Exploring new numerical methods for the simulation of soft tissue deformations in surgery assistance

Jean-Nicolas Brunet

UNIVERSITÉ DE STRASBOURG

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

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

THESIS

presented for the grade of

Docteur de l'Université de Strasbourg

Discipline (mention): Informatique (computer science)

by

Jean-Nicolas BRUNET

Exploring new numerical methods for the simulation of soft tissue deformations in surgery assistance

Defended publicly on the **November 4, 2020**

Members of the jury:

M. Adam Wittek ..... *Reviewer*
Professor at the University of Western Australia, Australia

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

M. Ole Jacob Elle ..... *Examiner*
Research director at the Oslo University Hospital, Norway

M. Yannick Privat ..... *Examiner*
Professor at the University of Strasbourg, France

M. Stéphane Cotin ..... *Advisor*
Research Director at the University of Strasbourg, France

Université de Strasbourg

Jean-Nicolas BRUNET

Exploring new numerical methods for the simulation of soft tissue deformations in surgery assistance

Résumé

Cette thèse aborde le problème de *simulation des tissus mous pour les applications de réalité augmentée en assistance peropératoire du foie* et, plus précisément, la mise en œuvre d'une procédure automatique de recalage non rigide entre une reconstruction préopératoire du foie d'un patient et les données acquises en temps réel pendant la chirurgie. Un cadre formel basé sur la physique est d'abord défini et utilisé comme base pour la construction d'un modèle biomécanique capable de reproduire les déformations du foie. Quatre directives de recherche ont guidé le développement du modèle: la précision, la rapidité, la stabilité et la simplicité de mise en œuvre. Les méthodes *sans maillage* et les méthodes *aux frontières immergées* sont toutes deux considérées comme des alternatives à la méthode traditionnelle des éléments finis. Un algorithme complet de recalage non rigide est documenté et testé avec des scénarios réels. Finalement, une introduction des solutions émergentes en apprentissage automatique et réseaux de neurones est également fournie.

Mots-clés: Modèle biomécanique, Méthodes sans maillage, Méthodes aux frontières immergées, Recalage non rigide, Déformation du foie, Assistance peropératoire, Réalité augmentée

Abstract

This thesis addresses the problem of *soft tissue simulation for augmented reality applications in liver surgery assistance* and, more specifically, the implementation of a non-rigid registration pipeline to be used by the medical staff to generate interactive deformations of a patient specific liver three-dimensional virtual representation. A formal physics-based framework is first defined and used as the basis for the construction of a biomechanical model capable of producing realistic deformations. Four basic requirements guided the development of the model: accuracy, speed, stability and simplicity of implementation. *Meshless* and *immersed-boundary* methods are both considered as alternatives to the traditional finite element method. A formal non-rigid registration algorithm is finally documented and tested with real-life scenarios. A comparison with new and rising machine learning and neural network solutions is also provided.

Keywords: Biomechanical model, Meshless methods, Immersed-Boundary methods, Non-rigid registration, Liver deformation, Intra-operative guidance, Augmented reality

À ma femme...

Acknowledgments

Tout d'abord, j'aimerais remercier mon directeur de thèse, Stéphane Cotin, qui non seulement m'a permis de réaliser ce projet de thèse, mais m'a également permis de vivre une expérience de vie inoubliable. Merci Stéphane pour ton soutient, tes sages conseilles, ta patience et ton écoute. Surtout, merci de m'avoir si bien transmis ta passion pour ce domaine de recherche.

I would like to express my gratitude to the members of my thesis committee: Adam Wittek, Fabrice Jaillet, Ole Jacob Elle and Yannick Privat. I want to address a special thanks to the two reviewers Adam Wittek and Fabrice Jaillet. Having its thesis read and reviewed by such renowned experts despite their busy schedule is truly an honor.

J'aimerais remercier Hugo Talbot, d'abord pour m'avoir si bien intégré dans l'équipe au début de cette thèse, puis pour ton support constant durant le reste ce celle-ci. Au-delà du travail, toi, Frédérique Roy et Andrea Mendizabal êtes devenus mes confidents, mes partenaires de soirées, et mes meilleurs amis. Vous avez éclairé les moments les plus sombres de cette thèse et je vous en serai toujours reconnaissant.

Merci à Vincent Magnoux de m'avoir accompagné dans ce début thèse au débrouillage et à la difficile implémentation des méthodes sans maillage. J'espère de tout coeur que nous aurons l'occasion de retravailler ensemble, nous formons une équipe incroyable. J'ai adoré travailler avec toi.

Merci à mon père, Gilles Brunet, pour son appui incommensurable du début à la fin de ce projet. Merci, papa, de m'avoir constamment encouragé à aller au bout de mes rêves. Merci pour l'aide colossale que tu m'as apportée à la rédaction et à la relecture de ce manuscrit.

Finalement, j'aimerais remercier ma femme, Aurélie, sans qui cette thèse n'aurait tout simplement pas été possible. Merci d'avoir été à mes côtés chaque jour de ce projet interminable. Merci pour ta patience, tes encouragements, et ton soutien inconditionnel. Cette thèse t'appartient tout autant qu'à moi. Nous formons une équipe exceptionnelle, et chaque jour je réalise un peu plus la chance que j'ai de t'avoir comme femme et mère de nos enfants. Merci d'être dans ma vie.

List of acronyms

AR Augmented Reality

CAI Computer-Assisted Interventions

CG Conjugate Gradient

CT Computed Tomography

DOF Degrees Of Freedom

FC Finite Cell

FCM Finite Cell Method

FE Finite Element

FEM Finite Element Method

FPK First Piola-Kirchhoff

HiPerNav High Perfomance Soft Tissue Navigation

IB Immersed boundary

ICP Iterative closest point

ITN Innovative Training Network

LMS Least median of squares

MIS Minimally Invasive Surgery

ML Machine Learning

MLAMBI Meshless Approximation Mesh-Based Integration

MLS Moving Least Squares

MRI Magnetic Resonance Imaging

MTLED Meshless Total Lagrangian Explicit Dynamics

NN Neural Network

NR Newton–Raphson

OR Operation Rooms

PBA Point-Based Animation

PC Points Cloud

PCA Principle Components analysis

PDE Partial Differential Equation

PIC Particle In Cell

SOFA Simulation Open Framework Architecture

SPH Smoothed Particle Hydrodynamics

SPK Second Piola-Kirchhoff

SVD Singular Value Decomposition

SVK Saint Venant–Kirchhoff

TFCM Tetrahedral Finite Cell Method

US Ultrasound

VR Virtual Reality

WC Weighted Cell

Contents

Chapter

1 INTRODUCTION

Simulations of physical phenomenons through numerical and computer engineering are nowadays well embedded in our society. Applications vary widely and range from structural design of infrastructure, realistic animations in movies and video games, intermolecular interactions of human cells, airflow and heat transfer in large cities, efficient design of microprocessors, and more recently, estimations of virus propagation from respiratory droplets in the COVID-19 pandemic situation.

In this thesis, we are interested in a very specific application: the *simulation of soft tissues for augmented reality in liver surgery interventions*. Our project was part of the European Innovative Training Network (ITN) *High Perfomance Soft Tissue Navigation (HiPerNav)* funded by the European Union's Horizon 2020

Research and Innovation program under a Marie Skłodowska-Curie grant. In this program, 16 PhD students worked jointly to research and develop new methods in various areas of soft tissue navigation.

In this chapter, we first discuss the relevance of physics-based soft tissue simulations for computer-assisted interventions. We will present some of the challenges found in this field, how they were approached in the past, and the reasons why they are still the object of many ongoing research. We will then discuss the problematic at the center of this thesis, and some of the hypotheses that will be used to address it. Finally, we will conclude with a clear record of the contributions found in this work and a detailed plan of the manuscript.

1.1 Liver surgery simulation

The human liver is a complex and fascinating organ. Located just below the diaphragm, it is one of the largest organs of the human body with an average width of 15 centimeters. Weighing approximately 1.5 kilograms, it is also the heaviest, just before the brain and the lungs. Its functions vary from the production of the bile, synthesis of many proteins and enzymes, breakdown of hormones such as insulin, and bare many more vital duties. It is also the only internal organ having the extraordinary capability to regenerate up to 25% of its lost tissue.

Unfortunately, it is also an extremely frequent target of various cancers. In fact, primary liver cancers are the sixth-most frequent cancers found in humans. With about 782 000 deaths in 2018, they also hold the tragic second place at being the deadliest cancers worldwide [Bray et al. (2018)]. Many efforts are made to reduce these numbers and partial liver resection remains the optimal treatment [Gilg et al. (2017); Fong and Tanabe (2014)]. The general idea here is to physically remove malignant tumors from the liver, hopefully before cancer cells spread elsewhere.

Liver resections have traditionally been done through open surgery, which means that surgeons make a large incision (usually around 20 centimeters) across the right upper abdomen of the patient in order to access its organ. However, depending on the location and size of tumors, a Minimally Invasive Surgery (MIS), often called a *laparoscopic surgery* in the context of abdominal interventions, can also be done (figure 1.1). During this type of intervention, only three to four tiny incisions (usually of less than 1.5 centimeters each) are made on the patient's abdomen. The abdomen of the patient is inflated with carbon dioxide, creating an artificial pneumoperitoneum (pneumatosis in the peritoneal cavity). A *laparoscope*,

an advanced surgical tool to which a camera is attached and connected to a fiber cable, is then inserted inside one of the incisions. The pneumoperitoneum creates a working space for the surgical team to move the laparoscope and other surgical tools inside the patient.

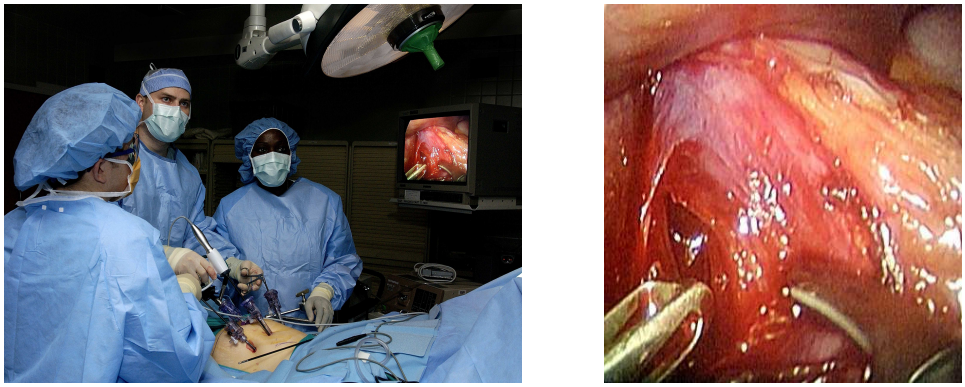


Figure 1.1: Laparoscopic surgery (Samuel Bendet, US Air Force, Wikipedia)

MIS benefits are numerous. The use of smaller incisions diminishes post-operation pain which contributes to reducing sedative medication. It also shortens the recovery time, lowers the chances of hemorrhaging, reduces the risk of infections, and in some cases avoid the need for general anesthesia [Viganò et al. (2009); Bajwa and Kulshrestha (2016); H. Li et al. (2017)].

However, a MIS procedure is more difficult to accomplish compared to a traditional open surgery. Movements of the laparoscopic instruments during the intervention are highly limited, thus surgeons require a lot of dexterity. They also require an extensive training which isn't always possible due to material, time and logistic costs. In fact, the training of surgeons for MIS procedures is the main motivation behind a large field of surgery simulations: the *virtual training* of laparoscopic intervention using surgery simulators. Instead of practicing MIS maneuvers on animals or human cadavers - which is not only expensive but also brings a lot of ethical considerations - surgeons can develop their skill in a completely virtual environment. There, physically plausible animations of a human liver and its surroundings are made available to students. When combined with a haptic device which mimics tactile sensations of laparoscopic tools, surgeons are immersed into a virtual world that prepares them to real life interventions. Benefits of virtual training have been demonstrated on many occasions [Haluck et al. (2001); Seymour et al. (2002); Feudner et al. (2009); Haller et al. (2009); Selvander and Åsman (2012)].

But even for a well-trained and experienced surgeon, a MIS intervention remains

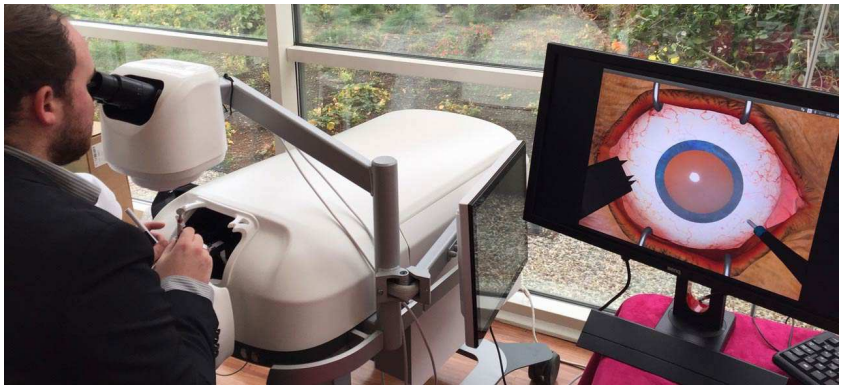


Figure 1.2: Retina surgery simulator with haptic feedback

a complex procedure. Poor depth perception from the laparoscope camera makes difficult the navigation of the surgical tools. Not being able to tactically feel the tissue makes it hard for the surgeon to locate vessels or tumors. Moreover, any surgical plan made before the operation using medical images such as Computed Tomography (CT) scan or Magnetic Resonance Imaging (MRI) can become difficult to interpret during the intervention since the shape of the liver will deform significantly due to the pneumoperitoneum. These difficulties motivated another large (and more recent) field in surgery simulations: *augmented reality for computer-aided intervention.*

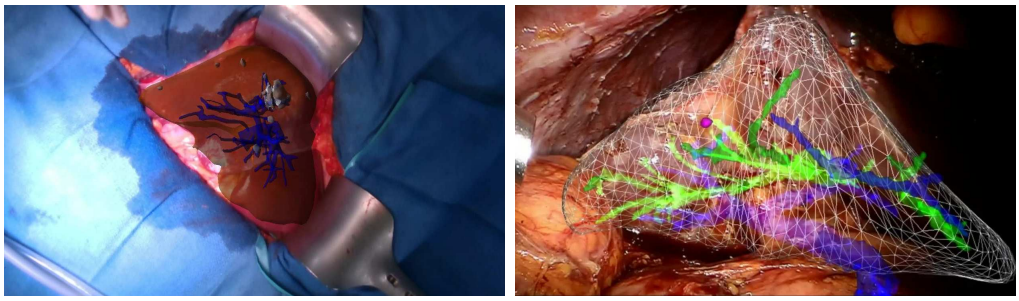
1.2 Augmented reality for liver surgery

The main goal of Augmented Reality (AR) applications is to enhance the objects found in the real-world environment of an user by overlaying computer-generated information across multiple sensory modalities. The enhancement can take many forms such as an augmented visual view of the surrounding objects, haptic feedback tools that simulate physical contacts, advanced 3D sound rendering, etc. When sensory modalities are completely substituted by computer-generated ones, we fall into the branch of Virtual Reality (VR) applications. The real-world environment is then replaced by a virtual one resulting in a complete immersion.

In this thesis, we are interested in applying the AR approach in the context of Computer-Assisted Interventions (CAI). The research field of CAI is quite extensive and includes applications varying from medical robotics to needle insertion assistance, surgical planning, medical-image processing, and many more. Some of them, in particular *surgical and interventional navigation* applications, aim at

improving the workflow of a medical team during a surgical intervention. When coupled with AR, it will usually take the form of projecting computer-generated information of the current procedure directly onto images taken in real time. This can be done, for example, by using specialized AR glasses, or by simply overlying the generated information on top of the images acquired by a camera such as a laparoscope.

Our research effort will focus on a very specific application of Augmented Reality (AR) to the Computer-Assisted Interventions (CAI) field: *augmented reality for computer-assisted surgery*. Our goal will be to automatically deform a three-dimensional virtual representation of a patient's liver such that it constantly matches the real shape of the organ during surgery, a concept called *non-rigid registration*. It can be seen as an extension to *rigid registration* whereby the virtual organ is overlaid onto acquired images of the patient's liver and it is viewed as "rigid", hence does not deform. Note that the registration concept is not restricted to 3D virtual models and is also applied to different medical modalities, such as registering MRI 2D slices taken before the surgery to ultrasound images acquired during the operation.



(a) Open surgery (b) Laparoscopic surgery

Figure 1.3: Non-rigid registration where the deformed virtual model of a liver is overlaid on the top of images taken during the surgery.

The overlay of the virtual 3D model (see figure 1.3) can be used by the medical team to better adjust the initial surgery plan as the procedure advances. For example, it can be used to visualize the location of tumors, or to avoid the resection of critical parts of the liver. This is especially useful for laparoscopic interventions since, as mentioned before, surgeons can only rely on surgical tools and do not have any tactile perceptions of the liver that would normally allow them to feel the location of a tumor or blood vessel. The liver having undergone extensive deformations from the pneumoperitoneum, surgeons might also have difficulties to locate its internal structures from medical images taken before the surgery when the liver had a completely different shape and orientation.

A typical AR workflow in non-rigid MIS goes as follows. First, an initial pre-operative MRI or CT scan of the patient's liver is done. In the case of a CT scan, a radio contrast (generally iodine-based) is usually injected to better highlight the internal structures such as blood vessels or tumors. Both MRI and CT scans will produce a series of 2D images that are merged into a single 3D image using volume rendering techniques [Udupa and Herman (1999)]. The 3D image is represented by a grid of voxels (3D rectangular cells). Each voxel is assigned a color intensity that can be used to differentiate between material properties at different locations within the patient's body. At this point, the 3D image does not differentiate the liver from its surrounding tissues and organs. The second step consists in isolating the liver from the rest of the image, a process called *segmentation*. It is normally done through either a manual, automatic or semi-automatic method using advanced medical imaging software. Additional segmentation is normally being done to isolate the internal structures of the liver such as blood vessels and tumors from the remaining of its parenchyma. Optionally, a triangulation step such as the marching cube algorithm [Newman and Yi (2006)] is performed on the segmented image to produce a surface mesh that delimits the boundaries of the liver and its internal structures.

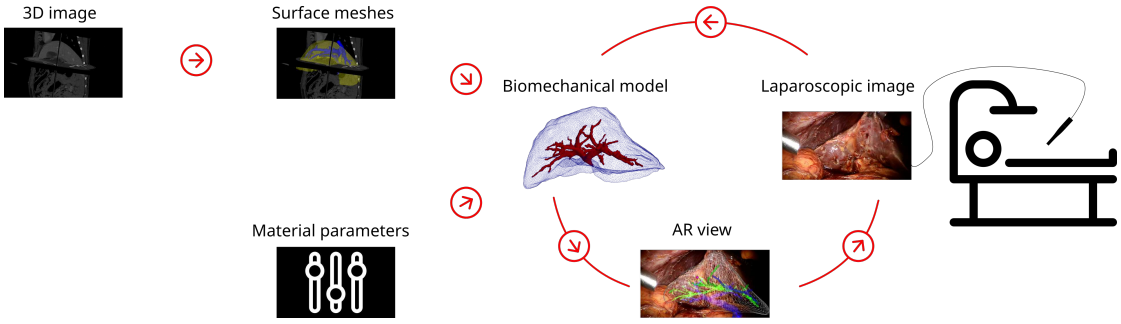


Figure 1.4: Typical non-rigid registration workflow for laparoscopic CAI

From the segmented images (or surface meshes), labels are assigned to different parts of the liver together with material parameters. These parameters are used to define the deformable behavior of a *biomechanical model* which is at the core of the workflow. As can be seen in figure 1.4, the biomechanical model is the glue between the pre-operative data, the real-time acquired data during the surgery (intra-operative data), and the overlaid AR information showed to the medical team and will be the general context of this thesis.

1.3 Key requirements

The biomechanical model can be seen as a black box having the pre-operative liver reconstruction (either in the form of segmented 3D images or surface meshes of its boundaries) and its elasticity parameters as initial inputs. For this reason, the model is often called the *patient-specific* biomechanical model. When given some information on the current state of the liver during the surgery, the model will generate a deformed shape of the pre-operative reconstruction. Several approaches can be used to produce a deformation. However, within the context of an AR application for CAI, we are looking for numerical methods that meet very specific requirements given the high standards of a medical environment and more importantly, the potential impacts on the patient.

Four key requirements will be used to guide the development of our biomechanical model: accuracy of the solution, computational speed, stability and the simplicity of the overall process.

1.3.1 Accuracy

Accuracy is usually the first requirement to be considered in medical applications such as surgery simulations. In our context, this requirement can be formulated as *how close the virtual liver generated by the biomechanical model is from the real organ*? It is measured using the internal structures of the liver and their positions predicted by the model. The model will therefore be considered accurate if the computed deformed shape of the liver and its internal structures match the true organ deformations observed during the surgery.

This requirement led us naturally to the theory of elasticity and other continuum mechanic concepts that allows the state of a material having undergone external forces to be described mathematically. In a surgical context, these external forces usually result from the surgeon's manipulations on the liver, the effect of gravity or the pneumoperitoneum. Hence, the mathematical equations describe the equilibrium between these external forces and the internal elastic forces of the organ that tries to regain its initial shape at rest. The solution produced by the biomechanical model is therefore the deformed shape that cancels out the internal and external forces. To reach this equilibrium, the model needs to execute a series of numerical operations. Given the complexity of mathematical equations required to produce exact solutions, approximations or heuristics must be used. Our choice of approximations will obviously be guided by the level of accuracy required for

surgical applications.

1.3.2 Speed

The main motivation of an AR workflow is to output accurate information dynamically to allow the medical team to adapt their surgical plan in real-time. If the workflow requires too much time to produce its output, the medical team might simply ignore the simulation tool because the gains associated with a slow AR solution would be outweighed by the risk of affecting the outcome of the surgery. Hence the speed objectives for our model depend entirely on where the medical team will place the risk of a slow response versus the usefulness of the simulation. This is in total contrast with the behavior found in surgical simulations for VR applications where the speed criterion is set for an optimal immersion, and is usually well defined in the literature. For example, a typical surgery simulator will usually try to update the visual of the virtual organ between 30 to 60 times per second to achieve a feeling of continuous motion, and an update rate of about 1000 Hz for haptic devices [J. Zhang, Zhong, and C. Gu (2017)].

In the context of this thesis, we assume that the medical team will not be constantly looking at the AR display and we will therefore tend to relax this constraint. Clearly, an accurate solution should be given in a much lower time frame than what is usually required for hybrid OR to acquired an intra-operative volumetric image. From our experience, contrast-enhanced CT and cone beam CT image acquisitions can easily take up to 15 minutes for an experienced surgical team.

1.3.3 Stability

In a simulation environment, stability is related to the robustness of the numerical method when facing unpredictable and often non-physical inputs. These inputs may come from a noisy and incomplete surface reconstruction of intra-operative liver images acquired during the surgery. For example, the reconstruction could misinterpret some parts of the tissues surrounding the liver as the liver itself. The model will therefore try to produce a shape that does not make any physical sense. As we will see in the following chapters, complex biomechanical models targeted at very accurate simulations will usually be very sensitive to these non-physical inputs and will often diverge completely, thus requiring a complete restart of the simulation. On the other hand, a method using weaker approximations will be less sensitive to those inputs, but might produce numerical artifacts. These artifacts

might accumulate and, at some point, reach a state where the simulation can no longer continue. This is obviously not desirable and the choice of the numerical method must reflect a balance between the two.

1.3.4 Simplicity

The last three requirements will tend to require a certain amount of complexity that will undoubtedly affect the implementation of the method, its operation or both. A balance between these three requirements against the amount of configuration work and technical knowledge imposed on the end user is required. This is even more essential in the context of *patient-specific* simulation. The biomechanical model has to be reconfigured for each patient in order to incorporate, for example, accurate information on the location of its tumors or blood vessels, external structures attached to the liver that restrict its movement inside the peritoneal cavity, and material properties. Since the time between a pre-operative scan and the surgery might be quite short, sometimes less than half a day, it is clear that the configuration of the model should remain as fast and effortless as possible. With the rise of fully automatic segmentation methods [Luo, X. Li, and J. Li (2014)], it seems natural to lean towards a method that requires a minimal amount of configuration work, thereby getting closer to a fully automatic CAI workflow.

A biomedical model that is simple to implement is also quite important as it may impact the decision of pursuing or not a given research approach. For example, and as we have realized during this thesis, obtaining the necessary resources to implement a complex method might simply be too expensive and ultimately resolves in the research team simply dropping this direction of research. Another consideration is the amount of work required to set up the biomechanical model. Depending on the approach used, this work may often be completely or partially lost as soon as the topology of the simulated object changes. In our context, these types of changes arise when the liver is cut, or teared in parts. A numerical method requiring that a large number of elements be recomputed after a topological change may quickly become too time consuming and expensive for the medical team.

1.4 Objectives of this thesis

Managing pre-operative data and real time acquired intra-operative images represents a challenging task and requires the usage of patient-specific biomechanical

models capable of accurate registrations between different imaging modalities. The model is normally based on a mathematical framework made of a set of Partial Differential Equation (PDE)s, often called *constitutive equations*, that has to be numerically solved. In this framework, the discretization of the virtual domain is required and multiple approaches are possible. However, very few meet the four key requirements listed in the previous section.

Finite Element methods have been the *de facto* approach for accurate surgery simulations, both for virtual training and CAI applications [J. Zhang, Zhong, C. Gu, and Coloe (2017); F.-y. Li et al. (2018); Malukhin and Ehmann (2018)]. The general idea behind FE methods is to reduce the initial Partial Differential Equation (PDE) problem to a finite set of weaker and discrete equations that have to be solved on geometrical elements. Hence, the interior volume of the simulated object, or the virtual organ in our case, is filled with a *mesh* of iso-parametric elements (tetrahedra, hexahedra, prisms, etc.). In this mesh, two adjacent elements are joined through common nodes between them. Since we can easily define geometrical procedures to integrate and interpolate a field function within these elements from the value of their nodes, the problem of finding a deformed state of the whole volume can be transposed to that of solving the deformed state of the mesh's nodes. Hence, in a three-dimensional problem, we will say that a discrete node has three unknown, often called Degrees Of Freedom (DOF), namely the x, y and z components of the nodal displacement vector $\boldsymbol{u} \in \Re^3$.

The main issue with classical FE methods is with the quality of the mesh which greatly influences the criteria associated with our key requirements. The domain of interest must be discretized by a mesh of geometrical elements that conforms to its boundary and the interface between different internal structures of the organ. For this purpose, a dense mesh made of a large quantity of elements around angled boundaries is usually required [Wu et al. (2001)]. Because the numerical operations of the methods are executed on every one of these elements, a dense mesh will rapidly degrade the speed of the simulation. Alternatively, a coarser mesh could be made from fewer but distorted elements (e.g. sliver elements). These badly shaped elements are known to cause various numerical issues, hence failing the robustness requirement [P.-L. et al. (2016); Bathe and L. Zhang (2017)]. Finally, a mesh could be built out of fewer and well-formed elements by relaxing this boundary conformance constraint, thus allowing elements to cross the boundaries of the simulated domain. As we will see in chapter 4, these nonconforming meshes directly affect the accuracy of the solution unless special attention is made for the computation over elements cut by the boundary. Figure 1.5 illustrates this problem. Hence, the discretization process can rapidly become a tedious task, especially in patient-specific models where the surface mesh is generated from medical images, leading

to complex, often incomplete or invalid surface representation with many internal structures. This is in total contradiction with the simplicity requirement we chose to impose on the overall process. In fact, an experienced engineer can easily spend several hours, sometimes even days, to create a FE mesh that accurately reflects the important structures of the liver, that is robust enough to handle large deformations, and that is able to produce real-time solutions.

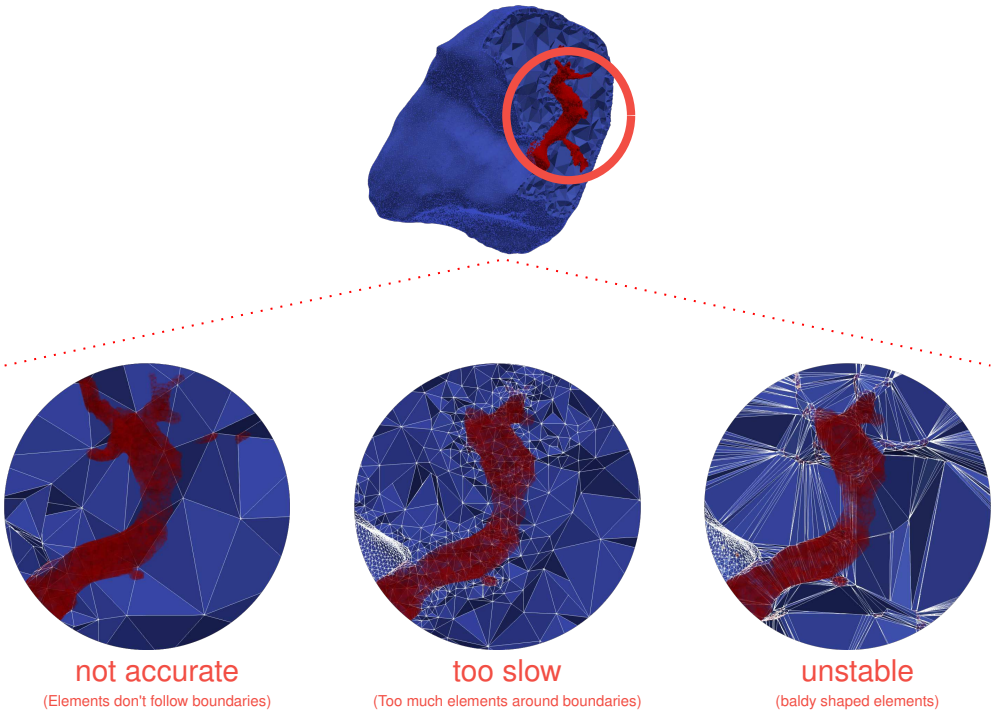


Figure 1.5: Typical meshing difficulties found with standard FE methods.

To alleviate the difficulty inherent to the meshing process inside the boundaries, we are looking for an alternative to the traditional use of Finite Element (FE) methods while preserving the concept of physics-based simulation concepts from the continuum mechanics theory. The main question that we wish to address is therefore:

"*Can we develop an alternative method driven by the balance of linear momentum and capable of patient-specific liver simulations for accurate non-rigid registration between pre-operative and intra-operative data?*"

More precisely, we will consider a subset of Galerkin methods where a system of continuous Partial Differential Equation (PDE)s is converted to a discrete problem of residual minimization over a geometrical discretization of the simulated domain.

Obviously, the elaboration of our candidate methods will be influenced by our four key requirements.

The Galerkin methods considered in this thesis are classified in three branches. The first one, *mesh*-based methods, includes methods where both the integration and the approximation of a field function over the domain is done through a mesh of geometrical elements. When the boundary of this mesh matches the boundary of the simulated domain, and when the approximation of a value within an element depends only on the element's nodes, the result is equivalent to the classical Finite Element method. When the mesh is allowed to overlap boundaries, we fall into the field of Immersed boundary (IB) methods. Here, the simulated domain is said to be *immersed* into a computational background mesh.

The next branch, *meshless* methods, includes methods where both the integration and the approximation are done using numerical analysis of clouds of points. Instead of discretizing the domain with a *mesh* of elements, the interior is simply filled with nodes representing the DOF of the system. The approximation of a value anywhere inside the domain is done by looking at the value of neighboring nodes. Special numerical schemes have to be constructed to integrate a field function over the domain.

The third and final branch are the *hybrid* methods which are a mixture of *mesh*-based methods and *meshless* methods. Usually, a mesh of elements is used for the numerical integration of the field function, whereas clouds of points are used for the approximation.

Our hypothesis is that, given an initial volumetric reconstruction of a liver before surgery, and given partial reconstructions of the same liver surface during the surgery, an IB method, a fully *meshless* method and a *hybrid* method can all be used to deform a biomechanical model and meet our four key requirements.

1.5 Contributions

The contributions made in this research project are divided in five parts:

Contribution 1 Standardization of the parts of the theory of hyper-elasticity in continuum mechanics that were necessary for our simulation framework

- Contribution 1.1 Identification of physical tensors and their simplifications
- Contribution 1.2 Identification of the constitutive laws
- Contribution 1.3 Description of the mathematical tools related to the theory
- Contribution 1.4 Description of the discrete terms that will have to be implemented by our considered numerical methods

Contribution 2 Meshless method

- Contribution 2.1 Analysis of particle-based approximation methods
- Contribution 2.2 Analysis of particle-based integration methods
- Contribution 2.3 Design and implementation of prospective meshless methods
- Contribution 2.4 Identification of the limits and sources of error of these methods

Contribution 3 Hybrid method

- Contribution 3.1 Analysis of particle-based approximation methods coupled with mesh-based integration methods
- Contribution 3.2 Design and implementation of prospective hybrid methods
- Contribution 3.3 Identification of the limits and sources of error of these methods

Contribution 4 Immersed boundary method

- Contribution 4.1 Analysis of current IB methods
- Contribution 4.2 Design and implementation of prospective IB methods
- Contribution 4.3 Identification of the limits and sources of error of these methods

Contribution 5 Non-rigid biomechanical registration framework

- Contribution 5.1 Design and implementation of a non-rigid registration pipeline based on a biomechanical model
- Contribution 5.2 Validation of our implementations in real-life scenarios

1.6 Outline

The outline of the thesis stems directly from our research objectives. We will begin with an introduction of the theory of hyper-elasticity. In chapter 2 we will explicitly state the system of partial differential equations which will be solved in our simulations. It will highlight the discrete terms that will be used to establish the weak formulation of the optimization problem that we wish to implement. It will also put the basis of the mathematical framework and the notation to be used throughout this document. Chapter 3 will describe our analysis of meshless and hybrid methods and their positions with respect to traditional finite element methods. A variant aimed for interactive simulations will be proposed. In chapter 4, we will review immersed-boundary methods. Similarly to the meshless methods, we will propose an immersed-boundary method well adapted to our main objective. Chapter 5 will present a non-rigid registration framework based on a physics-driven iterative closest point algorithm. Validation and experimentation of our considered methods within this framework will be presented. Chapter 6 will close this thesis with a conclusion and a summary of the observations made throughout our work and our recommendations for future research.

Chapter

2

CONTINUUM MECHANICS, FINITE ELEMENTS AND IMPLEMENTATION DESIGN

The numerical models discussed in this thesis are derived from basic elasticity principles that originated from the field of continuum mechanics. This allows us to represent the deformation of an object that undergoes an external force with some well-known mechanical concepts. One of them is the displacement function. For any point inside the simulated object, the displacement function returns a vector between the initial and terminating positions of a deformed point inside the object. After studying different kinds of elastic objects, researchers have

managed to build relationships between a force applied to an object, the resulting deformation, and the material properties (i.e., the resistance of a material against stretch and compression). These relationships are formulated as a function of the derivatives of the displacement function. In order to get an explicit representation of the deformation, a set of Partial Differential Equations (PDEs) must therefore be solved. The displacement function that results from the resolution of the PDEs can be viewed as a balance between the internal elastic forces of the object and the external forces exerted on it. In reality, the internal elastic force corresponds to the potential energy stored by the object from the deformation in order to regain its initial shape once the external force is removed. This potential energy must always be equal to the amount of external energy applied to the object, which corresponds to the equilibrium state of the system. Hence, solving the displacement function from the PDEs will yield the displacement of any point of the object from its initial position at rest to its deformed position once the equilibrium state has been reached.

This chapter presents the mathematical framework behind these basic concepts. It provides a formal derivation of the PDE problem to be solved as well as other important elements associated with the non-linear elasticity theory. This includes the transposition of a system of continuous equations into a discrete problem that can be solved using standard numerical algorithms. This discretization problem will be at the heart of the methods proposed later in this thesis.

Most of the concepts and mathematical equations presented in this chapter are documented in Wriggers (2008) (in English) and André Fortin and Garon (2018) (in French). These books will be extensively referenced throughout the remainder of this document. Because they form the foundations of our models, we have decided to explain in detail the derivation of some of the key equations. We are also presenting this information for reference purposes to facilitate the development of software tools in future research. Moreover, to study advanced numerical approaches and stepping outside of the traditional FE methodology, we implemented these concepts in an advanced simulation software library. Technical details and challenges encountered during the implementation are presented at the end of this chapter.

2.1 Lagrangian description of a deformation

The system of equations that defines a deformation can take two forms: the *Lagrangian description* or its counterpart, the *Eulerian description*. These two representations can be described as follows. For a dimension d, let the set $E = \{\boldsymbol{E}_i \in \mathbb{E}^d\}$ with $i = \{1, \ldots, d\}$ be the orthonormal basis vectors of a Euclidean vector space $\mathbb{E}^d$ centered at the origin O. The simulated object lying in this space is defined as the bounded domain $\Omega \subset \mathbb{E}^d$. We can view this domain as part of the space enclosed by the boundary of the object. By introducing the notion of time, we will say that the state of rest is the shape of the object at the time $t = t_0$ of the simulation, which is, the initial state of the object before any deformation. During the simulation, the applied forces will deform the object, making the space Ω a function of the time. We describe the position vector of a point inside the initial domain as $\boldsymbol{X} \in \Omega(t_0) = \Omega_0$. The coordinates of $\boldsymbol{X}$ are called the *material coordinates* and are the independent variables of the system. To get the position vector of a material point in the deformed domain at a subsequent time t_1, we define the linear map $\boldsymbol{x}_t : \Omega_0 \to \Omega_t$ as the rigid transformation

$$\boldsymbol{x}_t(\boldsymbol{X}) = \boldsymbol{R}_t \cdot \boldsymbol{X} + \boldsymbol{T}_t$$

where $\boldsymbol{R}_t$ is an orthogonal (rotation) matrix and $\boldsymbol{T}_t$ is a translation vector. The vector joining the two positions of the same point is called the displacement vector and is spawned by the vector field

$$\boldsymbol{u}_t(\boldsymbol{X}) = \boldsymbol{x}_t(\boldsymbol{X}) - \boldsymbol{X}$$

This is represented in the three-dimensional diagram provided in figure 2.1.

From now on, subscript t will be dropped with the understanding that $\boldsymbol{x}$ and $\boldsymbol{u}$ are evaluated at some time t. We say that the transformation $\boldsymbol{x}(\boldsymbol{X})$ and the displacement field $\boldsymbol{u}(\boldsymbol{X})$ are formulated in terms of *material coordinates* $(\boldsymbol{X})$. This corresponds to the *Lagrangian description* of the system. Conversely, if the coordinates of $\boldsymbol{x}$, called the *spacial coordinates*, are the independent variables of the system, we then have the *Eulerian description*. With the latter, $\boldsymbol{X}(\boldsymbol{x})$ is the transformation map and $\boldsymbol{U}(\boldsymbol{x})$ is the displacement field. From a theoretical point of view, both descriptions are equivalent. However, from an implementation perspective, it is often more efficient to use the Lagrangian description, which is what we will do from now on.

Now, taking the partial derivative of the transformation $\boldsymbol{x}(\boldsymbol{X})$ with respect to the material coordinates, $\boldsymbol{\nabla}_{\boldsymbol{X}}[\boldsymbol{x}(\boldsymbol{X})] = \boldsymbol{\nabla}_{\boldsymbol{X}}[\boldsymbol{X} + \boldsymbol{u}]$, we get the *deformation gradient*

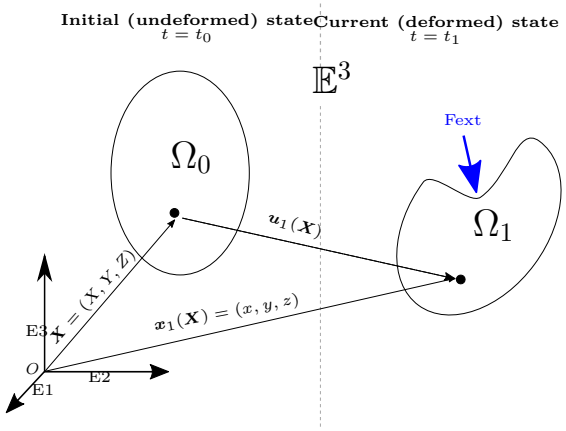


Figure 2.1: Left: initial state of an object. Right: Current state of the same object where an external force is applied to its top boundary.

tensor, which is defined as

$$\boxed{\mathbf{F} = \boldsymbol{I} + \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}} \tag{2.1}$$

where

$$\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u} = \begin{bmatrix} \frac{\partial u}{\partial X} & \frac{\partial u}{\partial Y} & \frac{\partial u}{\partial Z} \\ \frac{\partial v}{\partial X} & \frac{\partial v}{\partial Y} & \frac{\partial v}{\partial Z} \\ \frac{\partial w}{\partial X} & \frac{\partial w}{\partial Y} & \frac{\partial w}{\partial Z} \end{bmatrix} \tag{2.2}$$

is the *displacement gradient tensor*. The *deformation gradient tensor* $\mathbf{F}$ plays a major role in the local description of a deformation. It is used to transform a line $(\boldsymbol{X}, \boldsymbol{X} + d\boldsymbol{X})$ in the initial (undeformed) domain to its deformed shape $(\boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x})$ from the following relation

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

Using Nanson's relation, this equation can be extended to describe the transformation of a small area element dA around a point to its deformed shape da or a small volume element dV to its deformed shape dv. The relations in this case are written as follows

$$da\boldsymbol{n} = JdA\mathbf{F}^{-\intercal}\boldsymbol{N}$$
$$dv = JdV$$

where J is the determinant of $\mathbf{F}$, and $\boldsymbol{n}$ (resp. $\boldsymbol{N}$) is the unit vector normal to da (resp. dA). It is easy to see that, for an incompressible material (a material that

preserves its volume after a deformation), J must be equal to 1. As we will see later, this observation will be useful when simulating incompressible materials as special treatment will be required around J. Figure 2.2 illustrates the transformation of these three quantities.

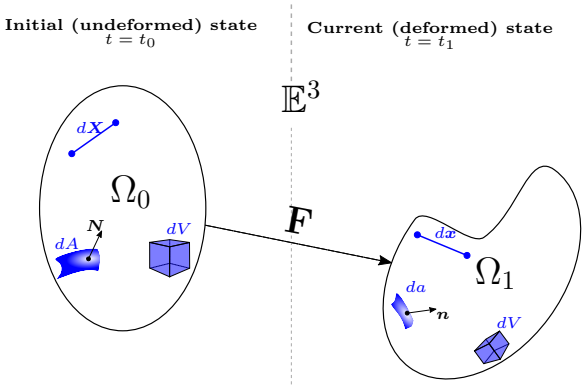


Figure 2.2: The deformation gradient tensor $\mathbf{F}$ transforms any line segment, surface area or volume element from the undeformed configuration to the deformed configuration

The displacement of a local body around a point consists of two parts, a rigid motion (rotation and/or translation) that does not change the body's shape or size; and a pure deformation that does change its shape or its size. Hence, using the polar decomposition, the deformation gradient tensor can be decomposed into a product of an orthogonal tensor $\mathbf{R}_F$ and a positive definite symmetric tensor :

$$\mathbf{F} = \mathbf{R}_F \mathbf{U}_F = \mathbf{V}_F \mathbf{R}_F$$

where $\mathbf{U}_F$ is the right stretch tensor and $\mathbf{V}_F$ is the left stretch tensor (see figure 2.3).

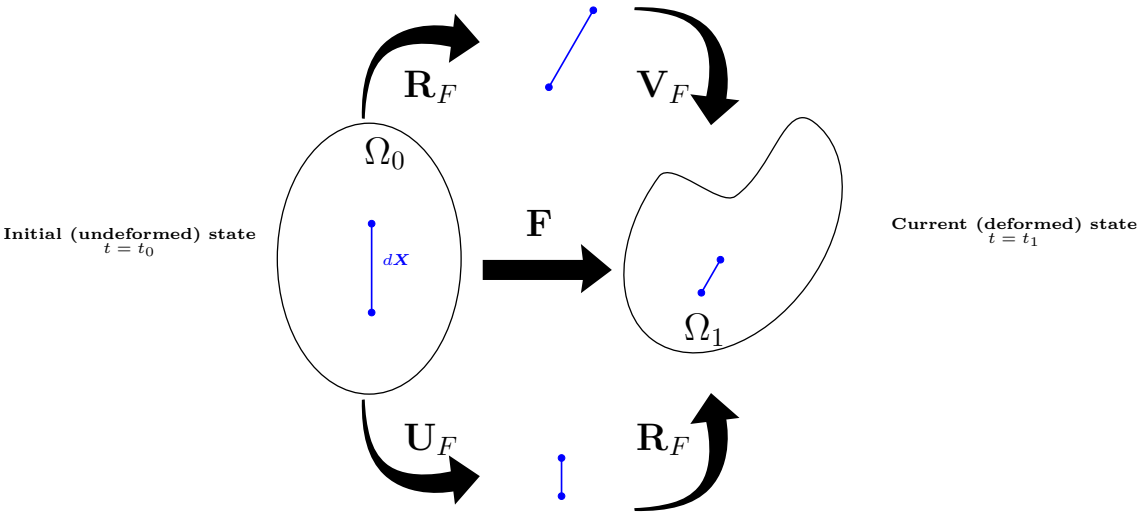


Figure 2.3: The deformation of a small line segment $d\boldsymbol{X}$ consists of a rigid displacement and a stretching deformation

Because a rigid motion does not produce any internal elastic potential energy (rotating or moving a body does not deform its shape), it is useful to isolate the component of the displacement that is associated with the sole deformation. This quantity, called *strain*, gives how much a given displacement differs locally from a rigid body motion. The orthogonality of $\mathbf{R}_F$ implies that $\mathbf{R}_F^\intercal \mathbf{R}_F = \mathbf{I}$. This naturally suggests the definition of a new deformation tensor that takes this orthogonality into account:

$$\mathbf{C} = \mathbf{F}^\intercal \mathbf{F} = \mathbf{U}_F^2 \tag{2.3}$$

where $\mathbf{C}$ is called the *right Cauchy–Green deformation tensor* and is independent of the rotation.

At this point, we have all the elements required to measure the strain. Recall that, for a local segment $d\boldsymbol{X}$ starting from any material point, a strain measurement should give us the amount of stretch this segment has undergone. Taking the difference between the squares of the local segment $d\boldsymbol{X}$ and its deformed counterpart $d\boldsymbol{x}$ yields

$$\begin{aligned} d\boldsymbol{x}^2 - d\boldsymbol{X}^2 &= d\boldsymbol{x} \cdot d\boldsymbol{x} \quad - \quad d\boldsymbol{X} \cdot d\boldsymbol{X} \\ &= d\boldsymbol{X} \cdot \mathbf{C} \cdot d\boldsymbol{X} \quad - \quad d\boldsymbol{X} \cdot d\boldsymbol{X} \\ &= d\boldsymbol{X} \cdot (\mathbf{C} - \mathbf{I}) \cdot d\boldsymbol{X} \\ &= d\boldsymbol{X} \cdot (\mathbf{F}^\intercal \mathbf{F} - \mathbf{I}) \cdot d\boldsymbol{X} \end{aligned}$$

Finally, by taking the symmetric part of the last equation, we define

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^\intercal\mathbf{F} - \mathbf{I}) \tag{2.4}$$

as the *Green-Lagrangian strain tensor* which provides a measure of how much $\mathbf{C}$ differs from $\mathbf{I}$.

2.2 Hyper-elasticity: a relation between strains and stresses

In continuum mechanics, *stress* is a physical quantity that measures the amount of internal forces found in an infinitesimal closed space of continuous material. Analogous to the *strain* which measures the change of shape between two particles of this closed space, the *stress* will give the amount of force exerted between the two. A stress that is represented by either a scalar or a vector is called a *simple stress*. The two most frequent simple stresses are without any doubts the *uniaxial (or normal) stress* and the *shear stress*. The uniaxial stress is usually conceptualized as the ratio between the magnitude of a tractive force applied to a small cross-section of a straight rod (in the direction of the rod), over the area of the cross-section, i.e., $\sigma = \frac{F}{A}$. The shear stress on the other hand can be viewed as a rectangular piece of elastic material where its top face is pulled in a direction parallel to the rectangle, and its bottom face in the opposite direction but also parallel. As the cross-section area grows bigger, it is clear that these two simple stresses will yield a rough approximation: they are in fact only an average of stress across the area. *Combined stress* is a physical quantity that describes multiple simple stresses at any given position. The *Cauchy stress* tensor is one of them, and can be used to express both the shears and uniaxial stresses across an imaginary surface perpendicular to a unit vector $\boldsymbol{n}$. It is written as a traction vector $\boldsymbol{t}$:

$$\boldsymbol{t} = \boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{\sigma}^\intercal \cdot \boldsymbol{n} \tag{2.5}$$

or

$$d\boldsymbol{f}_t = \boldsymbol{t}da = \boldsymbol{\sigma}^\intercal \cdot \boldsymbol{n}da \tag{2.6}$$

where $\boldsymbol{\sigma}$ is the second-order *Cauchy stress tensor* and is defined by :

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \tag{2.7}$$

which is symmetric. Here, σ and τ are, respectively, the uniaxial and shear stresses. Figure 2.4 illustrate the different components of the Cauchy stress tensor.

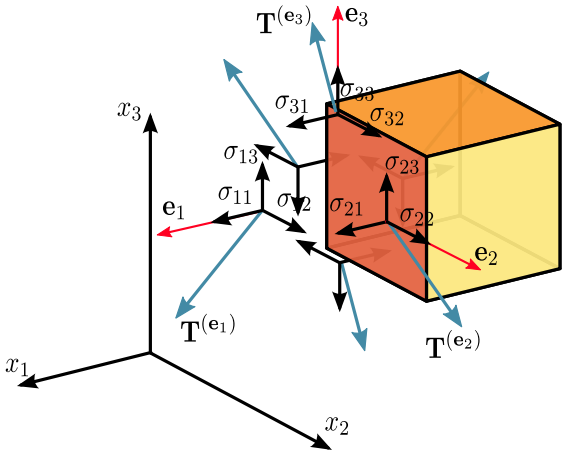


Figure 2.4: Cauchy stress tensor (Sanpaz, Wikipedia)

.

The Cauchy stress tensor relates the force acting on a body to the amount of deformation in the current deformed configuration, which means that it follows an Eulerian description. It is possible to transform the spacial coordinates of the unit vector $\boldsymbol{n}$ in equation (2.5) into material coordinates. Indeed, using again Nanson's formula, we have

$$\begin{aligned} \boldsymbol{\sigma}^\intercal \cdot \boldsymbol{n} da &= \boldsymbol{\sigma}^\intercal \cdot \left(J dA \mathbf{F}^{-\intercal} \cdot \boldsymbol{N} \right) \\ &= \left(J \boldsymbol{\sigma}^\intercal \mathbf{F}^{-\intercal} \right) \cdot \boldsymbol{N} dA \\ &= \mathbf{P} \cdot \boldsymbol{N} dA \\ &= d\boldsymbol{f}_t \end{aligned}$$

where $\mathbf{P} = J\boldsymbol{\sigma}^\intercal \mathbf{F}^{-\intercal}$ is the First Piola-Kirchhoff (FPK) stress tensor. Here, the tensor $\mathbf{P}$ is a two-point tensor, which means that it relates quantities in both spacial and material coordinates. It is therefore not symmetric. Recall any small spacial segment $d\boldsymbol{x}$ can be written in terms of material coordinates using $d\boldsymbol{x} = \mathbf{F} \cdot d\boldsymbol{X}$.

Therefore, the First Piola-Kirchhoff can be converted to its fully Lagrangian form using

$$\mathbf{P} \cdot \boldsymbol{N} dA = d\boldsymbol{f}_t = \mathbf{F} \cdot d\boldsymbol{F}_t$$

and

$$d\boldsymbol{F}_t = \mathbf{F}^{-1}\mathbf{P} \cdot \boldsymbol{N} dA = \mathbf{S} \cdot \boldsymbol{N} dA$$

where $\mathbf{S} = \mathbf{F}^{-1}\mathbf{P}$ is called the Second Piola-Kirchhoff (SPK) stress tensor and is symmetric.

So far, we managed to explicitly expressed a measure of the deformation with respect to the displacement of a point in material coordinates using the Green-Lagrangian strain tensor $\mathbf{E}$. We have also presented a symmetric stress tensor that can relate the amount of stress at any given material coordinates to an internal force vector. We still have to describe the relation between these two kinetic quantities.

A *constitutive relation*, often call a *constitutive model* is an equation specific to a type of material that ties two physical quantities together. In our case, it creates a relation between the amount of deformation undergone by a body (the strain), and the response of the material as internal forces (the stress). A *hyperelastic material* is a type of constitutive relation for which the relation between the strain and the stress is derived from a scalar-valued *strain energy density function*. Let $W(\mathbf{F})$ be this function. For an hyperelastic material, the FPK stress tensor can be calculated using

$$\mathbf{P} = \frac{\partial W}{\partial \mathbf{F}} = \mathbf{F} \cdot \frac{\partial W}{\partial \mathbf{E}} = 2\mathbf{F} \cdot \frac{\partial W}{\partial \mathbf{C}}$$

where we recall that the deformation gradient tensor is $\mathbf{F}$, the Green-Lagrangian strain tensor is $\mathbf{E}$ and the right Cauchy–Green deformation tensor is $\mathbf{C}$. We will say that the FPK stress tensor is energy conjugate to the deformation gradient.

Similarly, the SPK stress tensor is defined as

$$\mathbf{S} = \frac{\partial W}{\partial \mathbf{E}} = \mathbf{F}^{-1} \cdot \frac{\partial W}{\partial \mathbf{F}} = 2\frac{\partial W}{\partial \mathbf{C}}$$

and is energy conjugate to the Green-Lagrangian strain tensor.

The last element of the full Lagrangian description of the internal elastic force associated with an unknown displacement vector $\boldsymbol{u}$ that needs to be defined is the material strain energy density function W. This requires a brief introduction of some basic hyperelastic material models.

2.3 Hyperelastic materials

Hyperelastic materials are constitutive models that are well suited for modeling soft tissues and organs such as a human liver [Marchesseau, Chatelin, and Delingette (2017)]. These models are usually divided in two categories. The first one, the materials that are based on phenomenological descriptions, are those that can be classified based on the response or behavior of the object that is observed just after a known force has been applied. Hence, for this category, the modelization is built using empirical methods. Good examples are the Saint-Venant-Kirchhoff material, the Ogden material [R. W. Ogden (1973)] and the Mooney-Rivlin material [Mooney (1940); Rivlin (1948)]. The other category is the mechanistic models. In this case modelization is done from the underlying structure of the material, usually extracted at a molecular level. The Neo-Hookean material [R W Ogden (2013)] and the Arruda–Boyce material [Arruda and Boyce (1993)] fall in this category.

In this thesis, we have implemented two constitutive models, namely the Saint-Venant-Kirchhoff and the Neo-Hookean. Each of them will be described in the following subsections. But first, we need to describe two tensor operators which will be required for the derivation of the strain energy density function.

Let $\mathbf{A}$ be a second-order tensor. We define the operator $\mathbf{A} \otimes \mathbf{A}$ and its symmetric part $\mathbf{A} \bar{\otimes} \mathbf{A}$ as the fourth order tensors

$$\mathbf{A} \otimes \mathbf{A} = (\mathbf{A}_{ik}\mathbf{A}_{jl})\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l$$
$$\mathbf{A} \bar{\otimes} \mathbf{A} = \frac{1}{2}(\mathbf{A}_{ik}\mathbf{A}_{jl} + \mathbf{A}_{il}\mathbf{A}_{jk})\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l$$

Using these operators, we can build the following derivation with respect to the Green-Lagrange strain tensor $\mathbf{E}$:

$$\begin{aligned}
\left(\frac{\partial \operatorname{tr}(\mathbf{E})}{\partial \mathbf{E}}\right) &= \boldsymbol{I} \\
\left(\frac{\partial \mathbf{E}}{\partial \mathbf{E}}\right) &= \boldsymbol{I} \otimes \boldsymbol{I} \overset{\text{sym.}}{=} \boldsymbol{I} \bar{\otimes} \boldsymbol{I} \\
\left(\frac{\partial \mathbf{E}^{-1}}{\partial \mathbf{E}}\right) &= -\mathbf{E}^{-1} \otimes \mathbf{E}^{-1} \overset{\text{sym.}}{=} -\mathbf{E}^{-1} \bar{\otimes} \mathbf{E}^{-1} \\
\left(\frac{\partial J}{\partial \mathbf{E}}\right) &= 2\left(\frac{\partial J}{\partial \mathbf{C}}\right) = 2\left(\frac{1}{2}J\mathbf{C}^{-1}\right) = J\mathbf{C}^{-1} \\
\left(\frac{\partial f(\mathbf{E})\mathbf{T}(\mathbf{E})}{\partial \mathbf{E}}\right) &= \mathbf{T}(\mathbf{E}) \otimes \frac{\partial f(\mathbf{E})}{\partial \mathbf{E}} + f(\mathbf{E})\frac{\partial \mathbf{T}(\mathbf{E})}{\partial \mathbf{E}}
\end{aligned}$$

where $\text{tr}(\mathbf{E}) = \mathbf{E}_{00} + \mathbf{E}_{11} + \mathbf{E}_{22}$, $J = det(\mathbf{F})$, $\mathbf{C}$ is the right Cauchy-Green strain tensor, $f(\mathbf{E})$ is a scalar-valued function and $\mathbf{T}(\mathbf{E})$ is a second order tensor valued function.

In some cases, the strain energy density can be written as a function of the three invariant (I_1, I_2 and I_3) of the right Cauchy-Green strain tensor $\mathbf{C}$. We define them here with their derivatives with respect to $\mathbf{C}$ and the eigenvalues (L_1, L_2 and L_3) of $\mathbf{C}$.

$$
\begin{array}{lcl|lcl|lcl}
I_1 & = & \text{tr}(\mathbf{C}) & I_2 & = & \frac{1}{2}\left[(\text{tr}(\mathbf{C}))^2 - \text{tr}(\mathbf{C}^2))\right] & I_3 & = & det\ \mathbf{C} = J^2 \\
 & & & & = & \frac{1}{2}\left[(I_1^2 - \mathbf{C} : \mathbf{C})\right] & & & \\
\frac{\partial I_1}{\partial \mathbf{C}} & = & \boldsymbol{I} & \frac{\partial I_2}{\partial \mathbf{C}} & = & I_1 \boldsymbol{I} - \mathbf{C} & \frac{\partial I_3}{\partial \mathbf{C}} & = & I_3 \mathbf{C}^{-1} \\
\frac{\partial I_1}{\partial L_i} & = & 1 & \frac{\partial I_2}{\partial L_i} & = & I_1 - L_i & \frac{\partial I_3}{\partial L_i} & = & \frac{I_3}{L_i}
\end{array}
$$

Using the eigenvectors $\boldsymbol{N}_i$ of $\mathbf{C}$, we also note the following important relations:

$$
\boldsymbol{I} = \sum_{i=1}^{3} \boldsymbol{N}_i \otimes \boldsymbol{N}_i \qquad \mathbf{C} = \sum_{i=1}^{3} L_i \boldsymbol{N}_i \otimes \boldsymbol{N}_i \qquad \mathbf{C}^{-1} = \sum_{i=1}^{3} \frac{1}{L_i} \boldsymbol{N}_i \otimes \boldsymbol{N}_i
$$

2.3.1 St-Venant-Kirchhoff

The Saint Venant–Kirchhoff (SVK) material is probably the simplest of hyperelastic models. It is a direct extension of Hooke's law in linear elasticity. It also is a homogeneous and isotropic material, which means that its resistance to deformation is the same on every point, and for every direction. Let $\mu = \frac{E}{2(1+\nu)}$ and $\lambda = \frac{E\nu}{((1+\nu)(1-2\nu))}$ be the Lamé's first and second parameters, respectively, and where E is the Young's modulus and ν is the Poisson's ratio. The strain density energy function of the SVK material is defined as

$$
W = \frac{\lambda}{2}[\text{tr}(\mathbf{E})]^2 + \mu\, \text{tr}(\mathbf{E}^2) \tag{2.8}
$$

The first and second derivatives of W with respect to the Green-Lagrangian strain tensor yield the second order SPK stress tensor and the fourth order material

tensor, respectively,

$$\begin{aligned}\mathbf{S} = \left.\frac{\partial W}{\partial \mathbf{E}}\right|_{\boldsymbol{X}} &= \lambda \operatorname{tr}(\mathbf{E})\boldsymbol{I} + 2\mu\mathbf{E} \\ \mathbb{C} = \left.\frac{\partial \mathbf{S}}{\partial \mathbf{E}}\right|_{\boldsymbol{X}} &= \lambda(\boldsymbol{I}\otimes\boldsymbol{I}) + 2\mu(\boldsymbol{I}\bar{\underline{\otimes}}\boldsymbol{I})\end{aligned}$$

When only very small deformations are considered, the infinitesimal strain theory can be used and the non-linear Green-Lagrangian strain tensor $\mathbf{E} = \frac{1}{2}(\mathbf{F}^{\intercal}\mathbf{F} - \mathbf{I})$ can be approximated by

$$\boldsymbol{\epsilon} = \frac{1}{2}(\boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{u}^{\intercal} + \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{u})$$

where $\boldsymbol{\epsilon}$ is called the *small strain tensor*. Substituting $\boldsymbol{\epsilon}$ into equation (2.8) gives the linear Cauchy elastic stress tensor:

$$\begin{aligned}\boldsymbol{\sigma} &= \lambda \operatorname{tr}(\boldsymbol{\epsilon})\boldsymbol{I} + 2\mu\boldsymbol{\epsilon} \\ \mathbb{C} &= \lambda(\boldsymbol{I}\otimes\boldsymbol{I}) + 2\mu(\boldsymbol{I}\bar{\underline{\otimes}}\boldsymbol{I})\end{aligned}$$

which is the three-dimensional representation of the Hooke's law and will be discussed further in chapter 3.

The Saint Venant–Kirchhoff material is simple to implement, computationally efficient and numerically robust as it contains no asymptote, hence no undermined points. It is usually used for deformations having large displacements and rotation, but small strains. For large strains, it is usually not accurate enough. It is known to experience major deficiencies when the compression of a body approaches a volume of zero (i.e., when $J \to 0$). In this extreme case, the stress will also approaches zero instead of infinity [Wriggers (2008)].

2.3.2 Neo-hookean

The Neo-Hookean model can be seen as a non-linear extension of the SVK approach. In fact, the relationship between the strain and the stress is initially linear. At a certain point, however, the stress-strain curve will reach a plateau. The model was proposed by Ronald Rivlin [Rivlin (1948)] and does not exhibit the deficiencies of the SVK when a large compression arises. The strain energy density function reads as

$$
\begin{aligned}
W &= \mu \operatorname{tr}(\mathbf{E}) - \mu \ln(J) + \frac{\lambda}{2}(\ln(J))^2 \\
&= \frac{\mu}{2}(\operatorname{tr}(\mathbf{C}) - 3) - \mu \ln(J) + \frac{\lambda}{2}(\ln\ J)^2
\end{aligned} \tag{2.9}
$$

The first and second derivatives of W with respect to the Green-Lagrangian strain tensor yield the second order SPK stress tensor and the fourth order material tensor, respectively, which can also be written with respect to the invariant of $\mathbf{C}$:

$$
\begin{aligned}
\mathbf{S} = \left.\frac{\partial W}{\partial \mathbf{E}}\right|_{\boldsymbol{X}} = 2\left.\frac{\partial W}{\partial \mathbf{C}}\right|_{\boldsymbol{X}} &= \mu \left[\frac{\partial \operatorname{tr}(\mathbf{C})}{\partial \mathbf{C}}\right] - 2\mu \frac{1}{J}\left[\frac{\partial J}{\partial \mathbf{C}}\right] + 2\lambda \frac{1}{J}(\ln\ J)\left[\frac{\partial J}{\partial \mathbf{C}}\right] \\
&= \mu \left[\frac{\partial I_1}{\partial \mathbf{C}}\right] - 2\mu \frac{1}{J}\left[\frac{\partial \sqrt{I_3}}{\partial \mathbf{C}}\right] + 2\lambda \frac{1}{J}(\ln\ J)\left[\frac{\partial \sqrt{I_3}}{\partial \mathbf{C}}\right] \\
&= \mu \boldsymbol{I} - \mu \mathbf{C}^{-1} + \lambda(\ln\ J)\mathbf{C}^{-1} \\
&= \mu(\boldsymbol{I} - \mathbf{C}^{-1}) + \lambda(\ln\ J)\mathbf{C}^{-1}
\end{aligned}
$$

and

$$
\begin{aligned}
\mathbb{C} = \left.\frac{\partial \mathbf{S}}{\partial \mathbf{E}}\right|_{\boldsymbol{X}} = 2\left.\frac{\partial \mathbf{S}}{\partial \mathbf{C}}\right|_{\boldsymbol{X}} &= -2\mu \left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right] + 2\lambda \left(\mathbf{C}^{-1} \otimes \frac{\partial(\ln\ J)}{\partial \mathbf{C}} + (\ln\ J)\left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right]\right) \\
&= -2\mu \left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right] + 2\lambda \left(\frac{1}{J}\mathbf{C}^{-1} \otimes \left[\frac{\partial J}{\partial \mathbf{C}}\right] + (\ln\ J)\left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right]\right) \\
&= -2\mu \left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right] + 2\lambda \left(\frac{1}{2}(\mathbf{C}^{-1} \otimes \mathbf{C}^{-1}) + (\ln\ J)\left[\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right]\right) \\
&= \lambda(\mathbf{C}^{-1} \otimes \mathbf{C}^{-1}) + 2(\mu - \lambda \ln J)(\mathbf{C}^{-1} \underline{\bar{\otimes}} \mathbf{C}^{-1})
\end{aligned}
$$

Here, the density of strain energy is undetermined when $J \leq 0$. While reaching a null or negative jacobian looks counterintuitive, some numerical methods can produce solutions where elements of the mesh are inverted, or badly compressed. Special attention is therefore required when the Neo-Hookean material is used. These considerations will be discussed in chapters 4 and 5.

2.4 Balance equations

To be physically accurate, the simulation of a deformable object requires that a set of rules, or laws, be defined on the different kinematics quantities of the system. We have to keep in mind that at the beginning of the simulation, only the description of the initial shape of the object and the amount of external forces applied to it are known. The shape of the object at an upcoming time t is what we seek and is therefore unknown. The general idea behind these rules is to impose the resolution of a set of equations on the unknown shape of the object at each interval t in order to control or restrain the change (or rate of change) of the simulated kinematics quantities over time. In continuum mechanics, these rules are known as the *balance of mass*, the *balance of linear momentum* (Cauchy's first law of motion), the *balance of angular momentum* (Cauchy's second law of motion) and the *balance of energy* (first law of thermodynamics).

In the last two sections, we have defined kinematic quantities that describe the amount of deformation (strain) and the amount of internal force (stress) of a deformed state of a body at a given time. The remaining quantities are related to the motion of the body over time. The density function $\rho(\boldsymbol{X}, t) = \rho_t$ gives a measure of the mass per unit of volume at the material coordinates $\boldsymbol{X}$ (initial undeformed state). The velocity $\boldsymbol{v}(\boldsymbol{X}, t) = \frac{d}{dt}\boldsymbol{x}_t(\boldsymbol{X}) = \dot{\boldsymbol{x}}$ gives the rate of change of the position of an infinitesimal particle located at $\boldsymbol{X}$ and at the time t. Similarly, the acceleration $\boldsymbol{a}(\boldsymbol{X}, t) = \frac{d}{dt}\boldsymbol{v}(\boldsymbol{X}, t) = \ddot{\boldsymbol{x}}$ describes the rate of change of the velocity. The rules applied to these quantities can be written as follows:

Law	*Lagrangian description*	*Eulerian description*
Balance of mass	$\rho_0 - J\rho_t = 0$	$\dot{\rho} + \rho\boldsymbol{\nabla} \cdot \boldsymbol{v}$
Balance of linear momentum	$\rho_0\ddot{\boldsymbol{x}} - \nabla_{\boldsymbol{X}} \cdot \mathbf{P} = \rho_0\boldsymbol{b}$	$\rho\ddot{\boldsymbol{x}} - \nabla_{\boldsymbol{x}} \cdot \boldsymbol{\sigma} = \rho\boldsymbol{b}$
Balance of angular momentum	$\mathbf{P}\mathbf{F}^\intercal = \mathbf{F}\mathbf{P}^\intercal$	$\boldsymbol{\sigma} = \boldsymbol{\sigma}^\intercal$
Balance of energy	$\rho_0\dot{u} = \mathbf{S} \cdot \dot{\mathbf{E}} - \nabla_{\boldsymbol{X}}\boldsymbol{Q} + \rho_0 R$	$\rho\dot{u} = \boldsymbol{\sigma} \cdot \mathbf{d} - \nabla_{\boldsymbol{x}}\boldsymbol{q} + \rho r$

where the heat flux vector $\boldsymbol{q}$ (respectively $\boldsymbol{Q}$) and the heat source r (respectively R) has been introduced for the Eulerian (respectively Lagrangian) configuration. The vector $\boldsymbol{b}$ represents the external body forces that act everywhere within the domain. A good example of this could be the gravitational force exerted on the simulated object.

In this thesis, simulated materials will be considered of constant density throughout the simulation, and represented only by the symmetric Second Piola-Kirchhoff (SPK) stress tensor, hence we will assume that the conservation of mass and the angular momentum (first and third rules) are always respected. We will also only

consider simulations without heat. Hence, from the symmetry of the stress tensor, it can be shown that the balance of energy is equivalent to the balance of linear momentum [André Fortin and Garon (2018)]. The only rule remaining is therefore the balance of linear momentum and will be the one we will focus on.

2.5 Lagrangian description of the weak formulation

The balance of linear momentum rule presented in the preceding section can be viewed as a nonlinear initial boundary problem where a system of partial differential equations subject to boundary conditions has to be solved. To solve the system, the general idea is to replace the exact solution $\boldsymbol{u}$ of the system by an approximation $\boldsymbol{u}^h$ which, in the case of Finite Element (FE) methods, usually consists of a set of piece-wise continuous approximation functions. Inserting $\boldsymbol{u}^h$ in the balance equation yields

$$\rho_0 \ddot{\boldsymbol{x}} - \nabla_{\boldsymbol{X}} \cdot \mathbf{P}(\boldsymbol{u}^h) - \rho_0 \boldsymbol{b} = R \tag{2.10}$$

where R is a residual from the approximation error. The error can be minimized by multiplying the residual by a vector-valued weight function $\boldsymbol{w}$ and by integrating the resulting equation over the initial domain. The function $\boldsymbol{w}$ is usually called a *test* function, and is zero on the boundary region where an essential boundary condition (see below) is imposed. The resulting equation is called the *weak formulation* of equation (2.10), and the residual R is said to be minimized in a *weak sense*. The weak formulation is therefore given by:

$$G(\boldsymbol{u}^h, \boldsymbol{w}) - L(\boldsymbol{w}) =$$

$$\overbrace{\underbrace{\int_{\Omega_0} \mathbf{P} : \nabla_{\boldsymbol{X}} \boldsymbol{w} dv}_{\text{Internal virtual work}}}^{G(\boldsymbol{u}^h, \boldsymbol{w})} - \overbrace{\underbrace{\int_{\Omega_0} \rho_0 \ddot{\boldsymbol{x}} \cdot \boldsymbol{w} dv}_{\text{Virtual inertia}} - \underbrace{\int_{\Omega_0} \rho_0 \boldsymbol{b} \cdot \boldsymbol{w} dv - \int_{\Gamma_t} \boldsymbol{t} \cdot \boldsymbol{w} ds}_{\text{Virtual load work}}}^{L(\boldsymbol{w})} \tag{2.11}$$

$$= 0$$

subject to

$$\boldsymbol{u}^h = \boldsymbol{u}_g \text{ on } \Gamma_u$$

where $\Gamma = \Gamma_t \cup \Gamma_u$ is the boundary of Ω_0, $\boldsymbol{N}(\boldsymbol{X}) : \boldsymbol{X} \in \Gamma_t$ is the unit vector normal to the boundary. The imposed traction $\boldsymbol{t}$ and displacement $\boldsymbol{u}_g$ are respectively the *natural* and *essential* boundary conditions.

Because the stress tensor $\mathbf{P}$ is not guaranteed to be symmetrical, we can convert it to its Second Piola-Kirchhoff form with $\mathbf{P} = \mathbf{FS}$. The internal virtual work term then becomes:

$$\int_{\Omega_0} \mathbf{P} : \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} dv = \boxed{\int_{\Omega_0} (\mathbf{FS}) : \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} dv} \tag{2.12}$$

We can simplify this first term further by using the fact that the double dot product between a symmetrical tensor and an anti-symmetrical tensor is zero. Hence, if $\mathbf{B} = \mathbf{B}_S + \mathbf{B}_A$ where $\mathbf{B}_S$ and $\mathbf{B}_A$ are respectively the symmetrical and antisymmetric parts of $\mathbf{B}$, then if $\mathbf{A}$ is symmetric we have $\mathbf{A} : \mathbf{B} = \mathbf{A} : (\mathbf{B}_S + \mathbf{B}_A) = \mathbf{A} : \mathbf{B}_S + \mathbf{A} : \mathbf{B}_A = \mathbf{A} : \mathbf{B}_S + 0 = \mathbf{A} : \mathbf{B}_S$. This yield

$$\begin{aligned} \int_{\Omega_0} (\mathbf{FS}) : \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} dv &= \int_{\Omega_0} \mathbf{S} : \mathbf{F}^\intercal \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} dv \\ &= \int_{\Omega_0} \mathbf{S} : \frac{1}{2} (\mathbf{F}^\intercal \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} + \boldsymbol{\nabla}_{\boldsymbol{X}}^\intercal \boldsymbol{w} \mathbf{F}) dv \\ &= \boxed{\int_{\Omega_0} \mathbf{S} : \delta \mathbf{E} dv} \end{aligned} \tag{2.13}$$

where $\delta\mathbf{E}$ is the symmetrical part of $\mathbf{F}^\intercal \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w}$. We can also note that $\delta\mathbf{E} = D\mathbf{E} \cdot \boldsymbol{w} = \frac{\partial \mathbf{E}}{\partial \boldsymbol{u}} \cdot \boldsymbol{w}$, which is why it is often called the variation of the Green-Lagrange strain tensor.

Both equations (2.12) and (2.13) are equivalent. But we are interested in deriving both of them as they lead to two different implementations of an equivalent finite element code. We will address the internal virtual work, virtual inertia and virtual load work components of equation (2.11) later in the next section. This will be done using a discretization of the weakened continuous domain.

2.6 Discretization of the weak formulation

So far, we have managed to reduce the order of the PDEs by one and boundary terms to impose traction conditions on our system are now explicitly formulated. However, the different integration terms in equation (2.11) over a continuous 2D or 3D field of arbitrary shapes, such as a liver, is often not possible: we do not have a numerical description of its geometry. A useful trick to overcome this problem, which forms the basis of the Finite Element Method (FEM), consists in splitting the global shape of the object into a series of smaller geometrical elements, such as triangles and quads in two dimensions, or tetrahedrons and hexahedrons in three dimensions. This concept is called *discretization*.

The set of elements, often call the *mesh*, or the *tesselation*, fills the entire interior domain of the global shape. The continuous terms in equation (2.11) are replaced by a finite sum of piece-wise continuous terms on each element, and the field functions $\boldsymbol{u}^h$ and $\boldsymbol{w}^h$ by their approximation $\boldsymbol{u}_e^h$ and $\boldsymbol{w}_e^h$ restricted inside an element e. Hence, approximating the value of a field function at any given position $\boldsymbol{X}$ within the initial domain Ω_0 results in finding the element e containing $\boldsymbol{X}$, and computing

$$\boldsymbol{u}_e^h(\boldsymbol{X}) = \sum_{i=0}^{n_e-1} N_i(\boldsymbol{X})\boldsymbol{u}_i \tag{2.14}$$

where n_e is the number of nodes in the element e, $\boldsymbol{u}_i$ is the field value at node i, and $N_i(\boldsymbol{x}) : \Omega_0 \to \Re$ is the i$^{\text{th}}$ shape function of the element.

Likewise, the integration of a field function over Ω_0 is approximated by a finite sum of integral over element domains Ω_e:

$$\int_{\Omega_0} f(\boldsymbol{X}) d_{\Omega_0} \approx \sum_e \int_{\Omega_e} f(\boldsymbol{X}) d_{\Omega_e}$$

Since this piece-wise integration will be used to construct an algebraic system of equations, we use a notation similar to that proposed in Wriggers (2008) and introduce the operator $\biguplus_e$ which denotes that an assembly process takes place. For example, let's suppose that $\Omega_0 \in \mathbb{E}^3$, and that $0 \leq (i, j) \leq n_e$ with n_e being the number of geometric nodes in element e. Let's also define $NUM(i)$ as the numbering method which states that the i^{th} node of an element represents the $NUM(i)$, $NUM(i)+1$ and $NUM(i)+2$ unknowns of the algebraic system (often called the degrees of freedom). The following equation

$$\boldsymbol{R} = \biguplus_{e} \sum_{i=0}^{n_e-1} \underbrace{\int_{\Omega_e} \boldsymbol{f}(\boldsymbol{X}) d_{\Omega_e}}_{\boldsymbol{R}_i}$$

with $\boldsymbol{f} : \Omega_0 \rightarrow \Re^3$ indicates that, for each element e, the value of the integral $\boldsymbol{R}_i$ is accumulated into the entries $NUM(i)$, $NUM(i)+1$ and $NUM(i)+2$ of the vector $\boldsymbol{R}$. Similarly, the equation

$$\boldsymbol{K} = \biguplus_{e} \sum_{(i,j)}^{n_e-1} \sum \underbrace{\int_{\Omega_e} \mathbf{A}(\boldsymbol{X}) d_{\Omega_e}}_{\boldsymbol{K}_{ij}}$$

with $\boldsymbol{A} : \Omega_0 \rightarrow \Re^{3\times3}$ indicates that, for each element e, the value of the integral $\boldsymbol{K}_{ij}$ is accumulated into the 3×3 sub-matrix at $(NUM(i), NUM(j))$ of the matrix $\boldsymbol{K}$.

The approximation of the displacement gradient at any point inside the element e can be obtained with:

$$\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}_e^h = \sum_{i=0}^{n_e-1} \boldsymbol{u}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i \tag{2.15}$$

where $\boldsymbol{\nabla}_{\boldsymbol{X}} N_i = [\frac{\partial N_i}{\partial X}, \frac{\partial N_i}{\partial Y}, \frac{\partial N_i}{\partial Z}]^\intercal$ is the gradient of the i^{th} shape function with respect to material coordinates.

From this, we can approximate the deformation gradient of any point inside the element with:

$$\begin{aligned}
\mathbf{F}_e &= \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}_e + \boldsymbol{I} = \sum_{i=0}^{n_e-1} \boldsymbol{u}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i + \boldsymbol{I} \\
&= \sum_{i=0}^{n_e-1} \boldsymbol{x}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i - \underbrace{\sum_{i=0}^{n_e-1} \boldsymbol{X}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i}_{\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{X}_e = \boldsymbol{I}} + \boldsymbol{I} \\
&= \sum_{i=0}^{n_e-1} \boldsymbol{x}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i
\end{aligned}$$

Finally, the variation of the Green-Lagrange strain tensor in equation (2.13) can be expressed as:

$$\begin{aligned}\delta\mathbf{E}_e &= \frac{1}{2}(\mathbf{F}_e^\intercal \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} + \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w}^\intercal \mathbf{F}_e) \\ &= \frac{1}{2}\sum_{i=0}^{n_e-1}\sum_{j=0}^{n-1}(\boldsymbol{x}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i)^\intercal(\boldsymbol{w}_j \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_j) + (\boldsymbol{w}_j \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_j)^\intercal(\boldsymbol{x}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i)\end{aligned}$$

Using the Voigt notation, we can rewrite the last equation using the following matrix form:

$$\begin{aligned}\delta\tilde{\mathbf{E}}_e &= \begin{bmatrix}\delta E_{00}\\ \delta E_{11}\\ \delta E_{22}\\ 2\delta E_{01}\\ 2\delta E_{12}\\ 2\delta E_{02}\end{bmatrix} \\ &= \sum_{i=0}^{n_e-1}\begin{bmatrix} F_{00}N_{i,X} & F_{10}N_{i,X} & F_{20}N_{i,X}\\ F_{01}N_{i,Y} & F_{11}N_{i,Y} & F_{21}N_{i,Y}\\ F_{02}N_{i,Z} & F_{12}N_{i,Z} & F_{22}N_{i,Z}\\ F_{00}N_{i,Y}+F_{01}N_{i,X} & F_{10}N_{i,Y}+F_{11}N_{i,X} & F_{20}N_{i,Y}+F_{21}N_{i,X}\\ F_{01}N_{i,Z}+F_{02}N_{i,Y} & F_{11}N_{i,Z}+F_{12}N_{i,Y} & F_{21}N_{i,Z}+F_{22}N_{i,Y}\\ F_{00}N_{i,Z}+F_{02}N_{i,X} & F_{01}N_{i,Z}+F_{12}N_{i,X} & F_{20}N_{i,Z}+F_{22}N_{i,X}\end{bmatrix}\boldsymbol{w}_i \\ &= \sum_{i=0}^{n_e-1}\mathbf{B}_i\boldsymbol{w}_i\end{aligned}$$

Note that, using $\mathbf{F} = \boldsymbol{I} + \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{u}$ the matrix $\mathbf{B}_i$ can be reformulated with:

$$\mathbf{B}_i = \underbrace{\begin{bmatrix} N_{i,X} & 0 & 0 \\ 0 & N_{i,Y} & 0 \\ 0 & 0 & N_{i,Z} \\ N_{i,Y} & N_{i,X} & 0 \\ 0 & N_{i,Z} & N_{i,Y} \\ N_{i,Z} & 0 & N_{i,X} \end{bmatrix}}_{\mathbf{B}_{\mathrm{L}i}} + \underbrace{\begin{bmatrix} u_{0,X}N_{i,X} & u_{1,X}N_{i,X} & u_{2,X}N_{i,X} \\ u_{0,Y}N_{i,Y} & u_{1,Y}N_{i,Y} & u_{2,Y}N_{i,Y} \\ u_{0,Z}N_{i,Z} & u_{1,Z}N_{i,Z} & u_{2,Z}N_{i,Z} \\ u_{0,X}N_{i,Y} + u_{0,Y}N_{i,X} & u_{1,X}N_{i,Y} + u_{1,Y}N_{i,X} & u_{2,X}N_{i,Y} + u_{2,Y}N_{i,X} \\ u_{0,Y}N_{i,Z} + u_{0,Z}N_{i,Y} & u_{1,Y}N_{i,Z} + u_{1,Z}N_{i,Y} & u_{2,Y}N_{i,Z} + u_{2,Z}N_{i,Y} \\ u_{0,X}N_{i,Z} + u_{0,Z}N_{i,X} & u_{0,Y}N_{i,Z} + u_{1,Z}N_{i,X} & u_{2,X}N_{i,Z} + u_{2,Z}N_{i,X} \end{bmatrix}}_{\mathbf{B}_{\mathrm{NL}i}} \tag{2.16}$$

where $\mathbf{B}_{\mathrm{L}i}$ and $\mathbf{B}_{\mathrm{NL}i}$ represent respectively the linear and non-linear parts of the strain variation.

Internal Virtual Work:

We now have all discretized quantities required to describe the internal virtual work component of equation (2.11) using either the formulation of equation (2.12) or that of equation (2.13).

For a formulation using equation (2.12), we have

$$\begin{aligned} \int_{\Omega_0} \mathbf{FS} : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} dv &= \biguplus_e \sum_{i=0}^{n_e-1} \int_{\Omega_e} \mathbf{F}_e\mathbf{S}_e : (\boldsymbol{w}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i) dv \\ &= \biguplus_e \sum_{i=0}^{n_e-1} \int_{\Omega_e} (\mathbf{F}_e\mathbf{S}_e \cdot \boldsymbol{\nabla}_{\boldsymbol{X}} N_i) \cdot \boldsymbol{w}_i dv \\ &= \biguplus_e \sum_{i=0}^{n_e-1} \boxed{\int_{\Omega_e} (\mathbf{F}_e\mathbf{S}_e \cdot \boldsymbol{\nabla}_{\boldsymbol{X}} N_i) dv} \cdot \boldsymbol{w}_i \\ &= \biguplus_e \sum_{i=0}^{n_e-1} \boldsymbol{R}_i(\boldsymbol{u}_e) \cdot \boldsymbol{w}_i \\ &= \boldsymbol{R}(\boldsymbol{u}) \cdot \boldsymbol{w} \end{aligned} \tag{2.17}$$

where we used the fact that, for a second order tensor $\mathbf{A}$ and two vectors $\boldsymbol{v}$ and

$\boldsymbol{w}$, the following equality holds : $(\boldsymbol{v} \otimes \boldsymbol{w}) : \mathbf{A} = \mathbf{A} : (\boldsymbol{v} \otimes \boldsymbol{w}) = (\mathbf{A} \cdot \boldsymbol{w}) \cdot \boldsymbol{v}$. Here, $\boldsymbol{R}(\boldsymbol{u})$ denotes the internal force of the body.

Alternatively, if the formulation of equation (2.13) is used, we then have:

$$\begin{aligned}
\int_{\Omega_0} \mathbf{S} : \delta\mathbf{E} dv &= \biguplus_e \int_{\Omega_e} \mathbf{S}_e : \delta\mathbf{E}_e dv \\
&= \biguplus_e \int_{\Omega_e} \delta\tilde{\mathbf{E}}_e^\intercal \tilde{\mathbf{S}}_e dv \\
&= \biguplus_e \sum_{i=0}^{n_e-1} \boldsymbol{w}_i^\intercal \boxed{\int_{\Omega_e} \mathbf{B}_i^\intercal \tilde{\mathbf{S}}_e dv} \\
&= \biguplus_e \sum_{i=0}^{n_e-1} \boldsymbol{w}_i^\intercal \boldsymbol{R}_i(\boldsymbol{u}_e) \\
&= \boldsymbol{R}(\boldsymbol{u}) \cdot \boldsymbol{w}
\end{aligned} \tag{2.18}$$

where $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{S}}$ are the Voigt representation of $\mathbf{E}$ and $\mathbf{S}$, respectively.

Virtual Inertia:

Using the same approach for the virtual inertia component, we can approximate the acceleration field $\boldsymbol{a}_e(\boldsymbol{X}, t)$ within an element with

$$\ddot{\boldsymbol{x}}_e = \sum_{i=0}^{n_e-1} N_i(\boldsymbol{X}) \ddot{\boldsymbol{x}}_i$$

The discretization of the virtual inertia term in equation (2.11) then becomes

$$\begin{aligned}
\int_{\Omega_0} \rho_0 \ddot{\boldsymbol{x}} \cdot \boldsymbol{w} dv &= \biguplus_e \int_{\Omega_e} \boldsymbol{w}^\intercal \rho_0 \ddot{\boldsymbol{x}}_e dv \\
&= \biguplus_e \sum_{(i,j)}^{n_e-1} \sum \boldsymbol{w}_i^\intercal \left[\int_{\Omega_e} \rho_0 N_i N_j \boldsymbol{I} dv \right] \ddot{\boldsymbol{x}}_j \\
&= \biguplus_e \sum_{(i,j)}^{n_e-1} \sum \boldsymbol{w}_i^\intercal \boldsymbol{M}_{ij} \ddot{\boldsymbol{x}}_j \\
&= \boldsymbol{M} \ddot{\boldsymbol{x}} \cdot \boldsymbol{w}
\end{aligned} \tag{2.19}$$

where the matrix $\boldsymbol{M}$ is the mass matrix of the system and is constant throughout the simulation.

Virtual load work:

The discretization of the traction and body force terms can also be formulated in a similar way. However, the surface integration of the traction term will be translated into the integration over the element's boundary faces. Let $\mathfrak{F}$ be the set of faces lying on the natural boundary Γ_t, and $e_f \in \mathfrak{F}$ the face of an element e. We have

$$\begin{aligned}\int_{\Omega_0} \rho_0 \boldsymbol{b} \cdot \boldsymbol{w} dv + \int_{\Gamma_t} \boldsymbol{t} \cdot \boldsymbol{w} ds &= \biguplus_e \sum_{i=0}^{n_e-1} \boldsymbol{w}_i^\intercal \int_{\Omega_e} \rho_0 \boldsymbol{b}_i N_i dv + \biguplus_{e_f} \sum_{i=0}^{n_{e_f}-1} \boldsymbol{w}_i^\intercal \int_{\Omega_{e_f}} \rho_0 \boldsymbol{t}_i N_i da \\ &= \biguplus_e \sum_{i=0}^{n_e-1} \boldsymbol{w}_i^\intercal \boldsymbol{B}_i + \biguplus_{e_f} \sum_{i=0}^{n_{e_f}-1} \boldsymbol{w}_i^\intercal \boldsymbol{T}_i \\ &= \boldsymbol{B} \cdot \boldsymbol{w} + \boldsymbol{T} \cdot \boldsymbol{w} \end{aligned} \tag{2.20}$$

All components combined together

Combining equations (2.18), (2.19) and (2.20) together yields the following equation

$$[\boldsymbol{M}\ddot{\boldsymbol{x}} - \boldsymbol{R}(\boldsymbol{u}^h) - \boldsymbol{B} - \boldsymbol{T}] \cdot \boldsymbol{w} = 0$$

Note that for the type of simulations we are interested in, a damping matrix $\boldsymbol{D}$ is often introduced to model damping effects occurring in structures undergoing dynamic motion. These effects usually come from internal friction within the material. The damping matrix normally has the form of $\boldsymbol{D} = \alpha_m \boldsymbol{M} + \alpha_k \boldsymbol{K}$ where α_m and α_k are scalar parameters. The matrix $\boldsymbol{K}$ is called the tangent stiffness matrix and will be discussed later in section 2.7. The damping matrix can be used to produce a damping force vector linearly dependent on $\dot{\boldsymbol{x}}$, i.e., $\boldsymbol{F}_{damp} = \boldsymbol{D}\dot{\boldsymbol{x}}$, which gives

$$[\boldsymbol{M}\ddot{\boldsymbol{x}} - \boldsymbol{D}\dot{\boldsymbol{x}} - \boldsymbol{R}(\boldsymbol{u}^h) - \boldsymbol{B} - \boldsymbol{T}] \cdot \boldsymbol{w} = 0$$

Because test functions $\boldsymbol{w}$ are arbitrary, and since $\boldsymbol{x} = \boldsymbol{X} + \boldsymbol{u}$, the continuous problem of (2.11) is thereby transformed to the discrete problem of finding the unknown vector $\boldsymbol{u}^h = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}]$ such that

$$\boldsymbol{M}\ddot{\boldsymbol{u}} - \boldsymbol{D}\dot{\boldsymbol{u}} - \boldsymbol{R}(\boldsymbol{u}^h) - \boldsymbol{B} - \boldsymbol{T} = 0 \tag{2.21}$$

subject to

$$\boldsymbol{u}^h = \boldsymbol{u}_g \text{ on } \Gamma_u$$

For a problem of dimension d (i.e., $\Omega_0 \in \Re^d$) discretized with a mesh of n nodes, equation (2.21) translates into an algebraic system of nd equations having nd unknowns.

2.7 Linearization of the weak formulation

The dynamic equation (2.21) introduced in the last section can be solved through either explicit or implicit time integration methods. A good review of these methods in the context of hyperelasticity can be found in Wriggers (2008).

When an implicit scheme is used, a system of non-linear equations has to be solved. This is also the case when inertia terms are ignored and only the *static* system is considered. The most popular numerical method to solve these types of non-linear systems is without any doubt the Newton–Raphson (NR) iterative algorithm, which we will now present in the static case (the same principle can be easily extended to implicit formulations).

Dropping the inertia terms in equation (2.21) we get

$$\boldsymbol{R}(\boldsymbol{u}^h) = \boldsymbol{F} \tag{2.22}$$

where $\boldsymbol{F} = \boldsymbol{B} + \boldsymbol{T}$ is the vector containing the external forces applied to the simulated body. It is important to note here that the internal elastic residual vector $\boldsymbol{R}(\boldsymbol{u}^h)$ is non-linear in $\boldsymbol{u}^h$ due to the non-linearity of the Green-Lagrangian strain tensor $\mathbf{E}$. Hence, even a linear hyperelastic material such as the Saint Venant–Kirchhoff model will exhibit a non-linear relationship with the displacement.

The NR algorithm is based on a Taylor series development of equation (2.22) at a known displacement state $\bar{\boldsymbol{u}}$:

$$\boldsymbol{R}(\bar{\boldsymbol{u}} + \delta_{\boldsymbol{u}}) = \boldsymbol{F} + \boldsymbol{R}(\bar{\boldsymbol{u}}) + D\boldsymbol{R}(\bar{\boldsymbol{u}}) \cdot \delta_{\boldsymbol{u}} + \boldsymbol{r}(\bar{\boldsymbol{u}}) \tag{2.23}$$

where $D\boldsymbol{R}(\bar{\boldsymbol{u}})\cdot\delta_{\boldsymbol{u}}$ is the derivative of $\boldsymbol{R}$ in the direction of a displacement increment $\delta_{\boldsymbol{u}}$, and $\boldsymbol{r}$ is the residuum vector of the Taylor approximation. To derive the elastic force vector, we first introduce the linearization of the continuous weak term $G(\boldsymbol{u},\boldsymbol{w})$ in equation (2.11) around $\bar{\boldsymbol{u}}$:

$$\boldsymbol{L}[G]_{\Omega=\bar{\Omega}} = G(\bar{\boldsymbol{u}},\boldsymbol{w}) + DG(\bar{\boldsymbol{u}},\boldsymbol{w})\cdot\delta_{\boldsymbol{u}}$$

From now on, terms with a bar such as $\bar{\mathbf{F}}$ means the value of $\mathbf{F}$ evaluated at a known displacement $\bar{\boldsymbol{u}}$, ie $\bar{\mathbf{F}} = \mathbf{F}(\bar{\boldsymbol{u}})$. Here, $DG(\bar{\boldsymbol{u}},\boldsymbol{w})\cdot\delta_{\boldsymbol{u}} = \frac{\partial G(\bar{\boldsymbol{u}},\boldsymbol{w})}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}$ is the derivative of G with respect to $\boldsymbol{u}$ evaluated at the initial displacement $\bar{\boldsymbol{u}}$ and in the direction of $\delta_{\boldsymbol{u}}$.

For the formulation using equation (2.12), we then have:

$$\begin{aligned}
DG(\bar{\boldsymbol{u}},\boldsymbol{w})\cdot\delta_{\boldsymbol{u}} &= \int_{\Omega_0} \frac{\partial[(\bar{\mathbf{F}}\bar{\mathbf{S}}) : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w}]}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}} dv \\
&= \int_{\Omega_0} \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} : \left(\frac{\partial \bar{\mathbf{F}}\bar{\mathbf{S}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) + \bar{\mathbf{F}}\bar{\mathbf{S}} : \underbrace{\left(\frac{\partial \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right)}_{=0} dv \\
&= \int_{\Omega_0} \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} : \left[\left(\frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right)\bar{\mathbf{S}} + \bar{\mathbf{F}}\left(\frac{\partial \bar{\mathbf{S}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right)\right] dv \\
&= \int_{\Omega_0} \left(\frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right)\bar{\mathbf{S}} : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} dv + \int_{\Omega_0} \bar{\mathbf{F}}\left(\frac{\partial \bar{\mathbf{S}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} dv
\end{aligned}$$

with

$$\begin{aligned}
\left(\frac{\partial \bar{\mathbf{E}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) &= \left[\frac{1}{2}(\bar{\mathbf{F}}^{\intercal}\cdot\boldsymbol{\nabla}_{\boldsymbol{X}}\delta_{\boldsymbol{u}} + \boldsymbol{\nabla}_{\boldsymbol{X}}^{\intercal}\delta_{\boldsymbol{u}}\cdot\bar{\mathbf{F}})\right] = [\Delta\bar{\mathbf{E}}] \\
\left(\frac{\partial \bar{\mathbf{S}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) &= \frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}} : \left(\frac{\partial \bar{\mathbf{E}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) = \frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}} : (\bar{\mathbf{F}}^{\intercal}\cdot\boldsymbol{\nabla}_{\boldsymbol{X}}\delta_{\boldsymbol{u}}) \\
\left(\frac{\partial \bar{\mathbf{F}}}{\partial \boldsymbol{u}}\cdot\delta_{\boldsymbol{u}}\right) &= \boldsymbol{\nabla}_{\boldsymbol{X}}\delta_{\boldsymbol{u}}
\end{aligned}$$

which yields the final linearized version of the continuous weak equation (2.11):

$$DG(\bar{\boldsymbol{u}},\boldsymbol{w})\cdot\delta_{\boldsymbol{u}} = \int_{\Omega_0} \boldsymbol{\nabla}_{\boldsymbol{X}}\delta_{\boldsymbol{u}}\bar{\mathbf{S}} : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} dv + \int_{\Omega_0} \bar{\mathbf{F}}\left(\frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}} : (\bar{\mathbf{F}}^{\intercal}\cdot\boldsymbol{\nabla}_{\boldsymbol{X}}\delta_{\boldsymbol{u}})\right) : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w} dv \tag{2.24}$$

Equivalently, if the formulation of equation (2.13) is used, we have:

$$\begin{aligned} DG(\bar{\boldsymbol{u}}, \boldsymbol{w}) \cdot \delta_{\boldsymbol{u}} &= \int_{\Omega_0} \left(\frac{\partial \bar{\mathbf{S}}}{\partial \boldsymbol{u}} \cdot \delta_{\boldsymbol{u}} \right) : \delta \bar{\mathbf{E}} + \bar{\mathbf{S}} : \left(\frac{\partial \delta \bar{\mathbf{E}}}{\partial \boldsymbol{u}} \cdot \delta_{\boldsymbol{u}} \right) dv \\ &= \int_{\Omega_0} \left(\frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}} : [\Delta \bar{\mathbf{E}}] \right) : \delta \bar{\mathbf{E}} + \bar{\mathbf{S}} : (\boldsymbol{\nabla}_{\boldsymbol{X}} \delta_{\boldsymbol{u}} \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w}) dv \\ &= \int_{\Omega_0} \boldsymbol{\nabla}_{\boldsymbol{X}} \delta_{\boldsymbol{u}} \bar{\mathbf{S}} : \boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{w} dv + \int_{\Omega_0} \delta \bar{\mathbf{E}} : \frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}} : [\Delta \bar{\mathbf{E}}] dv \end{aligned} \tag{2.25}$$

Both derivations lead to the same discrete form:

$$DG(\bar{\boldsymbol{u}}, \boldsymbol{w}) \cdot \delta_{\boldsymbol{u}} = \biguplus_{e} \sum_{i=0}^{n_e-1} \sum_{j=0}^{n_e-1} \int_{\Omega_e} \boldsymbol{w}_i^{\intercal} \underbrace{\left[(\nabla_{\boldsymbol{X}}^{\intercal} N_i) \bar{\mathbf{S}} (\nabla_{\boldsymbol{X}} N_j) \boldsymbol{I} + \bar{\mathbf{B}}_i^{\intercal} \bar{\mathbb{C}} \bar{\mathbf{B}}_j \right]}_{\bar{\mathbf{K}}_{ij}} \cdot \delta_{\boldsymbol{u}_j} \tag{2.26}$$

with $\bar{\mathbb{C}} = \frac{\partial \bar{\mathbf{S}}}{\partial \mathbf{E}}$ and where $\bar{\mathbf{K}}_{ij}$ is the element's tangent stiffness matrix relating the force acting on the node i when a small displacement of the node j occurs.

Using this result, we can formulate the direction derivative of the elastic residual vector $\boldsymbol{R}$ as

$$\boldsymbol{K} = DR(\bar{\boldsymbol{u}}) \cdot \delta_{\boldsymbol{u}} = \biguplus_{e} \sum_{i=0}^{n_e-1} \sum_{j=0}^{n_e-1} \int_{\Omega_e} \underbrace{\left[(\nabla_{\boldsymbol{X}}^{\intercal} N_i) \bar{\mathbf{S}} (\nabla_{\boldsymbol{X}} N_j) \boldsymbol{I} + \bar{\mathbf{B}}_i^{\intercal} \bar{\mathbb{C}} \bar{\mathbf{B}}_j \right]}_{\bar{\mathbf{K}}_{ij}} \tag{2.27}$$

where $\boldsymbol{K}$ is the assembled global tangent stiffness matrix.

Dropping the residuum vector $\boldsymbol{r}$ in equation (2.23) leads to the linear system of equations that has to be solved at an iteration k of the NR algorithm:

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{u}^k + \delta_{\boldsymbol{u}}^k) \cdot \delta_{\boldsymbol{u}}^{k+1} &= \boldsymbol{F} + \boldsymbol{R}(\boldsymbol{u}^k) \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \delta_{\boldsymbol{u}}^{k+1} \end{aligned} \tag{2.28}$$

The overall procedure is presented in algorithms 1 below.

Algorithm 1 Newton-Raphson

procedure NEWTON-RAPHSON($n, \bar{\boldsymbol{u}}, \epsilon_r, \epsilon_u$)
  $\delta_{\boldsymbol{u}}^0 \leftarrow \mathbf{0}$
  $\boldsymbol{u}^0 \leftarrow \bar{\boldsymbol{u}}$
  Assemble $\boldsymbol{R}(\boldsymbol{u}^0)$ ▷ Using eq. (2.17)
  while $k \leq n$ **do**
    Assemble $\boldsymbol{K}(\boldsymbol{u}^k + \delta_{\boldsymbol{u}}^k)$ ▷ Using eq. (2.27)
    Solve $\delta_{\boldsymbol{u}}^{k+1}$ ▷ Using eq. (2.28)
    Assemble $\boldsymbol{R}(\boldsymbol{u}^k)$ ▷ Using eq. (2.17)
    $r_k \leftarrow \left\| \boldsymbol{R}(\boldsymbol{u}^k) \right\|$
    $\boldsymbol{u}^{k+1} \leftarrow \boldsymbol{u}^k + \delta_{\boldsymbol{u}}^{k+1}$
    if $\frac{r_k}{r_0} \leq \epsilon_r$ **then**
      Converged
    if $\frac{\left\|\delta_{\boldsymbol{u}}^{k+1}\right\|}{\left\|\boldsymbol{u}^{k+1}\right\|} \leq \epsilon_u$ **then**
      Converged
    $k \leftarrow k + 1$

2.8 Isoparametric elements

To complete the explicit formulation of the discrete problem stated in equation (2.21), two additional components are required. The first one is the element-wise shape functions $N_i(\boldsymbol{X})$ needed for the approximation of the displacement field. The second one is the integration of this field over the same element. These two components are directly related to the discretization approach being used.

As we recall from equation (2.14), the shape functions of an element are used to approximate a scalar or vector value at any given position inside the element's material subspace. A special case of shape functions are found when *isoparametric elements* are used for the discretization of the domain. Here the term *isoparametric element* is taken from the traditional Finite Element methodology, where the discrete subspace Ω_e is enclosed inside a geometrical object, usually a polytope (i.e., polygons for 2-dimensional manifolds, or polyhedrons for 3-dimensional manifolds). With the isoparametric concept both the interpolation of a field variable (e.g., the displacement vector field) and the geometry (e.g., the position vector field) inside an element are derived from the same shape functions. Hence, in addition to equation (2.14), we can also interpolate the position vector of any point within an element e having n_e nodes using

$$\boldsymbol{X}_e(\boldsymbol{\xi}) = \sum_{i=0}^{n_e-1} N_i(\boldsymbol{\xi})\boldsymbol{X}_i \tag{2.29}$$

$$\boldsymbol{x}_e(\boldsymbol{\xi}) = \sum_{i=0}^{n_e-1} N_i(\boldsymbol{\xi})\boldsymbol{x}_i \tag{2.30}$$

where $\boldsymbol{\xi} = [\xi, \eta, \zeta]$ is called the *local* (or *canonical*) coordinates vector. These coordinates represent the position of a point within a *reference* element denoted by the subspace $\Omega_\square$. Using these coordinates, the shape functions of an isoparametric element can therefore be used to map the position of a point in the reference element into its position inside either the original or deformed configuration, and vice versa (see figure 2.5).

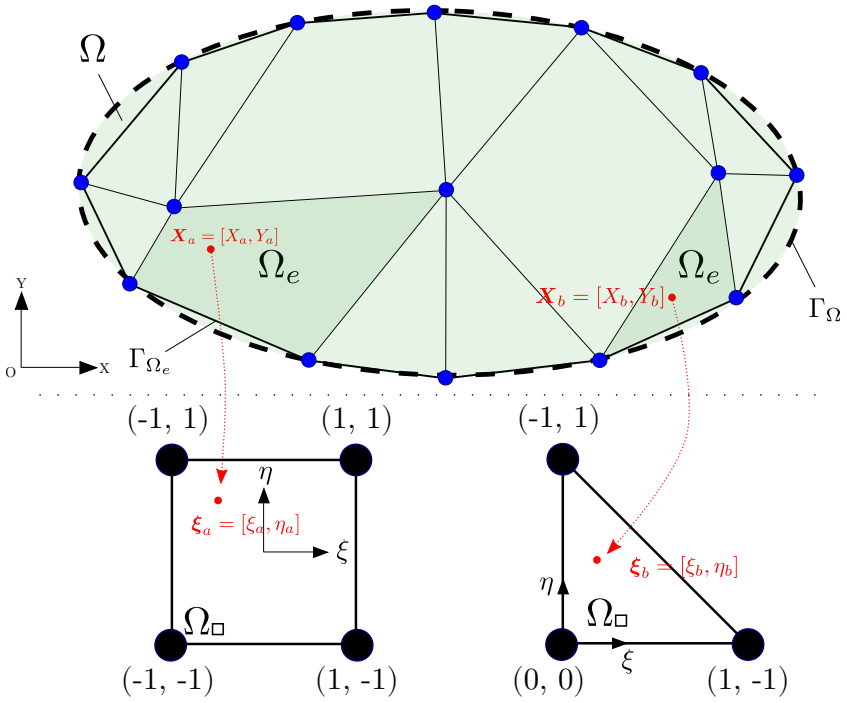


Figure 2.5: Transformation of the coordinates of a point in Ω_e to its canonical coordinates

.

In section 2.6, we also saw that the gradient of the displacement at any point inside an element can be approximated using the values of the displacement at the element's nodes and the gradient of their shape functions with respect to material

coordinates, i.e.:

$$\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}_e^h = \sum_{i=0}^{n_e-1} \boldsymbol{u}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{X}} N_i$$

The jacobian of the mapping described in equations (2.29) and (2.30) can be used to transform a gradient from one basis to its counterpart in the other configuration of the mapping. This is obtained by taking the derivatives of $\boldsymbol{X}_e$ and $\boldsymbol{x}_e$ with respect to the canonical coordinates, yielding

$$\mathbf{J}_e = \frac{d\boldsymbol{X}}{d\boldsymbol{\xi}} = \sum_{i=0}^{n_e-1} \boldsymbol{X}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i \tag{2.31}$$

$$\mathbf{j}_e = \frac{d\boldsymbol{x}}{d\boldsymbol{\xi}} = \sum_{i=0}^{n_e-1} \boldsymbol{x}_i \otimes \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i \tag{2.32}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i = [\frac{dN_i}{d\xi}, \frac{dN_i}{d\eta}, \frac{dN_i}{d\zeta}]$ is the gradient of the scalar-valued shape function N_i with respect to $\boldsymbol{\xi}$.

The gradient of the n_e shape functions N_i with respect to material coordinates then becomes

$$\boldsymbol{\nabla}_{\boldsymbol{X}} N_i = \mathbf{J}_e^{-1} \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i \tag{2.33}$$

and the displacement gradient can be computed at any points on the reference element using

$$\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}_e^h = \sum_{i=0}^{n_e-1} \boldsymbol{u}_i \otimes (\mathbf{J}_e^{-1} \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i) \tag{2.34}$$

where again, thanks to the Lagrangian description, the vector $(\mathbf{J}_e^{-1} \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i)$ depends only on material coordinates and can be precomputed before the simulation.

Using the isoparametric concept, we can also translate an integral on a material element domain Ω_e to an integral on the reference element domain $\Omega_\square$. From equation (2.31), we have $d\boldsymbol{X} = \mathbf{J}_e d\boldsymbol{\xi}$. An integration on Ω_e can therefore be transformed to

$$\int_{\Omega_e} f(\boldsymbol{X}) d\boldsymbol{X} = \int_{\Omega_\square} f(\boldsymbol{\xi}) \mathbf{J}_e(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{2.35}$$

When the product $f(\boldsymbol{\xi})\mathbf{J}_e(\boldsymbol{\xi})$ is a polynomial, the integration on the reference element can usually be analytically computed. However, when using the hyperelastic equations presented earlier in this chapter, this product generally yields a rational function. In this case, we usually rely on the Gauss numerical integration method which as proved to be very accurate. The integration is therefore approximated by a sum of values taken at different locations inside the reference element. These sampling locations are called the *Gauss* (or *integration*) points. If we let n_I be the number of Gauss points inside the reference element, the integration can then be approximated by

$$\int_{\Omega_\square} f(\boldsymbol{\xi}) \mathbf{J}_e(\boldsymbol{\xi}) d\boldsymbol{\xi} \approx \sum_{I=1}^{n_I} f(\boldsymbol{\xi}_I) \mathbf{J}_e(\boldsymbol{\xi}_I) w_I \tag{2.36}$$

where $\boldsymbol{\xi}_I$ is are canonical coordinates of the I^{th} Gauss points, and w_I its weight. A description of the shape functions N_i, their gradient with respect to canonical coordinates $\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_i$, the location $\boldsymbol{\xi}_I$ of Gauss points and their weights w_I for various isoparametric elements can be found in the work of Wriggers (2008) and André Fortin and Garon (2018).

An overview of the classical isoparametric elements for three-dimensional problems is illustrated in figure 2.6. Here, P_1 and H_1 are linear elements whereas P_2 and Q_2 are quadratic, and will hold optimal convergence of the same order. An excellent overview of their performance in the simulation of large deformations can be found in Chamberland, Fortin, and M. Fortin (2010). We quickly highlight here some of the results presented in their work. By designing a manufactured solution, they are able to compare the relative error of the displacement norm between various meshes made of different element types and the exact solution. Figure 2.7 illustrates the geometry of the problem where the domain is filled with a compressible SVK material (section 2.3.1) using a Young's modulus of 1 and a Poisson's ratio of 0.3.

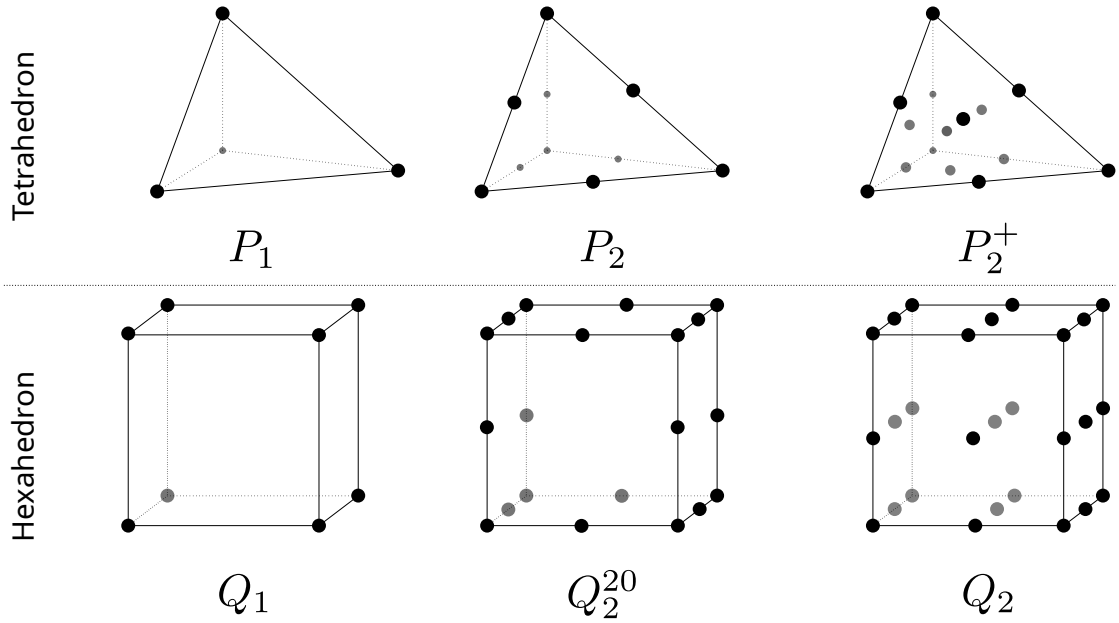
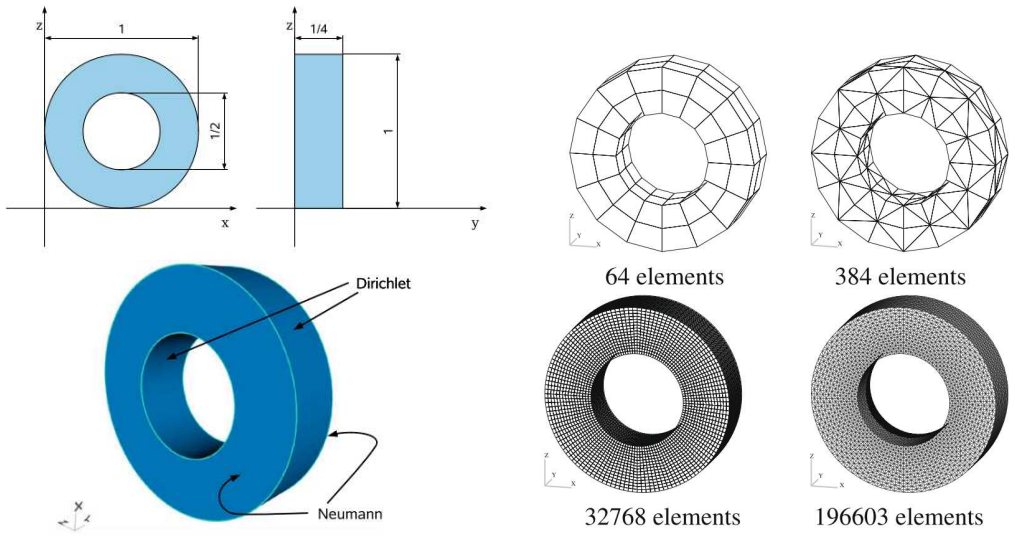


Figure 2.6: Comparison of the classical tetrahedral and hexahedral elements for 3D problems.

Using a log-log scale, the relative H1-norm of the displacement error is shown in figure 2.8a. To better appreciate the performance of these discretization choices, figures 2.8b and 2.8c show, respectively, the relative L2-norm of the Cauchy stress tensor error ($\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{0,\Omega}$) and the computation time taken from the start to the end of the simulation.

These results were obtained using a discretization made of well-formed elements on a very simple geometry. While the authors show similar results by adding small perturbations on the positions of the nodes, we are not sure if similar results could be acquired when modeling a very complex shape such as the one of a human liver. However, it does illustrate the benefits of using higher order elements. Similarly to the conclusion of Chamberland, Fortin, and M. Fortin (2010), we believe that, when it is possible, higher order elements should be used for the simulation of large deformation.

As aforementioned in the introduction, the difficulty with discretizing the domain with these standard isoparametric is found when complex shapes with sharp and concave boundaries have to be modeled. Mesh generation becomes increasingly complex and, while some research is currently done for automatic hexahedral mesh generation [Sokolov et al. (2016)], tetrahedral elements are generally preferred, especially around sharp angles. In this case, a large number of elements are used



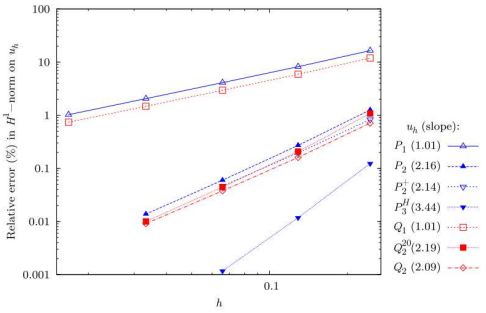
(a) Geometry and boundary conditions of the problem.

(b) Coarser and finer hexahedral (left) and tetrahedral (right) meshes.

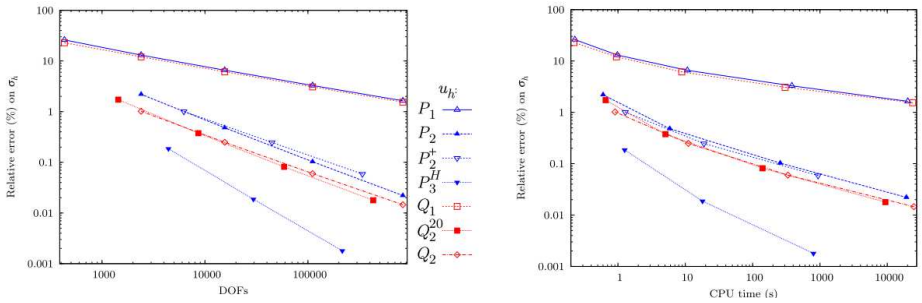
Figure 2.7: Geometry and discretization of the manufactured solution. Graphics and captions taken from Chamberland, Fortin, and M. Fortin (2010).

to respect the geometry of these angles. This rapidly becomes problematic as it goes against our computational time key constraint.

To overcome these difficulties and enable higher order discretizations, two research directions can be taken. The first one would be to improve current automatic meshing method, and represent a complete line of research on its own. Alternatively, we could try to find a numerical method that either does not require at all an element-based discretization, or at least relax the meshing constraint around sharp features. For this thesis, we have chosen the latter. As we will describe in the following chapters, mesh-free Galerkin methods and mesh-based immersed boundary methods will be studied. In all cases, while we focus on linear discretization for the sake of simplicity, all our studied methods are compatible with higher order approximations. Note that these two lines of research are not incompatible with each other. As we will see, most of the methods we have implemented still use a hybrid approach where a computation mesh is necessary at some point. Thus, while not required, we could still highly benefit from advanced automatic mesh generation.



(a) Relative error as a function of the maximum element length (h)



(b) relative error as a function of the number of DOFs.

(c) relative error as a function of the number of CPU time in seconds.

Figure 2.8: Convergence of the manufactured solution. Graphs and captions taken from Chamberland, Fortin, and M. Fortin (2010).

2.9 Development of a generic and efficient library

We quickly realized that the implementation of a discretization scheme outside of the traditional isoparametric approach was not possible with most of the software tools currently available. In most cases, these tools are simply not adequate for a real-time application like ours which required that the intra-operative data be dynamically injected into the simulation process. As a starting point, we therefore derived and implemented a complete set of numerical algorithms. These have been made available in an advanced simulation software library [1] which turned out to be a valuable asset for the entire research team. At the end of this thesis, the library accumulated about $17k$ lines of C++ code, and about $1k$ lines of Python code.

The technical requirements for our research were as follows. First, we needed a

[1] https://github.com/jnbrunet/caribou

way to quickly implements new shape functions and their derivatives. Similarly, we wanted to study different volumetric quadrature schemes. Finally, different hyperelastic materials had to be implemented. Hence, the software design had to be generic enough to allow a combination of all these requirements. It also had to be efficient enough to avoid the creation of a bottleneck that would prevent the biomechanical model from meeting its computational speed requirement.

We opted for a compile-time polymorphism design using generic C++ template programming. The idea here was to let the compiler decide how to best optimize the set of operations executed during the simulation, while keeping an object-oriented code. The lowest numerical building block of the simulation framework being the elements, we started our implementation there. We created the *Element* concept as a generic computational class that would be inherited by all element types. With this approach, a C++ class needs to provide the following properties to be considered a valid *Element*: (1) its canonical dimension, (2) its material dimension, (3) its number of nodes known at compile time, or -1 if it will be determined at runtime, (4) its number of quadrature nodes known at compile time, or again -1 in the dynamic case, (5) a method to obtain the material coordinates of its nodes, (6) a method to obtain the quadrature nodes and their respective weights, and (7) two methods to get the values of the shape functions, and their derivatives with respect to the material coordinates, respectively. Figure 2.9 illustrates the resulting simplicity for the implementation of a linear quadrangle element.

Once compiled, various utilities become available for the newly created element in both C++ and Python code. Hence, the *Element* concept provides a way to quickly add interpolation and quadrature numerical procedures. Since standard isoparametric elements have a number of nodes, quadrature points and shape functions already known at compile time, most modern compilers will be able to aggressively inline the code in order to optimize the computation. For meshless methods (chapter 3) or immersed-boundary methods (chapter 4), the number of nodes or quadrature nodes are often determined at run time, for example depending on the location of the boundaries, or the location of neighbor nodes. Our *Element* concept poses no difficulties in this case. This information can be given at run time during the construction of the element.

Elements are used all over the library, most of the time as a template parameter to different class components. An example is the *Domain* class, which creates a topology of elements and allows us to approximate or integrate field functions over a discrete domain made of a given *Element* type. For standard finite elements, a *Domain* will therefore represent the complete mesh or a part of it. It will store for each element the indices of its nodes. For IB or meshless methods, the domain

```cpp
template<UNSIGNED_INTEGER_TYPE _Dimension>
struct traits<Quad <_Dimension, Linear>> {
    static constexpr UNSIGNED_INTEGER_TYPE CanonicalDimension = 2;
    static constexpr UNSIGNED_INTEGER_TYPE Dimension = _Dimension;
    static constexpr INTEGER_TYPE NumberOfNodesAtCompileTime = 4;
    static constexpr INTEGER_TYPE NumberOfQuadratureNodesAtCompileTime = 4;
};

template<UNSIGNED_INTEGER_TYPE _Dimension>
struct Quad <_Dimension, Linear> : public Element<Quad <_Dimension, Linear>> {
private:
    // Shape values at canonical coordinates xi
    inline auto get_L(const LocalCoordinates & xi) const -> Vector<NumberOfNodesAtCompileTime> {
        const auto  & u = xi[0];
        const auto  & v = xi[1];
        return {
            1 / 4. * (1 - u) * (1 - v),  // Shape value of 1th node
            1 / 4. * (1 + u) * (1 - v),  // Shape value of 2th node
            1 / 4. * (1 + u) * (1 + v),  // Shape value of 3th node
            1 / 4. * (1 - u) * (1 + v)   // Shape value of 4th node
        };
    };

    // Shape derivatives at canonical coordinates xi
    inline auto get_dL(const LocalCoordinates & xi) const {
        const auto & u = xi[0];
        const auto & v = xi[1];

        Matrix<NumberOfNodesAtCompileTime, CanonicalDimension> m;
        //          dL/du                   dL/dv
        m << -1 / 4. * (1 - v),   -1 / 4. * (1 - u),     // Shape derivatives of 1th node
             +1 / 4. * (1 - v),   -1 / 4. * (1 + u),     // Shape derivatives of 2th node
             +1 / 4. * (1 + v),   +1 / 4. * (1 + u),     // Shape derivatives of 3th node
             -1 / 4. * (1 + v),   +1 / 4. * (1 - u);     // Shape derivatives of 4th node
        return m;
    };

    // Quadrature nodes
    inline auto get_quadrature_nodes() const -> const auto & {
        return {
            QuadratureNode{LocalCoordinates(-1/sqrt(3), -1/sqrt(3)),   Weight(1)},  // Quadrature node 1
            QuadratureNode{LocalCoordinates(+1/sqrt(3), -1/sqrt(3)),   Weight(1)},  // Quadrature node 2
            QuadratureNode{LocalCoordinates(+1/sqrt(3), +1/sqrt(3)),   Weight(1)},  // Quadrature node 3
            QuadratureNode{LocalCoordinates(-1/sqrt(3), +1/sqrt(3)),   Weight(1)}   // Quadrature node 4
        };
    }
};
```

Figure 2.9: Example of a linear quadrangle implementation.

will also store the location and weights of the quadrature points, and the values of the shape functions and their derivatives. As we will see in the chapter 3, one of our implementations of a meshless method uses a nodal integration scheme. In this case, a simple node (called a particle) is seen as an *Element* containing only one quadrature point (the particle itself), and its neighbors points (the nodes of the element used for the approximation). Hence, our *Element* concept paired with the *Domain* class becomes a powerful design that simplifies testing of different approximation and integration schemes.

Figures 2.10 and 2.11 illustrate code examples for the assembly of the internal residual vector of a Neo-Hookean material in both C++ and Python, respectively. We can see how intuitive the implementation of a new material becomes, and how

```
#include <Caribou/Topology/Mesh.h>
#include <Caribou/Topology/Domain.h>
#include <Caribou/Geometry/Tetrahedron.h>

// Domain creation
Mesh mesh (get_nodes());
auto domain = mesh.add_domain<Tetrahedron<Quadratic>>(get_tetra_indices());

// Material Lame's parameters
auto l = get_lambda();
auto m = get_mu();
auto Id = Matrix<3, 3>::Identity();

// Internal residual assembly
auto R = domain.assemble(
  [&l, &m, &id](const Tetrahedron<Quadratic> & e, const Matrix<10, 3> & dN_dx) -> Vector<3> {
    const auto F  = e.nodes().T() * dN_dx;       // Deformation tensor
    const auto J  = F.determinant();
    const auto C  = F.T() * F;                   // Right Cauchy-Green strain tensor
    const auto Ci = C.inverse();
    const auto S  = (Id - Ci)*m + Ci*(l*log(J)); // Second Piola-Kirchhoff stress tensor
    return  F*S;
  }
);
```

Figure 2.10: C++ example of the internal residual assembly (equation 2.17) for a Neo-Hookean material (equation 2.9).

it stays independent of the discretization.

Real-time interactive simulations involves many more numerical procedures, such as the visualization of the simulated bodies, collisions handling, time integration, and more. The open-source SOFA framework [Allard et al. (2007)] is a well-known set of numerical tools for interactive simulations and its contributions to the surgery simulation research field have grown considerably for the past decade. For this reason, we inserted our library as a form of plugin to this framework, hence inheriting its numerous numerical procedures. The plugin combines all the contributions we have made to the framework and the components that were necessary for our research. The structure of the plugin is illustrated in figure 2.12. A more thorough documentation of these components can be found at https://caribou.readthedocs.io.

```python
from Caribou.Topology import Mesh, Domain
from Caribou.Geometry import Tetrahedron
from Caribou import Quadratic
import numpy

# Domain creation
mesh = Mesh(get_nodes())
domain = mesh.add_domain(Tetrahedron(Quadratic), get_tetra_indices())

# Material Lame's parameters
l, m = get_lambda(), get_mu()
Id = numpy.identity(3)

# Internal residual assemby
def residual(e, dN_dx):
    F  = e.nodes().dot(dN_dx)                # Deformation tensor
    J  = numpy.linalg.det(F)
    C  = numpy.dot(F.T, F)
    Ci = numpy.linalg.inv(C)                 # Right Cauchy-Green strain tensor
    S  = (Id - Ci) * m + Ci*(l*numpy.log(J)) # Second Piola-Kirchhoff stress tensor
    return  numpy.dot(F, S)

R = domain.assemble(residual)
```

Figure 2.11: Python example of the internal residual assembly (equation 2.17) for a Neo-Hookean material (equation 2.9).

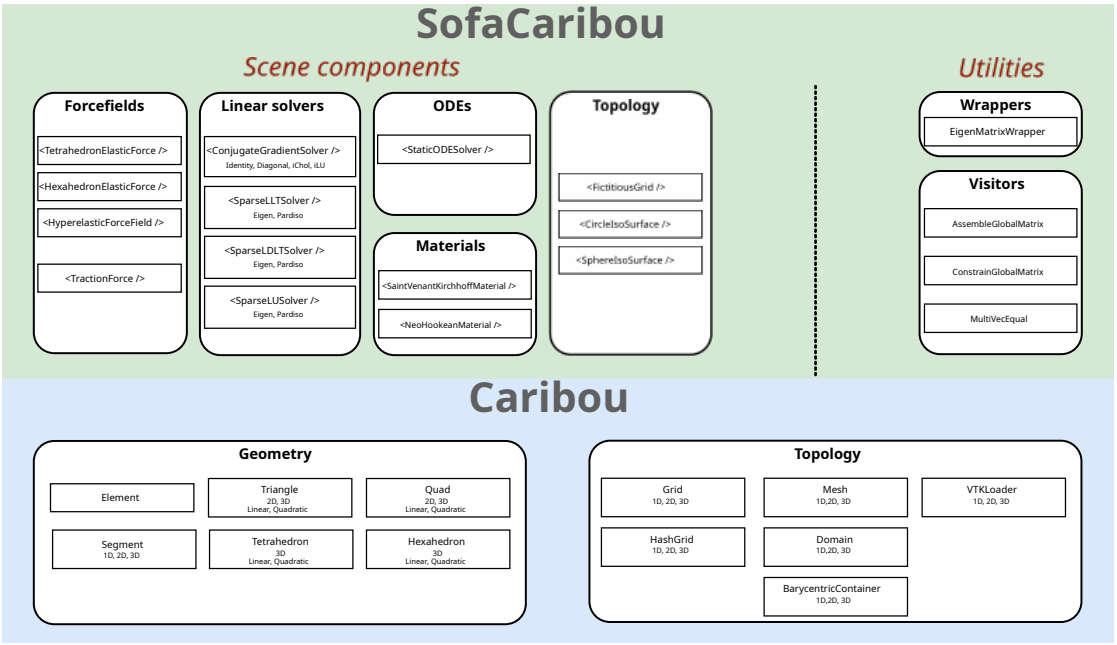


Figure 2.12: Structure of the Caribou SOFA plugin.

2.10 Discussion

In this chapter, we provided an extensive review of some of the basic concepts behind the theory of hyperelasticity. We also provided the mathematical framework associated with the numerical resolution process of the Partial Differential Equations for a FE method. This task was a challenging one and has consumed a non-negligible portion of our research effort. But this extensive review was a necessary step before exploring alternative numerical methods.

While software that uses the FE approach usually allow the implementation of different material models, most of the time the use of standard FE isoparametric elements is required. In the next chapters, we will extend the basic approximation and integration methods we have just described to arrive at a solution that does not impose such requirement.

We will also use the theory provided in this chapter to adapt the shape functions, their derivatives and the integration of the weak formulation over the domain for alternative discretization methods. We will see how the boundary conditions can be imposed when we are no longer following the traditional FE methodology. Finally, we will present the numerical challenges inherent to the field of augmented reality and how the basic concepts presented in this chapter must be adapted.

It is important to note that throughout the development of the alternative methods that will be discussed in the next chapters, most of the concepts introduced in this chapter had to be implemented. The detailed description of the key equations has been very useful in this context and we believe that it will remain a valuable reference for future software coding development.

Chapter

3

MESHLESS METHODS

As we saw in the previous chapter, the discretization of the simulated domain constitutes a key component of any Galerkin method. With traditional Finite Element (FE) methods, the process uses a mesh of isoparametric elements such as

those discussed in section 2.8. Meshless methods use a different approach whereby a neighborhood of points, often called particles, are used to approximate the displacement field and to impose boundary conditions. Hence, the term meshless comes from the fact that no "traditional" mesh of geometrical elements is needed. However, as we will see, a meshless discretization may also be coupled with a mesh of elements for the numerical integration of the weak formulation (equation (2.11)).

One of the main advantages of meshless methods come from the simplicity of their implementation. To discretize the simulated domain Ω_0, one simply needs to fill in the interior volume of the object with particles which, similarly to the nodes of elements in FE methods, represent the unknown degrees of freedom of the system to be solved. The density of particles can be dynamically increased in regions where higher precision of the solution is needed. The approximation of a field function at any given point is done by taking the value of neighboring particles, which are then weighted down using a decreasing monotone kernel (or weight) function. Shape functions are usually constructed from a given set of monomials. Hence, similarly to high order isoparametric elements, choosing monomials of higher degrees will enable approximation of higher degrees.

In the case of surgical simulations, these properties of meshless methods are highly desirable. 3D surface reconstructions of organs such as those done on the liver are complex. For example, a typical human liver will have a concave shape around its two lobes. Internal structures such as blood vessels and tumors introduce curved interfaces between different materials which make an automatic discretization of the interior difficult to be achieved when using only geometrical elements. Hence, in order to correctly represent these boundaries, traditional FE methods require a very fine mesh. When topological changes arise, such as cuts and tears made by the surgeon, new boundaries are introduced. Regions around these discontinuities need to be re-meshed dynamically during the simulation. This is where the simplicity of the meshless implementation comes very handy. The remeshing process required in FE methods is replaced by the simple process of finding new neighbors that do not cross the discontinuity.

The next section presents a review of the meshless discretization principles. We will then present two meshless approaches to construct the discrete algebraic system presented earlier (equation (2.21)). The first one, the *point-based* method, uses a fully meshless approach where both the numerical approximation and integration terms found in the weak internal virtual work component are given from the analysis of point neighborhoods. The second approach also uses a numerical approximation based on points, but where the numerical integration is computed

using a background mesh of elements, hence resulting in a mixed method.

3.1 Literature review

In the early days of computer simulations, Reeves (1983) introduced the concept of a particle which is simply a generated point that can move in space and that dies off after a certain amount of time. By filling an object with many particles, different phenomenons such as fire and clouds can then be simulated. The dynamics of the system is established from a stochastic model and particles are not interacting with each other. In Luciani et al. (1991), the particles represent balls of dynamic radius and, depending on the distance between them, will either attract or repel each other. The attraction-repulsion force is determined with the help of Lennard-Jones intermolecular potential. Later on, Szeliski and Tonnesen (1992) extended the use of a Lennard-Jones based potential by adding an orthogonal frame to each particle in order to enforce momentum preservation.

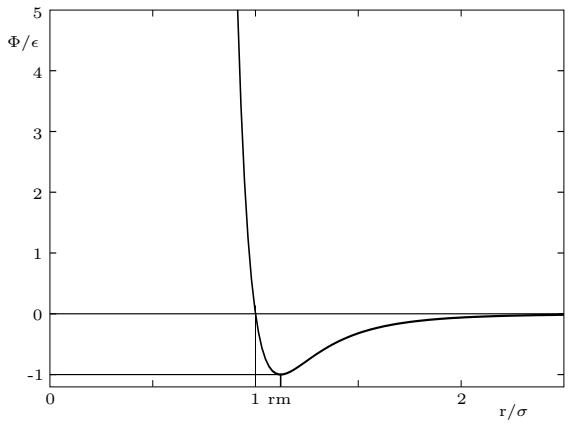


Figure 3.1: Lennard-Jones potential $\phi_{LJ}(r) = \frac{B}{r^n} - \frac{A}{r^m}$ with $n = 12$, $m = 6$, $\epsilon = \frac{B^2}{4A}$ and $\sigma = \sqrt[6]{\frac{A}{B}}$. Here, the scalar parameters A and B defines the attraction-repulsion behavior, and r denotes the distance between two particles.

While these techniques pioneered the field of meshless discretization, it is the work of Desbrun and Gascuel (1996) that first introduced a method, called Smoothed Particle Hydrodynamics (SPH) (from the work of Lucy (1977) and Monaghan (1992) in astrophysics applications), for animations of deformable objects driven entirely by force. The SPH method is able to interpolate both a field function and its derivatives with a discretization of the space entirely based on particles, hence without the need of a background grid as was usually done with Particle In Cell (PIC) methods. Their interpolant is defined as

$$A(\boldsymbol{r}) = \int A(\boldsymbol{r\prime}) W(\boldsymbol{r} - \boldsymbol{r\prime}, h) d\boldsymbol{r\prime}$$

for any field $A(\boldsymbol{r})$, and where $d\boldsymbol{r\prime}$ is a differential volume element around any position $\boldsymbol{r\prime}$. The weighing kernel W is a monotonic and decreasing scalar field as $\|\boldsymbol{r} - \boldsymbol{r\prime}\|$ approaches a distance of h, and has the following two key properties:

$$\int W(\boldsymbol{r} - \boldsymbol{r\prime}, h) d\boldsymbol{r\prime} = 1$$
$$\lim_{h \to 0} W(\boldsymbol{r} - \boldsymbol{r\prime}, h) = \delta(\boldsymbol{r} - \boldsymbol{r\prime})$$

The interpolant and its derivatives are approximated from a set of neighboring particles with

$$A(\boldsymbol{r}) = \sum_i m_i \frac{A_i}{\rho_i} W(\boldsymbol{r} - \boldsymbol{r}_i, h) \tag{3.1}$$
$$\nabla A(\boldsymbol{r}) = \sum_i m_i \frac{A_i}{\rho_i} \nabla W(\boldsymbol{r} - \boldsymbol{r}_i, h)$$

where m_i is the mass of a particle i, and ρ_i its density defined as

$$\rho(\boldsymbol{x}_i) = \sum_j m_j W(\boldsymbol{x}_i - \boldsymbol{x}_j, h)$$

While SPH was initially designed for fluid and astrophysics simulations, Desbrun and Gascuel (1996) first introduced it to the simulation of deformable bodies by defining an energy potential that would push particles to regain their initial volume. The stiffness of the material is set with a single scalar parameter k, and the force acting on a particle becomes

$$\boldsymbol{F}_i = -k m_i \left(\frac{\rho_i - \rho_0}{\rho_i^2} \sum_{j \neq i} m_j \nabla_i W(\boldsymbol{x}_i - \boldsymbol{x}_j, h) + \sum_{j \neq i} m_j \frac{\rho_j - \rho_0}{\rho_j^2} \nabla_i W(\boldsymbol{x}_i - \boldsymbol{x}_j, h) \right]$$

with the kernel W defined in equation (3.3).

The use of SPH by Desbrun and Gascuel (1996) established an important step in the field of deformable meshless simulations for computer animations. Later on, Müller, Keiser, et al. (2004) pushed further this concept, this time, by introducing the theory of linear elasticity as a ground for their energy potential. However, instead of using the SPH interpolant, they chose an interpolant based on the Moving Least Squares (MLS) method [Lancaster and Salkauskas (1981)] which consisted in minimizing the error of a first degree Taylor approximation of the displacement gradient $\boldsymbol{\nabla u}$. While their interpolant is one degree higher than the SPH one, special attention has to be taken for the placement of the particles as their method is incompatible with particles neighborhood having co-linear or coplanar configurations. Their energy potential consisted of pairing the linear Saint-Venant-Kirchhoff constitutive law with a volume conserving potential.

Solenthaler, Schläfli, and Pajarola (2007b) also presented a meshless method for linear elasticity, but this time reusing the SPH approximation. Unlike Müller, Keiser, et al. (2004), their shape function allows for co-linear and coplanar neighborhoods. Becker, Ihmsen, and Teschner (2009a) extended this strategy by carrying the corotational principle [Müller, Dorsey, et al. (2002)] to the SPH formulation. Here, a rotation frame is extracted and removed from the nodal position just before computing the strain, hence allowing the use of the small strain tensor $\boldsymbol{\epsilon}$ (cf. section 2.3.1). While these meshless methods present visually plausible results, their volumetric integration approach is based on an approximation of the particle density. This reduces the precision of the simulation and induces instabilities. It is especially true given that their particles represent both the degrees of freedom and the integration points.

The introduction of meshless methods for surgical simulation applications is more recent. Horton et al. (2010) has proposed a fully non-linear element-free Galerkin method [Ted Belytschko, Y. Y. Lu, and L. Gu (1994)] for certain types of surgical simulations. Although no mention was made with respect to real-time compliance, all indications are that they were the first to propose a fast Galerkin-based solution without element-based approximations. Since then, their work has been carried over to various surgical applications [K. Miller et al. (2012); G. Y. Zhang et al. (2014); Dehghan et al. (2016); Dong et al. (2016); Wittek et al. (2016)]. While these meshless methods are accurate, their use of fully non-linear material is time-consuming and might not be well adapted for applications requiring a real time or close to real-time framerate. Since they use an explicit time integration scheme, they are also restricted to small and regular time step intervals which implies a less robust solution in an interactive environment.

3.2 Kernel and shape functions

Similar to the Finite Element (FE) approach, meshless methods involve the approximation of a vector field $\boldsymbol{u}(\boldsymbol{X})$ at a given position $\boldsymbol{X}$ of the undeformed domain Ω_0. However, instead of using geometric properties of an iso-parametric element such as a tetrahedron or a quadrangle to approximate the field, meshless methods use a point cloud where each point, often call *particle*, represents the degrees of freedoms x_i, y_i and z_i as a compact support of influence. This support is usually called the influence domain Ω_i of the particle i. The approximation $\boldsymbol{u}^h(\boldsymbol{x}) \approx \boldsymbol{u}(\boldsymbol{x})$ is done using the scalar-valued shape functions $\phi_i(\boldsymbol{X})$ of all particles neighbor to $\boldsymbol{X}$:

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_i \phi_i(\boldsymbol{x})\boldsymbol{u}_i \tag{3.2}$$

where $\boldsymbol{u}_i$ is the field value at the particle i.

The support of a particle's shape function is usually built from a weight function $W(\boldsymbol{X} - \boldsymbol{X}_I, h_I)$ (often called a kernel or window function), which is monotonic and has its value decreasing as the distance between $\boldsymbol{X}$ and the particle i approaches the boundary of its influence radius h_i. Most support domains are either circular, or rectangular, as shown in figure 3.2.

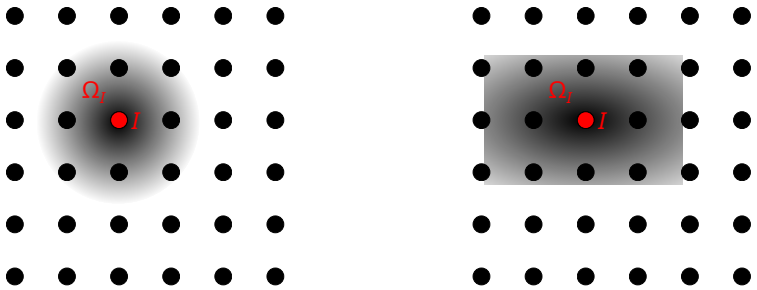


Figure 3.2: Influence domain from monotonic and decreasing weight functions $W(\boldsymbol{X} - \boldsymbol{X_i}, h_i)$

Because our modeling approach is using a Lagrangian formulation (see section 2.1), the influence a particle has on any material position $\boldsymbol{X}$ will remain constant throughout the simulation, unless topological changes happen (for example, if cuts or fractures are also simulated). This means that shape functions ϕ_i can be precomputed before starting the simulation, resulting in a great computation advantage as compared to the Eulerian formulations.

The choice of weight function usually varies from one application to another. However, a few conditions must be considered to ensure a good local representation of the field around a particle, and the consistency of the approximation towards the solution [Ted Belytschko, Yury Krongauz, et al. (1996)]. These conditions are:

1. $W(\boldsymbol{X} - \boldsymbol{X}_i, h_i) > 0$ for $x \in \Omega_i$
2. $W(\boldsymbol{X} - \boldsymbol{X}_i, h_i) = 0$ for $x \notin \Omega_i$
3. $\int_\Omega W(\boldsymbol{X} - \boldsymbol{X}_i, h_i) d\Omega = 1$
4. $W(\boldsymbol{X}_1 - \boldsymbol{X}_i, h_i) > W(\boldsymbol{X}_2 - \boldsymbol{X}_i, h_i)$ when $\|\boldsymbol{X}_1 - \boldsymbol{X}_i\| < \|\boldsymbol{X}_2 - \boldsymbol{X}_i\|$
5. $W(\boldsymbol{X} - \boldsymbol{X}_i, h_i) \to \delta(\|\boldsymbol{X} - \boldsymbol{X}_i\|)$ when $h_i \to 0$, $\delta(\|\boldsymbol{X} - \boldsymbol{X}_i\|)$ is the Dirac function.

The quality of the approximation is directly related to consistency criteria and the order of the field function. We will say that an approximation has a consistency of degree k if it can represent exactly a polynomial solution of degree k.

Many kernels have been documented in the literature. We have implemented and tested the following:

Spiky (Desbrun and Gascuel (1996))
$$w(q) = \begin{cases} \frac{15}{\pi h^3}(1-q)^3 & q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.3}$$

Poly6 (Müller, Charypar, and Gross (2003))
$$w(q) = \begin{cases} \frac{315}{64\pi h^3}(1-q^2)^3 & q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.4}$$

SPH (Solenthaler, Schläfli, and Pajarola (2007a))
$$w(q) = \begin{cases} \frac{cos(\frac{\pi}{2}(1+q))+1}{4h^3(\frac{\pi}{3}-\frac{8}{\pi}+\frac{16}{\pi^2})} & q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.5}$$

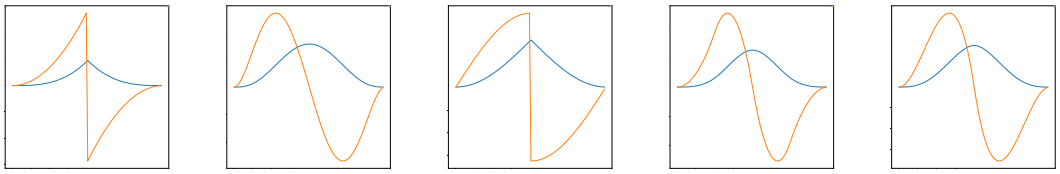
Cubic spline
$$w(q) = \begin{cases} \frac{8}{\pi h^3}(1-6q^2+6q^3) & q \leq \frac{1}{2} \\ \frac{16}{\pi h^3}(1-q)^3 & \frac{1}{2} \leq q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.6}$$

Quartic spline (Horton et al. (2010))
$$w(q) = \begin{cases} 1-6q^2+8q^3-3q^4 & q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.7}$$

with $q = \frac{\|\boldsymbol{x}-\boldsymbol{x}_I\|}{h}$.

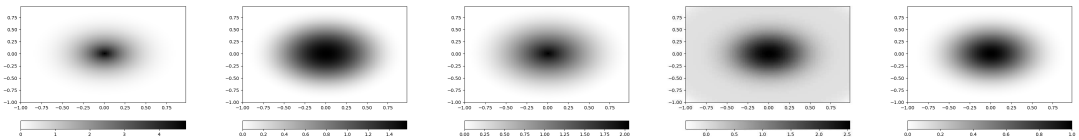
A graphical representation of those kernel can be found in figures 3.3 and 3.4. Note that the 1D figure 3.3 illustrates both W and its gradient ∇W with respect to material coordinates. Since this gradient will be directly involved with the

approximation of the deformation gradient used to compute internal elastic forces, special attention has to be taken such that it is not attenuated when two particles get closer to each other. Otherwise, the kernel might cause artificial stiffness and thereby a clustering artifact [Desbrun and Gascuel (1996)].



(a) Spiky (b) Poly6 (c) SPH (d) Cubic spline (e) Quartic spline

Figure 3.3: 1D kernels: The blue line denotes the weight function, and the orange line its derivative.



(a) Spiky (b) Poly6 (c) SPH (d) Cubic spline (e) Quartic spline

Figure 3.4: 2D kernels

We conducted many experiments with these different kernels. At the end, we were not able to differentiate the quality of the solution nor the impact on the convergence rate associated with each one. Our only explanation of this result is that the choice of the kernel as such a negligible impact that it is lost amongst all the other factors affecting the numerical simulation.

However, for all tested kernels, the radius of influence h did have a large impact. In fact, the quality of the solution is greatly dependent on h as this value determines the size of the support influence of a particle. At the time of this writing, finding an optimal value for h remains an open problem [T.-P. Fries and Matthies (2003)]. Nevertheless, we found in the literature three different methods to assign a value to each particle influence radius.

The first one simply consists in assigning the same value h to all particles. This method works well if the particle density is uniform.

$$h_i = h \tag{3.8}$$

With the second method, we find the k^{th} closest neighbors of a particle i, and multiply the distance between the two by a dilatation factor α (usually between 1 and 3).

$$\begin{aligned} h_i &= \alpha r_i \\ r_i &= \|\boldsymbol{x}_i - \boldsymbol{x}_k\| \end{aligned} \tag{3.9}$$

Finally, with the last method, we compute the average between the k closest neighbors and multiply this value by the dilatation factor.

$$\begin{aligned} h_i &= \alpha r_i \\ r_i &= \frac{1}{k}\sum_{j=1}^{k} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \end{aligned} \tag{3.10}$$

3.3 Point-based animation

The first meshless method we have considered is the Point-Based Animation (PBA) method introduced by Müller, Keiser, et al. (2004) in the context of computer animation of elastic, plastic and melting objects. With this approach, elasticity equations are solved using a discretization of the domain made of particles only. We have implemented two versions of the PBA method using both the SPH and MLS shape functions. Here is the general approach that was used.

Before the simulation, a set of particles are evenly spaced using a grid lattice. The particles that are found outside of the simulated object's surface (usually given by a triangle or quadrangle mesh defining the boundary of the object) are discarded. Then, choosing one of the three methods from equations (3.8) to (3.10), we compute the radius of influence h_i that a particle i has on its compact domain. We also choose one of the five kernels from equations (3.3) to (3.7). Note that, unfortunately, we have not been able to establish a procedure to guide us for the selection of the support distance and kernels. Most of the time, we simply used a trial and error approach.

For each particle, the indices of its k nearest neighbors are stored. If the first method (equation 3.8) for the radius of influence is used, indices will be those of

the k nearest neighbors that have a distance lesser or equal than h. Otherwise, we use the k nearest neighbors used to compute h_i.

3.3.1 Point-based numerical integration

In a certain way, with the Point-Based Animation (PBA) method, the compact support around a particle could be viewed as a special kind of element, and its closest neighbors as the element nodes. Hence, the discrete internal elastic residual and its linearization defined in the last section with equations (2.18) and (2.26) respectively can be used in the same way as with the FE methods. However, given the absence of isoparametric elements, numerical volumetric integration and field approximation methods have to be used.

Let Ω_i be the compact support of the particle i from its kernel $W_i(\boldsymbol{X}) = W(\boldsymbol{X} - \boldsymbol{X}_i, h_i)$. As the size of the support closes down on the particle, i.e., $h_i \to 0$, we can assume that the integration of a continuous field $f(\boldsymbol{X})$ tends to the value of f evaluated at $\boldsymbol{X}_i$. Phrased differently, if v_i is the volume taken by Ω_i (or the area in 2D), as v_i gets smaller, we would have

$$\int_{\Omega_i} f(\boldsymbol{X}) d\boldsymbol{X} \approx v_i f(\boldsymbol{X}_i) \tag{3.11}$$

where v_i is assumed to be the volume of the particle i. In order to approximate v_i, we follow a method given by Müller, Charypar, and Gross (2003) where the mass m_i and the density ρ_i of a particle (called "phyxel" in Müller, Keiser, et al. (2004)) are used:

$$v_i = \frac{m_i}{\rho_i} \tag{3.12}$$

Let $V(\boldsymbol{X})$ be the set of particles in the neighborhood of $\boldsymbol{X}$, i.e., $i \in V(\boldsymbol{X}) \Leftrightarrow \boldsymbol{X} \in \Omega_i$. Following the SPH method, the density of a particle is stated as:

$$\rho_i^h = \rho^h(\boldsymbol{X}_i) = \sum_{j \in V(\boldsymbol{X}_i)} m_j W_j(\boldsymbol{X}_i) \tag{3.13}$$

The remaining step required to approximate the volume is to formulate a measure of a particle's mass. For that, we first find the average r_i of the distance between a particle i and its k nearest neighbors, i.e., $r_i = \frac{1}{\|V(\boldsymbol{X}_i)\|}\sum_{j\in V(\boldsymbol{X}_i)}(\|\boldsymbol{X}_i - \boldsymbol{X}_j\|)$. Let ρ represent the material density of the simulated object. We pose $m_i = sr_i^d\rho$ with d the dimension of the system (i.e., 1D, 2D or 3D) and s a factor allowing ρ_i to get close to ρ. We find s by minimizing the error $e = \sum_i(\rho_i - \rho)^2$:

$$\begin{aligned}\frac{de}{ds} &= \frac{d}{ds}\sum_i(\rho_i - \rho)^2 \\ &= \frac{d}{ds}\sum_i((\sum_{j\in V(\boldsymbol{X}_i)} sr_j^d\rho W_{ij}) - \rho)^2\end{aligned}$$

where $W_{ij} = W_j(\boldsymbol{X}_i)$. Setting $\frac{de}{ds} = 0$, we find

$$s = \frac{\sum_i a_i}{\sum_i a_i^2}$$

with $a_i = \sum_{j\in V(\boldsymbol{X}_i)} r_j^d W_{ij}$.

This numerical scheme allows us to integrate the elastic potential energy to finally get a formulation of the internal force vector $\boldsymbol{R}$, and its jacobian, the tangent stiffness matrix $\mathbf{K}$. We reuse here the operator $\biguplus$ introduced in section 2.6 which denotes that an assembly process takes place. For an hyperelastic material, we have

$$\boldsymbol{R} = \biguplus_i \sum_{j\in V(\boldsymbol{X}_i)} \boxed{v_i\mathbf{F}_i\mathbf{S}_i\cdot\boldsymbol{\nabla}_{\boldsymbol{X}}\phi_j} \tag{3.14}$$

and

$$\mathbf{K} = \biguplus_i \sum_{(k,l)\in V(\boldsymbol{X}_i)} \boxed{v_i \underbrace{[(\nabla_{\boldsymbol{X}}^{\intercal}\phi_k)\mathbf{S}(\nabla_{\boldsymbol{X}}\phi_l)\boldsymbol{I} + \mathbf{B}_k^{\intercal}\mathbb{C}\mathbf{B}_l]}_{\mathbf{K}_{kl}}} \tag{3.15}$$

where $\mathbf{F}_i$, $\mathbf{S}_i$ and $\mathbf{B}_i$ are respectively the deformation gradient tensor, the second Piola-Kirchhoff stress tensor and the Green-Lagrange variational matrix evaluated

at the current position $\boldsymbol{x}_i$, and $\mathbb{C} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}}$. These quantities have been introduced in the chapter 2.

The next step consists in formulating both the shape function $\phi_i(\boldsymbol{X}) \rightarrow \Re$ of a particle i, and its derivatives with respect to its material coordinates $\nabla_{\boldsymbol{X}} \phi_i(\boldsymbol{X}) \rightarrow \Re^d$. We implemented two shape functions which we will now describe.

3.3.2 Shape function: SPH approximation

The first shape function we have implemented is based on the Smoothed Particle Hydrodynamics (SPH) method introduced by Lucy (1977) and Monaghan (1992) for astrophysics simulations, and brought to the field of deformable simulations by Desbrun and Gascuel (1996), Müller, Keiser, et al. (2004) and Solenthaler, Schläfli, and Pajarola (2007b).

With this approach, the approximation of the displacement field $\boldsymbol{u}(\boldsymbol{X})$ based on the SPH reads as follows:

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{X}) &= \sum_{i \in V(\boldsymbol{X})} \frac{m_i}{\rho_i} W_i(\boldsymbol{X}) \cdot \boldsymbol{u}_i \\ &= \sum_{i \in V(\boldsymbol{X})} v_i W_i(\boldsymbol{X}) \cdot \boldsymbol{u}_i \\ &= \sum_{i \in V(\boldsymbol{X})} \phi_i(\boldsymbol{X}) \cdot \boldsymbol{u}_i \end{aligned} \tag{3.16}$$

with v_i is the volume of the particle i defined in the last section, and is assumed constant throughout the simulation. The derivatives of the approximation with respect of the material coordinates then becomes:

$$\begin{aligned} \boldsymbol{\nabla} \boldsymbol{u}(\boldsymbol{X}) &= \sum_{i \in V(\boldsymbol{X})} v_i \boldsymbol{\nabla} W_i(\boldsymbol{X}) \cdot \boldsymbol{u}_i \\ &= \sum_{i \in V(\boldsymbol{X})} \boldsymbol{\nabla} \phi_i(\boldsymbol{X}) \cdot \boldsymbol{u}_i \end{aligned} \tag{3.17}$$

The displacement gradient is normally computed on the numerical integration points. With point-based methods, particles represents both the degrees of freedom and the integration points, hence many authors prefer to compute the elastic potential based on a displacement gradient of a locally undeformed configuration.

Instead of taking the displacement between the current position $\boldsymbol{x}_i$ of a particle and its undeformed position $\boldsymbol{X}_i$, they use the displacement of the particle relative to its closest neighbors. In three-dimensional, this yields:

$$\begin{aligned}
\boldsymbol{\nabla}\mathbf{u}(\boldsymbol{X}_i) = \boldsymbol{\nabla}\mathbf{u}_i &= \begin{bmatrix} u_{i,X} & u_{i,Y} & u_{i,Z} \\ v_{i,X} & v_{i,Y} & v_{i,Z} \\ w_{i,X} & w_{i,Y} & w_{i,Z} \end{bmatrix} \\
&\approx \sum_{j \in V(\boldsymbol{X}_i)} \begin{bmatrix} \phi_{j,X}(u_j - u_i) & \phi_{j,Y}(u_j - u_i) & \phi_{j,Z}(u_j - u_i) \\ \phi_{j,X}(v_j - v_i) & \phi_{j,Y}(v_j - v_i) & \phi_{j,Z}(v_j - v_i) \\ \phi_{j,X}(w_j - w_i) & \phi_{j,Y}(w_j - w_i) & \phi_{j,Z}(w_j - w_i) \end{bmatrix} \\
&= \sum_{j \in V(\boldsymbol{X}_i)} \begin{pmatrix} u_j - u_i \\ v_j - v_i \\ w_j - w_i \end{pmatrix} [\phi_{j,X} \ \phi_{j,Y} \ \phi_{j,Z}] \\
&= \sum_{j \in V(\boldsymbol{X}_i)} \begin{pmatrix} u_j - u_i \\ v_j - v_i \\ w_j - w_i \end{pmatrix} \begin{pmatrix} v_i W_{ij,X} \\ v_i W_{ij,Y} \\ v_i W_{ij,Z} \end{pmatrix}^T
\end{aligned}$$

with $\boldsymbol{u}_i = \begin{pmatrix} u_i \\ v_i \\ w_i \end{pmatrix}$, $u_{i,X} = \frac{du(\boldsymbol{X}_i)}{dX}$ and $W_{ij,X} = \frac{dW_j(\boldsymbol{X}_i)}{dX}$.

3.3.3 Shape function: MLS approximation

The second shape function we have implemented is based on the Moving Least Squares (MLS) approximation proposed in Müller, Keiser, et al. (2004). Here, starting from the scalar components $u(\boldsymbol{X})$, $v(\boldsymbol{X})$ and $w(\boldsymbol{X})$ of the displacement field $\boldsymbol{u}(\boldsymbol{X}) = [u, v, w]^\intercal$, a first order Taylor decomposition is done in the neighborhood of a particle i located at the material coordinate $\boldsymbol{X}_i$:

$$u(\boldsymbol{X}_i + \Delta\boldsymbol{X}) = u_i + \nabla_{\boldsymbol{X}} u_i \cdot \Delta X + O(\|\Delta X\|^2)$$

Then, for any particle j neighbors to i (i.e., $j \in V(\boldsymbol{X}_i)$), we have

$$\tilde{u}_j = u_i + (\boldsymbol{X}_j - \boldsymbol{X}_i)^T \nabla_{\boldsymbol{X}} u_i$$

Using a weighted least squares method, we minimize the approximation error $e = \tilde{u} - u$ with the usual kernel $W_{ij} = W_j(\boldsymbol{X}_i)$:

$$\nabla_{\boldsymbol{X}} e = \nabla_{\boldsymbol{X}} \sum_j (\tilde{u}_j - u_j)^2 W_{ij} = 0 \tag{3.18}$$

which yields the displacement gradient approximation at a particle i:

$$\begin{aligned} \nabla_{\boldsymbol{X}} u_i &= \left(\underbrace{\sum_j (\boldsymbol{X}_j - \boldsymbol{X}_i)(\boldsymbol{X}_j - \boldsymbol{X}_i)^T W_{ij}}_{\mathbf{A}} \right)^{-1} \sum_j (u_j - u_i)(\boldsymbol{X}_j - \boldsymbol{X}_i) W_{ij} \\ &= \sum_j [\mathbf{A}^{-1} \boldsymbol{X}_{ij} W_{ij}] u_{ij} \\ &= \sum_j [\nabla_{\boldsymbol{X}} \phi_j] u_{ij} \end{aligned} \tag{3.19}$$

Here, $\mathbf{A}$ is called the moment matrix and $\nabla_{\boldsymbol{X}} \phi_j = [\phi_{j,x}\ \phi_{j,y}\ \phi_{j,z}]^T$ is the gradient of the shape function associated with particle j. As it is usually done with a Lagrangian description, the inverse of the moment matrix can be precomputed before the simulation begins.

While this method yields a shape function of 1 degree higher than the SPH shape function, the fact that it relies on the inverse of a moment matrix means that it is dependent on the configuration of a neighborhood of particles around a point. Indeed, if the particles have either a co-linear or coplanar shape, $\mathbf{A}$ will become singular and its inversion will be impossible. Even if the configuration is not completely co-linear or coplanar, if it sufficiently close to it, $\mathbf{A}$ will become ill-conditioned. In those cases, we can rely on the stable Singular Value Decomposition (SVD) method proposed in Press et al. (1992). Note that using such decomposition is not a perfect solution, but it does reduce a little bit the numerical impact of bad neighborhood configurations.

The final formulation of the displacement gradient $\boldsymbol{\nabla}_{\boldsymbol{X}} \boldsymbol{u}_i$ at particle i is then expressed for a three-dimensional domain:

$$
\begin{aligned}
\boldsymbol{\nabla}(\boldsymbol{X_i}) = \boldsymbol{\nabla}\mathbf{u}_i &= \begin{bmatrix} u_{i,X} & u_{i,Y} & u_{i,Z} \\ v_{i,X} & v_{i,Y} & v_{i,Z} \\ w_{i,X} & w_{i,Y} & w_{i,Z} \end{bmatrix} \\
&\approx \sum_{j \in V(\boldsymbol{X}_i)} \begin{bmatrix} \phi_{j,X}(u_j - u_i) & \phi_{j,Y}(u_j - u_i) & \phi_{j,Z}(u_j - u_i) \\ \phi_{j,X}(v_j - v_i) & \phi_{j,Y}(v_j - v_i) & \phi_{j,Z}(v_j - v_i) \\ \phi_{j,X}(w_j - w_i) & \phi_{j,Y}(w_j - w_i) & \phi_{j,Z}(w_j - w_i) \end{bmatrix} \\
&= \sum_{j \in V(\boldsymbol{X}_i)} \begin{pmatrix} u_j - u_i \\ v_j - v_i \\ w_j - w_i \end{pmatrix} [\phi_{j,X} \ \phi_{j,Y} \ \phi_{j,Z}] \\
&= \sum_{j \in V(\boldsymbol{X}_i)} \begin{pmatrix} u_j - u_i \\ v_j - v_i \\ w_j - w_i \end{pmatrix} [\mathbf{A}_i{}^{-1} \boldsymbol{X}_{ij} W_{ij}]^T
\end{aligned} \tag{3.20}
$$

3.3.4 Discussion

The PBA approach presented in this section results in a method that relies completely on particles, both for the displacement field approximation and the numerical integration. This allows for a very quick and easy setup of the simulations, i.e., there is no need for a complex mesh generation. It is also quite straightforward to extend the model to cutting simulations, as we only need to adjust the weight function to remove the influence a particle has on a neighbor if the latter is located on the other side of a cut.

There is, however, a strong limitation inherent to this method: its numerical integration is based on a rough approximation of the volume occupied by a particle. This strongly impacts the accuracy of the solution. This has been observed in our experimental runs, the results of which will be described in section 3.11. In fact, the PBA method is often referred to as a visually plausible method, which means that its solution *looks* good, but might be far from the real mechanical solution. This might not be a problem for computer graphic applications such as animation movies or video games, but it is definitely not suitable for accurate-dependent applications such as surgery simulations.

3.4 Meshless Approximation Mesh-Based Integration

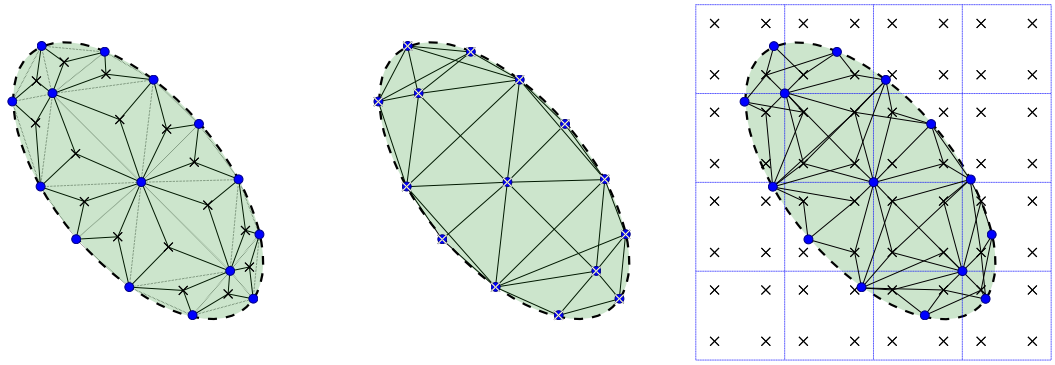
The second meshless method that we have implemented aims at reducing the integration error and the weak consistency of the shape function inherent to the PBA model. It is based on the Meshless Total Lagrangian Explicit Dynamics (MTLED) method presented in Horton et al. (2010). Here, however, we propose an extension of the MTLED that uses the corotational elasticity model and implicit time integration. We will denote our method as the Meshless Approximation Mesh-Based Integration (MLAMBI) method.

Unlike the PBA method that relies on a point-based volume approximation for its numerical integration, here we use a background grid made of regular hexahedral elements that is simply placed on top of the particles cloud. Hence, one could argue that this method is not 100% meshless. However, using the background grid does get rid of the difficult step associated with the creation of a conforming mesh made of well-formed elements, which is the main drawback of FE methods. In addition, the displacement field approximation used here is still based on meshless shape functions. But this time, we will use adaptive shape functions with higher-order basis that, similarly to high order elements in FE methods, enhance the consistency of the approximation.

With our MLAMBI method, the particles only represent the degrees of freedom, and the integration points are the standard gaussian quadrature points of the background grid elements. While this integration scheme is similar to that of the FE methods, the displacement field approximation at the Gauss points is formulated with respect to their closest particles which, unlike FEM, are independent of the integration mesh. Integration points that are found outside of the boundaries of the simulated object are simply discarded. Differences between the PBA method, our MLAMBI method and the FE method are illustrated at figure 3.5.

3.4.1 Shape function: MLS approximation

Similarly to the PBA method, we need to construct the shape functions for every particle in order to approximate the displacement field anywhere on the domain. While an SPH based shape function could have been used, we decided to follow the approach proposed by Horton et al. (2010) where shape functions are built using an Moving Least Squares (MLS) method. Unlike the MLS presented in section 3.3.3, here the approximation isn't evaluated directly onto a particle, but at any position in the domain. Hence instead of being used to build an approximation



(a) Finite element method (b) Point-based method (c) MLAMBI method

Figure 3.5: Relations between the integration points (crosses) and the DOF nodes (circles) where the green oval represents the simulated object. With the FE method, each triangular element has one integration point linked to its geometrical nodes. For the PBA method, each node represents both an integration point and a DOF node. They use their closest neighbors as DOFs. For the MLAMBI method, the background two-dimensional quad elements have their usual four integration nodes, however, each integration point uses its closest particles as DOFs.

of the displacement *gradient* field, the MLS is used to approximate directly the displacement field. Without loss of generality , we can split the displacement vector field $\boldsymbol{u}(\boldsymbol{X})$ into three scalar fields, namely $u(\boldsymbol{X})$, $v(\boldsymbol{X})$ and $w(\boldsymbol{X})$, with $\boldsymbol{X}$ being the material coordinates of a position in Ω_0. The approach to approximate each scalar is the same. Using $u(\boldsymbol{X})$, we have:

$$u^h(\boldsymbol{X}) = \sum_{i=0}^{m} p_i(\boldsymbol{X}) a_i(\boldsymbol{X}) = \boldsymbol{P}^T(\boldsymbol{X})\boldsymbol{a}(\boldsymbol{X}) \tag{3.21}$$

where p_i is a monomial basis function, and a_i its coefficient that is discussed just below. As suggested by Horton et al. (2010), a good starting point for the basis functions is to choose m monomials in Pascal's pyramid:

$$\boldsymbol{P}^T(\boldsymbol{X}) = [1|X\ Y\ Z|XY\ XZ\ YZ|X^2\ Y^2\ Z^2]$$

For the type of applications we are interested in, we usually choose m monomials, typically the 1, 4, 7 or 10 first monomials starting from the top of the pyramid. For example, with $m = 4$ the basis functions $[1, X, Y, Z]$ are selected. When

$m = 1$, the resulting approximation order is equivalent to that of the SPH method. Theoretically, we could use a more complex scheme, by choosing for example basis functions outside of Pascal's pyramid that better fit the solution's space.

To get the coefficients $a_i(\boldsymbol{X})$, the least squares method weighted by a kernel $W(\boldsymbol{X})$ (equations 3.3 to 3.7) in the neighborhood of $\boldsymbol{X}$ is used to minimize the approximation error:

$$\frac{\partial}{\partial \boldsymbol{a}} \left[\sum_{i \in V(\boldsymbol{X})} W_i(\boldsymbol{X}, h_i)(u^h(\boldsymbol{X}) - u(\boldsymbol{X}))^2 \right] = 0 \tag{3.22}$$

where h_i is found using one of the three methods described in section 3.2. By isolating the coefficients from equation (3.22), we get

$$\boldsymbol{a}(\boldsymbol{X}) = \mathbf{A}^{-1}(\boldsymbol{X}) \sum_{i \in V(\boldsymbol{X})} W_i(\boldsymbol{X}, h_i) \boldsymbol{P}(\boldsymbol{X}_i) u_i \tag{3.23}$$

where $\mathbf{A}(\boldsymbol{X}) = \sum_{i \in V(\boldsymbol{X})} W_i(\boldsymbol{X}) \boldsymbol{P}(\boldsymbol{X}_i) \boldsymbol{P}^T(\boldsymbol{X}_i)$ is the moment matrix at position $\boldsymbol{X}$ and is constant throughout the simulation.

Substituting equation (3.23) into (3.21) yields

$$\begin{aligned} u^h(\boldsymbol{X}) &= \sum_{i \in V(\boldsymbol{X})} \boldsymbol{P}(\boldsymbol{X}) \mathbf{A}^{-1}(\boldsymbol{X}) W_i(\boldsymbol{X}, h_i) \boldsymbol{P}(\boldsymbol{X}_i) u_i \\ &= \sum_{i \in V(\boldsymbol{X})} \phi_i(\boldsymbol{X}) u_j \end{aligned} \tag{3.24}$$

where $\phi_i(\boldsymbol{X}) = \boldsymbol{P}(\boldsymbol{X}) \mathbf{A}^{-1}(\boldsymbol{X}) W_i(\boldsymbol{X}) \boldsymbol{P}(\boldsymbol{X}_i)$ is the shape function of particle i evaluated at material coordinates $\boldsymbol{X}$.

Since the deformation tensor $\mathbf{F}$ depends on the displacement gradient $\boldsymbol{\nabla u}$, the derivatives of the shape function with respect to material coordinates X, Y and Z have to be calculated. We present here the equation for the derivative of the first scalar but again, the process is exactly the same for the v and w components.

$$\begin{aligned}\phi_{i,X}(\boldsymbol{X}) &= [\boldsymbol{P}_{,X}\mathbf{A}^{-1}W_i + \boldsymbol{P}(\mathbf{A}^{-1}_{,X}W_i + \mathbf{A}^{-1}W_{i,X})]\boldsymbol{P}_i \\ &= [\boldsymbol{P}_{,X}\mathbf{A}^{-1}W_i + \boldsymbol{P}((\mathbf{A}^{-1}\mathbf{A}_{,X}\mathbf{A}^{-1})W_i + \mathbf{A}^{-1}W_{i,X})]\boldsymbol{P}_i \\ &= [\boldsymbol{P}_{,X}\mathbf{A}^{-1}W_i + \boldsymbol{P}((\mathbf{A}^{-1}\left[\sum_{j\in V(\boldsymbol{X})} W_{j,X}\boldsymbol{P}_i\boldsymbol{P}_i^T\right]\mathbf{A}^{-1})W_i + \mathbf{A}^{-1}W_{i,X})]\boldsymbol{P}_i\end{aligned} \tag{3.25}$$

Here, the consistency of the approximation depends on the degree of the basis functions $\boldsymbol{P}$ and two factors must be considered. Firstly, if m is the number of basis functions in $\boldsymbol{P}$, and n is the number of particles in the neighborhood of $\boldsymbol{X}$, then n must be greater or equal to m for the moment matrix to be non-singular. In Horton et al. (2010), authors estimate that n should double the quantity of basis function, $n \approx 2m$. Secondly, an integration point must also have more than one particle support, or else it might receive too much stress.

3.5 Linear elasticity

Similar to FE methods, and because the continuous integration terms of the potential energy are discretized into a sum over the integration points, the continuous system is now reduced to a set of smaller systems of equations to be solved for each neighborhood around an integration element (or particle in the case of the PBA method). In chapter 2, we have seen that when a non-linear relationship between the stress tensor and the displacement gradient is used, a set of Newton–Raphson (NR) iterations have to be executed where the tangent stiffness matrix $\mathbf{K}$ has to be re-evaluated and inverted at each iteration. This can become quite expensive has the number of particles or integration points grows.

If we were to use the small strain assumption, which implies that the deformations of the simulated body are expected to remain very small, then the stress tensor would become linear in $\boldsymbol{u}$, yielding

$$\int_{\Omega_0} \mathbf{P} : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w}dv \approx \int_{\Omega_0} \boldsymbol{\sigma} : \boldsymbol{\nabla}_{\boldsymbol{X}}\boldsymbol{w}dv \tag{3.26}$$

where $\boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\epsilon})\boldsymbol{I} + 2\mu\boldsymbol{\epsilon}$ is the *Cauchy stress tensor* and $\boldsymbol{\epsilon} = \frac{1}{2}(\mathbf{F} + \mathbf{F}^\intercal)$ is the *small strain tensor*. The discrete elastic residual vector $\boldsymbol{R}$ of equation (3.14) then becomes

$$\boldsymbol{R} = \biguplus_i \sum_{j \in V(\boldsymbol{X}_i)} v_i \boldsymbol{\sigma}_i \cdot \boldsymbol{\nabla}_{\boldsymbol{X}} \phi_j \tag{3.27}$$

for the PBA method, and

$$\boldsymbol{R} = \biguplus_e \sum_{I \in \text{int}(e)} \sum_{i \in V(\boldsymbol{X}_I)} w_I \boldsymbol{\sigma}_i \cdot \boldsymbol{\nabla}_{\boldsymbol{X}} \phi_i det\mathbf{J}_I \tag{3.28}$$

for the MLAMBI method with $\text{int}(e)$ the set of integration points of element e, w_I the Gaussian weight of the integration point I, and $\mathbf{J}_I$ the jacobian of the transformation $T : \Omega_e \to \Omega_0$.

By deriving the elastic residual with respect to the displacement, we get the tangent stiffness matrix $\mathbf{K}$

$$\mathbf{K} = \biguplus_i \sum_{(k,l) \in V(\boldsymbol{X}_i)} \underbrace{v_i \left[\mathbf{B}_{Lk}^{\intercal} \mathbb{C} \mathbf{B}_{Ll}\right]}_{\mathbf{K}_{kl}} \tag{3.29}$$

for the PBA method, and

$$\mathbf{K} = \biguplus_e \sum_{I \in \text{int}(e)} \sum_{(i,j) \in V(\boldsymbol{X}_I)} \underbrace{w_I \left[\mathbf{B}_{Li}^{\intercal} \mathbb{C} \mathbf{B}_{Lj}\right] det\mathbf{J}_I}_{\mathbf{K}_{ij}} \tag{3.30}$$

for the MTLED method. Here, $\mathbf{B}_{Li}$ is the linear part of the deformation Green-Lagrange strain tensor variation $\delta\mathbf{E}$ defined in equation (2.16), and

$$\mathbb{C} = \frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{F}} = \lambda(\boldsymbol{I} \otimes \boldsymbol{I}) + 2\mu(\boldsymbol{I} \underline{\otimes} \boldsymbol{I}) = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{3.31}$$

is the material matrix.

The tangent stiffness matrix $\boldsymbol{K}$ is linear in $\boldsymbol{u}$, and can therefore be precomputed and inverted before the simulation starts. If no other non-linear terms are used in

the simulation, the system can be solved directly without requiring any NR iterations. This is, of course, the ideal scenario as it will enable very fast computation of each time steps. However, in the next section, we will see that the small strain assumption made here can present strong numerical artifacts.

Note that, here, the use of a constant tangent stiffness matrix is no different than with tradition FE methods. However, depending on the kernel support and, thereby, the number of particles in each neighborhood, the resulting matrix might become much denser than its FE counterpart using the same number of degrees of freedom, hence resulting in a more computationally expensive method. This is a non-negligible drawback of the meshless methods in general [Belytschko et al. (1996)].

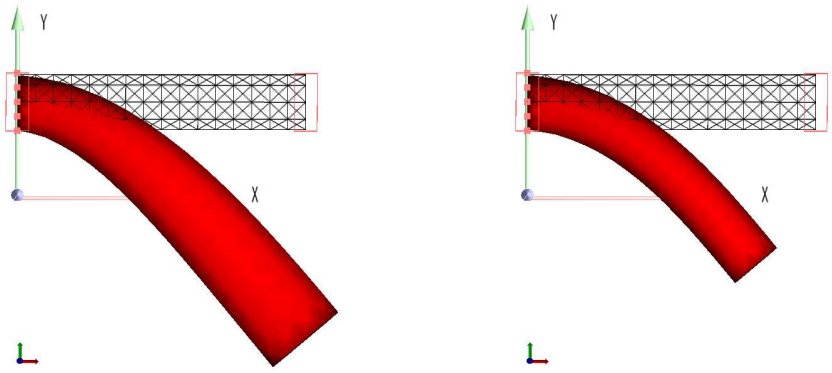
3.6 Corotational elasticity

Normally, a rigid transformation (translation or rotation) of a simulated body should not generate any elastic force since there is no strain involved. Indeed, as we have seen in section 2, the deformation tensor $\mathbf{F}$ is defined as the product of an orthogonal tensor $\mathbf{R}$ (representing the rotation part of the deformation) and a symmetric tensor $\mathbf{U}$ (representing the "changes of shape" part of the deformation), i.e., $\mathbf{F} = \mathbf{RU}$. When using the non-linear Green-Lagrangian deformation tensor, the rotational part of $\mathbf{F}$ cancels out, leaving

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^\intercal\mathbf{F} - \mathbf{I}) = \frac{1}{2}(\mathbf{U}^\intercal\mathbf{R}^\intercal\mathbf{RU} - \mathbf{I}) = \frac{1}{2}(\mathbf{U}^2 - \mathbf{I})$$

However, when a small strain assumption is made, the linear small strain tensor $\boldsymbol{\epsilon} = \frac{1}{2}(\mathbf{F}+\mathbf{F}^\intercal)$ does not have such property. This is because the small strain assumption intrinsically assumes an infinitesimal rotation. Hence, using this assumption for a simulation that has small deformations, but large rotation (like the deformation of a bent beam), will produce incorrect elastic forces as the rotations are viewed themselves as strain forces. These incorrect forces are often called *ghost forces* and are usually quickly identified: as the simulated object rotates, a large volume expansion takes place at the same time (see figure 3.6a).

There has been a lot of research on this numerical phenomenon, especially for real-time applications since the small strain assumption is computationally very efficient and therefore quite desirable. One solution consists of using corotational



(a) Ghost forces prevent the cylinder from rotating correctly.

(b) The corotational approach alleviates the effects of ghost forces by removing rotations before the computation of elastic forces.

Figure 3.6: Simulation of a cylinder fixed at the left and deformed by gravity (downward along the Y-axis).

elasticity models. This approach has been documented in the context of FE simulations in Carlos A Felippa (2000); Müller, Dorsey, et al. (2002); Carlos A. Felippa and Haugen (2005).

The general idea behind corotational methods is to extract the rotational part $\boldsymbol{R}$ of the displacement before computing the internal elastic forces, leaving only the part of the displacement that induces a change of shape. The tangent stiffness matrix then becomes

$$\mathbf{K} = \boxed{\biguplus_i \sum_{(k,l)\in V(\boldsymbol{X}_i)} \underbrace{v_i \left[\boldsymbol{R}_i \mathbf{B}_{Lk}^\intercal \mathbb{C} \mathbf{B}_{Ll} \boldsymbol{R}_i^\intercal\right]}_{\mathbf{K}_{kl}}} \tag{3.32}$$

for the PBA method, and

$$\mathbf{K} = \boxed{\biguplus_e \sum_{I\in \mathrm{int}(e)} \sum_{(i,j)\in V(\boldsymbol{X}_I)} \underbrace{w_I \left[\boldsymbol{R}_I \mathbf{B}_{Li}^\intercal \mathbb{C} \mathbf{B}_{Lj} \boldsymbol{R}_I^\intercal\right] det\mathbf{J}_I}_{\mathbf{K}_{ij}}} \tag{3.33}$$

for the MLAMBI method. Hence the rotational part is removed from the displacement before computing the Cauchy stress tensor, and the resulting stiffness

(or internal elastic force) is rotated back afterward, avoiding any numerical ghost forces as we illustrated in figure 3.6b.

The most natural way to extract the rotation matrix $\boldsymbol{R}$ is to separate the two parts (rotational and the change of shape) of the deformation gradient tensor $\mathbf{F}$ via either a polar or Singular Value Decomposition (SVD) decomposition. However, when a neighborhood around an integration point approaches a co-linear or a coplanar configuration, the jacobian of the deformation tensor tends to zero. In this case, both the polar and the SVD models will struggle to decompose the tensor. To alleviate this numerical problem, we implemented the *stable* SVD method proposed in Press et al. (1992). However, when very high forces are exerted on the simulated body, we saw that at a certain time step, the solver can still produce a solution whereby the neighborhood is completely coplanar or co-linear, or even inverted. The jacobian of the deformation becomes zero or negative, and even the *stable* SVD method fails to decompose the tensor. When reaching this point, the simulation must abort as it can no longer recover from this erratic configuration. This problem has motivated some authors to propose alternative methods to extract a rotation matrix. For example, Nesme, Payan, and Faure (2005) proposed an approach for FE methods where a transformation matrix is built from three edges of an element:

$$\boldsymbol{A} = [\boldsymbol{e}_1^0 \ \boldsymbol{e}_2^0 \ \boldsymbol{e}_3^0]^{-1}[\boldsymbol{e}_1 \ \boldsymbol{e}_2 \ \boldsymbol{e}_3]$$

where each $\boldsymbol{e}_i$ is one edge around an element's node. The *stable* SVD method is then used to decompose $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and the rotation matrix becomes $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T$. This approach, however, will only produce a rough approximation of the rotation for the whole element, and might not be valid for some of the integration points that are far from the element's node chosen to build $\boldsymbol{A}$. It is also not usable for PBA methods since in this case, as no element are used for the integration of the internal elastic energy. In our work, we used a method based on Müller, Heidelberger, et al. (2005) which was made for shape matching applications, and later used by Becker, Ihmsen, and Teschner (2009b) for PBA applications. The method consists of building the transformation matrix $\boldsymbol{A}$ from the distance between a point cloud and its center of mass

$$\boldsymbol{A} = \left(\sum_i m_i(\boldsymbol{x}_i - \boldsymbol{x}_{\text{cm}})(\boldsymbol{x}_i^0 - \boldsymbol{x}_{\text{cm}}^0)^T\right)\left(\sum_i m_i(\boldsymbol{x}_i^0 - \boldsymbol{x}_{\text{cm}}^0)(\boldsymbol{x}_i^0 - \boldsymbol{x}_{\text{cm}}^0)^T\right)^{-1}$$

When substituting the center of mass by an integration point, the transformation matrix can then be reformulated as

$$\mathbf{A}_i = \sum_j m_j W_j(\boldsymbol{X}_i)(\boldsymbol{x}_j - \boldsymbol{x}_i)(\boldsymbol{X}_j - \boldsymbol{X}_i)^T \tag{3.34}$$

for the PBA method, and

$$\mathbf{A}_I = w_I \sum_j W_j(\boldsymbol{X}_I)(\boldsymbol{x}_j - \boldsymbol{x}_I)(\boldsymbol{X}_j - \boldsymbol{X}_I)^T \tag{3.35}$$

for the MLAMBI method. This transformation matrix is sufficiently robust for neighborhood inversions, and is also computationally efficient. A single rotation matrix can also be extracted at the center of the element and used to approximate the rotation for all integration points inside the same element to further reduce the computational time required to build the tangent stiffness matrix. This will usually yield better results than the technique of extracting the rotation at a single element's node as proposed in Nesme, Payan, and Faure (2005).

3.7 Surface mapping

Up until now, our discussion has been concentrated on the volumetric part of the simulated object. We will now discuss its boundary representation. This representation may be required to impose the boundary conditions on our models, or simply to display the object's surface to the end user. For an implicit surface representation such as a level-set iso function, an explicit surface tessellation is usually created before the simulation, using for example a marching cube algorithm. The resulting surface tessellation is normally a mesh made of triangular or quadrangular elements.

Let $\mathfrak{M}^s$ be a surface mesh containing n_s nodes. In order to approximate the displacement at any nodes in $\mathfrak{M}^s$, we use the shape functions of neighboring particles (figure 3.7). Hence, the same principle used for interpolating the displacement gradient at the location of an integration point is reused here for a surface mesh node. Let $\boldsymbol{X}_i^s$ be the position vector of the i$^{\text{th}}$ surface node, and $V(\boldsymbol{X}_i^s)$ the set of neighboring particles. The displacement vector at this surface node is given by

$$\boldsymbol{u}_i^s = \sum_{j \in V(\boldsymbol{X}_i^s)} \phi_j(\boldsymbol{X}_i^s)\boldsymbol{u}_j \tag{3.36}$$

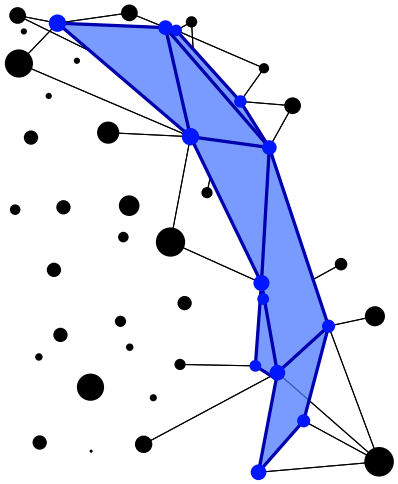


Figure 3.7: Three-dimensional mapping example of a surface mesh where the displacement of its nodes is approximated from the displacement of neighboring particles.

with ϕ_j the shape function evaluated from either the SPH method (equation 3.16) or the MLS method (equation 3.24).

In practice, we build a sparse and rectangular matrix $\boldsymbol{H}$ of size $3n \times 3n_s$ called the *mapping matrix* where n is the number of particles in the domain. Let $0 \leq i \leq n$ be the index of a particle neighbor to a surface node of index $0 \leq j \leq n_s$, i.e. $\boldsymbol{X}_i \in V(\boldsymbol{X}_j^s)$, we define

$$\boldsymbol{H}_{ij} = \mathbf{I}\phi_i(\boldsymbol{X}_j^s) \tag{3.37}$$

as the 3×3 mapping sub-matrix of $\boldsymbol{H}$. It is also sometimes more efficient to directly interpolate the position vector of the surface nodes. Let $\boldsymbol{P}$ be the $3n \times 1$ position vector of the particles. The $3n_s \times 1$ position vector $\boldsymbol{P}_s$ of the surface nodes is obtained with

$$\boldsymbol{P}_s = \boldsymbol{H}^\intercal \boldsymbol{P} \tag{3.38}$$

and can be computed efficiently with sparse matrix to dense vector multiplication routines.

Note that, unless topological changes occur, particle neighborhoods of the surface nodes and the resulting mapping matrix $\boldsymbol{H}$ can both be precomputed since they will not change during the simulation, thanks to the Lagrangian formulation of the system.

3.8 Neumann boundary condition

In classical Finite Element methods, the boundary of the discretized domain is represented directly by the faces of the elements lying on the surface of the mesh. Hence, the Neumann boundary conditions are applied by integrating the surface traction term of the weakened equation (2.11) on these faces. Here, however, the boundary is represented by a surface tesselation that is embedded inside the particle point cloud, as shown in figure 3.7. A specific scheme has to be built to project the tractive forces applied to the embedded mesh to the surrounding particles embedding it. This particularity is also found in Immersed boundary (IB) methods, which will be discussed in detail in chapter 4. The difference being that, here, the surface mesh is not embedded into a background mesh of elements, but is instead embedded inside a background point cloud.

The imposition of Neumann boundary condition is done in two steps. First, we accumulate a surface traction vector $\boldsymbol{T}^s$ of size $3n_s \times 1$ with

$$\boldsymbol{T}^s = \biguplus_{e_s} \sum_{i=1}^{n_s^e} \underbrace{\int_{\Omega_{e_s}} \boldsymbol{t} \cdot N_i^{e_s} \, d\Omega_{e_s}}_{\boldsymbol{t}_i^s} \tag{3.39}$$

where e_s is a surface element (for example, a triangle or quad) containing n_s^e nodes, and where $N_i^{e_s}$ is the i$^{\text{th}}$ shape function of e_s with respect to material coordinates.

Next, we propagate these tractive forces to neighboring particles using the same mapping matrix $\boldsymbol{H}$ defined earlier, leading to

$$\boldsymbol{T} = \boldsymbol{H}\boldsymbol{T}^s \tag{3.40}$$

where $\boldsymbol{T}$ is the global $3n \times 1$ traction vector of the system. Again, this can be pre-assembled before the simulation in case of a constant traction, or at each load increment otherwise.

3.9 Dirichlet boundary condition

For a number of reasons, the essential boundary conditions such as the strict imposition of Dirichlet displacements on parts of the boundary constitute a challenge with meshless methods. First, contrary to Finite Element (FE) methods, shape functions based on point clouds such as the SPH or the MLS presented in this chapter do not possess the Kronecker delta function property. This means that an approximation of a field function does not go through nodal values. In addition, and this was the case with the Neumann conditions in the last section, our boundary representation is immersed, and thereby is not represented explicitly by particles.

Various methods have been introduced to overcome these issues. Some will adapt the shape functions in order to recover the Kronecker delta property on boundary nodes [Gosz and W. K. Liu (1996); Chen et al. (1996); Mostt and Bucher (2005)]. Others will instead change the variational formulation in order to add new terms to enforce the Dirichlet conditions. This is the case, for example, with Lagrange multiplier based methods [Ted Belytschko, Y. Y. Lu, and L. Gu (1994)], Penalty based methods [Zhu and Atluri (1998)] or Nitsche's based methods [Griebel and Schweitzer (2003); Fernández-Méndez and Huerta (2004)]. Methods based on Lagrange multipliers are probably the most frequently used. They consist of adding a new set of unknown variables to the system, the *Lagrange multipliers*, that serve to enforce field values on boundary nodes. They, however, bring a non-negligible computational cost, especially when large portions of boundary nodes are present. Penalty methods, on the other hand, are quite efficient and work by adding a stiffness penalty force term. The stiffness factor is usually chosen to be very large and can produce an ill-conditioned system. Similarly, the Nitsche's method, also works by adding a penalty term, but this time the scalar value is chosen relative to each particular problem, and is usually not trivial to determine [Fernández-Méndez and Huerta (2004)]. Finally, some methods will simply blend their meshless approximation with a FEM discretization near the boundary region [T. Belytschko, Organ, and Y. Krongauz (1995); Günther and Wing Kam Liu (1998); Huerta and Fernández-Méndez (2000); G. Y. Zhang et al. (2014)].

For the models developed in this work, we first chose to explore a penalty based imposition of Dirichlet boundary conditions, as it is by far the simplest and quickest to implement. Throughout our experimentation, this method has proven to generate an accuracy that is more than acceptable and we did not observe any ill-conditioning problem. Given these results, we concluded that it was not necessary to spend more effort on alternative methods.

The steps to impose a penalty force on a Dirichlet boundary are similar to the surface traction process that was described in the last section. We first accumulate a surface force vector $\boldsymbol{S}^s$ (of size $3n_s \times 1$ for a surface mesh having n_s nodes). Let $\mathbb{S}_d$ be the set of surface nodes lying on a Dirichlet boundary. We define

$$\boldsymbol{S}^s = \biguplus_{i \in \mathbb{S}_d} k\mathbf{I} \left\| \boldsymbol{u}_i^k - \boldsymbol{u}^d \right\| \tag{3.41}$$

where k is the penalty parameter, $\boldsymbol{u}_i^k$ is the current displacement of the i$^{\text{th}}$ surface node, and $\boldsymbol{u}^d$ is the imposed displacement. This penalty can be seen as if we were attaching elastic springs of stiffness k between a surface node and its imposed position (see figure 3.8).

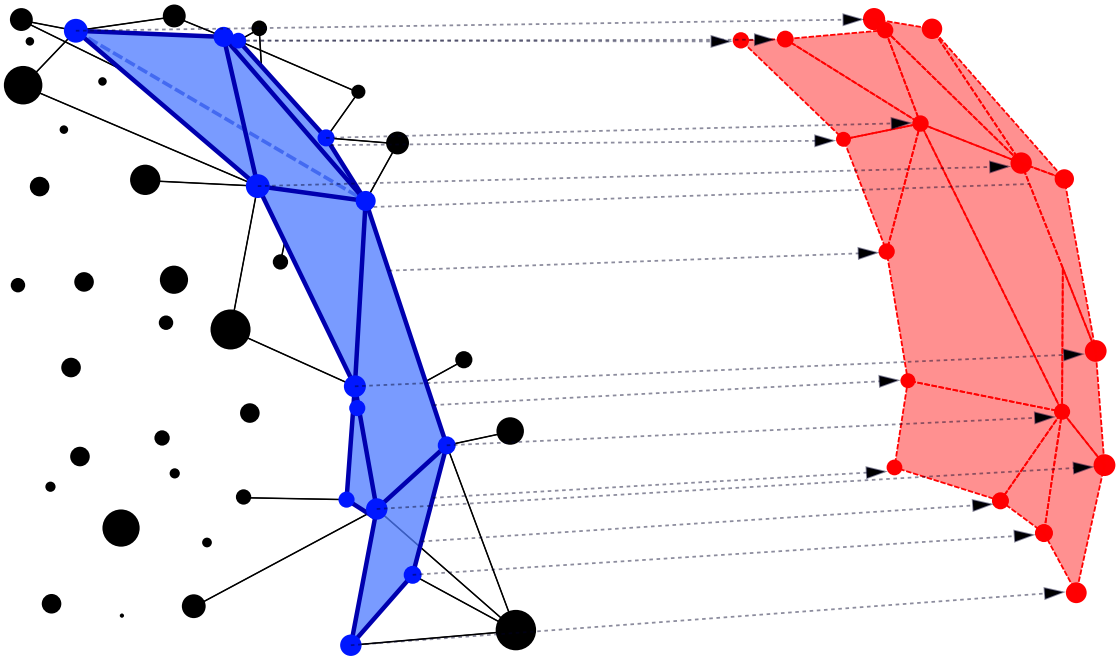


Figure 3.8: Penalty method where displacements are imposed on surface nodes that are embedded into surrounding particles.

To transfer the surface penalty forces back to the neighboring particles, we use the mapping system defined in section 3.8:

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{S}^s \tag{3.42}$$

This time, however, the forces are dependent on the current displacement, which means that in order to use the NR algorithm, the tangent stiffness matrix of those

forces is required. For the surface force, the derivation is straightforward

$$\boldsymbol{K}_s^s = \biguplus_{i \in \mathbb{S}_d} k\mathbf{I} \tag{3.43}$$

The stiffness contribution to the global matrix of the fictitious grid is again computed using the mapping matrix, i.e.:

$$\boldsymbol{K}_s = \boldsymbol{H}\boldsymbol{K}_s^s\boldsymbol{H}^\intercal \tag{3.44}$$

3.10 Solving the dynamic system

So far, we have described all the elements required to build the complete discrete system of equations needed to balance the linear momentum introduced in chapter 2. We have seen how the elastic forces are accumulated into a residual vector $\boldsymbol{R}$ that contains the internal nodal forces. Using a mapping between an immersed surface mesh and the internal particles of our meshless discretization, Neumann boundary conditions were imposed naturally using the integration of a traction function over the surface mesh. The displacement essential boundary conditions were replaced by natural boundary conditions using a penalty method. Combining all of these components together into the general equation (2.21), the system becomes as follows

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{D}\dot{\boldsymbol{u}} + \boldsymbol{R}(\boldsymbol{u}) + \boldsymbol{S}(\boldsymbol{u}) = \boldsymbol{B} + \boldsymbol{T}$$

where $\boldsymbol{M}$ and $\boldsymbol{D}$ are respectively the mass and damping matrix presented in section 2.6.

This problem can be written as a system of first order differential equations using

$$\boldsymbol{v} = \frac{d\boldsymbol{u}}{dt}, \quad \boldsymbol{a} = \frac{d\boldsymbol{v}}{dt}$$

where the acceleration $\boldsymbol{a}$ and the velocity $\boldsymbol{v}$ have been introduced as independent variable. Similarly to what we have done previously in discretizing a continuous domain into a finite set of subdomains, we can split a continuous function of time into a series of steps $t_0, \ldots, t_n$, called *time steps*. The previous system of equations is therefore replaced by a discrete system to be solved at a given time t_{i+1}:

$$\boldsymbol{M}\boldsymbol{a}_{i+1} + \boldsymbol{D}\boldsymbol{v}_{i+1} + \boldsymbol{R}(\boldsymbol{u}_{i+1}) + \boldsymbol{S}(\boldsymbol{u}_{i+1}) = \boldsymbol{B}_{i+1} + \boldsymbol{T}_{i+1} \tag{3.45}$$

where the indices i denotes that a quantity is taken at the time step t_i.

To solve this discrete equation, we have to rely on numerical methods. These can be classified into two distinct branches. *Explicit methods* are numerical procedures which give the solution of equation (3.45) at a time step t_{i+1} using only quantities computed at the time step t_i. When both the mass matrix and the damping matrix are diagonal, i.e., when using lumped matrices, explicit methods are very computationally efficient. The *central difference* scheme is a popular example of such explicit method, and is the one we have implemented. Posing $\boldsymbol{v}_{i+1}$ and $\boldsymbol{a}_{i+1}$ as

$$\begin{aligned}\boldsymbol{v}_{i+1} &= \frac{\boldsymbol{u}_{i+1} - \boldsymbol{u}_{i-1}}{2\Delta t} \\ \boldsymbol{a}_{i+1} &= \frac{\boldsymbol{u}_{i+1} - 2\boldsymbol{u}_i + \boldsymbol{u}_{i-1}}{(\Delta t)^2}\end{aligned}$$

and inserting them into the equation (3.45) gives the following linear system of equations:

$$\begin{aligned}(\boldsymbol{M} + \frac{\Delta t}{2}\boldsymbol{D})\boldsymbol{u}_{i+1} = (\Delta t)^2(\boldsymbol{B}_i + \boldsymbol{T}_i - \boldsymbol{R}(\boldsymbol{u}_i) - \boldsymbol{S}(\boldsymbol{u}_i)) \\ + \frac{\Delta t}{2}\boldsymbol{D}\boldsymbol{u}_{i-1} + \boldsymbol{M}(2\boldsymbol{u}_i - \boldsymbol{u}_{i-1})\end{aligned} \tag{3.46}$$

where $\Delta t = t_{i-1} - t_i$ is the time between two time steps.

Since both the mass matrix and the damping matrix are constant, the left-hand side component can be pre-inverted and only the right-hand side of the equation has to be computed at each time step.

Explicit methods present a disadvantage as very small time steps are required to generate a stable solution (one of our four key requirements). Conversely, the second branch of numerical methods to solve equation (3.45), the *implicit methods*, is unconditionally stable and therefore can handle larger time steps. Here, the solution of the system depends on quantities at the current time step t_i as well as

the unknown time step t_{i+1}. A popular choice of implicit method is the Newmark method, where the following approximations are made:

$$\begin{aligned}\boldsymbol{u}_{i+1} &= \boldsymbol{u}_i + \Delta t \boldsymbol{v}_i + \frac{(\Delta t)^2}{2}\left[(1-2\beta)\boldsymbol{a}_i + 2\beta \boldsymbol{a}_{i+1}\right] \\ \boldsymbol{v}_{i+1} &= \boldsymbol{v}_i + \Delta t\left[(1-\gamma)\boldsymbol{a}_i + \gamma \boldsymbol{a}_{i+1}\right]\end{aligned}$$

where $0 \leq \beta \leq 0.5$ and $0 \leq \gamma \leq 1$ are constant scalar parameters. In this work, we have used $\beta = 0$ and $\gamma = \frac{1}{2}$ which derives to the central difference formulation.

The system to be solved hence becomes

$$(\boldsymbol{M} + \gamma \Delta t \boldsymbol{D})\boldsymbol{a}_{i+1} + \boldsymbol{R}(\boldsymbol{u}_{i+1}) + \boldsymbol{S}(\boldsymbol{u}_{i+1}) - \boldsymbol{B}_{i+1} - \boldsymbol{T}_{i+1} = 0$$

When the elastic residual component $\boldsymbol{R}(\boldsymbol{u}_{i+1})$ is non-linear, the Newton–Raphson (NR) algorithm presented in section 2.7 has to be used.

3.11 Results

One of the main interests throughout this work was, of course, to establish how the meshless approach described in this chapter positions itself with respect to the traditional FE method. We therefore performed a series of experiments on different scenarios using the PBA method with the SPH and MLS shape functions described earlier. We also repeated these experiments with our MLAMBI method.

All our experiments were implemented within the Simulation Open Framework Architecture (SOFA) framework [Allard et al. (2007)]. The implementation of our meshless models has been made as a SOFA plugin. No multithreading maneuvers were used, hence leaving the place for future speed improvements. Computation times are given and were measured on an Intel(R) Core(TM) i7-6700K CPU @ 4.00 GHz computer with 16 GB of memory.

Performance of the PBA method

We ran a series of experiments with the Point-Based Animation (PBA) method using either an explicit or implicit Euler time integration scheme. The behavior of the method was evaluated with respect to two simulation scenarios: stretching and bending.

Stretching simulations

For this first scenario, the experiment consisted of observing the effect of gravity on a simulated parallelepiped beam having its top face fixed. The particle placement was done using a lattice grid of $10 \times 40 \times 10$ nodes. The simulated material was Saint-Venant-Kirchhoff with a mass density of 6.23, Young's modulus of 2000 [Pa] and a Poisson coefficient of 0.3. Time integration was done using a Central Difference scheme with a time step of 5 milliseconds for a total of 3 seconds of simulation. Results of the simulation with the SPH approximation method are shown in figure 3.9 and are compared against a 4-nodes tetrahedron FE method using the same number of nodes.

The results of this first experiment illustrates one of the main issues with the PBA method. Here, we can see that using only 10 neighbors per particles translates to a very weak force propagation. As the neighborhood increases, the simulated beam is able to stretch a bit more, getting a little closer to the reference solution. However, the low order of the approximation is restraining the beam from getting to its complete stretched shape. Furthermore, having too many particles in a neighborhood makes the approximation fail completely, creating numerical artifacts and, at some point, makes the simulation diverge. Note that to make sure these artifacts were not coming from the explicit integration scheme, we reran the experiments using this time steps of 0.05 [ms] and an implicit scheme using steps of 5 [ms], both of which resulted in exactly the same solution and artifacts.

As expected, a higher approximation order can improve the accuracy of the solution. The same stretching experiment was repeated with exactly the same configuration, but this time using the MLS shape function. The results shown in figure 3.10 indicate that starting with 10 neighbors, the beam is allowed to stretch further than with the SPH shape function. However, the same numerical artifact arises when using larger neighborhood.

Mean computational times for these experiments are provided in table 3.1 for both

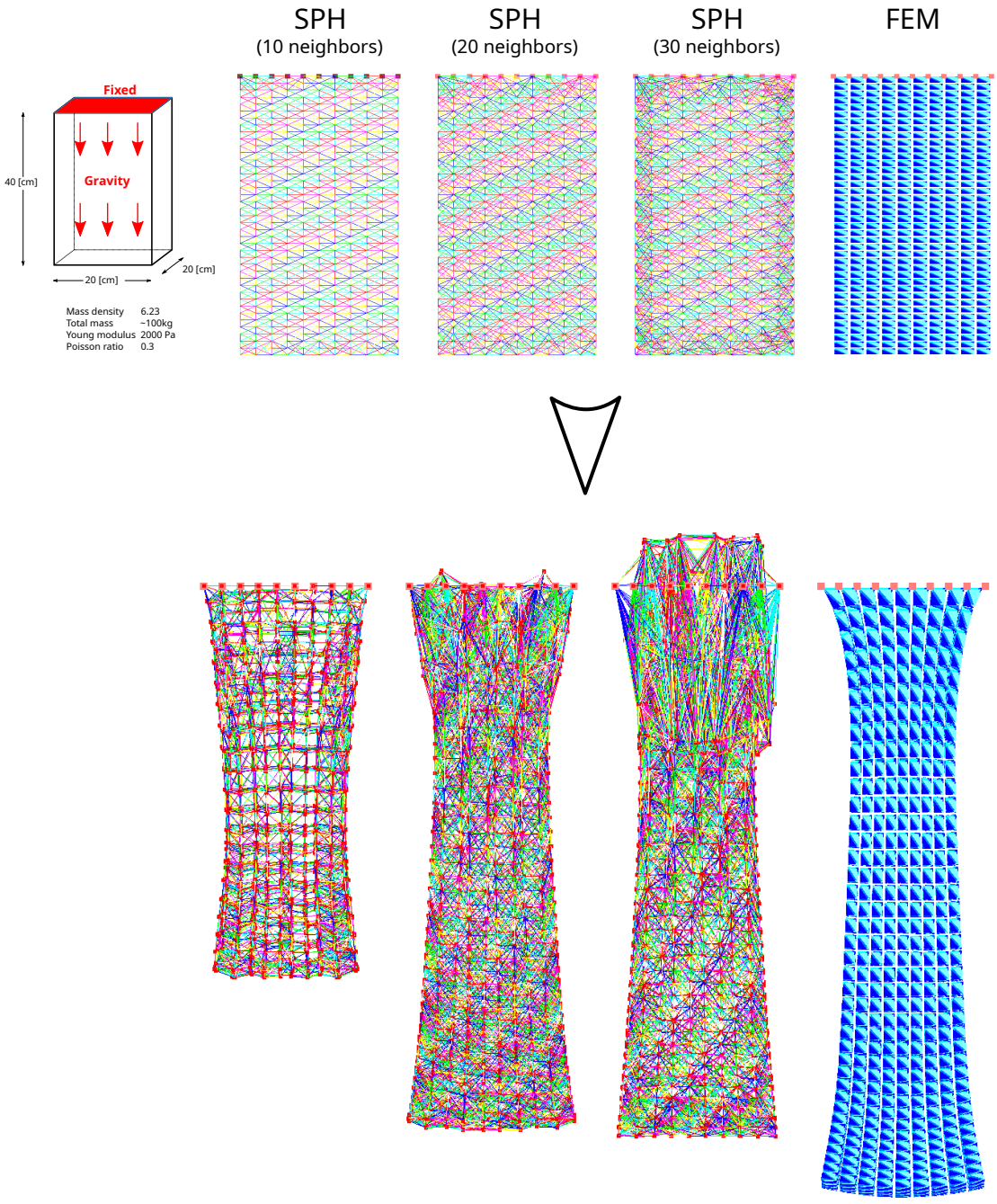


Figure 3.9: Stretching of a parallelepiped beam of $10 \times 40 \times 10$ particles: from left to right, SPH approximation with respectively 10, 20 and 30 neighbors for each particle. At the complete right, a reference FEM solution with the same number of nodes and same material.

the PBA and the MLAMBI methods. Without any surprise, the computation time increases with the size of influences of the particles. For the MLAMBI method, the number of integration points per background elements also plays a major role. Under-integration using only one integration point per element does improve the computation time of the internal residual assembly, but decreases the stiffness matrix conditioning, hence requiring more conjugate gradient iterations to solve the linear system of equation.

Bending

For the bending simulation scenario, we were interested in the behavior of the PBA method when the simulated object undergoes a large rotation. Using the same configuration as per the stretching experiments, we simply changed the direction of gravity in order to deform the beam sidewise.

Similarly to the stretching experiment, bending the beam using large neighborhoods creates numerical artifacts close to the fixed part of the beam (top face). However, here we are able to observe another drawback due to the lack of consistency associated with the shape functions. When using the non-linear Green-Lagrangian strain tensor $\mathbf{E}$, which normally should be insensible to rotation, ghost forces similar to those obtained with a small strain assumption arise. We suspect that the weakness of the approximation is at fault here. Figure 3.11 shows the results of the bending experiment with the MLS shape function. Similar results were obtained with the SPH method.

To validate our suspicions on the weakness of the shape functions, we borrowed the MLS shape functions of the MLAMBI method. This allowed us to enrich the solution by adding monomial basis function of higher degree to the approximation's space. However, contrarily to MLAMBI method, we kept the nodal integration of the PBA method. Results of the bending experiment with the non-linear Green-Lagrangian strain tensor are presented in figure 3.12.

A rapid observation of the deformed shapes is enough to confirm the lack of accuracy of the PBA method compared to a FE solution and that, for both stretching and bending experiments. As we will see in the next section, the rough approximation of a particle's volume which is the ground of the numerical integration method is clearly not enough. Nevertheless, the PBA method remains interesting for applications that merely require visually plausible deformations or when complex meshing scenarios which are typically addressed by FE methods, are not required.

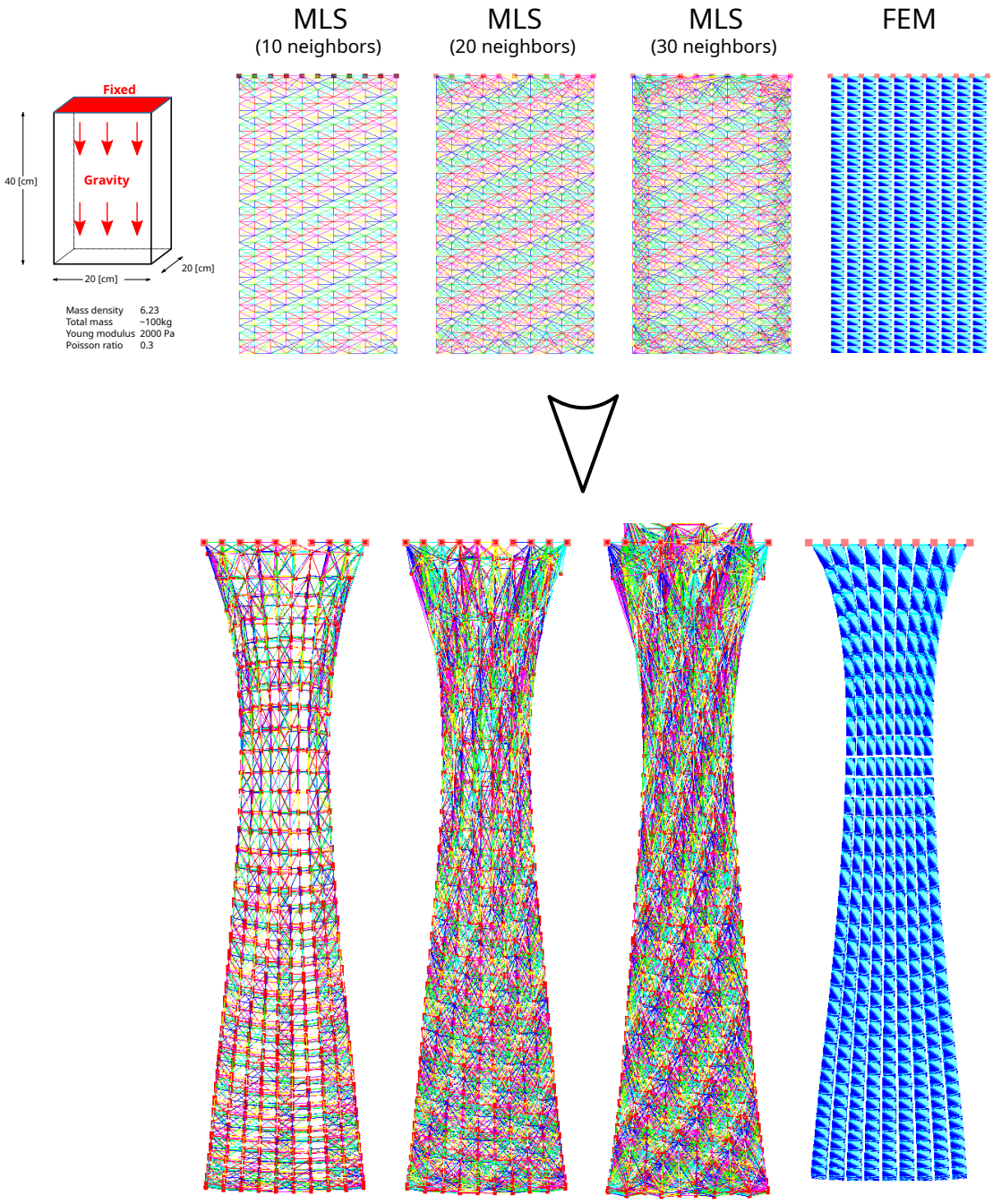


Figure 3.10: Stretching of a parallelepiped beam of $10 \times 40 \times 10$ particles: from left to right, MLS approximation with respectively 10, 20 and 30 neighbors for each particle. At the complete right, a reference FEM solution with the same number of nodes and same material.

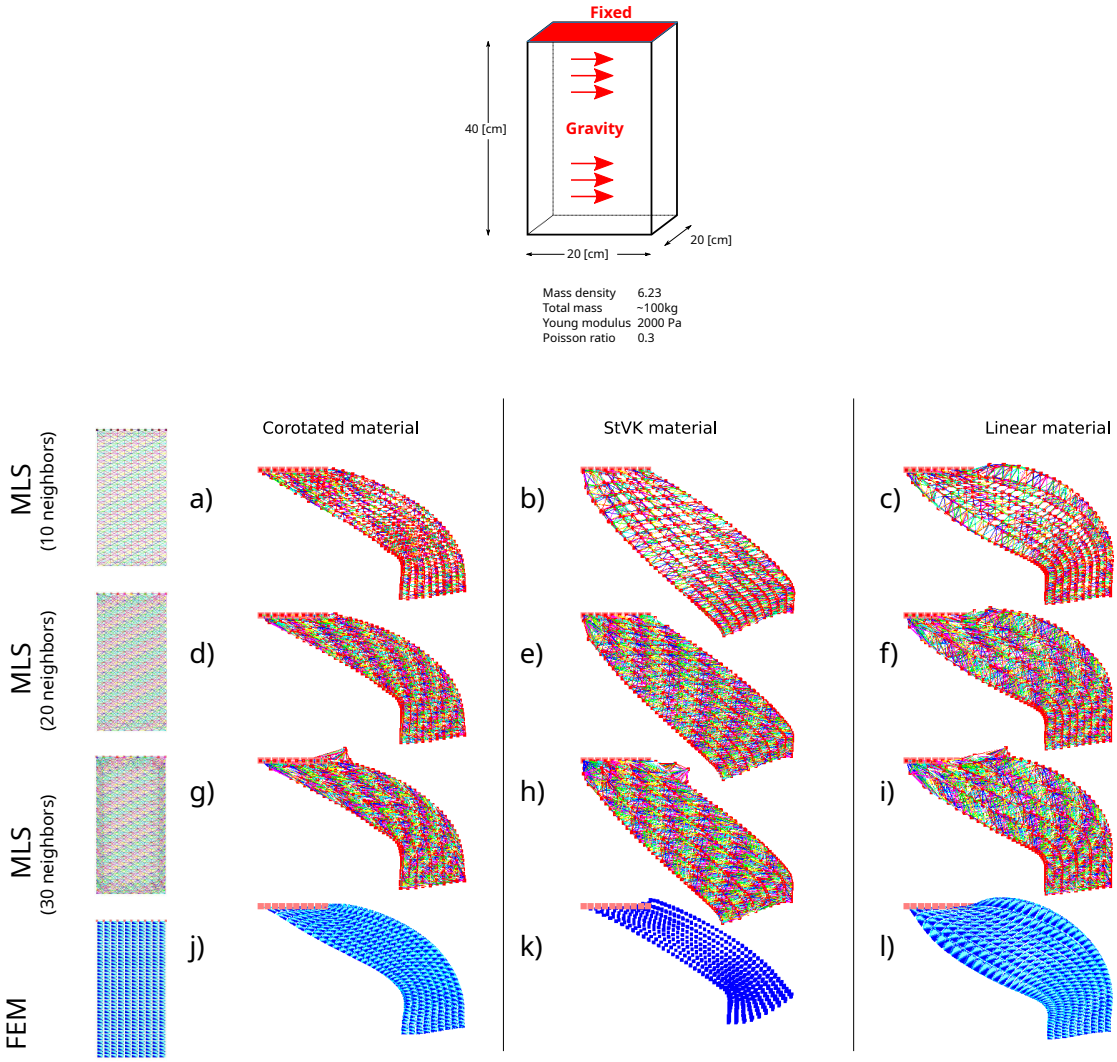


Figure 3.11: Bending of a parallelepiped beam of $10 \times 40 \times 10$ particles: comparison between the non-linear Saint-Venant-Kirchhoff material, the corotated Cauchy material and the linear Cauchy material. a-c) MLS 10 neighbors, d-e) MLS 20 neighbors, g-i) MLS 30 neighbors, j-l) FEM

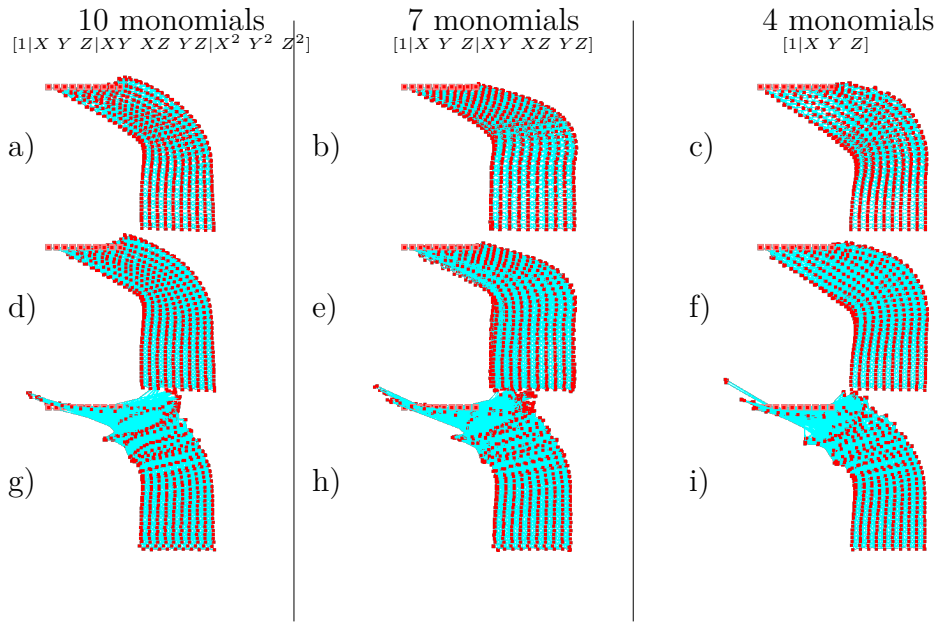


Figure 3.12: Bending of a parallelepiped beam of $10 \times 40 \times 10$ particles with 3159 integration points using MLS approximations of different orders: a-c) 10 neighbors, d-f) 15 neighbors, g-i) 20 neighbors

Performance of the MLAMBI method

We will now look at how the MLAMBI model proposed in this chapter performs in comparison to the PBA and FE methods. We also wish to validate that our implementation is adequate from a numerical point of view by analyzing its convergence characteristics from the reference FE solution. The main results presented here are also available in Brunet, Magnoux, et al. (2019).

As we did for the PBA analysis, our performance evaluation involved the bending and stretching experiments. In this case, however, we evaluated metrics over the displacement field instead of a simple visual validation. To remove any numerical error from the time integration, we analyzed static solutions using a maximum of 100 Newton–Raphson iterations. Here, we modeled a rectangular beam of $20 \times 20 \times 200 mm^3$ placed horizontally and having its top face fixed. For the constitutive law, we relied on the linear Cauchy elasticity using the corotational approach described in section 3.6 with a Young modulus of $50 N/mm^2$ and a Poisson ratio of 0.45. For the bending experiment, we applied a downward traction field of $12 N/mm^2$ on the end-face of the beam. For the stretching experiment, we applied a traction field of $1500 N/mm^2$ in the direction of the face normal. The reference solution

was obtained for both scenarios using the corotated FE method of Nesme, Payan, and Faure (2005) with approximately $70k$ hexahedrons.

We started our analysis by measuring the convergence rate of the method. As we can see in figure 3.13, as the number of nodes increases, our MLAMBI method converges towards a unique solution. The accuracy of the solution is irrelevant for this first metric as we only wish to validate that the method converges at a constant rate, which is the case for both the stretching and the bending experiments.

Having confirmed that the method converges, we can now look at how close we get to FE solutions using the same material formulation. This is done in figure 3.14, where the distance of the center point of the beam at the end of the simulation from the center point in the reference solution is provided. Note that for the bending experiment, we also provided the solution produced by the nodal SPH-based integration of the PBA method equipped with the MLS approximation of the MLAMBI method. We can see from these results how the integration of the elasticity over a nodal volume approximation can impact the accuracy of the simulation. Computation times for each simulation are provided in figure 3.14c.

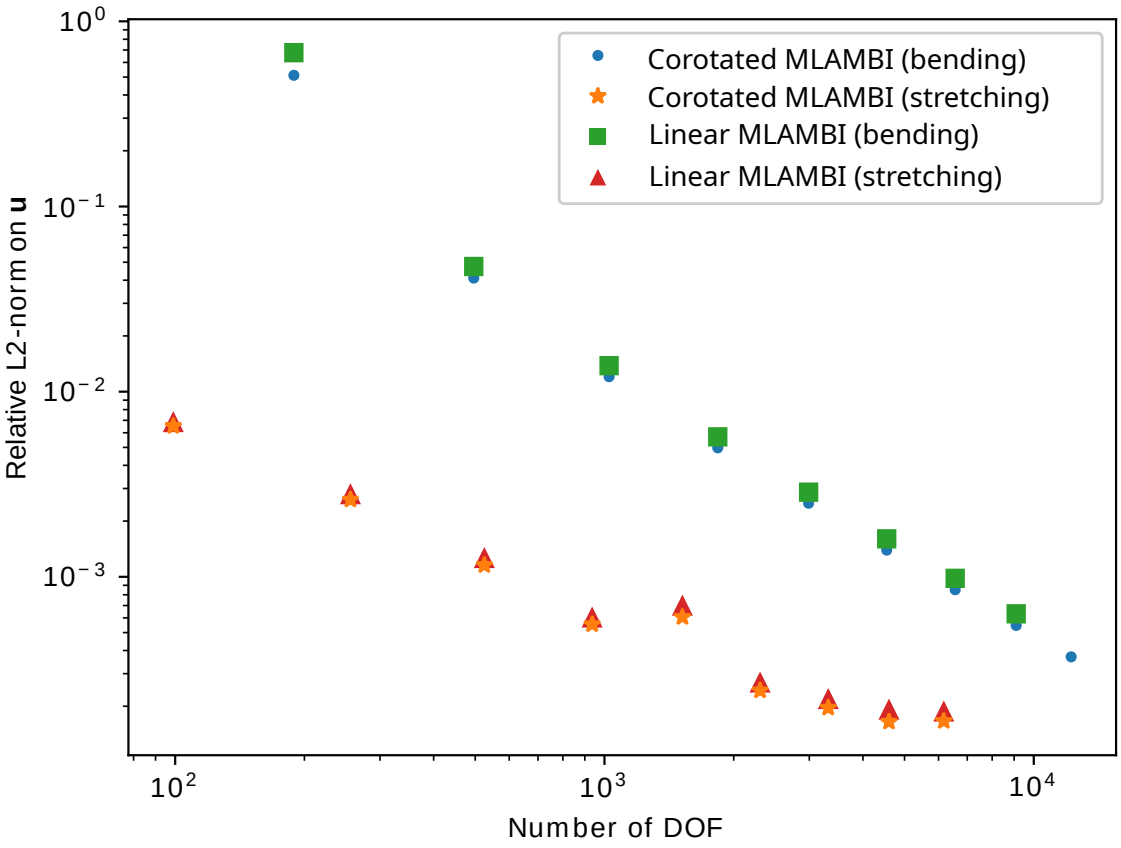
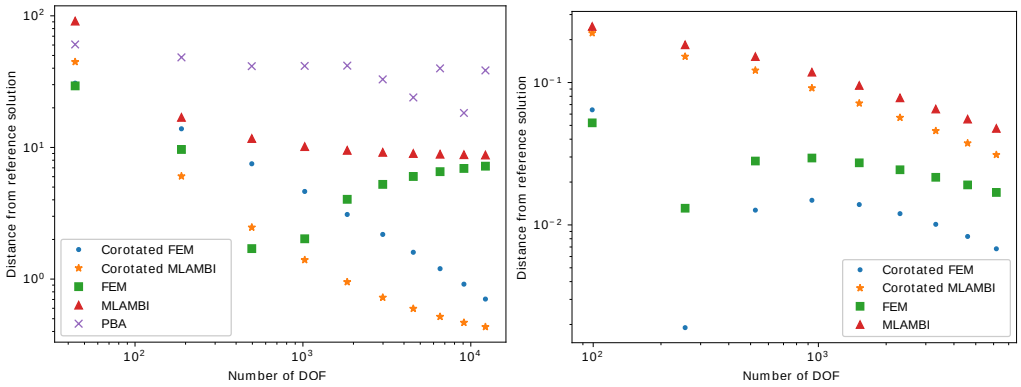


Figure 3.13: Relative L2-norm of the displacement error between the solution with n Degrees Of Freedom (DOF) and the solution with (n-1) DOF for stretching and bending scenarios, using a linear elasticity material with and without our corotated approach.

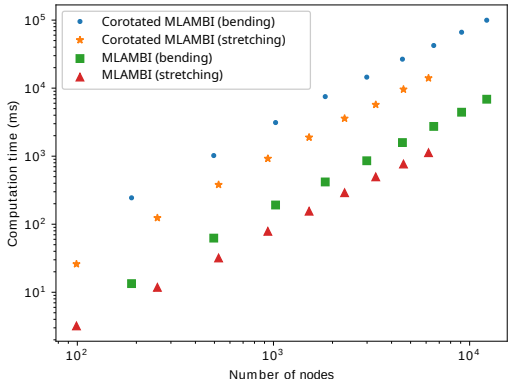
Table 3.1: Computational time - Linear elasticity

Method		Nb of neighbors	Internal forces assembly [ms]	Step time [ms]		Conjugate gradient	
				Explicit	Implicit	Nb iterations	Solve time
PBA	SPH	10	3.5	5.64	47.54	12	40.70
		20	6.52	8.47	101.40	13	87.97
		30	9.99	10.71	167.29	14	146.69
	MLS	10	3.26	5.81	50.16	13	43.44
		20	6.80	8.30	109.4	14	95.53
		30	10.48	11.98	183.20	15	161.30
MLAMBI	1 integration pt. per hexahedron	10	2.19	2.94	37.63	29	33.92
		15	2.96	3.60	63.28	30	58.06
		20	3.53	4.22	56.18	20	49.88
	8 integration pt. per hexahedron	10	6.35	6.61	41.21	9	32.86
		15	7.91	7.75	56.05	9	45.32
		20	10.06	10.50	84.05	19	69.93
FEM			1.55	2.34	45.44	34	42.39

Mean step time for the stretching experiments using $10 \times 40 \times 10$ particles (cf. figures 3.10).



(a) Bending experiment (b) Stretching experiment



(c) Simulation times

Figure 3.14: Accuracy of solutions compared against the corotated FE reference solution.

Table 3.2: Computational time - Corotated elasticity

Method		**Nb of neighbors**	**Internal forces assembly [ms]**	**Step time [ms]**		**Conjugate gradient**	
				Explicit	**Implicit**	**Nb iterations**	**Solve time**
PBA	SPH	10	7.17	9.11	51.88	12	40.93
		20	12.71	12.83	108.16	13	88.36
		30	18.43	18.57	176.85	14	147.64
	MLS	10	6.52	8.08	53.67	13	45.55
		20	11.96	12.52	108.15	14	94.51
		30	17.17	17.93	175.23	15	155.98
MLAMBI	1 integration pt per hexahadron	10	3.55	4.12	42.55	29	37.70
		15	4.02	4.96	67.70	30	59.91
		20	4.17	5.49	70.12	25	64.27
	8 integration pt per hexahadron	10	7.49	8.29	51.25	9	45.36
		15	8.25	8.91	62.23	10	58.44
		20	9.99	10.45	91.11	20	71.89
FEM			3.06	3.75	58.01	34	53.06

Mean step time for the bending experiments using $10 \times 40 \times 10$ particles (cf. figures 3.11).

3.12 Discussions

In this chapter, the first category of the simulation models considered in this thesis, the meshless methods, has been analyzed. These represent an alternative to traditional Finite Element (FE) methods that rely on a mesh made of well-formed and well-placed elements and that must comply with the boundaries of the simulated domain. The idea of using a method that can discretize in a Galerkin way a domain by simply filling it with a cloud of points sounded quite appealing.

Unfortunately, we quickly encountered many issues and challenges. The first one is associated with the complexity of the underlying theory. While many references can be found for FE methods based on standard iso-parametric elements, the situation is different with meshless methods. Most of the relevant literature was pertaining to the field of computational mechanics, which is usually not adequate for the type of interactive simulations we are interested in. The other drawback comes from a more technical aspect: most of the models analyzed in this chapter had to be implemented from scratch. The available FE software we found to support us during this first part of the thesis were extremely strict as to the extensions allowed, especially as soon as we were stepping outside of the usual iso-parametric element-based discretization area. Thus, our comparison between computation times from our meshless implementations and the finite element method should be considered with some reservations as the latter was computed using a highly optimized software.

Nevertheless, for an equal number of nodes and order of approximation, our experiments have shown that meshless methods will generally require more computational time compared to the FE methods since their approximations of a given field function are usually dependent on more degrees of freedom (neighboring particles) than with geometrical iso-parametric elements. Moreover, having a variable number of nodes per integration point will impact the computational performance as it prevents many useful optimization strategies for the assembly of the stiffness matrix. For example, the vectorization of CPU operations, or a strict alignment of computer memory blocks to avoid cache misses, become almost impossible to implement with this approach.

We chose to start our analysis of meshless methods with the PBA model because of its popularity in the research community and more importantly, because it is physically driven and aimed at interactive simulations. However, we saw that even if the basic rules of continuum mechanics are applied, the resulting model will not necessarily produce physically accurate solutions. In this case, while producing

visually pleasing simulations, the weakness of both the approximation space and the volumetric integration schemes turned out to be too inaccurate and definitely not adapted for surgical application involving large deformations.

To overcome these inaccuracies, the approach documented in Horton et al. (2010) looked interesting and has led us to the implementation of our MLAMBI method. This one relies on an MLS method where the approximation space can be enriched by adding monomial basis functions. As we described in this chapter, instead of doing a nodal integration based on an SPH-based volume estimation, we used a background grid of regular hexahedral elements where the standard gaussian quadrature can be used. We have proposed a way to apply a corotational approach to efficiently simulate large displacement with a linear elasticity, hence greatly improving the computational speed. Since our simulation context depends mostly upon low frequencies modes from large deformation induced by the surgeon's manipulations on the liver, we have shown how an implicit time integration scheme, which is optimal for these kinds of problems [Wriggers (2008)], can be formulated.

While we were quite satisfied with our results and believed our implementations represented real contributions, we decided to put aside meshless methods for the rest of this thesis. The meshing difficulties found in traditional FE methods are undoubtedly avoided here, but the complexity inherent to meshless methods has pushed us to reconsider our initial objectives. We realized that creating FE meshes is problematic only when boundaries have to be rigorously represented by the mesh. A method in which this constraint could be relaxed and where the advantages of the FE approach would be retained would constitute a significant contribution for our type of application. For example, such method could allow us to create a coarse FE mesh where geometrically challenging boundaries are simply embedded inside the mesh. The problem of meshing these boundaries would therefore be transformed to the problem of correcting the computation over elements cut by the boundary. Handling such discontinuity brings us into a totally new branch of numerical methods referred to as *fictitious domain* methods. These are also called the *immersed boundary* methods and will be the subject of the next chapter.

Chapter

4

IMMERSED BOUNDARY METHODS

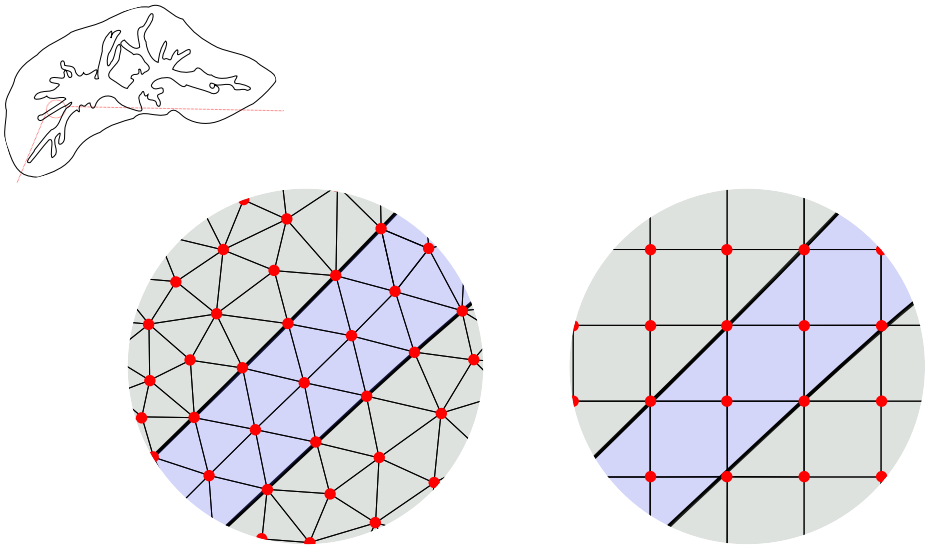
The *Immersed boundary (IB) methods*, sometimes referred to as *embedded domain methods*, consist of a subset of the Finite Element (FE) methods in which the concept of *fictitious domains* is applied. Here, the simulated object is placed directly into a computational background mesh (the fictitious domain) generally made of regular and simple elements (figure 4.1b). As we discussed in the previous chapters, with the traditional FE approach, the computational mesh needs to closely match the boundary of the simulated domain. With the IB methods, the numerical operations are executed on a set of elements that may not conform geometrically to the boundaries of the domain of interest.

Since their first introduction in the work of Peskin (1972) which was aimed at solving flow patterns around heart valves, IB methods have evolved to a wide range of fluid-fluid, fluid-solid and solid-solid implementations. Because the computational mesh does not conform to its embedded objects, the main challenges of IB methods reside within the computation of a background element that is cut by the boundaries region. We call *the boundary interface* of any element the representation of the cut surface that separates the region inside the simulated object from the outside domain. The accurate *identification and representation of this interface* inside a background element constitute the first step of the method and is not always trivial to implement. The exact location of a boundary interface within an element can be provided by an explicit representation (usually by means of particles or geometric descriptions). An implicit representation which uses a volumetric scalar-valued field function (such as those found in level-set methods) may also be used. In this case, a threshold will define whether the field value at a given position is considered inside, or outside of the domain. Note that the concept of interfaces with IB methods are not restricted to the inside/outside domain boundaries. They can also be extended to interfaces that separate a given simulated material against another. For instance, an interface could separate an elastic material whereby the stiffness is different whether it is evaluated on one side of the discontinuity or the other. It can also be extended to model different anisotropic directions or even scenarios which involves objects with material laws that are completely different such as a hyperelastic material on one side, and a fluid material on the other side.

We will begin this chapter with a review of the literature associated with IB methods. We will then present two implementations. The first one, the *Finite Cell (FC) method*, is a well-known embedded domain method [Parvizian, Düster, and Rank (2007)]. The second one, the *Weighted Cell (WC) method* is an extension of the FC method which we have designed based on the key requirements of our application. An analysis and preliminary validation of these two methods are provided. Finally, a more thorough set of simulation tests are performed using porcine and human livers. We conclude this chapter with a discussion of our results and the observations we have made throughout this second part of this thesis.

4.1 Literature review

The main challenge associated with the IB approach arises when the immersed object does not fit entirely in some of the underlying computational elements. These elements must therefore be separated in two regions: the region associated



(a) Single boundary conforming mesh.

(b) Boundary immersed into a fictitious (background) grid.

Figure 4.1: Simplified example of a simulated liver 2D cross-section where both the parenchyma and the internal vessels must be taken into account, each having a distinct material description

with the interior of the simulated object, and the region which is outside. The underlying computational element is therefore cut in two parts and must be treated as two distinct subspaces. The geometry of these two subspaces and their material must obviously be taken into account. This can be done by introducing a continuity constraint on the interface boundary between the two regions: there must be no jump on the solution field (in our case, the displacement field) with respect to both regions. Simply put, near (in a limit sense) the interface between these regions, the displacement must be equal on both sides.

Many approaches have been used in the past to address this constraint. Most of them are based on the following three methods: the penalty method [Bishop (2003); Ramière, Angot, and Belliard (2007)], the Lagrange multiplier method [Burman and P. Hansbo (2010); Glowinski and Kuznetsov (2007)] and the Nitsche's method [Nitsche (1971); A. Hansbo and P. Hansbo (2002); Dolbow and Harari (2009); Burman and P. Hansbo (2012); Schillinger and Ruess (2015)]. These methods may also serve as a means to impose essential boundary conditions when the external region is the empty space.

For cut elements, an integration scheme must also be defined to account for the two subspaces separated by their boundary interface. As documented in [Düster et al. (2008)], this is usually done by representing the geometry inside an element through a set of quadrature points which are usually refined until a prescribed accuracy is achieved. When both regions of the element are filled with linear materials, and when the boundary interface can be approximated into a set of surface elements (usually through triangular tessellation), the volume integrals can be converted to surface integrals using the divergence theorem [Mirtich (1996); Bishop (2003); T. P. Fries and Omerović (2016)]. The benefit of this approach is that the underlying weak equations can be integrated exactly.

Over the past 20 years, many combinations of these fictitious domain approaches have been proposed which resulted in some efficient methods for discretizing complex geometries. One of the most popular, *the Finite Cell Method (FCM)* [Parvizian, Düster, and Rank (2007)], uses a background grid of high order regular hexahedral elements and are often coupled with Nitsche's method [Nitsche (1971)] to impose Dirichlet boundaries. An adaptive gaussian quadrature scheme is then used to integrate the weak formulation of the partial differential system. In Ruess et al. (2012), the FCM was used to predict the bone mechanical response of the human femur where a patient-specific model is built directly from CT images. By using hierarchical splines, Verhoosel et al. (2015) reconstructed the smooth geometry of trabecular bone from voxel-based images and simulated the elastic body through the FCM. In Elhaddad et al. (2018), an hp-refinement scheme is added to the FCM in order to simulate a vertebra-implant model. An exhaustive review of finite cell methods can be found in Schillinger and Ruess (2015). We will implement a version of the FCM later in this chapter.

Burman and co-workers proposed another combination of the immersed boundary concepts called *cutFEM* [Burman and P. Hansbo (2012); Burman, Claus, et al. (2015)]. Similar to the FCM approach, they use a background grid made of iso-parametric elements to compute the approximate solution of PDEs. They enrich the weak formulation on cut elements with a ghost penalty parameter that improves the robustness of the method by making the conditioning of the matrix independent of the way the boundary cuts the computational mesh. A good overview of different strategies to deal with ill-conditioning matrices due to some background cells that intersect the physical domain on a very small fraction of their volume can be found in the work of Prenter et al. (2017).

For interactive simulations, which must quickly provide updated solutions in response to new boundary interactions in time (usually from collisions or external data constraints), linear background elements are usually preferred. In Nesme,

Payan, and Faure (2006) , authors embed non-homogeneous corotated elastic materials inside regular hexahedral elements. The stiffness matrix of an element is integrated on a finer grid of sub-cells, each of them matching exactly one voxel that contains the material properties at this location. The contribution of these smaller sub-cells are then propagated back to their parent coarse cell. This process, which falls under the branch of *homogenization methods*, was later extended by Nesme, Kry, et al. (2009) by proposing shape functions on the coarse cells that consider the heterogeneous parts of their sub-cells. By setting kinematic constraints on the sub-cells nodes (seen as virtual nodes, as they are not represented in the system's set of unknowns), the elastic properties of their coarse parent cell are solved in a pre-simulation stage such that the displacement on both the coarse and finer nodes matches.

In both Nesme, Payan, and Faure (2006) and Nesme, Kry, et al. (2009), the integration of the weak equations were done on corotated sub-cells with respect to the coarse hexahedron local frame. The authors of Jeřábková et al. (2010) proposed a framework aimed at medical simulations and based on a similar multigrid coarsening approach that enables interactive cutting of the underlying mesh. To do so, they built a connectivity graph between each coarse cells and their subcells. They proposed a simpler homogenization method to quickly update coarse stiffness and mass matrices after a topology change. These homogenization methods were then extended by the work of Torres, Espadero, et al. (2014) with a more flexible strategy that avoids the need for multiresolution meshes. Later on in Torres, Rodríguez, et al. (2016), they demonstrated that all of these corotated coarsening strategies fail to correctly represent boundary constraints when imposed on the finer subcells. They proposed a method to better handle some of these constraints, at the cost of being too complex to handle non-linear material or topological changes.

More recently, C. Paulus et al. (2017) demonstrated that, instead of subdividing each hexahedron into sub-cells, when linear materials are used, the divergence theorem can be applied to accurately integrate the element stiffness by tesselating the cut hexahedrons into a set of planar facets.

4.2 The choice of background element type

While the use of regular cuboid elements remains a popular choice in the research community, the concept of fictitious domain may also be extended to other geometrical element types. In three dimensions, the usual alternative is the Tetrahedral Finite Cell Method (TFCM) [Stavrev et al. (2016); Xu et al. (2016); Varduhn et

al. (2016)]. The problem of choosing between tetrahedral or hexahedral elements is not a new one and is definitely not exclusive to the FC method. Comparison between the two types of iso-parametric elements have been done numerous times. The rationale behind using tetrahedral elements is generally because it is much simpler to automatically generate a mesh that conforms to the domain's boundaries [Shepherd and Johnson (2008); Burkhart, Andrews, and Dunning (2013)]. Some advanced methods might be able to recombine groups of tetrahedra into hexahedral elements, yielding hybrid meshes. However, as soon as sharp features are found within the simulated domain, it becomes extremely difficult to mesh the domain with only hexahedral elements [Sokolov et al. (2016)]. The difficulty of meshing with hexahedral elements is quite unfortunate, as they often require 4 to 10 times fewer elements than a tetrahedral mesh to reach the same level of accuracy [Shepherd and Johnson (2008)]. We observed this phenomenon while simulating the bending of a three-dimensional rectangular beam (figure 4.2). Moreover, tetrahedral elements are known to be less accurate due to their high stiffness and predisposition to locking [Benzley et al. (1995); Tadepalli, Erdemir, and Cavanagh (2011); Raut (2012)]. Following the conclusions of Burkhart, Andrews, and Dunning (2013), we believe that hexahedral meshes should be used whenever biological structures and soft tissue material have to be modeled.

Fortunately, in the case of IB methods, the computational mesh does not need to be aligned with the simulated domain. It therefore seems natural to incline our choice of background elements towards hexahedrons as we are not affected by the constraints inherent to standard FE meshes. Moreover, as we will see in the following sections, using a three-dimensional Cartesian grid of rectangular hexahedral cells turns out to be quite convenient. They are easily refined around the regions of interest through octree algorithms. The mapping transformation function from the material coordinates into the canonical base is linear, hence its Jacobian is constant and its inverse can be computed directly. Moreover, these two characteristics make it a very good candidate for efficient geometry-based hierarchical solvers. Finally, since the model is built from pre-operative images of the patient made of rectangular voxels, it seems natural to use a grid made of regular hexahedral elements that could easily match the voxels.

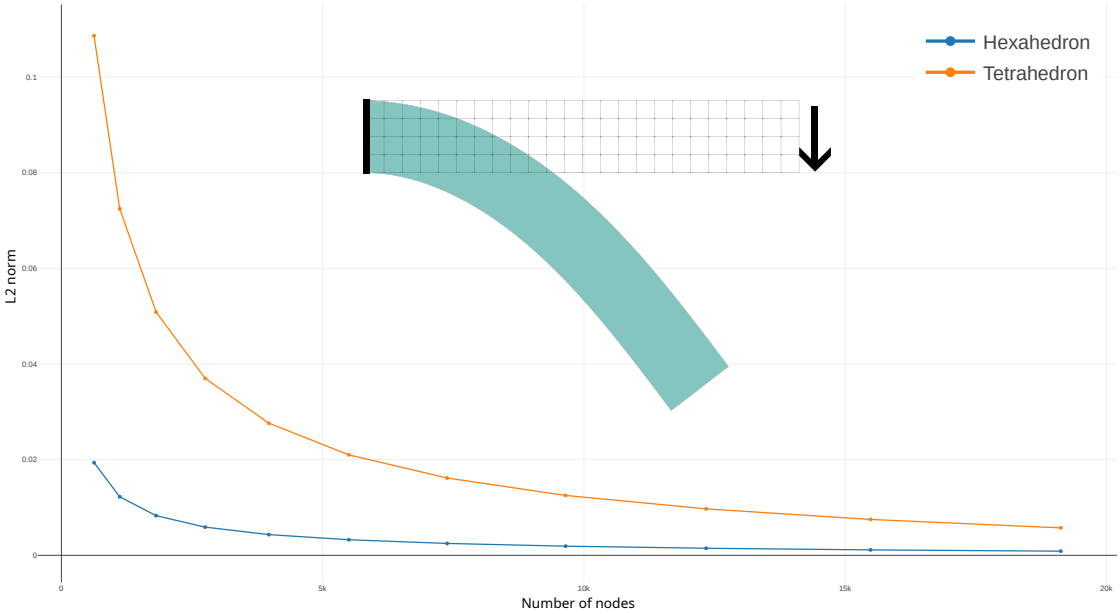


Figure 4.2: Comparison of a 3D beam of rectangular cross section modeled with an hyperelastic Saint-Venant-Kirchhoff material. The beam is fixed at one side and pulled down on the other side. The L2-norm of the displacement against a 10 nodes tetrahedral mesh solution is shown at different mesh sizes for both tetrahedral meshes and hexahedral meshes.

4.3 Immersed-boundary discretization and integration

Our next objective is to develop a model that can discretize the simulated liver with a computational background grid composed of rectangular hexahedral elements as discussed in the previous section. Often call cells, these elements, completely engulf the boundaries of the simulated object. The nodes of the cells represent the Degrees Of Freedom (DOF) of the system, as would any elements generated in traditional FE methods. To make this possible, we simply rely on an advanced technique to accurately integrate the hyperelasticity equations described in chapter 2 on the partially filled cells. Hence, the first step of the process consists simply in placing a regular grid on top of the pre-operative segmented surface mesh of the liver. Elements lying completely outside of the surface are removed from the simulation.

For the elements lying completely inside the surface, the simulation is done as per the standard hexahedral FE method. For cells that are cut by the boundary, another approach must be used. In the following sections, we analyze two of them.

The first one is based on the FCM method mentioned earlier and will be referred to as the FC method in the rest of this chapter. The second one is based on a new approach that led us to our proposed Weighted Cell (WC) method.

4.4 The Finite Cell approach

The mapping from the *local* (or *canonical*) coordinates $\boldsymbol{\xi} = [\xi, \eta, \zeta]^\intercal$ of a point inside the reference hexahedral element (section 2.8) to its *material* coordinates in Ω is given by $\boldsymbol{X} = \boldsymbol{T}(\boldsymbol{\xi})$ with

$$\boldsymbol{T}(\boldsymbol{\xi}) = \sum_{i=1}^{8} N_i(\boldsymbol{\xi}) \boldsymbol{X}_i \tag{4.1}$$

and where $N_i(\boldsymbol{\xi}) = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta)$ is the trilinear shape function of the element's i$^{\text{th}}$ node which is located at the initial position (in material coordinates) $\boldsymbol{X}_i$.

The inverse mapping $\boldsymbol{\xi} = \boldsymbol{T}^{-1}(\boldsymbol{X})$ is often useful to compute the intersection between the immersed boundary mesh and the grid. Since the shape functions are not linear, the local coordinates $\boldsymbol{\xi}$ can be approximated with an iterative Newton–Raphson (NR) algorithm. Starting at an initial guess $\boldsymbol{\xi}_0$ (usually the center point of the element or its first node), we then have:

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \mathbf{J}^{-1}(\boldsymbol{X} - \boldsymbol{T}(\boldsymbol{\xi}_k)) \tag{4.2}$$

where $\mathbf{J}$ is the jacobian of the transformation $\boldsymbol{T}(\boldsymbol{\xi}_k)$ at the k$^{\text{th}}$ iteration. The iterations usually stop when $\frac{|\boldsymbol{X}-\boldsymbol{T}(\boldsymbol{\xi}_k)|}{|\boldsymbol{X}-\boldsymbol{T}(\boldsymbol{\xi}_0)|} < \epsilon$ for a given threshold ϵ.

However, from our choice of a regular background grid, the transformation can be simplified to

$$\boldsymbol{T}(\boldsymbol{\xi}) = \begin{bmatrix} X_m + \frac{H_x}{2}\xi \\ Y_m + \frac{H_y}{2}\eta \\ Z_m + \frac{H_z}{2}\zeta \end{bmatrix} \tag{4.3}$$

where $\boldsymbol{X}_m = [X_m, Y_m, Z_m]^\intercal$ is the center (middle) point of the cell, and $\boldsymbol{H} = [H_x, H_y, H_z]^\intercal$ its size. Its jacobian is then given by $\mathbf{J} = \frac{1}{2}\begin{bmatrix} H_x & 0 & 0 \\ 0 & H_y & 0 \\ 0 & 0 & H_z \end{bmatrix}$ and is constant.

Thus, the inverse mapping is resolved from

$$\boldsymbol{T}^{-1}(\boldsymbol{X}) = \begin{bmatrix} \frac{2}{H_x}(X - X_m) \\ \frac{2}{H_y}(Y - Y_m) \\ \frac{2}{H_z}(Z - Z_m) \end{bmatrix} \tag{4.4}$$

and can therefore be computed directly.

The numerical integration of a vector field $\boldsymbol{f}(\boldsymbol{X})$ over an element e is done using a Gauss quadrature scheme:

$$\int_{\Omega_e} \boldsymbol{f}(\boldsymbol{X}) d\Omega_e \approx \sum_{I=1}^{8} \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{\xi}_I)) w_I \det(\mathbf{J}_I) \tag{4.5}$$

with w_I the Gauss weight at an integration point located at local coordinates $\boldsymbol{\xi}_I$, and $\mathbf{J}_I$ the Jacobian of its transformation $\boldsymbol{T}(\boldsymbol{\xi}_I)$. For elements cut at the boundary or at an interface between two materials, the Gauss quadrature does not hold anymore since the integrand is discontinuous. However, the regularity of the hexahedral cells can be exploited again, this time allowing for an easy way to split cells cut by the boundary into sub-cells. This technique is known as the Finite Cell Method (FCM) [Düster et al. (2008)].

With this technique, cut elements are recursively split into 8 sub-cells, each one having 1/8 of their parent cell dimensions, a well-known process called an *octree* decomposition. This is illustrated in Figure 4.3 for a two-dimensional quad element (4 sub-cells in this case). The transformation between local coordinates inside a sub-cell (sc) to its coordinates inside its parent cell (pc) is given by

$$\boldsymbol{\xi}^{pc} = \boldsymbol{T}^{sc}(\boldsymbol{\xi}^{sc}) = \begin{bmatrix} X_m^{sc} + \frac{H_x^{sc}}{2}\xi^{sc} \\ Y_m^{sc} + \frac{H_y^{sc}}{2}\eta^{sc} \\ Z_m^{sc} + \frac{H_z^{sc}}{2}\zeta^{sc} \end{bmatrix} \tag{4.6}$$

The integration of a cell (c) can thereby be formulated with respect to its sub-cells with

$$\int_{\Omega_c} \boldsymbol{f}(\boldsymbol{X}) d\Omega_c \approx \sum_{sc=1}^{8} \sum_{I=1}^{8} \boldsymbol{f}(\boldsymbol{T}^c(\boldsymbol{T}^{sc}(\boldsymbol{\xi}^{sc}))) w_I det\mathbf{J}_I^{sc} det\mathbf{J}_I^c \tag{4.7}$$

The integration of the element (e) is done by performing the Gauss quadrature on its leaf cells (lc), which consists of the sub-cells that have not been subdivided. Let

$\boldsymbol{T}^{lc\to e} = \boldsymbol{T}^{e} \circ \cdots \circ \boldsymbol{T}^{pc} \circ \boldsymbol{T}^{lc}$ be the transformation of local coordinates relative to the leaf cell lc to its material coordinates in Ω. Let also $det\mathbf{J}_I^{lc\to e} = det\mathbf{J}_I^{e} \times \cdots \times det\mathbf{J}_I^{lc}$ be the product of all Jacobian determinants of the transformations in $\boldsymbol{T}^{lc\to e}$. The integration on an element (e) that has n_{lc} leaf cells then becomes

$$\int_{\Omega_e} \boldsymbol{f}(\boldsymbol{X}) d\Omega_e \approx \sum_{lc=1}^{n_{lc}} \sum_{I=1}^{8} \boldsymbol{f}(\boldsymbol{T}^{lc\to e}(\boldsymbol{\xi}^{lc})) w_I det\mathbf{J}_I^{lc\to e} \tag{4.8}$$

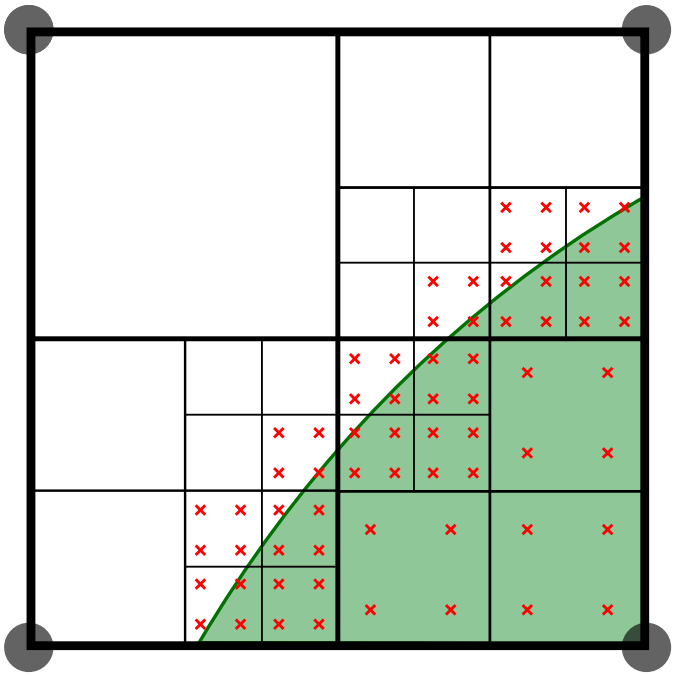


Figure 4.3: A quad element is recursively subdivided into sub-cells when it intersects the boundary. The numerical integration on the element is done by a Gauss quadrature on the integration points (red crosses) of its leaf cells.

While the FC approach described here provides an accurate integration over cut cells (given enough subdivisions), it remains unusable for real-time simulations of non-linear materials. Indeed, when the internal virtual work in equation (2.11) is non-linear, the system must be solved using a Newton–Raphson (NR) method, leading to the following set of iterative equations

$$\mathbf{K}(\boldsymbol{U}_k) \cdot \boldsymbol{U}_{k+1} = \boldsymbol{R}(\boldsymbol{U}_k) + \boldsymbol{B} + \boldsymbol{T} \tag{4.9}$$

where $\boldsymbol{U}_k$ is the displacement vector at iteration k, $\boldsymbol{B}$ and $\boldsymbol{T}$ are respectively the body forces and surface traction vectors. Recall here the operator $\uplus$ introduced

in section 2.6 which denotes that an assembly process takes place. The tangent stiffness matrix $\mathbf{K}(\boldsymbol{U}_k)$ and residual vector are finally defined as

$$\mathbf{K}(\boldsymbol{U}_k) = \biguplus_e \sum_{i=1}^{8}\sum_{j=1}^{8} \underbrace{\int_{\Omega_e} \left[(\nabla_{\boldsymbol{X}}^{\intercal} N_i)\mathbf{S}(\nabla_{\boldsymbol{X}} N_j)\boldsymbol{I} + \mathbf{B}_i^{\intercal}\frac{\partial \mathbf{S}}{\partial \mathbf{E}}\mathbf{B}_j \right] d\Omega_e}_{\mathbf{K}_{ij}} \tag{4.10}$$

$$\boldsymbol{R}(\boldsymbol{U}_k) = \biguplus_e \sum_{i=1}^{8} \underbrace{\int_{\Omega_e} \mathbf{F}\mathbf{S} \cdot \boldsymbol{\nabla}_{\boldsymbol{X}} N_i \, d\Omega_e}_{\mathbf{R}_i} \tag{4.11}$$

$\mathbf{S}$ and $\mathbf{F}$ are respectively the Second Piola-Kirchhoff (SPK) stress tensor and the deformation tensor evaluated with $\boldsymbol{U}_k$. Here, $\boldsymbol{\nabla}_{\boldsymbol{X}} N_i$ is the derivative vector of the i^{th} shape function with respect to the material coordinates.

With the FC approach, the integration on elements is done by evaluating the integrand of equation (4.10) at each integration points on the leaf cells. The computational effort hence grows exponentially ($\approx 2^n$) with the level of subdivision of the elements. In fact, it isn't unusual that the assembly of the matrix $\mathbf{K}(\boldsymbol{U}_k)$ takes even more time that the resolution of the system itself! This would not be a problem if $\mathbf{K}(\boldsymbol{U}_k)$ was constant, as it could be evaluated only once at the beginning of the simulation. However, because we are using a non-linear strain tensor $\mathbf{E}$, this matrix must be re-evaluated at each NR step.

We therefore have to find a middle ground that provides us a with good integration over cut elements while not increasing the time required to build the system of equations. This is covered by the WC approach that we will now introduce.

4.5 The Weighted Cell method

After only a few simulations with the the FC method, it became quite evident that the number of integration points per elements must remain small if we wish to achieve an acceptable computational performance. From a pure technical point of view, it would be even preferable to keep the same quantity of integration points in each cut element. This would allow us to align the integration points in the computer's memory, hence enabling various CPU optimizations. We recall that this is not possible with the Finite Cell method as the number of integration points in each cut elements depends on how the element is cut. In fact, a cut element could have more integration points than its neighborhood.

During our experiments, we found a quite simple alternative which was surprisingly very close to the Finite Cell method. We named it the Weighted Cell (WC) method. The general idea behind this method is to keep the initial 8 integration points of cut hexahedral elements and try to find integration weights that better represent the portion of an element that lies inside the simulated object. Doing so allows us to use existing Finite Element software tools that are very well optimized for 8-nodes/8-integration points elements.

The first level of subdivision yields 8 sub-cells, each one containing one integration point. We can thereby use the subsequent subdivisions of these 8 sub-cells to compute their volume portion lying inside the boundaries. Let tc be a top-level (first level of subdivision) sub-cell. We approximate its volume within the element using

$$v_{tc} \approx \sum_{lc=1}^{n_{lc}(tc)} \sum_{I=1}^{8} 1 \cdot w_I det\mathbf{J}_I^{lc \to tc} \tag{4.12}$$

where $n_{lc}(tc)$ is the number of leaf cells inside the top-level cell tc. Figure 4.4 illustrates this where the red crosses are the integration points of the element. The small circles are the integration points of the leaf cells and are only used to compute the volume v_{tc} of a top-level integration point (red crosses).

The integration of a vector field $\boldsymbol{f}(\boldsymbol{X})$ over an element e becomes

$$\int_{\Omega_e} \boldsymbol{f}(\boldsymbol{X}) d\Omega_e \approx \sum_{I=1}^{8} \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{\xi}_I)) w_I v_{tc}^I det\mathbf{J}_I \tag{4.13}$$

where $v_t^I c$ is the volume of the top-level sub-cell containing the integration point I.

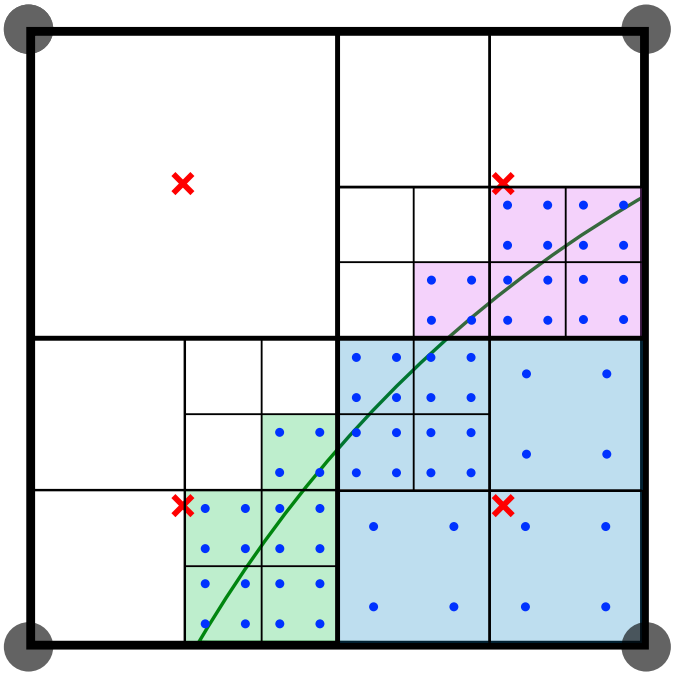


Figure 4.4: A quad element is recursively subdivided into sub-cells when it intersects the boundary. The numerical integration is done by a quadrature on the integration points (red crosses) of the element. The weight of an integration point is adjusted to better represent its portion lying inside the boundaries using the subsequent subdivisions of the top-level sub-cell containing it.

4.6 Neumann boundary condition

Imposing boundary conditions on parts of the surface embedded in the fictitious grid is a key process which has received a lot of attention from the scientific community over the last decade. One frequent approach consists of applying the Neumann boundary conditions by integrating the surface traction term of equation (2.11) on the surface tessellation. In our case, however, the nodes of the surface mesh do not necessarily match those of the underlying fictitious computational grid. We are therefore proposing the following workaround which involves two basic steps. First, we accumulate a surface traction vector $\boldsymbol{T}^s$ (of size $3n_s \times 1$ for a surface mesh having n_s nodes) from a traction field $\boldsymbol{t}(\boldsymbol{X})$ with

$$\boldsymbol{T}^s = \biguplus_{i \in e_s} \sum_{i=1}^{n_s^e} \underbrace{\int_{\Omega_{e_s}} \boldsymbol{t} \cdot N_i^{e_s} \, d\Omega_{e_s}}_{\boldsymbol{t}_i^s} \tag{4.14}$$

where e_s is a surface element containing n_s^e nodes and $N_i^{e_s}$ is the i$^{\text{th}}$ shape function of e_s with respect to material coordinates.

Next, using the inverse mapping $\mathbf{T}^{-1}$ defined in the equation (4.4), we can propagate the traction vector $\boldsymbol{t}_i^s$ of a surface node i to the nodes of its enclosing cell. Let $\mathbb{S}_e$ be the set of surface nodes that lie inside the fictitious grid element e. We construct the traction vector $\boldsymbol{T}$ of equation (4.10) using

$$\boldsymbol{T} = \biguplus_{e} \sum_{i=1}^{8} \sum_{i_s \in \mathbb{S}_e} \boldsymbol{t}_i^s \cdot N_i \tag{4.15}$$

where N_i is the i$^{\text{th}}$ shape function of element e evaluated at the local coordinates $\mathbf{T}_e^{-1}(\boldsymbol{X}_{i_s})$. In practice, we build a sparse and rectangular matrix $\boldsymbol{H}$ of size $3n \times 3n_s$ called the *mapping matrix*. Let $0 \leq i \leq n$ and $0 \leq j \leq n_s$, and let e_j be the hexahedral element that contains the j$^{\text{th}}$ surface node. We define

$$\boldsymbol{H}_{ij} = \begin{bmatrix} N_i^{e_j} & 0 & 0 \\ 0 & N_i^{e_j} & 0 \\ 0 & 0 & N_i^{e_j} \end{bmatrix} \tag{4.16}$$

as the 3×3 mapping sub-matrix where $N_i^{e_j}$ is the element e_j shape function at node i evaluated at local coordinates $\boldsymbol{T}^{-1}(\boldsymbol{X}_j)$. The traction vector $\boldsymbol{T}$ finally becomes

$$\boldsymbol{T} = \boldsymbol{H}\boldsymbol{T}^s \tag{4.17}$$

which can be pre-assembled before the simulation.

4.7 Dirichlet boundary condition

The methods used to apply the Dirichlet conditions vary between the penalty methods [Bishop (2003); Ramière, Angot, and Belliard (2007)], the Lagrange multiplier methods [Burman and P. Hansbo (2010); Glowinski and Kuznetsov (2007)] and Nitsche's based methods [Nitsche (1971); A. Hansbo and P. Hansbo (2002); Dolbow and Harari (2009); Burman and P. Hansbo (2012); Schillinger and Ruess (2015)]. The Lagrange multiplier and Nitsche's methods are usually the preferred approach when very accurate solutions around these boundaries must be found. However, they often require a considerable amount of computational time, and adding their implementation into a simulation framework can rapidly become complex.

For our type of application, a strict compliance of Dirichlet conditions is rarely necessary since the state of liver boundaries reconstructed during a surgery (usually

from stereo or RGB-D images) are usually incomplete and roughly approximated. Therefore, we selected a penalty method that has proven to be more than enough for our needs. Moreover, we finally ended up with a penalty method that has a useful property that is not available with the Lagrange multiplier and Nitsche's approaches: it is more robust under non-physical displacement impositions. As we will see in chapter 5, the non-rigid registration between a simulated liver and an intra-operative reconstruction can often be resolved by imposing displacements towards a noisy point cloud. Some of these displacements might not make any sense from a physical point of view, and a strict imposition method such as Lagrange multipliers and Nitsche's methods would probably prevent the simulation to converge. With our penalty method, we can dynamically vary the penalty parameter in order to match the global shape of the point cloud, without having the whole surface clipping it perfectly.

The steps to impose a penalty force on a Dirichlet boundary are similar to what we have done with the surface traction in the last section. We first accumulate a surface force vector $\boldsymbol{S}^s$ (of size $3n_s \times 1$ for a surface mesh having n_s nodes). Let $\mathbb{S}_d$ be the set of surface nodes lying on a Dirichlet boundary. We define

$$\boldsymbol{S}^s = \biguplus_{i \in \mathbb{S}_d} k\boldsymbol{I} \left\| \boldsymbol{u}_i^k - \boldsymbol{u}^d \right\| \tag{4.18}$$

where k is the penalty parameter, $\boldsymbol{u}_i^k$ is the displacement of the i^{th} surface node at the current NR iteration, and $\boldsymbol{u}^d$ is the imposed displacement. This penalty can be seen as if we were attaching elastic springs of stiffness k between a surface node and its imposed position (see figure 4.5).

To transfer the surface penalty forces back to our computational fictitious grid, we use the same mapping system defined in section 4.6:

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{S}^s \tag{4.19}$$

However, this time the forces are dependent on the current displacement, which means that in order to use the NR algorithm, the tangent stiffness matrix of those forces is required. For the surface force, the derivation is straight forward

$$\boldsymbol{K}_s^s = \biguplus_{i \in \mathbb{S}_d} k\mathbf{I} \tag{4.20}$$

The stiffness contribution to the global matrix of the fictitious grid is computed using again the mapping matrix, i.e.:

$$\boldsymbol{K}_s = \boldsymbol{H}\boldsymbol{K}_s^s\boldsymbol{H}^\intercal \tag{4.21}$$

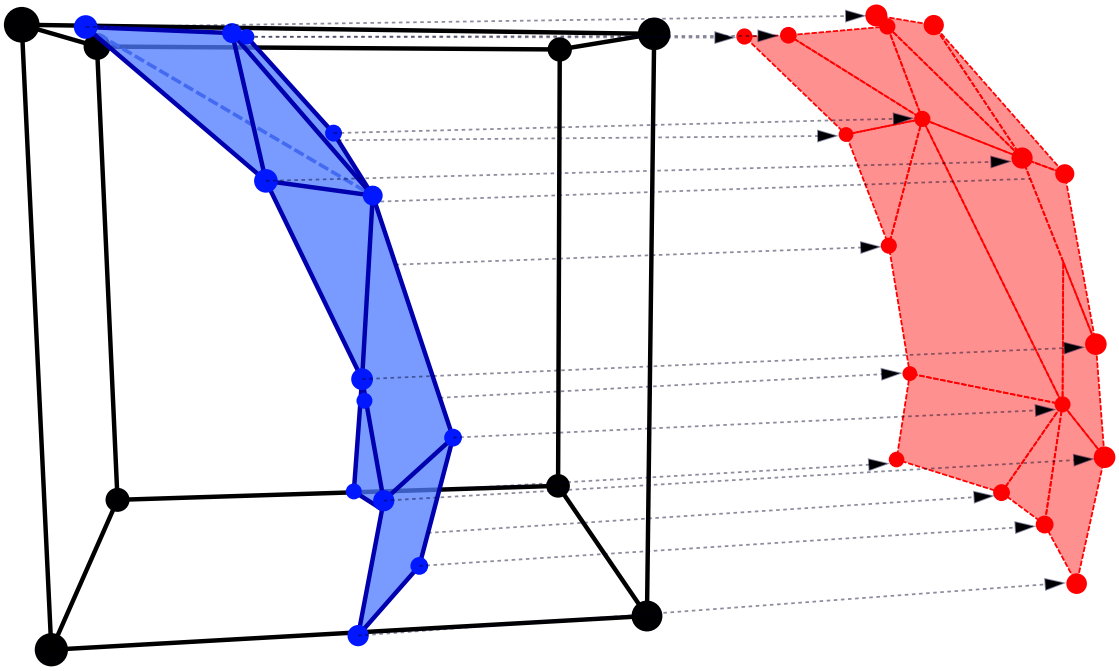


Figure 4.5: Penalty method where displacements are imposed on surface nodes that are embedded into one of the grid's cells.

Adding equations (4.19) and (4.21) into (4.9) gives the final set of equations

$$[\mathbf{K}(\boldsymbol{U}_k) + \boldsymbol{K}_s] \cdot \boldsymbol{U}_{k+1} = \boldsymbol{R}(\boldsymbol{U}_k) + \boldsymbol{S}(\boldsymbol{U}_k) + \boldsymbol{B} + \boldsymbol{T} \tag{4.22}$$

and has to be solved at each NR iteration k.

4.8 Preliminary validation of the Weighted Cell method

Intuitively, we expect that adjusting the weight of integration points while keeping their original positions within an element, as it is done with our proposed method, will reduce the accuracy of the solution. The exact numerical integration of a $2n-1$ degree polynomial used with a Gaussian quadrature of an n integration points element is no longer possible. Having fixed location for the points will also impact the quality of the displacement field representation. When keeping the original locations of the integration points of the regular 8-node hexahedron for an element that is only partially filled, some sections lying inside will probably not be well represented. The Finite Cell method is usually not impacted as much by this since it is adding a large number of integration points inside these regions.

However, it is important to put this accuracy drawback into perspective. While the solutions of the WC method may not be as accurate as the ones generated by the FC method, they could very well be within the acceptable error margin set by the surgeon during the simulation.

To measure the impact of our approximation over the FC method, we performed some preliminary experiments on some simple shapes. The objective of these experiments was to get a clear picture of the underlying properties of the method. In particular, we were interested in the accuracy of the solution for the following two deformation modes which are to be expected on a liver undergoing surgical manipulations : bending and stretching of the overall global shape of the organ. In both cases, the convergence rate and stability of the numerical process have been analyzed. The experiments have been done on cylindrical shape which allowed us to highlight the properties associated with a good diversity of cut elements, while keeping global proportions (length/width) that resemble those of a human liver.

Cylinder bending

Using a homogeneous neo-Hookean material with a Young's modulus of 5000 [Pa] and a Poisson's ratio of 0.3, we fixed the base of a cylinder having a radius of 5 [cm] and a length of 60 [cm]. Next, we applied a traction going down of 30 [Pa] at its top face (opposite face of the base). We computed a solution of the deformation (cf. figure 4.6) using a fine mesh of about 25k 10-nodes quadratic tetrahedral elements (~38k nodes). The relative L2-norm of the displacement error ($\|\boldsymbol{u} - \boldsymbol{u}_{\text{sol}}\|_{0,\Omega_{\text{sol}}} / \|\boldsymbol{u}_{\text{sol}}\|_{0,\Omega_{\text{sol}}}$) between our method (8-nodes regular hexahedrons), the Finite Cell method (8-nodes regular hexahedrons) and a 4-nodes tetrahedral mesh is shown in the figure 4.6.

As we increase the number of subdivisions of cut cells, both our method and the FC method tend to converge faster towards the solution. At an equivalent number of subdivisions, our method stayed within a 5% error from the Finite Cell method. However, the time required to compute the stiffness matrix of equation (4.10) remains the same with our method for any level of subdivisions. This is a very important distinction against the FC method where the time required by the later to build the system is directly related to both the number of subdivisions and the number of cut cells in the mesh. Figure 4.7 illustrates the average computational time (in milliseconds) taken to update the stiffness matrix at each Newton–Raphson (NR) iterations. Note that the 0-level of subdivision, often called a *sparse grid*, is simply a regular grid where hexahedral elements lying completely outside the surface are removed, and where elements cut by the boundary

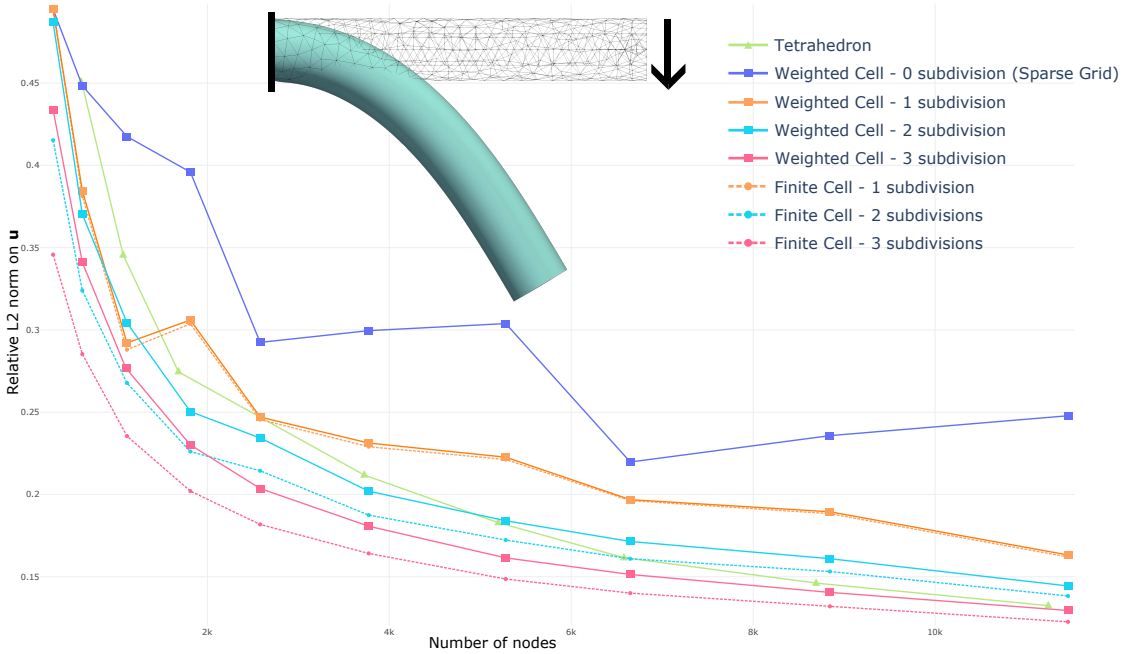


Figure 4.6: Comparison of a 3D cylinder modeled with an hyperelastic Neo-Hookean material, fixed at its base and bent by traction on its top face.

are treated as if they were lying completely inside the surface. As we refine the mesh, the sparse grid will sometimes fall into a configuration where many of its cells are badly cut by the surface. Because any cut cells are viewed as fully inside the surface, the solution at a given mesh size will often be worse than a coarser mesh where less badly cut elements were found, resulting in spikes in the convergence graph (as can be seen between the 2.5k and 7k nodes). These oscillations are rapidly smoothed out as the number of subdivisions grows.

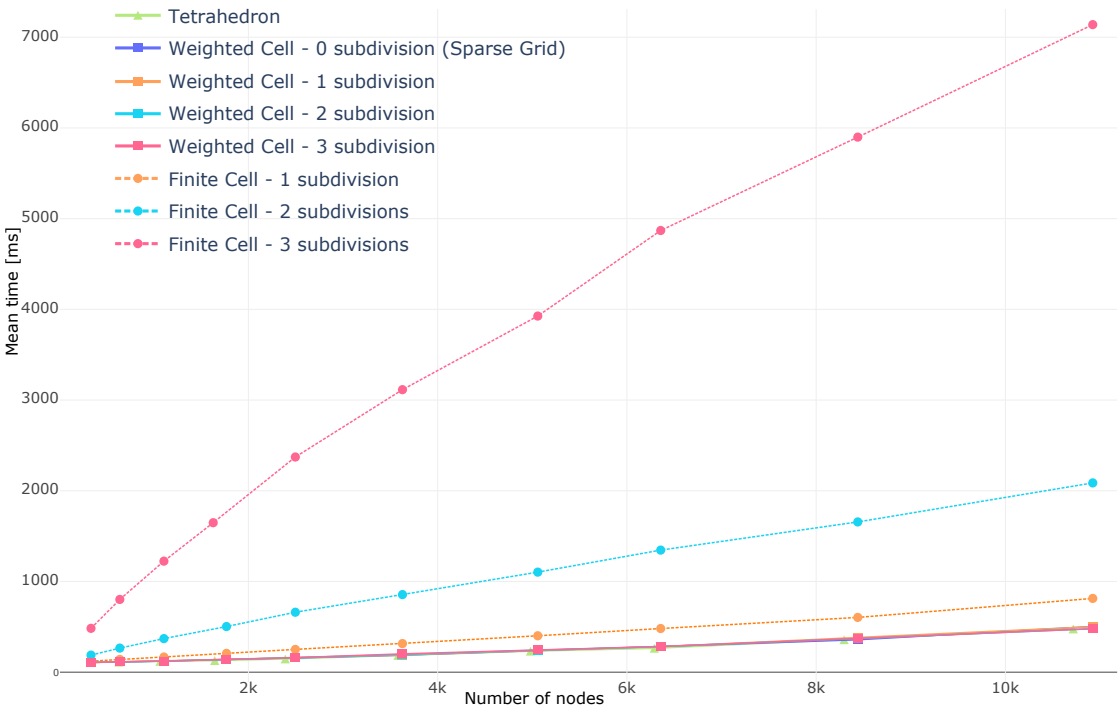


Figure 4.7: Mean time taken to update the stiffness matrix for the cylinder bending experiments.

Cylinder stretching

Stretch deformations are very interesting to study as they introduce a higher degree of complexity. Unlike bending, which introduces high displacements but low deformations, stretching rapidly produce high deformations which results in a numerical challenge. Using the same material as with the bending experiment, but this time setting the length of the cylinder to 5 [cm], we imposed a displacement on both the base and the top faces in the direction of their normals for a total expansion of 3 times its initial length (cf. figure 4.8). Again, the relative L2-norm of the displacement error between our method (8-nodes regular hexahedrons), the Finite Cell method (8-nodes regular hexahedrons) and a 4-nodes tetrahedral mesh is shown in the figure 4.8.

Here the accuracy curve looks similar to the one from the bending experiment, but this time with much lower relative displacement errors: the ratio between the number of elements and the length of the cylinder is higher, meaning that the meshes were finer compared to those used in the bending experiment. This is because it was not possible to create a sparser *and* boundary-conforming mesh (figure 4.1a)

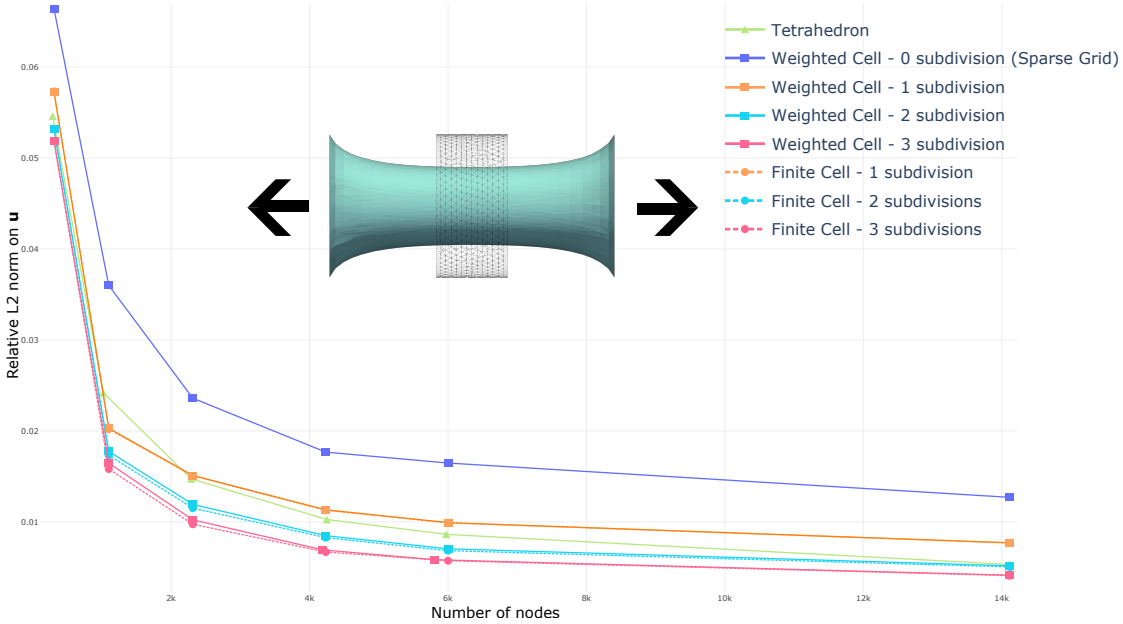


Figure 4.8: Comparison of a 3D cylinder modeled with an hyperelastic Neo-Hookean material. Both faces are imposed a displacement for a total expansion of 3 times the initial length of the cylinder.

using tetrahedral elements without creating degenerate elements. Degenerate elements have a tendency to rapidly create close to zero Jacobian of the deformation, which invalidates constitutive models that depend on it such as the NeoHookean material. Close to zero Jacobian can be viewed as having an element completely crushed. When the Jacobian becomes negative, the element is inverted (concave at a quadrature point).

These results highlight another important issue: large deformations increases the difficulty to numerically solve the system. As the mesh gets finer, smaller elements become more sensible to large deformation since the Jacobian quickly reaches an invalid state. The usual workaround is to split the imposed displacement into smaller increments. Since NR iterations have to be run at each increment, reaching the final solution will take significantly more time. Looking at the figure 4.9, we can see that as the mesh get finer, smaller displacement increments are required. This is even worse for the tetrahedral mesh. Note that we haven't included the Finite Cell method with more than two subdivisions. This is because, even when going as far as 150 increments, the method was always diverging at some point during the simulation. In fact, to compute the L2-norm presented in the figure 4.8 for the Finite Cell method, we had to remove cells that had less than 5% of their volume inside the boundary. Our Weighted Cell (WC) method stayed stable for

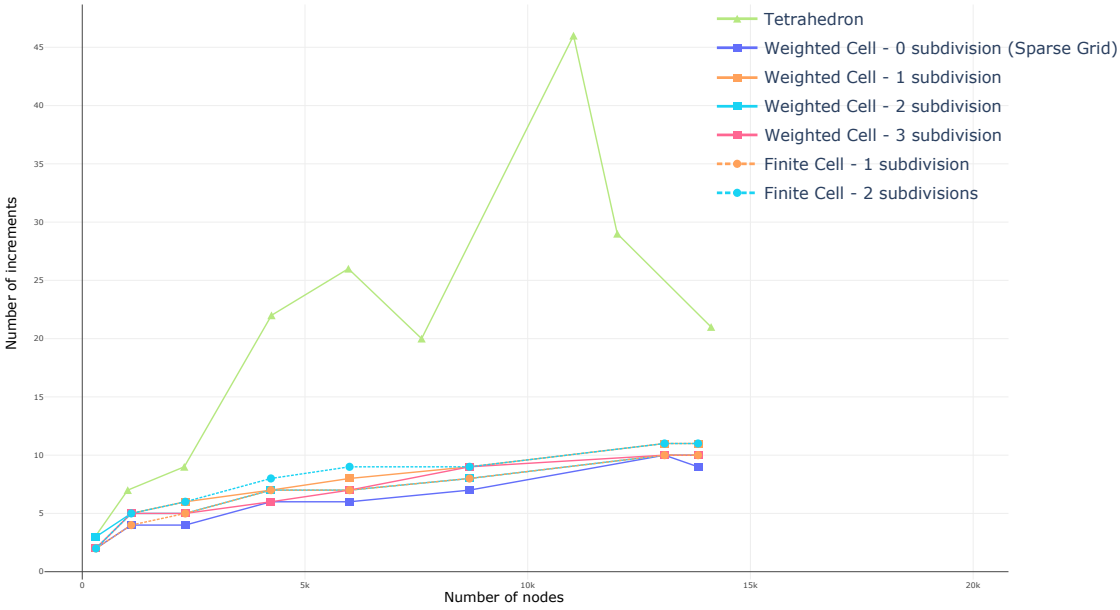


Figure 4.9: Minimum number of increments needed for the stretching experiment. Larger increments create negative jacobians and make the simulation diverge.

all levels of subdivisions without the need to remove any cells.

Preconditioning

To solve equation (4.22), an iterative linear solver such as the Conjugate Gradient (CG) method is preferred as it allows us to balance out the accuracy of the solution against the maximum computational time allowed by the application. It is also a great method to produce an approximate solution for a given displacement increment when, for example, the system is considered as under-constrained by a direct solver. This happens quite frequently with our type of applications as the boundary conditions are imposed using penalty forces.

Immersed boundary methods face a numerical hindrance when they involve a CG solver. When cells are cut by the boundary such that only a very small portion of their volume remains inside the simulated object, the tangent stiffness matrix becomes ill-conditioned. A large number of CG iterations are then required to achieve convergence. We ran an experiment where, using the same bending cylinder beam as in figure 4.6, we simply scaled the fictitious grid while keeping the same number of cells and embedded surface. Figure 4.10 illustrates this process. In the first configuration, all cut cells are considered as well cut by the boundary.

The second configuration contains cells that are moderately cut. Finally, the third configuration contains cells that are badly cut by the boundary. The residual of the CG is shown for all three configurations.

We can see that, as the outside region of a cut cell grows, the CG requires a lot more iterations per NR steps in order to converge. At the end of the simulation, all three configurations converged to the same solution. However the second and third configurations took a lot more time to compute than the first one. A good review of the source of this ill-conditioning behavior and some numerical methods that can alleviate it can be found in Prenter et al. (2017).

In our case, however, the tangent stiffness matrix is sparse and diagonally dominant. Hence, we found that some simple global preconditioners such as the diagonal (often called Jacobi) preconditioner and the Incomplete Cholesky preconditioner were more than enough to damp the numerical effect of badly cut cells on the CG convergence rate (figure 4.10). In fact, a diagonal preconditioner was used for all our experiments since it is computationally very efficient and holds comparable results as the Incomplete Cholesky. We are aware that more advanced methods such as the ghost penalty method [Burman, Claus, et al. (2015)] might be more appropriate for some scenarios. This is something that should be looked at in future research. However, throughout this work, we have not found any configuration for which the diagonal preconditioner was not sufficient.

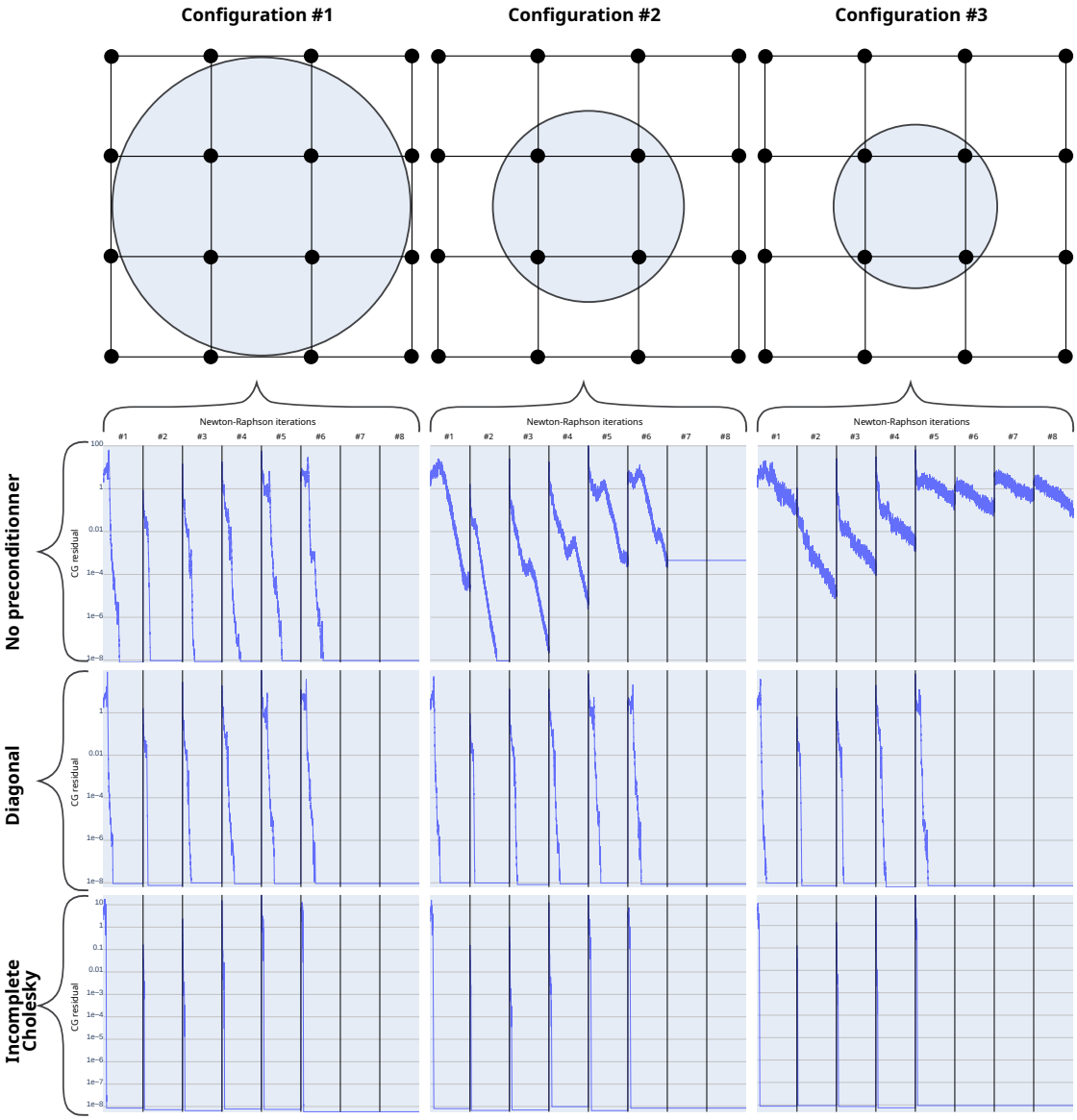


Figure 4.10: Convergence of the conjugate gradient algorithm for the beam experiment. By expanding the size of the grid, we can control how bad the surface of the cylinder is cutting elements. All three configurations yield the same solution. However, when no preconditioner are used, the number of CG iterations required for a Newton-Raphson step to converge grows as the quality of cut elements degrades.

4.9 Experiments performed on an in-vivo porcine liver

In order to validate our method on a more complex geometry, we used the data from three in-vivo porcine liver trials. For each trial, CT images were acquired before and after pneumoperitoneum set at 13 mmHg. Details on how these images were acquired is provided in chapter 5.

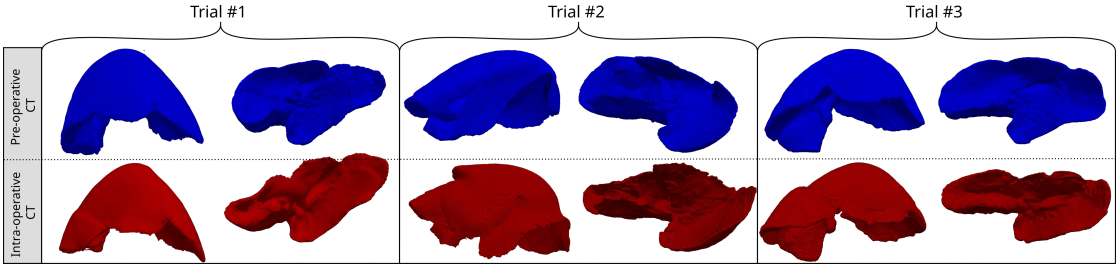


Figure 4.11: Pre-operative CT and intra-operative CT surface reconstructions for all three trials.

The general idea of the exercise was to embed the surface mesh of a pre-operative surface reconstruction into IB meshes of different sizes, and to generate large deformation by imposing displacements over its whole surface. Hence, the liver deformation caused by the pneumoperitoneum constituted an excellent candidate to this type of analysis as well as providing us with some real-life scenarios.

Generating a set of imposed displacements to deform a model from one surface reconstruction to another is a very complicated process. In fact, this process constitutes a complete branch of research in itself and is called the *non-rigid registration*. This will be the main focus of chapter 5. Here, however, we simply wanted to analyze the convergence rate of our method without incorporating errors which are inherent to the registration methods. In other words, we wanted to analyze how our two methods behave under a perfect registration between the pre-and intra-operative surface reconstructions. Also, for this validation exercise, we were not interested in the consequences of the choice of material on the accuracy of the solution. This analysis will be done in chapter 5 using intra-operative data and a more accurate validation process.

The experiments with porcine livers have allowed us to analyze our methods using realistic *synthetic* solutions which were generated as follows. Using a very fine IB mesh, a bio-mechanical model of the pre-operative reconstruction has been deformed using a non-rigid ICP registration method that we will describe in chapter 5 (see algorithms 3). The resulting deformed mesh was then stored and labeled

as our "synthetic intra-operative" solution. This workflow is illustrated in figure 4.12. Here, the deformed fine mesh gives us a complete displacement field solution which can then be used for an h-refinement convergence analysis. In addition, we can use the resulting displacement field as a way to impose the displacements over the initial and undeformed surface mesh of the liver. The accuracy of our penalty method can thereby be analyzed properly.

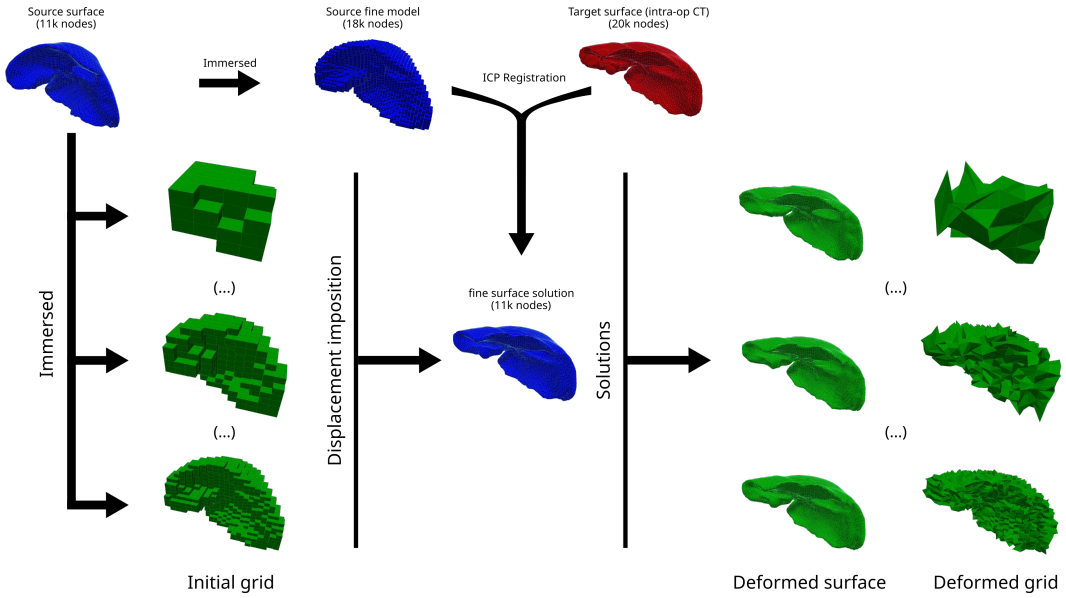


Figure 4.12: Workflow of the in-vivo porcine liver experiment (trial 1)

For each of the three trials, the experiment was conducted as follows. First, the pre-operative surface reconstruction was immersed into a coarse IB grid having a resolution of $n \times n \times n$ nodes. The dimensions of the grid boundaries were simply set to the dimensions of the surface mesh's bounding box. The model was then initialized using the Weighted Cell (WC) approach presented in section 4.5 with four levels of subdivisions. A Saint Venant–Kirchhoff material (section 2.3.1) was selected with a Young modulus set to 5000 [MPa] and Poisson's ratio set to 0.499. Using our displacement field solution from the very fine IB mesh, we stored the displacement vectors of each of the immersed surface nodes. These displacement vectors were imposed as Dirichlet conditions using our penalty method presented in the section 4.7. The stiffness of the penalty factors was initially set to a low value of about 100 [MPa], and then incrementally increased following an exponential curve discretized into 30 increments towards a maximum of $1e^7$ [MPa]. Between each penalty increment, our Newton–Raphson implementation (algorithm 1) was used to solve the system with a maximum of 10 iterations and a convergence criterion set to $1e^{-8}$. This scenario was repeated for different grid resolutions, i.e., using

resolutions of $(n+i) \times (n+i) \times (n+i)$ with $i = \{1, 2, \dots\}$.

The displacement field solution of the fine mesh was used afterwards to compute a full volumetric assessment of the approximation error between the different mesh sizes. If we let $\boldsymbol{u}(\boldsymbol{X})$ be the solution field of one of the meshes, and $\boldsymbol{u}_{fine}(\boldsymbol{X})$ the solution field of the fine mesh, then the relative L2-norm of the error as defined by

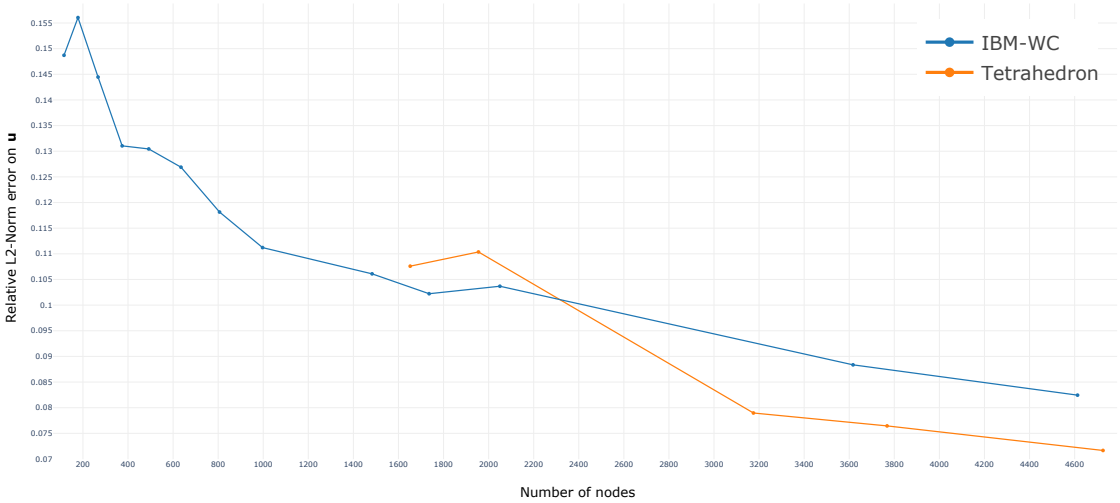
$$e = \left(\frac{\int_\Omega (\boldsymbol{u} - \boldsymbol{u}_{fine})^2}{\int_\Omega \boldsymbol{u}_{fine}^2} \right)^{\frac{1}{2}} \tag{4.23}$$

is illustrated in the figure 4.13 for all grid resolutions. We have also performed the same experiment with a FE method using meshes of different sizes and composed of linear tetrahedrons. The tetrahedral meshes were generated using the CGAL library [1]. Since we did not implement any IB method for tetrahedrons, special attention has been made to make sure these meshes were strictly following boundaries of the surface mesh. We also made sure that all generated elements were well formed. For this reason, we were not able to generate tetrahedral meshes that are as coarse as those made by the grids of our WC method, which highlight again a net benefit of IB methods.

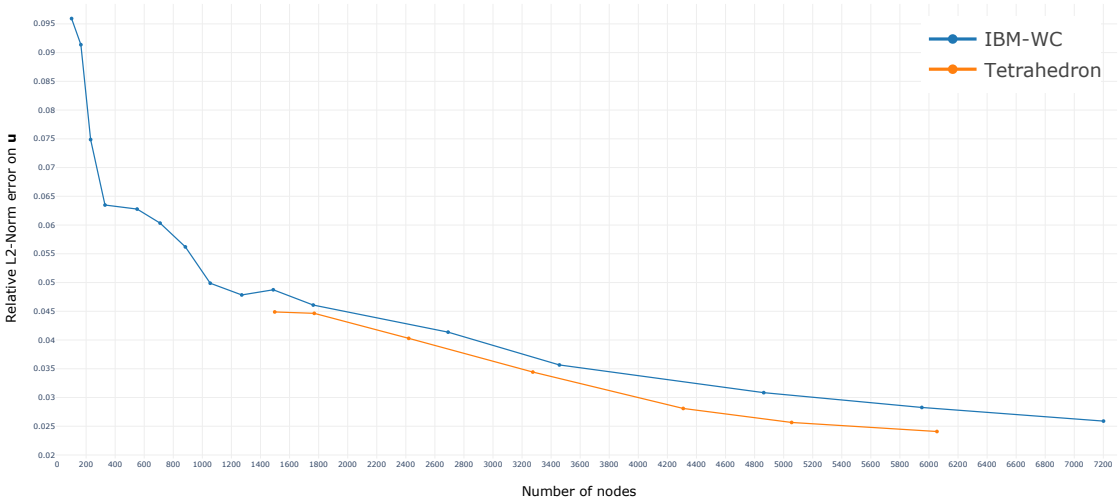
From the error measurement provided in figure 4.13, we can see that even in the worst case, the convergence rate is similar to the one of the standard finite element method. For the third trial, our method converges much faster than the FE approach, i.e., a prescribed accuracy would be achieved even if fewer nodes are used.

Figure 4.14 illustrates the mean error of the position of every node on the embedded surface mesh with respect to the expected solution. Using this additional measurement, we can further evaluate the performance of our displacement imposition method. It is quite surprising to observe such a small error when very coarse grids are used. For example, using merely 500 nodes, our method can provide an average error of less than 1.5 [mm] when compared to our solution made of approximately $18k$ nodes, or $54k$ degrees of freedom.

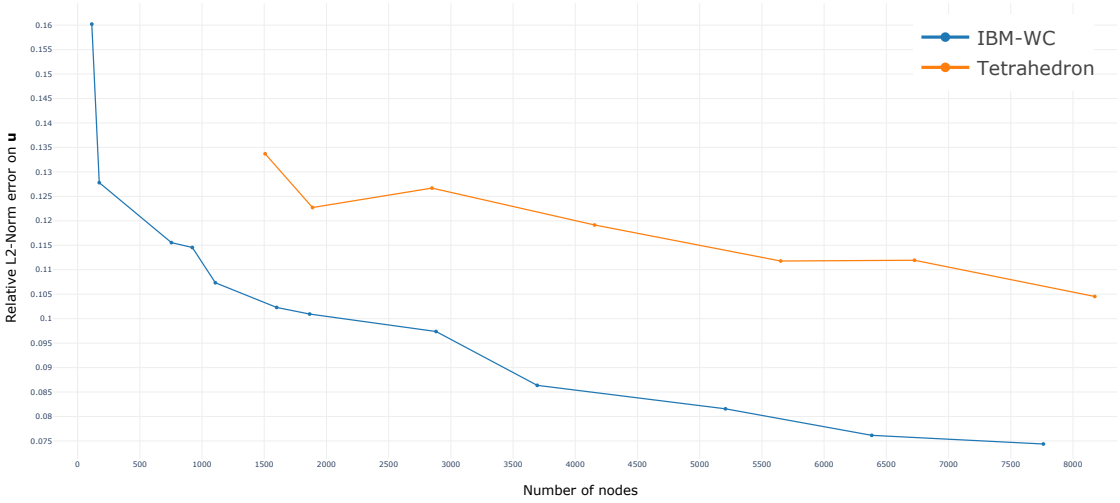
[1] https://www.cgal.org/



(a) Trial 1

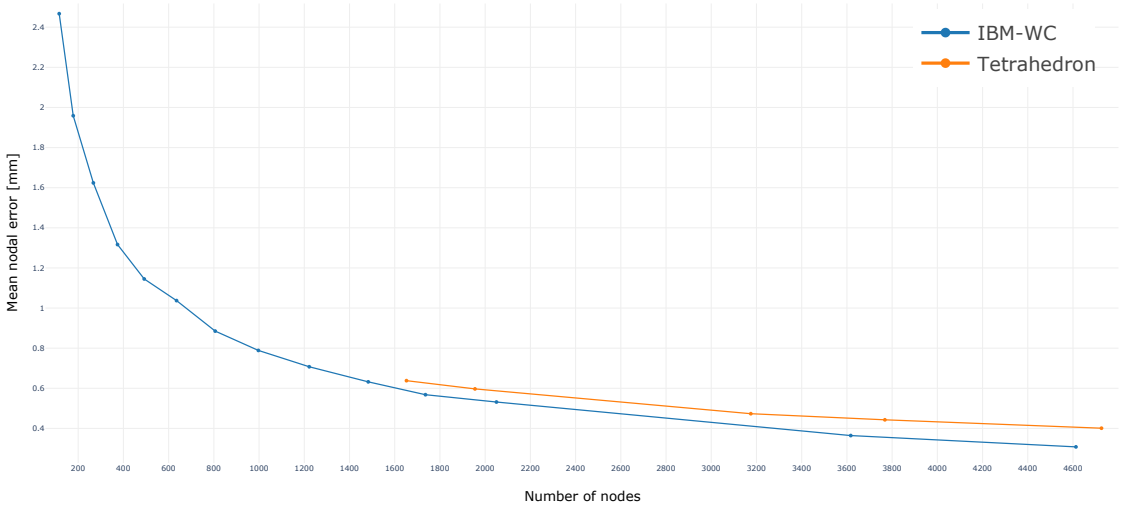


(b) Trial 2

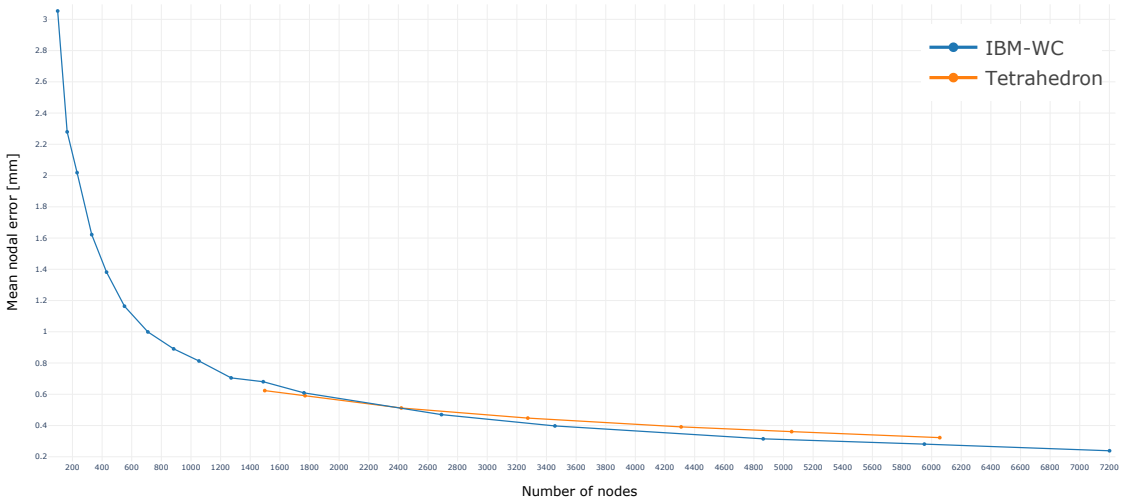


(c) Trial 3

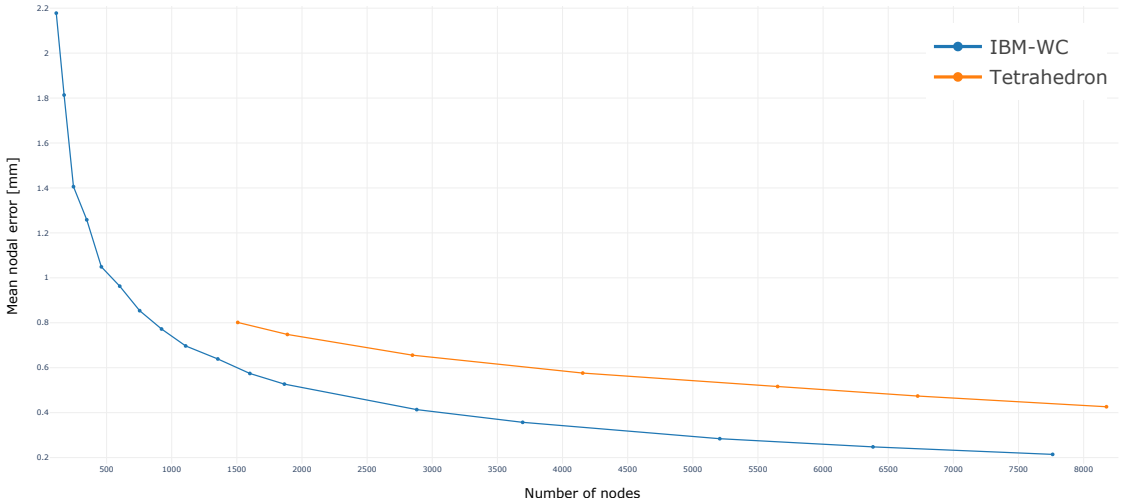
Figure 4.13: L2-norm of the error between different mesh sizes and the solution of the same method using a very fine mesh.



(a) Trial 1



(b) Trial 2

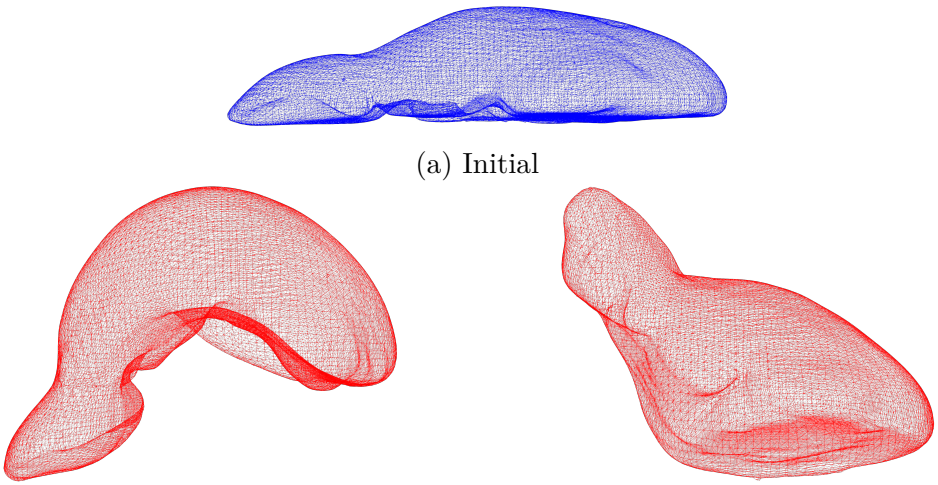


(c) Trial 3

Figure 4.14: Mean position error of all surface nodes against the solution of the same method using a very fine mesh and four levels of subdivision.

4.10 Experiment performed on an ex-vivo human liver

To pursue one step further our analysis with a realistic synthetic solution, we conducted the exact same experiment but this time using an ex-vivo human liver (the details of the acquisition process that was used for this experiment are provided in 5.5). In this case, an initial CT scan was taken with the organ lying at rest on a table (figure 4.15a). Two other scans were then taken after generating two distinct deformations (figures 4.15b and 4.15c).



(a) Initial

(b) Deformation #1

(c) Deformation #2

Figure 4.15: CT surface reconstructions made before and after deformation

The purpose of this additional experiment was twofold. First, the shape of a human liver is less complex than that of a porcine liver. The porcine liver has a concave shape due to the discontinuities between its lobes, while the shape of a human liver is globally more convex [Nykonenko, Vávra, and Zonča (2017)]. We were therefore interested to see the accuracy of our WC method when applied to a simpler shape. Secondly, we were interested in analyzing deformations that are as close as possible to those observed in an open surgery. Instead of having a uniform pressure field which was generated by the pneumoperitoneum in our previous experiment, here the liver is simply pulled up from either the tip of its lobe (figure 4.15c) or directly underneath it (figure 4.15b).

As it was done with the porcine trials, the human liver was initialized using our Weighted Cell (WC) approach and four levels of subdivisions. An Saint Venant–Kirchhoff material (section 2.3.1) was selected with a Young modulus set to 5000 [MPa] and Poisson's ratio set to 0.499.

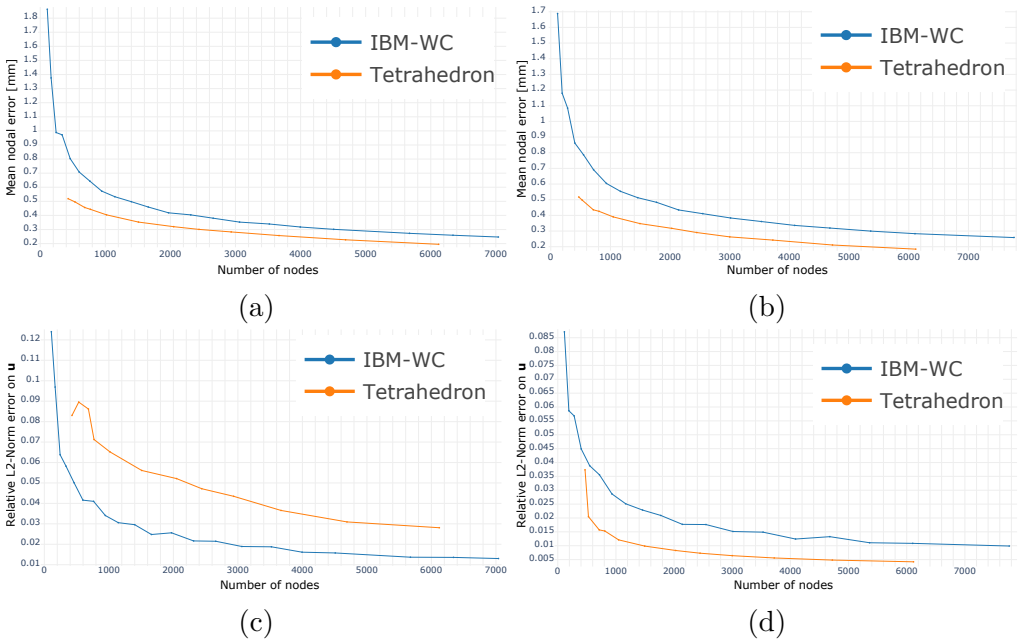


Figure 4.16: (a)-(b) Mean position error of all surface nodes against the solution. (c)-(d) L2-norm of the error between different mesh sizes and the solution. (a)-(c) Deformation #1 (b)-(d) Deformation #2

Figures 4.16a and 4.16b show the mean surface nodal error for the first and second deformations respectively. Similarly, figures 4.16c and 4.16d show the relative L2-norm error (equation 4.23) of the displacement field for the first and second deformations respectively. These results suggest once more that the converge rate of both our WC method and a standard FE are within the same order. However, the results yielded by the standard FE method appears to be slightly better. Since the human liver had a less complex shape, tetrahedral meshes were easily created for various sizes. The higher quality meshes would explain these better results from the standard FE method.

Although we would have preferred higher convergence rates, the results generated by our IB method are still very much in line with our initial objectives. Having a convergence rate so close to the FE one without having to build boundary-fit meshes is a very satisfying result.

4.11 Discussions

In this chapter, we have analyzed the Immersed boundary methods which are the second category of simulation models considered in our work. Contrary to the meshless methods studied in chapter 3, IB methods reuse the iso-parametric paradigm found in standard FE models for both the interpolation and the integration of a field function. This proximity to the FE methodology gives access to a rich research literature and allows us to reuse available software codes, which simplifies considerably the implementation effort. But unlike standard FE methods, the IB methods are allowed to use a computational mesh that does not conform to the boundaries of the simulated domain. This represents a substantial relief for the end user. In fact, the challenging work associated with the discretization of the domain of interest is reduced to a simple process of adding a background grid over the simulated object. To get finer solutions, the resolution of the grid needs only to be refined. Thenceforth, the only remaining difficulty pertains to those elements that are cut by the boundaries.

Our preliminary validation using simple shapes has shown that the Finite Cell method produces very accurate solutions while maintaining a better convergence rate compared to a standard FE method using tetrahedral elements. However, the time required for the assembly of the tangent stiffness matrix rapidly increases with the number of cell subdivisions. Since we are looking to simulate non-linear materials, this assembly has to be carried out at every NR iterations. Hence, the addition of many integration points required for cut cells results in computational time requirement that is definitely not appropriate for our type of applications.

On the other hand, the preliminary validation performed on our proposed WC method have shown that it can surprisingly produce a solution very close to the FC method, without the need for having additional integration points. Furthermore, its implementation can be easily integrated with existing FE software and thereby inherit the various computational optimizations made for standard 8-node 8-integration points elements. The results also demonstrated that the method is sufficiently robust against large load increments, even with the use of a Neo-Hookean material which is very sensitive to element inversions.

Our analysis involving more complex shapes such as the three porcine livers and the human liver revealed interesting results as well as some drawbacks that had not been observed during our preliminary tests. Contrary to our expectations, the WC method is not sufficiently robust for simulations involving a Neo-Hookean material. When very large penalty forces were imposed, some of the cut cells were

often reaching a degenerate state, which in turns resulted in a zero or negative jacobian of the deformation tensor. We are still not sure if this issue can be resolved. One solution could be to use another approach to impose displacements on the immersed boundary such as Lagrange multipliers. Restricting the material with addition constraints in order to have a fully incompressible behavior might also be a good candidate solution. However, this solution would add considerable computational effort. Note that our FE solutions also suffered from these numerical difficulties.

Finally, in all the trials performed on a real liver, large displacement impositions had to be broken into many smaller increments otherwise the results would be just too far from the expected solution and the number of Newton–Raphson iterations would increase dramatically. This is a non-negligible drawback and further research will have to be considered in order to allow bigger increments, especially for non-rigid registrations from a pre-operative liver reconstruction into a highly deformed shape occurring during the intervention. Since coarse grids were able to converge with much higher increment steps, we believe that some hierarchical schemes would be appropriate here as a good starting point. For instance, a coarse grid could be first solved as an approximate solution to an embedded finer grid. This approximation would then be used for the initial state of the NR method. It is important to note here that this drawback is not unique to our proposed method and is also present in the standard FE methods or in alternative IB methods. However, the implementation of hierarchical schemes is much more straightforward to implement on a lattice geometry. This is because a mapping between coarse and finer grids of rectangular hexahedral cells can be naturally prescribed. This constitutes a net advantage of our method.

Overall, although further research will be required, we are confident that our simple yet quite efficient WC approach constitutes a very good alternative to standard FE methods.

Chapter

5

IMPLEMENTATION OF A NON-RIGID REGISTRATION PIPELINE

We now move to the final phase of our work which is the implementation of a complete algorithm for our non-rigid registration pipeline. We will start with a quick introduction of the various techniques that are commonly used to reconstruct a three-dimensional point cloud from the partial surface of an organ viewed by a camera during surgery. We will then present a first rigid registration algorithm to rigidly align the pre-operative model of the liver to the augmented reality display using only simple rotations and displacements. We will explain how we expanded this algorithm to cover non-rigid transformations using a biomechanical model, and more specifically our WC method. The resulting non-rigid pipeline will finally be evaluated using real clinical scenarios involving in-vivo porcine and ex-vivo human livers.

This chapter also includes a brief discussion of some of the artificial intelligence

and machine learning concepts that are taking up more and more place in the literature. We will describe how this approach can be integrated in our proposed non-rigid pipeline to rapidly solve the deformed state of our biomechanical model. More importantly, we will show how the numerical methods developed in this thesis can be used to efficiently train a deep neural network.

5.1 Surface reconstruction

The concept of boundary conditions aforementioned in the previous chapters is one of the key elements that allow our model to be guided towards a unique deformed solution. The process used to transform the acquired intra-operative information into a point cloud, which can then be used to imposed boundary conditions is therefore the first component that needs to be addressed.

Extracting information on the state of the patient's liver during surgery brings a number of technical challenges. Some operation rooms are equipped with advanced 3D medical images tools such as 3D Ultrasound (US) probes, intra-operative CT scans located directly at the operation table, and sometimes even an MRI is even available just next door to the OR. This type of setup remains, however, quite rare and, even if it was more accessible, the time required to acquire the 3D images and go through the whole segmentation process would be simply too long for our workflow.

In laparoscopic interventions, a valuable source of information is directly available: the laparoscope stereo camera. In the case of open surgery, adding a camera on top of the patient is also a great way to acquire the same type information. In this case, special types of camera - 3D RGB-D (Red Green Blue - Depth) - can even be used to retrieve a lot more information on the deformed state of the patient's liver. The size of RGB-D cameras is, however, too large for any uses in laparoscopic interventions. Hopefully, this will change in the future as RGB-D cameras are getting smaller.

The general idea here is therefore to extract and reconstruct the partially visible part of the liver captured by a camera. As a starting point, matches between pairs of stereo images acquired at the same time are located and used to extract depth information [Scharstein and Szeliski (2002); Brown, Burschka, and Hager (2003); Hartley and Zisserman (2004)]. To circumvent some issues found in laparoscope images such as occlusions from surgical tools, homogeneous textures of the organ, and light reflections, specific methods targeted to CAI applications in MIS

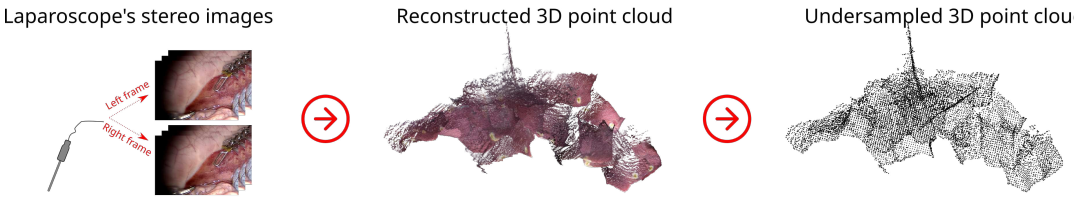


Figure 5.1: 3D point cloud reconstruction of the visible surface from laparoscope stereo images

were developed [Stoyanov et al. (2010); Röhl et al. (2012); Bernhardt, Abi-Nahed, and Abugharbieh (2012); Wang et al. (2018)]. Once correspondences are found, a triangulation process takes place, yielding a 3D point cloud. Post-processing algorithms are often used for smoothing and to remove outliers from the noisy point cloud [Haouchine, Roy, et al. (2016); Dey and Giesen (2001)].

Constructing 3D shapes from 2D images and producing accurate point clouds is in itself a complete field of research and was outside the scope of our work. The experiments performed on in-vivo porcine livers were done using a stereo reconstruction technique from Wang et al. (2018). For the experiments on the ex-vivo human liver, the point clouds were generated directly from an RGB-D camera.

5.2 Initial rigid registration

The 3D point cloud reconstructed from the visible surface part of the liver will initially lay inside its own coordinate system which is relative to the camera point of view. The original positioning will therefore be quite different from the one of the pre-operative biomechanical model. Phrased differently, the point cloud might be far and unaligned with respect to the position and orientation of the virtual model. As we have seen in chapter 2, deformations of a non-linear hyperelastic material are solved using a Newton–Raphson (NR) iterative method which requires that the simulated solution and the deformed virtual model are relatively close to each other. It would therefore be difficult, if not impossible, to directly impose the point cloud as the displacement constraints. An initial rigid registration method is therefore required to estimate the rigid motion (translation and rotation) that will project the coordinates of both the biomechanical model and the point cloud into a unified space where both representations are well aligned and oriented. This usually boils down to a minimization problem between two finite sets of points:

the *source* point cloud $\mathcal{S}$, in our case the pre-operative liver surface, and the *target* point cloud $\mathcal{T}$, the intra-operative surface reconstruction. We then seek to find the rigid transformation $\boldsymbol{T}$ such that the distance between $\boldsymbol{T}(\mathcal{S})$ and the target is small. The distance measure, defined as $meas : \boldsymbol{T}(\mathcal{S}) \times \Omega_s \rightarrow \Re$, is application specific and will give an approximation of how close the overall shape of the source is to the target one. In other words, we wish to find the optimal transformation that rigidly moves (i.e., the distance between two points does not change) the source's points cloud closer to the target:

$$\arg\min_{\boldsymbol{T}}[meas(\boldsymbol{T}(\mathcal{S}), \mathcal{T})] \tag{5.1}$$

A popular choice for the distance measure is the square of Euclidian distance between every pair of closest points, i.e.:

$$meas(\mathcal{M}, \mathcal{N}) = \sum_{m \in \mathcal{M}} \|m - n'\|_2^2 \qquad \text{with} \qquad n' = \arg\min_{n \in \mathcal{N}} \|m - n\|_2^2 \tag{5.2}$$

We found that the most frequently used method to extract the transformation is the Iterative closest point (ICP) method [Besl and McKay (1992); Plantefève, Peterlik, Haouchine, et al. (2016)]. It usually goes as follows:

1. Initialize $\boldsymbol{T} = (\boldsymbol{R}, \boldsymbol{t})$ where $\boldsymbol{R} \in \Re^{3\times3}$ is a rotation matrix, and $\boldsymbol{t} \in \Re^3$ is a translation vector
2. Compute the initial distance measure $d^0 = meas(\boldsymbol{T}(\mathcal{S}), \mathcal{T})$
3. For each point in the source $\mathcal{S}$, find its closest point in the target $\mathcal{T}$
4. Update alignment parameters $(\boldsymbol{R}, \boldsymbol{t})$
5. Compute the current distance measure $d = meas(\boldsymbol{T}(\mathcal{S}), \mathcal{T})$
6. If $\frac{d}{d^0} > \epsilon$ where ϵ is a given relative residual threshold, go back to 3

Many variants of the ICP method exist to improve either its execution time or its accuracy depending on the applications. For example, Z. Zhang (1994) proposed to improve the closest point search using a k-d tree algorithm. In Masuda and

Yokoya (1995), random point sampling sets are used together with an Least median of squares (LMS) estimator to find correspondences. Weighted distance measures are also often used. In Clements et al. (2008), they improve the ICP by adding weigh between correspondent anatomical features detected in the intra-operative image and the pre-operative reconstruction. A similar technique was proposed in Plantefève, Peterlik, Haouchine, et al. (2016) where the umbilical notch, the anterior margin of the liver, and the vena cava were used as landmarks to guide the ICP towards a more accurate solution.

In all cases, the outliers are usually removed by comparing the distance between corresponding points. We have used the following approach to identify outliers: a correspondence between two points is assumed to be an outlier if the Euclidian distance between them is greater than twice the standard deviation of the distance found for all correspondences.

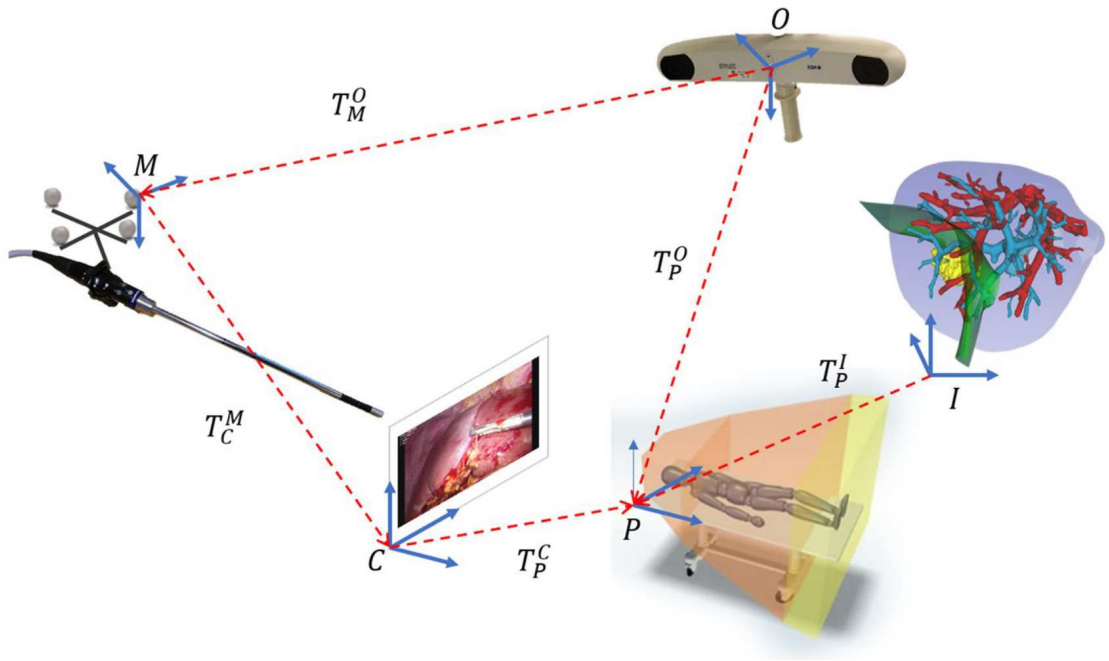


Figure 5.2: Initial rigid registration based on an optical tracking system. Image taken from Teatini et al. (2019).

When available, we also used an optical tracking system to compute the initial transformation before starting the ICP iterations. Based on the work of Teatini et al. (2019), we tracked the laparoscope by fixing physical markers on it, and using an optical camera inside the OR to locate it in space. Figure 5.2 illustrates this process where the initial transformation is given by

$$\boldsymbol{T}_{init} = \boldsymbol{T}_C^M \cdot \boldsymbol{T}_M^O \cdot (\boldsymbol{T}_P^O)^{-1} \cdot \boldsymbol{T}_P^I \tag{5.3}$$

Here, $\boldsymbol{M}$ is the laparoscope's frame with respect to the optical camera's frame $\boldsymbol{O}$ and calibrated using Z. Zhang (2000), $\boldsymbol{P}$ is the operation table frame, $\boldsymbol{C}$ is the point cloud frame, and $\boldsymbol{I}$ is the pre-operative reconstruction frame. The resulting ICP implementation version is provided in Algorithm 2.

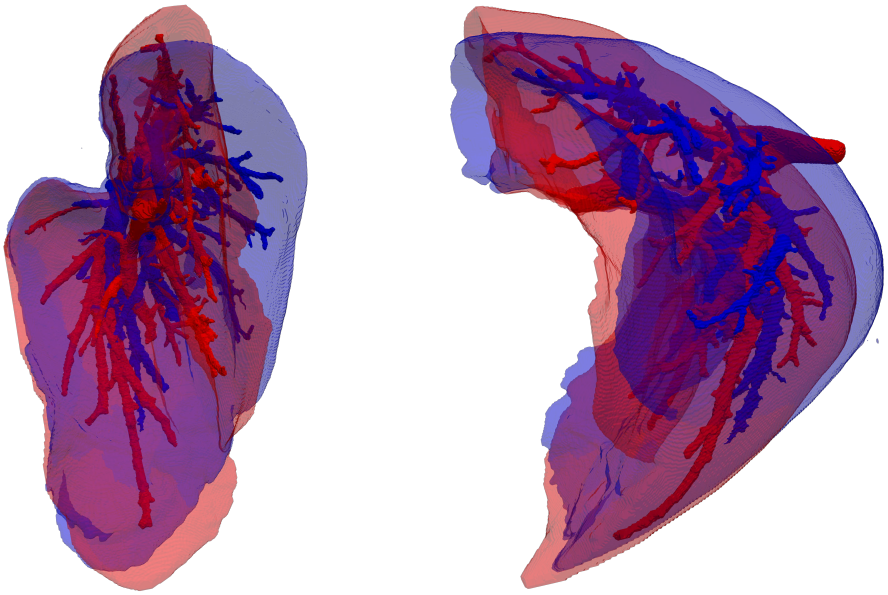
Algorithm 2 Rigid ICP registration

procedure RIGID_ICP($\mathcal{S}, \mathcal{T}, \epsilon$)
  $\boldsymbol{T} \leftarrow \boldsymbol{T}_{init}$ ▷ Using eq. (5.3)
  $\mathcal{S} \leftarrow \boldsymbol{T}(\mathcal{S})$
  $d^0 \leftarrow meas(\mathcal{S}, \mathcal{T})$ ▷ Using eq. (5.2)
  $d \leftarrow d^0$
  while $\frac{d}{d^0} > \epsilon$ **do**
    $\mathcal{N} \leftarrow find\_closest(\mathcal{S}, \mathcal{T})$ ▷ Using k-d tree
    $\mathcal{N} \leftarrow remove\_outliers(\mathcal{N})$
    $\boldsymbol{T} \leftarrow update\_transformation(\mathcal{S}, \mathcal{T}, \mathcal{N})$
    $\mathcal{S} \leftarrow \boldsymbol{T}(\mathcal{S})$
    $d \leftarrow meas(\mathcal{S}, \mathcal{T})$ ▷ Using eq. (5.2)

5.3 Deformable registration

Even with a perfect surface reconstruction of the intra-operative data and an optimal rigid transformation, chances are that the registration of the pre-operative model onto the intra-operative reconstruction will not be fully representative of the current shape of the patient's liver. This is because a rigid registration does not account for the deformation undergone by the liver either before or during the surgery. For example, the pneumoperitoneum in MIS will impose a surface pressure of about 13$[mmHg]$ (133$[Pa]$). Gravity might also play its part in the difference of shapes since pre-operative images are often taken when the patient is lying in a position that will be quite different than its position during the surgery.

Figure 5.3 shows an example of the deformation undergone by a porcine liver from the pneumoperitoneum. A CT scan was taken both before and during the surgery. The rigid registration described in the previous section was used to align both surface reconstructions together. We can see that the overall shape of the



(a) Bottom view (b) Side view

Figure 5.3: Rigid registration of a porcine liver using algorithm 2. The red mesh is the surface reconstruction from the intra-operative CT scan. The blue mesh is the surface reconstruction of the pre-operative CT scan.

liver changes significantly after the pneumoperitoneum. More importantly, the deformed shape of the liver makes the internal structures of the registered pre-operative reconstruction completely useless for a CAI. We clearly see that some sections of the virtual model do not match the actual shape of the liver.

Non-rigid registration methods are extended versions of the rigid registration techniques. On top of rigid motions such as rotations and translations, they can also address affine (rotation, translation, scale and shear), projective or curve transformations. Registration techniques that handle a curve transformation extraction are commonly called *deformable registration* or *elastic registration* methods. For the registration between medical image modalities, two types of curved techniques are typically used [Oliveira and Tavares (2014)]. The first ones are the *Free-form* methods that can extract any type of deformations between a source and a target domain using curved-fitting techniques such as b-spline [Sotiras, Davatzikos, and Paragios (2013)]. The second ones are the *Guided* methods that will typically be restricted to deformations that correspond to specific mechanical behavior such as soft tissue elasticity or viscoelasticity and fluids [Sotiras, Davatzikos, and Paragios (2013); Oliveira and Tavares (2014)]. We selected this second approach to let the hyperelastic behavior of the liver's parenchyma guide the registration process

between modalities.

Injecting partial knowledge of the deformed state into the biomechanical model is not a new concept. It has been studied in many ways and forms in the past two decades either by altering the biomechanical itself or by improving the intra-operative surface reconstruction using correspondence with some specific anatomical features. In Miga et al. (2003), authors have described a non-rigid registration method where a human liver is modeled using a homogeneous corotated linear elastic material [Müller, Dorsey, et al. (2002); Carlos A. Felippa and Haugen (2005)]. The intra-operative partial reconstruction was done using a laser-range scan (LRS) of the liver surface performed during an open surgery. A rigid registration using ICP was initially done to align the biomechanical model to the first reconstructed point cloud. Cash et al. (2005) later improved this method by proposing an incremental FEM approach coupled with better correspondences between point clouds for the application of boundary conditions. Speidel et al. (2011) investigated the impact of using linear elasticity to model deformations induced by respiratory motion and instrument collision using a comparison with a non-linear Neohookean material. Haouchine, Dequidt, et al. (2013) introduced a semi-heterogeneous model where large blood vessels are modeled through homogeneous anisotropic elastic beams [Duriez et al. (2006)] and the rest of the parenchyma with homogeneous corotated elastic tetrahedrons. They used elastic springs to couple both materials. Similarly, Courtecuisse et al. (2014) used the same semi-heterogeneous model but proposed a Lagrange multipliers approach instead of springs for the coupling. In Plantefève, Peterlik, Haouchine, et al. (2016), a similar model was used for the non-rigid registration, but using the Glisson's capsule in order to add an additional level of heterogeneity. For this purpose, constant strain triangular elements were used. Homogeneous corotated linear elastic materials were used for the Glisson's capsule and the parenchyma without blood vessels. C. J. Paulus et al. (2017) proposed a method to handle topological changes induced by a surgical cut of an organ during laparoscopic procedures. Their method automatically detects cuts using a Euclidian distance criterion between two feature points. Local linear tetrahedral remeshing is then performed in cut regions. Experiments were done on ex-vivo porcine kidneys and in-vivo porcine liver, both modeled using corotated linear elastic materials.

The approach that we used for our non-rigid pipeline is very similar to the one presented in these papers except, of course, for the biomechanical model that we used. As we will see in sections 5.4, 5.5 and 5.6 we actually tested our pipeline with different organs and also different biomechanical models. But in all cases, the non-rigid process follows the same structure. We first start with an initial rigid registration performed with algorithm 2. The resulting transformation is then

applied to the entire biomechanical mesh. At this point, the biomechanical model should be fairly close to the desired shape except for the differences caused by the intra-operative deformation. Next, a procedure similar to the rigid ICP is started. From the point of view of the rendering (virtual) camera, the set of visible nodes from the biomechanical model's surface mesh is extracted and labeled as the source point cloud $\mathcal{S}$. This step can be done quite efficiently by filtering visible triangles using the z-buffer of the 3D rendering pipeline. If the z-buffer isn't accessible, a ray can be traced between each surface nodes and the rendering camera. If the ray does not intersect any triangles, the node is marked as visible and appended to the source cloud $\mathcal{S}$. For each node in $\mathcal{S}$, its closest neighbor in the target intra-operative point cloud $\mathcal{T}$ is found, ignoring outliers. The same procedure is done for each target point, which is, its closest neighbor in the source cloud. Unlike the rigid ICP algorithm which uses these neighborhoods to extract a rigid transformation, here they are used to impose displacements as the boundary conditions of the biomechanical model. However, instead of imposing the displacement of a source node directly to its closest target point, we smooth this displacement using the set of target points from which this source node is an immediate neighbor. This can be expressed mathematically as follows. Let $\mathcal{N}_s : \mathcal{S} \to \mathcal{T}$ be a function which returns the closest target point of any source points, and $\mathcal{N}_t : \mathcal{T} \to \mathcal{S}$ the equivalent for any target points. Let also $\boldsymbol{p} : \mathcal{S} \times \mathcal{T} \to \Omega$ be the current position of a node from the source or the target point cloud. We define the imposed displacement at the i^{th} source node with

$$\boldsymbol{u}_i = \alpha \left[\boldsymbol{p}(\mathcal{N}_s(i)) - \boldsymbol{p}(i)\right] + \frac{1}{\|\mathcal{V}(i)\|} \sum_{j \in \mathcal{V}(i)} (1-\alpha)[\boldsymbol{p}(j) - \boldsymbol{p}(i)] \tag{5.4}$$

with

$$\mathcal{V}(i) = \{j \in \mathcal{T} \mid i = \mathcal{N}_t(j)\}$$
$$\alpha : [0, 1]$$

and where $\mathcal{V}(i)$ is the set of target nodes for which i is their closest neighbor, and α is called the *blending factor* and determines the ratio between the attraction of the closest targets point of i against its projection to $\mathcal{N}_s(i)$. Figure 5.4 illustrates how the imposed projected displacement is blended to the attraction of five close target points. The blending factor can be used in cases of local minimum energy where the density of the point cloud around the nearest neighbor is different of the density around a source node.

When a FE method is used, the liver volume is discretized with a boundary fitting mesh composed of iso-parametric elements. In this case, the source nodes

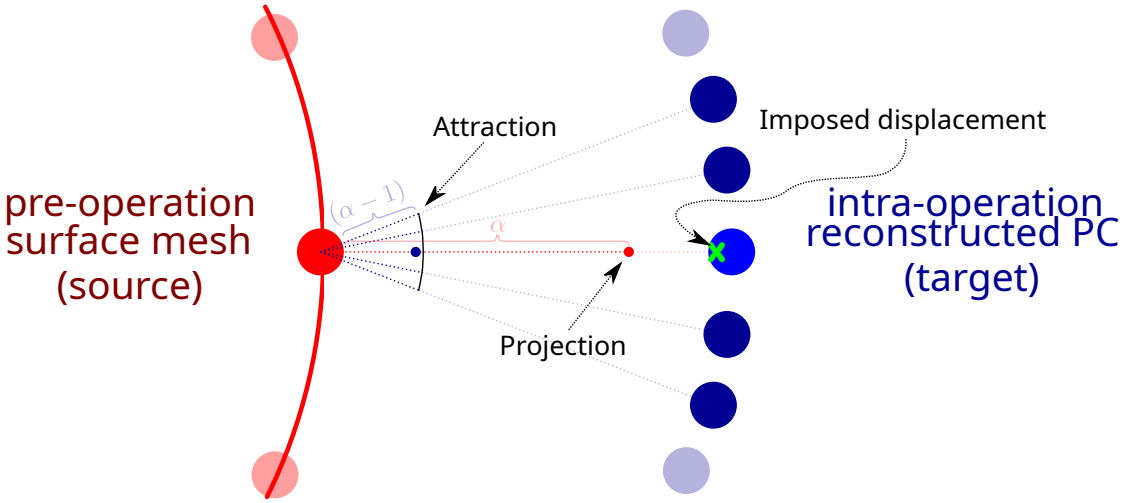


Figure 5.4: The imposed displacement is found by blending ($\alpha = 0.75$) the attraction of the five target neighbors against the projection to the nearest neighbor.

are simply the one located at the boundary of the mesh and the displacements are imposed directly on them. When using a non-fitting boundary method such as the meshless methods or our proposed Immersed boundary (IB) approach, the source nodes are taken from a surface mesh embedded inside the simulated domain. In this case, the location of source nodes are interpolated at each step of the simulation from their embedding elements and special care has to be taken for the imposition of the displacement. In any cases, we chose to impose displacements using a penalty method.

The method we have chosen consists of increasing progressively the stiffness of the penalty term until the relative displacements of the biomechanical model are almost unchanged between two non-rigid ICP iterations. When the relative displacements between two iterations are sufficiently close, the non-rigid ICP iterations are stopped and the method is ready to compute the next intra-operative target point cloud. The reasoning here is that even with a good outliers pruning, the target point cloud might still contain noisy data. Similarly, closest neighbors of the source nodes might be found within a local minimum of the target point cloud. A strict displacement imposition such as with a Lagrange multiplier method or with a penalty method using a large stiffness would incorrectly try to match the target nodes that are not making any physical sense. Solving the system of equations would then become difficult, or even impossible as the linear solver would diverge. The same thing happens if the target neighbors are simply too far away from the source nodes. In this case, it is the Newton–Raphson solver that would diverge as the deformed solution would be too different from the current

state of the model. We usually increase the stiffness of the penalty following an exponential curve such that as the source almost matches the target point cloud, the displacement imposition gets stricter. The complete procedure is summarized in Algorithm 3.

Algorithm 3 Non-rigid ICP registration

$\mathcal{S}$: Source surface nodes
$\mathcal{T}$: Target points cloud
$\boldsymbol{u}$: Current displacement vector of the biomechanical model
k_0: Starting penalty stiffness
k_1: Maximum penalty stiffness
n_p: Maximum number of penalty increments
n_i: Maximum number of ICP iterations
n_n: Maximum number of NR iterations
ϵ: Convergence criterion
α: Blending factor

procedure NONRIGID_ICP($\mathcal{S}, \mathcal{T}, \boldsymbol{u}, k_0, k_1, n_p, n_i, n_n, \epsilon, \alpha$)
    $\boldsymbol{T} \leftarrow \boldsymbol{T}_{init}$ ▷ Using eq. (5.3)
    $\mathcal{S} \leftarrow \boldsymbol{T}(\mathcal{S})$
    $\boldsymbol{u}^0 \leftarrow \boldsymbol{u}$
    $\boldsymbol{u}^i \leftarrow \boldsymbol{u}$
    while $i \leq n_p$ **do**
        $k \leftarrow k_0 * 2^{\frac{i*\log_2(k_1-k_0)}{n_p}}$ ▷ Penalty stiffness
        while $j \leq n_i$ **do**
            $\mathcal{S}_v \leftarrow visible\_from\_camera(\mathcal{S})$
            $\mathcal{N}_s,\ \mathcal{N}_t \leftarrow find\_closest(\mathcal{S}_v, \mathcal{T}),\ find\_closest(\mathcal{T}, \mathcal{S}_v)$ ▷ Using k-d tree
            $\mathcal{N}_s,\ \mathcal{N}_t \leftarrow remove\_outliers(\mathcal{N}_s),\ remove\_outliers(\mathcal{N}_t)$
            $\boldsymbol{u}_d \leftarrow imposed\_displacements(\mathcal{N}_s,\ \mathcal{N}_t, \alpha)$ ▷ Using eq. (5.4)
            $\boldsymbol{K}_p \leftarrow \boldsymbol{I} * k$
            $\delta\boldsymbol{u} \leftarrow 0$
            while $n \leq n_n$ **do**
                $\boldsymbol{f}_\mathrm{p} \leftarrow k \cdot (\boldsymbol{u} + \delta\boldsymbol{u})$
                $\boldsymbol{f}_\mathrm{e} \leftarrow \boldsymbol{R}(\boldsymbol{u} + \delta\boldsymbol{u})$ ▷ Using eq. (2.18)
                $\boldsymbol{K}_\mathrm{e} \leftarrow \boldsymbol{K}(\boldsymbol{u} + \delta\boldsymbol{u})$ ▷ Using eq. (2.26)
                solve $[(\boldsymbol{K}_\mathrm{e} + \boldsymbol{K}_p) \cdot \delta\boldsymbol{u} = \boldsymbol{f}_\mathrm{e} + \boldsymbol{f}_\mathrm{p}]$
                if $\frac{\|\boldsymbol{f}_\mathrm{e}+\boldsymbol{f}_\mathrm{p}\|}{\|\boldsymbol{f}_\mathrm{e}^0+\boldsymbol{f}_\mathrm{p}^0\|} \leq \epsilon$ **then** ▷ NR converged
                    break
                $n \leftarrow n + 1$
            $\boldsymbol{u} \leftarrow \boldsymbol{u} + \delta\boldsymbol{u}$
            if $\frac{\boldsymbol{u}^j}{\boldsymbol{u}^0} \leq \epsilon$ **then** ▷ ICP converged
                $j \leftarrow j + 1$
        if $\frac{\boldsymbol{u}^i}{\boldsymbol{u}^{i-1}} \leq \epsilon$ **then** ▷ Plateau reached
            $i \leftarrow i + 1$

5.4 Experiments performed on in-vivo porcine livers

We evaluated our proposed pipeline with real clinical scenarios using the same in-vivo porcine livers as for the experiments of the previous chapter. As mentioned before, the liver weighted between 59 and 70 kilograms and the CT images were acquired before and after pneumoperitoneum set at 13 mmHg. Semi-automatic segmentation was done on both the liver's parenchyma and blood vessels for all trials using ITK-SNAP [Yushkevich et al. (2006)] and 3D Slicer [Kikinis, Pieper, and Vosburgh (2014)]. Surgical navigation was conducted using an optical tracking camera and the hand-eye calibration was computed as described Teatini et al. (2019). After the pneumoperitoneum, a stereo reconstruction of the liver was computed based on Wang et al. (2018) using a laparoscope camera. Figure 5.5 shows the side and bottom views of these reconstructions.

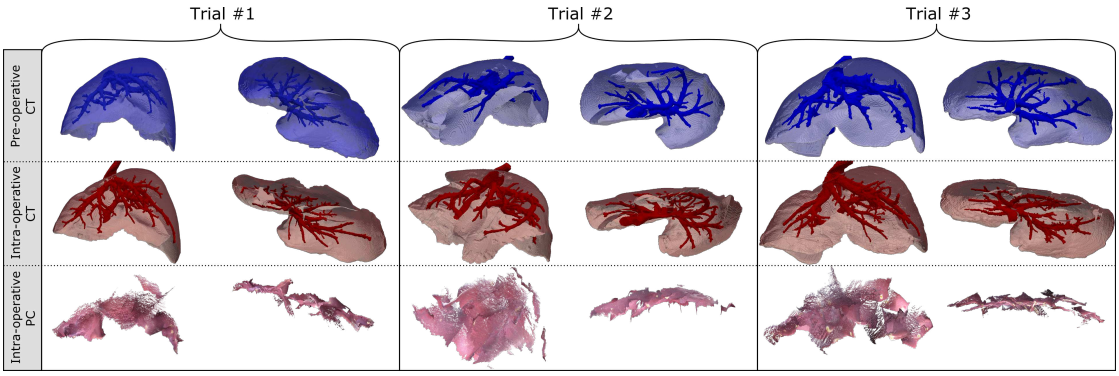


Figure 5.5: Pre-pneumoperitoneum CT reconstruction and post-pneumoperitoneum CT and PC reconstructions for our three porcine trials.

Only one stereo Points Cloud (PC) reconstruction was performed for our experiments since the objective was to validate the accuracy of the biomechanical model for the initial deformable registration of the porcine liver before and after the pneumoperitoneum. This first registration process is usually the one that encounters the most deformation since subsequent reconstructions benefit from temporal proximity between point clouds.

The biomechanical model was discretized using a background grid made of regular hexahedrons and filled with a neohookean material. The hyperelastic stiffness matrix update at each Newton–Raphson (NR) iteration of the algorithm 3 was done using our Weighted Cell (WC) method. We annotated 15 vessels bifurcation intersections in both the pre-operative and intra-operative CT reconstructions for each trial. We then performed both rigid and non-rigid ICP registrations.

Table 5.1 contains Euclidian target registration errors at the vessels bifurcation intersections.

Vessel Bifurcation Evaluation						
		mean	median	min	max	std
Trial 1	Rigid registration	19.80	17.77	5.17	55.31	14.38
	Non-rigid registration	22.66	22.79	6.53	34.84	7.24
Trial 2	Rigid registration	28.47	21.65	6.57	67.29	16.12
	Non-rigid registration	27.35	26.97	9.14	57.40	14.61
Trial 3	Rigid	17.27	12.04	4.33	60.67	14.88
	Non-rigid registration	26.56	24.75	15.96	43.32	7.40

Table 5.1: Results of the rigid and non-rigid registrations using the vessel bifurcation target registration errors, in [mm].

At first glance, these results are quite surprising. Using the mean target registration error as the reference, the rigid registration appears to outperform the non-rigid solution. Maximum error and standard deviation values are, however, much lower for the non-rigid method. To get a better view of what is actually causing this surprising behavior, figure 5.6 illustrates the visual output obtained from these two registration methods.

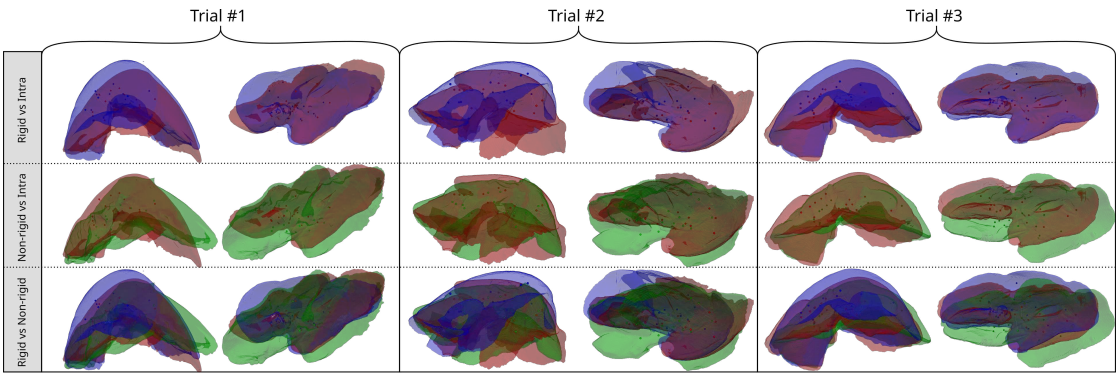


Figure 5.6: Rigid (blue) and non-rigid (green) registrations towards the reconstructed PC against the intra-operative CT reconstruction (red) for our three porcine trials.

Here we see that both the rigid and non-rigid registrations are not performing very well. While the use of a biomechanical model does induce a deformation that resembles the one from the intra-operation reconstruction, it is clear that the reconstructed PC is missing too much information. Looking again at figure 5.6,

the back of the liver is completely unconstrained, while in real life it would be restrained by the portal vein and the back of the pork. Hence, the biomechanical model does find a minimal energy solution that matches the reconstructed PC, but due to the lack of boundary conditions on the back of the liver, the solution is not in line with the real deformation undergone by the liver. Had the reconstructed PC been provided with more information, the non-rigid approach would have generated better results.

To validate this hypothesis, we decided to observe the solution produced by the model when using the non-rigid ICP method against the full intra-operative surface reconstruction instead of the incomplete reconstructed PC. We were thereby introducing a large number of boundary conditions. We were also ignoring the error from a misalignment of the reconstructed PC against the intra-operative CT reconstruction over which we had no control (recall that in a real-life scenario, intra-operative reconstruction is usually not accessible during the intervention). In the end, this approach allowed us to isolate the real error induced by the biomechanical model.

Vessel Bifurcation Evaluation					
	mean	median	min	max	std
Trial 1	10.90	9.85	3.22	29.50	6.46
Trial 2	18.80	17.70	7.75	35.70	7.40
Trial 3	7.75	7.38	2.20	14.40	3.31

Table 5.2: Vessel bifurcation target registration errors of the non-rigid registrations using the whole intra-operative surface reconstruction as the target, in [mm].

Table 5.2 contains the target registration errors between vessel bifurcations when using the whole intra-operative surface reconstruction as the target PC, i.e., a scenario where a perfect reconstruction of the surface would be available during the intervention. As expected, the non-rigid registration achieved much better results and outperformed the rigid process. Surprisingly, some vessel bifurcations still remain quite far off from their intra-operative counterparts. For example, in trial #2, one bifurcation was found at about 3.5 centimeters from the real solution. Moreover, looking at figures 5.7, 5.8 and 5.9, it is clear that the non-rigid ICP procedure successfully completed its task: the surfaces of both the biomechanical model and the intra-operative reconstruction match almost perfectly. However, internal blood vessels do not match. Note that this experiment was performed using a mesh size located at the convergence pivot point of the mesh refinements. In other world, we successively refined the background grid and reran the experiment

until the difference between the solutions generated by a mesh with element sizes of h and a finer mesh with element sizes of $h/2$ was negligible.

It is important to mention that we managed to produce similar results using a standard FE method with tetrahedral meshes. This suggests that these errors are not derived from the method itself, nor from the quality of the discretization, but instead from the heterogeneous behavior of the liver largely due to internal blood vessels. To overcome this, one would have to accurately model the different substructures of the liver. We note that this is in contradiction with Plantefève, Peterlik, and Cotin (2017) where authors claimed that the influence of the liver's heterogeneity on the registration process was marginal. Unfortunately, we were not able to find other studies to further validate our hypothesis.

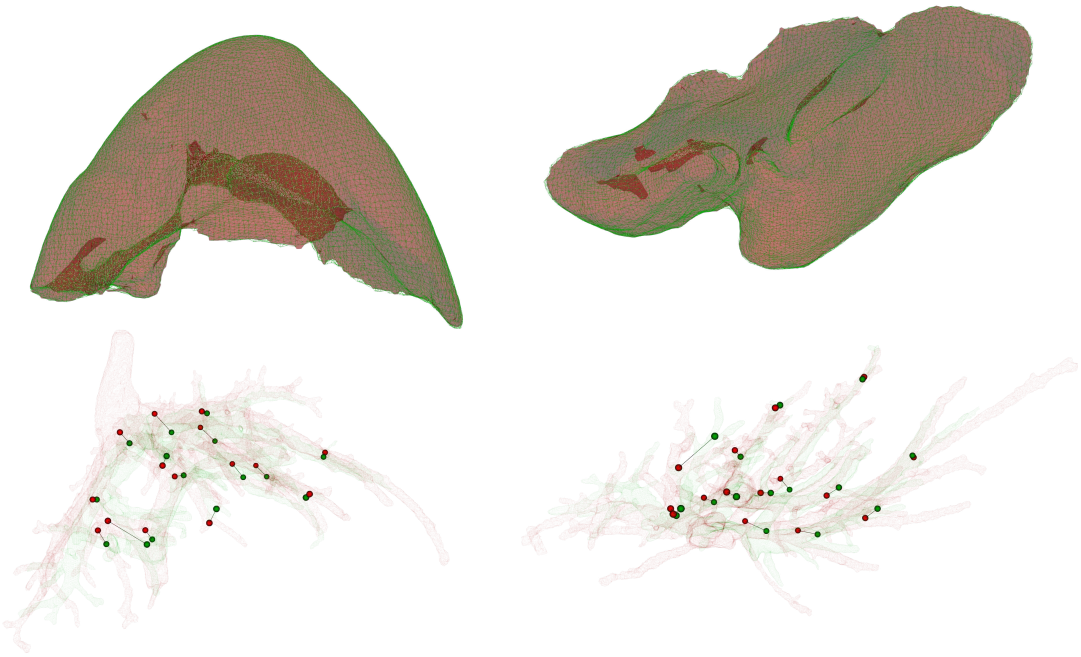


Figure 5.7: Non-rigid (green) registration towards the intra-operative CT reconstruction (red) for our trial #1.

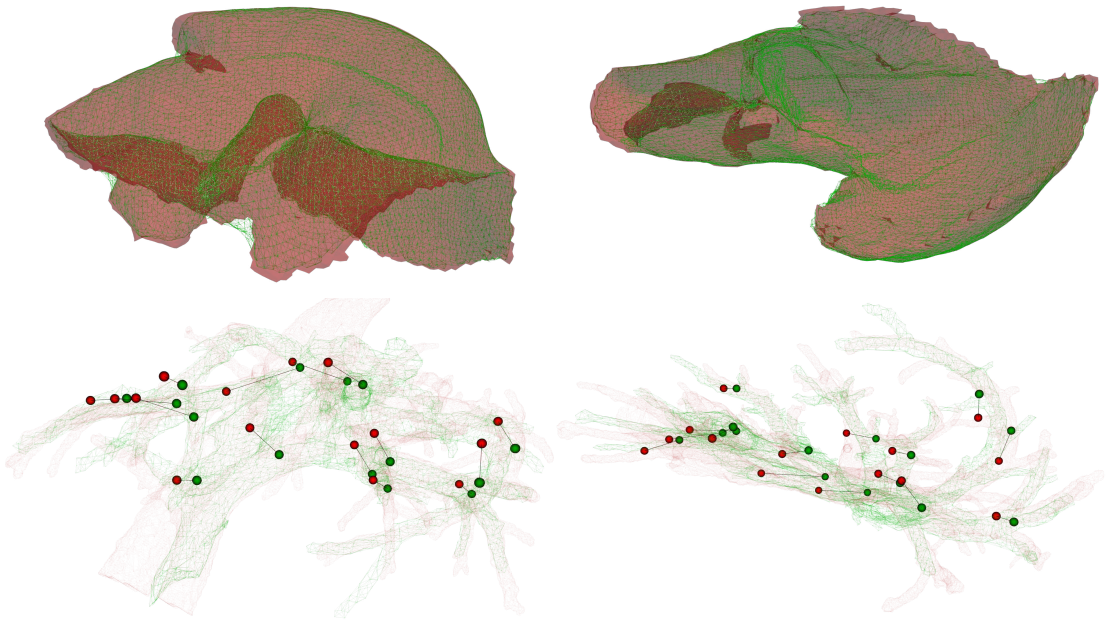


Figure 5.8: Non-rigid (green) registration towards the intra-operative CT reconstruction (red) for our trial #2.

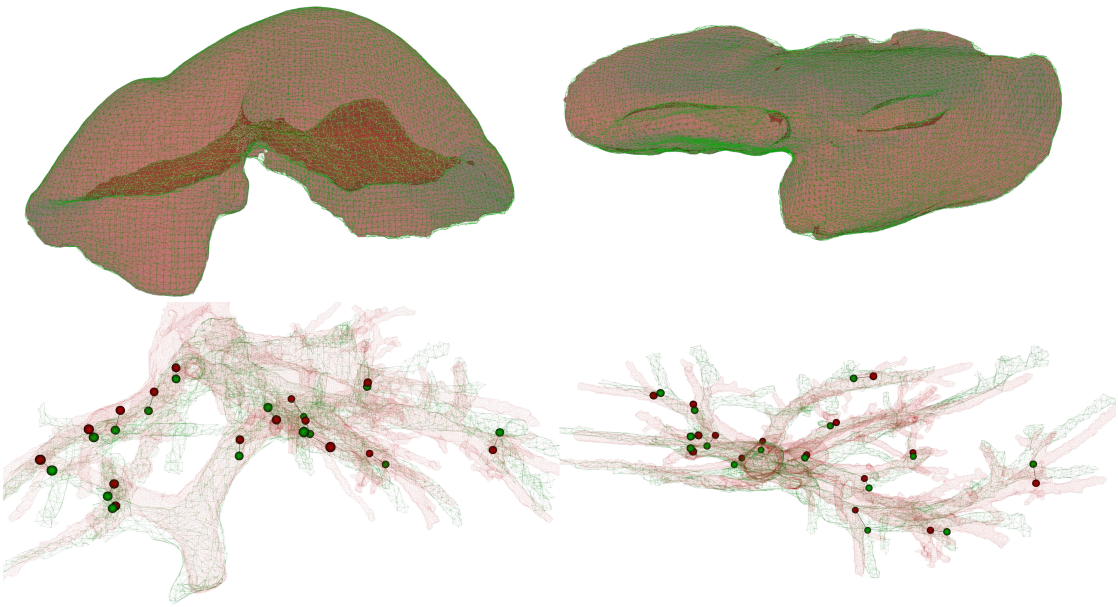


Figure 5.9: Non-rigid (green) registration towards the intra-operative CT reconstruction (red) for our trial #3.

5.5 Experiments performed on an ex-vivo human liver

We also conducted additional experiments using the same human liver that we used in the previous chapter to test our WC method. As already mentioned, the human liver was ex-vivo and, therefore, was not subject to pneumoperitoneum. It was also not possible to inject the liver with a radio-contrast; hence the vessel bifurcations could not be used as validation points. Instead, before any medical images were produced, we placed ten very small lead balls at different locations inside the liver's parenchyma. Since these balls would normally retain their position following a deformation, they were therefore used as the physical *markers* required to validate our method. The liver was placed on a flat table as depicted in figure 5.10a, and an initial CT scan which we labeled as our "pre-operative" data was performed. Two other scans were then taken after generating two distinct deformations (figures 5.10b and 5.10c). For each of those two deformations, a partial reconstruction of the liver was reconstructed using an RGB-D camera, resulting in a three-dimension point cloud.

(a) Initial "undeformed"

(b) Deformation #1

(c) Deformation #2

Figure 5.10: Pictures of the liver made before and after the two deformations

For this experiment, we denoted the first deformation of the liver as trial #1 and the second deformation as trial #2, even though both trials were conducted from

the same undeformed liver. We also labeled the reconstructed partial 3D point clouds and the CT scan reconstructions as our "intra-operative" data. Figure 5.11 shows both the side and bottom views of these reconstructions.

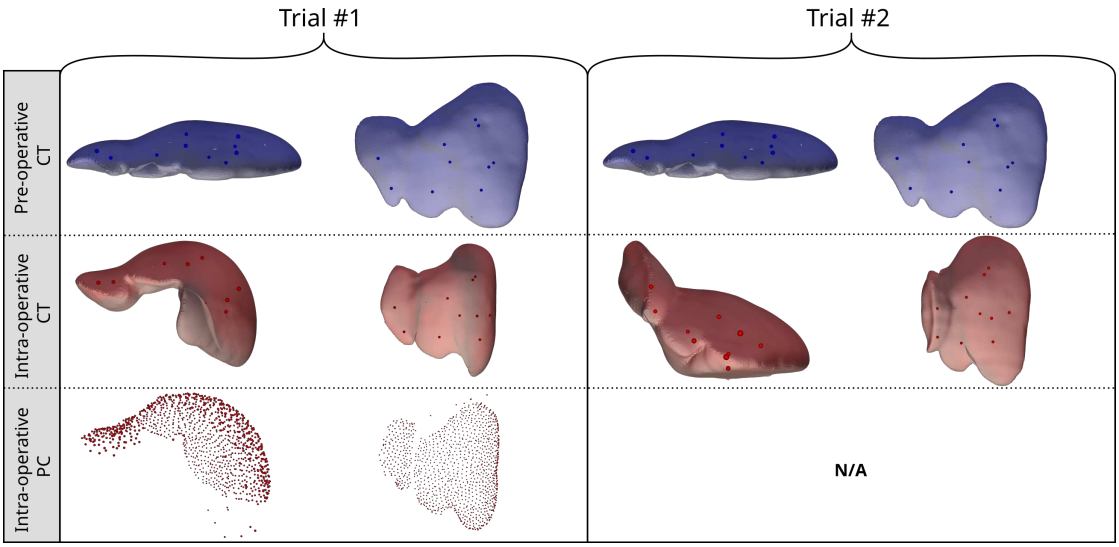


Figure 5.11: Pre-deformation CT reconstruction and post-deformation CT and PC reconstructions for our human trials.

As per the porcine experiments, the biomechanical model was discretized using a background grid made of regular hexahedrons and filled with a Neohookean material. The Young's modulus and Poisson's ratio were set to 5000 [MPa] and 0.499, respectively. The hyperelastic stiffness matrix update at each Newton–Raphson (NR) iteration of the algorithm 3 was done using our Weighted Cell (WC) method. Table 5.3 provides the Euclidian target registration errors at the marker positions.

Markers Evaluation						
		mean	median	min	max	std
Trial 1	Rigid registration	21.63	16.47	8.10	60.17	17.23
	Non-rigid registration	6.56	6.37	4.00	9.35	1.84
Trial 2	Rigid registration	-	-	-	-	-
	Non-rigid registration	-	-	-	-	-

Table 5.3: Results of the rigid and non-rigid registrations using the marker target registration errors, in [mm].

This time, we clearly see that the non-rigid registration generates a more accurate solution. The maximum error found at a marker position is less than 1 [cm] and

the standard deviation stays within a 2 [mm] margin. To better illustrate the advantage of the non-rigid approach, figure 5.12 provides visual output obtained from the two registration methods.

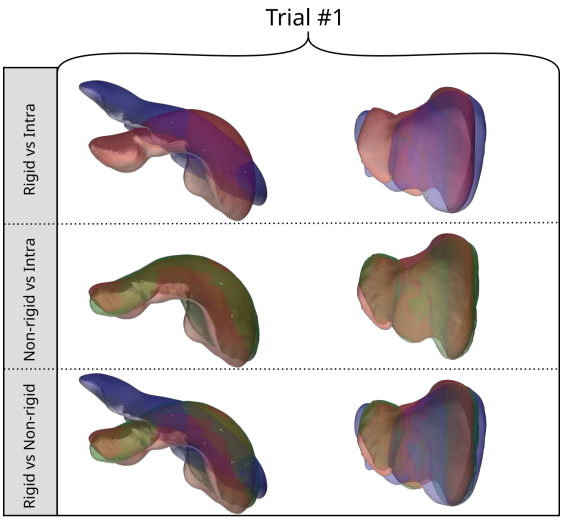


Figure 5.12: Rigid (blue) and non-rigid (green) registrations towards the reconstructed PC against the intra-operative CT reconstruction (red).

From these images, we see that the shape of the top surface of the biomechanical model matches accurately the top surface of the intra-operative ground truth. This suggests that the quality of the reconstructed point cloud from the RGB-D camera was better than that of the reconstructed point cloud from the stereo reconstructions produced in our three porcine trials. We also notice that there is still a maximum error of almost 1 [cm], which again, is most probably due to the partial surface reconstruction. In order to remove this component of the error, we followed an approach similar to the one used in the porcine experiment. Using the whole intra-operative surface reconstruction as the target point cloud, we thereby introduced a large amount of boundary information in the model, hence overcoming the error relative to the use of a partial surface reconstruction. We were also preventing possible errors associated with a misalignment of the reconstructed PC against the intra-operative CT reconstruction thus allowing us to identify the portion of the error that is truly attributable to the biomechanical model. The target registration errors are provided in table 5.4.

These results confirm once more the net advantage of the non-rigid registration algorithm. This is further illustrated in figures 5.13 and 5.14 where we clearly see that the non-rigid ICP procedure has again successfully achieved its task: the surfaces of both the biomechanical model and the intra-operative reconstruction

Markers Evaluation					
	mean	median	min	max	std
Trial 1	2.22	2.19	0.34	4.25	1.09
Trial 2	8.35	8.43	3.07	12.54	2.90

Table 5.4: Marker target registration errors of the non-rigid registrations using the whole intra-operative surface reconstruction as the target, in [mm].

match almost perfectly. However, this is not the case for some of the internal markers. As per the results of the porcine trials and based on our observations, these errors do not result from the method itself nor from the quality of the discretization but are a direct consequence of the heterogeneous behavior of the liver.

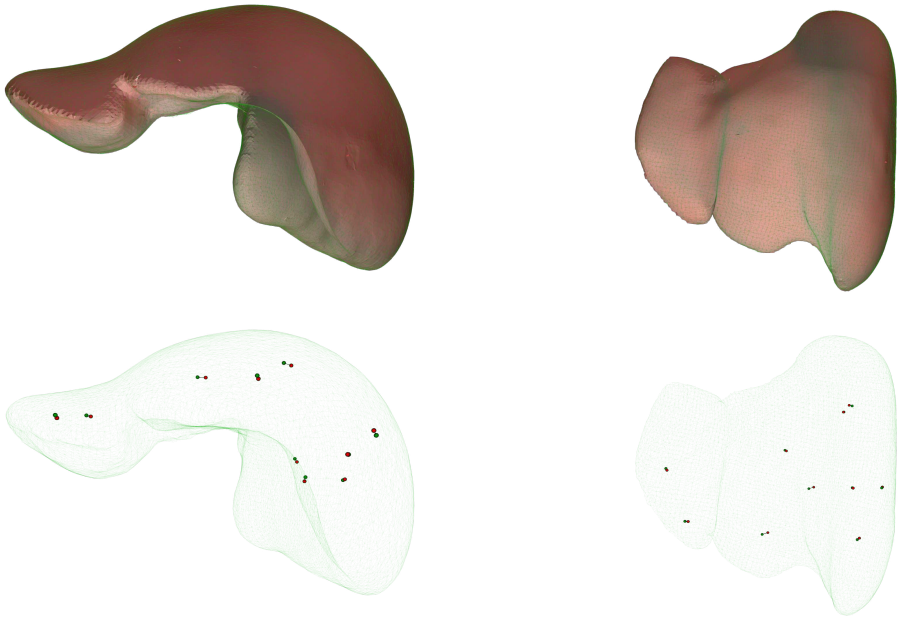


Figure 5.13: Non-rigid (green) registration towards the intra-operative CT reconstruction (red) for our trial #1.

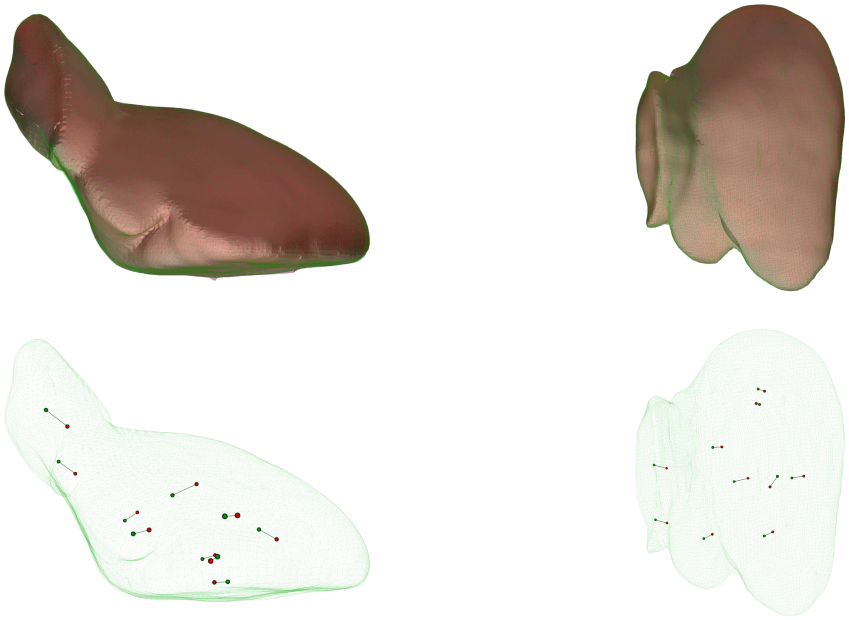


Figure 5.14: Non-rigid (green) registration towards the intra-operative CT reconstruction (red) for our trial #2.

5.6 Experiments mixing IBM and machine learning techniques

The accuracy of the biomechanical model used in our non-rigid registration algorithm comes with a non-negligible price. The non-linearity of the model requires that a series of Newton–Raphson iterations be ran at each step of the ICP routine, which implies that a complete linear system must be solved numerous times between each reconstructed intra-operative surface PC. This obviously impacts the speed time of our pipeline. To address this issue, one avenue currently considered is to substitute certain parts of the registration process by Machine Learning (ML) techniques.

Morooka et al. (2008) was one of the first to propose a ML approach to learn the relationship between external forces applied to a liver and the resulting deformations. Using a Principle Components analysis (PCA) approach to encode deformation modes, they trained a complete Neural Network (NN) based on synthetic data generated by a non-linear FE method. They showed that their method can generate an accurate deformation from an input vector of forces in about 0.3 [ms]. Following the same concept, Tonutti, Gras, and Yang (2017) was able to predict deformations of brain tumors in real time from an applied load using FE simulations to generate synthetic training data.

We constructed an experimental version of our pipeline using the well-known U-Net approach. To generate the synthetic training data, imposed displacements were used as input instead of the external force framework of Tonutti, Gras, and Yang (2017). The following presents an overview of the setup and findings. The complete results have been documented in [Pfeiffer et al. (2019); Brunet, Mendizabal, et al. (2019); Mendizabal et al. (2020)].

U-Net architecture

As the starting point of the ML approach, we note that the 3D point cloud generated by the intra-operative reconstruction provides sparse and partial views of the surface of the organ. In its simplest form, our problem can therefore be expressed as finding the function f that produces the best estimation of the internal deformation of the organ from this point cloud. The function f can be obtained by minimizing the expected error over a training set $\{(\boldsymbol{u}_s^n, \boldsymbol{u}_v^n)\}_{n=1}^N$ of N samples:

$$\min_\theta \frac{1}{N} \sum_{n=1}^{N} \|f(\boldsymbol{u}_s^n) - \boldsymbol{u}_v^n\|_2^2. \tag{5.5}$$

where θ is the set of parameters of the network f, and $\boldsymbol{u}_s$ and $\boldsymbol{u}_v$ are the surface input and the volumetric output displacement fields of each sample.

To characterize f, the U-Net approach of Ronneberger, Fischer, and Brox (2015) was used. This a modified fully convolutional network initially built for medical image segmentation applications. As depicted in figure 5.15, the network is similar to an auto-encoder, with an encoding path to transform the input space into a low-dimensional representation, and a decoding path to expand it back to the original size. Additional skip connections transfer detailed information along the matching levels from the encoding path to the decoding path.

The encoding path consists of k sequences of two padded $3 \times 3 \times 3$ convolutions ($k = 4$ in Ronneberger, Fischer, and Brox (2015)) and a $2 \times 2 \times 2$ max pooling operation (see figure 5.15). At each step, each feature map doubles the number of channels and halves the spatial dimensions. In the bottom part, there are two extra $3 \times 3 \times 3$ convolutional layers leading to a 1024-dimensional array. In a symmetric manner, the decoding path consists of k sequences of an up-sampling $2 \times 2 \times 2$ transposed convolution followed by two padded $3 \times 3 \times 3$ convolutions. The features of the encoding path at the same stage are cropped and concatenated to the up-sampled feature maps. At each step of the decoding path, each feature map halves the number of channels and doubles the spatial dimensions. There is a

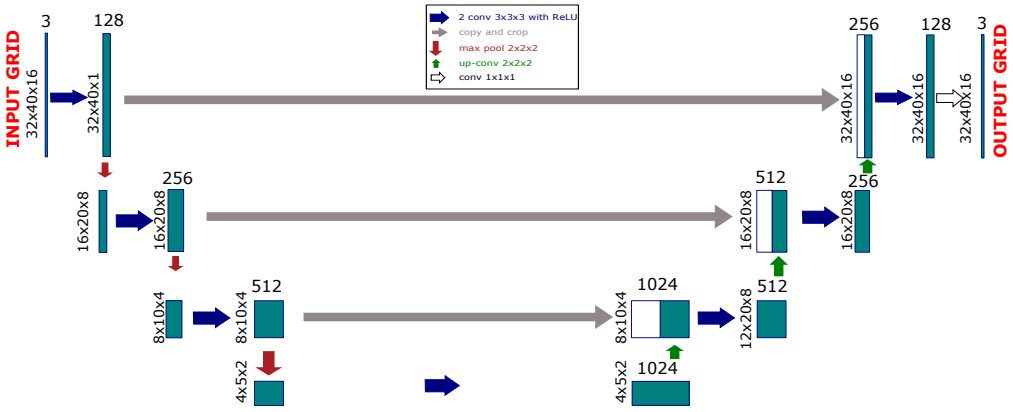


Figure 5.15: U-Net architecture with 3 steps and 128 channels in the first layer of a padded input grid of size $32 \times 40 \times 16 \times 3$.

final $1 \times 1 \times 1$ convolutional layer to transform the last feature map to the desired number of channels of the output (3 channels in our case).

A key observation here is that both the up-sampling and down-sampling processes imply the use of a grid-like structure for encoding the displacement information. On the other hand, our Immersed boundary approaches are based on a sparse grid made of rectangular hexahedral elements. Intuitively, this common property made the WC method a natural candidate to be used as network training technique for the U-Net pipeline.

Data generation and training using the WC method

To train the network, many samples consisting of a volumetric output displacement field $\boldsymbol{u_v}$ are generated from a nearly random input displacement $\boldsymbol{u_s}$. In other words, as many pairs as possible of *partial input surface displacement* $\rightarrow$ *complete output volumetric displacement* are generated. These pairs are then fed to the network for it to learn the complex relationship between inputs and outputs. Clearly, the accuracy of the training data will impact the reliability of the network when used to predict an unknown input. Hence, using a biomechanical model to generate the training data will greatly enhance the efficiency of the network. In addition, the model must also by computationally efficient. Even if the training of the network could theoretically run forever, in practice only a couple of hours would be allowed. For example, a patient-specific network could be trained between the acquisition of initial pre-operative medical image and the surgery procedure.

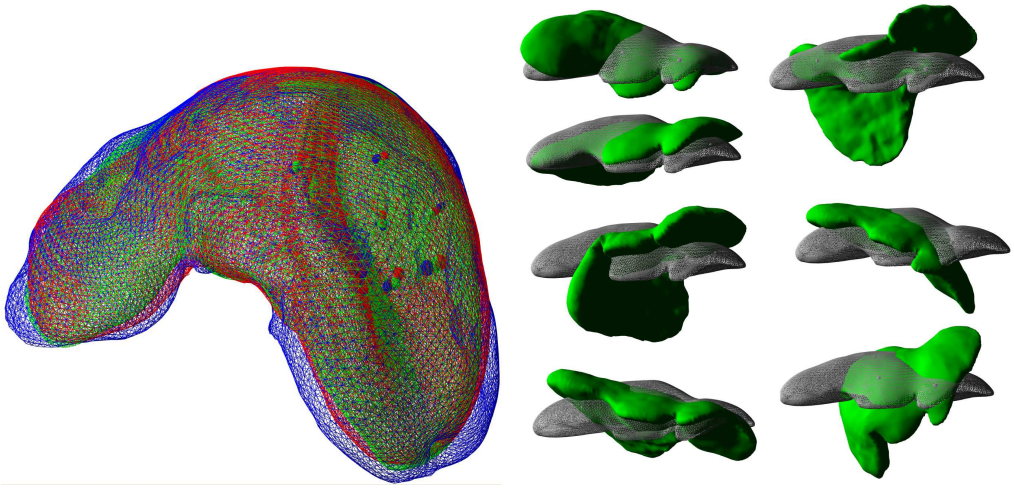
Using the biomechanical model, a data set of pairs $(\mathbf{u_s}, \mathbf{u_v})$ is therefore generated where the surface displacement corresponds to the mapped point cloud. We assume that, at most, half of the organ surface is visible from the camera. Hence, we uniformly sample 100 points on the visible surface mesh. Hundred simultaneous forces are then applied to this virtual point cloud with random amplitudes and directions. In order to generate a series of surface deformations (see figure 5.16b), these random forces are applied to their enclosing grid nodes using their barycentric coordinates. Patient-specific parameterization of the biomechanical model is not required here since for homogeneous materials, the relation between the surface and the volumetric displacements is independent of the stiffness of the object Karol Miller and J. Lu (2013), and only depends on the Poisson's ratio. As a result, a SVK material was used for the experiments given its relative simplicity, computational efficiency and robustness (see section 2.3.1 for more detail on this material). The Poisson's ratio was set to 0.49.

Results on ex-vivo human liver

The results obtained from our U-Net pipeline experiments have also been generated using an *ex vivo* human liver data set for which the surface data was obtained with an RGB-D camera and the ground truth data was acquired at different stages of the deformation using a CT scan [Brunet, Mendizabal, et al. (2019)]. Markers were embedded in the liver to compute Target Registration Errors (TRE).

Experiments using controlled synthetic deformations To validate the accuracy of the network, we first compared the predictions of the displacement field with those of the IB simulations generated by our WC method described in section 4. We generated $1,000$ samples and used $N = 800$ samples to train the network by minimizing equation (5.5). The minimization was performed using the Adam optimizer: a stochastic gradient descent procedure with parameter-wise adjusted learning rates. The remaining 200 samples were then used for validation. For each sample, the average TRE was computed over 10 virtual markers. Overall, the average TRE obtained over the entire validation set was $\overline{\text{TRE}} = 1.96 \pm 3.46$ mm.

Results in an augmented reality environment With a PyTorch implementation of the U-Net running on a GeForce 1080 Ti, the network was capable of predicting the volumetric deformation of the liver in only $3\,ms$. A prediction



(a) (b)

Figure 5.16: (a) U-Mesh prediction in green, co-rotational FEM based registration in blue and ground truth in red. A mean TRE of 2.92 mm is achieved over the 10 markers in about 3ms. (b) Samples of the generated deformations using our FE method. The rest shape is shown in gray.

can thus be applied each time the RGB-D camera generates a point cloud. Before computing the displacement field, the point cloud needs to be cropped to the portion of the surgical image that contains the liver. This is done by segmenting the associated color image, similar to what was done in Petit, Lippiello, and Siciliano (2015). The RGB-D point cloud is then interpolated onto the grid to obtain the per-node displacements on the surface (i.e. $\mathbf{u_s}$). Given this input, the network predicts the volumetric deformation, and the next point cloud can be processed. When compared to our *ex vivo* ground truth, the average TRE at the 10 markers was of 7.5 mm with a maximal value of 10.5 mm. Our non-rigid pipeline that is based on the same WC method that was used to train our model produced nearly the same error. This was obviously expected. But the pipeline generated a much longer computation time (a total of 1550ms) even when using a very efficient linear solver (Pardiso[1]).

Limitations of our U-Net network

Despite its incredible computational speed, the U-Net architecture has a non-negligible limitation. The accuracy of the U-Net prediction of a deformed object

[1] https://pardiso-project.org

from an unknown input is directly influenced by the amount of training the network has received for this type of deformation. In order words, the network is *a priori* very efficient to interpolate a solution between deformation modes it was trained with, but remains quite inaccurate when we need to extrapolate a prediction in a solution space it has never been encountered before. The generation of a large and rich enough panorama of deformation modes required to adequately train the network is not trivial. At this stage of the research, it is still impossible to determine if the learning process of the network can be sufficiently enhanced to predict intra-operative deformations that are accurate enough when partial surface reconstruction is used.

5.7 Discussions

The experiments performed on our two registration algorithms using real clinical scenarios have clearly demonstrated the net superiority of the non-rigid approach over its rigid counterpart. But we also observed that the organ surface reconstruction process occurring during the surgery will have a major impact on the accuracy of the deformations produced by the algorithm. In fact, for both porcine and human liver experiments, we saw that a simple partial surface reconstruction of the organ was not enough to accurately predict the deformation *inside* the organ. We also observed that even if a complete and accurate reconstruction of the whole surface is imposed for the displacement calculation, some portions within the liver may not be accurately simulated.

Obviously, this may be viewed as a major drawback of the overall non-rigid pipeline as it would simply be impossible for a surgeon to rely on the predicted location of a tumor or an important vessel if the model is not capable of limiting the maximum error to a few millimeters. However, it is important to put these observations in perspective. As aforementioned in the introduction of this thesis, without any computed-aided numerical registration, a surgeon has to mentally find the correspondences between pre-operative medical images and the partial view of the organ's surface during the surgery. One of the first objectives of a dynamic registration process is to assist the surgeon in this task and provide an approximate alignment of the three-dimensional pre-operative reconstruction of the liver onto the augmented reality images. A process that would not account for the intra-operative deformation undergone by the liver would result in highly different superposed shapes which would not be very useful to the medical staff. Hence, the ability of our non-rigid pipeline to account for the organ's deformation presents a real advantage.

Moreover, we saw that the biomechanical model used in our pipeline can be built with minimal effort by simply placing a background computational grid over the three-dimensional surface reconstruction of the patient's liver, resulting in a quasi-automatic workflow for the medical staff. Therefore, even if the resulting position for some internal structures of the overlaid deformed model is not precise down to the millimeter, the effort currently required by the surgeon to mentally register different medical modalities altogether would be considerably reduced.

Nonetheless, our non-rigid registration process could be further refined and its accuracy improved by simply injecting additional information into the model. This could be done in two different ways. Firstly, the model could be extended such that the behavior of the liver's internal structures are correctly simulated. Although this contradicts the conclusions of Plantefève, Peterlik, and Cotin (2017), we believe that incorporating heterogeneity into the model in the form of different stiffness values and anisotropy around the vessels and the Gibson capsule could considerably improve the solution inside the liver. Since the traditional FE method requires a fine mesh around these structures, our IB approach would obviously constitute an interesting alternative here. Secondly, a small amount of local knowledge about the deformed state inside the liver could be injected into the model during surgery. This could considerably improve the solution if this additional information is provided for a region close to the region of interest, for example a tumor or a vessel. Since Ultrasound (US) probes are usually accessible in most operation rooms, some US images could possibly be registered and added as additional constraints to the model.

Finally, the issue relative to very partial surface reconstruction could be overcome by introducing additional boundary conditions such as the organs surrounding the liver and soft tissues connected to it. These boundary conditions could be estimated prior to the surgery from the pre-operative images. During the surgery, these estimations could be automatically refined using observations from the camera or US images. Following the approach used in Nikolaev, Peterlik, and Cotin (2018), the refinement through intra-operative observations could be implemented using a reduced-order unscented Kalman filtering approach [Moireau and Chapelle (2011)].

Chapter

6

CONCLUSION

In this thesis, we addressed the problem of soft-tissue simulations in the context of computer-aided liver surgery. More specifically, we were interested in the development of an efficient and automated biomechanical model that incorporates *physical* characteristics of the patient's liver to enhance the registration process between pre- and intra-operative medical images. Our development work was articulated around four basic research guidelines. Most importantly, the model had to be capable of producing an accurate and patient-specific three-dimension virtual representation of a liver's deformations for the medical team. The model also had to be able to generate stable solutions within a timeframe that meets the requirements of a real-time environment. Finally, the model had to be designed to minimize the development effort and ensure a minimum involvement from the medical staff.

We began our work with a thorough review of the basic theoretical hyperelasticity principles and the well-documented finite element method which is used in a wide range of biomechanics simulations. We quickly realized that the implementation of a discretization scheme outside of the traditional isoparametric approach was not possible with most of the software tools currently available. In fact, these tools were simply not adequate for our application which required that the intra-operative data be dynamically injected into the simulation process. A complete set of numerical algorithms was therefore derived and implemented. These have been made available in an advanced simulation software library [1] which turned out to be a valuable asset for the entire research team and the stepping stone for the

[1] https://github.com/jnbrunet/caribou

development of new meshless and immersed-boundary discretization approaches.

The Point-Based Animation (PBA) model is the first of the two meshless approaches that we investigated. This model is gaining a lot of popularity in the research community as it is physically driven and aimed at interactive simulation applications. Unfortunately, while the model is built on a solid continuum mechanic basis, we saw that it rarely produces accurate solutions due to the approximations used for the shape functions and the integration scheme. To overcome these inaccuracies, the MTLED approach documented in Horton et al. (2010) was then considered. This allowed us to implement our Meshless Approximation Mesh-Based Integration (MLAMBI) method that relies on the MLS model which we have enriched by simply adding monomial basis functions. Instead of doing a nodal integration based on an SPH-based volume estimation, we used a background grid of regular hexahedral elements. We have also proposed a corotated approach to efficiently simulate objects undergoing large rotations but moderate deformations. However, we saw that while we can adjust the approximation space by avoiding particles outside of the simulated domain, the proposed method would still suffer from inaccuracies resulting from the integration over the elements that are cut by the boundary.

The issue related to the cut elements took us to a different category of numerical models: the Immersed boundary (IB) methods. Contrary to the meshless approach, IB methods reuse the isoparametric paradigm inherent to the standard FE method. We presented two IB implementations. The first one, the *Finite Cell (FC) method*, recursively adds integration points near the cut interface inside the background elements. The second one, the *Weighted Cell (WC) method*, uses the added integration points of the FC method to adjust the weights of the regular height quadrature points found in traditional hexahedral elements. With both methods, the discretization of the domain is transposed into a two-stage process that consists of creating a three-dimensional lattice grid of regular hexahedron and recursively subdividing cells cut by the boundaries.

Validation tests performed on simple shapes have shown that the Finite Cell method produces very accurate solutions with convergence rates comparable to a standard FE method using four nodes tetrahedral elements. However, the time required for the assembly of the tangent stiffness matrix rapidly increases with the number of cell subdivisions. On the other hand, the same tests on our proposed WC method has shown that it can produce a solution very close to the FC method, without the need for additional integration points. The WC method also produced positive results when tested on some more complex scenarios including experiments with porcine and the human livers. But the tests also highlighted

some limitations that will need to be addressed. We saw for example that the WC method is not appropriate for simulations involving a Neo-Hookean material. When very large penalty forces are imposed, some of the cut cells are often reaching a degenerate state, which in turns results in a zero or negative jacobian of the deformation tensor.

Nevertheless, we successfully completed the implementation of a complete non-rigid registration pipeline for our surgical workflow using the well-known Iterative closest point (ICP) method and the proposed WC method as the backbone of the biomedical model. The performance of the pipeline has been thoroughly evaluated using both in-vivo porcine and ex-vivo human livers. A comparison of our methods and some of the emerging artificial intelligence and machine learning techniques have also been documented.

Overall, the meshless methods considered in this thesis and the immersed boundary WC method that we have proposed have shown interesting and promising results over simple shapes and even more realistic simulation scenarios. But we saw that the use of an arbitrary cloud of nodes in the meshless approach introduces more complex shape functions which in turn complicate the configuration process. The real impact of our approximation of the volumetric integration based on weighted integration points is also still unclear and will require further investigation.

Future work

Throughout this work, we have treated the meshless methods and IB methods as two separated entities. It would be interesting to investigate the use of an IB integration paradigm paired with a meshless shape function to further enforce the representation of the discontinuities introduced by the boundaries of an immersed object. Also, all the numerical methods considered in this work (standard FE model, the meshless methods or an IB approach) require that large displacement impositions be broken into many smaller increments. This was a direct consequence of the nonlinearity characteristic of the system of equations used by our biomechanical model. Further research will be required to investigate methods that allow bigger increments, in particular for non-rigid registrations which involves the reconstruction of highly deformed shapes during the intervention. Since coarse grids are able to converge with much higher imposed displacements, we believe that some hierarchical schemes would be appropriate here as a good starting point of the investigation. For instance, a coarse grid could be first solved as an approximate solution to an embedded finer grid. This approximation would then be used as the initial state of the NR method. Such hierarchical schemes can be

naturally implemented on a lattice geometry and would therefore constitute an advantage that our propose WC method could exploit. Similarly, as we saw in our brief look at neural networks, machine learning techniques, such as the U-Net architecture, could be used in this case to quickly produce an approximate first solution for the NR method, hence allowing it to quickly converge to an accurate final solution.

Finally, throughout our experiments, we noticed that using partial knowledge of the liver's current shape has a major impact on the overall registration process. A better approximation could be generated if additional information and knowledge about the liver and its surrounding structures would be used. For example, the sketching of an approximate atlas of these structures could be included in the pre-operative data. During surgery, this atlas could be stochastically refined through real-time acquired observation and inserted in the model as additional boundary conditions.

Chapter

7

BRIEF SUMMARY IN FRENCH

7.1 Introduction

La simulation numérique de divers phénomènes physiques est maintenant largement ancrée dans notre société. Dans cette thèse, nous nous intéressons à une application très spécifique de la simulation: la *simulation des tissus mous pour la réalité augmentée en chirurgie du foie*. Cette recherche a été réalisée dans le cadre d'un projet européen "European Innovative Training Network (ITN)" nommé *High Performance Soft Tissue Navigation (HiPerNav)* et financée par une action Marie Skłodowska-Curie du programme européen pour la recherche et le développement *Horizon 2020 (H2020)*.

Traditionnellement, les chirurgiens établissent le plan de l'opération à partir d'images médicales prises avant l'opération. Ces images sont généralement générées à partir d'appareil de tomodensitométrie (*computerize tomography (CT) scan*) ou bien d'appareil à résonance magnétique, et permettre d'assembler la séquence d'images en une reconstruction trois dimensionnelle des structures anatomiques du patient. Durant l'opération, les chirurgiens utilisent cette reconstruction préopératoire pour suivre le plan initial et guider l'intervention en cours. Or, la forme du foie durant l'opération est généralement très différente de sa forme initiale. En effet, le patient n'a pas la même position que lors de la prise d'images. De plus, certaines chirurgies dites *minimalement invasives* ou *laparoscopique* sont effectuées en insufflant du dioxyde de carbone dans la cavité de l'abdomen (cavité péritonéale), créant ainsi un espace de travail. L'opération est alors effectuée à partir d'instruments insérés

dans la cavité à l'aide de petites incisions faites au patient. Un laparoscope est inséré dans l'une de ces incisions et projette en continu des images prises à l'intérieur de la cavité, guidant ainsi le chirurgien dans ses manoeuvres. L'insufflation de gaz à l'intérieur de l'abdomen crée un pneumopéritoine artificiel et déforme considérablement le foie du patient. Ayant seulement accès à un plan d'opération fait avant l'opération, le chirurgien doit alors mentalement faire le lien entre les images reconstruites initialement, et les images observées durant l'opération. Dans le but d'assister le chirurgien dans cette démarche, nous nous intéressons à la mise en correspondance automatique d'images prises avant et pendant l'opération, un processus appelé *recalage*. Pour ce faire, un modèle biomécanique virtuel du foie est construit à partir de l'image 3D initiale et permet de lier ensemble les données préopératoires spécifiques au patient et les données acquises en temps réel pendant la chirurgie. Le modèle est alors déformé et une représentation visuelle de celui-ci est superposée aux images de réalité augmentée montrées à l'équipe médicale. La figure 7.1 illustre les étapes typiques réalisées dans une application de recalage non rigide.

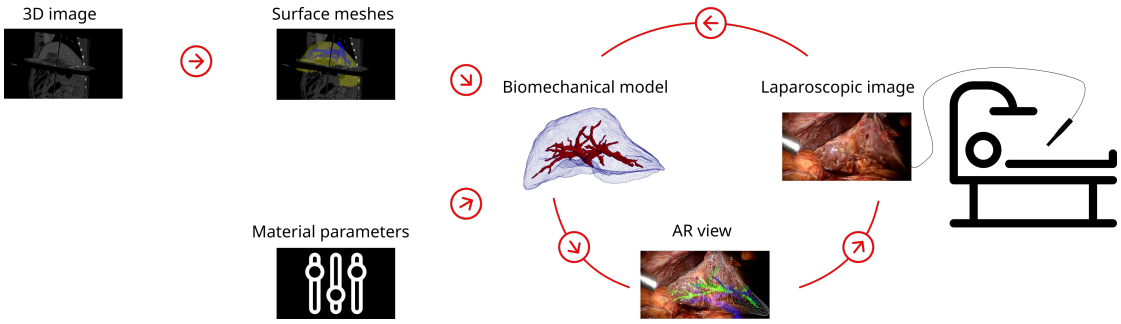


Figure 7.1: Étapes typiques pour un recalage non rigide en chirurgie laparoscopique.

Le modèle biomécanique peut être vu comme une boîte noire ayant en entrée la reconstruction surfacique initiale du foie et ses paramètres d'élasticité. Pour cette raison, le modèle est souvent appelé un modèle spécifique au patient. Lorsque certaines informations partielles du foie sont données au modèle pendant l'opération, ce dernier propose une estimation de l'état complet déformé. La méthode des éléments finis a été l'approche numérique de références pour la modélisation rigoureuse de divers organes. Cependant, cette méthode nécessite un processus de discrétisation où les frontières de l'organe sont représentées géométriquement. Pour la simulation du foie, ce processus devient rapidement fastidieux puisque ses frontières contiennent fréquemment des bords abrupts. C'est encore pire lorsque les structures internes sont également à modéliser. À cet égard, nous avons dédié ce travail de recherche à l'étude des méthodes numériques alternatives capables de modéliser un foie spécifique au patient tout en suivant des directives de recherche

clés établies pour une application de réalité augmentée à l'intérieur d'une salle d'opération: i) la précision de la solution, ii) la vitesse à laquelle cette solution est obtenue, iii) la stabilité du modèle, et iv) sa simplicité de mise en œuvre. Ainsi, la première partie de cette thèse est consacrée aux *méthodes sans maillage* où la discrétisation du domaine simulé est effectuée en remplissant simplement le volume à l'aide de voisinages de points, souvent appelés particules. Deux approches basées sur les méthodes sans maillage sont proposées. Pour la première approche, les termes d'approximation et d'intégration numérique utilisés dans l'équation discrète du problème d'hyperélasticité sont complètement sans maillage et se basent uniquement sur l'influence d'une particule par rapport à son voisinage. La seconde approche utilise également une approximation numérique basée sur des particules, mais où cette fois l'intégration numérique est construite à l'aide d'un maillage d'éléments placé en arrière-plan. La deuxième partie de cette thèse est dédiée à l'étude des *méthodes aux frontières immergées* où le concept de domaines fictifs est appliqué. Encore une fois, nous présentons deux implémentations. La première, la méthode des *cellules finies*, ajoute récursivement des points d'intégration près de l'interface de coupe à l'intérieur des éléments. La seconde, la méthode des *cellules pondérées*, utilise quant-a-t-elle les points d'intégration ajoutés par la méthode des cellules finies pour simplement ajuster les poids des points de quadrature normalement utilisés dans les éléments hexaédriques réguliers. Enfin, la dernière partie de cette thèse conclut ce travail de recherche avec une évaluation des concepts associés au recalage non rigide entre les informations préopératoires et peropératoires. Une routine complète utilisant nos méthodes numériques est implémentée et validée dans un contexte de chirurgie réaliste utilisant des données cliniques réelles.

7.2 Méthodes sans maillage

La première partie de cette thèse est consacrée aux *méthodes sans maillage*. Ici, la discrétisation du domaine à simuler est effectuée simplement en remplissant le volume de points, souvent appelés particules. Similairement aux noeuds des éléments dans la méthode des éléments finis, ces particules forment l'ensemble des degrés de liberté (inconnues) du système à résoudre. Ainsi, là où les méthodes traditionnelles d'éléments finis nécessitent une discrétisation complexe, les méthodes sans maillage semblent très attrayantes.

Lorsque l'animation par ordinateur en était qu'à ses tout débuts, Reeves (1983) a introduit une particule comme étant un simple point pouvant bouger dans l'espace de simulation et mourant après un certain temps donné. En remplissant un objet

de particules, le chercheur arrive à produire visuellement différents phénomènes tels que du feu ou de la fumée. Ici, les particules n'interagissent pas entre elles et sont plutôt dirigées par un modèle stochastique. L'interaction entre particules sera plus tard introduite par Luciani et al. (1991) et Szeliski and Tonnesen (1992) en ajoutant des forces de répulsion et d'attraction basées sur des potentiels d'énergie. Bien que ces techniques ont été les pionnières du domaine de la discrétisation sans maillage, c'est les travaux de Desbrun and Gascuel (1996) qui a d'abord introduit une méthode appelée Smoothed Particle Hydrodynamics (SPH) (provenant des recherches de Lucy (1977) et Monaghan (1992) en simulations astrophysiques) pour des animations d'objets déformables pilotés entièrement par les forces. Ces travaux ont établi une importante étape dans le domaine des simulations déformables sans maillage pour les animations informatiques. Ce concept a été repris par la suite dans diverses applications d'animation par ordinateur en introduisant cette fois la théorie d'élasticité linéaire comme potentiel d'énergie entre particules [Müller, Keiser, et al. (2004); Solenthaler, Schläfli, and Pajarola (2007b); Becker, Ihmsen, and Teschner (2009a)].

L'introduction de méthodes sans maillage pour les applications de simulation chirurgicale est plus récente. Dans Horton et al. (2010), une méthode de simulation sans maillage est proposée pour modéliser certains comportements non linéaires de déformations de tissus mous en cadre opératoire. Bien qu'aucune mention n'ait été faite concernant les temps de calcul, tout indique qu'ils ont été les premiers à proposer une solution sans maillage assez rapide basée sur une approche de Galerkin. Depuis, leurs travaux ont été étendus à diverses applications chirurgicales [K. Miller et al. (2012); G. Y. Zhang et al. (2014); Dehghan et al. (2016); Dong et al. (2016); Wittek et al. (2016)].

7.2.1 Animation basée sur les points (Méthode PBA)

Il existe plusieurs approches pour résoudre la dynamique d'élasticité d'un objet déformable. La première que nous avons considérée est la méthode d'animation élastique basée sur les points introduite dans Müller, Keiser, et al. (2004). Tout comme la méthode des éléments finis, les méthodes sans maillage permettent d'estimer un déplacement à une position quelconque du domaine. Cependant, plutôt que d'utiliser les propriétés géométriques d'un élément pour interpoler ce déplacement, les méthodes sans maillage utilisent un nuage de points où chaque point, ou particule, comporte un support compact d'influence, communément appelé le domaine d'influence. L'approximation d'une valeur à une position du domaine est alors obtenue à l'aide d'une fonction d'approximation, dite *fonction de forme*, qui est

évaluée pour chaque particule voisine. Le support de la fonction de forme d'une particule est généralement bâti à partir d'une fonction de poids (souvent appelée la fonction "noyau", ou "fenêtre"). Cette fonction est monotone et décroit de façon proportionnelle à la distance d'influence d'une particule. Pour notre projet, nous avons choisi une formulation lagrangienne. Ainsi, le poids d'une position dans le support compact d'une particule restera constant tout au long de la simulation et sera précalculé une seule fois au départ de la simulation.

Au début de la simulation, nous déterminons pour chaque particule le support de celle-ci en choisissant l'une des trois méthodes de calcul d'une distance d'influence (équations 3.8 à 3.10) ainsi qu'une des cinq méthodes de calcul du noyau (équations 3.3 à 3.7). Une fois le support d'influence obtenu, nous gardons pour chaque particule un vecteur de particules voisines ainsi que le poids (l'influence) de chacune. Le support des particules servira à établir les fonctions de formes de ces dernières.

Dans ce projet de recherche, nous avons implémenté deux fonctions de formes. La première est basée sur l'approximation du déplacement d'une particule à l'aide de la méthode SPH [Desbrun and Gascuel (1996)]. L'approximation du *gradient* du déplacement est obtenue en dérivant simplement la fonction de forme par rapport aux coordonnées du système. La seconde approche que nous avons implémentée utilise plutôt directement l'approximation du gradient du déplacement d'une particule au sens des moindres carrées pondérées, fréquemment appelée *Moving Least Squares (MLS)* en anglais. Dans les deux cas, les fonctions de formes sont de degré 1, et donc ne peuvent approximer exactement que des champs linéaires.

D'une certaine manière, avec la méthode PBA, le support compact autour d'une particule pourrait être considéré comme un type d'élément particulier, et ses plus proches voisins en tant que nœuds de cet élément. Par conséquent, les termes de gauche et de droite du système discret à résoudre (équations 2.18 et 2.26) peuvent être utilisés respectivement de la même manière qu'avec les méthodes des éléments finis. Cependant, compte tenu de l'absence d'éléments isoparamétriques, nous devons définir une méthode d'intégration numérique sur le support d'une particule. Ici, nous avons utilisé la méthode proposée par Müller, Keiser, et al. (2004) où la densité d'une particule est déterminée à l'aide de la méthode d'approximation SPH. Le volume du support compact en est alors déduit, et l'intégration numérique revient à calculer la valeur d'une fonction à la position d'une particule et de la multiplier par ce volume.

L'approche PBA aboutit à une méthode qui repose entièrement sur les particules, tant pour l'approximation du champ de déplacement que pour l'intégration

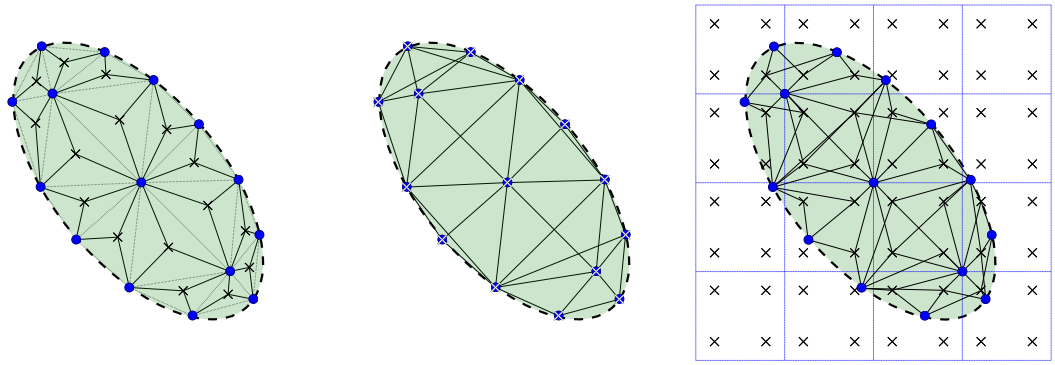
numérique. Cette caractéristique lui vaut une configuration très rapide et facile: il n'y a pas besoin de génération complexe d'un maillage d'éléments. Également, il est facile de voir qu'une extension aux applications de découpes interactives se fait naturellement en ajustant la fonction de poids pour supprimer l'influence d'une particule sur un voisin situé de l'autre côté de la coupure. Il existe cependant une forte limitation inhérente à cette méthode: son intégration numérique est basée sur une approximation grossière du volume occupé par une particule. Cette approximation a un impact important sur la précision de la solution, une observation faite durant nos essais expérimentaux dont les résultats sont décrits dans la section 3.11. En fait, la méthode PBA est souvent décrite comme étant une méthode visuellement plausible. Ses solutions semblent bonnes visuellement, mais pourraient être loin de la vraie solution au sens mécanique.

7.2.2 Approximation sans maillage, intégration avec maillage

La deuxième méthode sans maillage que nous avons implémentée vise à réduire l'erreur d'intégration et la faible cohérence de la fonction de forme inhérente à la méthode précédente. Elle est basée sur la méthode MTLED présentée dans Horton et al. (2010). Ici, cependant, nous proposons une extension de cette méthode visant l'utilisation d'un modèle d'élasticité linéaire couplant le principe corotationel. Contrairement à la méthode PBA, nous proposons d'intégrer les termes des équations (2.18) et (2.26) en utilisant une grille d'éléments hexaédriques réguliers placée en arrière-plan, donc superposée au domaine de simulation. Par conséquent, on pourrait soutenir que cette méthode n'est pas à 100% sans maillage. Cependant, l'utilisation de la grille d'arrière-plan élimine l'étape difficile associée à la création d'un maillage conforme qui est le principal inconvénient des méthodes des éléments finis. De plus, nous proposons d'utiliser des fonctions de formes adaptatives qui, de la même manière que les éléments d'ordre élevé, améliorent la cohérence de l'approximation ainsi que la convergence de la résolution du système. Pour ce faire, nous reprenons le concept de minimisation d'une erreur d'approximation au sens des moindres carrés. Cependant, ici nous minimisons l'approximation du champ de déplacement, et non son gradient.

Avec notre méthode MLAMBI, les particules ne représentent que les degrés de liberté, et les points d'intégration sont les points de quadrature gaussienne standard des éléments d'arrière-plan. Alors que ce schéma d'intégration est similaire à celui des éléments finis, l'approximation du champ de déplacement à chaque point de Gauss est formulée par rapport à ses particules voisines. Les différences entre la méthode PBA, notre méthode MLAMBI et la méthode des éléments finis sont

illustrées à la figure 7.2.



(a) Méthode des éléments finis (b) Méthode PBA (c) Méthode MLAMBI

Figure 7.2: Relations entre les points d'intégration (croix) et les noeuds DOF (cercles) où l'ovale vert représente l'objet simulé.

Un comparatif entre nos différentes implémentations sans maillage est décrit à la section 3.11. À partir de ces résultats, nous déterminons que, parmi les méthodes sans maillage étudiées, la méthode mixte utilisant des éléments d'arrière-plan pour l'intégration fournit la meilleure précision. Or, l'intégration sur les éléments coupés par la frontière immergée reste source d'erreurs numériques. Pour pallier ces erreurs, une méthode d'intégration plus adéquate doit être trouvée.

7.3 Méthode aux frontières immergées

La deuxième partie de cette thèse est dédiée à l'étude des *méthodes aux frontières immergées* ("Immersed boundary (IB)") où le concept de domaines fictifs est appliqué. Ici, nous revenons aux éléments finis où un maillage d'éléments isoparamétriques est utilisé autant pour l'approximation du déplacement que pour intégrer les termes des équations à résoudre. Par contre, contrairement aux méthodes traditionnelles des éléments finis, l'objet simulé est immergé dans une grille d'éléments isoparamétriques réguliers. C'est cette grille qui sera utilisée pour résoudre le problème initial. La difficulté de maillage d'une surface complexe en méthode des éléments finis est donc transposée à la prise en charge des éléments de la grille coupés par la surface de l'objet.

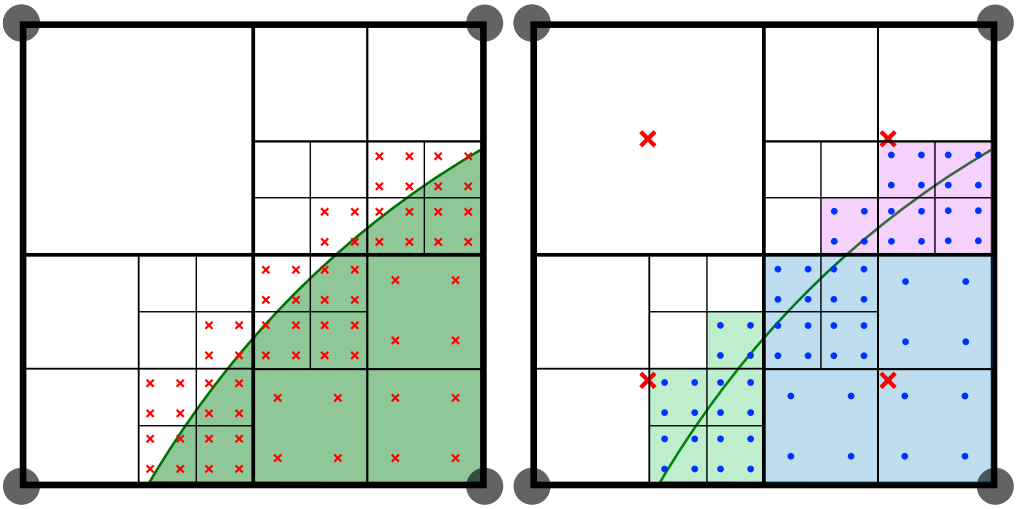
Depuis leur première introduction dans le travail de Peskin (1972) visant à résoudre l'écoulement vasculaire autour des valves cardiaques, les méthodes aux frontières

immergées ont évolué à une large gamme d'implémentations fluide-fluide, fluide-solide et solide-solide. Le principal défi associé aux méthodes IB se pose lorsque l'objet immergé ne rentre pas entièrement dans certains éléments de la grille de calcul. Ces éléments doivent donc être séparés en deux régions: la région associée à l'intérieur de l'objet simulé, et la région qui est à l'extérieur. La géométrie de ces deux sous-espaces et leurs matériaux respectifs doivent alors être pris en compte. Cela peut être fait en introduisant une contrainte de continuité sur l'interface entre les deux régions. De nombreuses approches ont été utilisées dans le passé pour résoudre cette contrainte. La plupart de celles-ci sont catégorisées en trois branches de méthodes: la méthode de pénalité [Bishop (2003); Ramière, Angot, and Belliard (2007)], la méthode des multiplicateurs de Lagrange [Burman and P. Hansbo (2010); Glowinski and Kuznetsov (2007)] et la méthode de Nitsche [Nitsche (1971); A. Hansbo and P. Hansbo (2002); Dolbow and Harari (2009); Burman and P. Hansbo (2012); Schillinger and Ruess (2015)]. Celles-ci peuvent également servir de moyen d'imposer des conditions aux limites essentielles lorsque la région externe est l'espace vide.

Une attention particulière doit également être donnée à l'intégration numérique sur les éléments coupés par la frontière. Comme documenté dans [Düster et al. (2008)], cela se fait généralement en représentant la géométrie à l'intérieur d'un élément par un ensemble de points de quadrature qui sont raffinés jusqu'à ce qu'une précision prescrite est obtenue [Parvizian, Düster, and Rank (2007); Ruess et al. (2012); Verhoosel et al. (2015); Elhaddad et al. (2018); Schillinger and Ruess (2015)]. Lorsque les deux régions de l'élément sont remplies avec des matériaux linéaires, et lorsque l'interface frontière peut être discrétisée en un ensemble d'éléments géométriques, les intégrales peuvent être converties en intégrales de surface en utilisant le théorème de la divergence [Mirtich (1996); Bishop (2003); T. P. Fries and Omerović (2016)].

L'utilisation d'une grille de calcul apporte de nombreux avantages. Déjà, elle ne comporte que des parallélépipèdes, qui sont bien formés par construction et convergent rapidement. Ensuite, elle permet naturellement une extension aux méthodes de décomposition de domaines. Tout comme avec les méthodes sans maillage, nous proposons deux implémentations. La première, la méthode des *cellules finies*, ajoute récursivement des points d'intégration près de l'interface de coupe à l'intérieur des éléments. La seconde, la méthode des *cellules pondérées*, utilise quant à elle les points d'intégration ajoutés par la méthode des cellules finies pour simplement ajuster les poids des points de quadrature normalement utilisés dans les éléments hexaédriques réguliers.

Pour notre type d'application, un respect strict des conditions de Dirichlet est



(a) Méthode des cellules finies (b) Méthode des cellules pondérées

Figure 7.3: Un élément quadrilatère est récursivement subdivisé en sous-cellules lorsque ce dernier intersecte l'interface frontière. L'intégration numérique est effectuée à l'aide d'une quadrature sur les points d'intégration (croix rouges). Pour la méthode des cellules pondérées, les poids de quadrature sont ajustés pour mieux représenter la portion de l'élément à l'intérieur de la frontière.

rarement nécessaire. En effet, la reconstruction des frontières du foie durant une chirurgie (généralement à partir d'images stéréo ou RGB-D) est souvent incomplète et grossièrement approximée. Par conséquent, nous avons opté pour une méthode de pénalité qui s'est avérée plus que suffisante pour nos besoins. D'ailleurs, cette méthode apporte une propriété utile qu'on ne retrouve pas avec les méthodes plus strictes (multiplicateurs de Lagrange et méthode de Nitsche): elle est plus robuste en cas d'imposition de déplacement non physique. Ce type d'imposition arrive fréquemment lors d'application de recalage non rigide où la reconstruction de la partie visible du foie est donnée sous forme de nuage de points souvent bruité. Lorsque nous utilisons ce nuage de points pour imposer les conditions de Dirichlet, certains déplacements imposés pourraient ne pas avoir de sens du point de vue physique et mécanique. Une imposition stricte via la méthode des multiplicateurs de Lagrange ou bien la méthode de Nitsche empêcherait probablement la simulation de converger.

Notre validation préliminaire (section 4.8) utilisant des formes simples a montré que la méthode des cellules finies produit des solutions très précises tout en conservant une meilleure convergence par rapport à une méthode standard des éléments finis utilisant des éléments tétraédriques. Cependant, le temps nécessaire

à l'assemblage des la matrice de rigidité augmente rapidement avec le nombre de subdivisions. Puisque nous cherchons à simuler des matériaux non linéaires, cet assemblage doit être réalisé à chaque itération de l'algorithme de Newton-Raphson. La validation préliminaire effectuée sur notre approche des cellules pondérées a montré quant à elle que cette méthode peut étonnamment produire des solutions très proches de celles obtenues avec les cellules finies, sans pour autant augmenter le temps de calcul. De plus, sa mise en oeuvre peut être facilement intégrée dans un logiciel d'éléments finis standard, héritant ainsi des diverses optimisations faites pour les éléments hexaédriques à 8 noeuds et 8 points d'intégration.

Notre analyse portant sur des formes plus complexes telles que la reconstruction complète de la surface de trois foies de porc et un foie humain a révélé des résultats intéressants ainsi que certains inconvénients qui n'avaient pas été observés lors de nos tests préliminaires. Contrairement à nos attentes, la méthode des cellules pondérées n'était pas suffisamment robuste pour les simulations impliquant un matériau Néo-Hookéen. Lorsque des forces de pénalité très importantes étaient imposées, certaines des cellules coupées par la surface atteignaient un état dégénéré, ce qui entraînait à son tour un jacobien nul ou négatif du tenseur de déformation.

Ensuite, l'imposition de grands déplacements a dû être divisée en plusieurs petits incréments sans quoi la solution se retrouvait trop loin de la configuration actuelle du modèle et l'algorithme de Newton-Raphson n'arrivait pas à converger. Ceci est un inconvénient non négligeable et des recherches supplémentaires devront être envisagées afin de permettre des incréments plus importants, en particulier pour une application de recalage entre une reconstruction hépatique préopératoire et sa forme largement déformée en peropératoire.

7.4 Application à la chirurgie

Au chapitre 5, une évaluation des concepts associés au recalage non rigide entre les informations préopératoires et peropératoires est effectuée. Une routine complète utilisant nos méthodes numériques est implémentée et validée dans un contexte de chirurgie réaliste utilisant des données cliniques réelles. D'abord, une reconstruction complète de trois foies de cochons est réalisée avant et après pneumopéritoine. Une reconstruction partielle de la partie visible des foies est effectuée à l'aide des images laparoscopiques après pneumopéritoine. Les résultats d'un recalage non rigide du modèle biomécanique vers la reconstruction partielle sont comparés à la reconstruction complète post-pneumopéritoine. L'expérience est

répétée en utilisant cette fois un foie humain ex vivo où deux déformations sont manuellement induites. Finalement, nous démontrons comment le modèle biomécanique peut être résous à l'aide d'un algorithme d'apprentissage d'un réseau de neurones. Ici, notre méthode des domaines fictifs est exploitée pour générer des milliers de champs de déformation d'un foie. Ces déformations sont utilisées pour la phase d'apprentissage du réseau. L'objectif est alors de prédire les déformations à l'intérieur d'un foie à partir de la déformation partielle de la surface visible.

Nos expériences ont montré que la reconstruction d'une simple partie de la surface du foie pendant la chirurgie ne suffit pas à permettre au modèle de prédire avec précision l'état de déformation à l'intérieur du foie. En fait, même en utilisant la surface complète du foie à l'état déformé, le recalage n'était toujours pas en mesure de prédire avec précision la position de certains marqueurs. Évidemment, c'est un inconvénient non négligeable puisque la précision de la solution est l'un des principaux objectifs de l'application. Pour améliorer la précision du processus de recalage, nous avons vu que des informations supplémentaires doivent être injectées dans notre modèle. Cela pourrait être fait de deux manières. Premièrement, le modèle pourrait être modifié de sorte que le comportement du foie du patient prenne bien en compte ses structures internes. Ainsi, bien que cela va à l'encontre des conclusions faites par Plantefève, Peterlik, and Cotin (2017), nous pensons que l'intégration de l'hétérogénéité du modèle pourrait considérablement améliorer la solution. Puisque les méthodes des éléments finis traditionnelles nécessitent un maillage fin autour de ces structures, notre approche des frontières immergées constituerait un net avantage. Deuxièmement, une petite quantité de connaissances pourrait être injectée à l'intérieur du modèle pendant l'opération. Par exemple, puisque des sondes ultra-son sont généralement accessibles dans la plupart des salles d'opération, certaines images provenant de telles sondes pourraient éventuellement être recalées et ajoutées en tant que contraintes supplémentaires. Ces deux recommandations nécessiteront une étude approfondie et devraient faire l'objet de recherches complémentaires.

Bibliography

Allard, J. et al. (2007). "SOFA-an open source framework for medical simulation". In: *Studies in Health Technology and Informatics*.

Arruda, Ellen M. and Mary C. Boyce (1993). "A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials". In: *Journal of the Mechanics and Physics of Solids*. DOI: 10.1016/0022-5096(93)90013-6.

Bajwa, Sukhminder Jit Singh and Ashish Kulshrestha (2016). *Anaesthesia for laparoscopic surgery: General vs regional anaesthesia*. DOI: 10.4103/0972-9941.169952.

Bathe, Klaus Jürgen and Lingbo Zhang (2017). "The finite element method with overlapping elements – A new paradigm for CAD driven simulations". In: *Computers and Structures*. DOI: 10.1016/j.compstruc.2016.10.020.

Becker, Markus, Markus Ihmsen, and Matthias Teschner (2009a). "Corotated sph for deformable solids". In: *Natural Phenomena*, pp. 27–34. DOI: 10.2312EG/DL/conf/EG2009/nph/027-034.

— (2009b). "Corotated SPH for Deformable Solids." In: *NPH*. Citeseer, pp. 27–34.

Belytschko, T et al. (1996). "Meshless Methods: An Overview and Recent Developments". In: *Computer Method in Applied Mechanics and Engineering* 139, pp. 3–47. DOI: 10.1016/S0045-7825(96)01078-X.

Belytschko, T., D. Organ, and Y. Krongauz (1995). "A coupled finite element-element-free Galerkin method". In: *Computational Mechanics*. DOI: 10.1007/BF00364080.

Belytschko, Ted, Yury Krongauz, et al. (1996). "Meshless methods: an overview and recent developments". In: *Computer methods in applied mechanics and engineering* 139.1-4, pp. 3–47.

Belytschko, Ted, Yun Yun Lu, and Lei Gu (1994). "Element-free Galerkin methods". In: *International Journal for Numerical Methods in Engineering* 37.2, pp. 229–256.

Benzley, Steven E et al. (1995). "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis". In: *Proceedings,*

4th international meshing roundtable. Vol. 17. Sandia National Laboratories Albuquerque, NM, pp. 179–191.

Bernhardt, Sylvain, Julien Abi-Nahed, and Rafeef Abugharbieh (2012). "Robust dense endoscopic stereo reconstruction for minimally invasive surgery". In: *International MICCAI workshop on medical computer vision*. Springer, pp. 254–262.

Besl, Paul J and Neil D McKay (1992). "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics, pp. 586–606.

Bishop, J (2003). "Rapid stress analysis of geometrically complex domains using implicit meshing". In: *Computational Mechanics* 30.5, pp. 460–478. DOI: 10.1007/s00466-003-0424-5.

Bray, Freddie et al. (2018). "Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries." In: *CA: a cancer journal for clinicians* 68 6, pp. 394–424.

Brown, Myron Z, Darius Burschka, and Gregory D Hager (2003). "Advances in computational stereo". In: *IEEE transactions on pattern analysis and machine intelligence* 25.8, pp. 993–1008.

Brunet, Jean-Nicolas, Vincent Magnoux, et al. (2019). "Corotated meshless implicit dynamics for deformable bodies". In: 27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. DOI: 10.24132/csrn.2019.2901.1.11.

Brunet, Jean-Nicolas, Andrea Mendizabal, et al. (2019). "Physics-Based Deep Neural Network for Augmented Reality During Liver Surgery". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*. Ed. by Dinggang Shen et al. Cham: Springer International Publishing, pp. 137–145.

Burkhart, Timothy A., David M. Andrews, and Cynthia E. Dunning (2013). *Finite element modeling mesh quality, energy balance and validation methods: A review with recommendations associated with the modeling of bone tissue*. DOI: 10.1016/j.jbiomech.2013.03.022.

Burman, Erik, Susanne Claus, et al. (2015). "CutFEM: Discretizing geometry and partial differential equations". In: *International Journal for Numerical Methods in Engineering*. DOI: 10.1002/nme.4823.

Burman, Erik and Peter Hansbo (Oct. 2010). "Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method". In: *Computer Methods in Applied Mechanics and Engineering* 199.41-44, pp. 2680–2686. DOI: 10.1016/J.CMA.2010.05.011.

— (Apr. 2012). "Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method". In: *Applied Numerical Mathematics* 62.4, pp. 328–341. DOI: 10.1016/J.APNUM.2011.01.008.

Cash, D M et al. (Nov. 2005). "Compensating for intraoperative soft-tissue deformations using incomplete surface data and finite elements". In: *IEEE Transactions on Medical Imaging* 24.11, pp. 1479–1491. DOI: 10.1109/TMI.2005.855434.

Chamberland, É, A Fortin, and M Fortin (2010). "Comparison of the performance of some finite element discretizations for large deformation elasticity problems". In: *Computers and Structures* 88.11, pp. 664–673. DOI: https://doi.org/10.1016/j.compstruc.2010.02.007.

Chen, Jiun Shyan et al. (1996). "Reproducing Kernel Particle Methods for large deformation analysis of non-linear structures". In: *Computer Methods in Applied Mechanics and Engineering*. DOI: 10.1016/S0045-7825(96)01083-3.

Clements, Logan W. et al. (2008). "Robust surface registration using salient anatomical features for image-guided liver surgery: Algorithm and validation". In: *Medical Physics*. DOI: 10.1118/1.2911920.

Courtecuisse, Hadrien et al. (2014). "Constraint-based simulation for non-rigid real-time registration". In: *Studies in Health Technology and Informatics*. DOI: 10.3233/978-1-61499-375-9-76.

Dehghan, Mohammad Reza et al. (2016). "A three-dimensional large deformation model for soft tissue using meshless method". In: *International Journal of Medical Robotics and Computer Assisted Surgery* 12.2. DOI: 10.1002/rcs.1682.

Desbrun, Mathieu and Marie-Paule Gascuel (1996). "Smoothed particles: A new paradigm for animating highly deformable bodies". In: *Computer Animation and Simulation'96*. Springer, pp. 61–76.

Dey, Tamal K and Joachim Giesen (2001). "Detecting undersampling in surface reconstruction". In: *Proceedings of the seventeenth annual symposium on Computational geometry*, pp. 257–263.

Dolbow, John and Isaac Harari (2009). "An efficient finite element method for embedded interface problems". In: *International Journal for Numerical Methods in Engineering* 78.2, pp. 229–252. DOI: 10.1002/nme.2486.

Dong, Yi et al. (2016). "A Nonlinear Viscoelastic Meshless Model for Soft Tissue Deformation". In: *2016 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 204–211. DOI: 10.1109/ICVRV.2016.42.

Duriez, C. et al. (2006). "New approaches to catheter navigation for interventional radiology simulation". In: *Computer Aided Surgery*. DOI: 10.1080/10929080601090623.

Düster, Alexander et al. (Aug. 2008). "The finite cell method for three-dimensional problems of solid mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 197.45-48, pp. 3768–3782. DOI: 10.1016/J.CMA.2008.02.036.

Elhaddad, Mohamed et al. (2018). "Multi-level hp-finite cell method for embedded interface problems with application in biomechanics". In: *International Journal*

for Numerical Methods in Biomedical Engineering 34.4, e2951. DOI: 10.1002/cnm.2951.

Felippa, Carlos A (2000). "A systematic approach to the element-independent corotational dynamics of finite elements". In: *Center for Aerospace Structures Document Number CU-CAS-00-03, College of Engineering, University of Colorado*.

Felippa, Carlos A. and Bjorn Haugen (2005). *A unified formulation of small-strain corotational finite elements: I. Theory*. DOI: 10.1016/j.cma.2004.07.035.

Fernández-Méndez, Sonia and Antonio Huerta (2004). "Imposing essential boundary conditions in mesh-free methods". In: *Computer Methods in Applied Mechanics and Engineering*. DOI: 10.1016/j.cma.2003.12.019.

Feudner, Elisabeth M et al. (2009). "Virtual reality training improves wet-lab performance of capsulorhexis: results of a randomized, controlled study". In: *Graefe's Archive for Clinical and Experimental Ophthalmology* 247.7, p. 955.

Fong, Zhi Ven and Kenneth K Tanabe (2014). "The clinical management of hepatocellular carcinoma in the United States, Europe, and Asia: A comprehensive and evidence-based comparison and review". In: *Cancer* 120.18, pp. 2824–2838. DOI: 10.1002/cncr.28730.

Fortin, André and André Garon (2018). *Les éléments finis : de la théorie à la pratique*.

Fries, Thomas Peter and Samir Omerović (2016). "Higher-order accurate integration of implicit geometries". In: *International Journal for Numerical Methods in Engineering*. DOI: 10.1002/nme.5121.

Fries, Thomas-Peter and Hermann G Matthies (2003). "Classification and overview of meshfree methods". In: *Department of Mathematics and Computer Science, Technical University of Braunschweig*.

Gilg, Stefan et al. (2017). "Mortality-related risk factors and long-term survival after 4460 liver resections in Sweden—a population-based study". In: *Langenbeck's archives of surgery* 402.1, pp. 105–113.

Glowinski, R. and Yu. Kuznetsov (Jan. 2007). "Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems". In: *Computer Methods in Applied Mechanics and Engineering* 196.8, pp. 1498–1506. DOI: 10.1016/J.CMA.2006.05.013.

Gosz, J. and W. K. Liu (1996). "Admissible approximations for essential boundary conditions in the reproducing kernel particle method". In: *Computational Mechanics*. DOI: 10.1007/bf02824850.

Griebel, M. and M. A. Schweitzer (2003). "A Particle-Partition of Unity Method Part V: Boundary Conditions". In: *Geometric Analysis and Nonlinear Partial Differential Equations*. DOI: 10.1007/978-3-642-55627-2_27.

Günther, Frank C. and Wing Kam Liu (1998). "Implementation of boundary conditions for meshless methods". In: *Computer Methods in Applied Mechanics and Engineering*. DOI: 10.1016/S0045-7825(98)00014-0.

Haller, Guy et al. (2009). "Rate of undesirable events at beginning of academic year: retrospective cohort study". In: *Bmj* 339, b3974.

Haluck, R S et al. (2001). "Are surgery training programs ready for virtual reality? A survey of program directors in general surgery." In: *Journal of the American College of Surgeons* 193, pp. 660–665.

Hansbo, Anita and Peter Hansbo (Nov. 2002). "An unfitted finite element method, based on Nitsche's method, for elliptic interface problems". In: *Computer Methods in Applied Mechanics and Engineering* 191.47-48, pp. 5537–5552. DOI: 10.1016/S0045-7825(02)00524-8.

Haouchine, Nazim, Jeremie Dequidt, et al. (2013). "Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery". In: *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*. DOI: 10.1109/ISMAR.2013.6671780.

Haouchine, Nazim, Frederick Roy, et al. (2016). "Using contours as boundary conditions for elastic registration during minimally invasive hepatic surgery". In: *IEEE International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IROS.2016.7759099.

Hartley, Richard and Andrew Zisserman (2004). *Multiple View Geometry in Computer Vision*. DOI: 10.1017/cbo9780511811685.

Horton, Ashley et al. (2010). "A meshless Total Lagrangian explicit dynamics algorithm for surgical simulation". In: *International Journal for Numerical Methods in Biomedical Engineering* 26.8, pp. 977–998. DOI: 10.1002/cnm.1374.

Huerta, Antonio and Sonia Fernández-Méndez (2000). "Enrichment and coupling of the finite element and meshless methods". In: *International Journal for Numerical Methods in Engineering*. DOI: 10.1002/1097-0207(20000820)48:11<1615::AID-NME883>3.0.CO;2-S.

Jeřábková, Lenka et al. (2010). "Volumetric modeling and interactive cutting of deformable bodies". In: *Progress in Biophysics and Molecular Biology*. DOI: 10.1016/j.pbiomolbio.2010.09.012.

Kikinis, Ron, Steve D Pieper, and Kirby G Vosburgh (2014). "3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support". In: *Intraoperative Imaging and Image-Guided Therapy*. Ed. by Ferenc A Jolesz. New York, NY: Springer New York, pp. 277–289. DOI: 10.1007/978-1-4614-7657-3_19.

Lancaster, Peter and Kes Salkauskas (1981). "Surfaces generated by moving least squares methods". In: *Mathematics of computation* 37.155, pp. 141–158.

Li, Fei-yan et al. (2018). "A Constitutive Model of Soft Tissue Deformation for Virtual Surgical Simulation: A Literature Review". In: *International Conference on Physics, Mathematics, Statistics Modelling and Simulation*. Pmsms.

Li, Hui et al. (2017). "Laparoscopic VS open hepatectomy for hepatolithiasis: An updated systematic review and meta-analysis". In: *World journal of gastroenterology*. DOI: 10.3748/wjg.v23.i43.7791.

Luciani, Annie et al. (1991). "An unified view of multitude behavior, flexibility, plasticity and fractures balls, bubbles and agglomerates". In: *Modeling in Computer Graphics*. Springer, pp. 55–74.

Lucy, L. B. (1977). "A numerical approach to the testing of the fission hypothesis". In: *The Astronomical Journal*. DOI: 10.1086/112164.

Luo, Suhuai, Xuechen Li, and Jiaming Li (2014). "Review on the Methods of Automatic Liver Segmentation from Abdominal Images". In: *Journal of Computer and Communications*. DOI: 10.4236/jcc.2014.22001.

Malukhin, Kostyantyn and Kornel Ehmann (Mar. 2018). "Mathematical Modeling and Virtual Reality Simulation of Surgical Tool Interactions With Soft Tissue: A Review and Prospective". In: *Journal of Engineering and Science in Medical Diagnostics and Therapy* 1.2, pp. 20802–20823.

Marchesseau, Stéphanie, Simon Chatelin, and Hervé Delingette (2017). "Non linear Biomechanical Model of the Liver". In: *Biomechanics of Living Organs*. Ed. by Yohan Payan and Jacques Ohayon. Elsevier. Chap. 10, p. 602.

Masuda, Takeshi and Naokazu Yokoya (1995). "A robust method for registration and segmentation of multiple range images". In: *Computer vision and image understanding* 61.3, pp. 295–307.

Mendizabal, Andrea et al. (2020). "Data-Driven Simulation for Augmented Surgery". In: *Developments and Novel Approaches in Biomechanics and Metamaterials*. Ed. by Bilen Emek Abali and Ivan Giorgio. Cham: Springer International Publishing, pp. 71–96. DOI: 10.1007/978-3-030-50464-9_5.

Miga, Michael I et al. (2003). "Intraoperative registration of the liver for image-guided surgery using laser range scanning and deformable models". In: *Medical Imaging 2003: Visualization, Image-Guided Procedures, and Display*. Ed. by Robert L Galloway Jr. Vol. 5029. International Society for Optics and Photonics. SPIE, pp. 350–359. DOI: 10.1117/12.480216.

Miller, K. et al. (2012). "Beyond finite elements: A comprehensive, patient-specific neurosurgical simulation utilizing a meshless method". In: *Journal of Biomechanics* 45.15, pp. 2698–2701. DOI: 10.1016/j.jbiomech.2012.07.031.

Miller, Karol and Jia Lu (2013). "On the prospect of patient-specific biomechanics without patient-specific properties of tissues". In: *Journal of the Mechanical Behavior of Biomedical Materials*. DOI: 10.1016/j.jmbbm.2013.01.013.

Mirtich, Brian (1996). "Fast and Accurate Computation of Polyhedral Mass Properties". In: *Journal of Graphics Tools* 1, pp. 31–50. DOI: 10.1080/10867651.1996.10487458.

Moireau, Philippe and Dominique Chapelle (2011). "Reduced-order Unscented Kalman Filtering with application to parameter identification in large-dimensional systems". In: *ESAIM - Control, Optimisation and Calculus of Variations*. DOI: 10.1051/cocv/2010006.

Monaghan, Joe J (1992). "Smoothed particle hydrodynamics". In: *Annual review of astronomy and astrophysics* 30, pp. 543–574.

Mooney, M. (1940). "A theory of large elastic deformation". In: *Journal of Applied Physics*. DOI: 10.1063/1.1712836.

Morooka, Ken'Ichi et al. (2008). "Real-time nonlinear FEM with neural network for simulating soft organ model deformation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. DOI: 10.1007/978-3-540-85990-1-89.

Mostt, Thomas and Christian Bucher (2005). "A Moving Least Squares weighting function for the Element-free Galerkin Method which almost fulfills essential boundary conditions". In: *Structural Engineering and Mechanics*. DOI: 10.12989/sem.2005.21.3.315.

Müller, Matthias, David Charypar, and Markus Gross (2003). "Particle-based fluid simulation for interactive applications". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, pp. 154–159.

Müller, Matthias, Julie Dorsey, et al. (2002). "Stable real-time deformations". In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '02* 166.2, p. 49. DOI: 10.1145/545261.545269.

Müller, Matthias, Bruno Heidelberger, et al. (2005). "Meshless deformations based on shape matching". In: *ACM transactions on graphics (TOG)* 24.3, pp. 471–478.

Müller, Matthias, Richard Keiser, et al. (2004). "Point based animation of elastic, plastic and melting objects". In: *Proceedings of the 2004 ACM SIGGRAPH/-Eurographics symposium on Computer animation*, pp. 141–151. DOI: 10.1145/1028523.1028542.

Nesme, Matthieu, Paul G. Kry, et al. (2009). "Preserving topology and elasticity for embedded deformable models". In: *ACM Transactions on Graphics*. DOI: 10.1145/1531326.1531358.

Nesme, Matthieu, Yohan Payan, and François Faure (2005). "Efficient, physically plausible finite elements". In: *Eurographics*.

— (2006). "Animating Shapes at Arbitrary Resolution with Non-Uniform Stiffness". In: *VRIPHYS*. Madrid, Spain.

Newman, Timothy S and Hong Yi (2006). "A survey of the marching cubes algorithm". In: *Computers & Graphics* 30.5, pp. 854–879.

Nikolaev, Sergei, Igor Peterlik, and Stéphane Cotin (Sept. 2018). "Stochastic Correction of Boundary Conditions during Liver Surgery". In: *CVCS 2018 - 9th Colour and Visual Computing Symposium 2018*. NTNU: Norwegian University of Science and Technology. Gjovik, Norway.

Nitsche, J. (1971). "Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind". In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*. DOI: 10.1007/BF02995904.

Nykonenko, Andriy, Petr Vávra, and Pavel Zonča (2017). *Anatomic peculiarities of pig and human liver*. DOI: 10.6002/ect.2016.0099.

Ogden, R W (2013). *Non-Linear Elastic Deformations*. Dover Civil and Mechanical Engineering. Dover Publications.

— (1973). "LARGE DEFORMATION ISOTROPIC ELASTICITY - ON THE CORRELATION OF THEORY AND EXPERIMENT FOR INCOMPRESSIBLE RUBBERLIKE SOLIDS." In: *Rubber Chemistry and Technology*. DOI: 10.5254/1.3542910.

Oliveira, Francisco P M and João Manuel R S Tavares (2014). "Medical image registration: a review". In: *Computer Methods in Biomechanics and Biomedical Engineering* 17.2, pp. 73–93. DOI: 10.1080/10255842.2012.670855.

P.-L., Manteaux et al. (June 2016). "Adaptive Physically Based Models in Computer Graphics". In: *Computer Graphics Forum* 36.6, pp. 312–337. DOI: 10.1111/cgf.12941.

Parvizian, Jamshid, Alexander Düster, and Ernst Rank (2007). "Finite cell method". In: *Computational Mechanics* 41.1, pp. 121–133. DOI: 10.1007/s00466-007-0173-y.

Paulus, Christoph et al. (Sept. 2017). "An Immersed Boundary Method for Detail-Preserving Soft Tissue Simulation from Medical Images". In: *Computational Biomechanics for Medicine*. Quebec.

Paulus, Christoph J. et al. (2017). "Handling topological changes during elastic registration: Application to augmented reality in laparoscopic surgery". In: *International Journal of Computer Assisted Radiology and Surgery* 12.3, pp. 461–470. DOI: 10.1007/s11548-016-1502-4.

Peskin, C S (1972). "Flow patterns around heart valves: A digital computer method for solving the equations of motion". In: *Flow Patterns Around Heart Valves: A Digital Computer Method for Solving the Equations of Motion*.

Petit, Antoine, Vincenzo Lippiello, and Bruno Siciliano (2015). "Real-time tracking of 3D elastic objects with an RGB-D sensor". In: *IEEE International Conference on Intelligent Robots and Systems*. DOI: 10.1109/IROS.2015.7353928.

Pfeiffer, Micha et al. (2019). "Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks". In: *International Journal of Computer Assisted Radiology and Surgery* 14.7, pp. 1147–1155. DOI: 10.1007/s11548-019-01965-7.

Plantefève, Rosalie, Igor Peterlik, and Stéphane Cotin (2017). "Intraoperative Biomechanical Registration of the Liver: Does the Heterogeneity of the Liver Matter?" In: *IRBM*.

Plantefève, Rosalie, Igor Peterlik, Nazim Haouchine, et al. (2016). "Patient-Specific Biomechanical Modeling for Guidance During Minimally-Invasive Hepatic Surgery". In: *Annals of Biomedical Engineering* 44.1, pp. 139–153. DOI: 10.1007/s10439-015-1419-z.

Prenter, F. de et al. (Apr. 2017). "Condition number analysis and preconditioning of the finite cell method". In: *Computer Methods in Applied Mechanics and Engineering* 316, pp. 297–327. DOI: 10.1016/J.CMA.2016.07.006.

Press, William H et al. (1992). *Numerical Recipes in C: The Art of Scientific Computing (; Cambridge*.

Ramière, Isabelle, Philippe Angot, and Michel Belliard (Jan. 2007). "A fictitious domain approach with spread interface for elliptic problems with general boundary conditions". In: *Computer Methods in Applied Mechanics and Engineering* 196.4-6, pp. 766–781. DOI: 10.1016/J.CMA.2006.05.012.

Raut, Padmakar (2012). "Impact of mesh quality parameters on elements such as beam, shell and 3D solid in structural analysis". In: *International Journal of Engineering Research and Applications*.

Reeves, William T (1983). "Particle systems—a technique for modeling a class of fuzzy objects". In: *ACM Transactions on Graphics (TOG)* 2.2, pp. 91–108.

Rivlin, RS (1948). "Large elastic deformations of isotropic materials IV. further developments of the general theory". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 241.835, pp. 379–397. DOI: 10.1098/rsta.1948.0024.

Röhl, Sebastian et al. (2012). "Dense GPU-enhanced surface reconstruction from stereo endoscopic images for intraoperative registration". In: *Medical physics* 39.3, pp. 1632–1645.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Ruess, Martin et al. (2012). "The finite cell method for bone simulations: verification and validation". In: *Biomechanics and Modeling in Mechanobiology* 11.3, pp. 425–437. DOI: 10.1007/s10237-011-0322-2.

Scharstein, Daniel and Richard Szeliski (2002). "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *International journal of computer vision* 47.1-3, pp. 7–42.

Schillinger, Dominik and Martin Ruess (2015). "The Finite Cell Method: A Review in the Context of Higher-Order Structural Analysis of CAD and Image-Based Geometric Models". In: *Archives of Computational Methods in Engineering* 22.3, pp. 391–455. DOI: 10.1007/s11831-014-9115-y.

Selvander, Madeleine and Peter Åsman (2012). "Virtual reality cataract surgery training: learning curves and concurrent validity". In: *Acta ophthalmologica* 90.5, pp. 412–417.

Seymour, Neal E et al. (2002). "Virtual reality training improves operating room performance: results of a randomized, double-blinded study". In: *Annals of surgery* 236.4, pp. 458–464.

Shepherd, Jason F and Chris R Johnson (2008). "Hexahedral mesh generation constraints". In: *Engineering with Computers* 24.3, pp. 195–213. DOI: 10.1007/s00366-008-0091-4.

Sokolov, Dmitry et al. (2016). "Hexahedral-dominant meshing". In: *ACM Transactions on Graphics*. DOI: 10.1145/2930662.

Solenthaler, Barbara, Jürg Schläfli, and Renato Pajarola (2007a). "A unified particle model for fluid–solid interactions". In: *Computer Animation and Virtual Worlds* 18.1, pp. 69–82.

— (2007b). "A unified particle model for fluid-solid interactions". In: *Computer Animation and Virtual Worlds*. Vol. 18. 1, pp. 69–82. DOI: 10.1002/cav.162.

Sotiras, A, C Davatzikos, and N Paragios (2013). "Deformable Medical Image Registration: A Survey". In: *IEEE Transactions on Medical Imaging* 32.7, pp. 1153–1190. DOI: 10.1109/TMI.2013.2265603.

Speidel, S et al. (2011). "Intraoperative surface reconstruction and biomechanical modeling for soft tissue registration". In: *Proc. Joint Workshop on New Technologies for Computer/Robot Assisted Surgery*.

Stavrev, Atanas et al. (2016). "Geometrically accurate, efficient, and flexible quadrature techniques for the tetrahedral finite cell method". In: *Computer Methods in Applied Mechanics and Engineering*. DOI: 10.1016/j.cma.2016.07.041.

Stoyanov, Danail et al. (2010). "Real-time stereo reconstruction in robotically assisted minimally invasive surgery". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 275–282.

Szeliski, Richard and David Tonnesen (1992). *Surface modeling with oriented particle systems*. Vol. 26. 2. ACM.

Tadepalli, Srinivas C., Ahmet Erdemir, and Peter R. Cavanagh (2011). "Comparison of hexahedral and tetrahedral elements in finite element analysis of the

foot and footwear". In: *Journal of Biomechanics*. DOI: 10.1016/j.jbiomech.2011.05.006.

Teatini, Andrea et al. (2019). "The effect of intraoperative imaging on surgical navigation for laparoscopic liver resection surgery". In: *Scientific Reports*. DOI: 10.1038/s41598-019-54915-3.

Tonutti, Michele, Gauthier Gras, and Guang-Zhong Yang (2017). "A machine learning approach for real-time modelling of tissue deformation in image-guided neurosurgery". In: *Artificial intelligence in medicine* 80, pp. 39–47.

Torres, Rosell, Jose M Espadero, et al. (2014). "Interactive Deformation of Heterogeneous Volume Data". In: *Biomedical Simulation*. Ed. by Fernando Bello and Stéphane Cotin. Cham: Springer International Publishing, pp. 131–140.

Torres, Rosell, Alejandro Rodríguez, et al. (Nov. 2016). "High-resolution Interaction with Corotational Coarsening Models". In: *ACM Trans. Graph.* 35.6, 211:1–211:11. DOI: 10.1145/2980179.2982414.

Udupa, Jayaram K and Gabor T Herman (1999). *3D imaging in medicine*. CRC press.

Varduhn, Vasco et al. (2016). "The tetrahedral finite cell method". In: *International Journal for Numerical Methods in Engineering*. DOI: 10.1002/nme.5207.

Verhoosel, C.V. et al. (Feb. 2015). "Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone". In: *Computer Methods in Applied Mechanics and Engineering* 284, pp. 138–164. DOI: 10.1016/J.CMA.2014.07.009.

Viganò, Luca et al. (2009). *Laparoscopic liver resection: A systematic review*. DOI: 10.1007/s00534-009-0120-8.

Wang, Congcong et al. (2018). "Liver surface reconstruction for image guided surgery". In: *Medical Imaging 2018: Image-Guided Procedures, Robotic Interventions, and Modeling*. Vol. 10576. International Society for Optics and Photonics, 105762H.

Wittek, Adam et al. (2016). "From Finite Element Meshes to Clouds of Points: A Review of Methods for Generation of Computational Biomechanics Models for Patient-Specific Applications". In: *Annals of Biomedical Engineering*. DOI: 10.1007/s10439-015-1469-2.

Wriggers, Peter (2008). *Nonlinear finite element methods*. DOI: 10.1007/978-3-540-71001-1.

Wu, Xunlei et al. (2001). "Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes". In: *Computer Graphics Forum*. DOI: 10.1111/1467-8659.00527.

Xu, Fei et al. (2016). "The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries". In: *Computers and Fluids*. DOI: 10.1016/j.compfluid.2015.08.027.

Yushkevich, Paul A et al. (2006). "User-Guided 3D Active Contour Segmentation of Anatomical Structures: Significantly Improved Efficiency and Reliability". In: *Neuroimage* 31.3, pp. 1116–1128.

Zhang, G. Y. et al. (2014). "A three-dimensional nonlinear meshfree algorithm for simulating mechanical responses of soft tissue". In: *Engineering Analysis with Boundary Elements* 42, pp. 60–66. DOI: 10.1016/j.enganabound.2013.08.014.

Zhang, Jinao, Yongmin Zhong, and Chengfan Gu (2017). "Deformable models for surgical simulation: a survey". In: *IEEE reviews in biomedical engineering* 11, pp. 143–164.

Zhang, Jinao, Yongmin Zhong, Chengfan Gu, and Peter Coloe (2017). "Deformable Models for Surgical Simulation: A Survey". In: *IEEE Reviews in Biomedical Engineering* PP.99, p. 1. DOI: 10.1109/RBME.2017.2773521.

Zhang, Zhengyou (1994). "Iterative point matching for registration of free-form curves and surfaces". In: *International journal of computer vision* 13.2, pp. 119–152.

— (2000). "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11, pp. 1330–1334.

Zhu, T. and S. N. Atluri (1998). "A modified collocation method and a penalty formulation for enforcing the essential boundary conditions in the element free Galerkin method". In: *Computational Mechanics*. DOI: 10.1007/s004660050296.

List of publications

Publication 1 Brunet J.N., Magnoux V., Ozell B., Cotin S. *"Corotated meshless implicit dynamics for deformable bodies"* **27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision** 2019

Publication 2 Brunet J.N., Mendizabal A., Petit A., Golse N., Vibert E., Cotin S. *"Physics-Based Deep Neural Network for Augmented Reality During Liver Surgery."* **Medical Image Computing and Computer Assisted Intervention (MICCAI)** 2019

Publication 3 Mendizabal A., Tagliabue E., Brunet J.N., Dall'alba D., Fiorini P., Cotin S. *"Physics-based Deep Neural Network for Real-Time Lesion Tracking in Ultrasound-guided Breast Biopsy"* **Computational Biomechanics for Medicine XIV** 2019

Publication 4 Teatini A., Brunet J.N., Nikolaev S., Wang C., Edwin B., Cotin S., Elle O.J. *"Use of stereo-laparoscopic liver surface reconstruction to compensate for pneumoperitoneum deformation through biomechanical modeling."* **Virtual Physiological Human (VPH)** 2020