes avec de grandes incisions, cette procédure reduit la morbidité post-opératoire, la douleur, la taille des cicatrices, et donc permet de diminuer le temps de récupération [121].

Ces effets positifs pour les patients entraînent des défis pour les chirurgiens. En effet leur visibilité se trouve reduit et les chirurgiens doivent appliquer des informations pré-opératoires, récupérées sur des imageries (TDM, IRM) à la situation peroperatoire qui inclut des mouvements et déplacements des tissus ciblés. Ainsi, les chirurgiens ont des difficultés à localiser des tumeurs, et la chirurgie mini-invasive prend plus de temps [121].

Ces nouveaux défis entraînent des erreurs de manipulation et un risque augmenté de dommages tissulaire. Aux États-unis, des erreurs similaires se produisent dans 3 pourcent des hospitalisations, avec des coûts entre 17 et 29 trillions de dollars américains [49]. Cependant, les chirurgiens bénéficient rarement d'un eintraînement intensif. Similairement au développement de l'industrie du vol dans le passé, des simulations et une assistance informatique pourraient être utilisées pour la formation, la planification, le soutien et l'orientation des chirurgiens. Pour la neurochirurgie et la chirurgie orthopédique, les méthodes de chirurgie assistées par ordinateur sont déjà bien connues et donnent un meilleur résultat pour le patient [35, 130]. Avec les systèmes de réalité augmentée, une image virtuelle est superposée à la vue de l'intervention chirurgicale, pour un meilleur guidage chirurgical [67]. Les algorithmes de guidage des chirurgiens reposent sur des systèmes d'imagerie (préopératoires) et des systèmes de capteurs intra-opératoires, par exemple la camera laparoscopique.

La formation par simulation ne peut pas s'appuyer sur une information per opératoire et doit fournir des modèles réalistes pour les manipulations effectuées lors d'une intervention chirurgicale. Seuls quelques systèmes proposent la simulation d'une procédure complète. L'un des systèmes les plus avancés est actuellement développé pour la chirurgie de la cataracte et utilise une approche basée sur la physique pour représenter les manipulations dans les simulations.

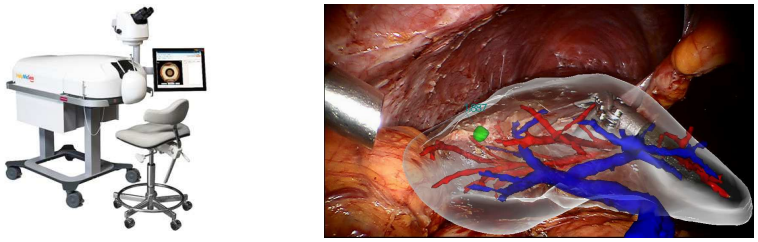


Figure A.23: Assistance informatique pour la formation de la chirurgie de la cataracte sur l'œil [109] *(gauche)* et à l'intérieur de la cavité abdominale montrant les structures internes du foie comme réalité augmentée [55] *(droite)*

Les approches basées sur la physique sont très prometteuses en termes de

réalisme, mais difficiles car de nombreux phénomènes différents doivent être traités. En chirurgie laparoscopique, les manipulations d'organes entraînent de grandes déformations, coupures et cautérisations. En raison des surfaces incurvées des organes, les méthodes basées sur la physique doivent fournir des modèles géométriques précis, de préférence avec des bordures courbes, potentiellement soumis aux grandes déformations [118].

De plus, la capacité de simuler des coupures ou des déchirures de tissus mous est essentielle pour l'application dans le contexte chirurgical. Pour la formation et l'application per-operatoire, les solutions proposées doivent être stables, simulée en temps réel et avec une grande précision. La combinaison de tous les points mentionnés dans ce paragraphe est particulièrement difficile et fait de ce sujet un domaine de recherche actif, la plupart des solutions aboutissent à des compromis car une plus grande précision met directement en danger l'aspect temps réel des simulations.

La découpe virtuelle d'objets déformables pourrait avoir applications dans d'autres domaines et donc les techniques proposées pourraient être utilisé dans d'autres contextes.

Cette thèse porte sur la propagation des changements topologiques (découpe, cassure) dans un modèle mécanique. Nous visons à préserver l'aspect de temps réel pendant toute la phase de la simulation. De nombreuses méthodes physiques pour la simulation des déformations existent dans la littérature, notre travail est basé sur la méthode des éléments finis. Les détections de changements topologiques basées sur des informations provenant d'une image sont discutées. Nous abordons les questions relatives à la stabilité et à l'aspect temps réel de la simulation.

Nous présentons les contributions suivantes:

−Un algorithme robuste et efficace pour couper des tétraèdres utilisant des polynomômes de premier degré, basé sur le mouvement des sommets et des techniques de remaillage

−Une extension de cet algorithme aux tétraèdres utilisant des polynomômes de second degré pour obtenir des surfaces lisses et courbes

−Une application aux interventions chirurgicales montrant des changements topologiques superposés à un organe qui est coupé, déchiré ou fracturé

−Visualisation des structures internes tout en coupant les organes

Ce manuscrit est organisé autour des publications et des travaux en cours de ce doctorat de trois années. Le but de ce document est de résumer le travail brièvement, plus de détails sont présentés dans la version anglaise de la thèse de doctorat.

Section 2 présente les contributions aux algorithmes de coupe: une implémentation de la méthode des éléments finis étendus et un nouvel algorithme de remaillage appliqué aux tétraèdres bassée sur des polynomômes du premier et second degré.

Section 3 discute les résultats lors de la détection de coupures à partir d'un flux vidéo monoscopique et en l'appliquant à un modèle tridimensionnel virtuel avec des structures internes.

2 Changements topologiques

De nombreuses méthodes physiques pour la simulation de déformations existent dans la littérature, par exemple les méthodes de mass-ressort, les méthodes sans maillage et la méthode des éléments finis. Dans leur forme classique, ces méthodes ne peuvent pas traiter les changements topologiques (comme la découpe) et ainsi des améliorations ou des adaptations sont nécessaires. L'objet représenté par la méthode classique des éléments finis, par exemple, ne permet que les séparations aux sommets de maillage et non à l'intérieur des éléments. Ainsi, soit quelques sommets doivent être déplacés vers la surface de la découpe, soit le maillage doit être adapté en insérant de nouveaux sommets à la surface de la découpe (remaillage), soit on doit appliquer une méthode adaptée d'éléments finis.

La littérature sur les coupes virtuelles montre que les éléments tétraédriques sont utilisés depuis l'origine des simulations de découpe [69, 14, 24, 68, 63, 107, 43, 123] alors que les coupes virtuelles sur éléments hexaédriques ont été introduites plus récemment [129, 26, 126, 101]. Pour les mailles polyédriques, il n'y a eu que quelques ouvrages [124, 57]. Et les méthodes sans maillage ont actuellement une importance croissante [65, 83, 88, 104], en particulier combinées avec d'autres approches [73, 74, 75, 92, 93]. Ce sont quelques repères essentiels, d'autres détails sur les algorithmes de coupe jusqu'à 2014 sont présentés par Wu et al [128].

Dans cette section, nous présentons nos contributions à ce champ de recherche: nous présentons d'abord nos résultats concernant la méthode des éléments finis étendus, qui permet des coupes à haute résolution dans des grilles grossières,

voir la sous-section 2.1. Ensuite nous expliquons un nouvel algorithme de remaillage est expliqué utilisant d'abord des polynômes du premier degré (voir la sous-section 2.2) puis du second (voir la sous-section 2.3).

2.1 Espace de fonctions adaptées - Méthode des éléments finis étendus

La méthode des éléments finis étendus (X-FEM) est particulièrement intéressante, car elle fournit des outils pour manipuler des éléments qui sont complètement et partiellement coupés avec des géométries complexes. Notez que cette méthode ne déplace pas ou n'introduit pas de sommets sur la surface de la coupe, elle étend les capacités de la méthode classique des éléments finis. Dans cette section, nous discuterons brièvement de notre travail sur ce sujet, alors que d'autres détails peuvent être trouvés dans [77]. Les contributions principales de notre travail [77] sont:

–Application du X-FEM à la simulation de la coupe dans les tissus mous

–Possibilité de simuler des éléments complètement et partiellement coupés

–Une évaluation qui démontre la précision supérieure par rapport aux FEM conventionnelle

–Une implémentation X-FEM open source [2]

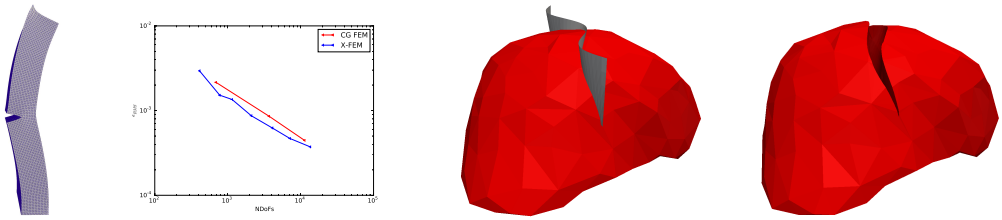


Figure A.24: Analyse de convergence montrant l 'erreur quadratique moyenne du X-FEM par rapport à la solution FEM conventionnelle sur un exemple de coupe d 'un faisceau *(left)*; Un foie avec 888 éléments tétraédriques linéaires est coupé par une lame sinusoïdale: non déformé avec la surface de coupe *(middle right)*, et la déformation après la coupe *(right)*

L'idée de X-FEM peut être énoncée ainsi: Les éléments coupés avec la surface de coupe S sont identifiés et classés entre complètement ou partiellement

[2]Code hosting platform bitbucket - https://bitbucket.org.
Download command: git clone https://chrijopa@bitbucket.org/chrijopa/xfem-in-cpp.git

coupés. Pour les sommets des éléments coupé de nouveaux degrés de liberté sont ajoutés, qui permettent des séparations à l'intérieur d'un élément.

Nous présentons les résultats sur un exemple de faisceau, coupé à mi-chemin , voir figure A.24 (à gauche). Les solutions utilisant la méthode des éléments finis classiques sont parfaitement alignées sur la surface de coupe et peuvent être considérées comme la meilleure approximation possible de la méthode sous cette forme. La méthode des éléments finis étendus utilise un maillage créé au hasard et n'a donc aucune adaptation à la coupe. La Figure A.24 (à gauche), montre que Même dans ce scénario désavantageux pour le X-FEM, la méthode donne de meilleurs résultats. En outre, nous avons appliqué la méthode à un exemple médical, en coupant une maille volumétrique d'un foie avec une coupe complexe , voir figure A.24 (à droite).

2.2 Un nouveau algorithme de remaillage

Le travail décrit dans cette section a été publié sous la forme d'une conférence [81] et d'un journal [80]. Ces publications servent d'orientation principale. Notre algorithme proposé peut être séparé en deux étapes: La première étape de notre méthode est l'échantillonnage ou la détection de la surface de séparation au niveau des arêtes de la maille FEM. Pour chaque arête e du maillage, nous calculons ou estimons un point de coupe ou de rupture. Si le point de rupture est proche d'un sommet existant du maillage, le sommet est déplacé vers la coupe et la surface de coupe S passe par ce sommet du maillage. La deuxième étape consiste en un remaillage du modèle, décrit plus en détail dans ce qui suit.

L'algorithme remaillage que nous présentons est une extension du [18] et gère des coupes partielles d'éléments. Pour simplifier la compréhension de cet algorithme, veuillez considérer l'exemple bidimensionnel de la figure A.25 (parti supérieur).

Pour le cas tridimensionnel, une arête qui traverse la surface de séparation indique que les volumes incidents sont coupés par la surface S. Pour tenir compte de cela, nous introduisons des sommets dans les tétraèdres incidents avec les arêtes croisées avec la surface de la coupe , voir figure A.25 (partie inférieur gauche). Les sommets nouvellement introduits, notés $\mathbf{p}_k$, sont positionnés sur la surface S. Les prochaines étapes procèdent à l'insertion des arêtes entre le $\mathbf{p}_k$ (voir la figure A.25 (inférieur au milieu)) et finalement l'insertion des triangles τ_j reliant les sommets nouvellement introduits (voir figure A.25 (partie inférieur droite)). Enfin, la séparation des triangles est

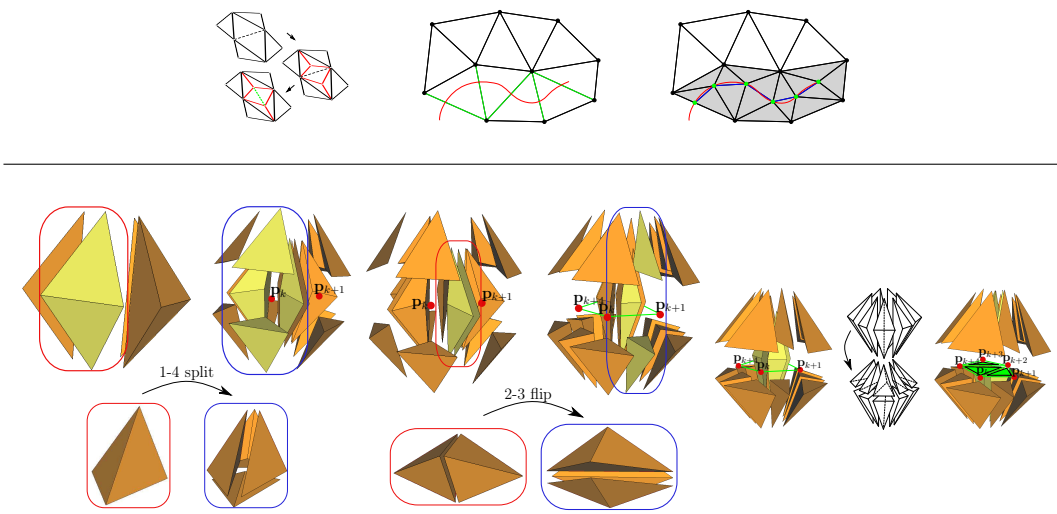


Figure A.25: *Partie supérieur*: algorithme de remaillage en deux dimensions; *Partie inférieur*: Algorithme de remaillage en trois dimensions autour d'un arêt

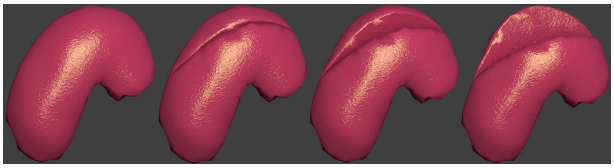
réalisée à l'aide d'une topologie basée sur des cartes combinatoires [51]. Notez que la limite doit être traitée séparément, ce qui est décrit dans [80].

Nous avons mis en œuvre différents exemples, basés sur la méthode proposée, en utilisant le framework open source SOFA [29]. Une analyse théorique révèle que notre méthode surpasse les idées rémanentes actuelles: elle introduit deux fois moins de sommets et deux fois moins d'éléments et est donc meilleure pour l'application en temps réel. De plus, la qualité numérique de notre système se détériore beaucoup moins qu'avec la méthode de Bielser et al. [50, 13, 14]. Notre approche permet de séparer un élément en six parties après avoir été coupé. Une amélioration potentielle serait un meilleur placement des sommets lors de la première étape.

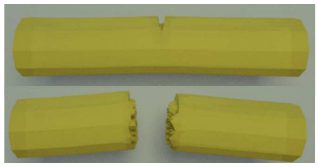
2.3 Expansion aux éléments des polynômes du second degré

En se basant sur les résultats encourageants de notre nouvel algorithme de remaillage pour les tétraèdres des polynômes du premier degré présentés dans la section 2.2, cette section applique l'approche de remaillage en utilisant des polynômes du second degré. Les tétraèdres des polynômes du second degré ont des sommets supplémentaires sur leurs arets qui permettent de représenter le bord lisse d'un élément , voir la figure A.27 (à gauche), normalement visualisée à l'aide d'une tesselation (voir figure A.27 milieu)). La position du point milieu change la représentation géométrique et mécanique de l'élément complet , voir la figure A.27 (partie supérier droite). Bien que les

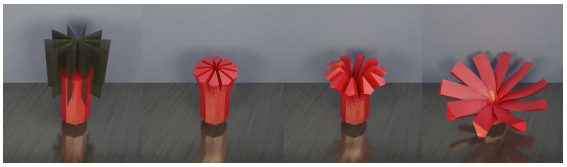
(a) Incision dans la chirurgie du genou, avec ouverture de la plaie



(b) Avancement d'une incision de la chirurgie hépatique



(c) Une découpe qui se traduit par une fissure de propagation



(d) Intersections des découpes

Figure A.26: Découpes virtuelles produit avec notre méthode.

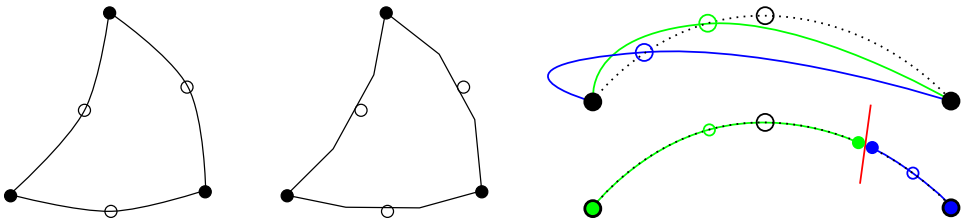


Figure A.27: Représentation d'un élément des polynômes du second degré: continu *(à gauche)* et la tesselation utilisé pour la visualisation *(au millieu)*; *Droite partie supérieur:* impact de la position du point médian; *Droite partie inférieur:* coupe d'une arête des polynômes du second degré

opérations topologiques du dernier chapitre puissent être réutilisées, la mise en place correcte des points est le défi majeur de l'application de l'algorithme de remaillage aux tétraèdres des polynômes du second degré. Veuillez vous référer à la version complète de ce manuscrit pour plus de détails.

Notre algorithme a été appliqué à un exemple de coupe sur une représentation virtuelle du foie , comme représenté dans la figure A.28. L'analyse théorique révèle qu'un grand nombre de nouveaux sommets sont insérés à cause des éléments de bord, ce qui rend l'algorithme peu adapté aux simulations en temps réel.

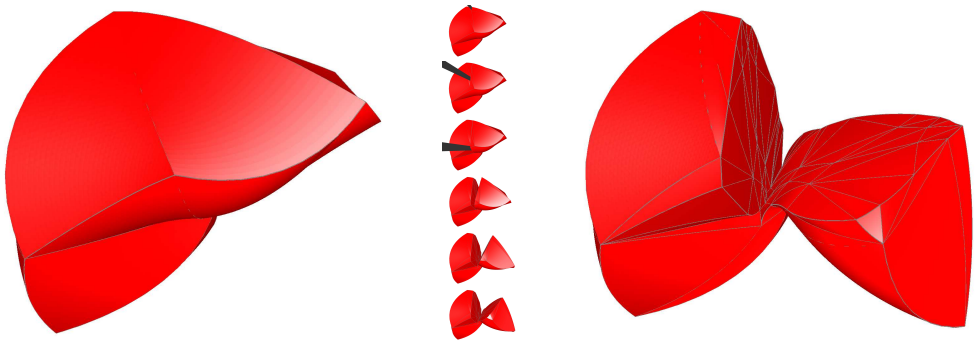


Figure A.28: Coupe du foie des polynômes du second degré avec une surface lisse *(rouge)* et des arêts courbes *(gris)*: configuration initiale *(à gauche)* et finale *(à droite)*; *Au millieu:* étapes intermédiaires de haut à bas *(outil de coupe en gris foncé)*

3 Réalité augmentée

La réalité augmentée s'est avérée prometteuse pour surmonter certains défis de visualisation et d'interaction dans divers domaines tels que la médecine, la construction, la publicité, la fabrication et le jeu. Malgré la promesse d'une réalité augmentée et son application réussie dans de nombreux domaines, d'importants défis de recherche demeurent. Parmi ces défis se trouve l'augmentation des structures elastiques qui peuvent subir des changements topologiques, tels que la rupture, le déchirement ou la coupe. Ce point est abordé dans cette section: nous présentons une nouvelle méthode, qui permet de détecter les changements topologiques à partir d'un flux vidéo monoscopiques.

Les travaux présentés dans ce qui suit sont basés sur notre travail publié [79, 78, 82]. La figure A.29 affiche le workflow de notre approche lorsqu'il est appliqué dans le cadre médical.

Un balayage préopératoire permet de construire un modèle virtuel de l'organe considéré. Ensuite, des caractéristiques sont identifiées dans un flux vidéo monoscopique, une copie virtuelle de ces caractéristiques est liée à l'organe virtuelle. Des forces d'étirement entre ces deux ensembles déplacent l'organe. Lorsque l'organe se déforme en douceur et sans modification topologique, le modèle et les entités virtuelles peuvent suivre correctement la vidéo. Dans notre travail nous utilisons la différence entre les caractéristiques réeles et virtuelles pour détecter les coupures et les exécuter sur le modèle virtuel et les structures internes du modèle , voir figure A.30.

Notre méthode a été évaluée sur plusieurs bandes de silicone , dont certaines sont représentées dans la figure A.31 (à gauche), cette methode peut détecter

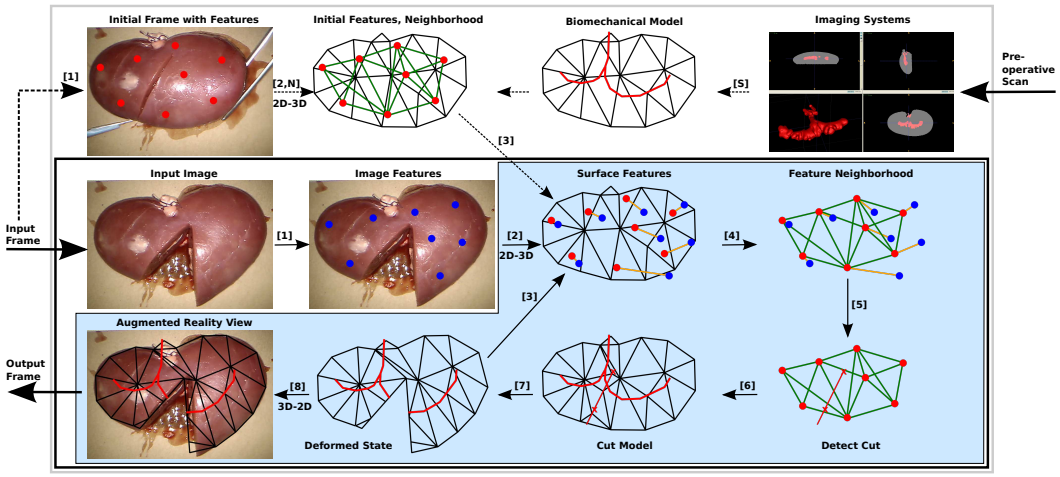


Figure A.29: Vue d'ensemble de notre méthode

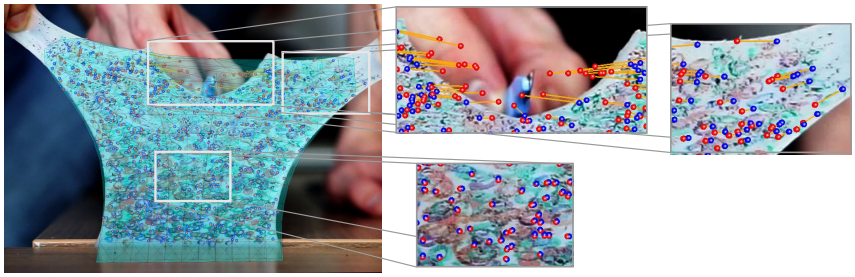


Figure A.30: Effets des changements topologiques dans l'image sur l'objet virtuel *(turquoise)* avec les characteristics dans l'image *(points bleus)* et sur l'objet virtuelle *(points rouges)* qui sont connectés par des ressorts *(lignes orange)*: *À gauche*: objet complet avec objet virtuel superposé; *À droite*: région de la coupure, région sans forte déformation et une région à forte déformation

des scénarios de découpage et de déchirure. Elle a également été évaluée sur les coupes d'un foie in vivo (voir figure A.31 (droite)) et les reins ex vivo avec des structures internes , voir figure A.32. Une évaluation avec la mesure de Dice et la distance de Hausdorff sur les exemples présentés confirme les résultats visuels positifs et montre l'amélioration par rapport aux méthodes proposées dans la literature, qui ne permettent pas de changements topologiques.

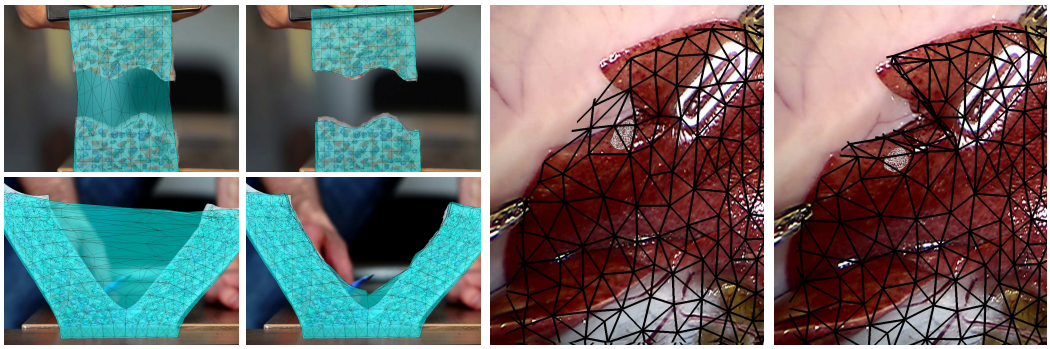


Figure A.31: Comparaison entre modèle virtuel non coupé et coupé: exemple de silicone *(à gauche)* et exemple de foie in vivo avec une tumeur insérée dans la représentation virtuelle *(à droite)*

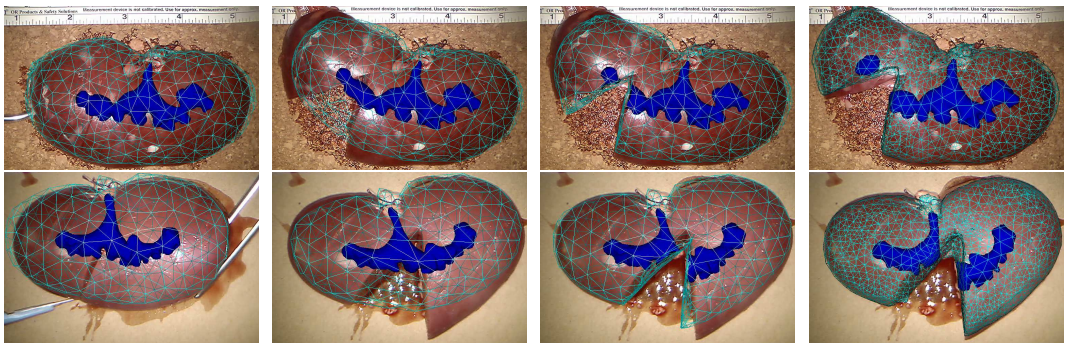


Figure A.32: Réalité augmentée sur des reins coupés et déformés, exemple 1 *(partie supérieur)* et 2 *(partie inférieur)* superposé par l'organe virtuel, l'enregistrement initial *(à gauche)*, enregistrements finaux: non découpé *(milieu gauche)*, découpé *(milieu droit)* et enregistrement de référence *(à droite)*.

Bibliography

[1] Antonio Agudo, Begona Calvo, and JMM Montiel. Finite element based sequential bayesian non-rigid structure from motion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1418–1425. IEEE, 2012.

[2] Jérémie Allard, Stéphane Cotin, François Faure, Pierre-Jean Bensoussan, François Poyer, Christian Duriez, Hervé Delingette, and Laurent Grisoni. Sofa-an open source framework for medical simulation. In *MMVR 15-Medicine Meets Virtual Reality*, volume 125, pages 13–18. IOP Press, 2007.

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1998.

[4] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[5] Adrien Bartoli and Andrew Zisserman. Direct estimation of non-rigid registrations. In *British machine vision conference*, pages 899–908. BMVA, 2004.

[6] Klaus-Jürgen Bathe. Finite element procedures in engineering analysis. 1982.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[8] Ted Belytschko and Tom Black. Elastic crack growth in finite elements with minimal remeshing. *International journal for numerical methods in engineering*, 45(5):601–620, 1999.

[9] Ted Belytschko, Yury Krongauz, Daniel Organ, Mark Fleming, and Petr Krysl. Meshless methods: an overview and recent developments. *Computer methods in applied mechanics and engineering*, 139(1):3–47, 1996.

[10] Ted Belytschko, Yun Yun Lu, and Lei Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[11] Jan Bender, Matthias Müller, Miguel A Otaduy, Matthias Teschner, and Miles Macklin. A survey on position-based simulation methods in computer graphics. In *Computer graphics forum*, volume 33, pages 228–251. Wiley Online Library, 2014.

[12] Jürgen Bey. Tetrahedral grid refinement. *Computing*, 55(4):355–378, April 1995.

[13] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. *Graphical Models*, 66(6):398–417, 2004.

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

[15] Jean Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm, 2002.

[16] 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.

[17] Huu Phuoc Bui, Satyendra Tomar, Hadrien Courtecuisse, Stéphane Cotin, and Stéphane P. A. Bordas. Real-time error controlled adaptive mesh refinement: Application to needle insertion simulation. *CoRR*, abs/1610.02570, 2016.

[18] D. Burkhart, B. Hamann, and G. Umlauf. Adaptive and feature-preserving subdivision for high-quality tetrahedral meshes. *Computer Graphics Forum*, 29(1):117–127, 2010.

[19] É Chamberland, André Fortin, and Michel Fortin. Comparison of the performance of some finite element discretizations for large deformation elasticity problems. *Computers & Structures*, 88(11):664–673, 2010.

[20] Indrajit Chowdhury and Shambhu P Dasgupta. Computation of rayleigh damping coefficients for large systems. *The Electronic Journal of Geotechnical Engineering*, 8(0), 2003.

[21] Philippe G Ciarlet and S Kesavan. *Lectures on three-dimensional elasticity*. Springer Berlin etc., 1983.

[22] Lenka Čížková and Pavel Čížek. Numerical linear algebra. In *Handbook of Computational Statistics*, pages 105–137. Springer, 2012.

[23] Olivier Comas, Zeike A Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger. Efficient nonlinear fem for soft tissue modelling and its gpu implementation within the open source framework sofa. In *International Symposium on Biomedical Simulation*, pages 28–39. Springer, 2008.

[24] Stéphane Cotin, Hervé Delingette, and Nicholas 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.

[25] Michele Diana, Peter Halvax, Damien Mertz, Andras Legner, Jean-Marcel Brulé, Eric Robinet, Didier Mutter, Patrick Pessaux, and Jacques Marescaux. Improving echo-guided procedures using an ultrasound-ct image fusion system. *Surgical innovation*, 22(3):217–222, 2015.

[26] Christian Dick, Joachim Georgii, and Rüdiger Westermann. A hexahedral multigrid approach for simulating cuts in deformable objects. *Visualization and Computer Graphics, IEEE Transactions on*, 17(11):1663–1675, 2011.

[27] Jack Edmonds. A combinatorial representation of polyhedral surfaces. *Notices of the American Mathematical Society*, 7, 1960.

[28] Francois Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*. Springer, June 2012.

[29] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, pages 283–321. Springer, 2012.

[30] Matthieu Ferrant, Arya Nabavi, Benoıt Macq, Peter M Black, Ferenc A Jolesz, Ron Kikinis, and Simon K Warfield. Serial registration of intraoperative mr images of the brain. *Medical image analysis*, 6(4):337–359, 2002.

[31] Elsa Fléchon, Florence Zara, Guillaume Damiand, and Fabrice Jaillet. A unified topological-physical model for adaptive refinement. In *Workshop on Virtual Reality Interaction and Physical Simulation*, pages 39–48, 2014.

[32] Isaac Fried. Condition of finite element matrices generated from nonuniform meshes. *Aiaa Journal*, 10(2):219–221, 1972.

[33] 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.

[34] Michael R Gosz. *Finite element method: Applications in solids, structures, and heat transfer*. CRC Press, 2005.

[35] Hartmut K Gumprecht, Darius C Widenka, and Christianto B Lumenta. Brainlab vectorvision neuronavigation system: technology and clinical experiences in 131 cases. *Neurosurgery*, 44(1):97–104, 1999.

[36] 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(3):375–385, 2006.

[37] W. Hackbusch and S.A. Sauter. Composite finite elements for the approximation of pdes on domains with complicated micro-structures. *Numerische Mathematik*, 75(4):447–472, 1997.

[38] Nazim Haouchine, Jérémie Dequidt, Marie-Odile Berger, and Stéphane Cotin. Single view augmentation of 3d elastic objects. In *Mixed and Augmented Reality (ISMAR)*, pages 229–236, 2014.

[39] Nazim Haouchine, Jérémie Dequidt, Erwan Kerrien, Marie-Odile Berger, and Stéphane Cotin. Physics-based Augmented Reality for 3D Deformable Object. In *VRIPHYS - Virtual Reality Interaction and Physical Simulation*, pages 31–38, Darmstadt, Germany, 2012.

[40] Nazim Haouchine, Jérémie Dequidt, Igor Peterlik, Erwan Kerrien, Marie-Odile Berger, and Stéphane Cotin. Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 199–208, 2013.

[41] Stefan Hartmann and Patrizio Neff. Polyconvexity of generalized polynomial-type hyperelastic strain energy functions for near-incompressibility. *International journal of solids and structures*, 40(11):2767–2791, 2003.

[42] Gerhard A Holzapfel. *Nonlinear solid mechanics*, volume 24. Wiley Chichester, 2000.

[43] Shiyu Jia, Weizhong Zhang, Xiaokang Yu, and Zhenkuan Pan. Cpu–gpu mixed implementation of virtual node method for real-time interactive cutting of deformable objects using opencl. *International journal of computer assisted radiology and surgery*, 10(9):1477–1491, 2015.

[44] Grand Roman Joldes, Adam Wittek, Mathieu Couton, Simon K Warfield, and Karol Miller. Real-time prediction of brain shift using nonlinear finite element algorithms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 300–307. Springer, 2009.

[45] Grand Roman Joldes, Adam Wittek, and Karol Miller. Suite of finite element algorithms for accurate computation of soft tissue deformation for surgical simulation. *Medical Image Analysis*, 13(6):912–919, 2009.

[46] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[47] Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. Enrichment textures for detailed cutting of shells. In *ACM Transactions on Graphics (TOG)*, volume 28, page 50. ACM, 2009.

[48] Nam-Ho Kim. *Introduction to nonlinear finite element analysis*. Springer Science & Business Media, 2014.

[49] Linda T Kohn, Janet M Corrigan, Molla S Donaldson, et al. *To err is human: building a safer health system*, volume 627. National Academies Press, 2000.

[50] Dan Koschier, Sebastian Lipponer, and Jan Bender. Adaptive tetrahedral meshes for brittle fracture simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 57–66, 2014.

[51] Pierre Kraemer, Lionel Untereiner, Thomas Jund, Sylvain Thery, and David Cazier. CGoGN: N-dimensional meshes with combinatorial maps. In *22nd International Meshing Roundtable*, pages 485–503. Springer International Publishing, October 2013.

[52] Ibai Leizea, Hugo Álvarez, Iker Aguinaga, and Diego Borro. Real-time deformation, registration and tracking of solids based on physical simulation. In *Mixed and Augmented Reality (ISMAR)*, pages 165–170, 2014.

[53] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[54] Abed Malti, Richard Hartley, Adrien Bartoli, and Jae-Hak Kim. Monocular template-based 3d reconstruction of extensible surfaces with local linear elasticity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1522–1529, 2013.

[55] Bruno Marques, Nazim Haouchine, Rosalie Plantefeve, and Stephane Cotin. Improving depth perception during surgical augmented reality. In *ACM SIGGRAPH 2015 Posters*, page 24. ACM, 2015.

[56] Jerrold E Marsden and Thomas JR Hughes. *Mathematical foundations of elasticity*. Courier Corporation, 1994.

[57] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. Polyhedral finite elements using harmonic basis functions. In *Computer Graphics Forum*, volume 27, pages 1521–1529. Wiley Online Library, 2008.

[58] Andrea Mendizabal, Rémi Bessard Duparc, Huu Phuoc Bui, Christoph J. Paulus, Igor Peterlik, and Stéphane Cotin. Towards a face-based smoothed finite element method for real-time simulation of the brain shift. 2017.

[59] Philippe Meseure, Emmanuelle Darles, Xavier Skapin, and Yazid Touileb. Adaptive resolution for topology modifications in physically-based animation. 2014.

[60] Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Interactive physically-based shape editing. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 79–89. ACM, 2008.

[61] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. 1996.

[62] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[63] AndrewB. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 1935 of *Lecture Notes in Computer Science*, pages 598–607. Springer, 2000.

[64] Francesc Moreno-Noguer, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Capturing 3d stretchable surfaces from single images in closed form. In *Computer Vision and Pattern Recognition CVPR*, pages 1842–1849, 2009.

[65] Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151. Eurographics Association, 2004.

[66] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. In *Computer graphics forum*, volume 25, pages 809–836. Wiley Online Library, 2006.

[67] Stéphane Nicolau, Luc Soler, Didier Mutter, and Jacques Marescaux. Augmented reality in laparoscopic surgical oncology. *Surgical oncology*, 20(3):189–201, 2011.

[68] Han-Wen Nienhuys and A Frank van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 145–152, 2001.

[69] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[70] Raymond W Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997.

[71] Daniel Organ, Mark Fleming, T Terry, and Ted Belytschko. Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Computational mechanics*, 18(3):225–235, 1996.

[72] James M Ortega. *Scientific Computing: eine Einführung in das wissenschaftliche Rechnen und parallele Numerik*. Springer-Verlag, 2013.

[73] Junjun Pan, Junxuan Bai, Xin Zhao, Aimin Hao, and Hong Qin. Dissection of hybrid soft tissue models using position-based dynamics. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pages 219–220. ACM, 2014.

[74] Junjun Pan, Junxuan Bai, Xin Zhao, Aimin Hao, and Hong Qin. Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics. *Computer Animation and Virtual Worlds*, 26(3-4):321–335, 2015.

[75] Junjun Pan, Shizeng Yan, Hong Qin, and Aimin Hao. Real-time dissection of organs via hybrid coupling of geometric metaballs and physics-centric mesh-free method. *The Visual Computer*, pages 1–12, 2016.

[76] Junjun Pan, Chengkai Zhao, Xin Zhao, Aimin Hao, and Hong Qin. Metaballs-based physical modeling and deformation of organs for virtual surgery. *The Visual Computer*, 31(6-8):947–957, 2015.

[77] Christoph Paulus, Stefan Suwelack, Nicolai Schoch, Stefanie Speidel, Rüdiger Dillmann, and Vincent Heuveline. Simulation of complex cuts in soft tissue with the extended finite element method (x-fem). *Preprint Series of the Engineering Mathematics and Computing Lab*, (02), 2014.

[78] Christoph J Paulus, Nazim Haouchine, David Cazier, and Stéphane Cotin. Augmented reality during cutting and tearing of deformable objects. In *Mixed and Augmented Reality (ISMAR)*, pages 54–59, 2015.

[79] Christoph J Paulus, Nazim Haouchine, David Cazier, and Stéphane Cotin. Surgical augmented reality with topological changes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[80] Christoph J. Paulus, Lionel Untereiner, Hadrien Courtecuisse, Stéphane Cotin, and David Cazier. Virtual cutting of deformable objects based on efficient topological operations. *The Visual Computer*, 31(6-8):831–841, 2015.

[81] Christoph J. Paulus, Lionel Untereiner, Hadrien Courtecuisse, Stéphane Cotin, and David Cazier. Virtual cutting of deformable objects based on efficient topological operations. In *Computer Graphics International (CGI)*, 2015.

[82] Christoph Joachim Paulus, Nazim Haouchine, Seong-Ho Kong, Renato Vianna Soares, David Cazier, and Stéphane Cotin. Handling topological changes during elastic registration: Application to augmented reality in laparoscopic surgery. *International Journal of Computer Assisted Radiology and Surgery (IJCARS)*, 2016.

[83] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J Guibas. Meshless animation of fracturing solids. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 957–964. ACM, 2005.

[84] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J Guibas. Meshless animation of fracturing solids. *ACM Transactions on Graphics (TOG)*, 24(3):957–964, 2005.

[85] Mathieu Perriollat, Richard Hartley, and Adrien Bartoli. Monocular template-based reconstruction of inextensible surfaces. *International Journal of Computer Vision*, 95(2):124–137, 2011.

[86] Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano. Real-time tracking of 3d elastic objects with an rgb-d sensor. In *Intelligent Robots and Systems (IROS)*, pages 3914–3921, 2015.

[87] Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano. Tracking fractures of deformable objects in real-time with an rgb-d sensor. In *International Conference on 3D Vision (3DV)*, pages 632–639, 2015.

[88] 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.

[89] Julien Pilet, Vincent Lepetit, and Pascal Fua. Fast non-rigid surface detection, registration and realistic augmentation. *International Journal Computer Vision*, 76(2):109–122, 2008.

[90] Rosalie Plantefeve, Igor Peterlik, Hadrien Courtecuisse, Raffaella Trivisonne, Jean-Pierre Radoux, and Stéphane Cotin. Atlas-based transfer of boundary conditions for biomechanical simulation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 33–40. Springer, 2014.

[91] Philip Pratt, Danail Stoyanov, Marco Visentini-Scarzanella, and Guang-Zhong Yang. Dynamic guidance for robotic surgery using image-constrained biomechanical models. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 77–85, 2010.

[92] Kun Qian, Junxuan Bai, Xiaosong Yang, Junjun Pan, and Jianjun Zhang. Virtual reality based laparoscopic surgery simulation. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, pages 69–78. ACM, 2015.

[93] Kun Qian, Junxuan Bai, Xiaosong Yang, Junjun Pan, and Jianjun Zhang. Essential techniques for laparoscopic surgery simulation. *Computer Animation and Virtual Worlds*, 2016.

[94] María-Cecilia Rivara. Local modification of meshes for adaptive and/or multigrid finite-element methods. *Journal of Computational and Applied Mathematics*, 36(1):79–89, August 1991. Special Issue on Adaptive Methods.

[95] M. Salzmann and P. Fua. Linear local models for monocular reconstruction of deformable surfaces. *Pattern Analysis and Machine Intelligence*, 33(5):931–944, May 2011.

[96] Mathieu Salzmann, Julien Pilet, Slobodan Ilic, and Pascal Fua. Surface deformation models for nonrigid 3d shape recovery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(8):1481–1487, August 2007.

[97] S.A. Sauter and R. Warnke. Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing*, 77(1):29–55, 2006.

[98] Scott Schaefer, Jan Philipp Hakenberg, and Joe Warren. Smooth subdivision of tetrahedral meshes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on geometry processing*, SGP '04, pages 147–154, New York, NY, USA, 2004. ACM.

[99] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[100] N. Schoch, S. Suwelack, R. Dillmann, and V. Heuveline. Simulation of surgical cutting of soft tissue using the x-fem. *Preprint Series of the Engineering Mathematics and Computing Lab*, (04), 2013.

[101] Martin Seiler, Denis Steinemann, Jonas Spillmann, and Matthias Harders. Robust interactive cutting based on an adaptive octree simulation mesh. *The Visual Computer*, 27(6-8):519–529, 2011.

[102] J Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*, 73, 2002.

[103] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[104] Prateek Shrivastava and Sukhendu Das. Particle coding for meshfree cutting of deformable assets. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, page 6. ACM, 2014.

[105] Hang Si and A TetGen. A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany*, 2006.

[106] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, page 20, 2012.

[107] Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 73–80, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[108] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–90. Eurographics Association, 2007.

[109] Ajay Singh and Glenn H Strauss. High-fidelity cataract surgery simulation and third world blindness. *Surgical innovation*, page 1553350614537120, 2014.

[110] Michael A Slawinski. *Waves and rays in elastic continua*. World Scientific, 2010.

[111] S Speidel, E Kuhn, S Bodenstedt, S Röhl, H Kenngott, B Müller-Stich, and R Dillmann. Visual tracking of da vinci instruments for laparoscopic surgery. In *SPIE Medical Imaging*, pages 903608–903608. International Society for Optics and Photonics, 2014.

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

[113] Denis Steinemann, Miguel A Otaduy, and Markus Gross. Fast arbitrary splitting of deforming objects. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 63–72. Eurographics Association, 2006.

[114] Baiquan Su, Xiaole Wang, and Hongen Liao. Soft tissue cutting control method with surgical scalpel.

[115] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[116] S. Suwelack. *Real-Time Biomechanical Modeling for Intraoperative Soft Tissue Registration*. Karlsruhe Institute of Technology – Scientific Publishing, 2013.

[117] Stefan Suwelack, Sebastian Röhl, Sebastian Bodenstedt, Daniel Reichard, Rüdiger Dillmann, Thiago dos Santos, Lena Maier-Hein, Martin Wagner, Josephine Wünscher, Hannes Kenngott, Beat P. Müller, and Stefanie Speidel. Physics-based shape matching for intraoperative image guidance. *Medical physics*, 41(11):111901, 2014.

[118] Stefan Suwelack, Sebastian Röhl, Rüdiger Dillmann, Anna-Laura Wekerle, Hannes Kenngott, Beat Müller-Stich, Céline Alt, and Stefanie Speidel. Quadratic corotated finite elements for real-time soft tissue registration. In *Computational Biomechanics for Medicine*, pages 39–50. Springer, 2012.

[119] Joseph Teran, Neil Molino, Ronald Fedkiw, and Robert Bridson. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with computers*, 21(1):2–18, 2005.

[120] Aggeliki Tsoli and Antonis A Argyros. Tracking deformable surfaces that undergo topological changes using an rgb-d camera. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 333–341. IEEE, 2016.

[121] Luca Viganò, Claude Tayar, Alexis Laurent, and Daniel Cherqui. Laparoscopic liver resection: a systematic review. *Journal of hepato-biliary-pancreatic surgery*, 16(4):410–421, 2009.

[122] Erke Wang, Thomas Nelson, and Rainer Rauch. Back to elements-tetrahedra vs. hexahedra. In *Proceedings of the 2004 International ANSYS Conference*, 2004.

[123] Yuting Wang, Chenfanfu Jiang, Craig Schroeder, and Joseph Teran. An adaptive virtual node algorithm with robust mesh cutting. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 77–85. Eurographics Association, 2014.

[124] Martin Wicke, Mario Botsch, and Markus Gross. A finite element method on convex polyhedra. In *Computer Graphics Forum*, volume 26, pages 355–364. Wiley Online Library, 2007.

[125] Peter Wriggers. *Nonlinear finite element methods*. Springer Science & Business Media, 2008.

[126] 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 (VRIPHYS) 2011*, pages 29–38, 2011.

[127] 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.

[128] Jun Wu, Rüdiger Westermann, and Christian Dick. Physically-based simulation of cuts in deformable bodies: A survey. In *Eurographics State-of-the-Art Report*, 2014.

[129] 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)*, 196:469–475, 2014.

[130] Ziv Yaniv and Kevin Cleary. Image-guided procedures: A review. *Computer Aided Interventions and Medical Robotics*, 3:1–63, 2006.

[131] Yongjie Zhang, Chandrajit Bajaj, and Bong-Soo Sohn. 3d finite element meshing from imaging data. *Computer methods in applied mechanics and engineering*, 194(48):5083–5106, 2005.

[132] Jianke Zhu and Michael R Lyu. Progressive finite newton approach to real-time nonrigid surface detection. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[133] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.